HEC MONTRÉAL

École affiliée à l'Université de Montréal

Primal algorithms for degenerate linear and network flow problems

par

Jean Bertrand Gauthier

Thèse présentée en vue de l'obtention du grade de Ph. D. en administration
(option Méthodes quantitatives)

Mai 2016

HEC MONTRÉAL

École affiliée à l'Université de Montréal

cette thèse intitulée :

Primal algorithms for degenerate linear and network flow problems

présentée par :

Jean Bertrand Gauthier

a été évaluée par un jury composé des personnes suivantes :

Gilbert Laporte
HEC Montréal
Président-rapporteur

Jacques Desrosiers
HEC Montréal
Directeur de recherche

Claudio Contardo
Université du Québec à Montréal
Membre du jury

Stefan Irnich
Johannes Gutenberg University Mainz
Examinateur externe

Jacques Robert
HEC Montréal
Représentant du directeur de HEC Montréal

RÉSUMÉ

Alors que l'algorithme du simplexe primal (PS) fonctionne remarquablement bien pour une grande quantité de scénarios, ce dernier souffre du phénomène de dégénérescence à de nombreux égards. Parmi ces faiblesses se retrouvent les opérations de calculs effectuées sur les variables entrantes à pas nuls. L'algorithme du simplexe primal amélioré (*Improved Primal Simplex*) (IPS) vise à combattre la dégénérescence pendant la résolution.

La revue de littérature prend du recul par rapport à IPS et cherche à mieux comprendre le contenu du problème de tarification. Celle-ci est séparée en deux parties chacune ayant donné le jour à un article à part entière. La première partie résume les connaissances autour de l'algorithme du *Minimum Mean Cycle-Canceling* (MMCC). Dans la foulée de cet effort de synthèse apparaît par ailleurs un nouveau résultat de complexité fortement polynomiale dont la construction est incidemment également utile en pratique. Quant à la deuxième partie, nous espérons éliminer toute confusion qui puisse survenir entre IPS et un autre outil développé pour combattre la dégénérescence sur les problèmes de partitionnement : Agrégation de Contraintes Dynamique (*Dynamic Constraint Aggregation*) (DCA).

Le cœur de la thèse construit sur ces acquis et s'articule autour de trois axes. Le premier axe généralise des résultats établis pour les problèmes de flots à la programmation linéaire, et introduit des conditions d'optimalité nécessaires et suffisantes pour ces derniers. Nous avons constaté qu'il est possible d'établir de nombreuses connexions entre les principes régissant MMCC et ceux d'IPS dans le cadre des problèmes de réseaux. C'est ainsi que le deuxième axe accompagne la naissance d'un algorithme fortement polynomial de Contraction-Expansion à une étude computationnelle. Le troisième axe place IPS, PS, DCA et MMCC dans un unique cadre permettant en outre d'autres variantes algorithmiques.

Mots-clé : Programmation linéaire, Dégénérescence, Algorithme primal, Décomposition Dantzig-Wolfe, Conditions d'optimalité nécessaires, Problème réduit en lignes, Combinaison de variables entrantes, Base du sous-espace vectoriel, Problèmes de flots, *Minimum mean cycle-canceling*, *Cancel-and-Tighten*, Contraction-Expansion, Analyse de complexité.

Méthodes de recherche : La programmation linéaire renferme des problèmes mathématiques bien définis. Ce qui préoccupe les chercheurs est plutôt la résolution desdits problèmes. La revue de littérature rassemble l'état des connaissances autour du phénomène de dégénérescence lequel pose des embûches à l'une des plus populaires méthodes de résolution, à savoir le simplexe. Nous explorons des algorithmes alternatifs de façon théorique, et menons des expériences numériques qui implémentent les idées avancées dans cette thèse.

ABSTRACT

While the traditional primal simplex algorithm (PS) does work wonders in several scenarios, it suffers from degeneracy in more ways than one. The computational effort wasted on pivoting entering variables with null step sizes is one such concern. The Improved Primal Simplex algorithm (IPS) aims to fight degeneracy during the solving process.

The literature review takes a step back from IPS and aims to better understand the content of the pricing problem. It is separated in two parts both of which finding home in independent papers. In the first part, a survey examines the Minimum Mean Cycle-Canceling algorithm (MMCC) and further contributes a new strongly polynomial complexity result whose construction incidentally also brings practical improvements. In the second part, we hope to dissolve the confusion that often arises between IPS and another tool designed to fight degeneracy in set partitioning problems: Dynamic Constraint Aggregation (DCA). The paper contains three degeneracy fighting tools: IPS, DCA and the Positive Edge rule (PE), devised for different applications.

The dissertation's core builds upon this knowledge on three axes. The first axis generalizes results from network flows to linear programming, and introduces necessary and sufficient optimality conditions for these problems. As it turns out, the principles behind MMCC have several connections with those of IPS when applied to network problems. Such is the content of the second axis from which emanates a so-called Contraction-Expansion strongly polynomial algorithm along with a computational study. The third axis generalizes IPS, PS, DCA and MMCC under a unique framework while allowing other variants.

Keywords: Linear programming, Degeneracy, Primal algorithm, Dantzig-Wolfe decomposition, Necessary optimality conditions, Row-reduced problem, Combination of entering variables, Vector subspace basis, Network flows, Minimum mean cycle-canceling, Cancel-and-Tighten, Contraction-Expansion, Complexity analysis.

Research methods: Linear programming houses well defined mathematical problems. The resolution of these problems is typically what interests researchers. The literature review brings together degeneracy related topics causing havoc in the simplex algorithm, one of the most prominent resolution method used today. We explore alternative algorithms on theoretical grounds, and conduct computational experiments implementing ideas put forward in this dissertation.

CONTENTS

LIST OF PAPERS

Part of the dissertation

1. Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. About the minimum mean cycle-canceling algorithm. *Discrete Applied Mathematics*, 196:115–134, 2015a. doi:10.1016/j.dam.2014.07.005. Advances in Combinatorial Optimization.
2. Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. Tools for primal degenerate linear programs: IPS, DCA, and PE. *EURO Journal on Transportation and Logistics*, pages 1–44, 2015b. doi:10.1007/s13676-015-0077-5.
3. Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. Decomposition theorems for linear programs. *Operations Research Letters*, 42(8):553–557, December 2014a. doi:10.1016/j.orl.2014.10.001.
4. Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. A strongly polynomial Contraction-Expansion algorithm for network flow problems. Les Cahiers du GERAD G-2016-18, HEC Montréal, Montreal, QC, Canada, March 2016. Submitted to *Computers & Operations Research*.
5. Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. Vector space decomposition for linear programs. Les Cahiers du GERAD G-2015-26, HEC Montréal, Montreal, QC, Canada, March 2015a. (Preliminary version).

Not part of the dissertation

6. Jacques Desrosiers, Jean Bertrand Gauthier, and Marco E. Lübbecke. Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, 236(2):453–460, 2014. doi:10.1016/j.ejor.2013.12.016.
7. Jean Bertrand Gauthier and Antoine Legrain. Operating room management under uncertainty. *Constraints*, pages 1–20, 2015. doi:10.1007/s10601-015-9236-4.

Awards and recognitions

- Paper 1 was awarded the *prix Esdras-Minville, 2014* by *la direction du programme de doctorat de HEC Montréal*.
- Paper 6 was awarded the *prix du meilleur mémoire HEC Montréal, 2011-12*.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

- **CE** contraction-expansion algorithm
- **CMCF** capacitated minimum cost flow problem
- **CT** cancel-and-tighten strategy
- **DCA** dynamic constraint aggregation
- **DVS** dual variable stabilization
- **IPS** improved primal simplex algorithm
- **ISUD** integral simplex using decomposition
- **LP** linear programming *or* specific linear program
- **MMCC** minimum mean cycle-canceling algorithm
- **PE** positive edge rule
- **PS** primal simplex algorithm
- **SPP** set partitioning problem
- **VSD** vector space decomposition

–To anyone who ever went astray.

REMERCIEMENTS

Ce sera sans surprise pour quiconque aura été présent à mes côtés durant ces dernières années si je dédie mes premiers remerciements à monsieur Jacques Desrosiers. C'est en effet avec beaucoup de reconnaissance que je me remémore toutes les activités auxquelles j'ai eu l'occasion de participer depuis le début de ce projet. Alors que je pensais déjà à lui comme un mentor à l'époque de ma maîtrise, jamais cette impression n'aura au final été ternie. Je me considère en somme privilégié d'avoir travaillé sous sa direction. J'en profite aussi pour saluer les commentaires et les idées toujours à propos de Marco E. Lübbecke qui ont systématiquement amélioré le rendu final de nos travaux.

Je tiens à remercier tous mes collègues et amis gravitant dans l'univers du pavillon André-Aisenstadt avec qui discuter et surtout apprendre toutes sortes de choses connexes ou non à mes recherches furent possible. Un grand merci donc à Antoine, Atoosa, Charles, Hélène-Sarah, Jérémy, Mélisende, Romain, Sara, Sébastien, Serge, Vincent ! À toutes les autres personnes dont l'influence se sera fait sentir en dehors du cadre académique, j'exprime autant de gratitude. À commencer par Evangelos, Jasmine, grand maître Lee, Livia, Malik, Manon et Shazia. Cette liste ne saurait être complète sans une mention spéciale à mes parents pour le soutien indéfectible qu'ils m'ont accordé. Marilène, Philippe, Samuel et Thibault, est-ce vraiment utile d'expliciter qu'il ne s'agit pas d'inadvertance si vous êtes mis en exergue, c'est de tout cœur que je vous embrasse.

J'ai également une pensée pour tout le personnel du GERAD comme celui de HEC Montréal qui ont facilité les démarches administratives et réglé les problèmes techniques auxquels j'ai été confronté durant mon parcours. Je souligne par ailleurs l'appui et le confort dont j'ai bénéficié au sein de ces mêmes établissements.

J'estime en outre important de laisser une trace du concours appréciable de MM. Jean-François Cordeau et Claudio Contardo dans mon parcours doctoral. De plus, ce dernier n'ayant été possible que par mon passage à la maîtrise, je me permets de reprendre mot à mot une réflexion que j'ai eue à l'intention de M. Sylvain Perron dans mon mémoire. *Étudier, c'est avant tout poser un geste de confiance en soi : les aptitudes ne suffisent pas, encore faut-il savoir les orienter, les maîtriser et leur offrir des voies d'épanouissement. C'est pourquoi je pense encore à ses encouragements décisifs dans l'entreprise de ce projet de maîtrise.*

Je termine avec un clin d'œil à tous les étudiants dont le choix de cours aura fait en sorte que nos chemins se croisent. Vous m'avez, peut-être bien malgré vous, fait grandir sur bien des facettes de l'art oratoire.

Avec reconnaissance et affection !

1. INTRODUCTION (FR)

Si la programmation linéaire dispose du simplexe primal (Dantzig 1963) pour parvenir à ses fins, nombreux sont les problèmes pour lesquels cet outil de résolution de prédilection doit composer avec la dégénérescence, voire le cyclage. Le simplexe primal poursuit ses opérations de pivotage jusqu'à la vérification d'un critère d'arrêt affichant les couleurs d'une condition suffisante, mais non nécessaire. En présence de dégénérescence, ce fonctionnement itératif réunit justement les éléments pour mener droit vers un imbroglio. La robustesse du pivotage est compromise et, par suite, l'efficacité de l'outil. Il est d'ailleurs notoire que la famille des problèmes de réseau affiche dans une très large mesure ce phénomène de dégénérescence. Malgré que celui-ci reçoive de l'attention depuis la création de la programmation linéaire, la possibilité d'éradiquer les embûches qu'il cause demeure encore aujourd'hui une question ouverte.

Il n'est pas surprenant que les premières réflexions sur la dégénérescence attaquent de front les opérations de pivotage. Que ce soit les travaux de Bland (1977) ou Fukuda (1982) ou encore Terlaky and Zhang (1993), toutes ces règles de pivot constituent des astuces pour accélérer la convergence pratique du simplexe primal sans pour autant apporter de réponse théorique. Il en est de même pour la perturbation du membre droit de Charnes (1952) qui modifie de manière infinitésimale les hyperplans du polyèdre.

Retenons surtout que toutes ces astuces portent sur le problème global et ne sont plus en mesure, à elles seules, de faire face à la taille sans cesse grandissante des problèmes abordés. Les plus fécondes des nouvelles méthodes proposent de décomposer le problème original pour mieux orienter sa résolution. L'algorithme du simplexe primal amélioré (*Improved Primal Simplex* (IPS) appartient justement à cette classe de méthodes (Elhallaoui et al. 2005, 2008, 2010). Si ce dernier tente de tirer profit de la dégénérescence, plusieurs questions théoriques demeurent sans réponses et tracent le travail accompli dans cette thèse.

1.1 Aperçu

La revue de littérature prend du recul par rapport à IPS et cherche à mieux comprendre le contenu du problème de tarification. Celle-ci est séparée en deux parties chacune ayant donné le jour à un article à part entière. La première partie résume les connaissances autour de l'algorithme du *Minimum Mean Cycle-Canceling* (MMCC). Dans la foulée de cet effort de synthèse apparaît par ailleurs un nouveau résultat de complexité fortement polynomiale dont la construction est incidemment également utile en pratique. Quant à la deuxième partie, nous espérons éliminer toute confusion qui puisse survenir entre IPS et un autre outil développé pour combattre la dégénérescence sur les problèmes de partitionnement : Agrégation de Contraintes Dynamique (*Dynamic Constraint Aggregation*) (DCA). Nous avons envisagé la nécessité de cette revue de littérature détaillée en vertu de la récupération des idées éparpillées tous azimuts.

Le cœur de la thèse construit sur ces acquis et s'articule autour de trois axes. Le premier axe généralise le problème résiduel, traditionnellement associé aux problèmes de flots, aux problèmes linéaires. Des conditions nécessaires et suffisantes pour la programmation linéaire sont ainsi exposées. Nous avons constaté qu'il est possible d'établir de nombreuses connexions entre les principes régissant MMCC et ceux d'IPS dans le cadre des problèmes de réseaux. C'est ainsi que le deuxième axe accompagne la naissance d'un algorithme fortement polynomial de Contraction-Expansion à une étude computationnelle. Le troisième axe place IPS, PS, DCA et MMCC dans un unique cadre permettant en outre d'autres variantes algorithmiques. C'est en utilisant une base du sous-espace vectoriel que cet algorithme générique prend forme. Voilà l'aperçu de la thèse lequel est revisité de manière plus détaillée dans la prochaine section. Chaque paragraphe résume la contribution du travail publié (ou soumis pour publication) ce qui d'office caractérise le statut *par articles* de cette thèse.

1.2 Contribution

Revue de littérature

Dans *About the minimum mean cycle-canceling algorithm*, nous résumons plusieurs articles qui ont chacun contribué à l'analyse de l'algorithme du *minimum mean cycle-canceling* afin de réunir les connaissances sous un seul numéro. Ce processus nous a permis d'établir certaines preuves avec des arguments connus de la programmation linéaire. En outre, le Thèoreme 6 est un aboutissement inattendu qui révèle un nouveau résultat de complexité fortement

polynomiale. Ce dernier est obtenu grâce à la combinaison de résultats précédemment établis et l'élaboration d'une mesure heuristique qui permet d'accéder à un facteur de saut plus important dans un laps de temps plus court. Cet article est publié dans *Discrete Applied Mathematics*, (Gauthier et al. 2015b, voir List of Papers 1).

Dans *Tools for primal degenerate linear programs: IPS, DCA, and PE*, le lecteur trouvera une littérature abondante traitant de la dégénérescence. Nous nous intéressons également à une forme plus générale d'IPS. L'aspect algèbre linéaire qui ressort de cette étude permet d'ailleurs de tracer des parallèles entre ce dernier et l'agrégation de contraintes dynamique ainsi que la règle du *Positive Edge*. Chacun de ces trois outils convient au final à différentes applications. Cet article est publié dans *EURO Journal on Transportation and Logistics*, (Gauthier et al. 2015c, voir List of Papers 2).

Decomposition theorems for linear programs

Dans *Decomposition theorems for linear programs*, le problème résiduel, la pierre angulaire de MMCC, est généralisé aux problèmes linéaires permettant ainsi une éventuelle analyse de complexité d'un algorithme basé sur ces spécifications. En particulier, des conditions nécessaires et suffisantes pour la programmation linéaire en découlent. Ces résultats sont obtenus par le biais d'une décomposition Dantzig-Wolfe dont le sous-problème est à la recherche de soi-disant *cycles pondérés*. Le contenu de cet article est de façon surprenante le fruit d'une question ouverte faisant partie de mon examen de synthèse. Cet article est publié dans *Operations Research Letters*, (Gauthier et al. 2014, voir List of Papers 3).

A strongly polynomial Contraction-Expansion algorithm for network flow problems

La façon la plus simple de décrire cet algorithme est en faisant appel au réseau résiduel. Dans MMCC, chaque arc ayant un flot dont la valeur ne correspond pas à l'une des bornes est dédoublé. Notre proposition consiste à cacher certains arcs du réseau résiduel de telle manière que des arbres indépendants sont identifiés et remplacés par des nœuds uniques dans un réseau alternatif. Cette mécanique est appelée contraction. Le *réseau contracté* ainsi produit est beaucoup plus dense que l'original; il contient moins de nœuds et drastiquement moins d'arcs. Cette *perte* d'information ne compromet ni l'existence de cycles négatifs ni le coût unitaire qui leur est associé. En revanche, l'évaluation de leur coût moyen est modifié

puisque certains arcs ne font plus partis du décompte. En comparaison avec MMCC, l'ordre d'annulation des cycles négatifs est forcément bouleversé.

Les outils utilisés pour construire l'analyse de complexité dans MMCC nous permettent d'énoncer certaines propriétés concernant cette gymnastique de contraction. En témoigne la stratégie *Cancel-and-Tighten* : ce qui ressort de notre étude sur MMCC est que le comportement sur les phases prime devant celui sur les itérations. Les cycles de Type 3 sont désormais introduits pour justement insister sur l'aspect du saut qui est désiré entre les phases. Ceux-ci peuvent être observés dans cet article lorsqu'une contraction partielle est utilisée afin d'imiter le comportement vu dans MMCC assurant ainsi une déclinaison fortement polynomiale. Cette contraction partielle est obtenue en modifiant le choix des arcs cachés au fur et à mesure que l'algorithme progresse. La sélection est faite de telle sorte qu'il s'agit au final d'une *expansion* du réseau contracté. La nécessité de cette expansion pour obtenir ce résultat de complexité demeure pour l'instant sans réponse. Il est par contre intéressant de noter que la contraction partielle peut effectivement être omise pour des applications spécifiques, en particulier celles qui se présentent sous un format binaire comme le problème d'affectation, de plus court chemin ou de flot maximal avec capacité unitaire. L'étude computationelle compare le comportement de l'algorithme de Contraction-Expansion avec ceux de MMCC et *Cancel-and-Tighten*. Cet article soumis pour publication sous le nom de *A strongly polynomial Contraction-Expansion algorithm for network flow problems* adresse ces points (Gauthier et al. 2016, voir List of Papers 4).

Vector space decomposition for linear and network flow problems

Le but de *Vector Space Decomposition for linear programs* est de renforcer les liens entre plusieurs algorithmes existants. Gracieuseté d'une présentation constructive de l'algorithme de Contraction-Expansion, l'article précédent ne contient aucune décomposition matricielle et le rend en l'occurence plus accessible. Les liens entre MMCC et PS sont en revanche cristallisés. Il s'avère que ceux-ci sont diamétralement opposés dans l'univers qui met en lumière un vaste champ de variantes intermédiaires. Également inspiré par le deuxième article en revue de littérature *Tools for degenerate linear programs*, un format générique de décomposition algorithmique est dévoilé.

Puisque différentes propriétés sont avancées pour chacun des cas, nous espérons pouvoir les exploiter au besoin durant la résolution. L'idée principale de cette décomposition de l'espace vectoriel est que la base du sous-espace vectoriel est aussi flexible que l'on désire. Elle peut bien sûr correspondre à une base type simplexe ou bien évoquer l'ensemble des variables libres

ou bien encore décrire l'ensemble vide puis au final tout autre variante. La décomposition de l'espace vectoriel dissocie la base du simplexe des colonnes constituant la solution. Dans tous les cas, un problème de tarification identifiant une direction est appelé. Une famille qui renvoit des directions induisant des pivots non-dégénérés est identifiée ainsi qu'une autre qui trouve naturellement des directions intérieures. IPS est la seule variante appartenant aux deux familles simultanément. Une version préliminaire de l'article est disponible dans *Les Cahiers du GERAD* (Gauthier et al. 2015a, voir List of Papers 5).

Sujets divers

Voici sans ordre particulier plusieurs routes d'exploration. Une analyse de complexité pour les problèmes linéaires reposant sur le paradigme du problème résiduel nécessite une transposition des éléments de preuves vues dans l'analyse de MMCC. En outre, Tardos (1990) montre que tout problème linéaire peut être réduit à un problème de 2-commodités généralisé en temps fortement polynomial. C'est pour ainsi dire qu'il serait assurément plus facile d'aborder ce dernier sans perte de généralité. En effet, le problème de multi-commodités généralisé est similaire à un problème de réseau à plusieurs égards notamment par son support visuel qui contribue largement à mieux comprendre ce qui se passe pendant la résolution.

En ce qui concerne l'implémentation du simplexe, `CPLEX` favorise largement le régime dual. Inutile de se poser en faux devant le leader de l'industrie ce qui nous amène évidemment à considérer une extension dite du *simplexe dual amélioré*. Le vaste domaine de l'optimisation convexe pourrait également bénéficier des propriétés dérivées dans VSD. L'incorporation de la stabilisation des variables duales est une autre stratégie méritant d'être étudiée dans ce cadre. Une autre observation particulièrement à propos, qui ne semble d'ailleurs jamais avoir été relevée, est que le fonctionnement de *Cancel-and-Tighten* fait en sorte que les variables duales sont amenées à une valeur optimale de manière non-décroissante.

Tout ceci est retenu au niveau de la programmation linéaire, mais des extensions visant la programmation en nombres entiers sont également à l'étude. Pensons par exemple aux travaux de Zaghrouti et al. (2014) ainsi que Rosat et al. (2014) traitant du *Integral Simplex Using Decomposition* (ISUD) dont les résultats préliminaires publiés dernièrement sont très positifs. Pour l'instant mis en œuvre autour des problèmes de partitionnement, le problème de tarification est d'autant plus manipulé afin d'extraire des directions améliorantes menant vers des solutions entières. L'intégration de ces connaissances dans un contexte de génération de colonnes est un enjeu prioritaire.

Finalement, ISUD est un algorithme très axé sur l'efficacité pratique. La structure du problème résolu semble être un facteur déterminant puisque ISUD hérite pour l'instant de la même complexité exponentielle que la méthode de séparation et évaluation progressive. Il s'avère ainsi difficile d'imaginer à quel point ISUD est en mesure de s'appliquer hors du contexte des problèmes de partitionnement. Il est cependant remarquable que le problème standard de 2-commodités affiche une relaxation linéaire dont les valeurs fractionnaires se comportent un peu comme dans le problème de partitionnement. Si tout problème linéaire peut être réduit à un problème de 2-commodités, nous disposons en ce moment uniquement d'une réduction en temps polynomial proposée par Itai (1978).

1. INTRODUCTION (EN)

Degeneracy can greatly affect the resolution process of the primal simplex algorithm (PS) going as far as jeopardizing the convergence of the latter. It is in fact the only phenomenon which questions the efficiency of this algorithm. The presence of degeneracy across a broad range of linear programs makes this phenomenon worth studying. This is especially true for network flow problems which are well known for very high level of degenerate pivots. It has drawn attention since the fifties alongside the birth of linear programming. As it stands today, the possibility of eradicating the drawbacks attributed to degeneracy is still an open question.

The primal simplex pursues optimality using pivoting operations until a sufficient stopping criterion is met. When facing degeneracy, this iterative process suffers from null step operations. The robustness of the pivot operation is compromised and, by extension, the efficacy of the resolution tool. It is not surprising that a plethora of pivot rules have fed the content of several papers (Bland 1977, Fukuda 1982, Terlaky and Zhang 1993). Unfortunately, while practical concerns have sometimes been met, none of these rules have provided theoretical answers. The same is true for the work of Charnes (1952) which introduces right-hand side perturbation to modify the way the polyhedron's hyperplanes intersect.

All of these tricks are intended to be used on the global problem and are unable to tackle the problems unceasingly increasing in size. The more successful methods propose to decompose the original problem in order to better guide the resolution. The *Improved Primal Simplex* algorithm (IPS) is one such method (Elhallaoui et al. 2005, 2008, 2010). While the latter aims to capitalize on degeneracy, several theoretical questions remain unanswered and lay the work to be done in this dissertation.

1.1 Outline

The literature review takes a step back from IPS and aims to better understand the content of the pricing problem. It is separated in two parts both of which finding home in inde-

pendent papers. In the first part, a survey examines the Minimum Mean Cycle-Canceling algorithm (MMCC) and further contributes a new strongly polynomial time complexity result whose construction incidentally also brings practical improvements. In the second part, we hope to dissolve the confusion that often arises between IPS and another tool designed to fight degeneracy in set partitioning problems: Dynamic Constraint Aggregation (DCA). We argue for the need of this extensive review on the grounds of the necessity of many ideas that were used previously to accomplish the work ahead.

The dissertation's core builds upon the literature review on three axes. First, we introduce a generalization of the residual problem, traditionally seen in network flow problems, for linear programs. Incidentally, necessary and sufficient optimality conditions for linear programs are also derived from the residual problem. In the second axis, we concentrate on network flow problems and design a strongly polynomial algorithm that uses concepts from both IPS and MMCC. Then again, as we opted for a constructive presentation, this paper omits the ties with IPS altogether and concentrates on the computational study instead. The so-called contraction mechanism is flexible and allows one to go from MMCC to PS depending on the contraction choice. This overarching framework is formally shown in the third axis where this generic algorithm makes use of a vector space decomposition. Such is the dissertation's outline which is revisited in the following section. Each paragraph synthesizes the published (or submitted) work of the contribution officially sealing the *by publication* nature of this dissertation.

1.2 Contribution

Literature review

In *About the minimum mean cycle-canceling algorithm*, we survey different papers which analyze the minimum mean cycle-canceling algorithm to recuperate all the ideas under one roof. This process allows us to establish some of the proofs using a new line of arguments, that is, with linear programming tools. Ultimately, it even leads to an unexpected improved strongly polynomial result stated in Theorem 6. The latter is obtained via a mixture of previous results and the elaboration of a heuristic measure which allows the generalization for a higher jump factor in a smaller time period. This paper is published in *Discrete Applied Mathematics*, (Gauthier et al. 2015b, see List of Papers 1).

In *Tools for primal degenerate linear programs: IPS, DCA, and PE*, the reader shall find extensive degeneracy related literature. We are additionally interested about a more general

form of IPS. The linear algebra framework that emanates from the study actually allows us to draw connections between the latter and Dynamic Constraint Aggregation as well as Positive Edge. Each of these three degeneracy fighting tool is suitable for different applications. This paper is published in *EURO Journal on Transportation and Logistics*, (Gauthier et al. 2015c, see List of Papers 2).

Decomposition theorems for linear programs

In *Decomposition theorems for linear programs*, the residual problem, MMCC's core, is generalized to linear programs and provides for a solid building block for an eventual complexity analysis of an algorithm based on these specifications. Necessary and sufficient optimality conditions for linear programs are also derived. These results are obtained using a Dantzig-Wolfe decomposition which issues a subproblem angled towards finding so-called *weighted cycles*. Surprisingly enough, the article's content is born from an open question raised in my comprehensive exam. This paper is published in *Operations Research Letters*, (Gauthier et al. 2014, see List of Papers 3).

A strongly polynomial Contraction-Expansion algorithm for network flow problems

The easiest way to describe this algorithm is with respect to the residual network. In MMCC, every arc for which the current solution displays a flow that is not at its bound is doubled. Our proposal hides some of the arcs from the residual network. In doing so, independent trees are identified and replaced by single nodes in an alternative network. This mechanic is called contraction. The resulting so-called *contracted* network is much more dense than the original one, it contains less nodes and drastically less arcs. This *loss* of information does not compromise the existence of negative cycles nor their unit cost. It does however modify the average evaluation of this cost since some arcs are no longer accounted for. The order in which the negative cycles are canceled is thus modified when one compares it with MMCC.

The tools provided by the complexity analysis of MMCC allow us to state certain properties regarding this contraction gymnastic. The most important behavior noted with MMCC is that of phases as seen in the *Cancel-and-Tighten* strategy. In fact, the paper also introduces a so-called Type 3 cycle which insists on the measurable jump aspect we wish for between phases. It can also be observed in this framework when using partial contraction to mimic the

behavior of MMCC thus ensuring a strongly polynomial algorithm. This partial contraction is obtained by modifying the choice of hidden arcs as the algorithm progresses. The selection is made in such a way that it actually corresponds to an *expansion* of the contracted network. Whether this expansion is required to achieve strongly polynomial properties is still an open question. Although, it is interesting to note that strongly polynomial time complexity is indeed verified without partial contraction for special applications, namely those in binary format (assignment, shortest path, maximum flow with unit capacities). The computational study compares the behavior of the Contraction-Expansion with those of MMCC and Cancel-and-Tighten. The submitted paper titled *A strongly polynomial Contraction-Expansion algorithm for network flow problems* aims to address these points (Gauthier et al. 2016, see List of Papers 4).

Vector space decomposition for linear and network flow problems

The purpose of *Vector space decomposition for linear and network flow problems* is to further establish the different ties that exist among several algorithmic variants we have faced this far including IPS. Indeed, the Contraction-Expansion algorithm is presented in a constructive manner thus eliminating any involved matrix decomposition and maybe even becoming more accessible as such. The ties with MMCC and PS are however made clear, and it appears that MMCC and PS sit diametrically opposed to one another yet a whole range of variants in between is brought to light. Also inspired by the second literature review paper *Tools for degenerate linear programs*, a generic format for algorithmic decomposition is ultimately presented.

Since different properties are established for each of these cases, the hope is that it will be possible to exploit these properties as needed during the resolution process. The fundamental idea of this vector space decomposition is that the vector subspace basis is as flexible as one wishes. It can correspond to a simplex basis or evoke the set of free variables or even be the null-space and in fact just about everything in between. The vector space decomposition dissociates the simplex basis from the column components of the solution. In all cases, a pricing problem which identifies a direction is called upon. A family which fetches directions that induce nondegenerate pivots is identified as well as another that naturally finds interior directions. IPS is the only variant that belongs to both families simultaneously. A preliminary version is available in *Les Cahiers du GERAD* (Gauthier et al. 2015a, see List of Papers 5).

Miscellaneous subjects

We list in no particular order several paths of study that remain to be explored. A complexity analysis for linear programs within the paradigm of the residual problem requires a translation of the necessary elements of MMCC's analysis. As a matter of fact, Tardos (1990) shows that any linear program can be reduced to a 2-commodity generalized flow problem in strongly polynomial time. For all intents and purposes, this also means that it might be much more interesting to work on this particular problem. Indeed, the multi-commodity generalized flow problem is similar to network flow problems on a variety of points including the visual support which highly contributes to a better understanding.

`CPLEX` swears by its dual simplex implementation. Fair enough, it's hard to argue with the measuring stick of the industry, it therefore seems only natural to work on an *improved dual simplex* extension. The broad field of convex optimization might also benefit from the properties derived in VSD. The incorporation of dual variables stabilization is another strategy worth considering in this framework. On this subject, Cancel-and-Tighten also features a nondecreasing property of the dual variables which does not seem to have been mentioned anywhere.

All of this remains true the field of linear programming but integer programming extensions are also in the works. In this respect, a great deal of work has been done on the *Integral Simplex Using Decomposition* (ISUD). Applied on set partitioning problems, the pricing problem is further manipulated to extract improving directions that also maintain integrality of the solution. Many promising results, such as those of Zaghrouti et al. (2014) and Rosat et al. (2014), have been published in the recent years. The underlying goal is to integrate this knowledge in column generation applications.

Furthermore, ISUD is a very practical algorithm for which the problem's structure seems to be of considerable importance. Indeed, alike branch-and-bound methods, it too features exponential time complexity. How well ISUD extends outside the scope of set partitioning problems is unclear. Then again, it is the noteworthy to observe that the standard 2-commodity flow problem features a linear relaxation that sports fractional values similar to those of the set partitioning problem. Moreover, all linear programs can be reduced to a 2-commodity flow problem, although only a transformation in polynomial time due to Itai (1978) is available today.

About the minimum mean cycle-canceling algorithm

Jean Bertrand Gauthier and Jacques Desrosiers

GERAD & HEC Montréal, Canada

3000, chemin de la Côte-Sainte-Catherine

Montréal (Québec) Canada, H3T 2A7

<jean-bertrand.gauthier, jacques.desrosiers>@hec.ca

Marco E. Lübbecke

RWTH Aachen University, Germany

Kackertstraße 7

D-52072 Aachen, Germany

marco.luebbecke@rwth-aachen.de

ABSTRACT

This paper focuses on the resolution of the capacitated minimum cost flow problem on a network comprising n nodes and m arcs. We present a method that counts imperviousness to degeneracy among its strengths, namely the *minimum mean cycle-canceling* algorithm (MMCC). At each iteration, primal feasibility is maintained and the objective function strictly improves. The goal is to write a uniform and hopefully more accessible paper which centralizes the ideas presented in the seminal work of Goldberg and Tarjan (1989) as well as the additional paper of Radzik and Goldberg (1994) where the complexity analysis is refined. Important properties are proven using linear programming rather than constructive arguments.

We also retrieve Cancel-and-Tighten from the former paper, where each so-called phase which can be seen as a group of iterations requires $O(m \log n)$ time. MMCC turns out to be a strongly polynomial algorithm which runs in $O(mn)$ phases, hence in $O(m^2 n \log n)$ time. This new complexity result is obtained with a combined analysis of the results in both papers along with original contributions which allows us to enlist Cancel-and-Tighten as an acceleration strategy.

Keywords: Network flow problem, residual network, flow decomposition, minimum mean cycle, complexity analysis, strongly polynomial algorithm.

2.1 Introduction

This paper addresses the resolution of the capacitated minimum cost flow problem (CMCF) on a network defined by n nodes and m arcs. We present the *minimum mean cycle-canceling* algorithm (MMCC). The seminal work of Goldberg and Tarjan (1989), as presented in the book of Ahuja et al. (1993), as well as the paper of Radzik and Goldberg (1994), where the complexity analysis is refined, are the underlying foundations of this document. The current literature states that MMCC is a strongly polynomial algorithm that performs $O(m^2n)$ iterations, a tight bound, and runs in $O(m^3n^2)$ time.

While Goldberg and Tarjan (1989) present Cancel-and-Tighten as a self-standing algorithm, we feel it belongs to the realm of acceleration strategies incidentally granting the reduction of the theoretical complexity. Our understanding is that this strategy can be shared at any level of the complexity analysis. Indeed, its very construction aims to assimilate the so-called notion of *phase* which can be seen as a group of iterations. This strategy exploits an approximation scheme to manage this assimilation and as such nevertheless necessitates a careful analysis. We propose a new approximation structure which allows us to reduce the global runtime to $O(m^2n \log n)$. It is namely the product of a refined analysis that accounts for $O(mn)$ phases, each one requiring $O(m \log n)$ time.

The reader should view this work as much more than a synthesis. It is the accumulation of years of research surrounding degeneracy that led us to realize the ties with theories drafted some forty years ago. We not only hope to clarify the behavior of the minimum mean cycle-canceling algorithm but also provide strong insights about the ins and outs of its idiosyncrasies and more importantly establish a solid unified framework against which we can rest current and future work. On that note, let us underline the linear programming mindset which simplifies the construction of one of the most important part of the algorithm, namely the pricing problem. The justification of some of its properties also benefit from straightforward implications provided by that mindset. Some fundamental properties of network problems are also incorporated throughout the text which sometimes facilitate if not, certainly enlighten, the comprehension of the proofs presented by the listed authors.

The paper is organized as follows. The elaboration of MMCC takes place in Section 2.2 where the combination of the so-called *residual network* along with optimality conditions give birth to a pricing problem which is put to use in an iterative process. Section 2.3 analyzes its complexity which is decomposed in two parts: the *outer loop* and the *bottleneck*. Although the latter comes at the very last, it acts as the binding substance of the whole paper. It is indeed

where the behavior of the algorithm can be seen at a glance alongside the justification for the significance of the aforementioned *phases*. This is followed by the conclusion in Section 2.4.

2.2 Minimum mean cycle-canceling algorithm

Consider the formulation of CMCF on a directed graph $G = (N, A)$, where N is the set of n nodes associated with an assumed balanced set $b_i, i \in N$, of supply or demand defined respectively by a positive or negative value such that $\sum_{i \in N} b_i = 0$, A is the set of m arcs of cost $\mathbf{c} := [c_{ij}]_{(i,j)\in A}$, and $\mathbf{x} := [x_{ij}]_{(i,j)\in A}$ is the vector of bounded flow variables:

$$
\begin{aligned}
z^* := \quad & \min \sum_{(i,j)\in A} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i, \qquad [\pi_i] \qquad \forall i \in N \\
& 0 \leq \ \ell_{ij} \leq x_{ij} \leq u_{ij}, \qquad \forall (i,j) \in A,
\end{aligned}
\tag{2.1}
$$

where $\boldsymbol{\pi} := [\pi_i]_{i\in N}$ is the vector of dual variables, also known as node potentials. When right-hand side $\mathbf{b} := [b_i]_{i\in N}$ is the null vector, formulation (2.1) is called a *circulation* problem.

Let us enter the world of network solutions with a fundamental proposition whose omitted proof traditionally relies on a constructive argument. It is so rooted in the network design that, case in point, straightforward derivatives are used throughout this document.

Proposition 1. *(Ahuja et al. 1993, Theorem 3.5 and Property 3.6) Any feasible solution* $\mathbf{x}$ *to* (2.1) *can be represented as a combination of paths and cycles flows (though not necessarily uniquely) with the following properties:*
(a) Every directed path with positive flow connects a supply node to a demand node; at most $n + m$ *directed paths and cycles have non-zero flow among which at most* m *cycles.*
(b) In the case of a circulation problem, by definition there are no supply nor demand nodes, which means the representation can be restricted to at most m *directed cycles.*

This section derives MMCC, devised to solve instances of CMCF, in the following manner. Section 2.2.1 defines the corner stone of the resolution process, namely the residual network. Whether its inception goes back to the optimality conditions or its usage came as an afterthought is an enigma for which we have no answer. Either way, the latter are introduced thereafter and pave the way for the pricing problem in Section 2.2.2. Section 2.2.3 exhibits the algorithmic process which is ultimately information sharing between a control loop and a

pricing problem. The former ensures primal feasibility while the latter provides a strictly improving direction at each iteration. Section 2.2.4 illustrates the behavior of the algorithm on the maximum flow problem.

2.2.1 Residual network and optimality conditions

The residual network takes form with respect to a feasible flow $\mathbf{x}^0 := [x^0_{ij}]_{(i,j)\in A}$ and is denoted $G(\mathbf{x}^0) = (N, A(\mathbf{x}^0))$. As eloquently resumed in Figure 2.1, each arc $(i,j) \in A$ is replaced by two arcs representing upwards and downwards possible flow variations:

- arc (i,j) with cost $d_{ij} = c_{ij}$ and residual flow $0 \leq y_{ij} \leq r^0_{ij} := u_{ij} - x^0_{ij}$;
- arc (j,i) with cost $d_{ji} = -c_{ij}$ and residual flow $0 \leq y_{ji} \leq r^0_{ji} := x^0_{ij} - \ell_{ij}$.

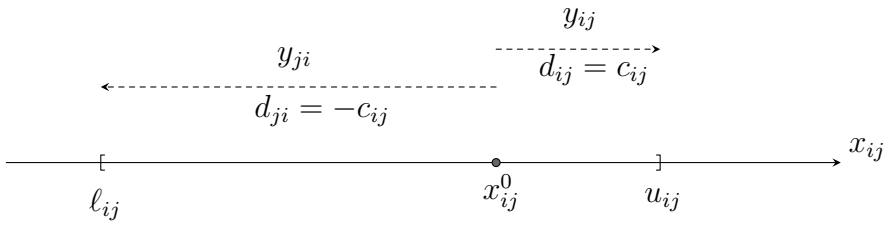


Fig. 2.1: A change of variables

Denote $A' := \{(i,j) \cup (j,i) \mid (i,j) \in A\}$ as the complete possible arc support of any residual network. The residual network $G(\mathbf{x}^0)$ consists of only the residual arcs, i.e., those with strictly positive residual capacities, that is, $A(\mathbf{x}^0) := \{(i,j) \in A' \mid r^0_{ij} > 0\}$. The combination of the current solution $\mathbf{x}^0$ along with the optimal marginal flow computed on the residual network is optimal for the original formulation. Indeed, the residual network with respect to $\mathbf{x}^0$ corresponds to the change of variables $x_{ij} = x^0_{ij} + (y_{ij} - y_{ji}),\ \forall (i,j) \in A$. Observe that traveling in both directions would be counterproductive and can be simplified to the net flow in a single direction. This means that the marginal flow must be such that $y_{ij}\, y_{ji} = 0,\ \forall (i,j) \in A$, which is naturally verified by any practical solution.

Letting $z^0 = \mathbf{c}^\intercal \mathbf{x}^0$ means that CMCF can be reformulated as:

$$z^* := z^0 + \min \sum_{(i,j)\in A(\mathbf{x}^0)} d_{ij} y_{ij} \tag{2.2}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A(\mathbf{x}^0)} y_{ij} - \sum_{j:(j,i)\in A(\mathbf{x}^0)} y_{ji} = 0, \qquad [\pi_i] \qquad \forall i \in N \tag{2.3}$$

$$0 \leq y_{ij} \leq r^0_{ij}, \qquad \forall (i,j) \in A(\mathbf{x}^0). \tag{2.4}$$

Take a moment to consider an optimal solution of (2.2)–(2.4) on the residual network $G(\mathbf{x}^0)$. Mathematically speaking, it corresponds to a circulation problem. The right-hand side in (2.3) is zero everywhere, we are thus looking for a solution that respects the equilibrium already present in the current solution $\mathbf{x}^0$. Verifying that a directed cycle in $G(\mathbf{x}^0)$ exists as a cycle in G is as straightforward as applying the flow conservation principle, that is, we use the forward direction of arc (i, j) when $y_{ij} > 0$ or the backward direction when $y_{ji} > 0$.

Suppose that $\mathbf{x}$ and $\mathbf{x}^0$ are any two feasible solutions to formulation (2.1). Therefore some feasible circulation $\mathbf{y}$ in $G(\mathbf{x}^0)$ satisfies the property that $\mathbf{x} = \mathbf{x}^0 + \mathbf{y}$, and the cost of solution $\mathbf{x}$ is given by $\mathbf{c}^\intercal\mathbf{x} = \mathbf{c}^\intercal\mathbf{x}^0 + \mathbf{d}^\intercal\mathbf{y}$, where $\mathbf{d} := [d_{ij}]_{(i,j)\in A(\mathbf{x}^0)}$. Moreover, Proposition 1(b) means that there exists a way to decompose $\mathbf{y}$ in at most m cycles. We can therefore think of an optimal solution of (2.2)–(2.4) on $G(\mathbf{x}^0)$ as a collection of intertwined cycles. This is stated in the following proposition.

Proposition 2. *(Ahuja et al. 1993, Theorem 3.7) Let* $\mathbf{x}$ *and* $\mathbf{x}^0$ *be any two feasible solutions of a network flow problem. Then* $\mathbf{x}$ *equals* $\mathbf{x}^0$ *plus the flow on at most* m *directed cycles in* $G(\mathbf{x}^0)$*. Furthermore, the cost of* $\mathbf{x}$ *equals* $\mathbf{c}^\intercal\mathbf{x}^0$ *plus the cost of flow on these augmenting cycles.*

Hence, there exists a way to move between any two feasible solutions in at most m cycles! It is quite a testament to how trivial reaching $\mathbf{x}^*$, *granted it is actually known*. The fact of the matter is that the residual network problem (2.2)–(2.4) is not easier to solve than the original problem (2.1). Nevertheless, there exists at least one sequence of transitions which constructs a series of residual networks allowing to move from $\mathbf{x}^0$ to an optimal solution $\mathbf{x}^*$ in a finite number of iterations.

It is indeed possible to think of the cycles contained in the residual network as transitioning possibilities. Consider the marginal changes instilled in $\mathbf{x}^0$ with respect to *some* negative (or improving) cycle and repeat this step until no such cycle remains. As simple as it may sound, we have stated the *generic cycle-canceling* algorithm as proposed by Klein (1967), which ultimately amounts to a line search optimization method. Showing finiteness, at least as far as today's computer tractability is concerned (Ford and Fulkerson (1962) show that pathological instances with irrational data could misbehave indefinitely or even worse converge to a bad solution), is as trivial as realizing this procedure performs a strict improvement in the objective function at each iteration until optimality is reached. However, it turns out that the order in which these cycles are identified has tremendous impact on the performance of this generic algorithm. Given integer data, denote the greatest absolute cost value by $C := \max_{(i,j)\in A} |c_{ij}|$ and the greatest interval range value by $U := \max_{(i,j)\in A} u_{ij} - \ell_{ij}$. Then the number of iterations of the generic algorithm ranges anywhere from $O(m)$ to $O(mCU)$.

Optimality conditions. With respect to $\boldsymbol{\pi}$, the reduced cost of variable $x_{ij}, (i,j) \in A$, is given by $\bar{c}_{ij} := c_{ij} - \pi_i + \pi_j$. Let the reduced cost $\bar{d}_{ij}$ of variable $y_{ij}, (i,j) \in A(\mathbf{x}^0)$, be computed in the same way, i.e., $\bar{d}_{ij} := d_{ij} - \pi_i + \pi_j$. For a feasible flow $\mathbf{x}^0$, we distinguish three equivalent necessary and sufficient optimality conditions. With respect to linear programming vocabulary, the first two can be qualified of primal and dual nature on the residual network $G(\mathbf{x}^0)$ while the third is that of complementary slackness on network G, see Ahuja et al. (1993, Theorems 9.1, 9.3, and 9.4):

Primal: $G(\mathbf{x}^0)$ contains no negative cycle.

Dual: $\exists \boldsymbol{\pi}$ such that $\bar{d}_{ij} \geq 0, \forall (i,j) \in A(\mathbf{x}^0)$.

Complementary slackness: $\exists \boldsymbol{\pi}$ such that, for every arc $(i,j) \in A$,

$$x^0_{ij} = \ell_{ij} \text{ if } \bar{c}_{ij} > 0; \qquad x^0_{ij} = u_{ij} \text{ if } \bar{c}_{ij} < 0; \qquad \bar{c}_{ij} = 0 \text{ if } \ell_{ij} < x^0_{ij} < u_{ij}. \tag{2.5}$$

We underscore that if feasible flow $\mathbf{x}^0$ is actually optimal, all these conditions are verified simultaneously. Observe that the primal and dual conditions are only verifiable on the residual network $G(\mathbf{x}^0)$. The complementary slackness conditions however are verified on G by the combination of the current primal solution and the information gathered by the dual vector.

2.2.2 Pricing step: maximizing the minimum reduced cost

The pricing step elaborated in this section is derived from the residual network by capturing the rationale of the optimality conditions. According to the dual optimality condition, $\mathbf{x}^0$ is optimal if and only if there exists a vector $\boldsymbol{\pi}$ such that $d_{ij} - \pi_i + \pi_j \geq 0, \ \forall (i,j) \in A(\mathbf{x}^0)$. This can be verified by maximizing the smallest reduced cost, denoted μ^0, and formulated as the following linear program:

$$\mu^0 := \max \quad \mu \tag{2.6}$$

$$\text{s.t.} \quad \mu + \pi_i - \pi_j \leq d_{ij}, \qquad [y_{ij}] \quad \forall (i,j) \in A(\mathbf{x}^0). \tag{2.7}$$

Observe that $\boldsymbol{\pi}$ is not fixed but optimized in formulation (2.6)–(2.7). Its dual is expressed in terms of flow variables $[y_{ij}]_{(i,j)\in A(\mathbf{x}^0)}$:

$$\mu^0 := \min \sum_{(i,j)\in A(\mathbf{x}^0)} d_{ij} y_{ij} \tag{2.8}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A(\mathbf{x}^0)} y_{ij} - \sum_{j:(j,i)\in A(\mathbf{x}^0)} y_{ji} = 0, \quad [\pi_i] \quad \forall i \in N \tag{2.9}$$

$$\sum_{(i,j)\in A(\mathbf{x}^0)} y_{ij} = 1, \quad [\mu] \tag{2.10}$$

$$y_{ij} \geq 0, \quad \forall (i,j) \in A(\mathbf{x}^0), \tag{2.11}$$

where $\boldsymbol{\pi}$ is associated with the flow conservation constraints (2.9) while μ is a dual scalar associated with the convexity constraint (2.10). We already know optimality conditions provide alternative ways to prove the optimality status of feasible solution $\mathbf{x}^0$. It might not be all that surprising that if pricing problem (2.6)–(2.7) expresses the dual optimality condition on $G(\mathbf{x}^0)$, formulation (2.8)–(2.11) echoes the verification of the primal optimality condition on the residual network. Indeed, the latter is known as the *minimum mean cycle* problem, arguably giving all meaning to the algorithm's name. The following paragraphs contain the explanation and a word of justification that allows it to stand on its own.

The convexity constraint (2.10) produces a scaling in the $\mathbf{y}$-variables which is echoed in the objective function. As a matter of fact, problem (2.8)–(2.11) no longer belongs to the family of network problems. Nevertheless, that scaling does not compromise the existence of a cycle in $G(\mathbf{x}^0)$, but it does create a distortion of the cost associated with said cycle. The meaning of this distortion resides in the fact that (2.8)–(2.11) finds a single directed cycle with the smallest average cost, the average being taken over the number of arcs (or nodes) in the cycle. Notice the use of the word cycle against formulation (2.8)–(2.11) which we have explicitly excluded from the network family. The concept is so important, we take the time to break the flow of the text to carry an explanation.

Define $W^0 := \{(i,j) \in A(\mathbf{x}^0) \mid y^0_{ij} > 0\}$ as the set of active variables in an optimal solution $\mathbf{y}^0$ to formulation (2.8)–(2.11). Granted W^0 describes a single cycle, constraint set (2.9) guarantees that the value is the same for all the variables that are actually present in that selected cycle. Therefore, we must have $y^0_{ij} = 1/|W^0|, \ \forall (i,j) \in W^0$. Furthermore, we say W^0 is directed with respect to the orientation of the arcs in $G(\mathbf{x}^0)$ corresponding to the selected positive variables in $\mathbf{y}^0$. While the notation is abusive, the burden of an additional variable for values so closely related is not worthwhile. In any single cycle, at most one of y^0_{ij} or y^0_{ji} may be positive which in turn satisfies the flow condition $y_{ij}y_{ji} = 0$. Fortunately, the expectancy of this particular kind of solution is not a strong restriction as it is synonymous of an extreme point solution of the linear program (2.8)–(2.11). There is one notable exception to this one-way rule which can only happen when $\mathbf{x}^0$ is optimal. Since the identified cycle can be discarded for lack of improvement, so can the exception. From now on, a solution to the pricing step is assumed to honor the design of the minimum mean cycle problem meaning

that W^0 is a single directed cycle. Furthermore, we can interchangeably speak of cycle W^0 on $G(\mathbf{x}^0)$ or $A(\mathbf{x}^0)$. Finally, note that in the dual formulation, one can additionally impose $\mu \leq 0$. As a consequence, the associated convexity constraint in (2.8)–(2.11) becomes a less than or equal inequality and the primal pricing problem is always feasible even if the residual network is acyclic (in which case $\mu = 0$). Observe that if the obtained cycle has a negative mean reduced cost, $\mathbf{x}^0$ is not optimal for (2.1). The solution of the pricing step can therefore be seen as a *direction*. By definition of the residual network, it even qualifies as a strictly improving direction.

Remark. The justification for the validity of the primal version of the pricing step might go as follows. We are looking to improve current solution $\mathbf{x}^0$ using the concept of negative cost cycles. More specifically, we are looking for the existence of such a cycle, say W^0 on $G(\mathbf{x}^0)$. Observe that the residual capacities are not relevant in the cycle identification process. Then again, omitting these quantities from formulation (2.2)–(2.4), that is, removing $y_{ij} \leq r^0_{ij},\ \forall (i,j) \in A(\mathbf{x}^0)$, creates an unbounded circulation problem. The silver lining comes from the realization that this new problem is a cone for which the unique extreme point solution $\mathbf{y} = \mathbf{0}$ reflects *status quo*. Non linear optimization has an impressive inventory of choices to search among improving directions. These choices all have their rationalization but ultimately mean that the cone is cut according to some metric. One of the recognized choice is the convexity constraint imposed on the selection. Indeed, for any non-zero solution in the cone, there exists a scaled one such that $\mathbf{1}^\intercal \mathbf{y} = 1$. Historically speaking, the pricing step is reported as an abstract form of its interpretation, that is, $\min_W \sum_{(i,j)\in W} d_{ij}/|W|$. Whether Goldberg and Tarjan (1989) accidentally built the minimum mean cycle-canceling algorithm according to these principles or it was meticulously devised is unclear, the conclusion is all the same: the convexity constraint is indeed enlisted in (2.8)–(2.11).

2.2.3 Algorithmic process

MMCC is initialized with a feasible flow $\mathbf{x}^0$. At every iteration $k \geq 0$, the pricing step solution $\mathbf{y}^k$ identifies a minimum mean cost cycle $W^k := \{(i,j) \in A(\mathbf{x}^k) \mid y^k_{ij} > 0\}$ taking value μ^k in $G(\mathbf{x}^k)$. Flow units are sent along this cycle according to a control mechanism which relies solely on the residual capacities $\mathbf{r}^k$. A new solution $\mathbf{x}^{k+1}$ is obtained and $G(\mathbf{x}^{k+1})$ is updated accordingly. This process is repeated until the residual network contains no negative cycle.

Let us take a look at some computations that can be done regarding the transition between two iterations. The flow of every arc in the negative cost cycle W^k can be augmented by the

smallest residual capacity on the cycle $\delta^k := \min_{(i,j)\in W^k} r_{ij}^k$, hence the new solution becomes

$$x_{ij}^{k+1} = x_{ij}^k + \delta^k |W^k|(y_{ij}^k - y_{ji}^k), \forall (i,j) \in A, \tag{2.12}$$

and the improvement Δz^k of the objective function in (2.1) is given by

$$\Delta z^k := \delta^k |W^k| \mu^k = \delta^k \sum_{(i,j)\in W^k} d_{ij} < 0. \tag{2.13}$$

Notice that both δ^k and Δz^k evaluate to integers if certain conditions are verified. The former requires the integrality of the bounds as well as the demands/supplies while the latter depends on the integrality of the costs.

It stands to reason that MMCC is already in the works with Ford and Fulkerson (1956) providing the concept of *augmenting paths* between solutions while Edmonds and Karp (1972) show that a particular selection of augmenting paths would be more efficient on the maximum flow problem. While we use the latter to present an application of MMCC, the reader is invited to appreciate the narrative description as a tribute to the aforementioned papers. The illustrative example also serves to get a feel for the subsequent complexity analysis.

2.2.4 Illustrative example: the maximum flow problem

The maximum flow problem is a particular instance of network optimization in which a source s and a sink t are connected through a capacitated subnetwork. The goal is to maximize the outgoing flow of the source under the restriction of the usual flow conservation constraints. One should realize the null cost structure of all the arcs except $x_{ts} \geq 0$ for which $c_{ts} = -1$. Let us apply MMCC and assume lower bounds are null for all arcs, meaning that $\mathbf{x}^0 = \mathbf{0}$ is feasible.

It is worthwhile to notice that the cycle found on the residual network at any iteration $k \geq 0$ is constructed in two parts: a path from s to t and the lone variable y_{ts}. Let W^k be the negative cycle identified at iteration k, hence $\mu^k = -1/|W^k|$. The pricing step sequentially favors the smallest paths (in number of arcs) from s to t starting from length 1 to $n-1$ until optimality is reached. The sequence of μ^k is non-decreasing and takes its values from the finite set $\left\{\frac{-1}{2}, \frac{-1}{3}, \frac{-1}{4}, \ldots, \frac{-1}{n}\right\}$. When the path length changes from 2 to a longer one, the increase factor of μ takes a value among range $\left\{\frac{2}{3}, \frac{2}{4}, \ldots, \frac{2}{n}\right\}$. Observe that higher values of the increase factor induce smaller jumps on μ. Therefore, the minimal increase factors $\left\{\frac{2}{3}, \frac{3}{4}, \ldots, \frac{n-1}{n}\right\}$ of μ^k, for each length level, are computed using adjacent values of μ, the smallest possible one

being attained when going from level $\frac{-1}{n-1}$ to $\frac{-1}{n}$ and measured by $(1 - 1/n)$. Figure 2.2 depicts the behavior of $\mu^k, k \geq 0$, on a network comprising 502 nodes and 10,007 arcs: only four levels of μ are required within 188 iterations at which point $\mu = 0$ thus proving optimality and discarding the associated cycle defined by variables $y_{st}^{188} = y_{ts}^{188} = 1/2$. Observe that in this particular example, the increase factors are $\frac{5}{6}$, $\frac{6}{7}$ and $\frac{7}{8}$ all of which producing a bigger increase on μ than would have $\frac{501}{502}$.

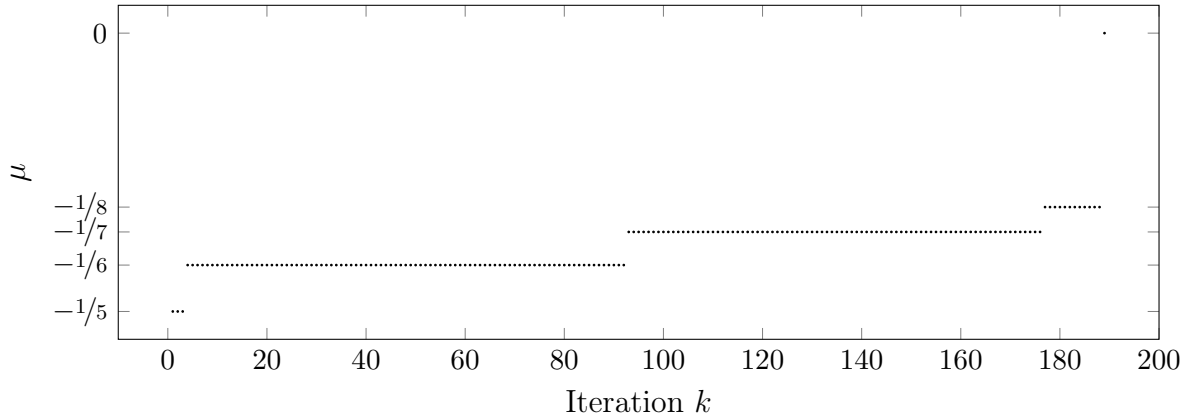


Fig. 2.2: Smallest reduced cost μ for a maximum flow problem

We draw the reader's attention on solving the pricing step. We are looking, on the residual network, for the shortest path (in number of arcs) from s to t. This can be done in $O(m)$ using a breadth-first-search algorithm. Next, we derive from the previous paragraph that there are at most $n-1$ increases of μ, and since every iteration identifies a path through which at least one arc is saturated with the step size, each increase is attained within m iterations, hence $O(mn)$ iterations. In total, Goldberg and Tarjan (1989) realize that applying MMCC to the maximum flow problem exactly corresponds to the strongly polynomial algorithm of Edmonds and Karp (1972) which runs in $O(m^2n)$ time.

2.3 Complexity analysis

The complexity analysis is implicitly decomposed in two parts: the *outer loop* and the *bottleneck*. Obtaining the global runtime is then a matter of factoring out these complexities. In MMCC, the natural definition of the bottleneck relates to the pricing step and we therefore study the upper bound on the number of calls made to the latter. For the sake of argument, one can think of the bottleneck as a group of calls which, in the end, is purely cosmetic. Yet, an efficacy gain is made if solving for a group can be done more efficiently than would the sequential operations. That is where lies the significance of the so-called *phases*.

Although this paper recruits its inspiration from the very fine presentation of the minimum mean cycle-canceling algorithm proposed by Goldberg and Tarjan (1989) and described in Ahuja et al. (1993), the presentation is reorganized to first thoroughly discuss the outer loop analysis (Sections 2.3.1 to 2.3.4) and then spend time on the bottleneck management (Section 2.3.5). We also rapidly divert to phase-wise results in accordance with our understanding of the Cancel-and-Tighten strategy. The latter is presented in the bottleneck management section opposite the traditional pricing problem. Ultimately, we consolidate all complexity results in the summary (Section 2.3.6). The latter also points out different practical aspects of the algorithm by using computational results.

We start with basic properties of the algorithm which lead to the actual complexity analysis. The first proposal of Goldberg and Tarjan (1989) is a weakly polynomial behavior of $O(mn \log(nC))$ iterations for integer arc costs while the second establishes the strongly polynomial result of $O(m^2 n \log n)$ iterations for arbitrary real-valued arc costs. Radzik and Goldberg (1994) refine it to $O(m^2 n)$ iterations and also show this bound to be tight. The concept of phase is strategically positioned after the first complexity result expressed in terms of iterations to allow the reader to appreciate the similarity. The bottleneck management brings the Cancel-and-Tighten strategy into play and reduces the global runtime complexity to $O(m^2 n \log n)$, our new complexity result for the minimum mean cycle-canceling algorithm.

2.3.1 In embryo

Let us recall a fundamental network flow property before dwelling in the algorithmic analysis. It examines the relationship with the cost and the reduced cost of a cycle.

Cycle cost. For any vector $\boldsymbol{\pi}$ of node potentials, the cost and the reduced cost of a directed cycle W in $G(\mathbf{x}^k), k \geq 0$, are equal. Finding a minimum mean cost cycle W^k in $G(\mathbf{x}^k)$ is therefore equivalent to finding a minimum mean *reduced* cost cycle W^k in $G(\mathbf{x}^k)$. Hence, the optimal value of the objective function in pricing problem (2.8)–(2.11) computes

$$\mu^k = \sum_{(i,j)\in W^k} d_{ij} y_{ij}^k = \sum_{(i,j)\in W^k} \frac{d_{ij}}{|W^k|} = \sum_{(i,j)\in W^k} \frac{\bar{d}_{ij}}{|W^k|}. \tag{2.14}$$

The first equality sums over the optimal cycle, the second uses the fact that we know all strictly positive $\mathbf{y}$-variables are equal to one another, and the last recalls the equivalence between the cost and the reduced cost of a cycle.

Optimality parameter μ. The mechanics of the minimum mean cycle-canceling algorithm do not require the use or computation of reduced costs. Indeed, MMCC relies solely on the primal optimality condition to achieve optimality. The complexity analysis however exploits the dual and complementary slackness conditions using the equivalences provided by (2.14). Ultimately, the idea is to study the convergence towards zero of μ^k, the current most negative reduced cost. The synonymy is granted as a side effect of Proposition 3, for which the proof is given using linear programming tools, and is the reason we interchangeably use the expression *optimality parameter*. With respect to $\boldsymbol{\pi}^k := [\pi_i^k]_{i \in N}$ at iteration $k \geq 0$, let $\bar{c}_{ij}^k := c_{ij} - \pi_i^k + \pi_j^k, (i,j) \in A$, be the reduced cost of variable x_{ij}. In the same way, $\bar{d}_{ij}^k := d_{ij} - \pi_i^k + \pi_j^k$ is the reduced cost of variable $y_{ij}, (i,j) \in A(\mathbf{x}^k)$. Observe that the superscript is understood to mean the computation is done with the corresponding vector $\boldsymbol{\pi}^k$ of node potentials.

Proposition 3. *(Goldberg and Tarjan 1989,* Theorem 3.3*) Given a non-optimal solution $\mathbf{x}^k$, $k \geq 0$, there exists some vector $\boldsymbol{\pi}^k$ such that the optimality parameter is equal to the most negative reduced cost, i.e., $\mu^k = \min_{(i,j) \in A(\mathbf{x}^k)} \bar{d}_{ij}^k$. Moreover all arcs of the identified cycle W^k share that same value, i.e., $\bar{d}_{ij}^k = \mu^k, \forall (i,j) \in W^k$.*

Proof. At iteration k, constraint set (2.7) can be written as $\mu \leq d_{ij} - \pi_i + \pi_j, \forall (i,j) \in A(\mathbf{x}^k)$. At optimality, $\mu^k \leq \bar{d}_{ij}^k, \forall (i,j) \in A(\mathbf{x}^k)$ and the objective function (2.6) pushes μ^k to the smallest reduced cost. Furthermore, the complementary slackness conditions guarantee that the equality holds in (2.7) for all $y_{ij}^k > 0$, that is, $\mu^k = \bar{d}_{ij}^k, \forall (i,j) \in W^k$. □

ϵ-optimality conditions. The complexity analysis is born out of a chain of arguments that is bound by a series of equivalences. Many strongly polynomial algorithms for CMCF use the concept of ϵ-optimality obtained by relaxing the complementary slackness constraints, see for example, Bertsekas (1979), Röck (1980), Tardos (1985), Fujishige (1986), Goldberg and Tarjan (1989), Radzik and Goldberg (1994). Ultimately, it turns out that parameters ϵ and μ are linked by an equality expression, that is, $\epsilon = -\mu$. We argue that it all comes together with the linear programming formulation of the pricing problem's dual. This line of thoughts allows us to discard the ϵ-parameter and is indeed the reason we cannot say with certainty it was understood as such. In the spirit of the coined expression, a μ-optimal solution $\mathbf{x}^k$ can be understood in the same way as its ϵ-counterpart, that is, relaxed complementary slackness conditions which provide approximate optimality, see Ahuja et al. (1993, relations

(10.1)–(10.2)). Feel free to compare (2.5) with the following relaxed conditions:

$$x_{ij}^k = \ell_{ij} \text{ if } \bar{c}_{ij} > -\mu; \qquad x_{ij}^k = u_{ij} \text{ if } \bar{c}_{ij} < \mu; \qquad \ell_{ij} \leq x_{ij}^k \leq u_{ij} \text{ if } \mu \leq \bar{c}_{ij} \leq -\mu. \tag{2.15}$$

It should come as no surprise that solution $\mathbf{x}^k$ is μ^k-optimal. The reader may want to verify that the equivalent condition on the residual network $G(\mathbf{x}^k)$ questions whether there exists $\boldsymbol{\pi}$ such that $\bar{d}_{ij} \geq \mu,\ \forall (i,j) \in A(\mathbf{x}^k)$.

The following propositions stand outside the scope of complexity theorems for several reasons. The first is that they are very strong results for MMCC. The second is that their validity is independent of any assumptions regarding problem data. The third is that we ascertain the comprehension of the transitive mechanics between two solutions.

Proposition 4. *(Goldberg and Tarjan 1989,* Lemma 3.5*) For any two consecutive iterations k and $k+1$, the value of μ is non-decreasing, that is, $\mu^k \leq \mu^{k+1},\ k \geq 0$.*

Proof. The proof consists of examining the effect of canceling cycle W^k in light of the new solution $\mathbf{x}^{k+1}$ and more specifically the marginal modifications incurred in $G(\mathbf{x}^{k+1})$. There are only four possibilities as displayed in Figure 2.3. First off, either the residual network $G(\mathbf{x}^k)$ contains arcs in both directions or only one between nodes i and j. Secondly, either the flow that passes on an arc of cycle W^k saturates it (arcs in bold) or not.

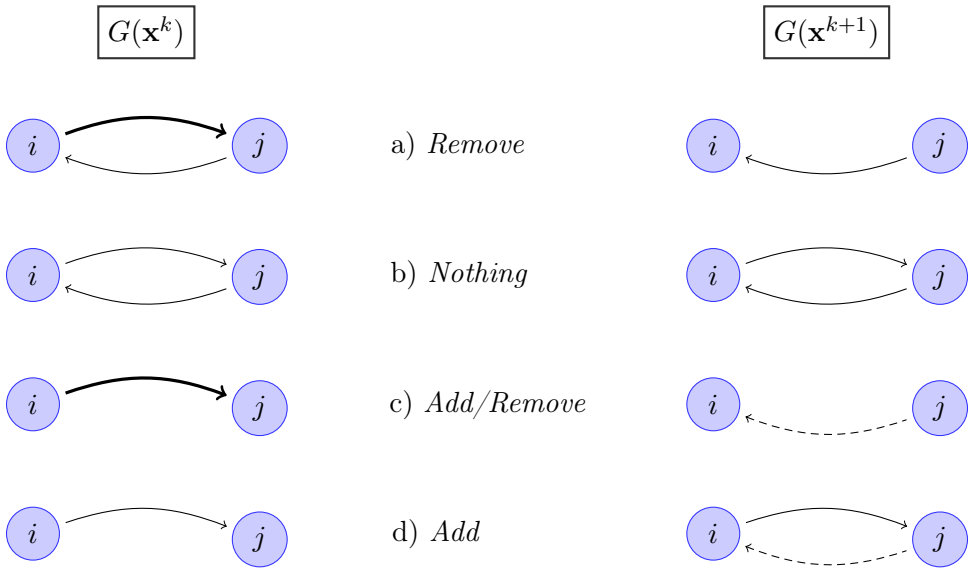


Fig. 2.3: Aftermath of cycle-canceling in the residual network

By Proposition 3, vector $\boldsymbol{\pi}^k$ ensures that $\bar{d}_{ij}^k \geq \mu^k$ in $G(\mathbf{x}^k)$ such that $\bar{d}_{ij}^k = \mu^k$, $\forall (i,j) \in W^k$. In $G(\mathbf{x}^{k+1})$, the saturated arcs in cycle W^k are removed and new arcs appear in the reverse direction with a reduced cost equal to $-\mu^k > 0$. Therefore, by construction, every arc of $G(\mathbf{x}^{k+1})$ has a reduced cost $\bar{d}_{ij}^k \geq \mu^k$ computed with respect to $\boldsymbol{\pi}^k$. Since the mean cost of a cycle is at least as great as the minimum cost of any of its terms, $\mu^{k+1} \geq \mu^k$. □

We take the time to stress the fact that MMCC is an iterative algorithm. Even though the previous proposition is true of any two consecutive iterations, the proof still uses a point of reference. The following proofs base their arguments on a sequence of iterations and it is imperative to take a step back to appreciate the global picture.

Proposition 4 is not sufficient to provide convergence properties. It is indeed mandatory for μ to strictly increase sporadically towards zero. Define a *jump* on the optimality parameter as the situation where there exists some factor $0 \leq \tau < 1$ such that $\mu^{k+1} \geq \tau \mu^k > \mu^k$, for $k \geq 0$. Recall Figure 2.2 which exhibits this behavior for the maximum flow problem.

Proposition 5. *(Goldberg and Tarjan 1989,* Lemma 3.6*) Given a non-optimal solution* $\mathbf{x}^k$*,* $k \geq 0$*, a sequence of no more than* m *iterations allows* μ *to jump by a factor of at least* $(1 - 1/n)$*.*

Proof. Recall that the cost and the reduced cost of a cycle take the same value. Hence, the reduced costs can be computed with any set of potentials. In order to show the statement, we use vector $\boldsymbol{\pi}^0$ found at iteration 0, hence $\bar{d}_{ij}^0 \geq \mu^0$, $\forall (i,j) \in A(\mathbf{x}^0)$. At iteration $k \geq 1$, we distinguish two types of cycles according to the following definitions. A cycle W^k of Type 1 contains only arcs of strictly negative reduced costs, i.e., $\bar{d}_{ij}^0 < 0$, $\forall (i,j) \in W^k$, while a cycle of Type 2 contains at least one arc with a non-negative reduced cost, i.e., $\exists (i,j) \in W^k \mid \bar{d}_{ij}^0 \geq 0$.

We prove that within m iterations, the algorithm finds a Type 2 cycle otherwise the optimal solution to (2.1) has been reached. The same reasoning used in Proposition 4 allows us to realize two things regarding Type 1 cycles. First, there is at least one saturated arc of strictly negative reduced cost that is removed in the next residual network. Second, reversed arcs added all have strictly positive reduced costs with respect to $\boldsymbol{\pi}^0$. Since there are no more than m arcs with strictly negative costs, optimality is reached after at most m consecutive Type 1 cancellations.

For $l \leq m$, assume a Type 2 cycle W^l is found, then at least one of its arcs has a non-negative reduced cost. The worst case scenario in terms of mean reduced cost is to pass through $|W^l| - 1 \leq n - 1$ arcs of cost μ^0 and one of zero. As such, $\mu^l \geq \dfrac{(|W^l| - 1)\, \mu^0}{|W^l|} \geq \dfrac{(n-1)\, \mu^0}{n} = \left(1 - \dfrac{1}{n}\right) \mu^0$. □

Proposition 6. *Given a non-optimal solution* $\mathbf{x}^k$, $k \geq 0$, *a sequence of no more than* mn *iterations allows* μ *to jump by a factor of at least* $1/2$.

Proof. Basic calculus shows that $(1 - 1/n)^n < 1/2$, $\forall n \geq 2$ as it converges to $1/e$. This means that every mn iterations, the value of μ increases by a factor of at least $1/2$. □

2.3.2 Integer costs: $O(n \log(nC))$ phases

This section contains the first installment regarding the actual complexity of MMCC. The only assumption is that all cost data are integers. Recall that $C := \max_{(i,j)\in A} |c_{ij}|$.

Theorem 1. *(Goldberg and Tarjan 1989,* Theorem 3.7*) Given a capacitated network with integer arc costs, MMCC performs* $O(mn \log(nC))$ *iterations.*

Proof. Let us consider the values of μ obtained for each iteration as a sequence. This sequence may be bounded below because no mean cycle can cost less than $-C$. It is also bounded above by the highest strictly negative reduced cost cycle computed as $\frac{(-1)+(n-1)0}{n} = -1/n$. In light of Proposition 6, it is possible to construct a geometric progression of reason at least $1/2$ with some elements of the sequence $\{\mu^k\}$. As it stands, solving for the power of $-C\left(\frac{1}{2}\right)^\kappa = -1/n$, the geometric system that prevails every mn iterations, one obtains $\kappa = \log(nC)$. Therefore, the image of the objective function can always be traversed in $O(mn \log(nC))$ iterations. □

Markers. Let us revisit Proposition 5 in order to extract what, in retrospect, seems like a key property. Indeed, Goldberg and Tarjan (1989) use a Type 2 cancellation as a marker for the jump factor on the optimality parameter μ. We stress that while it is an elegant (read sufficient) condition, it is certainly not necessary. Type 1 cancellations can indeed run into jumps however we still do not know how to measure them. Type 2 cancellations are thus markers for *measurable* jumps. It is probably what prompts Radzik and Goldberg (1994) to define a phase with a definition that is much closer to the spirit of Proposition 5.

Definition 1. *A* phase *is a sequence of iterations terminated by a Type 2 cycle. A* phase solution $\mathbf{x}^h$, $h \geq 0$, *is understood as the solution at the beginning of phase* h.

We stress that while the two phase numbers h and $h+1$ are consecutive, by Proposition 5, the number of cycles canceled within phase h is at most m. Let $l \leq m$ be that length. For the remainder of this article, the notations k and h are respectively reserved for iteration

and phase based operations. The notation l is used to refer to the last cycle identified in the phase, that is, the Type 2 cycle.

Proposition 7. *Given a non-optimal phase solution* $\mathbf{x}^h$, $h \geq 0$, *the optimality parameter obtained on the following phase solution strictly increases by a factor of least* $(1 - 1/n)$ *and increases by a factor of at least* $1/2$ *every* n *phases.*

Proof. The proof is immediate from the definition of a phase. Indeed, a Type 2 cycle implies a measurable jump on μ and means that the relation of the optimality parameter between two consecutive phases is $\mu^{h+1} \geq (1 - 1/n)\,\mu^h > \mu^h$. Moreover, $\mu^{h+n} \geq (1 - 1/n)^n\,\mu^h > \frac{1}{2}\,\mu^h$. □

Assuming a mechanism that allows the resolution to terminate a phase under the same conditions, we can make abstraction of the Type 1 iterations and express the outer loop in terms of phases instead of iterations. For instance, the weakly polynomial complexity of $O(mn \log(nC))$ iterations obtained in Theorem 1 can be rewritten as $O(n \log(nC))$ phases. While one might argue that this change is purely cosmetic, Section 2.3.5 holds the key to the justification. In a nutshell, the implementation choice dictates the actual complexity depending on whether a phase is solved in an integrated manner or not.

Although the following idea is already present in the proof of Proposition 5, we feel it warrants a repetition. The same node potentials $\boldsymbol{\pi}^h$ can be used throughout the whole phase. As such, in the adaptation to a phase-wise analysis, the reduced costs of each arc is also unchanged during the phase. Consider a sequence of iterations consisting of consecutive Type 1 cycles starting from phase solution $\mathbf{x}^h$. Solving the pricing step at every iteration implicitly associates the minimum mean cycle found with a tight vector of node potentials along with the optimality parameter in accordance with the primal-dual formulations. However, by definition the very first vector of node potentials, $\boldsymbol{\pi}^h$, validates the Type 1 condition of the cancellations throughout the series of residual networks traversed. In other words, while $\boldsymbol{\pi}^h$ might not be the tightest vector of node potentials for every residual networks associated with this sequence, it is sufficient to allow the identification of these cycles. Let it be clear that the existence of this sustainable vector $\boldsymbol{\pi}^h$, valid until a new vector $\boldsymbol{\pi}^{h+1}$ is required, is irrelevant to the solving process of MMCC. What is important to retain is that the node potentials can have a lasting effect of at most m iterations, the maximal length of a phase during the resolution process.

2.3.3 Arbitrary costs: $O(mn \log n)$ phases

So far, we have shown that MMCC runs in weakly polynomial time from two different perspectives: the strictly decreasing objective function and the sequence of strictly increasing μ^h which can, in some way, be interpreted as a subsequence of $\{\mu^k\}$. In order to speak of strongly polynomial time, a new angle is required. That angle is named *arc fixing*. The idea is to tag an arc as being fixed at one of its bounds. In line with the complementary slackness optimality conditions (2.5), this tagging occurs when the reduced cost associated with an arc is sufficiently far from zero, that is, positively large enough for fixing a variable at its lower bound or negatively small enough for fixing it at its upper bound.

The following proposition states the arc fixing rule. The argument is based on the logical implication of the complementary slackness conditions (2.5) to create a correspondence between the current value of the optimality parameter μ^k and the flow value of certain arcs.

Proposition 8. *(Goldberg and Tarjan 1989,* Theorem 3.8*) Let $k \geq 0$ denote a non-optimal iteration number. Arc fixing occurs for arc $(i,j) \in A$ if and when $|\bar{c}_{ij}^k| \geq -2n\mu^k$. Expressed in terms of $G(\mathbf{x}^k)$, we have: arc $(i,j) \in A(\mathbf{x}^k)$ is fixed at zero if and when $\bar{d}_{ij}^k \geq -2n\mu^k$.*

Proof. The proof establishes a contradiction with the previous properties when a fixed arc is used. Assume, without loss of generality, that arc $(i,j) \in A$ has reached $\bar{c}_{ij}^k \geq -2n\mu^k$ at iteration k. According to Proposition 3, $A(\mathbf{x}^k)$ must contain arc (i,j) and not (j,i) because its reduced cost would be less than μ^k, which means that $x_{ij}^k = \ell_{ij}$ and $\bar{d}_{ij}^k = \bar{c}_{ij}^k \geq -2n\mu^k$.

Assume $\mathbf{x}^s$, $s > k$, values $x_{ij}^s > 0$. By Proposition 2, $\mathbf{x}^s$ equals $\mathbf{x}^k$ plus the flow on at most m directed cycles in $G(\mathbf{x}^k)$, and *vice versa*. Hence, there exists a cycle in $G(\mathbf{x}^k)$, say W^+, using arc (i,j). The mean reduced cost of this cycle, denoted $\mu(W^+)$, could even be valued by

$$\mu(W^+) \geq \frac{-2n\mu^k + (|W^+| - 1)\,\mu^k}{|W^+|} \geq \frac{-2n\mu^k + (n-1)\,\mu^k}{|W^+|} = \frac{-(n+1)}{|W^+|}\mu^k.$$

The existence of cycle W^+ in $G(\mathbf{x}^k)$ means that the reverse cycle denoted W^- exists in $G(\mathbf{x}^s)$ with a mean reduced cost of $\mu(W^-) = -\mu(W^+) \leq \frac{n+1}{|W^+|}\mu^k < \mu^k$, which is a contradiction with Proposition 4 on the fact that the optimality parameter μ is non-decreasing.

The correspondence between the original and the residual arcs is subtle. Arc $(i,j) \in A$ is fixed at ℓ_{ij} because $\bar{d}_{ij}^k \geq -2n\mu^k$ such that arc $(i,j) \in A(\mathbf{x}^s)$, $s > k$ is fixed at zero. The proof for arc $(i,j) \in A$ being fixed at its upper bound u_{ij} when $\bar{c}_{ij}^k \leq 2n\mu^k$ or equivalently for arc $(j,i) \in A(\mathbf{x}^k)$ being fixed at 0 when $\bar{d}_{ji}^k \geq -2n\mu^k$ is similar. □

This proof assumes the non-decreasing property of the optimality parameter over the iterations which is lost when using the Cancel-and-Tighten strategy. A careful yet straightforward adaptation can be made using the following proposition.

Proposition 9. *Let $h \geq 0$ denote a non-optimal phase number. Arc fixing occurs for arc $(i,j) \in A$ if and when $|\bar{c}_{ij}^h| \geq -2n\mu^h$. Expressed in terms of $G(\mathbf{x}^h)$, we have: arc $(i,j) \in A(\mathbf{x}^h)$ is fixed at zero if and when $\bar{d}_{ij}^h \geq -2n\mu^h$.*

This proposition holds exactly the same value for the complexity analysis because the second installment of the theoretical complexity only uses phases to capture the concept of arc fixing. Let us define a *block* accordingly.

Definition 2. *A* block *is a sequence of phases terminated by the fixing of at least one arc. A* block solution *is understood as the solution at the beginning of a block.*

Theorem 2. *(Goldberg and Tarjan 1989,* Theorem 3.9*) Given a capacitated network with arbitrary real-valued arc costs, MMCC performs $O(mn \log n)$ phases.*

Proof. This proof relies on the concept of arc fixing. The idea is to show that at least one new arc is fixed within a limited number of phases. We show this bound to be $n(\lceil \log n \rceil + 1) \equiv O(n \log n)$ phases. As such, consider this particular sequence of phases as a block.

For phase number h, let $\mathbf{x}^\circ = \mathbf{x}^h$ and $\mathbf{x}^\bullet = \mathbf{x}^{h+n(\lceil \log n \rceil + 1)}$ be respectively the solutions prior to the first and after the last iteration of any given block. By Proposition 7, we know μ increases by a factor of at least $1/2$ every n phases, so the increase in a block is

$$\mu^\bullet = \mu^{\circ + n(\lceil \log n \rceil + 1)} \geq \frac{1}{2^{\lceil \log n \rceil + 1}} \mu^\circ \geq \frac{1}{2n} \mu^\circ.$$

Found on $G(\mathbf{x}^\circ)$, consider cycle W° with mean reduced cost μ°, a value independent of the potentials used. Therefore, with respect to $\boldsymbol{\pi}^\bullet$, the arc reduced costs in W° cannot all be greater than μ°. Hence, there exists a variable, say $y_{ji}, (j,i) \in W^\circ$, with a reduced cost $\bar{d}_{ji}^\bullet$ at most equal to μ°, that is, $\bar{d}_{ji}^\bullet \leq \mu^\circ \leq 2n\mu^\bullet$. On $G(\mathbf{x}^\bullet)$, variable y_{ij} appears with a reduced cost of $\bar{d}_{ij}^\bullet = -\bar{d}_{ji}^\bullet \geq -2n\mu^\bullet$. By Proposition 8, the value of variable y_{ij} does not change anymore and the corresponding variable x_{ij} is fixed at its lower bound. Moreover, as part of the optimal cycle W°, the algorithm modifies the value of x_{ij} in the very first iteration of the block. In retrospective, arc $(j,i) \in G(\mathbf{x}^\circ)$ must have saturated the residual capacity and it is quite interesting to note that the confirmation of the flow value comes later than the time at which it is established. The proof for arc fixing at upper bound is a straightforward adaptation.

Let it be clear that $\mathbf{x}^\bullet$ is a block solution in that it becomes $\mathbf{x}^\circ$ in the following block. All in all, each block fixes a different arc. Since there are m arcs, the proposed complexity is achieved. □

2.3.4 Arbitrary costs: $O(mn)$ phases

The final piece of the complexity puzzle comes much later than the seminal paper of Goldberg and Tarjan (1989) which elaborates all the propositions we have seen thus far. Although they utilize properties of the complementary slackness conditions (2.5) at every iteration, Radzik and Goldberg (1994) make use of the properties of an optimal primal-dual pair in their complexity analysis, establishing the best possible strongly polynomial result for the minimum mean cycle-canceling algorithm.

Radzik and Goldberg (1994) obviously have such a good understanding of MMCC that it really allows them to think outside the box. While an optimal vector of node potentials $\boldsymbol{\pi}^*$ is unknown, we can use the fact that it does exist. Denote the reduced costs computed using such a set of optimal potentials by $\bar{c}^*_{ij} := c_{ij} - \pi^*_i + \pi^*_j,\ \forall (i,j) \in A$.

Proposition 10. *(Radzik and Goldberg 1994, Lemma 9) Let $k \geq 0$ denote a non-optimal iteration number. Implicit arc fixing occurs for arc $(i,j) \in A$ if and when $|\bar{c}^*_{ij}| > -n\mu^k$. Expressed in terms of $G(\mathbf{x}^k)$, we have: arc $(i,j) \in A(\mathbf{x}^k)$ is implicitly fixed at zero if and when $\bar{d}^*_{ij} > -n\mu^k$.*

Proof. If $\bar{c}^*_{ij} > -n\mu^k > 0$, then $x^*_{ij} = \ell_{ij}$ by the complementary slackness optimality conditions (2.5). Now assume arc $(i,j) \in A$ has reached $\bar{c}^*_{ij} > -n\mu^k$ at iteration k but $x^k_{ij} > \ell_{ij}$. By Proposition 2, $\mathbf{x}^*$ equals $\mathbf{x}^k$ plus the flow on at most m directed cycles in $G(\mathbf{x}^k)$, and *vice versa*. Hence there exists a cycle W^+ in $G(\mathbf{x}^k)$ using variable y_{ji} to push the flow back towards ℓ_{ij}. The reverse cycle W^- exists in $G(\mathbf{x}^*)$ using arc (i,j) with $\bar{d}^*_{ij} = \bar{c}^*_{ij} > -n\mu^k$. Because optimal arc reduced costs on $G(\mathbf{x}^*)$ are greater than or equal to zero, $\mu(W^-) > \frac{-n\mu^k}{|W^-|}$. Therefore $\mu(W^+) = -\mu(W^-) < \frac{n\mu^k}{|W^-|} < \mu^k$, a contradiction on the optimality of μ^k at iteration k. The proof for an arc $(i,j) \in A$ being implicitly fixed at its upper bound u_{ij} when $\bar{c}^*_{ij} < n\mu^k$ is similar. □

Once again, the following analysis still revolves around phases. Using the Cancel-and-Tighten strategy modifies the statement of the previous proposition without compromising its value. The comparison must be done against the optimality parameter computed at each phase h.

Proposition 11. *Let $h \geq 0$ denote a non-optimal phase number. Implicit arc fixing occurs for arc $(i,j) \in A$ if and when $|\bar{c}^*_{ij}| > -n\mu^h$. Expressed in terms of $G(\mathbf{x}^h)$, we have: arc $(i,j) \in A(\mathbf{x}^h)$ is implicitly fixed at zero if and when $\bar{d}^*_{ij} > -n\mu^h$.*

The third and last installment of the complexity analysis is at hand. The avid reader might even recall the introductory complexity proposition. With a statement so closely matching that of Theorem 2, the expectation of an analogous proof is annihilated from the start. While the previous proof bounds the length of each block against a uniquely defined value, the following propositions show that the more subtle kind of implicit arc fixing of Propositions 10 and 11 happens in a unpredictable manner. Nevertheless, Radzik and Goldberg (1994) are able to prove by a global analysis that the whole implicit arc fixing process is itself bounded. The first step towards this result is to bind the number of phases to the number of arcs contained in the cycles identified within these phases. The latter in turn grants a tighter measurable jump factor for the optimality parameter according to the following observation.

Tighter jump factor. Given a non-optimal phase solution $\mathbf{x}^h$, $h \geq 0$, the impact of the jump factor for a phase of length $l \leq m$ can be tightened by the size of the Type 2 cycle W^l. This is evaluated by

$$\mu^{h+1} \geq \left(1 - \frac{1}{|W^l|}\right)\mu^h \geq \left(1 - \frac{1}{n}\right)\mu^h. \tag{2.16}$$

The proof of Proposition 5 can be trivially modified to accommodate this observation by using, for the Type 2 cycle, $|W^l|-1$ instead of $n-1$. Yet, this new tighter jump factor strongly endorses the empirical behavior of cycle sizes much smaller than n. The parameter, $L(\mathbf{x}^h)$, used to bind these values together emerges from the following construction.

Let $e \in A$ denote an arc of network G with optimal reduced cost $\bar{c}^*_e$. For the purpose of the complexity analysis, the arcs are sorted in decreasing order of their *absolute optimal reduced cost* values, that is, $0 \leq |\bar{c}^*_m| \leq \ldots \leq |\bar{c}^*_2| \leq |\bar{c}^*_1|$. While the notation is introduced hereafter, the rightmost part of Figure 2.4 visually explains the sort.

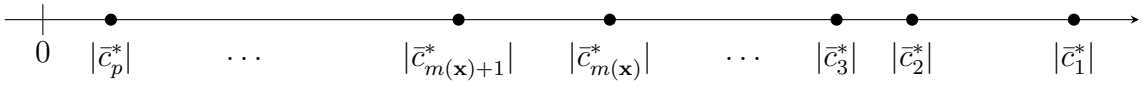


Fig. 2.4: Optimal absolute arc reduced costs on the axis

For a given solution $\mathbf{x}$, let $m(\mathbf{x}) := \min\{e \in A \ : \ |\bar{c}^*_e| \leq -n\mu(\mathbf{x})\}$ be the smallest index for which arcs $e \geq m(\mathbf{x})$ have not yet been implicitly fixed and $S_{m(\mathbf{x})} := \sum_{e=m(\mathbf{x})}^{m} |\bar{c}^*_e|$ be the sum of their absolute reduced costs.

As optimality parameter μ increases towards zero, $|\bar{c}^*_1| > -n\mu$ and arc $e = 1$ is the first arc implicitly fixed. Next implicit fixing occurs for arc $e = 2$, and so forth for the remaining arcs. Arc variable x_e with an index value $e < m(\mathbf{x})$ cannot be part of an improving cycle for a non-optimal solution $\mathbf{x}$ because, by Proposition 10, its value does not change anymore. Therefore, index $m(\mathbf{x})$ increases towards m whereas largest absolute reduced cost $|\bar{c}^*_{m(\mathbf{x})}|$ and sum $S_{m(\mathbf{x})}$ decrease towards zero. When $S_{m(\mathbf{x})} = 0$, every variable with an optimal reduced cost different from zero has been implicitly fixed while any free variable lies within its interval domain with a zero-value optimal reduced cost, hence satisfying the necessary and sufficient complementary slackness optimality conditions (2.5). As such, let $p \leq m$ denote the largest arc index for which the optimal absolute reduced cost is strictly positive. The latter is the last arc variable which can answer to the implicit arc fixing rule. Observe that this means that $S_{m(\mathbf{x})} := \sum_{e=m(\mathbf{x})}^{m} |\bar{c}^*_e| = \sum_{e=m(\mathbf{x})}^{p} |\bar{c}^*_e|$.

The complexity analysis is based on ratio value $\frac{S_{m(\mathbf{x})}}{|LB_\mu|}$ computing, for the unfixed variables, the sum of their absolute reduced cost values over $|LB_\mu|$, where LB_μ is a lower bound on $\mu(\mathbf{x})$. Given a non-optimal phase solution $\mathbf{x}^h, h \geq 0$, a lower bound on the optimality parameter is $LB_\mu = -|\bar{c}^*_{m(\mathbf{x}^h)}|$. The next proposition determines how many phases are required to increase this lower bound by a factor greater than $1/2$. The same question is then answered for $1/n$ in the following proposition. In both cases, the number of phases required to reach their respective jumps is established as a function of ratio value $\frac{S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|}$.

Proposition 12. *(Radzik and Goldberg 1994, Lemma 12) Let $h \geq 0$ denote a non-optimal phase number. A sequence of no more than $L(\mathbf{x}^h) := \min\left\{\left\lfloor \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|} \right\rfloor, n\right\}$ phases allows μ to jump by a factor over $1/2$.*

Proof. For non-optimal phase solution $\mathbf{x}^h$, we have $\mu(\mathbf{x}^h) \geq -|\bar{c}^*_{m(\mathbf{x}^h)}|$. Suppose some Type 2 cycle W^l at the end of phase $l \leq L(\mathbf{x}^h)$ has cardinality $|W^l| > \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|}$. Then,

$$\mu^{L(\mathbf{x}^h)} \geq \mu^l \geq \frac{-S_{m(\mathbf{x}^h)}}{|W^l|} > \frac{-S_{m(\mathbf{x}^h)}}{\frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|}} = \frac{-|\bar{c}^*_{m(\mathbf{x}^h)}|}{2}.$$

Otherwise, all phase cycles in these $L(\mathbf{x}^h)$ phases have cardinalities at most $L(\mathbf{x}^h)$ arcs. Therefore, by (2.16),

$$\mu^{L(\mathbf{x}^h)} \geq \mu(\mathbf{x}^h)\left(1 - \frac{1}{L(\mathbf{x}^h)}\right)^{L(\mathbf{x}^h)} \geq -|\bar{c}^*_{m(\mathbf{x}^h)}|\left(1 - \frac{1}{L(\mathbf{x}^h)}\right)^{L(\mathbf{x}^h)} > \frac{-|\bar{c}^*_{m(\mathbf{x}^h)}|}{2}. \qquad \square$$

Proposition 13. *(Radzik and Goldberg 1994, Lemma 13) Let $h \geq 0$ denote a non-optimal phase number. A sequence of no more than $O\left(\frac{nS_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|}\right)$ phases allows μ to jump by a factor over $1/n$.*

Proof. For non-optimal phase solution $\mathbf{x}^h$, we have $\mu(\mathbf{x}^h) \geq -|\bar{c}^*_{m(\mathbf{x}^h)}|$. As long as variable $x_{m(\mathbf{x}^h)}$ is not fixed, $S_{m(\mathbf{x}^h)}$ remains the same. By Proposition 12, we have $\mu > \frac{-|\bar{c}^*_{m(\mathbf{x}^h)}|}{2}$ after $\left\lfloor \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|} \right\rfloor$ phases, $\mu > \frac{-|\bar{c}^*_{m(\mathbf{x}^h)}|}{4}$ after $\left\lfloor \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|/2} \right\rfloor$ new phases, $\mu > \frac{-|\bar{c}^*_{m(\mathbf{x}^h)}|}{8}$ after $\left\lfloor \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|/4} \right\rfloor$ additional phases, and so on for the following steps increasing each time the lower bound by a factor greater than one half. The number of steps t such that $2^t \geq n$ is $t = \lceil \log n \rceil$, so that the total number of phases during these t steps is given by

$$\begin{aligned} \sum_{t=0}^{\lceil \log n \rceil - 1} \left\lfloor \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|/2^t} \right\rfloor &\leq \lceil \log n \rceil + \sum_{t=0}^{\lceil \log n \rceil - 1} \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|/2^t} \leq \lceil \log n \rceil + \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|} \sum_{t=0}^{\lceil \log n \rceil - 1} 2^t \\ &= \lceil \log n \rceil + \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|} \left(2^{\lceil \log n \rceil} - 1\right) \leq \lceil \log n \rceil + \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|}(2n-1). \end{aligned} \tag{2.17}$$

Hence, the number of phases needed to increase $\mu(\mathbf{x}^h)$ by a factor of over $1/n$ is $O\left(\frac{nS_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|}\right)$. □

The following theorem brings all these elements together. The idea is that it does not require the same amount of phases for each variable to be implicitly fixed. For some variables, the jump provided by Proposition 12 is sufficient while others must wait for that of Proposition 13.

Theorem 3. *(Radzik and Goldberg 1994, Theorem 1) Given a capacitated network with arbitrary real-valued arc costs, MMCC performs $O(mn)$ phases.*

Proof. Starting with x_1 followed by $x_2, \ldots, x_p$, these variables are (implicitly) fixed one by one as μ increases. The fixing of $x_e, 1 \leq e \leq p$, the yet unfixed variable with the largest absolute reduced cost value, to its lower or upper bound is done with Proposition 11. Observe that until x_{e+1} is fixed, $S_e = |\bar{c}^*_p| + \cdots + |\bar{c}^*_{e+2}| + |\bar{c}^*_{e+1}| + |\bar{c}^*_e|$ does not change.

The fixing of x_e is *fast* if $\frac{|\bar{c}^*_{e-1}|}{2} \leq |\bar{c}^*_e| \leq |\bar{c}^*_{e-1}|$, that is, the successive absolute reduced cost values are relatively close to each other. Otherwise, the fixing of x_e is *slow*.

By Proposition 12, for every variable x_e for which the fixing is fast, the number of phases to do so is at most n. Thus, the total number of phases of fast fixing for at most $p \leq m$ variables is $O(pn) \equiv O(mn)$.

Regarding the slow fixing process of a variable x_{e+1}, we have $|\bar{c}^*_{e+1}| < \frac{|\bar{c}^*_e|}{2}$, hence $\frac{|\bar{c}^*_{e+1}|}{|\bar{c}^*_e|} < \frac{1}{2}$. If the fixing process of x_{e+2} is also slow, we have $\frac{|\bar{c}^*_{e+2}|}{|\bar{c}^*_{e+1}|} < \frac{1}{2}$, hence $\frac{|\bar{c}^*_{e+2}|}{|\bar{c}^*_e|} = \frac{|\bar{c}^*_{e+2}|}{|\bar{c}^*_{e+1}|} \times \frac{|\bar{c}^*_{e+1}|}{|\bar{c}^*_e|} < \frac{1}{2^2}$. In the worst case, the fixing process is slow for at most p variables. By Proposition 13, the number of phases is bounded above by

$$\begin{aligned}
\sum_{e=1}^{p} O\left(\frac{nS_e}{|\bar{c}^*_e|}\right) &= O\left(n\sum_{e=1}^{p}\frac{S_e}{|\bar{c}^*_e|}\right) = O\left(n\sum_{e=1}^{p}\frac{|\bar{c}^*_e| + |\bar{c}^*_{e+1}| + |\bar{c}^*_{e+2}| + \ldots + |\bar{c}^*_p|}{|\bar{c}^*_e|}\right) \\
&= O\left(n\sum_{e=1}^{p}\left(1 + \frac{|\bar{c}^*_{e+1}|}{|\bar{c}^*_e|} + \frac{|\bar{c}^*_{e+2}|}{|\bar{c}^*_e|} + \ldots + \frac{|\bar{c}^*_p|}{|\bar{c}^*_e|}\right)\right) \\
&< O\left(n\sum_{e=1}^{p}\left(1 + \frac{1}{2} + \frac{1}{2^2} + \ldots + \frac{1}{2^{p-e}}\right)\right) \\
&< O\left(n\sum_{e=1}^{p}(1+1)\right) = O(pn) \equiv O(mn).
\end{aligned} \tag{2.18}$$

Whether it is fast or slow, the total implicit fixing process takes $O(mn)$ phases. □

Remark. The beauty of the slow analysis is that while the current candidate variable, x_e, may take longer to fix, it also implies that the optimal reduced cost distribution of the remaining non-fixed variables is skewed towards zero exponentially faster than the current candidate's optimal reduced cost. In other words, the higher number of phases required to implicitly fix x_e is eventually amortized by the much faster implicit fixing of the remaining $x_{e+1}, \ldots, x_p$ variables.

Radzik and Goldberg (1994) also show this bound is tight by using certain minimum cost flow examples that behave as bad as the worst case complexity would have it. This means that the absolute reduced cost spread is such that arcs are implicitly fixed precisely at the bounds computed previously.

2.3.5 Bottleneck management

Although we have presented the complexity analysis in terms of phases, each of these can be seen as a group of iterations. As such, this section is separated in two parts. The first is reserved to the actual resolution of the pricing step while the second considers the phase as a whole, that is, using the Cancel-and-Tighten strategy. Several plots are presented to help grasp some of the ideas as well as appreciate the empirical behavior on a relatively large

capacitated network flow problem comprising 1,025 nodes and 91,220 arcs. We refer to the latter as Instance 1.

Iteration base

Solving the pricing step at every iteration corresponds to the traditional form of MMCC. Figure 2.5 depicts the behavior of $\mu^k, k \geq 0$, on Instance 1: 1,937 iterations are performed to reach optimality at $\mu = 0$. As expected, values of μ are non-decreasing satisfying Proposition 4. In Figure 2.6, we see that the optimal cycle size $|W|$ however exhibits no pattern, a behavior quite different from the one observed for the maximum flow problem in Figure 2.2.

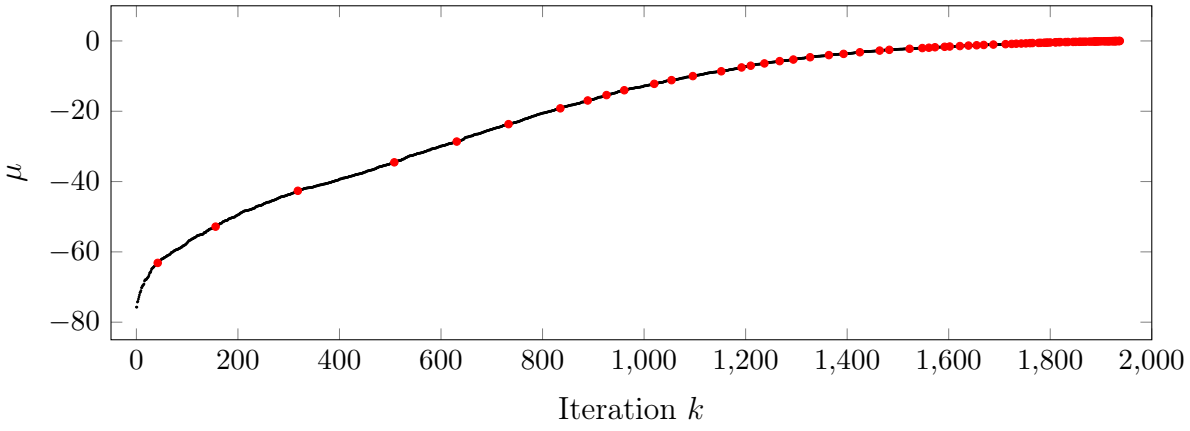


Fig. 2.5: Optimality parameter μ for Instance 1 [iteration base]

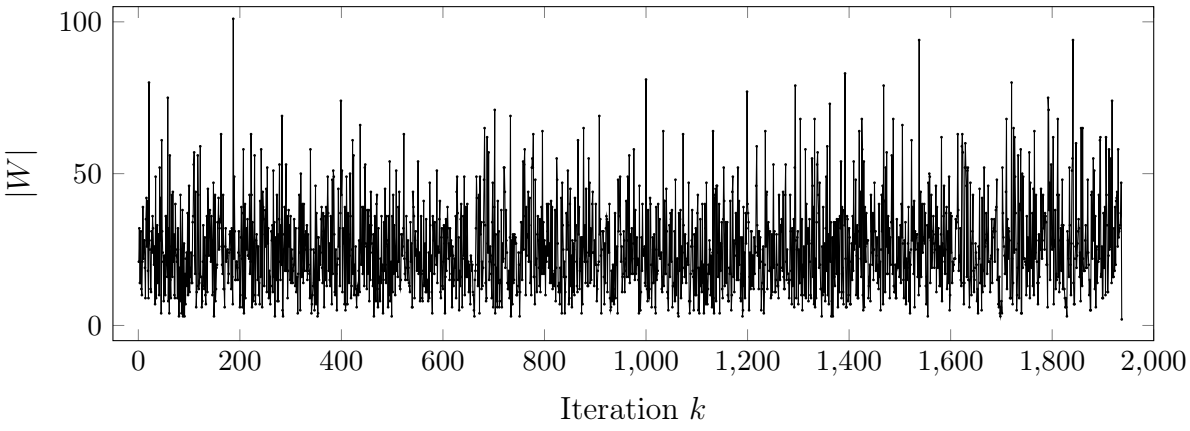


Fig. 2.6: Cycle size $|W|$ for Instance 1 [iteration base]

For the purpose of the complexity analysis, the dynamic programming approach devised by Karp (1978) runs in $O(mn)$ time. The proof of complexity is actually more simple than its

design. While it is true that the first layer of the algorithm is based on dynamic programming, finding the optimal solution μ requires an additional layer of computations which break an important feature of the strategy matrix, that is, the knowledge of optimal strategies for each action state. If that is not enough, extracting the associated optimal cycle requires additional computations.

We also state without proof that it is a best-case complexity making the resolution of the pricing step systematically expensive. From a practical point of view, a word of caution is therefore in order. In fact, this problem is at the heart of many industrial challenges, see Dasdan et al. (1999), Dasdan (2004), or Georgiadis et al. (2009) for in-depth experimental analysis of different algorithms. Ahuja et al. (1993, Chapter 5) also review several of them. While practice shows that the two algorithms of Howard (1960) (as specialized by Cochet-Terrasson et al. (1998)) and Young et al. (1991) are top performers, their theoretical complexities are higher than that of Karp's.

Theorem 4. *Goldberg and Tarjan (1989, Theorem 3.10) and Radzik and Goldberg (1994). MMCC runs in $O(m^2n^2 \log n \log(nC))$ time for integer arc costs and $O(m^3n^2)$ time for arbitrary real-valued arc costs.*

Proof. The proof is immediate from the runtime complexity $O(mn)$ of each iteration combined with either Theorem 1 or Theorem 3 depending on the data type. Note that the theorems must first be translated back to iteration results. □

Regardless of the algorithm selected to solve the pricing step, it is still no match to the better design of Cancel-and-Tighten. In order to support this claim, we argue that the performance of the iteration based algorithm is vulnerable to the starting solution. We have launched the resolution process using $\mathbf{x}^0$ as the optimal solution of the maximization problem. The reader is now invited to consider the markings on Figure 2.5. These markings indicate the starting point of each phase. Both cases traverse roughly the same number of phases, namely 90 and 87. The second launch actually requires 31,231 iterations to reach optimality at $\mu = 0$, yet the first 4 phases contain over 90% of the total iterations. Let us move on to a more integrated approach.

Phase base

We have already underlined the importance of the node potentials $\boldsymbol{\pi}^h$ established at the beginning of phase h for their ability to determine several Type 1 cycles. To appreciate

the validity of alternative strategies, we underline that the complexity analysis addresses only the behavior of the optimality parameter; the actual path of resolution along with the corresponding primal solutions are irrelevant. There is in fact absolutely no reason for any two implementations to reach the same phase solution. From this rationale emanates what is fundamentally important in the complexity analysis: the phases. The importance of Proposition 4, the non-decreasing optimality parameter μ, is discarded along with the order of cancellation in favor of the strict increase from one phase to the next. This means that any strategy falling under the premises of a phase can benefit from the outer loop complexity analysis. The remainder of this section is dedicated to one such approach, namely *Cancel-and-Tighten* (Goldberg and Tarjan 1989, Section 4). Of course, the latter is subtle by nature as it aims to analyze the complexity of the whole phase using an efficient system carefully designed to ensure a phase is indeed delivered.

The name of Cancel-and-Tighten is self-explanatory of its two main steps: the cycle cancellations and the adjustment of node potentials. This strategy shines by the way these two steps are carried out. Recall that a phase is a group of sequential iterations defined by consecutive Type 1 cycles and terminated by a Type 2 cycle. The idea of the first step defers to the first part whereby only cycles of Type 1 are canceled. While the cliffhanger is not intentional, the repercussion of the Type 2 cycle is handled in the second step which refines the node potentials using the measurable jump property. Since the Cancel step focuses on Type 1 cycles, the idea of working on a subgraph that does the same is quite natural. Before moving to the technical aspects of the steps, let us describe the so-called *admissible* network.

Admissible network. Given $\boldsymbol{\pi}$, the nature of Type 1 cycles is to contain only negative reduced cost arcs with respect to these node potentials. Let us define the admissible network with respect to solution $\mathbf{x}$ accordingly: $G(\mathbf{x}, \boldsymbol{\pi}) := (N, A(\mathbf{x}, \boldsymbol{\pi}))$, where $A(\mathbf{x}, \boldsymbol{\pi}) := \{(i, j) \in A(\mathbf{x}) \mid \bar{d}_{ij} < 0\}$, that is, a residual arc is *admissible* if its reduced cost is strictly negative.

Cancel step. Let $h \geq 0$ denote a non-optimal phase number, $\mathbf{x}^h$ the solution at the beginning of the phase, and $\boldsymbol{\pi}$ some dual vector. By definition of the admissible network $G(\mathbf{x}^h, \boldsymbol{\pi})$, any and all cycles it contains are of Type 1. This means that by sequentially eliminating at most m Type 1 cycles, however arbitrary the order, one reaches some solution $\mathbf{x}$ which can be substantially different than the input $\mathbf{x}^h$. As the Cancel step progresses, the content of the admissible network becomes difficult to describe in mathematical terms because the notation looses track of the current solution. Nevertheless, recall that the reduced cost of every arc stays the same during the whole phase regardless of the cycles canceled because

the node potentials are fixed to $\boldsymbol{\pi}$ throughout the step. In other words, as Type 1 cycles are canceled, only the residual capacities are modified and the admissible network is updated accordingly. The update is actually simpler to carry out than in the residual network. Indeed, by definition of admissibility, an arc and its reverse cannot be admissible simultaneously. This also means that the admissible network $G(\mathbf{x}, \boldsymbol{\pi})$ gets sparser because at least one new arc is saturated each time a Type 1 cycle is canceled.

Regardless of how the Cancel step is performed, one eventually reaches a solution $\mathbf{x}^{h+1}$ such that $G(\mathbf{x}^{h+1}, \boldsymbol{\pi})$ is acyclic. Of course, we have yet to prove optimality. In fact, we have yet to actually terminate the phase since the Type 2 cycle is still unconsidered. Let us see how this last operation is handled in the Tighten step.

Tighten step. Assume optimal values $[\boldsymbol{\pi}^h, \mu^h]$ are known and that the admissible network $G(\mathbf{x}^{h+1}, \boldsymbol{\pi}^h)$ is acyclic. By definition of a phase, we know the would be following iteration, say l, induces a Type 2 cycle W^l in the residual network $G(\mathbf{x}^{h+1})$. By Proposition 3, we also know that there exists some optimal vector of node potentials $\boldsymbol{\pi}^l$ such that all arcs on this cycle have the same negative reduced cost evaluated at μ^l. Observe that W^l is a Type 2 cycle with respect to selected $\boldsymbol{\pi} = \boldsymbol{\pi}^h$ but, by considering a new phase and modifying the node potentials to $\boldsymbol{\pi}^l$, this same cycle is a Type 1 cycle in $G(\mathbf{x}^{h+1}, \boldsymbol{\pi}^l)$. Let us rephrase this. The admissible network $G(\mathbf{x}^{h+1}, \boldsymbol{\pi})$ is defined by the current solution $\mathbf{x}^{h+1}$ and the reduced costs which are themselves defined by the selected vector $\boldsymbol{\pi}$ of node potentials. A solution $\mathbf{x}^{h+1}$ can induce different admissible networks depending on the selection of $\boldsymbol{\pi}$. The latter therefore sits at the top of the chain of command.

The Tighten step can be seen as the last operation that must be completed to start a new phase. Solving the pricing step effectively fetches the wanted information, that is, the best μ^{h+1} along with an optimal dual vector $\boldsymbol{\pi}^{h+1}$ which can be used for the duration of the next phase. This can of course be done with the dynamic programming approach of Karp (1978). Figure 2.7 shows the result of the experiment we have carried on Instance 1. The resolution process requires 88 phases. Even though the node potentials are updated under one hundred times, the truth of the matter is that this approach is still too expensive. Nevertheless, the reader should think of this result as a reference convergence performance for Instance 1.

The alternative is to estimate both the new node potentials and the optimality parameter. However, in order for Cancel-and-Tighten to benefit from the complexity analysis, one must have a valid combination of such estimates at the beginning of each phase. Goldberg and Tarjan (1989) call this the explicit maintenance of a price function $[\hat{\boldsymbol{\pi}}^h, \hat{\mu}^h]$, $h \geq 0$. Recall the dual version of the pricing problem (2.6)–(2.7). Once the node potentials are fixed, there is

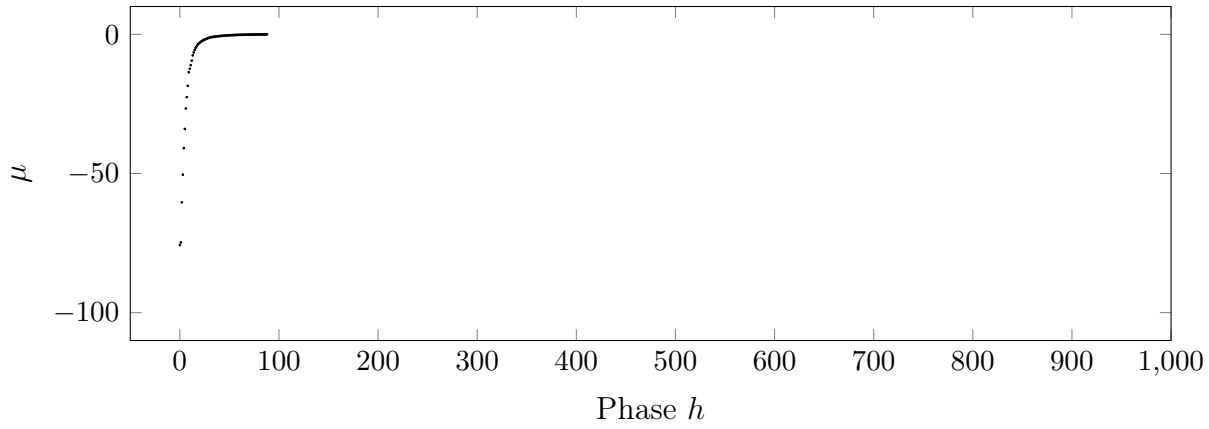


Fig. 2.7: Optimality parameter μ for Instance 1 [phase base - Optimal]

actually little room for $\hat{\mu}^h$. Indeed, $\hat{\mu}^h := \min_{(i,j)\in A(\mathbf{x}^h)} \hat{d}^h_{ij}$, where $\hat{d}^h_{ij} := d_{ij} - \hat{\pi}^h_i + \hat{\pi}^h_j$, which means that $\mu^h \geq \hat{\mu}^h$.

Given that μ^{h+1} would be the optimal solution to the pricing step at the current state of the algorithm (canceling a Type 2 cycle W^l), the goal is simple: establish $[\hat{\boldsymbol{\pi}}^{h+1}, \hat{\mu}^{h+1}]$ such that $\hat{\mu}^{h+1} = \mu^{h+1}$. Of course the latter is only wishful thinking but fortunately we still have one card up our sleeve. Assume $\underline{\mu}^{h+1}$ is a valid lower bound for μ^{h+1}. The new node potentials $\hat{\boldsymbol{\pi}}^{h+1}$ must therefore be such that

$$\hat{\mu}^{h+1} \geq \underline{\mu}^{h+1} \geq (1 - 1/n)\, \hat{\mu}^h. \tag{2.19}$$

The right-hand side inequality ensures that $\hat{\mu}^{h+1} > \hat{\mu}^h$ within the minimalist specifications of the $(1 - 1/n)$ jump property. The importance of the lower bound lies in its ability to transfer additional information to the node potentials in a constructive manner. As such, this lower bound not only aims to lift the optimality parameter as much as possible, it must also sport some intrinsic value with respect to the definition of a measurable jump obtained at the end of a phase. We provide a new approximation structure influenced by the tighter jump factor seen in (2.16).

In the spirit of the leading premise of the Tighten step, assume estimates $[\hat{\boldsymbol{\pi}}^h, \hat{\mu}^h]$ are readily available. Let us establish what we do know about the next minimum mean cycle. Since $G(\mathbf{x}^{h+1}, \hat{\boldsymbol{\pi}}^h)$ is acyclic, it is possible to associate a level, $L^h_i, i \in N$, to each node using a topological ordering. These levels are recursively defined as

$$L^h_j := \max_{(i,j)\in A(\mathbf{x}^{h+1}, \hat{\boldsymbol{\pi}}^h)} L^h_i + 1,$$

with $L_i^h := 0$ if node i has no incoming admissible arcs. By construction of the ordering, if an arc $(i,j) \in A(\mathbf{x}^{h+1})$ is still admissible with respect to $\hat{\boldsymbol{\pi}}^h$ then the levels L_i^h and L_j^h are such that $L_j^h > L_i^h$. All other arcs have $L_j^h \leq L_i^h$.

Proposition 14. *The marginal update* $\hat{\pi}_i^{h+1} := \hat{\pi}_i^h - \frac{L_i^h}{L^h+1}\hat{\mu}^h$, $\forall i \in N$, *yields a valid estimation for* $\hat{\mu}^{h+1}$. *Three possible values for* L^h *are*

$$L_{(a)}^h := n-1, \qquad L_{(b)}^h := \max_{i \in N} L_i^h, \qquad L_{(c)}^h := \max_{(i,j) \in A(\mathbf{x}^{h+1}) | L_i^h > L_j^h} L_i^h - L_j^h.$$

Proof. Since the levels start at 0, the value L^h can be seen as the length of a path in the admissible network. Update $L_{(a)}^h$ is the first of two implementations proposed by Goldberg and Tarjan (1989) and obviously extracts nothing from the level values. Update $L_{(b)}^h$ can be seen as the length of the longest path in terms of the number of arcs. Update $L_{(c)}^h$ refines this value by checking in the residual network for the existence of an arc capable of inducing a cycle on the longest path. In order to show that all options for L^h induce valid lower bounds for μ^{h+1}, it suffice to realize that any arc added to create a cycle on the path referred by L^h has a non-negative reduced cost. Observe that these arc length values can be ordered as $L_{(a)}^h \geq L_{(b)}^h \geq L_{(c)}^h$.

The proof that the transformation is valid is reminiscent of (2.19) and shows that the inequality is indeed verified. The modified reduced costs are evaluated by

$$\hat{d}_{ij}^{h+1} := d_{ij} - \left(\hat{\pi}_i^h - \frac{L_i^h}{L^h+1}\,\hat{\mu}^h\right) + \left(\hat{\pi}_j^h - \frac{L_j^h}{L^h+1}\,\hat{\mu}^h\right) = \hat{d}_{ij}^h - \frac{L_j^h - L_i^h}{L^h+1}\,\hat{\mu}^h, \quad \forall (i,j) \in A(\mathbf{x}^{h+1}). \tag{2.20}$$

In the case of admissible arcs, we have $1 \leq L_j^h - L_i^h \leq L^h$. Therefore,

$$\hat{d}_{ij}^{h+1} \geq \hat{\mu}^h - \frac{L_j^h - L_i^h}{L^h+1}\,\hat{\mu}^h = \left(1 - \frac{L_j^h - L_i^h}{L^h+1}\right)\hat{\mu}^h \geq \left(1 - \frac{1}{L^h+1}\right)\hat{\mu}^h.$$

In the case of non-admissible arcs, we have $0 \leq L_i^h - L_j^h \leq L^h$. We also know that these arcs have a non-negative reduced cost. Therefore,

$$\hat{d}_{ij}^{h+1} \geq 0 - \frac{L_j^h - L_i^h}{L^h+1}\,\hat{\mu}^h = \frac{L_i^h - L_j^h}{L^h+1}\,\hat{\mu}^h \geq \left(1 - \frac{1}{L^h+1}\right)\hat{\mu}^h.$$

Since $\underline{\mu}^{h+1} = \left(1 - \frac{1}{L^h+1}\right)\hat{\mu}^h$ is an acceptable lower bound for μ^{h+1}, the proof is concluded. □

Remark. Although we did not include the second implementation proposed in Goldberg and Tarjan (1989), call it Update (d), we have carried out experiments using all four. The reasons for this omission are twofold. First, it does not perform as well as our two new update proposals $L^h_{(b)}$ and $L^h_{(c)}$. Second, it does not land itself naturally to our presentation with value L^h which is needed to reduce the complexity. The update is as follow: $\hat{\pi}^{h+1}_i := \hat{\pi}^h_i - qL^h_i,\ \forall i \in N$, where $q := \min_{(i,j)\in A(\mathbf{x}^{h+1})|L^h_i > L^h_j} \dfrac{\hat{d}^h_{ij} - \hat{\mu}^h}{L^h_i - L^h_j + 1}$. Observe that using value 0 instead of $\hat{d}^h_{ij}$ delivers Update $L^h_{(c)}$.

Initialization. Notice that while optimal values $[\boldsymbol{\pi}^h, \mu^h]$ do exist, the point of the approximation scheme is to carry out the computations with the approximation values $[\hat{\boldsymbol{\pi}}^h, \hat{\mu}^h]$ instead. As such, the explicit price function can be trivially initialized to $\hat{\pi}^0_i := 0,\ \forall i \in N$, and $\hat{\mu}^0 := \min_{(i,j)\in A(\mathbf{x}^0)} d_{ij}$.

The importance of properly updating the node potentials is capital to the speed of convergence. Indeed, the systematic computations over approximation values means that the associated errors are cumulative. Intuitively, a poor update of the node potentials induces a new admissible network that is not sufficiently altered to reveal new Type 1 cycles. Figures 2.8 to 2.10 illustrate the point. Each figure contains three elements: two plots and a marking. The top level plot is the optimal value of μ^h retrieved for the sake of evaluating the quality of the estimate $\hat{\mu}^h$ as seen in the lower level plot. The marking indicates the phase number at which the optimal solution is reached although without proving it. Figure 2.8 uses Update $L^h_{(a)}$ and suffers from an extremely poor convergence rate. There are in fact over 7,000 additional phases required to prove optimality. In Figure 2.9, we see that Update (d) performs much better than Update $L^h_{(a)}$. However, as can be seen in Figure 2.10, it is still outperformed by Update $L^h_{(b)}$. These approximations require respectively 464 (343) phases to prove optimality. The results for Update $L^h_{(c)}$ are omitted because they are almost the same as those of Update $L^h_{(b)}$. As expected by the update mechanism, the estimate $\hat{\mu}^h$ is a lower bound for the optimal value μ^h. The quality of the update clearly influences how fast this bound is increased.

Proposition 15. *(Goldberg and Tarjan 1989) The combination of the Cancel and Tighten steps runs in $O(m \log n)$ time.*

Proof. The Tighten step consists of a succession of basic operations on the arcs or the nodes, the most complex one being the topological ordering which runs in $O(m)$ time. The proof that the Cancel step terminates in $O(m \log n)$ time, making it the dominant method, is influenced by the works of Sleator and Tarjan (1983). The key lies in the realm of computer

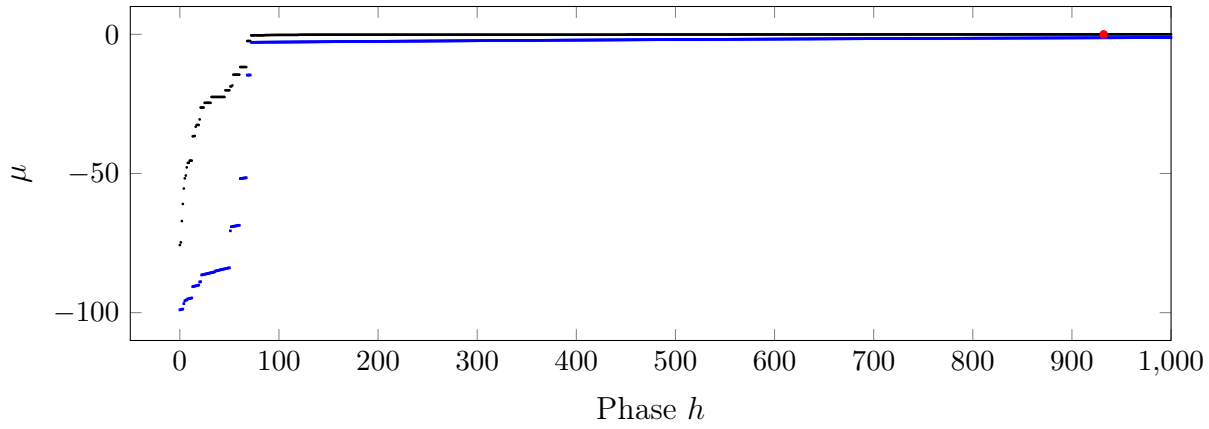


Fig. 2.8: Optimality parameter μ and estimate $\hat{\mu}$ for Instance 1 [phase base - Update $L^h_{(a)}$]

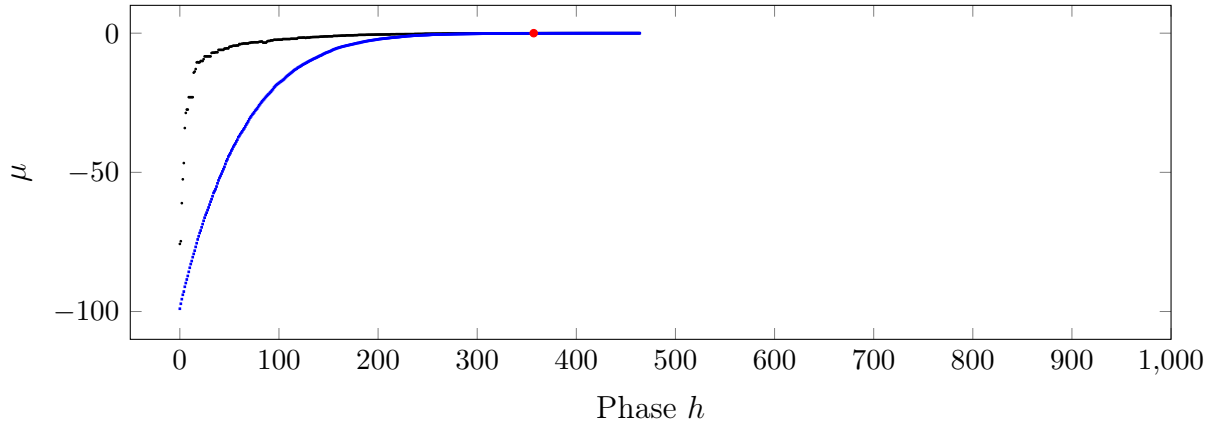


Fig. 2.9: Optimality parameter μ and estimate $\hat{\mu}$ for Instance 1 [phase base - Update (d)]

science whereby the sophisticated *splay tree* data structure allows for an efficient way to exhaustively search the admissible network. □

Using this strategy transcends the bottleneck incarnated by the pricing step with a convoluted approach. The bottleneck operation is now the completion of a phase which effectively allows the complexity analysis to trade the initial $O(m^2 n)$ time per phase in favor of an amortized $O(m \log n)$ time. The improvement provided by the Cancel-and-Tighten strategy is the fruit of careful design. It is obtained by shedding another light on the iteration-wise analysis.

While it is possible to fetch optimal node potentials at the end of every phase, this measure brings the global complexity to $O(n \log(nC)) \times [O(m \log n) + O(mn)] \equiv O(mn^2 \log(nC))$ for integer arc costs and $O(mn) \times [O(m \log n) + O(mn)] \equiv O(m^2 n^2)$ for arbitrary real-valued arc costs.

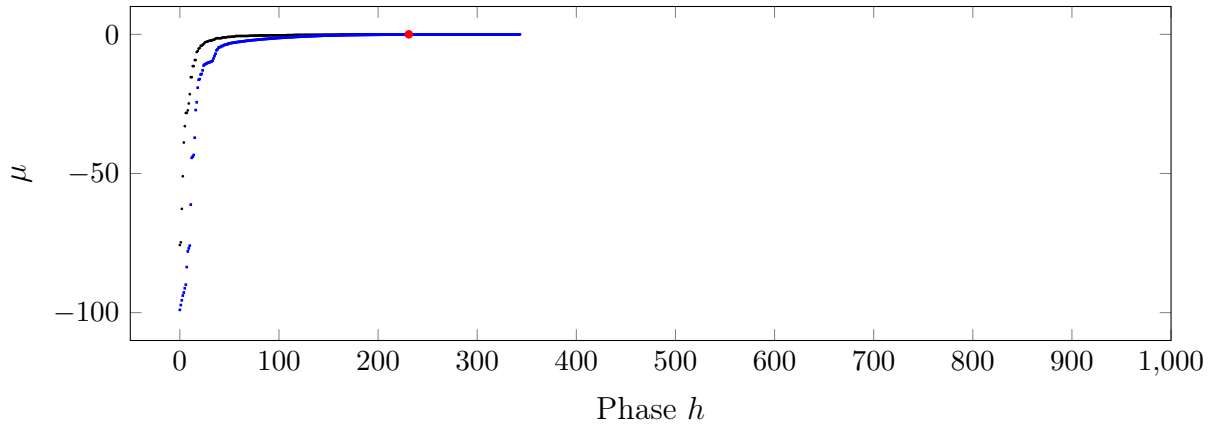


Fig. 2.10: Optimality parameter μ and estimate $\hat{\mu}$ for Instance 1 [phase base - Update $L^h_{(b)}$]

The following two theorems present the global runtime complexity of MMCC when the Cancel-and-Tighten strategy is incorporated along with the approximation scheme. Although we have separated these theorems to highlight our new proposal in the second one, the reader is invited to read them as one. The idea behind their proofs is to place the resolution process in the same conditions as the complexity analysis of the outer loop. Since $\hat{\mu}^h \leq \mu^h$, it is conceivable to rewrite Propositions 9 and 11 using approximation values $[\hat{\boldsymbol{\pi}}^h, \hat{\mu}^h]$ instead. Indeed, arc fixing than occurs in a more conservative fashion.

Theorem 5. *(Goldberg and Tarjan 1989,* Theorem 4.3*) MMCC accompanied by the Cancel-and-Tighten strategy runs in $O(mn \log n \log(nC))$ time for integer arc costs and $O(m^2 n (\log n)^2)$ time for arbitrary real-valued arc costs.*

Proof. With respect to integer arc costs, the adaptation is straightforward. It turns out that the approximation values $\hat{\mu}$ follow the same geometric progression as the optimal ones. Indeed, the construction of the proof of Theorem 1 also holds with the approximation values. This means that $\hat{\mu} > -1/n$ is a valid stopping criterion and $O(n \log(nC))$ phases are performed.

In the case of arbitrary real-valued arc costs, we consider here the point of view of Theorem 2. It is therefore sufficient to have a measurable jump of $(1 - 1/n)$ at the end of each phase to obtain the wanted complexity. In the Tighten step, irrespectively of whether these phases are approximated or not, this same property is verified by construction, see (2.19). The approximation values therefore still follow the same behavior as the optimal ones. Hence, the number of phases using the Cancel-and-Tighten strategy is also $O(mn \log n)$. As far as the stopping criterion is concerned, polling for the optimal value of μ every n-th phase to assert the optimality certificate can effectively be done without compromising the complexity result. Indeed, the $O(mn)$ runtime of this operation can be discarded

with respect to the amortization against the runtime of the $n-1$ previous approximated phases, i.e., $O((n-1)m\log n + mn) \equiv O(mn\log n)$.

Since each phase runs in $O(m\log n)$ time, the proof is brought to terms with its statement. □

Theorem 6. *MMCC accompanied by the Cancel-and-Tighten strategy runs in $O(m^2 n\log n)$ time for arbitrary real-valued arc costs.*

Proof. From the point of view of Theorem 3, the use of the explicit price function $[\hat{\boldsymbol{\pi}}, \hat{\mu}]$ in accordance with Proposition 14 commands another look at Proposition 12. The following modified version of the proof basically suggests that a valid size indicator, say L^h+1, for the jump is sufficient, the actual size of the Type 2 cycle is not really needed.

Assume all update coefficients $L^h+1 \leq L(\mathbf{x}^h)$ in these $L(\mathbf{x}^h)$ phases. Then, by (2.16),

$$\hat{\mu}^{L(\mathbf{x}^h)} \geq -|\bar{c}^*_{m(\mathbf{x}^h)}| \left(1 - \frac{1}{L(\mathbf{x}^h)}\right)^{L(\mathbf{x}^h)} > \frac{-|\bar{c}^*_{m(\mathbf{x}^h)}|}{2}.$$

Otherwise, some coefficient L^l+1 at the end of phase $l \leq L(\mathbf{x}^h)$ has cardinality $L^l+1 > \frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|}$. Then,

$$\hat{\mu}^{L(\mathbf{x}^h)} \geq \hat{\mu}^l \geq \frac{L^l\,\mu(\mathbf{x}^h)}{L^l+1} \geq \frac{-S_{m(\mathbf{x}^h)}}{L^l+1} > \frac{-S_{m(\mathbf{x}^h)}}{\frac{2S_{m(\mathbf{x}^h)}}{|\bar{c}^*_{m(\mathbf{x}^h)}|}} = \frac{-|\bar{c}^*_{m(\mathbf{x}^h)}|}{2}.$$

Combine this with Proposition 13 and the $O(mn)$ result of Theorem 3 still stands. The global complexity ensues once the per phase runtime of $O(m\log n)$ is accounted for. The polling argument for the optimal value of μ is still applicable for the stopping criterion. □

Remark. While the original intent of Update (d) is not aligned with that of the approximation structure proposed, a value $L^h_{(d)}$ could still be extracted from the determination of the value q and therefore would still have the same complexity as the other updates listed.

2.3.6 Summary and observations

The first part of this section assembles all the complexity results we have seen while the second raises several worthy observations about technical and mechanical aspects of the algorithm.

Table 2.1 summarizes the content of the complexity analysis. The first two columns display the complexity of the outer loop of MMCC depending on whether one thinks in terms of $O(m^2n)$ iterations or $O(mn)$ phases. An implementation which uses the Cancel-and-Tighten strategy grants a better theoretical complexity by consolidating several bottleneck operations. The last two columns show the global complexity of MMCC depending on whether or not it is incorporated.

	Outer loop		Global runtime complexity Cancel-and-Tighten strategy	
Point of view	No. of iterations	No. of phases	Without	With
Theorem 1	$O(mn\log(nC))$	$O(n\log(nC))$	$O(m^2n^2\log(nC))$	$O(mn\log n\log(nC))$
Theorem 2	$O(m^2n\log n)$	$O(mn\log n)$	$O(m^3n^2\log n)$	$O(m^2n(\log n)^2)$
Theorem 3	$O(m^2n)$	$O(mn)$	$O(m^3n^2)$	$O(m^2n\log n)$

Tab. 2.1: Complexity analysis summary

Observe that the better complexity achieved by the integration of the Cancel-and-Tighten strategy within the MMCC framework is not a contradiction with the tight complexity of Radzik and Goldberg (1994). It is rather a testament to the importance of careful design. Indeed, the tight bound of $O(m^2n)$ iterations is superseded by the equivalent one of $O(mn)$ phases.

Aside from the theoretical improvements, the very essence of Cancel-and-Tighten exudes efficiency on several fronts. First, its conception allows for a more straightforward approach to the identification of negative cycles which further benefits from running on a sparser graph, that is, $A(\mathbf{x}^{h+1}, \hat{\boldsymbol{\pi}}^h) \subseteq A(\mathbf{x}^h, \hat{\boldsymbol{\pi}}^h) \subseteq A(\mathbf{x}^h)$. Second, the data structure allows better redundancy control than the iteration base approach which systematically restarts from scratch. Third, the ability to reuse information regarding the node potentials is not only appealing but also proves to be useful. Finally, its design matches the critical component of the complexity analysis and only aims to reach these important Type 2 cycle checkpoints as fast as possible.

Explicit arc fixing. The strongly polynomial complexity is obtained by introducing the concept of arc fixing. Truth be told, the bidirectional verification of the alternative statement for Proposition 8 can actually be carried out in any vanilla implementation of MMCC without compromising the resolution process nor the theoretical complexity. Fixed arcs can be removed from the system thus giving the rule a very practical effect. The arc fixing rule

provided by Proposition 8 (or Proposition 9, extended to phases) can be, for all intent and purposes, *explicit*. Furthermore, observe that arc fixing is reserved for non-free variables. When $\mu = 0$, optimality is achieved. Post applying Proposition 8 would imply *all* variables get fixed regardless of their status. We underscore the fact that, in Proposition 10 (or Proposition 11, extended to phases), arc fixing is invariably implicit because it depends on an unknown optimal set of node potentials.

Cancellations. Recall that the iteration base necessitates 1,937 iterations to terminate (Figure 2.5). This means that the same number of cycles are canceled. Whether the phase base contains more or less cancellations is matter of resolution course but it is possible to give meaning to what we have observed. Let us speak numbers. In the optimal update method (Figure 2.7), 4,176 Type 1 cycles are canceled while in the three approximations (Figures 2.8 to 2.10), these numbers are respectively 2,192 and 5,202 and 4,417. The first approximation is strikingly different from the rest and actually is much closer to the iteration base number. Let us address this case first. With a poor update of the node potentials, the content of the admissible network is limited to a small fraction of arcs. This means that although we are not looking expressly for it, the minimum mean cycle is more likely to be identified. In other words, the Cancel-and-Tighten strategy behaves similarly to the iteration base. In the other two approximations, the admissible network contains a lot of negative arcs and Type 1 cycles are identified in a random manner. It appears natural that more cycles are identified in this way than does the meticulous process of finding the best ones sequentially. What comes out of this interpretation is that it is more important to identify negative cycles fast than it is identifying the *best* one.

Tailing-off. The iteration base suffers from a tailing-off effect which can be explained by the nature of line search optimization and also that of MMCC. Regardless of the quality of the solution, all the little improvements must be accounted for before granting the optimality certificate. Since the optimality parameter is a gage for the expected improvement and that it converges to zero from below, the end of the resolution process is very much like a quest for crumbs. In the phase base, notice that all three approximation updates also suffer from a tailing-off effect which is even present in the optimal update method. Since the optimality parameter still intervenes, the same explanation holds as well for this approach. But there is more. The quality of the update plays a great role in shortening the tail. Indeed, the latter dictates both the content of the admissible network and the distance to optimality.

Switch offs. In practice, the tailing-off effect leads to believe that, when the optimality parameter reaches a very small value, it might be worthwhile to switch off to the iteration base. Indeed, as the process nears the optimal solution, the number of negative cycles becomes very small which makes the content of the admissible network even more limited in terms of Type 1 cycles. Furthermore, the important graph reduction induced by the arc fixing dramatically reduces the computational penalty of the iteration base. On another note, the usefulness of Update $L^h_{(c)}$ is still unclear but the otherwise lack of tractable benefit suggests it might be wise to postpone its usage until the first variables get fixed in order to limit the impact of the additional $O(m)$ time it requires.

Dantzig-Wolfe decomposition. The decomposition of the residual problem (2.2)–(2.4) using a Dantzig-Wolfe decomposition scheme (Dantzig and Wolfe 1960) brings yet another perspective to Cancel-and-Tighten. Let $k \geq 0$ denote a non-optimal iteration number. At every iteration, define the master problem as the set of upper bound constraints $y_{ij} \leq r^k_{ij}, (i,j) \in A(\mathbf{x}^k)$, and formulate the subproblem for maximizing the minimum reduced cost. This results in the proposed pricing step defined by the primal-dual pair of linear programs (2.6)–(2.7) and (2.8)–(2.11). Its solution provides an improving cycle for which the maximum step size is computed using the master problem constraints. Therefore, Cancel-and-Tighten corresponds to heuristic solutions of that pricing problem, indeed a partial pricing devoted to Type 1 cycles only. An optimal solution to the residual problem (2.2)–(2.4) can only be guaranteed by solving the pricing subproblem to optimality and finding $\mu = 0$.

2.4 Conclusion

This paper aims to present the minimum mean cycle-canceling algorithm in its entirety by regrouping the knowledge from different sources. Ranging from the objective function to implicit arc fixing, a key component which traverses most of the proofs is a line of rationalization which questions the existence of admissible solutions. This admissibility defers to a fundamental piece of network theory, namely the flow decomposition principle. In the end, the same algorithm has been studied under numerous angles, each one providing theoretical breakthroughs. A very interesting point is that better bounds are obtained via very practical observations such as tighter jump factors or well conditioned admissible networks.

The original purpose of this work was a literature review preliminary to further researches, but we have come to see it as more than a summary. First of all, we harness the power of duality to simplify one of the building blocks of the algorithm. Secondly, the new way to look

at the analysis directly in terms of $O(mn)$ phases is elegant in itself. We feel that we have integrated Cancel-and-Tighten to the minimum mean cycle-canceling framework under a new perspective and even contributed to its performance with new update approximations in the Tighten step. This in turn grants the reduction of the global runtime to $O(m^2 n \log n)$ and speaks volume about the importance of thinking in terms of phases. This third contribution is made possible by the generalization of Proposition 12. Finally, the computation results are enlightening of the resolution course of the minimum mean cycle-canceling algorithm. The different observations which come out of this study serve the practical side of things by reducing the wall-clock time of any generic implementation.

As a final note, this work is part of a much broader plan which includes generalizations to linear programming as well as understanding the ramifications with the *Improved Primal Simplex* method (Elhallaoui et al. 2011, Raymond et al. 2010b) in order to extract necessary adjustments required to recuperate some of the properties established herein.

Tools for degenerate linear programs: IPS, DCA and PE

Jean Bertrand Gauthier and Jacques Desrosiers

GERAD & HEC Montréal, Canada

3000, chemin de la Côte-Sainte-Catherine

Montréal (Québec) Canada, H3T 2A7

<jean-bertrand.gauthier, jacques.desrosiers>@hec.ca

Marco E. Lübbecke

RWTH Aachen University, Germany

Kackertstraße 7

D-52072 Aachen, Germany

marco.luebbecke@rwth-aachen.de

ABSTRACT

This paper describes three recent tools for dealing with primal degeneracy in linear programming. The first one is the *Improved Primal Simplex* (IPS) algorithm which turns degeneracy into a possible advantage. The constraints of the original problem are dynamically partitioned based on the numerical values of the current basic variables. The idea is to work only with those constraints that correspond to nondegenerate basic variables. This leads to a row-reduced problem which decreases the size of the current working basis. The main feature of IPS is that it provides a nondegenerate pivot at every iteration of the solution process until optimality is reached. To achieve such a result, a negative reduced cost convex combination of the variables at their bounds is selected, if any. This pricing step provides a necessary and sufficient optimality condition for linear programming. The second tool is the *Dynamic Constraint Aggregation* (DCA), a constructive strategy specifically designed for set partitioning constraints. It heuristically aims to achieve the properties provided by the IPS methodology. We bridge the similarities and differences of IPS and DCA on set partitioning models. The final tool is the *Positive Edge* (PE) rule. It capitalizes on the compatibility definition to determine the status of a column vector and the associated variable during the reduced cost computation. Within IPS, the selection of a compatible variable to enter the basis ensures a nondegenerate pivot, hence PE permits a trade-off between strict improvement and high reduced cost degenerate pivots. This added value is obtained without explicitly computing the updated column components in the simplex tableau. Ultimately, we establish tight bonds between these three tools by going back to the linear algebra framework from which emanates the so-called concept of subspace basis.

Keywords: Primal simplex, degeneracy, combination of entering variables, Positive Edge rule, nondegenerate pivot algorithm, dynamic Dantzig-Wolfe decomposition, vector subspace.

3.1 Introduction

When solving a linear program with the primal simplex (PS) algorithm (see Dantzig 1963), degeneracy comes in two flavors: degenerate solutions and degenerate pivots. The first case is a question of observation, it is a dichotomous state of the solution which either exhibits degenerate basic variables or not. A basic solution is degenerate if at least one of its basic variables is degenerate, that is, at its lower or upper bound. Geometrically speaking, it corresponds to an over-represented vertex meaning that several equivalent bases are associated with the same solution. The second case is the algorithm's culprit in more ways than one. In fact, it is the only phenomenon which jeopardizes its convergence both theoretically and empirically. Degeneracy questions PS' efficiency and creates ambiguity in the post-analysis. On the one hand, degeneracy can affect the efficiency in obtaining an optimal solution because it creates redundant work. More specifically, a degenerate pivot amounts to trading one degenerate basic variable for a nonbasic one. Since no gain is made with respect to the objective function, it is in the aftermath of the computations that one ultimately realizes the wasted effort. It is even possible to cycle meaning that PS moves through a series of bases eventually returning to an already visited one. If this happens indefinitely, PS may not even converge (Schrijver 1986). On the other hand, a by-product of PS is the sensitivity analysis done after the optimization. Each constraint is associated with a dual variable whose value depends on the chosen basis. Since an optimal degenerate basis is not uniquely defined, it can mislead the interpretation of two otherwise equivalent solutions.

It should be noted that column generation, used to solve linear programs with a huge number of variables (Barnhart et al. 1998, Lübbecke and Desrosiers 2005), is a natural extension of PS, and as such suffers from degeneracy as well. With that being said, degeneracy is a phenomenon encountered particularly often for linear programming relaxations of combinatorial optimization problems. Set partitioning and set covering models are prominent examples of practical relevance: vehicle routing and crew scheduling problems (and many related problems in transportation and logistics) are most successfully formulated this way (Desrosiers et al. 1995, Desaulniers et al. 1998).

Degeneracy has been under scrutiny for practically as long as linear programming. We distinguish two lines of studies from the literature. The first aims to eliminate degeneracy altogether and the other provides guidelines to alleviate its impact. On the first count, think of the work of Charnes (1952) which revolves around modifying the polytope of the whole solution space in such a way that no two solutions ever share the same vertex. The concept amounts to right-hand side perturbations thus creating slight variations in the way

the hyperplanes intersect. While the idea of eradicating degeneracy altogether is appealing, today's simplex codes use a more *ad hoc* strategy which sends us to the second count. The contributions of Wolfe (1963) and Ryan and Osborne (1988) are abundant evidence that applying this strategy as necessary is highly effective. The perturbations are now applied in an adaptive manner and on a more local scale. *Stabilization* extends the idea of perturbation by incorporating dual information. Penalty functions, trust regions and expansion strategies are among the instrumental concepts of stabilization as described in the papers of du Merle et al. (1999) and Ben Amor et al. (2009). Column generation benefits from the latter as it tackles the particular sensitivity to the values of dual variables during the resolution process.

Numerous pivot rules have also been proposed to avoid performing degenerate pivots. In this regards, the work of Terlaky and Zhang (1993) is enlightening in many respects. Indeed, while many of these rules share common properties and sometimes even correspond to special cases of one another, they are distinguished according to certain properties: feasibility maintenance, anti-cycling feature and recursive nature. While there might have been hope about the performance of many of these rules, even nondegenerate instances can be difficult to optimize as supported by Klee and Minty (1972). The performance of a pivot rule may therefore be considered as a trade-off between its complexity and the savings it procures with respect to the number of iterations. The state of the art in terms of degenerate problems seems to be the Devex rule of Harris (1973), see Terlaky and Zhang (1993). We underline that regardless of their intricacies, all of these rules have a limited gain with respect to the absence of guaranteed efficiency. That is, zero step size pivots could still ensue from the chosen direction. The anti-cycling feature present in Bland (1977) or Fukuda (1982) ensures this behavior does not happen indefinitely. It is however generally accepted that taking expensive measures to protect against cycling is not worthwhile.

A new trend appears in the late nineties with the paper of Pan (1998) who formulates a generic basis for degenerate solutions. Embedding this concept in a column generation scheme led to the *Dynamic Constraint Aggregation* (DCA) algorithm of Elhallaoui et al. (2005, 2008) for the set partitioning problem. This problem lends itself particularly well to such a concept because of its peculiar structure. Indeed, it is this very structure that allows DCA to heuristically harness the power of a generic basis and quite often find strictly improving pivots. The paper of Elhallaoui et al. (2011) extends the algorithmic methodology with the *Improved Primal Simplex* (IPS). As its name would have it, this extension takes place with regards to any linear programming problem. In a nut shell, the structure of a solution is preemptively taken into account in order to drive the next pivot in a strictly

improving direction. That structure is dynamically updated with respect to the current solution.

The methodological paper at hand describes three tools for dealing with primal degeneracy. At the heart of this framework lies the search for so-called *compatible* column vectors and associated variables. Whether such a column exists as is in the original problem or is constructed as a convex combination of these, it corresponds to a direction in a polytope induced by a transformation of the original simplex. As such, we believe that IPS provides a better starting point from which the other two tools can benefit in terms of presentation. The second tool is of course DCA (which was incidentally designed prior to IPS) and the third is the *Positive Edge* (PE) rule (Raymond et al. 2010a, Towhidi et al. 2014). It is safe to say that DCA is a method steered by practical imperatives. Yet, explaining the reason behind its performance can now also be done in a straightforward manner in light of the IPS framework. PE is yet another example of benefits obtained from a higher level of abstraction. Indeed, while manipulating the set of compatible vectors can be computationally efficient, identifying said set can be time consuming for large problems. PE aims to simplify this verification by extracting the compatibility status during the reduced cost computation using the original column data.

The paper is organized as follows. Section 3.2 first exposes the theory of IPS with several hints to PS. By casting the linear algebra framework on our study, Section 3.3 presents another perspective of IPS. Section 3.4 addresses the more practical side with regards to several implementation choices. The importance of compatibility in the design of specialized applications is highlighted in Section 3.5. The similarities and differences between IPS and DCA are examined in Section 3.6 while Section 3.7 reveals PE. Various results from the literature are reported at the end of Sections 3.4, 3.6 and 3.7 depending on the context of the underlying tool. Our conclusions end the paper in Section 3.8.

Motivation. In the words of Perold (1980), *a great many degenerate iterations* is usually the resulting observation of degeneracy. As a matter of fact, it is not unusual to see that when an average of 20% of basic columns are degenerate, 50% of the iterations are degenerate. While the former statement gives a feel for the negative impact of degeneracy, the second statement rapidly frames it within a quantitative measure. The degenerate variables percentage of each basic solution encountered during the resolution process is averaged over the number of iterations. As such, it is certainly possible to characterize a linear program as degenerate if some basis exhibits such a quality, yet it is much more interesting to measure the extent of this pathology. The latter is based on empirical evidence. A linear program is thus said to have a *degeneracy level* of $\beta\%$, where $\beta = 20$ corresponds to the average in Perold's example.

In the same vein, a whole class of linear programs can be qualified in the same manner by computing the mean of these values. For instance, assignment network problems have a degeneracy level of 50%, or even 100% if upper bounds are explicit. We do not know of any guidelines to state that family classes are degenerate, but it is fair to say that the level should be at least 20%. In vehicle routing, it is immediate how degeneracy occurs: The constraints represent tasks (often large numbers) to be covered by relatively few vehicles or crew members, that is, only few variables assume a positive value, especially in an integer solution.

Notation and terminology. Vectors and matrices are written in bold face. We denote by $\mathbf{I}_\ell$ the $\ell \times \ell$ identity matrix and by $\mathbf{0}$ (resp. $\mathbf{1}$) a vector/matrix with all zeros (resp. ones) entries of contextually appropriate dimension. For a subset $I \subseteq \{1, \ldots, m\}$ of row indices and a subset $J \subseteq \{1, \ldots, n\}$ of column indices, we denote by $\mathbf{A}_{IJ}$ the sub-matrix of $\mathbf{A}$ containing the rows and columns indexed by I and J, respectively. We further use standard linear programming notation like $\mathbf{A}_J\mathbf{x}_J$, the subset of columns of $\mathbf{A}$ indexed by J multiplied by the corresponding sub-vector of variables $\mathbf{x}_J$. The lower case notation is reserved for vectors and uses the same subset index rules. In particular, the matrix $\mathbf{A} := (\mathbf{a}_j)_{j\in\{1,\ldots,n\}}$ contains n column vectors. Finally, there is one notable exception: The set N does *not* denote the nonbasis but rather the set of basic and nonbasic variables at their lower or upper bounds. Hence, for a linear program in standard form, $\mathbf{x}_N$ represents the vector of null variables.

The pricing step in the seminal IPS papers refers to solving a *complementary problem* whereas it was later shown that IPS can be seen as a dynamic Dantzig-Wolfe decomposition at every iteration. As a survey paper, we use a unifying terminology and choose to define the pricing step as solving a *pricing problem*.

3.2 Improved Primal Simplex

This section first exposes the theory of IPS in the context of a linear program with lower and upper bounded variables. It is based on the original papers of Elhallaoui et al. (2011), Raymond et al. (2009, 2010b), Metrane et al. (2010) and its generalization to row-reduced column generation (Desrosiers et al. 2014). However, contrary to the original presentation, the choice of using a bounded linear program in the description of IPS is becoming of its purpose. For instance, in set partitioning problems, degenerate upper bounds are exploited for a faster resolution. Moreover, the change of variables utilized for the row partition also becomes more apparent with upper bounds.

We present in Section 3.2.1 the algorithmic steps of IPS. Section 3.2.2 provides the proof of a necessary and sufficient optimality condition derived from the improved pricing step. Section 3.2.3 presents a simplified version of IPS for linear programs in standard form. For a better understanding of the concepts, an illustrative example is given in Section 3.2.4 on a small linear program.

Consider a linear program (LP) with lower and upper bounded variables:

$$\begin{aligned} z^\star := \min \quad & \mathbf{c}^\intercal \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \quad [\boldsymbol{\pi}] \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned} \tag{3.21}$$

where $\mathbf{x}, \mathbf{c}, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $m < n$. We assume that $\mathbf{A}$ is a matrix of full row rank and that LP is feasible and bounded. Finally, $\boldsymbol{\pi} \in \mathbb{R}^m$ is a vector of dual variables associated with the equality constraints.

3.2.1 Algorithmic steps

The main idea in IPS is to reduce the number of constraints from m to f, the number of nondegenerate or free variables in a basic solution. The advantage of this row reduction is a smaller working basis of dimension $f \times f$ rather than the usual larger one of dimension $m \times m$. This comes at the expense of a more involved pricing step which solves a linear program of row size $m - f + 1$ to select an improving subset of columns, that is, a convex combination of columns with two properties: this selection is compatible with the current row-reduced problem (see Definition 3) and its reduced cost is negative. If such a combination exists, a strict improvement in the objective function value occurs, otherwise the current solution is optimal. Figure 3.11 contains an overview of the main steps of IPS. The initialization contains the change of variables, the input basic solution $\mathbf{x}^0$, and the associated column partition with the null variables set N. The main loop provides: (1) the construction of a generic basis and the resulting linear transformation and row partition; (2) the definition of compatibility; (3) the development of an improving pricing step; (4) the exchange mechanism from a solution $\mathbf{x}^0$ to the next $\mathbf{x}^1$ which incidentally brings an inspiring twist to the pivoting rule; (5) the update of the column partition.

Initialization Let $\mathbf{x}^0$, represented by $(\mathbf{x}^0_F; \mathbf{x}^0_L; \mathbf{x}^0_U)$, be a basic solution where the three sub-vectors are defined according to the value of their variables: $\mathbf{x}^0_L$ at their lower bounds, $\mathbf{x}^0_U$ at their upper bounds, and *free* variables $\mathbf{l}_F < \mathbf{x}^0_F < \mathbf{u}_F$. Free variables are basic and they

Initialization : basic solution $\mathbf{x}^0$;
change of variables;
column partition $\{F, L, U\}$ and $N := L \cup U$;
1 Generic basis $\mathbf{B}$, transformation $\mathbf{B}^{-1}$, row partition $\{P, Z\}$ of $\mathbf{A}_F$;
2 Compatibility with the row partition $\{P, Z\}$ of $\mathbf{A}_F$ <optional>;
3 Improved pricing step: maximize the minimum reduced cost μ;
4 Exchange mechanism from $\mathbf{x}^0$ to $\mathbf{x}^1$;
5 Set $\mathbf{x}^0 := \mathbf{x}^1$, update the column partition $\{F, L, U\}$ and goto Step 1;

Fig. 3.11: IPS algorithmic steps

can move below or above their current value which obviously makes them nondegenerate. Let there be $f := |F|$ such free variables, $0 \leq f \leq m$. Partition the matrix $\mathbf{A} = [\mathbf{A}_F, \mathbf{A}_L, \mathbf{A}_U]$ and cost vector $\mathbf{c}^\intercal = [\mathbf{c}_F^\intercal, \mathbf{c}_L^\intercal, \mathbf{c}_U^\intercal]$ accordingly. Although the change of variables is blindly applied, IPS retains only those pertinent to the construction:

$$\begin{aligned} \mathbf{y}_L &:= \mathbf{x}_L - \mathbf{x}_L^0, \;\; \mathbf{y}_L \geq \mathbf{0} \\ \mathbf{y}_U &:= \mathbf{x}_U^0 - \mathbf{x}_U, \;\; \mathbf{y}_U \geq \mathbf{0}. \end{aligned} \tag{3.22}$$

Let $N := L \cup U$ to form $\mathbf{y}_N = (\mathbf{y}_L; \mathbf{y}_U)$, the vector of currently null $\mathbf{y}$-variables, bounded above by $\mathbf{r}_N$, where $r_j := u_j - \ell_j, \;\forall j \in N$. Let $\mathbf{d}_N^\intercal := [\mathbf{c}_L^\intercal, -\mathbf{c}_U^\intercal]$ and define $\mathbf{A}_N^0 := [\mathbf{A}_L, -\mathbf{A}_U]$, that is, $\mathbf{a}_j^0 = \mathbf{a}_j, \;\forall j \in L$, and $\mathbf{a}_j^0 = -\mathbf{a}_j, \;\forall j \in U$. Given the adjusted right-hand side $\mathbf{b}^0 := \mathbf{b} - \mathbf{A}_L\mathbf{x}_L^0 - \mathbf{A}_U\mathbf{x}_U^0$, LP becomes:

$$\begin{array}{rlcccll} z^\star = \mathbf{c}_L^\intercal\mathbf{x}_L^0 + \mathbf{c}_U^\intercal\mathbf{x}_U^0 + & \min & \mathbf{c}_F^\intercal\mathbf{x}_F & + & \mathbf{d}_N^\intercal\mathbf{y}_N & & \\ & \text{s.t.} & \mathbf{A}_F\mathbf{x}_F & + & \mathbf{A}_N^0\mathbf{y}_N & = \mathbf{b}^0, \quad [\boldsymbol{\pi}] & \\ & & \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, & & \mathbf{0} \leq \mathbf{y}_N \leq \mathbf{r}_N. & & \end{array} \tag{3.23}$$

1 – Generic basis B, transformation $\mathbf{B}^{-1}$, row partition $\{P, Z\}$ of $\mathbf{A}_F$. The current solution being basic, the columns of $\mathbf{A}_F$ are linearly independent. When $f = m$, there is no row reduction but the current solution is nondegenerate, and so is the next pivot. Assume that $f < m$ such that the basis associated with $\mathbf{x}^0$ contains degenerate variables. Let us call *basis completion* the process of selecting $m - f$ variables taking value zero which complement $\mathbf{A}_F$ by forming a nonsingular basis matrix. Since any and all combinations of degenerate variables which may complete the basis is as good as the next one, let us construct a *generic* $m \times m$ basis denoted $\mathbf{B}$. Such a basis is readily available using the f free variables associated with

the columns of $\mathbf{A}_F$ together with $m - f$ artificial variables. The selection of an appropriate set of artificial variables can be done by solving a *restricted* primal simplex phase I problem over columns of $\mathbf{A}_F$ and those of the identity matrix $\mathbf{I}_m$ with the corresponding vector of artificial variables here denoted $\boldsymbol{\lambda}$:

$$\begin{array}{lrcl} \min & \mathbf{1}^\intercal\boldsymbol{\lambda} & & \\ \text{s.t.} \quad \mathbf{A}_F\mathbf{x}_F & + \mathbf{I}_m\boldsymbol{\lambda} & = & \mathbf{b}^0, \\ \mathbf{x}_F \geq \mathbf{0}, & \boldsymbol{\lambda} \geq \mathbf{0}. & & \end{array} \tag{3.24}$$

Solving this problem is undoubtedly successful in accordance with the fact that $\mathbf{A}_F\mathbf{x}_F^0 = \mathbf{b}^0$. Furthermore, this restricted phase I differs from a cold start phase I on one key point: only the former can guarantee a basis in which all degenerate basic variables are artificial ones. Let it be clear that this construction process identifies some subset $\mathbf{A}_{PF}$ of exactly f independent rows from matrix $\mathbf{A}_F$. This provides the row partition $\{P, \bar{P}\}$ of $\mathbf{A}_F$, where we use $Z := \bar{P}$ for notational convenience. The generic basis $\mathbf{B}$ and its inverse $\mathbf{B}^{-1}$ are as follows:

$$\mathbf{B} = \begin{bmatrix} \mathbf{A}_{PF} & \mathbf{0} \\ \mathbf{A}_{ZF} & \mathbf{I}_{m-f} \end{bmatrix} \quad \text{and} \quad \mathbf{B}^{-1} = \begin{bmatrix} \mathbf{A}_{PF}^{-1} & \mathbf{0} \\ -\mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1} & \mathbf{I}_{m-f} \end{bmatrix}, \tag{3.25}$$

where the matrix $\mathbf{A}_{PF}$ of dimension $f \times f$ is the working basis. The basis $\mathbf{B}$ is one of the many bases available to identify the over-represented vertex $\mathbf{x}^0$. As such, observe the sensitivity of the dual vector $\boldsymbol{\pi}^\intercal := \mathbf{c}_B^\intercal\mathbf{B}^{-1}$ with respect to the choice of basis completion. LP becomes

$$\begin{array}{rlrcll} z^\star = \mathbf{c}_L^\intercal\mathbf{x}_L^0 + \mathbf{c}_U^\intercal\mathbf{x}_U^0 + \min & \mathbf{c}_F^\intercal\mathbf{x}_F & + & \mathbf{d}_N^\intercal\mathbf{y}_N & & \\ \text{s.t.} & \mathbf{A}_{PF}\mathbf{x}_F & + & \mathbf{A}_{PN}^0\mathbf{y}_N & = \mathbf{b}_P^0, & [\boldsymbol{\pi}_P] \\ & \mathbf{A}_{ZF}\mathbf{x}_F & + & \mathbf{A}_{ZN}^0\mathbf{y}_N & = \mathbf{b}_Z^0, & [\boldsymbol{\pi}_Z] \\ & \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, & & \mathbf{0} \leq \mathbf{y}_N \leq \mathbf{r}_N. & & \end{array} \tag{3.26}$$

Let $\bar{\mathbf{b}}^0 := \mathbf{B}^{-1}\mathbf{b}^0$ and

$$\bar{\mathbf{A}}_N^0 := \mathbf{B}^{-1}\mathbf{A}_N^0 = \begin{bmatrix} \bar{\mathbf{A}}_{PN}^0 \\ \bar{\mathbf{A}}_{ZN}^0 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{PF}^{-1}\mathbf{A}_{PN}^0 \\ \mathbf{A}_{ZN}^0 - \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}\mathbf{A}_{PN}^0 \end{bmatrix}. \tag{3.27}$$

The new LP formulation obtained after the change of $\mathbf{y}$-variables and left-multiplication by the linear transformation $\mathbf{B}^{-1}$ of the set of equality constraints (which incidentally also

transforms the dual variables) makes degeneracy more evident:

$$\begin{array}{rrrrrrrl} z^\star = \mathbf{c}_L^\intercal \mathbf{x}_L^0 + \mathbf{c}_U^\intercal \mathbf{x}_U^0 + \min & & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{d}_N^\intercal \mathbf{y}_N & & & \\ \text{s.t.} & & \mathbf{x}_F & + & \bar{\mathbf{A}}_{PN}^0 \mathbf{y}_N & = & \bar{\mathbf{b}}_P^0, & [\boldsymbol{\psi}_P] \\ & & & & \bar{\mathbf{A}}_{ZN}^0 \mathbf{y}_N & = & \mathbf{0}, & [\boldsymbol{\psi}_Z] \\ & \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, & & & \mathbf{0} \leq \mathbf{y}_N \leq \mathbf{r}_N. & & & \end{array} \tag{3.28}$$

The current solution is given by $\mathbf{x}_F = \mathbf{x}_F^0 = \mathbf{A}_{PF}^{-1}\mathbf{b}_P^0 = \bar{\mathbf{b}}_P^0$ while $\mathbf{y}_N = \mathbf{0}$. Observe that the constraints of LP are divided according to the actual values of $\bar{\mathbf{b}}^0$: for the row set P, $\bar{\mathbf{b}}_P^0 > \mathbf{0}$; for the remaining rows in the set Z, $\bar{\mathbf{b}}_Z^0 = \mathbf{0}$. The dual vector $\boldsymbol{\pi}$ can be retrieved from the above transformed dual vector $\boldsymbol{\psi}$ using the expression $\boldsymbol{\pi}^\intercal = \boldsymbol{\psi}^\intercal \mathbf{B}^{-1}$:

$$\boldsymbol{\pi}_P^\intercal = \boldsymbol{\psi}_P^\intercal \mathbf{A}_{PF}^{-1} - \boldsymbol{\psi}_Z^\intercal \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1} \tag{3.29}$$

$$\boldsymbol{\pi}_Z^\intercal = \boldsymbol{\psi}_Z^\intercal. \tag{3.30}$$

2 – Compatibility with the row partition $\{P, Z\}$ of $\mathbf{A}_F$. Observe that any solution to (3.28), optimal or not, must satisfy $\bar{\mathbf{A}}_{ZN}^0 \mathbf{y}_N = \mathbf{0}$. This leads us to the first definition of compatibility.

Definition 3. *A vector $\mathbf{a} \in \mathbb{R}^m$ (and the associated variable, if any) is* compatible with the row partition $\{P, Z\}$ of $\mathbf{A}_F$ *if and only if* $\bar{\mathbf{a}}_Z := \mathbf{a}_Z - \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = \mathbf{0}$.

One can derive from the formulation (3.28) that the column vectors of $\mathbf{A}_F$ are compatible (hence the free variables $x_j,\ j \in F$) as well as the transformed right-hand side vector $\mathbf{b}^0$ (with no associated variable) but degenerate basic variables are not.

3 – Improved pricing step: maximize the minimum reduced cost μ. The variables $\mathbf{x}_F$ are basic in the row set P, hence the reduced cost vector $\bar{\mathbf{c}}_F = \mathbf{c}_F - \boldsymbol{\psi}_P = \mathbf{0}$ which means that $\boldsymbol{\psi}_P = \mathbf{c}_F$. With respect to the values of $\boldsymbol{\psi}_Z$, we recall the basis completion paradigm whereby the selection of degenerate variables that complete the basis influences the values of their associated dual variables. In other words, it is possible to capitalize on this freedom and consider them undetermined. The current solution $\mathbf{x}^0 = (\mathbf{x}_F^0; \mathbf{x}_L^0; \mathbf{x}_U^0)$ is optimal for (3.21), or equivalently $(\mathbf{x}_F; \mathbf{y}_N) = (\mathbf{x}_F^0; \mathbf{0})$ is optimal for (3.28), if there exists some dual vector $\boldsymbol{\psi}_Z$ such that the reduced cost $\bar{d}_j$ of every variable $y_j,\ j \in N$, is nonnegative, that is, $\bar{d}_j := d_j - \mathbf{c}_F^\intercal \bar{\mathbf{a}}_{Pj}^0 - \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{a}}_{Zj}^0 \geq 0,\ \forall j \in N$.

Let $\mu := \min_{j \in N} \bar{d}_j$ be the smallest reduced cost for $\mathbf{y}_N$ given $\boldsymbol{\psi}_P = \mathbf{c}_F$ but optimized over $\boldsymbol{\psi}_Z$. Finding μ can be formulated as a linear program:

$$\begin{array}{lrcll} \max & \mu & & & \\ \text{s.t.} & \mu & \leq & d_j - \mathbf{c}_F^\intercal \bar{\mathbf{a}}_{Pj}^0 - \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{a}}_{Zj}^0, & [y_j] \qquad \forall j \in N, \end{array} \tag{3.31}$$

where $y_j \geq 0,\ j \in N$, is the dual variable associated with the corresponding inequality constraint. Let $\tilde{d}_j := d_j - \mathbf{c}_F^\intercal \bar{\mathbf{a}}_{Pj}^0$ be the *partial reduced cost* of y_j computed by using the dual vector $\boldsymbol{\psi}_P = \mathbf{c}_F$, or equivalently $\tilde{\mathbf{d}}_N^\intercal := \mathbf{d}_N^\intercal - \mathbf{c}_F^\intercal \bar{\mathbf{A}}_{PN}^0$ in vector form. Therefore, (3.31) becomes

$$\begin{array}{lrcll} \max & \mu & & & \\ \text{s.t.} & \mathbf{1}\mu & + \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{A}}_{ZN}^0 & \leq \tilde{\mathbf{d}}_N, & [\mathbf{y}_N]. \end{array} \tag{3.32}$$

Taking the dual of (3.32), the pricing problem is written in terms of $\mathbf{y}_N$, the vector of currently null variables to price out:

$$\begin{array}{rlr} \mu = \min & \tilde{\mathbf{d}}_N^\intercal \mathbf{y}_N & \\ \text{s.t.} & \mathbf{1}^\intercal \mathbf{y}_N = 1, & [\mu] \\ & \bar{\mathbf{A}}_{ZN}^0 \mathbf{y}_N = \mathbf{0}, & [\boldsymbol{\psi}_Z] \\ & \mathbf{y}_N \geq \mathbf{0}. & \end{array} \tag{3.33}$$

The pricing problem (3.33) can be solved by the dual simplex algorithm because only the convexity constraint $\mathbf{1}^\intercal \mathbf{y}_N = 1$ is not satisfied by the current value $\mathbf{y}_N = \mathbf{0}$. For a more recent analysis of the resolution of the pricing problem, Omer et al. (2014) explore ways to warm start the basis notably with the use of more elaborate coefficients for the convexity constraints. Alternatively, specialized algorithms can be used in some applications. This is the case for LP defined as a capacitated minimum cost network flow problem where the pricing problem (3.33) corresponds to a minimum mean cost cycle problem which can be solved in $O(mn)$ time by dynamic programming (Karp 1978). What ultimately matters is that we are looking for extreme point solutions to (3.33) (see Gauthier et al. 2015b).

The number of positive variables in an optimal solution $\mathbf{y}_N^0$ to (3.33) is at most $m - f + 1$, the row dimension of the pricing problem. The solution $\mathbf{x}^0$ is optimal for LP if $\mu \geq 0$. Otherwise, $\mu < 0$ and $\mathbf{y}_N^0$ identifies a convex combination of columns such that $\bar{\mathbf{A}}_{ZN}^0 \mathbf{y}_N^0 = \mathbf{0}$. Observe that by Definition 3, the vector $\mathbf{A}_N^0 \mathbf{y}_N^0 \in \mathbb{R}^m$ is compatible with the row partition $\{P, Z\}$ of $\mathbf{A}_F$. Let Ω be the set of all such compatible convex combinations of columns.

4 – Exchange mechanism from $\mathbf{x}^0$ to $\mathbf{x}^1$. The solution $\mathbf{y}_N^0$ is utilized to move from $\mathbf{x}^0$ to $\mathbf{x}^1$. Let the compatible column $\mathbf{A}_N^0\mathbf{y}_N^0$ be associated with a surrogate variable θ_ω, $\omega \in \Omega$, nonexistent from the original formulation. The parameters of θ_ω relative to the formulation (3.28) are as follows: $\bar{\mathbf{a}}_w^0 = \begin{bmatrix} \bar{\mathbf{a}}_{Pw}^0 \\ \bar{\mathbf{a}}_{Zw}^0 \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{A}}_{PN}^0\mathbf{y}_N^0 \\ \mathbf{0} \end{bmatrix}$, reduced cost μ, cost $\mathbf{d}_N^\intercal\mathbf{y}_N^0$, and $\mathbf{y}_N^0 \neq \mathbf{0}$. With the addition of the variable θ_ω to the LP model in (3.28), we have the following relations, where the relevant parameters are indicated within brackets:

$$\begin{array}{rcl} \mathbf{x}_F & + & [\bar{\mathbf{a}}_{P\omega}^0]\ \theta_\omega = \bar{\mathbf{b}}_P^0, \\ \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, & & 0 \leq [\mathbf{y}_N^0]\ \theta_\omega \leq \mathbf{r}_N. \end{array} \tag{3.34}$$

Regardless of its solution, observe that the pricing problem finds a *partial* improving direction $\mathbf{y}_N^0$ of negative reduced cost value μ, if one exists, uniquely completed by $\mathbf{y}_F^0$, the impact in the row set P:

$$\begin{bmatrix} \mathbf{y}_F^0 \\ \mathbf{y}_N^0 \end{bmatrix} = \begin{bmatrix} -\bar{\mathbf{a}}_{P\omega}^0 \\ \mathbf{y}_N^0 \end{bmatrix} = \begin{bmatrix} -\bar{\mathbf{A}}_{PN}^0\mathbf{y}_N^0 \\ \mathbf{y}_N^0 \end{bmatrix} \in \mathbb{R}^n. \tag{3.35}$$

The step size ρ is governed by the usual pivot rule. In (3.34), the entering variable θ_ω can increase up to the maximum change for $\mathbf{y}_N$, that is, $\mathbf{y}_N^0\theta_\omega \leq \mathbf{r}_N$, or according to the maximum change for $\mathbf{x}_F$, that is, $\mathbf{l}_F \leq \bar{\mathbf{b}}_P^0 - \bar{\mathbf{a}}_{P\omega}^0\theta_\omega \leq \mathbf{u}_F$. The step size ρ on θ_ω is given by

$$\rho := \min\left\{ \min_{j\in N|y_j^0>0}\left\{\frac{r_j}{y_j^0}\right\}, \min_{i\in P|\bar{a}_{iw}^0>0}\left\{\frac{\bar{b}_i^0 - l_i}{\bar{a}_{iw}^0}\right\}, \min_{i\in P|\bar{a}_{iw}^0<0}\left\{\frac{u_i - \bar{b}_i^0}{-\bar{a}_{iw}^0}\right\}\right\}. \tag{3.36}$$

A nondegenerate pivot occurs ($\rho > 0$) and the objective function strictly improves by

$$\Delta z = \rho\mu = \rho\,\tilde{\mathbf{d}}_N^\intercal\mathbf{y}_N^0\ . \tag{3.37}$$

The solution $\mathbf{x}^0$ is updated to $\mathbf{x}^1$ according to the direction expression in (3.35):

$$\begin{array}{rcl} \mathbf{x}_F^1 := \mathbf{x}_F^0 & - & \rho\,\bar{\mathbf{a}}_{P\omega}^0 \\ \mathbf{x}_L^1 := \mathbf{x}_L^0 & + & \rho\,\mathbf{y}_L^0 \\ \mathbf{x}_U^1 := \mathbf{x}_U^0 & - & \rho\,\mathbf{y}_U^0\ . \end{array} \tag{3.38}$$

The number of free variables in $\mathbf{x}^1$ is at most $f + (m - f + 1) - 1 = m$, that is, the new solution can be more degenerate but it can also be less degenerate when several variables of the convex combination become free.

Regardless of the manner in which one updates the current solution, the aftermath is the result of an *exchange mechanism*. Even the ratio test performed to identify the exiting variable in PS echoes this notion. Indeed, the exchange always happens in a one-to-one fashion, while we have just seen that it can be more involved. Given the current solution, the exchange mechanism provided in (3.38) starts in the pricing problem (3.33) for the rows in the set Z by finding in $\bar{\mathbf{A}}^0_{ZN}\mathbf{y}_N = \mathbf{0},\ \mathbf{y}_N \geq \mathbf{0}$, which induces the partial directions $\mathbf{y}^0_L$ and $-\mathbf{y}^0_U$ for the vectors $\mathbf{x}_L$ and $\mathbf{x}_U$, respectively. The exchange process is afterward completed by using the rows in the set P and interval constraints in (3.34): the partial direction for the vector $\mathbf{x}_F$ is given by $-\bar{\mathbf{a}}^0_{P\omega} = -\bar{\mathbf{A}}^0_{PN}\mathbf{y}^0_N$ and the step size is derived in expression (3.36). In the latter, it occurs between $\mathbf{x}_F$ and the entering variable $\theta_\omega,\ \omega \in \Omega$.

5 – Update column partition $\{F, L, U\}$ In the midst of obtaining the new solution $\mathbf{x}^1$, every variable affected by the direction is identified. It is therefore easy to modify the status of each of these variables if necessary. Notice that the generic basis $\mathbf{B}$ is inspired by the column partition F.

Special case: $\mathbf{y}^0 = \mathbf{e}_j,\ j \in N$**.** The reader is invited to contemplate the special case where the convex combination contains a single variable $y^0_j = 1$, for some $j \in N$. The repercussions are many in terms of mathematical simplifications but we are most interested in the following one. The surrogate variable actually exists as is in the original formulation (3.28) which means that some existing variables in the set N are compatible with the row partition $\{P, Z\}$ of $\mathbf{A}_F$. In that case, the column $\mathbf{a}^0_j,\ j \in N$, enters the basis $\mathbf{B}$, $\mu = \tilde{d}_j$, and the step size ρ is computed according to the maximum increase of variable y_j. From (3.28), we have the following relations:

$$\begin{aligned} \mathbf{x}_F \quad &+ \quad \bar{\mathbf{a}}^0_{Pj} y_j \quad = \quad \bar{\mathbf{b}}^0_P, \\ \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, \quad & \quad 0 \leq y_j \leq r_j. \end{aligned} \tag{3.39}$$

The step size ρ on y_j can increase up to the upper bound r_j, or according to the maximum change in the vector of free variables $\mathbf{l}_F \leq \bar{\mathbf{b}}^0_P - \bar{\mathbf{a}}^0_{Pj} y_j \leq \mathbf{u}_F$:

$$\rho := \min \left\{ r_j, \min_{i \in P | \bar{a}^0_{ij} > 0} \left\{ \frac{\bar{b}^0_i - l_i}{\bar{a}^0_{ij}} \right\}, \min_{i \in P | \bar{a}^0_{ij} < 0} \left\{ \frac{u_i - \bar{b}^0_i}{-\bar{a}^0_{ij}} \right\} \right\} > 0 \,. \tag{3.40}$$

The objective function z improves by $\Delta z = \rho \tilde{d}_j = \rho\mu$. Either $j \in L$ (x_j is at its lower bound) or $j \in U$ (x_j is at its upper bound) and $\mathbf{x}^0$ is updated to $\mathbf{x}^1$ as

$$\begin{aligned} \mathbf{x}^1_F &:= \mathbf{x}^0_F - \rho\, \bar{\mathbf{a}}^0_{Pj} \\ \mathbf{x}^1_L &:= \mathbf{x}^0_L + \rho\, \mathbf{y}^0_L \\ \mathbf{x}^1_U &:= \mathbf{x}^0_U - \rho\, \mathbf{y}^0_U \,. \end{aligned} \tag{3.41}$$

The number of free variables in $\mathbf{x}^1$ is at most f, that is, the new solution can be more degenerate. If $\rho < r_j$, f decreases if more than one of the free variables reach their bounds. Otherwise $\rho = r_j$, the corresponding x_j variable changes bound and therefore stays degenerate in the new solution; the number of free variables decreases if at least one free variable reaches a bound.

3.2.2 Characterization of linear programming optimality

In summary, when $\mu \geq 0$, the current solution $\mathbf{x}^0$ is optimal. Otherwise, $\mu < 0$ and we obtain a strict improvement of the objective function, update the current solution from $\mathbf{x}^0$ to $\mathbf{x}^1$, and the process is repeated until the following *necessary and sufficient optimality condition* is met.

Proposition 16. *A basic feasible solution $\mathbf{x}^0 = (\mathbf{x}^0_F; \mathbf{x}^0_L; \mathbf{x}^0_U)$ is an optimal solution to the linear program (3.21) if and only if there exists a dual vector $\boldsymbol{\psi}_Z$ such that $\mu \geq 0$, as optimized by the primal-dual pair (3.32)–(3.33) of the pricing problem.*

Proof. The formulations (3.21) and (3.28) are equivalent. Because $\bar{\mathbf{c}}_F = \mathbf{0}$, if there exists some dual vector $\boldsymbol{\psi}_Z$ such that $\mathbf{d}^\intercal_N - \mathbf{c}^\intercal_F \bar{\mathbf{A}}^0_{PN} - \boldsymbol{\psi}^\intercal_Z \bar{\mathbf{A}}^0_{ZN} \geq \mathbf{0}^\intercal$, $N := L \cup U$, then $(\bar{\mathbf{c}}_F, \bar{\mathbf{d}}_N) \geq \mathbf{0}$. Therefore, $\boldsymbol{\psi}^\intercal = (\mathbf{c}^\intercal_F, \boldsymbol{\psi}^\intercal_Z)$ provides a feasible dual solution to (3.28). Since $\boldsymbol{\psi}^\intercal_P \bar{\mathbf{b}}^0_P = \mathbf{c}^\intercal_F \mathbf{x}^0_F$, the primal and dual objective functions are equal and the current feasible solution $\mathbf{x}^0$ is optimal for (3.21).

To show the converse, let $\mathbf{x}^0$ be an optimal solution to (3.21) and assume that $\mu < 0$. An optimal solution to the pricing problem (3.33) identifies a convex combination of variables

such that a nondegenerate pivot occurs ($\rho > 0$) and the objective function strictly improves by $\rho\mu < 0$. This contradicts the optimality of $\mathbf{x}^0$ and completes the proof. □

All simplex derivatives work according to the presumption of innocence. Optimality is indeed assumed until proven otherwise. It is no different in IPS, yet it is an amazing feat that the content of the pricing problem be reminiscent of the no more, no less punch line. The sufficient condition answers to the first part, while the necessary condition to the second.

3.2.3 IPS for a linear program in standard form

The reader may recall that incorporating lower and upper bounds in PS adds a plethora of intricacies in the algorithmic analysis. Although the same is true of IPS, we assumed the reader was sufficiently accustomed with the traditional algorithm. In the spirit of conveying the general idea of IPS, it might be worthwhile to present a simpler version. This basically amounts to removing the dimension U from the formulation. The simplifications are threesome and correspond to the main steps of IPS: creating the column and row partitions, building the pricing problem, and modifying the current solution. Given LP in standard form

$$\begin{array}{rlll} z^\star := & \min & \mathbf{c}^\intercal \mathbf{x} & \\ & \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b}, & [\boldsymbol{\pi}] \\ & & \mathbf{x} \geq \mathbf{0}, & \end{array} \tag{3.42}$$

and a feasible solution $\mathbf{x}^0 = (\mathbf{x}_F^0; \mathbf{x}_N^0)$, the column partition step distinguishes between the currently nondegenerate (or free) basic vector $\mathbf{x}_F^0$ and null vector $\mathbf{x}_N^0$:

$$\begin{array}{rlllll} z^\star = \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + \mathbf{c}_N^\intercal \mathbf{x}_N & & \\ \text{s.t.} & \mathbf{A}_F \mathbf{x}_F & + \mathbf{A}_N \mathbf{x}_N = \mathbf{b}, & & [\boldsymbol{\pi}] \\ & \mathbf{x}_F \geq \mathbf{0}, & \mathbf{x}_N \geq \mathbf{0}. & & \end{array} \tag{3.43}$$

Recall the previous change of variables in (3.22). Since N now only contains variables at their lower bounds, $\mathbf{x}_N$ could be used interchangeably with $\mathbf{y}_N$. We maintain the general presentation to underscore that the construction aims to find an improving direction induced

by $\mathbf{y}_N$. It should also be clear that $\mathbf{A}_N^0 = \mathbf{A}_N$, and $\mathbf{b}^0 = \mathbf{b}$.

$$\begin{array}{rrcrcll} z^\star = \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{c}_N^\intercal \mathbf{y}_N & & & \\ \text{s.t.} & \mathbf{x}_F & + & \bar{\mathbf{A}}_{PN}\mathbf{y}_N & = & \bar{\mathbf{b}}_P, & [\boldsymbol{\psi}_P] \\ & & & \bar{\mathbf{A}}_{ZN}\mathbf{y}_N & = & \mathbf{0}, & [\boldsymbol{\psi}_Z] \\ & \mathbf{x}_F \geq \mathbf{0}, & & \mathbf{y}_N \geq \mathbf{0}. & & & \end{array} \tag{3.44}$$

Once again, the linear transformation $\mathbf{B}^{-1}$ performed on the original system underlines the degeneracy of the current solution. Furthermore, any solution must satisfy $\bar{\mathbf{A}}_{ZN}\mathbf{y}_N = \mathbf{0}$ in (3.44). Therefore, the pricing problem can be written in terms of the vector of null variables to price out, and the current partial reduced cost vector $\tilde{\mathbf{c}}_N^\intercal := \mathbf{c}_N^\intercal - \boldsymbol{\psi}_P^\intercal \bar{\mathbf{A}}_{PN} = \mathbf{c}_N^\intercal - \mathbf{c}_F^\intercal \bar{\mathbf{A}}_{PN}$:

$$\begin{array}{rrcll} \mu := \min & \tilde{\mathbf{c}}_N^\intercal \mathbf{y}_N & & & \\ \text{s.t.} & \mathbf{1}^\intercal \mathbf{y}_N & = & 1, & [\mu] \\ & \bar{\mathbf{A}}_{ZN}\mathbf{y}_N & = & \mathbf{0}, & [\boldsymbol{\psi}_Z] \\ & \mathbf{y}_N & \geq & \mathbf{0}. & \end{array} \tag{3.45}$$

The solution $\mathbf{x}^0 = (\mathbf{x}_F^0 > \mathbf{0}; \mathbf{x}_N^0 = \mathbf{0})$ is optimal for LP in (3.42) if $\mu \geq 0$. Otherwise $\mu < 0$ and an optimal solution $\mathbf{y}_N^0$ to (3.45) identifies a convex combination of variables such that $\bar{\mathbf{A}}_{ZN}\mathbf{y}_N^0 = \mathbf{0}$. The convex combination established by the pricing problem may once again contain one or several $\mathbf{y}$-variables. Let θ_ω, $\omega \in \Omega$, be the entering variable with the following parameters: reduced cost μ, cost $\mathbf{c}_N^\intercal \mathbf{y}_N^0$, and $\bar{\mathbf{a}}_\omega = \begin{bmatrix} \bar{\mathbf{a}}_{P\omega} \\ \bar{\mathbf{a}}_{Z\omega} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{A}}_{PN}\mathbf{y}_N^0 \\ \mathbf{0} \end{bmatrix}$. What matters is that the ratio test (3.36) is now computed with a single component:

$$\rho := \min_{i \in P | \bar{a}_{i\omega} > 0} \left\{ \frac{\bar{b}_i}{\bar{a}_{i\omega}} \right\} > 0. \tag{3.46}$$

A nondegenerate pivot occurs and LP's objective in (3.42) strictly improves by $\Delta z = \rho\,\mu$. Finally, $\mathbf{x}^0$ is updated to $\mathbf{x}^1$ as follows:

$$\begin{array}{rcrl} \mathbf{x}_F^1 := & \mathbf{x}_F^0 & - & \rho\,\bar{\mathbf{a}}_{P\omega} \\ \mathbf{x}_N^1 := & & & \rho\,\mathbf{y}_N^0. \end{array} \tag{3.47}$$

3.2.4 Numerical example

Table 3.2 depicts a linear program in standard form comprising eight $\mathbf{x}$-variables and six constraints. The degenerate solution $\mathbf{x}^0$ is already presented in the simplex tableau format: $(x_1^0, x_2^0, x_3^0) = (30, 25, 50)$ are the positive (or free) basic variables and the basis has been completed with artificial $\boldsymbol{\lambda}$-variables in rows 4, 5 and 6. The cost of this solution is $z^0 = 185$.

	x_1	x_2	x_3	λ_4	λ_5	λ_6	x_4	x_5	x_6	x_7	x_8			θ_ω
$\mathbf{c}$	2	3	1				10	17	-20	14	-4			-5
	1						2	2	1	-5	7	=	30	1
		1					4	3	-5	10	-10	=	25	-2
			1				-3	1	2	3	11	=	50	5
				1			0	0	6	5	-13	=	0	0
					1		0	0	3	4	-8	=	0	0
						1	0	0	3	-4	0	=	0	0
$\mathbf{x}^0$	30	25	50									$z^0 =$	185	
$\tilde{\mathbf{c}}$							-3	3	-9	-9	1			-6

Tab. 3.2: The simplex tableau at $\mathbf{x}^0$

The vector $\mathbf{c}_F^\intercal = \begin{bmatrix}2, 3, 1\end{bmatrix}$, equal to the dual vector for the top rows, is used for computing the partial reduced cost vector $\tilde{\mathbf{c}}_N^\intercal = \begin{bmatrix}\text{-3, 3, -9, -9, 1}\end{bmatrix}$. By inspection, we see that x_4 and x_5 are compatible with the row partition derived from the right-hand side values. One can observe that the associated columns are (trivial) combinations of the (unit) vectors of the free variables x_1, x_2 and x_3.

Both compatible variables would provide a nondegenerate pivot if chosen as entering variables but only x_4 has a negative partial reduced cost value $\tilde{c}_4 = \text{-3}$ (which is also equal to its reduced cost $\bar{c}_4$). The incompatible variables x_6 and x_7 possess a negative partial reduced cost value of -9 whereas $\tilde{c}_8 = 1$. The selection of incompatible variable x_6 or x_7 would result in a degenerate pivot while that of x_8 would increase the objective function by $1 \times (\frac{30}{7})$.

However, solving the pricing problem (3.45) over the last three rows results in a combination of the incompatible vectors with the following weights: $(y_6^0, y_7^0, y_8^0) = (0.4, 0.3, 0.3)$. This provides the compatible vector $\bar{\mathbf{a}}_\omega^\intercal = \begin{bmatrix}1 & \text{-2} & 5 & 0 & 0 & 0\end{bmatrix}$ for the variable θ_ω of reduced cost $\mu = \text{-9}(0.4) + \text{-9}(0.3) + 1(0.3) = \text{-6}$ and cost -5. The ratio test on the top three rows results in $\rho = \min\left\{\frac{30}{1}, -, \frac{50}{5}\right\} = 10$. The entering variable θ_ω takes the value 10 and provides a strict improvement of the objective function of $\text{-6} \times 10 = \text{-60}$. As a result, x_3 goes out of the basis, and other free variables x_1 and x_2 are respectively updated to 20 and 45. Alternatively, the variables x_6, x_7 and x_8 can be entered one by one in the basis, in any order, and this produces the same result. In the new solution of cost 125, the positive

variables are $(x_1, x_2, \theta_\omega) = (20, 45, 10)$ or equivalently for $\mathbf{x}^1$ in terms of the original variables, $(x_1, x_2, x_6, x_7, x_8) = (20, 45, 4, 3, 3)$ while x_3, x_4 and x_5 are null variables.

From the five columns corresponding to the positive variables, the first five rows are independent and the artificial variable λ_6 is basic with a zero value in the last row. The inverse basis $\mathbf{B}^{-1}$ at $\mathbf{x}^1$ appears in Table 3.3 and is used to construct the next degenerate simplex tableau in Table 3.4.

$$\mathbf{B} = \left[\begin{array}{ccccc|c} 1 & & 1 & -5 & 7 & \\ & 1 & -5 & 10 & -10 & \\ & & 2 & 3 & 11 & \\ & & 6 & 5 & -13 & \\ & & 3 & 4 & -8 & \\ \hline & & 3 & -4 & 0 & 1 \end{array}\right] \quad \mathbf{B}^{-1} = \left[\begin{array}{ccccc|c} 1 & & -0.20 & -2.133 & 4.067 & \\ & 1 & 0.40 & 5.600 & -9.800 & \\ & & 0.08 & 0.453 & -0.627 & \\ & & 0.06 & -0.327 & 0.613 & \\ & & 0.06 & 0.007 & -0.053 & \\ \hline 0 & 0 & 0 & -2.667 & 4.333 & 1 \end{array}\right]$$

Tab. 3.3: The basis $\mathbf{B}$ and its inverse at $\mathbf{x}^1$

	x_1	x_2	x_6	x_7	x_8	λ_6	x_3	x_4	x_5		
$\mathbf{c}$	2	3	-20	14	-4		1	10	17		
	1						-0.20	2.60	1.80	=	20
		1					0.40	2.80	3.40	=	45
			1				0.08	-0.24	0.08	=	4
				1			0.06	-0.18	0.06	=	3
					1		0.06	-0.18	0.06	=	3
						1	0	0	0	=	0
$\mathbf{x}^1$	20	45	4	3	3					$z^1 =$	125
$\tilde{\mathbf{c}}$							1.2	-6.6	4.2		

Tab. 3.4: The simplex tableau at $\mathbf{x}^1$

$\mathbf{A}_{PF}^{-1}$, the inverse of the working basis within $\mathbf{B}^{-1}$, is used to compute the dual vector $\mathbf{c}_F^\intercal \mathbf{A}_{PF}^{-1} = \begin{bmatrix} 2, & 3, & -0.2, & -1.133, & 0.067 \end{bmatrix}$ of the row set P and partial reduced costs $(\tilde{c}_3, \tilde{c}_4, \tilde{c}_5) =$ (1.2, -6.6, 4.2). Moreover,

$$\bar{\mathbf{a}}_{Zj} = -\mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}\mathbf{a}_{Pj} + \mathbf{a}_{Zj} = \mathbf{0}, \ j \in \{3, 4, 5\},$$

characterizes column compatibility by computing

$$\begin{bmatrix} \bar{\mathbf{a}}_{Z3} & \bar{\mathbf{a}}_{Z4} & \bar{\mathbf{a}}_{Z5} \end{bmatrix} = (0, 0, 0, -2.667, 4.333) \begin{bmatrix} \mathbf{a}_{P3} & \mathbf{a}_{P4} & \mathbf{a}_{P5} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}.$$

The null variables x_3, x_4 and x_5 are compatible with the current row partition, and the optimal solution to the pricing problem at iteration 1 is $y_4^1 = 1$: x_4 enters the basis, being the only one with a negative reduced cost of -6.6. The ratio test on the top five rows results

in $\rho = \min\left\{\frac{20}{2.6}, \frac{45}{2.8}, -, -, -\right\} = 7.692$ and the entering variable x_4 provides an objective function improvement of -6.6 × 7.692 = -50.769. The variable x_1 goes out of the basis, and updated free variables x_2, x_6, x_7 and x_8 appear in Table 3.5, here presented in terms of the simplex tableau at $\mathbf{x}^2$ before being updated. Observe that the actual combination of variables x_6, x_7 and x_8 satisfies the last three rows at zero right-hand side. The cost of this solution is $z^2 = 74.231$.

	x_4	x_2	x_6	x_7	x_8	λ_6	x_3	x_1	x_5		
$\mathbf{c}$	10	3	-20	14	-4		1	2	17		
	2		1	-5	7			1	2	=	30
	4	1	-5	10	-10				3	=	25
	-3		2	3	11		1		1	=	50
			6	5	-13					=	0
			3	4	-8					=	0
			3	-4	0	1				=	0
$\mathbf{x}^2$	7.692	23.462	5.846	4.385	4.385					$z^2 =$	74.231
$\tilde{\mathbf{c}}$							0.692	2.538	8.769		

Tab. 3.5: The simplex tableau at $\mathbf{x}^2$ before being updated

$\mathbf{B}^{-1}$ for $\mathbf{x}^2$ appears in Table 3.6 from which $\mathbf{c}_F^\intercal \mathbf{A}_{PF}^{-1} = \begin{bmatrix}-0.538, 3, 0.308, 4.282, -10.256\end{bmatrix}$ is computed and the partial reduced costs $(\tilde{c}_3, \tilde{c}_1, \tilde{c}_5) = (0.692, 2.538, 8.769)$. Since these are positive, $\mathbf{x}^2$ is optimal.

$$\mathbf{B}^{-1} = \left[\begin{array}{ccccc|c} 0.385 & & -0.077 & -0.821 & 1.564 & \\ -1.077 & 1 & 0.615 & 7.897 & -14.179 & \\ 0.092 & & 0.062 & 0.256 & -0.251 & \\ 0.069 & & 0.046 & -0.474 & 0.895 & \\ 0.069 & & 0.046 & -0.141 & 0.228 & \\ \hline 0 & 0 & 0 & -2.667 & 4.333 & 1 \end{array}\right]$$

Tab. 3.6: The inverse basis $\mathbf{B}^{-1}$ at $\mathbf{x}^2$

3.3 Linear Algebra Framework

To appreciate the generality of IPS, the reader is invited to consider its presentation only borrows from the algebraic manipulations of PS. The linear algebra framework is put forth to derive another way to look at the row/column partition. Section 3.3.1 introduces the vector subspace $\mathbf{V}(\mathbf{A}_F)$ spanned by the column vectors of $\mathbf{A}_F$. This is followed in Section 3.3.2 by the practical use of an equivalent subspace basis $\mathbf{\Lambda}_f$. In Section 3.3.3, we examine a different subspace basis, $\mathbf{\Lambda}_r$, of possibly larger dimension $r \geq f$ that is sufficient to span $\mathbf{A}_F$.

Section 3.3.4 discusses the pitfalls of this more general subspace basis and a modified algorithm is given in Section 3.3.5. For the record, the vector subspace notion is first mentioned in Benchimol et al. (2012) for the implementation of a stabilized DCA algorithm for the set partitioning problem.

3.3.1 Vector subspace $\mathbf{V}(\mathbf{A}_F)$

The concept of compatibility is contextual by nature since it assumes a row partition $\{P, \bar{P}\}$ of $\mathbf{A}_F$, where $\bar{P} = Z$. The reader might have observed that we have taken the liberty to omit this precision outside the definition of compatibility. It turns out that this omission works well in our favor. The following result holds the explanation (Desrosiers et al. 2014) whereas Proposition 18 presents an alternative definition of compatibility which is impervious to the partition.

Proposition 17. *Let $\mathbf{A}_{PF}$ and $\mathbf{A}_{QF}$ be two working bases identifying different row partitions of $\mathbf{A}_F$. If the vector $\mathbf{a}$ is compatible with partition $\{P, \bar{P}\}$ then it is also compatible with $\{Q, \bar{Q}\}$. Hence, we say* $\mathbf{a}$ is compatible with $\mathbf{A}_F$.

Proof. Assume the vector $\mathbf{a}$ is compatible with the partition $\{P, \bar{P}\}$ and consider the following relation on the set Q:

$$\mathbf{A}_{PF}^{-1}\mathbf{a}_P = \mathbf{A}_{QF}^{-1}\mathbf{a}_Q \quad \Longleftrightarrow \quad \mathbf{a}_Q - \mathbf{A}_{QF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = \mathbf{0}. \tag{3.48}$$

The right part is verified for every component $i \in Q$: true for $i \in Q \cap \bar{P}$ since the vector $\mathbf{a}$ is compatible whereas for $i \in Q \cap P$, $a_i - \mathbf{A}_{iF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = a_i - a_i = 0$. Hence,

$$\mathbf{a}_{\bar{Q}} - \mathbf{A}_{\bar{Q}F}\mathbf{A}_{QF}^{-1}\mathbf{a}_Q = \begin{cases} a_i - \mathbf{A}_{iF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = a_i - a_i = 0 & \forall i \in \bar{Q} \cap P \\ a_i - \mathbf{A}_{iF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = 0 & \forall i \in \bar{Q} \cap \bar{P}, \end{cases} \tag{3.49}$$

the last equality being true since the vector $\mathbf{a}$ is compatible with the partition $\{P, \bar{P}\}$. □

Proposition 18. *A vector $\mathbf{a} \in \mathbb{R}^m$ (and the associated variable, if any) is* compatible with $\mathbf{A}_F$ *if and only if it belongs to $\mathbf{V}(\mathbf{A}_F)$.*

Proof. We first show that if Definition 3 is satisfied for some partition $\{P, Z\}$ then the statement rings true. We then show that the converse is also true. Assume that the vector $\mathbf{a}$ is compatible such that $\bar{\mathbf{a}}_Z = \mathbf{a}_Z - \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}\mathbf{a}_P = \mathbf{0}$. Let $\boldsymbol{\alpha} := \mathbf{A}_{PF}^{-1}\mathbf{a}_P$. Then,

$\begin{bmatrix}\mathbf{a}_P\\ \mathbf{a}_Z\end{bmatrix} = \begin{bmatrix}\mathbf{A}_{PF}\boldsymbol{\alpha}\\ \mathbf{A}_{ZF}\boldsymbol{\alpha}\end{bmatrix}$ meaning that the vector $\mathbf{a}$ indeed belongs to $\mathbf{V}(\mathbf{a}_F)$. Let us now assume that there exists some $\boldsymbol{\alpha} \in \mathbb{R}^f$ such that the vector $\mathbf{a}$ is a linear combination of the column vectors of $\mathbf{A}_F$. Since $\mathbf{A}_F$ is a subspace basis, there exists some row set P such that $\mathbf{A}_{PF}$ is invertible. Then, $\boldsymbol{\alpha} = \mathbf{A}_{PF}^{-1}\mathbf{a}_P$ and compatibility of the vector $\mathbf{a}$ follows. □

A consequence of Proposition 18 is that every subset $\boldsymbol{\Lambda}_f$ of f independent vectors of $\mathbf{V}(\mathbf{A}_F)$ can be used as a subspace basis for $\mathbf{V}(\mathbf{A}_F)$. Let us explicitly recall the definition of a vector basis as a linearly independent spanning set. A simple but important observation is the following: The set of f independent vectors of $\mathbf{A}_F$ identified in IPS is therefore a *minimal* spanning set capable of representing the current solution, $\mathbf{A}_F\mathbf{x}_F^0 = \mathbf{b}^0$. Indeed, the very construction of the working basis in $\mathbf{B}$ implies that $\mathbf{A}_F$ spans $\mathbf{b}^0$, that is, $\mathbf{x}_F^0 = \mathbf{A}_{PF}^{-1}\mathbf{b}_P^0$, see the system of linear equations in (3.26) or (3.28).

3.3.2 Subspace basis $\boldsymbol{\Lambda}_f$

The identification of the working basis is one of the bottleneck operations of IPS. Furthermore, as the reader can observe from the formulation (3.28), it is useless to multiply the row set P by $\mathbf{A}_{PF}^{-1}$ to identify the improving variable θ_ω, $\omega \in \Omega$, if any. Indeed, only $\bar{\mathbf{a}}_{P\omega}$ needs to be computed to perform the ratio test (3.36). An alternative set to $\mathbf{A}_F$ of f independent vectors that spans $\mathbf{V}(\mathbf{A}_F)$ is $\boldsymbol{\Lambda}_f = \begin{bmatrix}\mathbf{I}_f\\ \mathbf{M}\end{bmatrix}$, where $\mathbf{M} = \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1}$. Together with $\boldsymbol{\Lambda}_f^\perp = \begin{bmatrix}\mathbf{0}\\ \mathbf{I}_{m-f}\end{bmatrix}$, it provides the basis $\mathbf{T} := \left[\boldsymbol{\Lambda}_f, \boldsymbol{\Lambda}_f^\perp\right]$ of $\mathbb{R}^m$ and its inverse:

$$\mathbf{T} = \begin{bmatrix}\mathbf{I}_f & \mathbf{0}\\ \mathbf{M} & \mathbf{I}_{m-f}\end{bmatrix} \quad \text{and} \quad \mathbf{T}^{-1} = \begin{bmatrix}\mathbf{I}_f & \mathbf{0}\\ -\mathbf{M} & \mathbf{I}_{m-f}\end{bmatrix}. \tag{3.50}$$

The LP formulation obtained after the change of variables and the transformation by the more practical $\mathbf{T}^{-1}$ results in an equivalent system for which only the rows in set Z are transformed:

$$\begin{aligned}
z^\star = \mathbf{c}_L^\intercal\mathbf{x}_L^0 + \mathbf{c}_U^\intercal\mathbf{x}_U^0 & & & & & & & \\
+\min \quad & \mathbf{c}_F^\intercal\mathbf{x}_F & + & \mathbf{d}_N^\intercal\mathbf{y}_N & & & & \\
\text{s.t.} \quad & \mathbf{A}_{PF}\mathbf{x}_F & + & \mathbf{A}_{PN}^0\mathbf{y}_N & = & \mathbf{b}_P^0, & [\boldsymbol{\psi}_P] \\
& & & \bar{\mathbf{A}}_{ZN}^0\mathbf{y}_N & = & \mathbf{0}, & [\boldsymbol{\psi}_Z] \\
& \mathbf{l}_F \le \mathbf{x}_F \le \mathbf{u}_F, & & \mathbf{0} \le \mathbf{y}_N \le \mathbf{r}_N, & & & &
\end{aligned} \tag{3.51}$$

where $\bar{\mathbf{A}}^0_{ZN} = \mathbf{A}^0_{ZN} - \mathbf{M}\mathbf{A}^0_{RN}$. Similarly to (3.29) and (3.30), $\boldsymbol{\pi}$ can be retrieved from the dual vector $\boldsymbol{\psi}$ in (3.51) using the expression $\boldsymbol{\pi}^\intercal = \boldsymbol{\psi}^\intercal \mathbf{T}^{-1}$:

$$\left[\boldsymbol{\pi}_P^\intercal, \boldsymbol{\pi}_Z^\intercal\right] = \left[\boldsymbol{\psi}_P^\intercal - \boldsymbol{\psi}_Z^\intercal \mathbf{M},\ \boldsymbol{\psi}_Z^\intercal\right]. \tag{3.52}$$

When all is said and done, using vector subspace properties enables one to derive a working basis using any and all efficient methods to extract an equivalent subspace basis. Furthermore, depending on the application, the inverse is implicitly obtained in $\mathbf{M}$ as a by-product of the decomposition. Of course, having access to the LP solver's own LU-decomposition would be quite practical. Note that although $\mathbf{B}$ constructed in (3.25) is implicitly considered as a simplex basis in IPS, $\mathbf{T}$ is more generally defined as a basis in $\mathbb{R}^m$, that is, an invertible linear transformation in $\mathbb{R}^m$.

3.3.3 Subspace basis $\boldsymbol{\Lambda}_r, r \geq f$

Let us consider the general situation where r, the dimension of the subspace basis $\boldsymbol{\Lambda}_r$ spanning the columns of $\mathbf{A}_F$, is larger than or equal to f, the number of free variables. Assume $\boldsymbol{\Lambda}_r$ includes the f columns of $\mathbf{A}_F$ and $r - f \geq 0$ additional columns such that these r columns are linearly independent. Using a restricted phase I, one identifies r independent rows in the subset $R \subseteq \{1, \ldots, m\}$ and the subspace basis can take the form $\boldsymbol{\Lambda}_r = \begin{bmatrix} \mathbf{I}_r \\ \mathbf{M} \end{bmatrix}$, where $\mathbf{M}$ is an $(m-r) \times r$ matrix, whereas $\boldsymbol{\Lambda}_r^\perp = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}_{m-r} \end{bmatrix}$. Let $\mathbf{V}(\boldsymbol{\Lambda}_r)$ be the vector subspace spanned by $\boldsymbol{\Lambda}_r$. At the end of the day, the definition of compatibility can be enlarged to the spanning set of the chosen subspace basis.

Definition 4. *A vector* $\mathbf{a} \in \mathbb{R}^m$ *(and the associated variable, if any) is* compatible with $\boldsymbol{\Lambda}_r$ *if and only if it belongs to* $\mathbf{V}(\boldsymbol{\Lambda}_r)$.

3.3.4 Words of caution about compatibility

Once the general form of the subspace basis $\boldsymbol{\Lambda}_r$ is retained, it is delicate to still claim this modified version as IPS. If the latter can be seen as a *poorer* vector subspace which obviously includes $\mathbf{A}_F$, the added granularity provided by the superfluous columns yields a denser compatible set.

The danger of over-spanning $\mathbf{A}_F$ is that a compatible surrogate variable $\theta_\omega,\ \omega \in \Omega$, found by the pricing problem does not guarantee a strictly improving pivot. Indeed, any value $\bar{\mathbf{a}}^0_{i\omega} \neq 0,\ i \in R$, corresponding to $\bar{b}^0_i = 0,\ i \in R$, is potential cause for a zero step size, hence

a degenerate pivot. Observe that the magnitude of the value $\bar{\mathbf{a}}^0_{i\omega}$ is irrelevant, what really matters is its sign. In a very superficial sense, the probability of making a degenerate pivot thus increases exponentially by one half for every extra row, i.e., $1 - (1/2)^{r-f}$. When $r > f$, the probability ranges from one half to almost surely very rapidly. For that matter, even an incompatible variable might induce a nondegenerate pivot with probability $(1/2)^{m-f}$. The reader is invited to take a look at the variable x_8 in the numerical example of Section 3.2.4 to be convinced of the nondegenerate potential. Of course, a more refined analysis of the probabilities would include the configuration of matrix $\mathbf{A}$ and at this point falls outside the purpose of this paper.

In most if not all literature surrounding IPS and its derivatives, the concept of compatibility is associated with a nondegenerate pivot. While it is true that in the purest form of IPS, a compatible variable necessarily induces a nondegenerate pivot, the implication of the previous paragraph denies this synonymy for the general form of the subspace basis, as it stands the implemented version. What does this all mean? The linear algebra framework that surrounds IPS provides a more robust definition of compatibility. As the latter gains in flexibility, it loses in certainty. Compatibility provides a way to categorize variables by their capacity to induce nondegenerate pivots with fair accuracy. We guess researchers have taken the liberty to address one for the other because of the intent behind the partition scheme. A leap of faith comes to mind.

To sum up, this larger subspace basis $\mathbf{\Lambda}_r$ breaks away from the strictly improving pivot construction of IPS. It is however a necessary evil that gives a lot of freedom in the implementation and, more importantly, closes the theoretical gap between IPS and DCA.

3.3.5 Modified IPS algorithm

Figure 3.12 contains the modifications necessary to include the linear algebra framework to the vanilla version of IPS. In Step 1, the construction of the working basis uses the representation $\mathbf{T}$. For Step 2, the compatible set is constructed with the alternative Definition 4. Steps 3 and 4 rely on the modified row partition $\{R, \bar{R}\}$, but their essence remains otherwise untouched and therefore see no particular caveat. Neither does Step 5.

3.4 Aiming for efficiency

This section serves the practical side of an implementation of IPS. The fourth step of IPS, namely the exchange mechanism, brings the solution of the improved pricing step back to

1 Generic basis $\mathbf{T}$, transformation $\mathbf{T}^{-1}$, row partition $\{R, \bar{R}\}$ of $\mathbf{\Lambda}_r$;
2 Compatibility with the row partition $\{R, \bar{R}\}$ of $\mathbf{\Lambda}_r$ <optional>;
3 Improved pricing step: maximize the minimum reduced cost μ;
4 Exchange mechanism from $\mathbf{x}^0$ to $\mathbf{x}^1$;
5 Set $\mathbf{x}^0 := \mathbf{x}^1$, update the column partition $\{F, L, U\}$ and goto Step 1;

Fig. 3.12: Modified IPS algorithmic steps

what can be called a *control system*. This is indeed where feasibility is maintained by using the flexibility of the free variables and the interval bounds otherwise omitted from the pricing step. With that being said, this process of information sharing between the pricing step and the control system is quite close to a master problem/subproblem paradigm. In fact, with a better understanding of the pricing step, we argue in Section 3.4.1 that IPS corresponds to dynamically applying a Dantzig-Wolfe reformulation (Dantzig and Wolfe 1960) at every iteration, the row partition being done according to the current solution vector $\mathbf{x}^0$ given by $[\mathbf{x}^0_F; \mathbf{x}^0_L; \mathbf{x}^0_U]$.

This interpretation of IPS can result in very flexible resolution strategies. Among these is the usage of the convex combination $\omega \in \Omega$ and its surrogate variable θ_ω opposed to the column components of ω and their respective original $\mathbf{x}$-variables. We also know from column generation that generating several columns during one iteration of the pricing step is highly efficient. In line with this idea also comes that of using heuristics to solve the pricing problem during the early stage of the resolution process. The fourth and perhaps most important idea defers to the time consuming task of updating the row partition. Such is the content of the three subsequent subsections (Sections 3.4.2, 3.4.3 and 3.4.4) which examine these various ways to accelerate IPS. Section 3.4.5 presents the dynamic Dantzig-Wolfe implementation of IPS while Section 3.4.6 shares computational results gathered from different papers.

3.4.1 Dynamic Dantzig-Wolfe decomposition

We now present an interpretation of IPS in terms of a decomposition scheme proposed by Metrane et al. (2010) for standard linear programs. Here is an adaptation for the bounded case.

Consider a Dantzig-Wolfe decomposition of the previous so-called compact formulation (3.51) which has a block angular structure. The equality constraints in the row set P together with the interval constraints $\mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F$ and upper bounds $\mathbf{y}_N \leq \mathbf{r}_N$ stay in the master problem structure. The equality constraints in the row set Z and the nonnegativity

constraints $\mathbf{y}_N \geq \mathbf{0}$ form the subproblem domain:

$$\mathcal{SP} := \{\mathbf{y}_N \geq \mathbf{0} \mid \bar{\mathbf{A}}^0_{ZN}\mathbf{y}_N = \mathbf{0}\}. \tag{3.53}$$

The Dantzig-Wolfe decomposition builds on the representation theorems by Minkowski and Weyl (see Schrijver 1986, Desrosiers and Lübbecke 2011) that any vector $\mathbf{y}_N \in \mathcal{SP}$ can be reformulated as a convex combination of extreme points plus a nonnegative combination of extreme rays of $\mathcal{SP}$. Moreover, $\mathcal{SP}$ is a cone for which the only extreme point is the null vector $\mathbf{y}_N = \mathbf{0}$ at zero cost. Since this extreme point does not contribute to the master problem constraints, it can as such be discarded from the reformulation. The vector $\mathbf{y}_N$ can thus be expressed as a nonnegative combination of the extreme rays $\{\mathbf{y}^\omega_N\}_{\omega\in\Omega}$:

$$\mathbf{y}_N = \sum_{\omega\in\Omega} \mathbf{y}^\omega_N \theta_\omega, \;\; \theta_\omega \geq 0, \quad \forall \omega \in \Omega.$$

Substituting in the master problem structure, LP becomes:

$$\begin{array}{rrrrrrl} z^\star = \mathbf{c}_L^\intercal \mathbf{x}^0_L + \mathbf{c}_U^\intercal \mathbf{x}^0_U + \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \sum_{\omega\in\Omega}[\mathbf{d}_N^\intercal \mathbf{y}^\omega_N]\theta_\omega & & & \\ \text{s.t.} & \mathbf{A}_{PF}\mathbf{x}_F & + & \sum_{\omega\in\Omega}[\mathbf{A}^0_{PN}\mathbf{y}^\omega_N]\theta_\omega & = & \mathbf{b}^0_P, & [\boldsymbol{\psi}_P] \\ & & & \sum_{\omega\in\Omega}[\mathbf{y}^\omega_N]\theta_\omega & \leq & \mathbf{r}_N, & \\ & \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, & & \theta_\omega \geq 0, \;\forall \omega \in \Omega. & & & \end{array} \tag{3.54}$$

At any iteration of IPS, none of the θ-variables are yet generated and the inequality constraints in (3.54) are not binding. Therefore, the dual vector for these constraints is null and the reduced cost of variable $\theta_\omega, \;\omega \in \Omega$, is given by:

$$[\mathbf{d}_N^\intercal \mathbf{y}^\omega_N] - \boldsymbol{\psi}_P^\intercal [\mathbf{A}^0_{PN}\mathbf{y}^\omega_N] = (\mathbf{d}_N^\intercal - \boldsymbol{\psi}_P^\intercal \mathbf{A}^0_{PN})\mathbf{y}^\omega_N = \tilde{\mathbf{d}}_N^\intercal \mathbf{y}^\omega_N,$$

where $\tilde{\mathbf{d}}_N$ is the partial reduced cost vector already used in IPS, see formulation (3.32). Now, any negative reduced cost ray in $\mathcal{SP}$ results in the same subproblem minimum objective value, that is, $-\infty$. However, observe that for any nonzero solution in the cone defined by $\mathcal{SP}$ in (3.53), there exists a scaled one such that $\mathbf{1}^\intercal \mathbf{y}_N = 1$. Therefore, without loss of generality, the domain of the subproblem can be rewritten as

$$\mathcal{SP}_N := \{\mathbf{y}_N \geq \mathbf{0} \mid \bar{\mathbf{A}}^0_{ZN}\mathbf{y}_N = \mathbf{0}, \;\; \mathbf{1}^\intercal \mathbf{y}_N = 1\}. \tag{3.55}$$

Hence, an equivalent subproblem in this Dantzig-Wolfe decomposition, searching for a negative reduced cost column until optimality is reached, is exactly the one defined by the primal pricing problem (3.33) in IPS:

$$\min \ \tilde{\mathbf{d}}_N^{\intercal}\mathbf{y}_N \quad \text{s.t.} \qquad \mathbf{y}_N \in \mathcal{SP}_N. \tag{3.56}$$

The bottleneck of this algorithm is the improved pricing step. Recall that the content of the latter is a ripple effect of the decomposition choice. These ideas can therefore be separated in two categories: the first supports the idea that the master problem and the pricing step are communicating vessels, the second is solely aimed at the pricing step in an effort to find solutions, not necessarily optimal, faster. Before moving on to the three subsections which examine the aforementioned various ways to accelerate IPS, let us recall how the master problem may be fed with surrogate variables or their original column vector content.

Convex combination vs. column components. The weights $\mathbf{y}_N^0$ dictate the content of the convex combination ω and ascertain the compatibility requirement of the entering variable θ_ω. By neglecting these weights, the original column components of the convex combination can be fed directly to the compact formulation (3.51) along with their associated original $\mathbf{x}$-variables. While this certainly seems counterproductive for the one direction, let us go through the mechanics for the sake of argument. Discarding the weights also implies that the compatible faith of this group of columns is lost. The active constraints in the pricing problem must therefore also be passed to the compact formulation. The latter can then obviously be solved to $\mathbf{x}_N = \mathbf{y}_N^0\theta_\omega$, a process that leads to the same objective value as would pivoting θ_ω. The column components mechanics has the potential to shine when one thinks of a column generation framework where multiple columns are brought back to the restricted master problem. In this perspective, the original column components fed to the compact formulation could be arranged with their siblings from other directions at different levels thus granting more freedom than the surrogate variables provide. Similar techniques to solve large-scale linear multi-commodity flow problems were previously used by Löbel (1998) and Mamer and McBride (2000), whereas Valério de Carvalho (1999, 2002) propose a network-based compact formulation of the cutting stock problem in which the classical knapsack subproblem is solved as a shortest path problem. In all these applications, the compact formulation is written in terms of arc flow variables. When a subproblem generates a *path* with a negative reduced cost, the *arcs* of this path are iteratively added to the compact formulation. This process allows the implicit combination of arcs into paths without having to generate these. Sadykov and Vanderbeck (2013) describe this in generality.

3.4.2 Subspace basis update

Postponing the subspace basis update can be taken advantage of on two fronts: before and after updating to the new solution $\mathbf{x}^1$. On the first front, it is indeed a basic idea to harvest more information from the pricing problem than the one iteration. Let this agenda be known as *multiple improving directions*. We present two specific scenarios before the general one. The first scenario is the particular case of *independent improving directions* while the second is the *compatible restricted master problem*. On the second front, from the Dantzig-Wolfe mindset, it becomes clear that entering an improving variable θ_w in the master problem (3.54) does not necessarily warrant an update of the subspace basis. In either case, it is in effect a matter of manipulating the dual variables. The price to pay is the possibility of making degenerate pivots on some of these directions.

Independent improving directions. IPS relies on the strictly improving property of the algorithm to guarantee that the exchange mechanism goes through the components of θ_ω with a strictly positive step size, see (3.36). If two variables θ_{ω_1} and θ_{ω_2} can be identified from the pricing problem such that compatibility is obtained from orthogonal vectors of $\mathbf{V}(\mathbf{A}_F)$, then $\bar{\mathbf{a}}^0_{\omega_1}$ and $\bar{\mathbf{a}}^0_{\omega_2}$ are independent from each other and can be added to the current solution in any order both yielding a predictable improvement. Independence constraints need not be added to the pricing problem in order to carry out this strategy, it suffices to remove variables that already *contribute* in the first direction of the variable θ_{ω_1}. Indeed, the selection of columns the latter contains should of course be removed from the pricing problem. Among themselves and variables of $\mathbf{x}_F$ used to complete the direction, these columns have nonzero elements on several rows, i.e., they contribute on each of these rows. Any variable that sports a nonzero value on any of these same rows shares a contribution and can therefore be removed from the pricing problem. In other words, removing every variable that contributes to the aforementioned rows amounts, for all intents and purposes, to discarding these constraints as well.

Compatible restricted master problem (RMP_{FC})**.** Consider the row partition $\{R, Z\}$ of $\mathbf{\Lambda}_r$, where $Z = \bar{R}$. By Definition 4, the columns of $\mathbf{A}_F$ are compatible with $\mathbf{\Lambda}_r$. Denote by $\mathbf{A}^0_C$, $C \subseteq N$, the columns of $\mathbf{A}^0_N$ compatible with $\mathbf{\Lambda}_r$. Any of these can easily be identified in $O(m)$ time using PE, see Section 3.7. Let $\mathbf{A}_I$ be the incompatible columns, $I := N \setminus C$. Using $\mathbf{T} = [\mathbf{\Lambda}_r, \mathbf{\Lambda}_r^\perp]$ as a basis of $\mathbb{R}^m$ and applying the transformation $\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{I}_r & \mathbf{0} \\ -\mathbf{M} & \mathbf{I}_{m-r} \end{bmatrix}$ on the formulation (3.23), we have $\bar{\mathbf{a}}^0_{Zj} = \bar{\mathbf{b}}^0_Z = \mathbf{0},\ \forall j \in F \cup C$. Let $\bar{\mathbf{A}}^0_{ZI} := \mathbf{A}^0_{ZI} - \mathbf{M}\mathbf{A}^0_{RI}$.

LP becomes

$$\begin{array}{rrrrrrrrl} z^\star = & \mathbf{c}_L^\intercal \mathbf{x}_L^0 + \mathbf{c}_U^\intercal \mathbf{x}_U^0 & & & & & & & \\ + \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{d}_C^\intercal \mathbf{y}_C & + & \mathbf{d}_I^\intercal \mathbf{y}_I & & & \\ \text{s.t.} & \mathbf{A}_{RF}\mathbf{x}_F & + & \mathbf{A}_{RC}^0 \mathbf{y}_C & + & \mathbf{A}_{RI}^0 \mathbf{y}_I & = & \mathbf{b}_R^0, & [\boldsymbol{\psi}_R] \\ & & & & & \bar{\mathbf{A}}_{ZI}^0 \mathbf{y}_I & = & \mathbf{0}, & [\boldsymbol{\psi}_Z] \end{array} \tag{3.57}$$
$$\mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, \quad \mathbf{0} \leq \mathbf{y}_C \leq \mathbf{r}_C, \quad \mathbf{0} \leq \mathbf{y}_I \leq \mathbf{r}_I.$$

Restricting the formulation (3.57) to the columns in the set $F \cup C$ (the compatible variables) yields the compatible restricted master problem RMP_{FC} defined on the row set R, much easier to solve than (3.21) as it involves fewer variables and, more importantly, fewer constraints. As such, it is less subject to degeneracy.

Of course, its optimal solution does not necessarily solve LP. It is equivalent to having exhausted the pricing step of all improving compatible variables one iteration after another without having updated the subspace basis. Whether one should update the latter now or retrieve more directions from the pricing step is arguably the exact same question as one would face after the *first* direction is retrieved from the complete pricing step.

Multiple improving directions. To have access to valid combinations of incompatible columns in $\mathbf{A}_I^0$, the Dantzig-Wolfe decomposition obtained from $\mathbf{x}^0$ is maintained. Keeping in the subproblem the equalities from the row set Z, the nonnegativity requirements $\mathbf{y}_I \geq \mathbf{0}$, and a scaling constraint on $\mathbf{y}_I$ leads to the following formulation.

$$\min \ \tilde{\mathbf{d}}_I^\intercal \mathbf{y}_I \quad \text{s.t.} \qquad \mathbf{y}_I \in \mathcal{SP}_I := \left\{ \mathbf{y}_I \geq \mathbf{0} \mid \bar{\mathbf{A}}_{ZI}^0 \mathbf{y}_I = \mathbf{0}, \ \mathbf{1}^\intercal \mathbf{y}_I = 1 \right\}. \tag{3.58}$$

As previously derived in Section 3.4.1, the substitution of the extreme rays generated from $\mathcal{SP}_I$, $\mathbf{y}_I^\omega$, $\omega \in \Omega$, into the master problem gives

$$\begin{array}{rrrrrrrrl} z^\star = & \mathbf{c}_L^\intercal \mathbf{x}_L^0 + \mathbf{c}_U^\intercal \mathbf{x}_U^0 & & & & & & & \\ + \min & \mathbf{c}_F^\intercal \mathbf{x}_F & + & \mathbf{d}_C^\intercal \mathbf{y}_C & + & \sum_{\omega\in\Omega} [\mathbf{d}_I^\intercal \mathbf{y}_I^\omega]\theta_\omega & & & \\ \text{s.t.} & \mathbf{A}_{RF}\mathbf{x}_F & + & \mathbf{A}_{RC}^0 \mathbf{y}_C & + & \sum_{\omega\in\Omega} [\mathbf{A}_{RI}^0 \mathbf{y}_I^\omega]\theta_\omega & = \mathbf{b}_R^0, & [\boldsymbol{\psi}_R] \\ & & & & & \sum_{\omega\in\Omega} [\mathbf{y}_I^\omega]\theta_\omega & \leq \mathbf{r}_N, & & \end{array} \tag{3.59}$$
$$\mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, \quad \mathbf{0} \leq \mathbf{y}_C \leq \mathbf{r}_C, \quad \theta_\omega \geq 0, \ \forall \omega \in \Omega,$$

and similarly to (3.29)–(3.30) or (3.52), the dual vector $\boldsymbol{\pi}$ can be retrieved using the expression $\boldsymbol{\pi}^\intercal = \boldsymbol{\psi}^\intercal \mathbf{T}^{-1}$:

$$\left[\boldsymbol{\pi}_R^\intercal, \boldsymbol{\pi}_Z^\intercal\right] = \left[\boldsymbol{\psi}_R^\intercal - \boldsymbol{\psi}_Z^\intercal \mathbf{M},\ \boldsymbol{\psi}_Z^\intercal\right]. \tag{3.60}$$

Ranging from heuristically modifying the dual variables to discarding certain $\mathbf{y}$-variables or type of solutions, extracting many interesting directions from the pricing problem is then a matter of creativity. Notice that using all the compatible variables in RMP_{FC} is one such heuristic.

Postponing subspace basis update past $\mathbf{x}^1$. Once again, the row partition is only the fruit of a linear transformation $\mathbf{T}^{-1}$ at a given iteration. We argue that using surrogate variables allows to reuse the previous subspace basis because it is simply the result of the particular Dantzig-Wolfe decomposition partitioning the rows into $\{R, \bar{R}\}$. Unfortunately, when more than one former free variable becomes degenerate, the old subspace basis now spans degenerate basic variables. It is therefore possible to maintain the old subspace basis but it implies the use of the more general form $\boldsymbol{\Lambda}_r$. In accordance with Section 3.3.3, we state that an update is in order when the actual number of free variables $|F|$ is relatively different from $|R|$, the row size of the master problem.

3.4.3 *Vector subspace flexibility*

Given that the vector subspace is defined with respect to the matrix of free variables $\mathbf{A}_F$, this section shows that it is even possible to play with the set of free variables as we see fit. The first trick cheats the free status with algebraic manipulations while the other considers a particular type of upper bounds.

Coerced degeneracy. Another highly important concept is that of *coerced* degeneracy. This is used in the capacitated minimum cost network flow problem which can artificially render any current free variable into two degenerate ones on the residual network, see Ahuja et al. (1993). Indeed, an arc variable x_{ij} taking a value $\ell_{ij} < x_{ij}^0 < u_{ij}$ on the original network formulation can be replaced by two variables representing upward ($0 \leq y_{ij} \leq u_{ij} - x_{ij}^0$) and downward ($0 \leq y_{ji} \leq x_{ij}^0 - \ell_{ij}$) possible flow variations. The current values of these $\mathbf{y}$-variables is null and again this can modify the relative row sizes of the master and the pricing problems. On either count, the choice of the vector subspace results in a degenerate free pricing step.

Implicit upper bounds. Some applications have a structure that *implicitly* bounds some variables by the sheer force of the technological constraints. For instance, the assignment and the set partitioning models have such a feature. As a matter of fact, all variables in both of these problems are bounded above by 1, yet the explicit bound needs not be added to the formulation. That is to say that a variable x_j features an *implicit* upper bound u_j if $x_j > u_j$ is infeasible regardless of the values of the remaining variables.

Taking upper bounds into account is an obligatory path to guarantee strictly improving directions. We argue that, in presence of implicit upper bounds, IPS can be applied in two different manners with respect to the way these upper bounds are taken into account. In the first case, upper bounds are stated in the formulation whereas the second case omits them altogether. Assume the variable $x_j = u_j$ has reached its implicit upper bound. In the explicit formulation, when an upper bound is reached, it is taken into account thus sending the variable x_j in the pricing problem. In the silenced formulation, a variable $x_j = u_j$ would be *assumed* free.

Since the bound is implicit, it should be obvious that both pricing problems should yield only nondegenerate directions. It is trivial in the first case since the upper bound is explicitly taken into account. In the second case, it must be shown that the pricing problem cannot identify a direction that would increase the variable x_j. The fact that $x_j \leq u_j$ is implicit from the set of technological constraints translates into the coefficients of the θ_ω-variables in (3.34) as these can be derived from the Dantzig-Wolfe reformulation in (3.54), *equivalent to the original formulation*. Therefore, one finds the following equality constraint for x_j when it reaches its implicit upper bound:

$$x_j + [\bar{a}^0_{j\omega}]\theta_\omega = u_j,$$

where $\theta_\omega \geq 0,\ \forall \omega \in \Omega$. Since $x_j \leq u_j$, the coefficients $\left[\bar{a}^0_{j\omega}\right] \geq 0,\ \forall \omega \in \Omega$, and the *assumed* free variable x_j can only decrease during the exchange mechanism.

The main difference between these two choices echoes the vector subspace $\mathbf{V}(\mathbf{A}_F)$ and thus the set of compatible variables, see Proposition 18. Consider the vector subspaces spanned by $\mathbf{A}_F$ which contains or not implicitly bounded variables. The distinction lies in the compatibility set and different $\mathbf{A}_F$ modify the relative row sizes of the master (f) and the subproblem $(m - f + 1)$. The added granularity provided by the additional vectors in the first case creates a denser linear span and thus allows more variable compatibility.

Observe that variables in N are always treated correctly by the pricing problem since they are observably at one of their bounds. The extension of this result is that a variable in N

could be *assumed* to be free, if it can be shown that its current value is an implicit upper bound. While some very preliminary results are available in Section 3.4.6 with respect to implicit bounds formulations, both the theoretical and practical implications have yet to be explored meaningfully. This concept even has an impact within the scope of PS; a variable at its implicit bound could be either basic or nonbasic in the former case whereas it would necessarily be basic in the second.

3.4.4 Partial pricing

In this subsection, we discuss possible partial pricing choices to accelerate the resolution process of the pricing step without compromising optimality. That is, as long as the last iteration uses the complete model, intermediate pricing steps can use heuristic notions. Partial pricing strategies become appealing in diversified aspects. For example, one can use various subsets of compatible and incompatible variables to reduce the density of the pricing problem. We present three such biases: partial cost, residual capacity, and rank-incompatibility. These ideas can of course be mixed as deemed worthy.

Partial cost bias. An incompatible variable $j \in N$ can be temporarily discarded if its partial reduced cost $\tilde{d}_j$ is greater than some threshold value, the most simple one being zero.

Residual capacity bias. The idea of this bias is to guarantee a minimum step size ρ. One look at the modified ratio test (3.36) suffices to see that the residual capacity bias also involves free variables. On the one hand, we want to keep the variable $j \in N$ if its residual capacity r_j is relatively large. On the other hand, the coerced degeneracy principle must be used on free variables to keep only those where both values $\bar{b}^0_i - l_i$ and $u_i - \bar{b}^0_i$, $i \in R$ are large. Since the ratio test also depends on $\bar{a}^0_{i\omega}$, it makes this guarantee all the more difficult to appreciate on arbitrary matrices $\mathbf{A}$. Nevertheless, the idea works well when it is embedded in the minimum mean cycle-canceling algorithm, an extreme case of row partition where $|F| = 0$ since coerced degeneracy is applied on *all* free variables. Observe that once the coerced degeneracy is applied, it might be possible to keep one of the coerced free variables in the pricing problem.

Rank-incompatibility bias. Another possibility is to define the pricing step against *rank-incompatibility*. This means that the incompatible variables are attributed a rank according to the degree of incompatibility they display. The pricing problem sequentially uses lower

rank incompatible variables. Intuitively, the point is not to stray too far from the current compatibility definition and thus limit the perturbation caused by modifying it. This concept is first seen under the name *Multi-phase DCA* (MPDCA) in the paper of Elhallaoui et al. (2010).

3.4.5 Dynamic Dantzig-Wolfe algorithm

Figure 3.13 presents the implemented version of IPS inspired by the dynamic Dantzig-Wolfe construction. The first two steps recuperate the linear algebra work. The biggest modification thus entails the work done in Step 3 which is now broken down into smaller components. The first utilizes RMP_{FC} by solving a row-reduced master problem with compatible variables only. The second calls the pricing problem where the dual multipliers are updated and only incompatible variables remain. The latter can be solved several times to retrieve many possibly improving directions using whatever arsenal available to the user to accomplish said task. Finally, the exchange mechanism in Step 5 can be applied to every predetermined direction, yet it is simpler to let a PS code create a new working basis using all the gathered information simultaneously. The reason for this are threefold. First, it ensures that the solution $\mathbf{x}^1$ is basic. Second, it fetches updated dual multipliers for the row set R in case the generic basis is not updated. Third, it allows for a possibly better solution than the sequential work. With this new solution $\mathbf{x}^1$, the algorithm loops and the partition may (goto Step 1) or may not (goto Step 3b) (in the surrogate variable environment) be updated. The pricing step will be influenced by new dual variables either way.

1 Generic basis $\mathbf{T}$, transformation $\mathbf{T}^{-1}$, row partition $\{R, \bar{R}\}$ of $\mathbf{\Lambda}_r$;
2 Compatibility with the row partition $\{R, \bar{R}\}$ of $\mathbf{\Lambda}_r$ <optional>;
3a Restricted master problem: solve RMP_{FC} to optimality <optional goto Step 5>;
3b Improved pricing step: maximize the minimum reduced cost μ <optional repeat>;
4 Exchange mechanism from $\mathbf{x}^0$ to $\mathbf{x}^1$;
5 Set $\mathbf{x}^0 := \mathbf{x}^1$, update the column partition $\{F, L, U\}$ and goto Step 1 <optional goto Step 3b instead>;

Fig. 3.13: Dynamic Dantzig-Wolfe algorithmic steps

3.4.6 Computational results for IPS

As the reader might have guessed, these ideas must be meticulously handled in a practical implementation of IPS for it to be competitive. The computational results for the latter have

therefore been postponed until this point. Linear programs in standard form have been used for the comparison between IPS and CPLEX's PS.

The two main ideas used to obtain these results are the subspace basis update along with the compatible restricted master problem and the multiple improving directions. On 10 instances involving 2,000 constraints and up to 10,000 variables for *simultaneous vehicle and crew scheduling problems in urban mass transit systems* (VCS), IPS reduces CPU times by a factor of 3.53 compared to CPLEX's PS (Elhallaoui et al. 2011, Raymond et al. 2010b). These set partitioning problems have degeneracy levels of about 50%. IPS is also tested on 14 instances of *aircraft fleet assignment* (FA). These consist in maximizing the profits of assigning a type of aircraft to each flight segment over a horizon of one week. The content of the multi-commodity flow formulation for each of these instances can be resumed with these ballpark figures: a degeneracy level of 65%, 5,000 constraints and 25,000 variables. IPS reduces CPU times by a factor of 12.30 on average compared to CPLEX's PS. While both types of problems are solved by column generation, the IPS methodology is tested by saving thousands of variables from the generator obviously including the optimal ones.

In these fleet assignment problems, an upper bound of 1 can explicitly be imposed on arc flow variables (see the discussion in Section 3.4.3). Hence, degeneracy occurs for basic variables at 0 or at 1. The comparison is still done against CPLEX's PS but the upper bounds are explicitly added in both solvers. CPU times are reduced by a factor of 20.23 on average for these LPs (Raymond et al. 2009). These IPS algorithms have yet to be compared together.

On another note, opposing the convex combination to its column components content has been tested as follows. Computational experiments conducted with a hybrid algorithm starting with the classical generated columns (the surrogate variables) for the restricted master problem and ending with the column components (the original $\mathbf{x}$-variables) for the compact formulation shows improving average factors of 3.32 and 13.16 compared to CPLEX's PS on the previously mentioned VCS and FA problems (Metrane et al. 2010).

3.5 Designing around compatibility

As supported by the vector subspace and the subspace basis flexibility, the compatibility notion is indeed quite flexible. In fact, when solving particular linear programs, the existing specialized algorithms, devised within the confines of IPS, that have proved to be successful share the common trait of being designed around and for compatibility. Sections 3.5.1 and 3.5.2 respectively address network and set partitioning problems.

3.5.1 Network flow

In the context of the capacitated minimum cost flow problem, one refers to a solution $\mathbf{x}$ as a cycle free solution if the network contains no cycle composed only of free arcs, see Ahuja et al. (1993). Any such solution can be represented as a collection of free arcs (the nondegenerate basic arcs forming a forest) and all other arcs at their lower or upper bounds. The column vectors of the free arcs form $\mathbf{A}_F$, see Figure 3.14.

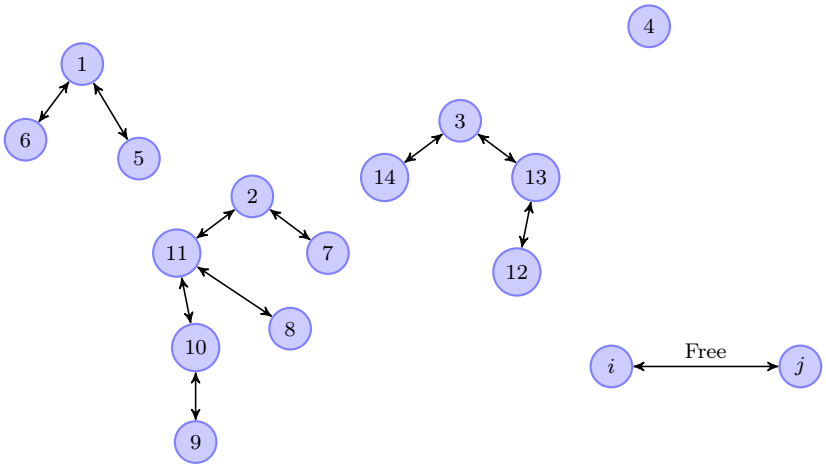


Fig. 3.14: Forest of free arcs in $\mathbf{A}_F$ on a residual network

According to Proposition 18 and the flow conservation equations, an arc at its lower or upper bound is compatible if and only if it can be written in terms of the unique subset of free arcs forming a cycle with it (Desrosiers et al. 2014). Therefore, a compatible arc simply links two nodes belonging to the same tree of the forest. By opposition, an incompatible arc links two nodes of two different trees.

This characterization allows us to better understand the mechanism of improving cycles in networks. A feasible solution is optimal if and only if there is no negative cost directed cycle on the residual network. Two types of cycles can result from the pricing problem: a cycle containing a single compatible arc together with some free arcs of the same tree, or a cycle containing at least two incompatible arcs together with possibly some free arcs from different trees of the forest.

In Figure 3.15, the dotted arc $(8, 9)$ is compatible and forms a directed cycle with the free arcs $(9, 10)$, $(10, 11)$, and $(11, 8)$. Indeed, the associated column in rows 8 through 11 are such that

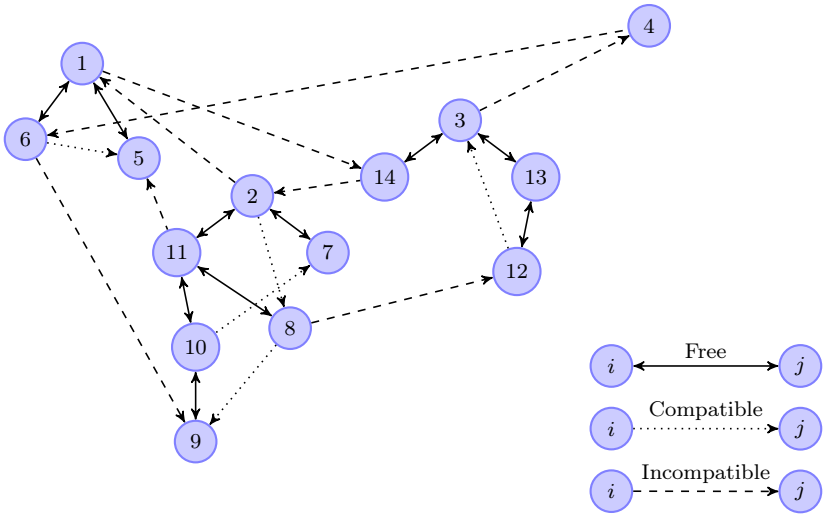


Fig. 3.15: Compatibility characterization of degenerate arcs on a residual network

$$\begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \end{bmatrix} = - \begin{bmatrix} 0 \\ 1 \\ -1 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

The dashed arc $(6, 9)$ links two different trees and is therefore incompatible. This is also the case for some other arcs, e.g., $(8, 12)$, $(3, 4)$ and $(4, 6)$. The reader may verify that the sum of the associated four columns is compatible as it can be written as the negated sum of the columns associated with the free arcs $(9, 10)$, $(10, 11)$, $(11, 8)$ and $(12, 13)$, $(13, 3)$. Indeed, these nine arcs form a directed cycle in the residual network.

Since IPS only makes nondegenerate pivots, it converges to optimality in a finite number of iterations on integral data network flow problems. Desrosiers et al. (2013) show that IPS is strongly polynomial for binary network problems, e.g., assignment, single shortest path, and unit capacity maximum flow. With a slight modification, it becomes strongly polynomial for solving the capacitated minimum cost network flow problem. The proposed contraction-expansion IPS-based algorithm is similar to the minimum mean cycle-canceling algorithm, see Goldberg and Tarjan (1989), Radzik and Goldberg (1994). On a network comprising n nodes and m arcs, it performs $O(m^2 n)$ iterations and runs in $O(m^3 n^2)$ time for arbitrary real-valued arc costs.

3.5.2 Set partitioning

The set partitioning problem (SPP) can be formulated as the binary linear program

$$\min \quad \mathbf{c}^\intercal \mathbf{x} \quad \text{s.t.} \qquad \mathbf{A}\mathbf{x} = \mathbf{1}, \quad \mathbf{x} \in \mathbb{B}^n, \tag{3.61}$$

where $\mathbf{A} \in \mathbb{B}^{m \times n}$. This formulation can be seen as a generic model encountered in various applications, namely, in vehicle routing and crew scheduling, and many more where the aim is to perform a clustering of the rows. In such applications, each set partitioning constraint can be associated with a task $i \in \{1, \ldots, m\}$ that must be *covered* exactly once. Such a formulation arises naturally from applying a Dantzig-Wolfe reformulation to a multi-commodity flow model in which each vehicle or crew member is represented by a separate commodity, see Desrosiers et al. (1995) and Desaulniers et al. (1998).

In order to express the fundamental exchange mechanism of set partitioning solutions, we assume that the current vector $\mathbf{x}_F$ is binary. Figure 3.16 should help demystify the concept of compatibility on SPP.

$\mathbf{A}_F$			x_4	x_5	x_6	x_7	x_8	x_9
1			1	1				
1			1		1			
	1			1		1		1
	1				1	1	1	
	1				1		1	1
		1	1				1	1

Fig. 3.16: Compatibility characterization for set partitioning binary solution $\mathbf{x}_F$

On the left-hand side, we find the binary input solution defined by the three independent columns of $\mathbf{A}_F$. According to Proposition 18, the next column identified by x_4 is compatible with the given partition as it covers exactly the first and third clusters. The third set shows the two incompatible columns x_5 and x_6. None can be generated by the columns of $\mathbf{A}_F$. However, their addition is compatible with the given partition as it covers the first and second clusters of rows. Finally, the right-hand side set exhibits three incompatible columns, x_7, x_8 and x_9: their combination with equal weights of $1/2$ is compatible as it covers the second and third clusters of the row partition. Notice that this combination breaks the integrality of the next solution.

The compatible columns are readily available using the spanning set $\mathbf{A}_F$ as per Proposition 18: a binary column is compatible if and only if it covers some of the clusters. Therefore,

the interpretation of compatibility can be seen as a question of coverage. When the selected column is a combination of incompatible columns, the exchange mechanism removes elements from some clusters to insert them back in other clusters.

When the input solution is fractional, the mathematical definition of compatibility (Definition 3 or Proposition 18) still holds but the interpretation loses practical meaning. In order to sidestep this unfortunate loss, we can fall back on $\mathbf{\Lambda}_r$ (Definition 4) and adjust the subspace basis interpretation with respect to an aggregation/clustering scheme. The idea is to assume that certain tasks are done together and it is the cornerstone of DCA as described in the following section. Historically speaking, DCA is a self-standing algorithm devised for set partitioning models which provides an easy way to define a specialized vector subspace that *often* shares the properties of the one designed for IPS. The next section discusses the differences and similarities that arise between the two methods. We insist that it is in retrospective that the ties between DCA and IPS have been better understood.

3.6 Dynamic Constraint Aggregation

It is the first time DCA and IPS are studied in parallel. While they share several similarities, we hope to dissolve the confusion that arises between the two theories by highlighting their differences. IPS relies on the linear algebra framework to ascertain its faith and is therefore constructive by nature. It turns out that DCA is also born from a constructive design. This design is however limited by the embryonic intuition of a *reduced basis*. Let it be said that DCA is an intuitive precursor to IPS.

In a nut shell, the differences spring forth from the choice of the vector subspace to represent the current solution. Recall the subspace basis $\mathbf{\Lambda}_f$ and the equivalent generic transformation $\mathbf{T}^{-1}$, DCA disregards this choice and uses the general subspace basis format. It constructs $\mathbf{\Lambda}_r,\ r \geq f$, large enough to span $\mathbf{A}_F$. Let us see how and why it performs well.

In Section 3.6.1, we derive a row partition using a simple construction. The method is then applied in Section 3.6.2 on a set partitioning problem. The *inexistent* pricing step of DCA is explained in Section 3.6.3. An overview of the algorithm is illustrated in Section 3.6.4. Section 3.6.5 meditates on the integrality dimension of SPP. Finally, computational results are summarized in Section 3.6.6.

3.6.1 $\mathbf{\Lambda}_r$ derived from the identical rows of $\mathbf{A}_F$

The idea behind DCA is similar to the first step of a LU-decomposition for $\mathbf{A}_F$. Some of the rows which can easily be identified as null entries after elimination are actually identical rows. Of course, such a strategy might propose a set of constraints where some rows are redundant because linear independence is not thoroughly verified. Nevertheless, the separation between unique and identical rows induces a partition of the rows. The size of the partition is expressed as *the number of clusters* $r \geq f$ *of identical rows in* $\mathbf{A}_F$. Consider the following generic example where $\mathbf{A}_F$ contains six rows distributed into three clusters. The first step ($\mapsto$) consists of a permutation of the lines such that the top rows are unique and the bottom rows are duplicates. The second step ($\equiv$) provides a subspace basis $\mathbf{\Lambda}_r$ where each row cluster is associated with a unique 1-column identifier. Observe that by construction the top rows always correspond to I_r in the vector subspace, hence the subspace basis is of the form $\mathbf{\Lambda}_r = \left[\begin{smallmatrix}\mathbf{I}_r \\ \mathbf{M}\end{smallmatrix}\right]$:

$$\overset{\mathbf{A}_F}{\begin{pmatrix}\mathbf{r}_1\\ \mathbf{r}_1\\ \mathbf{r}_2\\ \mathbf{r}_3\\ \mathbf{r}_3\\ \mathbf{r}_3\end{pmatrix}} \mapsto \begin{pmatrix}\mathbf{r}_1\\ \mathbf{r}_2\\ \mathbf{r}_3\\ \hline \mathbf{r}_1\\ \mathbf{r}_3\\ \mathbf{r}_3\end{pmatrix} \equiv \overset{\mathbf{\Lambda}_r}{\left(\begin{array}{ccc} 1 & & \\ & 1 & \\ & & 1 \\ \hline 1 & & \\ & & 1 \\ & & 1 \end{array}\right)}$$

The subspace basis $\mathbf{\Lambda}_r$ may over-span $\mathbf{A}_F$ if $r > f$. In other words, when $r = f$ we get from Proposition 18 that the decomposition is minimal and exactly corresponds to a generic basis of IPS. When $r > f$, $\mathbf{\Lambda}_r$ may lead to degenerate pivots although hopefully less than with PS.

3.6.2 DCA on set partitioning models

DCA is devised solely for the set partitioning problem. It capitalizes on the compatibility interpretation and characterization of set partitioning optimal solutions. A binary solution to (3.61) is usually highly degenerate. Indeed, in typical vehicle routing and crew scheduling applications, a cluster covers several tasks, say on average $\bar{m}$, which implies that the number of variables assuming value one in the basis is of the order $m/\bar{m}$. The idea of row *aggregation* is born.

Assume for the moment that the linear relaxation of the set partitioning formulation (3.61) is written in standard form, that is,

$$z^\star := \min \quad \mathbf{c}^\intercal \mathbf{x} \quad \text{s.t.} \qquad \mathbf{A}\mathbf{x} = \mathbf{1}, \quad \mathbf{x} \geq \mathbf{0}. \tag{3.62}$$

We present three situations that can occur in DCA. The first assumes the current solution is binary while the second and third consider a fractional input for which the partition is the same as the IPS decomposition for the former and different for the latter.

If $\mathbf{x}_F$ is binary, the corresponding columns of $\mathbf{A}_F$ are disjoint and induce a partition of the row set into f clusters. From $\mathbf{A}_F$, it is easy to construct a reduced working basis: take a single row from each cluster and therefore, upon a permutation of the columns and rows of $\mathbf{A}$, matrix $\mathbf{A}_{PF}$ is $\mathbf{I}_f$. This is illustrated with the following integer solution $(x_1, x_2, x_3) = (1, 1, 1)$:

$$\overset{\mathbf{A}_F}{\begin{pmatrix} 1 & \\ 1 & \\ 1 & \\ & 1 \\ & 1 \\ & \\ & \end{pmatrix}} \begin{matrix} \\ \\ \\ \\ \\ 1 \\ 1 \end{matrix} \mapsto \left(\begin{array}{cc} 1 & \\ & 1 \\ & \\ \hline 1 & \\ 1 & \\ & 1 \end{array}\right) \begin{matrix} \\ \\ 1 \\ \\ \\ \\ \end{matrix} \equiv \overset{\mathbf{\Lambda}_f}{\left(\begin{array}{cc} 1 & \\ & 1 \\ & \\ \hline 1 & \\ 1 & \\ & 1 \end{array}\right)} \begin{matrix} \\ \\ 1 \\ \\ \\ \\ \end{matrix}.$$

If $\mathbf{x}_F$ is fractional, the row partition is again derived from the number of clusters $r \geq f$ of identical rows of $\mathbf{A}_F$. If $r = f$, we can again construct a working basis as in IPS. Take the first row from each cluster to form $\mathbf{A}_{PF}$ while the $m-f$ rows of $\mathbf{A}_{ZF}$ are copies of the f independent rows of $\mathbf{A}_{PF}$. Right multiplying $\mathbf{A}_F$ by $\mathbf{A}_{PF}^{-1}$ provides the subspace basis $\mathbf{\Lambda}_f = \begin{bmatrix} \mathbf{I}_f \\ \mathbf{A}_{ZF}\mathbf{A}_{PF}^{-1} \end{bmatrix}$. This alternative subspace basis is similar to the one obtained from a binary solution. This is illustrated with the following 3-variable fractional solution $(x_1, x_2, x_3) = (0.5, 0.5, 0.5)$:

$$\mathbf{A}_F \qquad\qquad \mathbf{\Lambda}_f$$
$$\begin{pmatrix} & 1 & 1 \\ & 1 & 1 \\ & 1 & 1 \\ 1 & & 1 \\ 1 & & 1 \\ 1 & 1 & \\ 1 & 1 & \\ 1 & 1 & \end{pmatrix} \mapsto \left(\begin{array}{ccc} & 1 & 1 \\ 1 & & 1 \\ 1 & 1 & \\ \hline & 1 & 1 \\ & 1 & 1 \\ 1 & & 1 \\ 1 & 1 & \\ 1 & 1 & \end{array}\right) \equiv \left(\begin{array}{ccc} 1 & & \\ & 1 & \\ & & 1 \\ \hline 1 & & \\ 1 & & \\ & 1 & \\ & & 1 \\ & & 1 \end{array}\right).$$

The third example shows a subspace basis $\mathbf{\Lambda}_r$ with $r > f$ induced by $\mathbf{x}_F = (x_1, x_2, x_3, x_4) = (0.5, 0.5, 0.5, 0.5)$. $\mathbf{A}_F$ comprises five clusters of identical rows, hence $\mathbf{\Lambda}_r$ has a dimension of $r = 5$. The row vectors satisfy $\mathbf{r}_1 + \mathbf{r}_3 = \mathbf{r}_2 + \mathbf{r}_5$ and IPS would have discarded one of these to construct $\mathbf{A}_{PF}$ of dimension $f = 4$.

$$\mathbf{A}_F \qquad\qquad\qquad \mathbf{\Lambda}_r$$
$$\begin{pmatrix} 1 & 1 & & \\ & 1 & 1 & \\ & & 1 & 1 \\ 1 & & 1 & \\ 1 & & & 1 \\ 1 & 1 & & \\ & 1 & 1 & \\ 1 & & & 1 \end{pmatrix} \mapsto \left(\begin{array}{ccc} 1 & 1 & \\ & 1 & 1 \\ & & 1 \\ 1 & & 1 \\ 1 & & \\ \hline 1 & 1 & \\ & 1 & 1 \\ 1 & & \end{array}\right) \equiv \left(\begin{array}{cccc} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ & & & \\ \hline 1 & & & \\ & 1 & & \\ & & & 1 \end{array}\right).$$

The idea of compatibility steered the research and the implementation of DCA and its variants. In the context of routing and scheduling, *compatibility* thus means generating itineraries or schedules which precisely respect the current clustering into multi-task activities. Of course, this is in perfect harmony with Definition 4. From the above discussion, we see that the subspace basis $\mathbf{\Lambda}_r$ is derived from the solution of the linear relaxation formulation (3.62). However, the process can be initialized from any partition $\mathbf{\Lambda}_r$ of the rows: this can be done in a heuristic manner, even infeasible. This simply results in a linear transformation $\mathbf{T}^{-1}$ applied on the system of equality constraints, updated when needed.

3.6.3 Resolution process

The resolution process of DCA uses several properties presented in Section 3.4. In fact, RMP_{FC} is implemented under the name of *aggregated restricted master problem* by Elhallaoui et al. (2005). The missing part of the puzzle, now provided by IPS, is the pricing problem (3.58). Since DCA did not have such a feature when it was originally designed, let us take a look how it manages to pursue the optimal solution.

Let us go back to basics and consider an optimal basic solution to RMP_{FC}, that is, the formulation (3.57) restricted to the column set $F \cup C$ (the compatible variables). From now on assume F represents the index set of the free variables in this so-called current solution, where $f \leq r$, and $\boldsymbol{\psi}_R$ is an optimal dual vector. If $f = r$, the matrix $\mathbf{A}_{RF}$ is the current working basis as in IPS; otherwise $f < r$ and the working basis is the final PS basis provided with the optimal solution, the one that serves to compute $\boldsymbol{\psi}_R$. In any case, we have $\bar{\mathbf{c}}_F = \mathbf{c}_F - \boldsymbol{\psi}_R^\intercal \mathbf{A}_{RF} = \mathbf{0}$ for basic variables $\mathbf{x}_F$ and $\bar{\mathbf{c}}_C = \mathbf{c}_C - \boldsymbol{\psi}_R^\intercal \mathbf{A}_{RC} \geq \mathbf{0}$ by optimality. The current solution is optimal if

$$\bar{\mathbf{c}}_I = \mathbf{c}_I - \boldsymbol{\psi}_R^\intercal \mathbf{A}_{RI} - \boldsymbol{\psi}_Z^\intercal \bar{\mathbf{A}}_{ZI} \geq \mathbf{0}.$$

While solving RMP_{FC}, the neglected constraints in the row set Z have no dual information on $\boldsymbol{\psi}_Z$. As in IPS, reduced costs $\bar{\mathbf{c}}_I$ are partial to the chosen partition. Yet, optimality of current solution is either true or false. The same reduced costs written with respect to the original dual vector, $\bar{\mathbf{c}}_I = \mathbf{c}_I - \boldsymbol{\pi}^\intercal \mathbf{A}_I$, highlight the possibility of *adapting* dual vector $\boldsymbol{\psi}_R$ to $\boldsymbol{\pi}$. The answer appears in (3.60), $\left[\boldsymbol{\pi}_R^\intercal, \boldsymbol{\pi}_Z^\intercal\right] = \left[\boldsymbol{\psi}_R^\intercal - \boldsymbol{\psi}_Z^\intercal \mathbf{M}, \boldsymbol{\psi}_Z^\intercal\right]$, which leads us to

$$\boldsymbol{\pi}_R^\intercal + \boldsymbol{\pi}_Z^\intercal \mathbf{M} = \boldsymbol{\psi}_R^\intercal. \tag{3.63}$$

As every column of the binary matrix $\mathbf{M}$ in a set partitioning problem identifies the remaining rows of a cluster, it means that $\psi_i,\ i \in R$, the dual variable of a cluster, must be distributed across the rows of its cluster, that is, (3.63) reads as $\sum_{\ell \in R_i} \pi_\ell = \psi_i,\ \forall i \in R$. Note that $\forall i \in R$, no matter how the $\boldsymbol{\psi}_i$ are distributed over their respective clusters, $\bar{\mathbf{c}}_F = \mathbf{0}$ and $\bar{\mathbf{c}}_C \geq \mathbf{0}$ remain satisfied. Therefore, consider the following set of constraints:

$$\boldsymbol{\pi}^\intercal \mathbf{a}_j \leq c_j, \quad \forall j \in I, \tag{3.64}$$

$$\sum_{\ell \in R_i} \pi_\ell = \psi_i, \quad \forall i \in R. \tag{3.65}$$

Notice that the system (3.64)–(3.65) is about feasibility. On the one hand, it is indeed feasible which means that the existence of acceptable dual variables certifies the optimal status of the current solution. On the other hand, it is infeasible: some constraints from (3.64) are removed until one retrieves a vector $\boldsymbol{\pi}$. Given those $\boldsymbol{\pi}$ values, DCA next prices out variables as in PS. A small selection of negative reduced costs incompatible variables is presumptuously added to current $\mathbf{A}_F$, say columns $\mathbf{A}_{I'}$, $I' \subset I$, such that a new partition is induced by the identical rows in $[\mathbf{A}_F, \mathbf{A}_{I'}]$, where $f + |I'| \leq m$. This yields a new subspace basis $\boldsymbol{\Lambda}_r$, $r > f$, at which point the algorithm proceeds with a new iteration, solving RMP_{FC} over a new set of constraints and compatible variables.

In DCA, the exercise of distributing the dual multipliers is called *dual variable disaggregation*. Since the expectation of an optimal solution to (3.64)–(3.65) can be put on hold, the system can be preemptively constructed in such a way that the algorithm expects a new partition. In particular, Elhallaoui et al. (2010) use a low-rank incompatibility strategy. In this respect, the disaggregation is heuristic by design.

Column generation. The pricing step established in IPS is however now available for DCA. Furthermore, in the context of column generation, columns of $\mathbf{A}$ being unknown and thus provided by a column generation pricing problem, the transformed matrix $\bar{\mathbf{A}}$ needs to be fabricated just the same. Indeed, the reader can verify that the improved pricing step would optimally solve by column generation (see Desrosiers et al. 2014) the following problem, equivalent to (3.58) for a set partitioning problem written in standard form:

$$\begin{array}{lllll} \max & \mu & & & \\ \text{s.t.} & \mu & \leq c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j, & [y_j] & \forall j \in I, \\ & \displaystyle\sum_{\ell \in R_i} \pi_\ell & = \psi_i, & & \forall i \in R. \end{array} \tag{3.66}$$

In SPP formulations of vehicle routing problems, the column generation pricing problem is a constrained shortest path. Since $\bar{\mathbf{A}}$ carries the information about the chosen partition $\{R, S\}$, modifying the constrained shortest path generator to account for this assumed partition effectively lightens its computational burden. Coined *Bi-dynamic constraint aggregation* (BDCA), this strategy is accounted for in the second revision of DCA, see Elhallaoui et al. (2008). While the idea of transferring the partition information might seem intuitive, the content of the latter paper is hardly summarizable in a few lines. We therefore insist on the former idea rather than this one successful application.

3.6.4 DCA algorithm

Figure 3.17 presents the algorithm for DCA. Let us concentrate on the modifications brought to Steps 3–5. The dual variable disaggregation replaces the pricing problem in Step 3b. Since the latter no longer provides an improving direction, Step 4 is skipped altogether. In Step 5, we move on directly to the column partition update. The small selection of incompatible variables $I' \subseteq I$ accompanies the column set F in Step 1 to create a new row partition.

1	Generic basis $\mathbf{T}$, transformation $\mathbf{T}^{-1}$, row partition $\{R, \bar{R}\}$ of $\mathbf{\Lambda}_r$;
2	Compatibility with the row partition $\{R, \bar{R}\}$ of $\mathbf{\Lambda}_r$ <optional>;
3a	Restricted master problem: solve RMP_{FC} to optimality <optional goto Step 5>;
3b	Fetch a subset $I' \subseteq I$ from the dual variable disaggregation;
4	Skip the exchange mechanism;
5	Update the column partition $\{F, L, U\}$ and goto Step 1;

Fig. 3.17: DCA algorithmic steps

3.6.5 Maintaining integrality

In SPP, wishful thinking unites with practicality. Indeed, the binary structure of the technological constraints is highly prejudicial to identical appearances in $\mathbf{A}_F$ represented by positive variables. Recall the three proposed examples to support this claim. In fact, when the solution is binary, the master problem of DCA is the same as in IPS. Furthermore, the nature of the pricing problem is such that it identifies convex combination containing few number of variables. Researchers such as Zaghrouti et al. (2014) rapidly turned a keen eye on this feature in an effort to maintain the integrality throughout the resolution process. It is known as *Integral Simplex Using Decomposition* (ISUD).

It amounts to verifying that the solution of the pricing problem, finally recuperated from IPS, yields an improved binary solution, rejecting the associated direction otherwise. In this respect, the binary restriction is transferred to the pricing problem, yet it is only used on a needed basis. Of course, the additional work imposed *de facto* on the pricing problem makes it more difficult to solve but if an optimal binary solution is obtained, it means the elimination of the branch-and-bound requirement. That is to say that when we aim to maintain integrality in the resolution process, it makes it hard not to endorse DCA's strategy. The latter exploits a fast partition scheme which works out exactly when the solution is binary and compatibility is easy to verify without the need of an inverse.

3.6.6 Computational results for DCA

For the aforementioned VCS problems with some 2,000 constraints together with average degeneracy levels between 40 and 50%, the combination of these ideas within `GENCOL`, a column generation software system, allows a reduction of solution times by factors of 50 to 100 (4.86 DCA $\times$ 4.38 BDCA $\times$ 4.53 MPDCA). The improvement factors are compounded as the strategies mix well together. DCA is the original version which is improved upon when the partition information is exploited in the pricing problem for generating negative reduced cost columns. Firstly with the modified constrained shortest path problem (BDCA) and secondly with the use of a low-rank incompatibility strategy (MPDCA).

To overcome degeneracy encountered during the resolution of the restricted master problem, Benchimol et al. (2012) propose a *stabilized* DCA (SDCA) that incorporates the above mentioned DCA into the dual variable stabilization (DVS) method of Oukil et al. (2007). The rationale behind this combination is that DCA reduces the primal space whereas DVS acts in the dual space. Combining both thus allows to fight degeneracy from primal and dual perspectives simultaneously. This method is again designed for solving the linear relaxation of set partitioning type models only. The computational results obtained on randomly generated instances of the multi-depot vehicle scheduling problem show that the performance of SDCA is not affected by the level of degeneracy and that it can reduce the average computational time of the master problem by a factor of up to 7 with respect to DVS. While this is not a direct comparison with DCA, the reduction factor would be even greater. Indeed, many instances solved by DVS could not be solved by DCA alone.

While DCA is implemented in a column generation context, ISUD is still in the early phase and applies only to known columns. The latest work of Rosat et al. (2014) shows that the pricing problem can be modified with relatively simple cuts when directions are rejected. These cuts have a major impact on the mean optimality gap dropping to 0.21% on some aircrew scheduling problems from 33.92% in the first ISUD paper.

3.7 Positive Edge

The identification of variables compatible with the row set R requires the computation of the transformed matrix $\bar{\mathbf{A}}^0_{ZN} = \mathbf{T}_Z^{-1}\mathbf{A}^0_{ZN}$, where $\mathbf{T}_Z^{-1} := \begin{bmatrix} -\mathbf{M} & \mathbf{I}_{m-r} \end{bmatrix}$. For large-scale problems, this can be time consuming. To overcome this situation, Raymond et al. (2010a) propose the Positive Edge rule. The latter exploits a creative stochastic argument which in turn allows the use of matrix multiplication rules to reduce the computational penalty. PE allows to

determine whether a variable $y_j,\ j \in N$, is compatible or not without explicitly computing the vector $\bar{\mathbf{a}}_{Zj} = \mathbf{T}_Z^{-1}\mathbf{a}_j$. It is based on the observations put forth in Section 3.7.1. Its statement is unveiled in Section 3.7.2 which also discusses its scope. Computational results are made available in Section 3.7.3.

3.7.1 Observations

Recall that if $\mathbf{a}_j$ is compatible, then $\bar{\mathbf{a}}_{Zj} = \mathbf{0}$. Hence, for any vector $\mathbf{v} \in \mathbb{R}^{m-r}$, we must have $\mathbf{v}^\intercal \bar{\mathbf{a}}_{Zj} = 0$. Otherwise, $\bar{\mathbf{a}}_{Zj} \neq \mathbf{0}$ and

$$\mathbf{v}^\intercal \bar{\mathbf{a}}_{Zj} = 0 \quad \text{if and only if} \quad \mathbf{v} \perp \bar{\mathbf{a}}_{Zj}, \tag{3.67}$$

that is, if and only if $\mathbf{v}$ and $\bar{\mathbf{a}}_{Zj}$ are orthogonal. Intuitively, this has a probability of zero for a continuous random vector $\mathbf{v}$. Define $\mathbf{w}^\intercal := \mathbf{v}^\intercal \mathbf{T}_Z^{-1}$. Then, for any variable $y_j,\ j \in N$,

$$\mathbf{v}^\intercal \bar{\mathbf{a}}_{Zj} = \mathbf{v}^\intercal \mathbf{T}_Z^{-1}\mathbf{a}_j = \mathbf{w}^\intercal \mathbf{a}_j. \tag{3.68}$$

The expression (3.68) is similar to $\mathbf{c}_B^\top \bar{\mathbf{a}}_j = \boldsymbol{\pi}^\top \mathbf{a}_j$ in the computation of the reduced cost of the variable x_j, where $\boldsymbol{\pi}$ is the vector of dual variables associated with the constraints $\mathbf{Ax} = \mathbf{b}$.

Computer architecture obliges, the continuous support is traded for a discrete one thus rendering the orthogonal probability to a nonzero value, although shown to be very small by Raymond et al. (2010a). We skip the proof and present only the random vector construction whose elements answer to the following definition.

Definition 5. *Within the specifications of a computer with 32-bit words, a nonzero rational number $\mathcal{F} \in \mathbb{Q}_0$ with a discrete random distribution SEM_{32} is a single precision floating-point number whose sign bit S, exponent E, and mantissa M are independent and follow the discrete uniform distributions $S \sim U[0,1]$, $E \sim U[64,191]$, and $M \sim U[0, 2^{23}-1]$.*

The random distribution SEM_{32} is symmetric around a zero mean ($\mu_{\mathcal{F}} = 0$) with a huge dispersion, its standard deviation being $\sigma_{\mathcal{F}} > 2^{60}$ (Towhidi et al. 2014). The random vector $\mathbf{v} \in \mathbb{Q}_0^{m-r}$ is such that all $m-r$ components are independent and identically distributed SEM_{32}.

3.7.2 PE rule

Within the scope of IPS, PE is a compatibility test which identifies nondegenerate improving pivots. It is a statistical test whereby the null hypothesis assumes the vector $\mathbf{a}$ is incompatible

until sufficient evidence is provided to conclude otherwise. On a more abstract level, PE is a membership test. In a linear algebra framework, PE indeed amounts to stochastically testing whether a given vector belongs to a subspace. Two types of error may surface, the vector $\mathbf{a}$ is assumed compatible but it is not or the vector is assumed incompatible when it is.

Positive Edge rule. Let $\mathbf{v} \in \mathbb{Q}_0^{m-r}$ be a random SEM_{32} vector and premultiply the vector $\mathbf{w}^\intercal := \mathbf{v}^\intercal \mathbf{T}_Z^{-1}$ as in (3.68). A vector $\mathbf{a} \in \mathbb{R}^m$ is *considered compatible with the vector subspace* $\mathbf{V}(\mathbf{\Lambda}_r)$ if $\mathbf{w}^\intercal \mathbf{a}_j = \mathbf{0}$.

Since the operation amounts to a dot product, this means that a compatible variable is recognized in $O(m)$ time using the original vector $\mathbf{a}_j$. Researchers suggested to first consider compatible variables. Indeed, since these variables exist in the original formulation, their identification can benefit both the pricing step in IPS as well as provide additional information for pivot-selection rules in PS. As such, we believe the foremost purpose of PE is the *identification* of compatible variables.

With respect to IPS, the pricing problem over compatible variables reduces to $\min_{j \in C} \tilde{d}_j$ while that over the incompatible ones is identical to (3.33) [or (3.45) for linear programs in standard form] but with $\mathbf{y}_N$ replaced by $\mathbf{y}_I$. Notice that considering the set C is solely from a theoretical point of view, while in practice the compatibility test is only performed for variables with negative reduced costs or a subset of them.

PE has also been tested independently of IPS, that is, by selecting entering variables in PS among the compatible set in priority. We provide details in the computational results below.

3.7.3 Computational results for PE

The proof of concept is provided in Raymond et al. (2010a) by using two external procedures within `CPLEX`'s black box environment. A direct implementation in `COIN-OR`'s CLP, where it has been combined with the Devex pricing criterion, is presented in Towhidi et al. (2014). The proposed implementation uses a two-dimensional rule: for a variable $x_j,\ j \in N$, the first dimension computes the reduced cost $\bar{c}_j = c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j$, whereas the second evaluates $\mathbf{w}^\intercal \mathbf{a}_j$. PE identifies $C_\mathbf{w} = \{j \in N | \mathbf{w}^\intercal \mathbf{a}_j = 0\}$ and $I_\mathbf{w} = \{j \in N | \mathbf{w}^\intercal \mathbf{a}_j \neq 0\}$. Let $\bar{c}_{j_\star},\ j_\star \in C_\mathbf{w} \cup I_\mathbf{w}$, be the smallest reduced cost and $\bar{c}_{j_\mathbf{w}},\ j_\mathbf{w} \in C_\mathbf{w}$, be the smallest one for a compatible variable. The current solution is optimal if $\bar{c}_{j_\star} \geq 0$. Compatible variables are preferred to enter the basis except if $\bar{c}_{j_\star}$ is much smaller than $\bar{c}_{j_\mathbf{w}}$. Given a parameter $0 \leq \alpha < 1$, the selection rule is:

$$\text{if } \bar{c}_{j_\star} < 0 \text{ and } \bar{c}_{j_\mathbf{w}} < \alpha \bar{c}_{j_\star}, \text{ then select } x_{j_\mathbf{w}} \text{ else } x_{j_\star}. \tag{3.69}$$

Tested with $\alpha = 0.5$ on 32 problems from Mittelmann's library (Koch et al. 2011) which contains instances with a wide range of degeneracy levels, computational results show that below a degeneracy level of 25%, PE is on average neutral while above this threshold, it reaches an average runtime speedup of 2.72, with a maximum of 4.23 on an instance with a 75% degeneracy level.

3.8 Conclusions

This paper presents a survey of three recent tools for dealing with primal degeneracy in linear programming. While DCA appears first in the context of set partitioning models encountered in many routing and scheduling formulations solved by branch-and-price, IPS extends the concept to linear programs in general. Both methods partition the set of constraints in two parts, at every iteration, based on the values taken by the basic variables. This can be seen as a dynamic application of the Dantzig-Wolfe decomposition principle. More specifically in IPS, one part of the constraints appears in the pricing problem as a homogeneous linear system (together with nonnegative variables) while the other part (together with the bound intervals on the variables) is used in the master problem to complete the exchange mechanism from one feasible solution to the next.

This paper also unifies IPS and DCA through a new interpretation in terms of the usage of two different subspace bases spanning the columns of the master problem. On the one hand, the subspace basis of IPS is made of the column vectors associated with the nondegenerate (or free) basic variables. As such every iteration of IPS is nondegenerate. On the other hand, that in DCA is derived from a partition of the rows into clusters, such as the one observed in any integer solution. This subspace basis has the fundamental property that it at least spans the free variable vectors. Therefore, the dimension of the subspace basis in DCA may be sometimes larger rather than equal to the number of free variables and this is the reason why some degenerate pivots may occur.

PE adds a compatibility test layer, done in polynomial time, to the traditional reduced cost pricing of nonbasic variables. That is, it identifies those entering variables that belong to the current vector subspace and are likely to lead to nondegenerate pivots, if any. Otherwise, the IPS pricing step identifies a convex combination of incompatible ones which also ultimately leads to a nondegenerate pivot until optimality is reached in a finite number of iterations.

Computational results reported from the literature show a large reduction on CPU times attributed to the diminution of degenerate pivots. What does the future look like? While

the theory behind IPS is sound and relatively straightforward, a general implementation is certainly a major concern. It is however hard to discard the specific structures of different families of LP problems. In this respect, the reader can think of several specializations of IPS to well structured problems such as network flows, multi-commodity flows, and linear relaxations of set partitioning and set covering problems.

The improved pricing step is the bottleneck of the method and needs to be handled with a great deal of insight. An efficient all-purpose implementation requires a significant amount of work and forces us to think about the different acceleration strategies that were presented herein. To name but a few, we have the partial pricing, the flexible subspace basis, the Dantzig-Wolfe surrogate variable environment and of course the infamous compatibility concept. The question that remains to be answered is whether trigger points for the usage of these ideas can be automated.

We are also looking at an implementation of these ideas within column generation, its adaptation to the dual simplex algorithm and to convex optimization, and its impact on the right-hand side sensitivity analysis, indeed the interpretation of the dual variables in the context of optimal degenerate solutions. Finally, the design of a completely efficient Integral Simplex algorithm for the set partitioning problem is a major goal.

Acknowledgements

Jacques Desrosiers acknowledges the Natural Sciences and Engineering Research Council of Canada for its financial support.

Decomposition theorems for linear programs

Jean Bertrand Gauthier and Jacques Desrosiers

GERAD & HEC Montréal, Canada

3000, chemin de la Côte-Sainte-Catherine

Montréal (Québec) Canada, H3T 2A7

<jean-bertrand.gauthier, jacques.desrosiers>@hec.ca

Marco E. Lübbecke

RWTH Aachen University, Germany

Kackertstraße 7

D-52072 Aachen, Germany

marco.luebbecke@rwth-aachen.de

ABSTRACT

It is well known that any feasible arc-flow solution to a network problem defined on a graph $G = (N, A)$, where N is the set of nodes whereas A is the set of arcs, can be expressed using at most $|A| + |N|$ paths and cycles having nonzero flow, out of these, at most $|A|$ cycles. This *existence theorem* is used in a number of proofs establishing the complexity of strongly polynomial algorithms for network flow problems such as the minimum mean cycle-canceling algorithm. While it is the crucial component of the analysis, the heart of the algorithm is the residual network upon which are derived two theorems that demonstrate its validity, the Augmenting Cycle Theorem and the Negative Cycle Optimality Theorem. This paper generalizes these theorems to linear programming.

Given a linear program (LP) with m constraints and n lower and upper bounded variables, any solution $\mathbf{x}^0$ to LP can be represented as a nonnegative combination of at most $m + n$ so-called weighted paths and weighted cycles, among which at most n weighted cycles. This fundamental decomposition theorem leads us to derive, on the residual problem $LP(\mathbf{x}^0)$, two alternative optimality conditions for linear programming, and eventually, a class of primal algorithms that rely on an Augmenting Weighted Cycle Theorem.

Keywords: network problems, flow decomposition, linear programming, residual problem, optimality conditions.

4.1 Introduction

Network flow problems can be formulated either by defining flows on arcs or, equivalently, flows on paths and cycles, see Ahuja et al. (1993). A feasible solution established in terms of path and cycle flow determines arc flows uniquely. The converse result, that is the *existence* of a decomposition as a path and cycle flow equivalent to a feasible arc-flow solution $\mathbf{x}^0$, is also shown to be true by the *Flow Decomposition Theorem*, although the decomposition might not be unique. This result can be refined for circulation problems, establishing that a feasible circulation can be represented along cycles only. Originally developed by Ford and Fulkerson (1962) for the maximum flow problem, the flow decomposition theory intervenes in various situations, notably on the residual network. It is used to prove, among many other results, the *Augmenting Cycle Theorem* and the *Negative Cycle Optimality Theorem*. The first allows to build one solution from another by a sequence of cycles. The second states that arc-flow solution $\mathbf{x}^0$ is optimal if and only if the residual network contains no negative cost cycle therefore providing optimality characterization for network flow problems. The *Flow Decomposition Theorem* is a fundamental theorem as it is an essential tool in the complexity analysis of several strongly polynomial algorithms such as the *minimum mean cycle-canceling* algorithm, see Goldberg and Tarjan (1989), Radzik and Goldberg (1994), and Gauthier et al. (2015b) for an improved complexity result. This paper generalizes these network flow theorems to linear programming.

The presentation adopts the organization of the introduction as follows. In Section 4.2, we first present a proof of the *Flow Decomposition Theorem* on networks based on linear programming arguments rather than the classical constructive ones. This provides an inspiration for the general case of linear programming. Section 4.3 establishes our main result based on a specific application of the Dantzig-Wolfe decomposition principle. This is followed in Section 4.4 by the proof of an *Augmenting Weighted Cycle Theorem* used to derive in Section 4.5 two alternative optimality conditions for linear programs that are based on the properties of a residual linear problem. We open a discussion in Section 4.6 which addresses the adaptation to linear programs of the *minimum mean cycle-canceling* algorithm and the design of a column generation based algorithm.

Notation. Vectors and matrices are written in bold face characters. We denote by $\mathbf{0}$ or $\mathbf{1}$ a vector with all zero or one entries of appropriate contextual dimensions.

4.2 A decomposition theorem for network flow problems

Consider the capacitated minimum cost flow problem (CMCF) on a directed graph $G = (N, A)$, where N is the set of nodes associated with an assumed balanced set $b_i, i \in N$, of supply or demand defined respectively by a positive or negative value such that $\sum_{i \in N} b_i = 0$, A is the set of arcs of cost $\mathbf{c} := [c_{ij}]_{(i,j)\in A}$, and $\mathbf{x} := [x_{ij}]_{(i,j)\in A}$ is the vector of lower and upper bounded flow variables. An arc-flow formulation of CMCF, where dual variables $\pi_i,\ i \in N$, appear in brackets, is given by

$$\begin{aligned} z^\star := \quad & \min \quad \sum_{(i,j)\in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i, \quad [\pi_i] \quad \forall i \in N \\ & 0 \le \ell_{ij} \le x_{ij} \le u_{ij}, \qquad \forall (i,j) \in A. \end{aligned} \tag{4.70}$$

When right-hand side $\mathbf{b} := [b_i]_{i\in N}$ is the null vector, formulation (4.70) is called a *circulation problem*. The *Flow Decomposition Theorem* for network solutions is as follows.

Theorem 7. (Ahuja et al. 1993, Theorem 3.5 and Property 3.6) *Any feasible solution* $\mathbf{x}^0$ *to CMCF* (4.70) *can be represented as a combination of directed path and cycle flows —though not necessarily uniquely— with the following properties:*

(a) *Every path with positive flow connects a supply node to a demand node.*

(b) *At most* $|A| + |N|$ *paths and cycles have positive flow among which at most* $|A|$ *cycles.*

(c) *A circulation* $\mathbf{x}^0$ *is restricted to at most* $|A|$ *cycles.*

Proof. The proof of the above theorem traditionally relies on a constructive argument. We propose an alternative one based on the application of the Dantzig-Wolfe decomposition principle (Dantzig and Wolfe 1960). The network problem is first converted into a circulation problem, partitioning the set of nodes N in three subsets: supply nodes in $S := \{i \in N \mid b_i > 0\}$, demand nodes in $D := \{i \in N \mid b_i < 0\}$, and transshipment nodes in $N \setminus \{S \cup D\}$ for which $b_i = 0,\ i \in N$. Supplementary nodes s and t are added to N for a convenient representation of the circulation problem together with zero-cost arc sets $\{(s,i) \mid i \in S\}$, $\{(i,t) \mid i \in D\}$, and arc (t,s). Supply and demand requirements are transferred on the corresponding arcs, that is, $\ell_{si} = u_{si} = b_i,\ i \in S$, and $\ell_{it} = u_{it} = -b_i,\ i \in D$. Let $G^+ = (N^+, A^+)$ be the new network on which is defined the circulation problem.

Flow conservation equations for nodes in N^+ together with the nonnegativity requirements on arcs in A^+ portray a circulation problem with no upper bounds. These define the domain $\mathcal{SP}$ of the Dantzig-Wolfe subproblem whereas lower and upper bound constraints remain in the master problem. By the Minkowski-Weil's theorem (see Schrijver 1986, Desrosiers and Lübbecke 2011), there is a vertex-representation for the domain $\mathcal{SP}$. The latter actually forms a cone that can be described in terms of a single extreme point (the null flow vector) and a finite number of extreme rays, see Lübbecke and Desrosiers (2005) for additional representation applications.

These extreme rays are translated to the original network upon which is done the unit flow interpretation in terms of paths and cycles. For an extreme ray with $x_{ts} = 1$, we face an external cycle in G^+, that is, a *path* within G from a supply node to a demand node, while an extreme ray with $x_{ts} = 0$ implies an internal cycle in G^+, that is, a *cycle* within G. Furthermore, the extreme ray solutions to $\mathcal{SP}$ naturally satisfy the flow conservation constraints and therefore respect the directed nature of G. Paths and cycles are therefore understood to be directed even though we omit the precision in the spirit of concision.

Let $\mathcal{P}$ and $\mathcal{C}$ be respectively the sets of paths and cycles in G. The null extreme point at no cost can be removed from the Dantzig-Wolfe reformulation as it has no contribution in the constraint set of the master problem. Any nonnull solution $\left[\mathbf{x}, \mathbf{x}_S, \mathbf{x}_D, x_{ts}\right]^\intercal$ to $\mathcal{SP}$ can therefore be written as a nonnegative combination of the extreme rays only, that is, in terms of the supply-demand paths $\left[\mathbf{x}_p, \mathbf{x}_{Sp}, \mathbf{x}_{Dp}, 1\right]^\intercal$, $p \in \mathcal{P}$, and internal cycles $\left[\mathbf{x}_c, \mathbf{0}, \mathbf{0}, 0\right]^\intercal$, $c \in \mathcal{C}$:

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{x}_S \\ \mathbf{x}_D \\ x_{ts} \end{bmatrix} = \sum_{p \in \mathcal{P}} \begin{bmatrix} \mathbf{x}_p \\ \mathbf{x}_{Sp} \\ \mathbf{x}_{Dp} \\ 1 \end{bmatrix} \theta_p + \sum_{c \in \mathcal{C}} \begin{bmatrix} \mathbf{x}_c \\ \mathbf{0} \\ \mathbf{0} \\ 0 \end{bmatrix} \phi_c, \quad \theta_p \geq 0, \ \forall p \in \mathcal{P}, \phi_c \geq 0, \ \forall c \in \mathcal{C}. \tag{4.71}$$

Define $c_p = \mathbf{c}^\intercal \mathbf{x}_p, \ p \in \mathcal{P}$, as the cost of a path and $c_c = \mathbf{c}^\intercal \mathbf{x}_c, \ c \in \mathcal{C}$, as the cost of a cycle. The Dantzig-Wolfe master problem, an alternative formulation of CMCF (4.70) written in

terms of nonnegative paths and cycles, respectively θ and ϕ-variables, is given as

$$\begin{array}{lrcl} z^\star := & \min & \displaystyle\sum_{p\in\mathcal{P}} c_p\theta_p + \sum_{c\in\mathcal{C}} c_c\phi_c & \\ \text{s.t.} & \mathbf{l} \leq \displaystyle\sum_{p\in\mathcal{P}} \mathbf{x}_p\theta_p + \sum_{c\in\mathcal{C}} \mathbf{x}_c\phi_c & \leq & \mathbf{u} \\ & \displaystyle\sum_{p\in\mathcal{P}} \mathbf{x}_{Sp}\theta_p & = & \mathbf{b}_S \\ & \displaystyle\sum_{p\in\mathcal{P}} \mathbf{x}_{Dp}\theta_p & = & -\mathbf{b}_D \\ & \theta_p \geq 0,\ \forall p\in\mathcal{P}, \quad \phi_c \geq 0,\ \forall c\in\mathcal{C}. & & \end{array} \tag{4.72}$$

The rest of the proof relies on the dimension of any basis representing a feasible solution $\mathbf{x}^0$ to (4.70). The latter can be expressed in terms of the change of variables in (4.71) and satisfies the system of equality constraints in (4.72):

$$\begin{array}{lcl} \displaystyle\sum_{p\in\mathcal{P}} \mathbf{x}_p\theta_p + \sum_{c\in\mathcal{C}} \mathbf{x}_c\phi_c & = & \mathbf{x}^0 \\ \displaystyle\sum_{p\in\mathcal{P}} \mathbf{x}_{Sp}\theta_p & = & \mathbf{b}_S \\ \displaystyle\sum_{p\in\mathcal{P}} \mathbf{x}_{Dp}\theta_p & = & -\mathbf{b}_D \\ \theta_p \geq 0,\ \forall p\in\mathcal{P}, \quad \phi_c \geq 0,\ \forall c\in\mathcal{C}. & & \end{array} \tag{4.73}$$

Since any basic solution to (4.73) involves at most $|A|+|S|+|D|$ nonnegative θ,ϕ-variables, there exists a representation for $\mathbf{x}^0$ that uses at most $|A|+|N|$ path and cycle variables, among which at most $|A|$ cycles (ϕ-variables). In the case of a circulation problem for which $\mathbf{b}=\mathbf{0}$, there are no paths involved (no θ-variables) and $\mathbf{x}^0$ can be written as a combination of at most $|A|$ cycles. □

4.3 A decomposition theorem for linear programs

In this section, we generalize Theorem 7 to the feasible solutions of a linear program (LP). Although it is usually frowned upon, we warn the reader that we reuse some of the same notations previously seen in networks. While the semantics are a little bit distorted, we wish to retain the ideas attached to them. The proof again relies on a specific Dantzig-Wolfe decomposition. Consider the following LP formulation with lower and upper bounded

variables:

$$\begin{aligned} z^\star := \quad \min \quad & \mathbf{c}^\intercal \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \qquad [\boldsymbol{\pi}] \\ & \mathbf{0} \leq \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned} \tag{4.74}$$

where $\mathbf{x}, \mathbf{c}, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $m \leq n$. Without loss of generality, we also assume that right-hand side vector $\mathbf{b} \geq \mathbf{0}$. If $\mathbf{b} = \mathbf{0}$, we face a homogeneous system of constraints. The vector of dual variables $\boldsymbol{\pi} \in \mathbb{R}^m$ associated with the equality constraints appears within brackets. In order to perform our specific decomposition, we introduce a vector of nonnegative variables $\mathbf{v} \in \mathbb{R}^m$ and rewrite LP (4.74), splitting the constraints in two subsets:

$$\begin{aligned} z^\star := \quad \min \quad & \mathbf{c}^\intercal \mathbf{x} \\ \text{s.t.} \quad & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & \mathbf{v} = \mathbf{b} \\ & \mathbf{A}\mathbf{x} - \mathbf{v} = \mathbf{0} \\ & \mathbf{x} \geq \mathbf{0}, \ \mathbf{v} \geq \mathbf{0}. \end{aligned} \tag{4.75}$$

Let the subproblem domain be the cone defined by $\mathcal{SP} := \{\mathbf{x} \geq \mathbf{0}, \ \mathbf{v} \geq \mathbf{0} \mid \mathbf{A}\mathbf{x} - \mathbf{v} = \mathbf{0}\}$ whereas objective function as well as constraint sets $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$ and $\mathbf{v} = \mathbf{b}$ remain in the master problem. With Minkowski-Weil's theorem in mind, take a look at the possible solution types of $\mathcal{SP}$. On the one hand, it comprises a single extreme point, the null solution at zero cost. On the other hand, the extreme rays, indexed by $r \in R$, are of two types: $\begin{pmatrix} \mathbf{x}_r \\ \mathbf{v}_r \end{pmatrix}$, $r \in R$, with either $\mathbf{v}_r \neq \mathbf{0}$ or $\mathbf{v}_r = \mathbf{0}$. Discarding the null extreme point from the Dantzig-Wolfe reformulation as it does not contribute to any constraints of the master problem nor to its objective function, index set R is exhaustively partitioned in two mutually exclusive subsets according to the value of $\mathbf{v}_r$: $\mathcal{P} := \{r \in R \mid \mathbf{v}_r \neq \mathbf{0}\}$ and $\mathcal{C} := \{r \in R \mid \mathbf{v}_r = \mathbf{0}\}$. Any nonnull solution $\begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix} \in \mathcal{SP}$ can therefore be solely expressed as a nonnegative combination of the extreme rays of $\mathcal{SP}$:

$$\begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix} = \sum_{p \in \mathcal{P}} \begin{pmatrix} \mathbf{x}_p \\ \mathbf{v}_p \end{pmatrix} \theta_p + \sum_{c \in \mathcal{C}} \begin{pmatrix} \mathbf{x}_c \\ \mathbf{0} \end{pmatrix} \phi_c, \quad \theta_p \geq 0, \ \forall p \in \mathcal{P}, \quad \phi_c \geq 0, \ \forall c \in \mathcal{C}. \tag{4.76}$$

Recall that the interpretation of network paths and cycles is done with respect to the original network. In the LP case, we chose a less intrusive transformation for which the content of $\mathbf{v}$ in the solutions of $\mathcal{SP}$ still grants meaning to the concept of paths and cycles.

Indeed, when $\mathbf{v} = \mathbf{0}$, the homogeneous system $\mathbf{Ax} = \mathbf{0}$ is verified whilst $\mathbf{v} \neq \mathbf{0}$ implies an impact on the right-hand side $\mathbf{b}$. The directed notion is of course once again implicit to the nature of the solutions found in $\mathcal{SP}$.

A more subtle concept to pass is the unit in which is measured these paths and cycles. The essence of an extreme ray is to be malleable by its multiplier. Unfortunately, this does not translate as well in LP as it does in networks. Indeed, a solution to $\mathcal{SP}$ found for network problems can always be scaled back to a unit flow cycle. In LP, the positive variables contained in an extreme ray can display different values. It is thus impossible to get rid of the scaling effect. As such, let an extreme ray $\binom{\mathbf{x}_p}{\mathbf{v}_p}$, $p \in \mathcal{P}$, be called a *weighted path* and an extreme ray $\binom{\mathbf{x}_c}{\mathbf{0}}$, $c \in \mathcal{C}$, be called a *weighted cycle*. The cost of these objects is thus measured in accordance with that scale and not against an arbitrary unit measure. Define $c_r = \mathbf{c}^\intercal \mathbf{x}_r$ as the cost of an extreme ray, $r \in R = \mathcal{P} \cup \mathcal{C}$. Substituting for $\mathbf{x}$ and $\mathbf{v}$ in (4.75), the Dantzig-Wolfe master problem (MP), a reformulation of the original LP (4.74), becomes

$$\begin{aligned} z^\star := \quad & \min \quad && \sum_{p\in\mathcal{P}} c_p\theta_p && + && \sum_{c\in\mathcal{C}} c_c\phi_c && \\ \text{s.t.} \quad & && \sum_{p\in\mathcal{P}} \mathbf{x}_p\theta_p && + && \sum_{c\in\mathcal{C}} \mathbf{x}_c\phi_c && \leq \mathbf{u} \\ & && \sum_{p\in\mathcal{P}} \mathbf{v}_p\theta_p && && && = \mathbf{b} \\ & && \theta_p \geq 0,\ p \in \mathcal{P}, && && \phi_c \geq 0,\ c \in \mathcal{C}. && \end{aligned} \tag{4.77}$$

Let $\mathbf{x}^0$ be a feasible solution to LP (4.74), that is, $\mathbf{Ax}^0 = \mathbf{b}$, $\mathbf{l} \leq \mathbf{x}^0 \leq \mathbf{u}$. Therefore, $\mathbf{x}^0$ must satisfy the following system derived from the change of variables (4.76) and the equality constraints in (4.77):

$$\begin{aligned} & \sum_{p\in\mathcal{P}} \mathbf{x}_p\theta_p && + && \sum_{c\in\mathcal{C}} \mathbf{x}_c\phi_c && = \mathbf{x}^0 \\ & \sum_{p\in\mathcal{P}} \mathbf{v}_p\theta_p && && && = \mathbf{b} \\ & \theta_p \geq 0,\ p \in \mathcal{P}, && && \phi_c \geq 0,\ c \in \mathcal{C}. && \end{aligned} \tag{4.78}$$

This linear system of equations comprises $m + n$ constraints for which any basic solution involves at most $m + n$ positive variables, among which at most n variables ϕ_c, $c \in \mathcal{C}$. The above discussion on the Dantzig-Wolfe reformulation of LP (4.74) constitutes the proof of our fundamental decomposition theorem for linear programming.

Theorem 8. *Any feasible solution* $\mathbf{x}^0$ *to LP* (4.74) *can be represented as a nonnegative combination of weighted paths and cycles —though not necessarily uniquely— with the following properties with respect to the Dantzig-Wolfe master problem reformulation* (4.77) *of LP:*

(a) *Every selected weighted path* $\binom{\mathbf{x}_p}{\mathbf{v}_p}, p \in \mathcal{P}$, *contributes to the right-hand side vector* $\mathbf{b}$.

(b) *At most* $m + n$ *weighted paths and cycles are selected among which at most* n *cycles* $\binom{\mathbf{x}_c}{\mathbf{0}}, c \in \mathcal{C}$.

(c) *For a homogeneous system* ($\mathbf{b} = \mathbf{0}$), *the representation requires at most* n *weighted cycles.*

4.4 An augmenting weighted cycle theorem

For network problems, Theorem 7 serves to prove the *Augmenting Cycle Theorem* (Ahuja et al. 1993, Theorem 3.7) formulated in terms of residual networks. In this section, we provide the counterpart for linear programs. Let us first start by the definition of the linear programming residual problem.

Let $\mathbf{x}^0$ be any feasible solution to (4.74), that is, a vector $\mathbf{x}^0 \in [\mathbf{l}, \mathbf{u}]$ satisfying $\mathbf{A}\mathbf{x}^0 = \mathbf{b}$. The cost of this solution is denoted $z^0 := \mathbf{c}^\intercal \mathbf{x}^0$. Perform the following change of variables (see Figure 4.18):

$$\mathbf{x} := \mathbf{x}^0 + (\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}}), \quad \vec{\mathbf{y}}^\intercal \overleftarrow{\mathbf{y}} = 0, \quad \vec{\mathbf{y}}, \overleftarrow{\mathbf{y}} \geq \mathbf{0}. \tag{4.79}$$

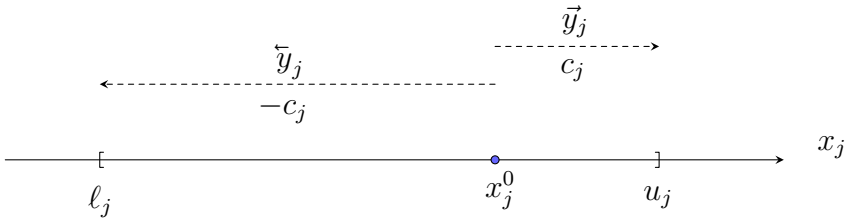


Fig. 4.18: A change of variables

We define the residual problem $LP(\mathbf{x}^0)$ with respect to a given solution $\mathbf{x}^0$ as follows. Each variable x_j, $j \in \{1, \ldots, n\}$, in the original LP is replaced by two variables: $\vec{y}_j \geq 0$ represents the possible increase of x_j relatively to x_j^0 while $\overleftarrow{y}_j \geq 0$ represents its possible decrease; moreover, only one can be used with a positive value ($\vec{y}_j \overleftarrow{y}_j = 0$). Variable $\vec{y}_j \leq \vec{r}_j^0 := u_j - x_j^0$

whereas $\overleftarrow{y}_j \leq \overleftarrow{r}^0_j := x^0_j - \ell_j$. Equivalent to LP (4.74), a formulation for $LP(\mathbf{x}^0)$ is as follows:

$$\begin{array}{rcccll} z^\star := & z^0 & + & \min & \mathbf{c}^\intercal(\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}}) & \\ & & & \text{s.t.} & \mathbf{A}(\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}}) = \mathbf{0}, & [\boldsymbol{\pi}] \\ & & & & \mathbf{0} \leq \vec{\mathbf{y}} \leq \vec{\mathbf{r}}^0 & \\ & & & & \mathbf{0} \leq \overleftarrow{\mathbf{y}} \leq \overleftarrow{\mathbf{r}}^0. & \end{array} \tag{4.80}$$

Consider now another feasible solution $\mathbf{x}$ to (4.74). Regardless of the number of iterations to reach it, Theorem 9 states that it is possible to move from $\mathbf{x}^0$ to the former in at most n weighted cycles.

Theorem 9. *Let $\mathbf{x}^0$ and $\mathbf{x}$ be two feasible solutions to LP (4.74). Then, $\mathbf{x}$ equals $\mathbf{x}^0$ plus the value on at most n weighted cycles in $LP(\mathbf{x}^0)$. Furthermore, the cost of $\mathbf{x}$ equals the cost of $\mathbf{x}^0$ plus the cost on these weighted cycles.*

Proof. There exists a correspondence between solution $\mathbf{x}$ on the original problem LP (4.74) and a solution $(\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}})$ on the residual problem $LP(\mathbf{x}^0)$. This is given by the change of variables in (4.79). If $x_j \geq x^0_j$, we set $\vec{y}_j = x_j - x^0_j$ and $\overleftarrow{y}_j = 0$; otherwise $\overleftarrow{y}_j = -(x_j - x^0_j)$ and $\vec{y}_j = 0$. As $\mathbf{x}^0$ and $\mathbf{x}$ are feasible, $\mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{x}^0 = \mathbf{b}$, therefore $(\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}})$ satisfies

$$\begin{aligned} (\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}}) &= \mathbf{x} - \mathbf{x}^0 \\ \mathbf{A}(\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}}) &= \mathbf{0} \\ \vec{\mathbf{y}},\ \overleftarrow{\mathbf{y}} &\geq \mathbf{0}. \end{aligned} \tag{4.81}$$

By the Decomposition Theorem 8, the homogeneous system $\{\vec{\mathbf{y}}, \overleftarrow{\mathbf{y}} \geq \mathbf{0} \mid \mathbf{A}(\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}}) = \mathbf{0}\}$ at $\mathbf{x}^0$ can be decomposed into weighted cycles only, indexed by $c \in \mathcal{C}(\mathbf{x}^0)$. Hence,

$$\sum_{c \in \mathcal{C}(\mathbf{x}^0)} (\vec{\mathbf{y}}_c - \overleftarrow{\mathbf{y}}_c)\phi_c = \mathbf{x} - \mathbf{x}^0, \qquad \phi_c \geq 0,\ \forall c \in \mathcal{C}(\mathbf{x}^0). \tag{4.82}$$

Therefore, any basic solution to (4.82) comprises at most n positive cycle-variables, proving the first part of the statement. Regarding the second part, we have $\mathbf{c}^\intercal\mathbf{x} = \mathbf{c}^\intercal\mathbf{x}^0 + \sum_{c \in \mathcal{C}(\mathbf{x}^0)} \mathbf{c}^\intercal(\vec{\mathbf{y}}_c - \overleftarrow{\mathbf{y}}_c)\phi_c$ from which we derive the requested result by restricting the cycle cost to the selected weighted cycles in a basic solution of (4.82). □

4.5 Primal and dual optimality conditions on $LP(\mathbf{x}^0)$

Let $\mathbf{A} = [\mathbf{a}_j]_{j\in\{1,\dots,n\}}$. The reduced cost of x_j in LP (4.74) is defined as $\bar{c}_j = c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j$. In addition to the complementary slackness optimality conditions on LP based on the reduced cost of the $\mathbf{x}$-variables (see Schrijver 1986), we provide two alternative conditions characterizing optimality for linear programs. These are based on the above *Augmenting Weighted Cycle Theorem* 9 which is itself derived from the *Decomposition Theorem* 8.

Theorem 10. *A feasible solution $\mathbf{x}^0$ to LP (4.74) is optimal if and only if the following equivalent conditions are satisfied:*

(a) *$LP(\mathbf{x}^0)$ contains no weighted cycle of negative cost.*

(b) *$\exists \boldsymbol{\pi}$ such that the reduced cost of every variable of $LP(\mathbf{x}^0)$ is nonnegative.*

(c) *$\exists \boldsymbol{\pi}$ such that $\forall j \in \{1,\dots,n\}$:*

$$x_j^0 = \ell_j \text{ if } \bar{c}_j > 0, \quad x_j^0 = u_j \text{ if } \bar{c}_j < 0, \quad \bar{c}_j = 0 \text{ if } \ell_j < x_j^0 < u_j.$$

Proof. Firstly, we prove that conditions (a) and (b) on the residual problem $LP(\mathbf{x}^0)$ are equivalent by providing linear programming models for these. Secondly, we show that the primal condition (a) characterizes linear programming optimality. Complementary slackness ones in (c) are only stated for the completeness of the presentation. It should however be clear that the necessary and sufficient quality of (a) and (b) make them equivalent to (c).

Assume a feasible solution $\mathbf{x}^0$ of cost z^0 from which are derived the residual upper bound vectors $\vec{\mathbf{r}}^0$ and $\overleftarrow{\mathbf{r}}^0$. Recall that $\boldsymbol{\pi} \in \mathbb{R}^m$ denotes the dual vector associated with the homogeneous linear system in $LP(\mathbf{x}^0)$ (4.80). Fixing to zero all $\mathbf{y}$-variables with null residual upper bounds, we formulate a problem for maximizing $\mu \leq 0$, the smallest reduced cost:

$$\begin{array}{lll} \max & \mu & \\ \text{s.t.} & \mu \leq \quad (c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j), & [\vec{y}_j] \quad \forall j \in \{1,\dots,n\} \mid \vec{r}_j^0 > 0 \\ & \mu \leq -(c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j), & [\overleftarrow{y}_j] \quad \forall j \in \{1,\dots,n\} \mid \overleftarrow{r}_j^0 > 0 \\ & \mu \leq 0. & \end{array} \tag{4.83}$$

We underscore that the vector $\boldsymbol{\pi}$ is also optimized in (4.83). When $\mu = 0$, every variable in $LP(\mathbf{x}^0)$ has a nonnegative reduced cost and condition (b) is satisfied. Let the $\mathbf{y}$-variables in brackets be the dual variables associated with the inequality constraints. The dual formulation

of (4.83) is

$$\begin{aligned} \min \quad & \mathbf{c}^\intercal(\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}}) \\ \text{s.t.} \quad & \mathbf{A}(\vec{\mathbf{y}} - \overleftarrow{\mathbf{y}}) = \mathbf{0}, \quad [\boldsymbol{\pi}] \\ & \mathbf{1}^\intercal\vec{\mathbf{y}} + \mathbf{1}^\intercal\overleftarrow{\mathbf{y}} \leq 1, \quad [\mu] \\ & \vec{\mathbf{y}},\ \overleftarrow{\mathbf{y}} \geq \mathbf{0}, \\ & \vec{y}_j = 0, \qquad \forall j \in \{1,\dots,n\} \mid \vec{r}_j^{\,0} = 0 \\ & \overleftarrow{y}_j = 0, \qquad \forall j \in \{1,\dots,n\} \mid \overleftarrow{r}_j^{\,0} = 0. \end{aligned} \tag{4.84}$$

Therefore, finding an improving direction $(\vec{\mathbf{y}}^{\,0} - \overleftarrow{\mathbf{y}}^{0}) \in \mathbb{R}^n$ from $\mathbf{x}^0$, a so-called *weighted cycle* of negative cost, consists in solving the above pricing problem. These two formulations form a primal-dual pair for the pricing step. The primal one (4.84) searches for an optimal weighted cycle $(\vec{\mathbf{y}}^{\,0} - \overleftarrow{\mathbf{y}}^{0})$ of negative cost, if any. Its dual version (4.83) maximizes the smallest reduced cost μ^0 on $LP(\mathbf{x}^0)$ by an optimization of the dual vector $\boldsymbol{\pi}$. By duality, $\mathbf{c}^\intercal(\vec{\mathbf{y}}^{\,0} - \overleftarrow{\mathbf{y}}^{0}) = \mu^0$, hence $LP(\mathbf{x}^0)$ contains no weighted cycle of negative cost if and only if $\exists\boldsymbol{\pi}^0$ such that $\mu^0 = 0$, that is, the reduced cost of every variable of $LP(\mathbf{x}^0)$ is nonnegative. This concludes the equivalence between (a) and (b).

In order to prove (a), suppose $\mathbf{x}^0$ is feasible and $LP(\mathbf{x}^0)$ contains a weighted cycle $(\vec{\mathbf{y}}^{\,0} - \overleftarrow{\mathbf{y}}^{0})$ of negative cost μ^0. In that case, $\mathbf{x}^0$ can be improved according to the change of variables in (4.79), that is, $\mathbf{x}^1 := \mathbf{x}^0 + \rho^0\,(\vec{\mathbf{y}}^{\,0} - \overleftarrow{\mathbf{y}}^{0})$ and $z^1 := z^0 + \rho^0\,\mu^0$, where the maximum step size $\rho^0 > 0$ is limited in the residual problem $LP(\mathbf{x}^0)$ (4.80) by $\rho\,\vec{\mathbf{y}}^{\,0} \leq \vec{\mathbf{r}}^{\,0}$ and $\rho\,\overleftarrow{\mathbf{y}}^{0} \leq \overleftarrow{\mathbf{r}}^{0}$. Hence, $\mathbf{x}^0$ cannot be optimal.

To show the converse, assume $\mathbf{x}^0$ is feasible and $LP(\mathbf{x}^0)$ contains no weighted cycle of negative cost. Let $\mathbf{x}^\star \neq \mathbf{x}^0$ be an optimal solution. Theorem 9 shows that the difference vector $\mathbf{x}^\star - \mathbf{x}^0$ can be decomposed into at most n augmenting weighted cycles in $LP(\mathbf{x}^0)$, where the total cost on these cycles equals $\mathbf{c}^\intercal\mathbf{x}^\star - \mathbf{c}^\intercal\mathbf{x}^0 = \sum_{c\in\mathcal{C}(\mathbf{x}^0)} \mathbf{c}^\intercal(\vec{\mathbf{y}}_c - \overleftarrow{\mathbf{y}}_c)\phi_c \geq 0$, the costs of all the cycles in $LP(\mathbf{x}^0)$ being nonnegative. Since the vector $\mathbf{x}^\star$ is optimal, we also have $\mathbf{c}^\intercal\mathbf{x}^\star - \mathbf{c}^\intercal\mathbf{x}^0 \leq 0$ meaning that $\mathbf{c}^\intercal\mathbf{x}^\star = \mathbf{c}^\intercal\mathbf{x}^0$. Ultimately, $\mathbf{x}^0$ must also be optimal thus completing the proof of the theorem. □

4.6 Discussion

In this section, we present two lines of research stemming from Theorems 8–10 and the residual problem $LP(\mathbf{x}^0)$. Keep in mind that these results are extensions of widely used pieces of theory all of which necessary to ascertain the validity of some network flow algorithms. The first

direction of research is reminiscent of the *minimum mean cycle-canceling* algorithm (MMCC) of Goldberg and Tarjan (1989) and aims to generalize the latter to linear programming. The goal is thus to determine analogous results using the linear programming counterparts defined herein. Some ideas and difficulties ahead are exposed in Section 4.6.1.

The second direction of research refers to Theorem 9 which shows that all weighted cycles required to reach an optimal solution $\mathbf{x}^\star$ exist on $LP(\mathbf{x}^0)$. We argue that an algorithmic process could be explicitly constructed around this observation. The idea presented in Section 4.6.2 is to insert the process in a column generation scheme and rely on the Dantzig-Wolfe decomposition provided in (4.76)-(4.77).

4.6.1 Adaptation of MMCC to linear programs

As simple as the statement may be, the Decomposition Theorem 7 acts as a cornerstone to numerous results in network flows. It is in particular the core idea behind the mechanics of the *minimum mean cycle-canceling* algorithm of Goldberg and Tarjan (1989). Given a feasible solution $\mathbf{x}^0$, the algorithm moves to solution $\mathbf{x}^1$ by using a cycle of minimum mean cost on the residual network $G(\mathbf{x}^0)$. The solution process therefore traverses a series of residual networks $G(\mathbf{x}^k),\ k \geq 0$, eventually reaching one that contains no negative cost cycle. The mechanics of the algorithm limits the search to cycles because of the Augmenting Cycle Theorem while the Negative Cycle Optimality Theorem further limits this search to negative cost cycles only.

In theory, it seems like the iterative process of MMCC can be adapted to linear programming in a relatively straightforward manner using the residual problems $LP(\mathbf{x}^k),\ k \geq 0$. At iteration k, one has on hand a feasible solution $\mathbf{x}^k$ from which are derived the residual upper bound vectors $\vec{\mathbf{r}}^k$ and $\overleftarrow{\mathbf{r}}^k$. Fixing to zero all $\mathbf{y}$-variables in $LP(\mathbf{x}^k)$ with null residual upper bounds, one searches in (4.84) for a weighted cycle $(\vec{\mathbf{y}}^k - \overleftarrow{\mathbf{y}}^k)$ of minimum negative cost according to Theorem 10 until optimality is reached. We have already mentioned the subtlety of the interpretation of the LP weighted cycle. Once again, notice the construction of the sentence opposed to that of MMCC: in both cases the cost is weighted and measured against the convexity constraint but the weights are lifted from the cycle definition in networks. If such a weighted cycle is found in (4.84), the current solution can be improved, that is, $\mathbf{x}^{k+1} := \mathbf{x}^k + \rho^k(\vec{\mathbf{y}}^k - \overleftarrow{\mathbf{y}}^k),\ z^{k+1} := z^k + \rho^k \mathbf{c}^\intercal(\vec{\mathbf{y}}^k - \overleftarrow{\mathbf{y}}^k)$, where the maximum step size $\rho^k > 0$ is limited by $\rho\,\vec{\mathbf{y}}^k \leq \vec{\mathbf{r}}^k$ and $\rho\,\overleftarrow{\mathbf{y}}^k \leq \overleftarrow{\mathbf{r}}^k$.

Although it remains to be proven, the fact that the objective function is modified at every iteration bodes well the convergence. Furthermore, whether the theoretical complexity of

such a process can be established using the same line of arguments as seen in MMCC is an interesting question which deserves an analysis. This analysis revolves around the behavior of the minimum reduced cost μ. The measure of the sporadic jumps that can be shown on the latter poses several challenges in the LP adaptation, most notably by the scaling impact of the convexity constraint. The quest for polynomial properties, even on part of the algorithm, is set forth.

4.6.2 On the solution of $MP(\mathbf{x}^0)$ by column generation

One of the most successful ideas of column generation is to harvest a lot of information from the subproblem even though some of it might turn out irrelevant (see Lübbecke and Desrosiers 2005). From this principle, we venture the idea that the Dantzig-Wolfe framework allows for another algorithmic process that might share several features of MMCC. The idea is to build the decomposition only once and refine its parameters within a master/subproblem paradigm. The idea actually comes forth quite naturally when one ponders at strategic ways to implement the previous algorithm.

Let $\mathbf{x}^0$ be a feasible solution to LP (4.74) and apply a Dantzig-Wolfe decomposition on the residual problem $LP(\mathbf{x}^0)$ (4.80) while keeping only the (positive) residual upper bound constraints in the master problem $MP(\mathbf{x}^0)$. Therefore, the domain of the pricing subproblem defined in (4.84) is the cone for which we added a convexity constraint. Let $\mathcal{C}(\mathbf{x}^0)$ denote the set of its weighted cycles. The formulation of $MP(\mathbf{x}^0)$, with the dual vectors $\vec{\boldsymbol{\omega}}$ and $\overleftarrow{\boldsymbol{\omega}}$ in brackets, is

$$\begin{aligned} z^\star := \quad z^0 \; + \; \min \quad & \sum_{c \in \mathcal{C}(\mathbf{x}^0)} \mathbf{c}^\intercal (\vec{\mathbf{y}}_c - \overleftarrow{\mathbf{y}}_c)\phi_c & & \\ \text{s.t.} \quad & \sum_{c \in \mathcal{C}(\mathbf{x}^0)} \vec{\mathbf{y}}_c \phi_c & \leq \vec{\mathbf{r}}^0, \quad & [\vec{\boldsymbol{\omega}}] \\ & \sum_{c \in \mathcal{C}(\mathbf{x}^0)} \overleftarrow{\mathbf{y}}_c \phi_c & \leq \overleftarrow{\mathbf{r}}^0, \quad & [\overleftarrow{\boldsymbol{\omega}}] \\ & \phi_c \geq 0, \; \forall c \in \mathcal{C}(\mathbf{x}^0). & & \end{aligned} \tag{4.85}$$

Given any feasible solution $\mathbf{x}^0$ to LP (4.74), we *see* the optimal solution $\mathbf{x}^\star$ using a combination of extreme rays derived on the residual problem $LP(\mathbf{x}^0)$ (the cone located at $\mathbf{x}^0$ is a convex set). This means that we can stay at that solution point $\mathbf{x}^0$ rather than move at every iteration, as in simplex type algorithms or even MMCC. In other words, extreme ray vectors are brought into $MP(\mathbf{x}^0)$ until optimality is reached.

Observe that the algorithmic adaptation of MMCC to LP in Section 4.6.1 does not solve $MP(\mathbf{x}^0)$ (4.85). It rather performs a single iteration of the Dantzig-Wolfe decomposition on $LP(\mathbf{x}^0)$. Indeed, at any iteration k, it brings into $MP(\mathbf{x}^k)$ a single weighted cycle $(\vec{\mathbf{y}}^k - \overleftarrow{\mathbf{y}}^k) = (\vec{\mathbf{y}}_c - \overleftarrow{\mathbf{y}}_c)$, $c \in \mathcal{C}(\mathbf{x}^k)$, of cost μ^k, then computes $\phi_c = \rho^k$ and $z^{k+1} = z^k + \rho^k \mu^k$. The solution moves to $\mathbf{x}^{k+1} = \mathbf{x}^k + \rho^k(\vec{\mathbf{y}}^k - \overleftarrow{\mathbf{y}}^k)$ and one reiterates by reapplying the decomposition procedure on $LP(\mathbf{x}^{k+1})$. Full decomposition at $\mathbf{x}^0$ would use dual vectors $\vec{\boldsymbol{\omega}}$ and $\overleftarrow{\boldsymbol{\omega}}$ in $MP(\mathbf{x}^0)$ to select by column generation the rays to fill in the master problem. The pricing problem (4.84) which finds weighted cycles of minimum reduced cost has its objective function min $(\mathbf{c} - \vec{\omega})^\intercal \vec{\mathbf{y}} - (\mathbf{c} - \overleftarrow{\omega})^\intercal \overleftarrow{\mathbf{y}}$ updated over the column generation iterations.

4.6.3 Final remarks

Although the minimum mean cycle-canceling algorithm can be stated within a paragraph, it is more difficult to capture the depth of its actual ramifications. Case in point, Radzik and Goldberg (1994) and Gauthier et al. (2015b) improve upon the original complexity analysis of this algorithm several years apart. These improvements stem on the one hand from mathematical arguments and on the other hand from the way operations are conducted within the resolution process. The room for improvement of the seminal work gives hope that alternative ways of thinking may result in more efficient implementations. Indeed, careful design choices can be made to improve a white paper algorithm thus showing that even tight complexities must be interpreted with care.

In this spirit, the study combination of the complexity analysis for linear programs along with the column generation dimension might instill an emulation environment for these two lines of research. The adaptation of the three network-based theorems provided herein are essential components of the analysis that lies ahead. As final remarks, we think the linear programming proof of the decomposition theorem for networks is an interesting result in itself. Also, the two alternative necessary and sufficient optimality conditions for LPs drive the interest of the two proposed algorithms. The first works in the original space of the linear program whereas the second works in the vertex space where the variables have a richer content. While the first guarantees degeneracy immunity, the second makes no such promises.

Acknowledgements

Jacques Desrosiers acknowledges the National Science and Engineering Research Council of Canada for its financial support.

A strongly polynomial Contraction-Expansion algorithm for network flow problems

Jean Bertrand Gauthier and Jacques Desrosiers

GERAD & HEC Montréal, Canada

3000, chemin de la Côte-Sainte-Catherine

Montréal (Québec) Canada, H3T 2A7

<jean-bertrand.gauthier, jacques.desrosiers>@hec.ca

Marco E. Lübbecke

RWTH Aachen University, Germany

Kackertstraße 7

D-52072 Aachen, Germany

marco.luebbecke@rwth-aachen.de

ABSTRACT

This paper addresses the resolution of the capacitated minimum cost flow problem on a network containing n nodes and m arcs. Verifying necessary and sufficient optimality conditions can be done on the residual network although it can be quite time consuming as testified by the minimum mean cycle-canceling algorithm. We introduce a contracted network which exploits these conditions on a much smaller network. Since the construction of this contracted network is very flexible, we study its properties depending on the construction choice. A generic contraction algorithm is then produced around the contracted network. Interestingly enough, it turns out it encapsulates both the minimum mean cycle-canceling and primal simplex algorithms. By guiding the resolution using a particular expansion scheme, we are able to recuperate theoretical results from the minimum mean cycle-canceling algorithm. As such, we obtain a strongly polynomial Contraction-Expansion algorithm which runs in $O(m^3n^2)$ time. The algorithmic choices stick to very practical observations of the minimum mean cycle-canceling algorithm's resolution behavior yet are not exploited by latter, namely that of phases and jumps on the optimality parameter. The resolution time is ultimately significantly reduced, even more so as the size of the instance increases.

Keywords: Network flow problem, residual network, contracted network, minimum mean cost cycle, complexity analysis, strongly polynomial algorithm.

5.1 Introduction

The primal network simplex algorithm performs surprisingly well considering that only a tiny fraction of iterations induce nondegenerate pivots (see Ahuja et al. 1993, Figure 18.7). In fact, the network simplex and the cost scaling methods are typically preferred over competing alternatives for the resolution of network flow problems also known as capacitated minimum cost flow problems (CMCF). Kovács (2015) suggests the use of the former for smaller networks and the latter for larger instances.

The minimum mean cycle-canceling algorithm (MMCC) is one such alternative. Introduced by Goldberg and Tarjan (1989), this algorithm copes with degeneracy at the expense of a more involved pricing problem able to only identify improving directions with positive step sizes. Despite its strongly polynomial time complexity, the theoretical behavior of MMCC is in practice no match for other methods. Radzik and Goldberg (1994) even improve the complexity some five years later and introduce the concept of phases. Two decades further down the roads bring another improvement due to Gauthier et al. (2015b) which combines phases with Cancel-and-Tighten (CT) presented along side MMCC in the initial paper as a self-standing algorithm.

This paper presents a *Contraction-Expansion* algorithm (CE) inspired by MMCC and its complexity proof. Since the visual aid granted by the network flow formulation gives a lot of perspective to the theoretical analysis, we opt to present algorithmic choices in a constructive fashion. Furthermore, we adopt the definitions and nomenclature of Ahuja et al. (1993) in which the reader may find several network flow properties.

The paper is organized as follows. Section 5.2 defines the network problem and exposes the building block of this paper, namely the so-called contracted network. The contracted network is a flexible construction which gives rise to a generic contraction algorithm whose properties are discussed in Section 5.3. A behavioral study of the so-called *optimality parameter* follows in Section 5.4. From these observations, an expansion scheme guiding the resolution process is incidentally drafted in Section 5.5, where the ensuing complexity analysis expands upon theoretical results from the minimum mean cycle-canceling algorithm. Our final thoughts can be found in Section 5.6.

5.2 Network problem

Assume a capacitated directed network $G := (N, A)$, where N is the set of n nodes and A is the set of m arcs. The arc parametrization is captured by the cost vector $\mathbf{c} := [c_{ij}]_{(i,j)\in A}$ and nonnegative bounds $[\ell_{ij}, u_{ij}], \forall (i,j) \in A$ whereas a balanced set $b_i, i \in N$ of supply or demand respectively defined by a positive or negative value is associated with the nodes such that $\sum_{i\in N} b_i = 0$. Supported by G and the vector of bounded flow variables $\mathbf{x} := [x_{ij}]_{(i,j)\in A}$, the formulation of CMCF is given by:

$$\begin{aligned} z^* := \quad & \min \sum_{(i,j)\in A} c_{ij} x_{ij} \\ \text{s.t.} \quad & \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = b_i, \quad [\pi_i], \quad \forall i \in N, \\ & \ell_{ij} \leq x_{ij} \leq u_{ij}, \qquad\qquad \forall (i,j) \in A, \end{aligned} \tag{5.86}$$

where $\boldsymbol{\pi} := [\pi_i]_{i\in N}$ is the vector of dual variables, also known as node potentials. We further assume that G does not contain any multiarc, i.e., arcs sharing the same head and tail.

When right-hand side $\mathbf{b} := [b_i]_{i\in N}$ is the null vector, the formulation (5.86) is called a *circulation* problem. The circulation problem is a cornerstone of network flow problems. Case in point, it is brought to light within the minimum mean cycle-canceling algorithm where it justifies the validity of the latter. Indeed, its core component, namely the so-called residual network is a circulation problem for which upper bounds are eventually neglected.

For the sake of formality, let us recall the path and cycle definitions used by Ahuja et al. (1993). A *path* in a directed graph $G = (N, A)$ is a sequence without repetition of nodes and arcs $i_1 - a_1 - i_2 - a_2 - \ldots - i_{r-1} - a_{r-1} - i_r$, satisfying the property that for all $i_k, 1 \leq k \leq r$, either $a_k = (i_k, i_{k+1}) \in A$ or $a_k = (i_{k+1}, i_k) \in A$. The sequence is typically given using nodes only. A *directed path* is an *oriented* version of a path in the sense that for any two consecutive nodes i_k and i_{k+1} on the path, the arc $(i_k, i_{k+1}) \in A$. A *cycle* is a path together with the arc (i_r, i_1) or (i_1, i_r) whereas a *directed cycle* is a directed path together with the arc (i_r, i_1). The *cost* of a path or a cycle is the total unit cost of traveling on the forward arcs minus that on the backward arcs. Note there is no backward arc in a directed path nor a directed cycle.

With respect to any dual variable vector $\boldsymbol{\pi} := (\pi_i)_{i\in N}$, the reduced cost of $x_{ij}, (i,j) \in A$ is defined as $\bar{c}_{ij} := c_{ij} - \pi_i + \pi_j$. Although Ahuja et al. (1993) limit the following fundamental property to directed cycles, the same telescopic summation argument proves to be enough regardless of the directed nature.

Proposition 19. *The cost and reduced cost of a cycle, directed or not, are equal.*

In the following subsections, we define the traditional residual network which exposes the improving disposition of the current solution $\mathbf{x}^0 := [x^0_{ij}]_{(i,j)\in A}$. An optimality certificate is then provided which indeed amounts to the statement of *cycle-canceling* algorithms. We finally move on to a *contraction* gymnastic which induces a so-called contracted network.

5.2.1 Residual network

The residual network is a marginal construction of the flow that may traverse the network aside the current flow $\mathbf{x}^0$. The principle is simple, the combination of $\mathbf{x}^0$ along with the optimal marginal flow computed on the residual network is optimal for the original formulation. As eloquently resumed in Figure 5.19, each arc $(i,j) \in A$ can be replaced by two arcs representing upwards and downwards possible flow variations:

- arc (i,j) with cost $d_{ij} := c_{ij}$ and residual flow $0 \leq y_{ij} \leq r^0_{ij} := u_{ij} - x^0_{ij}$;
- arc (j,i) with cost $d_{ji} := -c_{ij}$ and residual flow $0 \leq y_{ji} \leq r^0_{ji} := x^0_{ij} - \ell_{ij}$.

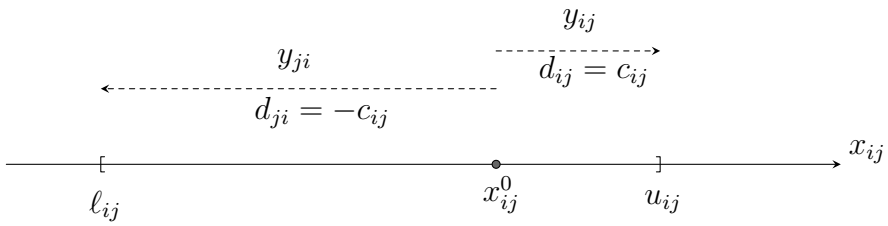


Fig. 5.19: A change of variables

Denoted $G(\mathbf{x}^0) := (N, A(\mathbf{x}^0))$, the residual network with respect to $\mathbf{x}^0$ corresponds to the change of variables $y_{ij} - y_{ji} := x_{ij} - x^0_{ij}$, $\forall (i,j) \in A$. It is based on the original nodes in N and the set of *residual* arcs $A(\mathbf{x}^0)$. Indeed, among the possible arc support $A' := \{(i,j) \cup (j,i) \mid (i,j) \in A\}$ only those arcs with positive residual capacities are of interest, i.e.,

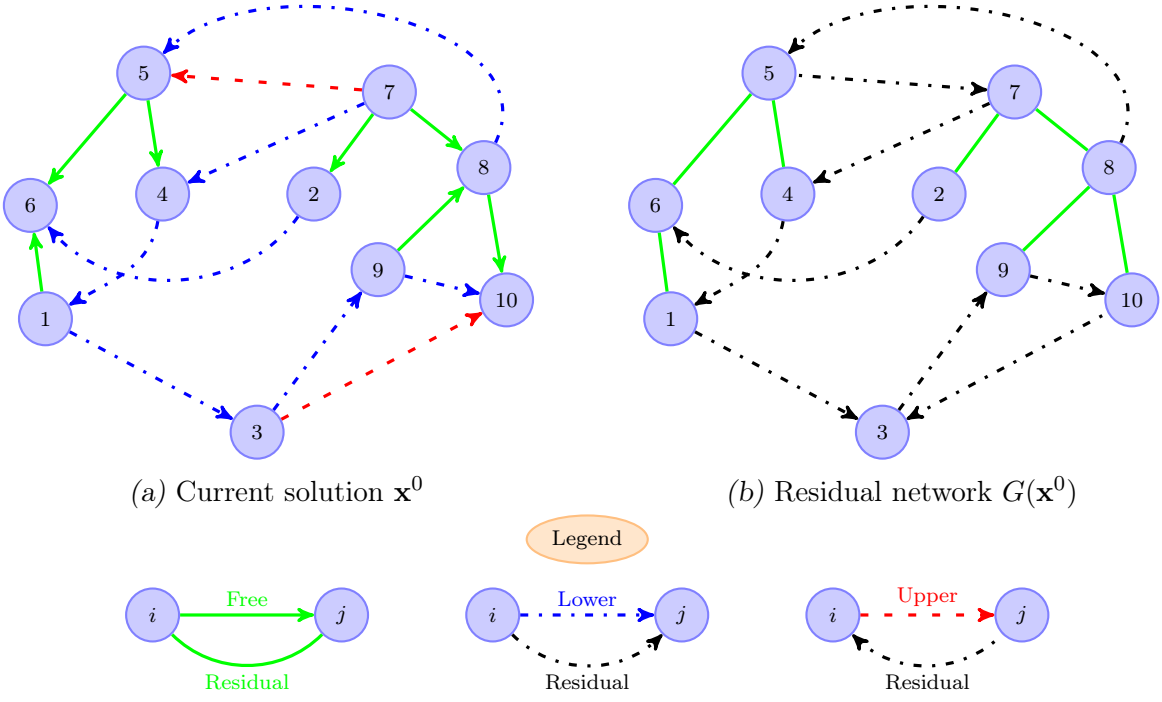
$$A(\mathbf{x}^0) := \{(i,j) \in A' \mid r^0_{ij} > 0\}. \tag{5.87}$$

An equivalent formulation of (5.86) states as

$$\begin{aligned} z^* = z^0 + \quad & \min \sum_{(i,j)\in A(\mathbf{x}^0)} d_{ij} y_{ij} \\ \text{s.t.} \quad & \sum_{j:(i,j)\in A(\mathbf{x}^0)} y_{ij} - \sum_{j:(j,i)\in A(\mathbf{x}^0)} y_{ji} = 0, \quad [\pi_i], \quad \forall i \in N, \\ & 0 \leq y_{ij} \leq r^0_{ij}, \qquad\qquad \forall (i,j) \in A(\mathbf{x}^0), \end{aligned} \tag{5.88}$$

where $z^0 := \mathbf{c}^\intercal \mathbf{x}^0$ is the objective function evaluation of solution $\mathbf{x}^0$. Furthermore, observe that traveling in both directions would be counterproductive and can be simplified to the net flow in a single direction. This means that the marginal flow must be such that $y_{ij}\, y_{ji} = 0,\ \forall (i,j) \in A$.

Take a look at Figure 5.20 which exhibits the construction of the residual network $G(\mathbf{x}^0)$. Figure 5.20a contains arc flow variables $x^0_{ij}, (i,j) \in A$ at different values. Each of these can be attributed a status depending on its actual value: *lower* when $x^0_{ij} = \ell_{ij}$, *upper* when $x^0_{ij} = u_{ij}$, or *free* when $\ell_{ij} < x^0_{ij} < u_{ij}$. When a variable is free, the flow can be carried in either direction thus meaning the presence of two residual arcs. However, when a variable is lower (resp. upper), this induces only one arc oriented in the forward (resp. backward) direction. Handled in a very similar fashion, arcs at their lower or upper bound are also said to be *restricted*, a generic term which simplifies the text when the differentiation makes no difference. Let the disjoint sets $L(\mathbf{x}^0) \cup U(\mathbf{x}^0) \cup F(\mathbf{x}^0) = A$ respectively echo this status partition of the original set of arcs at $\mathbf{x}^0$. In order to simplify the presentation, this partition is denoted as $\{L^k, U^k, F^k\}$, where $k \geq 0$ refers to the solution $\mathbf{x}^k$.



(a) Current solution $\mathbf{x}^0$ *(b)* Residual network $G(\mathbf{x}^0)$

Fig. 5.20: Residual network construction

5.2.2 Optimality conditions

It can be shown that a flow solution $\mathbf{x}^*$ is optimal if and only if the residual network $G(\mathbf{x}^*)$ contains no negative cost directed cycle, shortened to *negative cycle*, see Ahuja et al. (1993, Theorem 3.8). By iteratively canceling a negative cycle and updating the residual network accordingly, one obtains a generic cycle-canceling algorithm which terminates when there remains no negative cycle (Klein 1967).

For the record, although the notion of cycle cancellation can be seen as an intuitive step size method, its foremost intention is the *elimination* of said cycle from the current residual network. Indeed, passing a positive flow on any directed cycle W in $G(\mathbf{x}^0)$ obviously permits a transition between solutions, yet maintaining feasibility is ensured by limiting the flow to at most the smallest residual capacity of the arcs forming the cycle, say $\rho := \min_{(i,j)\in W} r^0_{ij}$. A directed cycle is *canceled* when the step size is equal to ρ such that at least one of the residual capacity $r^0_{ij}, (i,j) \in W$ is saturated.

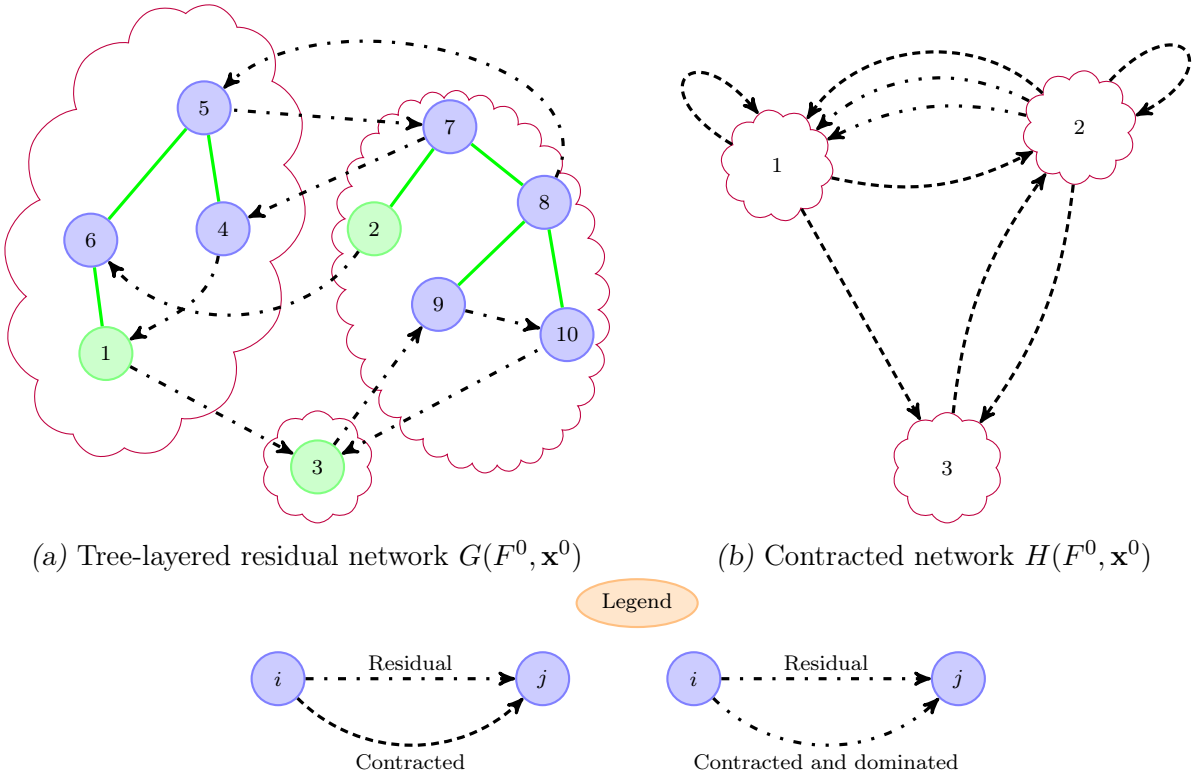
Since negative cycles are canceled sequentially, a cycle-canceling algorithm is basically defined by its oracle. The minimum mean cycle-canceling is one such algorithm famous for its strongly polynomial runtime complexity, see Gauthier et al. (2015b) for a recent survey. MMCC uses an oracle which dismisses residual capacities from the residual network and only identifies negative cycles of minimum average cost. Technical aspects of this algorithm are presented in Section 5.4.2 where we build upon these results to show the runtime complexity of the proposed Contraction-Expansion algorithm. The contracted network permits the exploration of alternative oracle constructions hoping to identify negative cycles faster. Whether these cycles are directed or not on the residual network remains to be seen.

5.2.3 Contracted network

The definition of a *cycle free* solution can be read in Ahuja et al. (1993) as a solution $\mathbf{x}^0$ which contains no cycle of free arcs. In linear programming terminology, such and only such solutions are basic. By extension, a linearly independent set of arcs cannot form a cycle. Also recall that a *tree* T is a connected graph that contains no cycle. With this in mind, if one assumes a basic solution $\mathbf{x}^0$, the set of free arcs F^0 then defines a forest, that is, a collection of node-disjoint trees.

Consider the following construction. By definition, each of these trees encapsulates a nonoverlapping set of nodes connected using a subset of arcs in F^0. Let each tree be rooted by an arbitrary node $\mathcal{R}(i) = i \in N$ such that any two nodes $i \neq j$ in N belonging to the same

tree must have the same root node $\mathcal{R}(i) = \mathcal{R}(j)$. As depicted in Figure 5.21a, root nodes 1, 2 and 3 have been chosen such that $\mathcal{R}(1) = \mathcal{R}(4) = \mathcal{R}(5) = \mathcal{R}(6) = 1$. Let $G(F^0, \mathbf{x}^0)$ be the residual network on $\mathbf{x}^0$ where the *tree-layer* defined with respect to the set of free arcs F^0 is superposed. Take notice that the tree-layer appears only as a visual aid such that for all intents and purposes, $G(F^0, \mathbf{x}^0) \equiv G(\mathbf{x}^0)$. In fact, the arcs within the clouds in Figure 5.21a are bidirectional such that one must imagine the tree-layer with respect to the original arcs in A. In performing the *contraction* of every tree to its root node, null variables y_{ij} corresponding to the respective $x^0_{ij}, (i,j) \in \{L^0, U^0\}$ are also caught in the crossfire. Each arc (i,j) is rerouted directly to the root nodes associated with its tail and head, that is, $\mathcal{R}(i)$ and $\mathcal{R}(j)$ respectively. Contrary to the initial assumption, this contraction gymnastic is likely to produce multiarcs. For instance, the arcs $(2,6), (7,4)$ and $(8,5)$ have the same head and tail in the contracted network. Granted the actual costs computation have yet to be addressed, trivial cost dominance rules can be applied on such multiarcs. The end result appears in Figure 5.21b where the presence of the two dominated arcs concerns only efficiency matters.



(a) Tree-layered residual network $G(F^0, \mathbf{x}^0)$ *(b)* Contracted network $H(F^0, \mathbf{x}^0)$

Fig. 5.21: Contracted network based on the set of free arcs

Let us formalize the construction of the contracted network using a general tree-layer definition. Indeed, while a great deal of attention is given to the set of free arcs F^0, the truth is that the tree-layer can be defined with respect to an arbitrary although *linearly independent*

set of arcs, say $P^0 \subset A$, where the superscript is omitted unless iterate comparisons are required. The set P is then used to partition the arcs of $A(\mathbf{x}^0)$ in two categories. As such, one can think of the sets $H_P(\mathbf{x}^0)$ and $V_P(\mathbf{x}^0)$ as those arcs that are *hidden* and *visible* in the contracted network. These sets are formed by

$$H_P(\mathbf{x}^0) := \bigcup_{(i,j)\in P} \begin{cases} (i,j),(j,i), & \text{if } (i,j) \in F^0 \\ (i,j), & \text{if } (i,j) \in L^0 \\ (j,i), & \text{if } (i,j) \in U^0 \end{cases} \tag{5.89}$$

$$V_P(\mathbf{x}^0) := A(\mathbf{x}^0) \setminus H_P(\mathbf{x}^0). \tag{5.90}$$

Definition 6. *With respect to the set P, the* tree-layer *identifies root nodes in the set*

$$N_P(\mathbf{x}^0) := \{i \in N \mid i = \mathcal{R}(i)\}, \tag{5.91}$$

as well as the arc partition $H_P(\mathbf{x}^0)$ and $V_P(\mathbf{x}^0)$ of the residual network $G(\mathbf{x}^0)$.

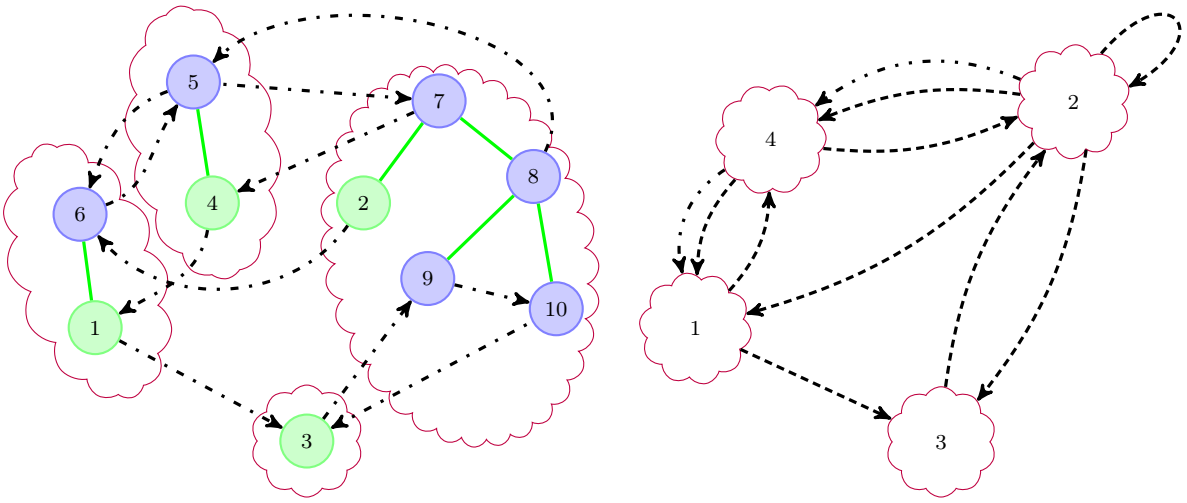
The structure of the contracted network $H(P, \mathbf{x}^0)$ is then obtained by using the sets of root nodes and visible arcs:

$$H(P, \mathbf{x}^0) := (N_P(\mathbf{x}^0), V_P(\mathbf{x}^0)). \tag{5.92}$$

Furthermore, the arcs of the contracted network could have also been defined with respect to said root nodes, say with the set $A_P(\mathbf{x}^0) := \{(\mathcal{R}(i), \mathcal{R}(j)) \mid (i,j) \in V_P(\mathbf{x}^0)\}$. Moreover, it is possible to maintain a bijection between the sets $V_P(\mathbf{x}^0)$ and $A_P(\mathbf{x}^0)$ which might be more perceivable when omitting dominance.

We present two alternative contraction examples which morally cover all possibilities. The first example uses a subset of free arcs $P \subset F^0$ while the second combines free arcs with some restricted arcs such that $P \supset F^0$. In the most general case, one can mix both possibilities by using some elements of F^0 and some of $A \setminus F^0$.

Figure 5.22 covers the first example where the subset of free arcs used is $P = F^0 \setminus \{(5,6)\}$. In Figure 5.22a, the free arc $(5,6)$ has been duplicated in both directions and sent to the visible set $V_P(\mathbf{x}^0)$. We refer to this kind of manipulation on free arcs as *coerced degeneracy*. This yields a larger contracted network since there is now a forest with four trees to handle. Figure 5.22b portrays the consequent contracted network.

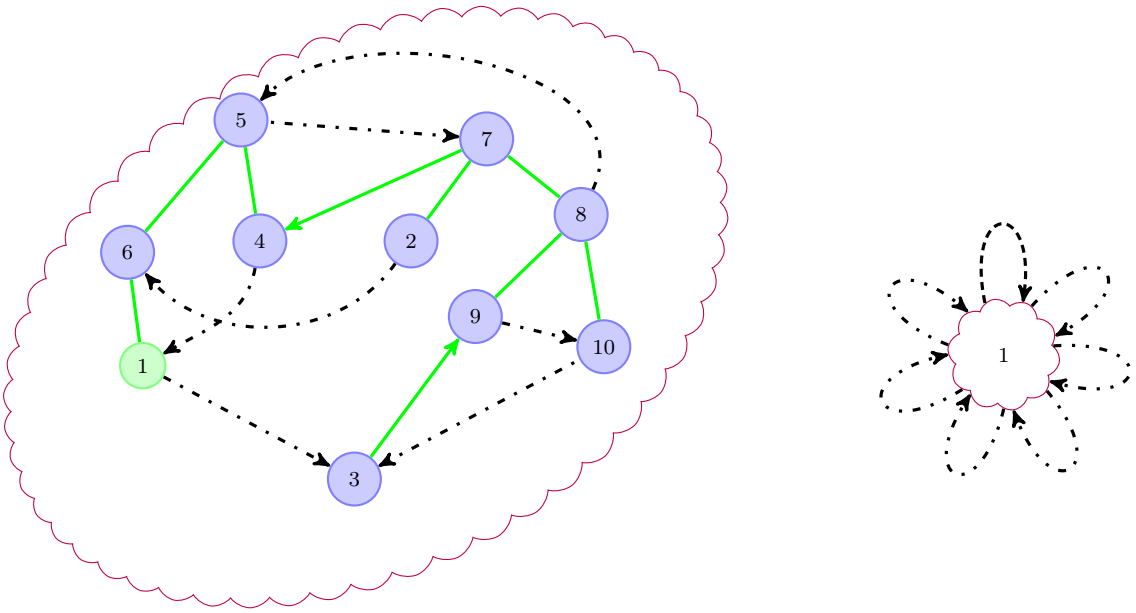


(a) Tree-layered residual network $G(P, \mathbf{x}^0)$ *(b)* Contracted network $H(P, \mathbf{x}^0)$

Fig. 5.22: Contracted network with coerced degeneracy on the free arc $(5, 6)$

As additional free arcs are duplicated in both directions, one eventually reaches a point where the set $P \subseteq F^0$ is empty, such that no free arcs are hidden at all, hence yielding a contracted network whose arc set is the same as that of the residual network, i.e., $V_\emptyset(\mathbf{x}^0) = A(\mathbf{x}^0)$. The reader is now invited to consider the other extreme case where the tree-layer consists of a single spanning tree. Such is the content of Figure 5.23 where the set $P = F^0 \cup \{(7, 4), (3, 9)\}$ consists of the union of the free arcs along with two additional independent arcs at their bound. Without loss of generality, assume these two arcs are basic degenerate in the simplex sense such that $P = B^0 \equiv B(\mathbf{x}^0)$ corresponds to a set of basic arcs at $\mathbf{x}^0$. The tree-layer $G(B^0, \mathbf{x}^0)$ seen in Figure 5.23a then splits the arcs in two subsets: the nine basic arcs of the spanning tree and the seven nonbasic arcs. The contracted network $H(B^0, \mathbf{x}^0)$ appears in Figure 5.23b, where the spanning tree is contracted to the sole root node and where each nonbasic arc becomes a loop, indeed, a directed cycle on $H(B^0, \mathbf{x}^0)$.

When all is said and done, only two cases are possible for an arc contained in the selected set P: it is either free or it is not. Bare with us, the arc cost computations of the contracted network are coming in the midst of the following section where features and properties of the contracted network are derived according to the choice of the set P, most notably the nature of the optimality conditions the oracle derived from the contracted network is able to verify.



(a) Tree-layered residual network $G(B^0, \mathbf{x}^0)$ *(b)* Contracted network $H(B^0, \mathbf{x}^0)$

Fig. 5.23: Contracted network with a set of basic arcs (primal network simplex algorithm)

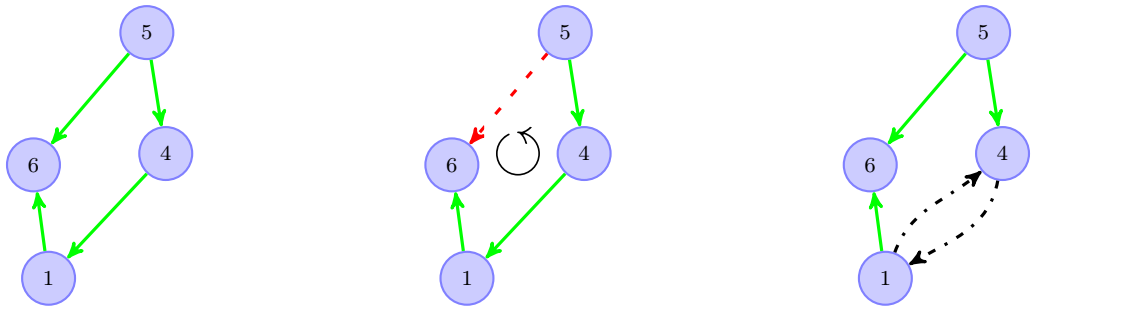
5.3 Contracted network properties

Although the selection of the set P is fairly arbitrary, Section 5.3.1 addresses how easy it is to meet the linear independence requirement regardless of the nonbasic nature of the current solution $\mathbf{x}^0$. Section 5.3.2 then state that any *contracted cycle*, a directed cycle identified on the contracted network, is *uniquely* extended on the residual network $G(\mathbf{x}^0)$. The analysis of the contracted network's arc costs is performed in Section 5.3.3 which is followed with the so-called pricing problem and the algorithm in Section 5.3.4. We then derive in Section 5.3.5 the optimality certificate whose nature depends on the selected set P. As the set P influences the content of the contracted network, different known algorithms are referenced in Section 5.3.6 by examining the possible outcomes of the pricing problem. Finally, Section 5.3.7 shows that once the set P is selected, the remaining arbitrary decisions that must be made have no impact on the algorithm.

5.3.1 Nonbasic solution

When defined with respect to the set F^0 which in turn trivially verifies the linear independence assumption when the current solution $\mathbf{x}^0$ is basic, we show that the basic status is not restrictive for the selection of the set P. Figure 5.24a presents a nonbasic solution where a cycle of free arcs is detected in G. Figures 5.24b and 5.24c handle the issue using two alternative

mechanisms. With the first mechanism, the cycle of free arcs is canceled, in either direction, yielding at least one restricted arc within the cycle. Since the cancellation ultimately modifies the current solution, one may altogether prefer the improving direction, say saturating the arc $(5, 6)$ to its upper bound. With the second mechanism, the arc $(4, 1)$ is coerced degenerate in G. This duplicating manipulation does not change the current solution yet provides a fast way to eliminate the cycles of free arcs thus allowing one to define the set P using only independent arcs of F^0.



(a) Cycle of free arcs *(b)* Cycle canceled *(c)* Arc $(4, 1)$ coerced degenerate

Fig. 5.24: Working with a nonbasic feasible solution on network G

Once one lets $f := |F^0| \leq m$, Proposition 20 asserts that it is easy to maintain a working tree-layer regardless of whether $\mathbf{x}^0$ is basic or not.

Proposition 20. *A linear independent set P can be derived from F^0 by either removing any cycle of free arcs using at most f cycle cancellations or applying a trivial coerce degeneracy scheme to at most f arcs.*

Proof. It is trivial to verify that rendering at most f arcs coerced degenerate means that there remains a suitable linearly independent subset of free variables capable of forming a tree-layer. In fact, one may think of these coerced degenerate variables as super-basics such that canceling a cycle of free arcs amounts to the *recovery of an optimal [or not] basis* which can be done in a straightforward manner according to Marsten et al. (1989). □

5.3.2 Uniqueness of the extended cycle

Observe that in the tree-layered residual network, any visible arc $(i, j) \in V_P(\mathbf{x})$ connects two root nodes using first a path from $\mathcal{R}(i)$ to i followed by arc (i, j) and second a path from j to $\mathcal{R}(j)$. Figure 5.25 portrays such an alternated path-arc-path sequence from now on called a *rooted path*. When performing the contraction, it is convenient to let $\mathcal{P}(i), i \in N$

be the path from node i to root node $\mathcal{R}(i)$. The rooted path associated with the visible arc $(i,j) \in V_P(\mathbf{x})$ can therefore be decomposed as $\{-\mathcal{P}(i), (i,j), \mathcal{P}(j)\}$, where $-\mathcal{P}(i)$ is the reversed path $\mathcal{P}(i)$ and the orientation of a free arc is the one given by (i,j).

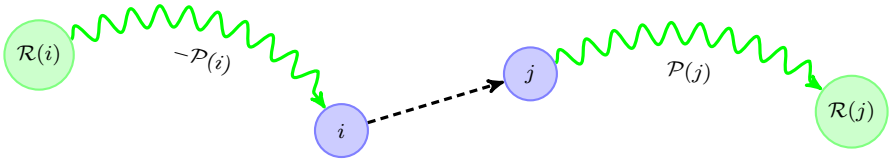


Fig. 5.25: Rooted path

Several remarks about the rooted path are noteworthy. The two root nodes can eventually be the same, see the arc compatibility Definition 7. The two distinct paths are formed using only arcs in $H_P(\mathbf{x}^0)$, that is, hidden arcs in tree structures. Either or both of these paths could be of null length such that the defining arc $(i,j) \in V_P(\mathbf{x}^0)$ could support the rooted path by itself. Finally, by definition of a tree, any visible arc $(i,j) \in V_P(\mathbf{x}^0)$ induces a *unique* rooted path. Given the bijection between $V_P(\mathbf{x}^0)$ and $A_P(\mathbf{x}^0)$, the same description holds for any contracted arc $(\mathcal{R}(i), \mathcal{R}(j)) \in A_P(\mathbf{x}^0)$.

Proposition 21. *Any contracted cycle W_H obtained on the network $H(P, \mathbf{x}^0)$ yields a unique extended cycle $W_{H:G}$ on the residual network $G(\mathbf{x}^0)$.*

Proof. Since only arcs part of the bijection are used to produce contracted cycles, the uniqueness of rooted paths guarantees the extended cycle uniquely exists in the tree-layered residual network. Once again, the orientation of the free arcs in the extended cycle is given by that of the contracted cycle. □

Implementation. The remainder of this section is dedicated to describing a contracted cycle obtained on the contracted network $H(P, \mathbf{x}^0)$ in more details. Let

$$W_H := \{(\mathcal{R}(i_1), \mathcal{R}(j_1)), (\mathcal{R}(i_2), \mathcal{R}(j_2)), \ldots, (\mathcal{R}(i_{|W_H|}), \mathcal{R}(j_{|W_H|}))\}$$

be one such directed cycle on the contracted network. Observe that it produces a sequence of *entry* and *exit* nodes in the tree-layer which ultimately cycle through the same root nodes, i.e., $\mathcal{R}_s := \mathcal{R}(j_s) = \mathcal{R}(i_{s+1}), \forall s \in \{1, \ldots, |W_H|\}$, where $i_{|W_H|+1}$ abusively equals to i_1. The extended cycle, seen on the residual network $G(\mathbf{x}^0)$, can then be expressed as a combination of rooted paths. Although, while the rooted paths simplify the work of extracting the extended

cycle via concatenation, we underscore that the latter eventually requires a quick fix whose goal is to provide an elementary path between j_s and i_{s+1}.

One may take a look at Figure 5.26 should the following explanation require visual support. Think of the path contained in a given tree, say between the arcs (i_s, j_s) and $(i_{s+1}, j_{s+1}), s \in \{1, \ldots, |W_H|\}$. Notice that both paths $\mathcal{P}(j_s)$ and $-\mathcal{P}(i_{s+1})$ have at least one node in common which is incidentally the root node $\mathcal{R}_s$. A cycle is formed during the concatenation if and only if there exists some other common node $u_s \neq \mathcal{R}_s$. Consider for instance that $\mathcal{P}(j_s)$ follows $j_s \rightsquigarrow u_s$ and then $u_s \rightsquigarrow \mathcal{R}_s$ while $-\mathcal{P}(i_{s+1})$ travels from $\mathcal{R}_s \rightsquigarrow u_s$ followed by $u_s \rightsquigarrow i_{s+1}$. Since all nodes from u_s to $\mathcal{R}_s$ are common to both paths, an elementary path is found by detecting the node $u_s \in \mathcal{P}(j_s)$ closest to j_s thus eliminating any back and forth play across opposing arcs. At last, the extended cycle $W_{H:G}$ extracted from the contracted cycle W_H is given by

$$W_{H:G} := \bigcup_{s=1}^{|W_H|} \{(i_s, j_s), \mathcal{P}(j_s) \setminus \mathcal{P}(u_s), -\mathcal{P}(i_{s+1}) \setminus -\mathcal{P}(u_s)\}, \tag{5.93}$$

where some of the composing paths $\mathcal{P}(j_s) \cup -\mathcal{P}(i_{s+1}), \forall s \in \{1, \ldots, |W_H|\}$ may be truncated to obtain an elementary description.

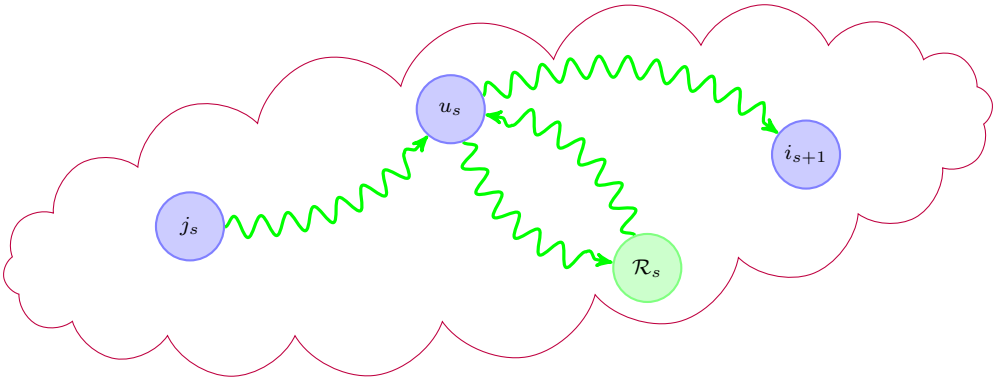


Fig. 5.26: Elementary tree path detection

5.3.3 Arc cost transfer policy

So far, it has been established that any contracted cycle W_H exists uniquely as $W_{H:G}$ on the residual network. In order to extend optimality conditions to the contracted network, we require a somewhat opposite feature: if the residual network contains a negative cycle then so must the contracted network. While there are probably many ways to ensure this, one straightforward way is to manipulate the dual variables such that Proposition 22 is

true. Recall that with respect to any dual variable vector $\boldsymbol{\pi} := (\pi_i)_{i \in N}$, the reduced cost of $x_{ij}, (i,j) \in A$ is defined as $\bar{c}_{ij} := c_{ij} - \pi_i + \pi_j$. Let the reduced cost $\bar{d}_{ij} := d_{ij} - \pi_i + \pi_j$ of $y_{ij}, (i,j) \in A(\mathbf{x}^0)$ be computed in the same way.

Proposition 22. *The reduced cost of the contracted cycle W_H is equal to the cost of its extended counterpart $W_{H:G}$.*

Proof. For $i \in N$, let π_i be the cost of the path $\mathcal{P}(i)$, that is, from i to $\mathcal{R}(i)$ in the tree-layered residual network. All root nodes are consequently assigned a null dual variable value and moreover the reduced cost of any arc in the forest is zero, i.e., $\bar{d}_{ij} = 0, \forall (i,j) \in H_P(\mathbf{x}^0)$, indeed those arcs that are hidden in the contracted network. Hiding the zero-cost arcs of $W_{H:G}$ makes the reduced cost of the contracted cycle W_H equal to the reduced cost of its extended version. By Proposition 19, as the reduced cost and the cost of a cycle (directed or not) are the same, we have the requested result. □

5.3.4 Contraction algorithm

As all relevant parameters have been established, we are ready to present a contraction algorithm. Alike the pricing problem of the minimum mean cycle-canceling algorithm, residual capacities are omitted (not that we would have a meaningful way to attribute these values to contracted arcs anyway) and we aim to detect the presence of a negative cycle using the contracted network $H(P, \mathbf{x}^0) = (N_P(\mathbf{x}^0), V_P(\mathbf{x}^0))$ as well as a convexity constraint. Proposition 23 states the kind of cycles that are identified using the following pricing problem:

$$
\begin{aligned}
\mu_H \quad &:= \quad \min_{(i,j) \in V_P(\mathbf{x}^0)} \bar{d}_{ij} y_{ij} \\
\text{s.t.} \quad \sum_{(i,j) \in V_P(\mathbf{x}^0) | \mathcal{R}(i) = \ell} y_{ij} &- \sum_{(i,j) \in V_P(\mathbf{x}^0) | \mathcal{R}(j) = \ell} y_{ij} = 0, \quad [\pi_\ell], \quad \forall \ell \in N_P(\mathbf{x}^0), \\
&\sum_{(i,j) \in V_P(\mathbf{x}^0)} y_{ij} = 1, \quad [\mu], \\
&y_{ij} \geq 0, \qquad \forall (i,j) \in V_P(\mathbf{x}^0),
\end{aligned}
\tag{5.94}
$$

where dual variables appear on the right between brackets. Flow conservation constraints are defined for each root node where the indexes of the summations simply specify all arcs from $V_P(\mathbf{x}^0)$ that are related to the specified root nodes.

Proposition 23. *The pricing problem (5.94) finds a minimum average cost directed cycle on $H(P, \mathbf{x}^0)$, averaged over the number of arcs it contains.*

Proof. Charnes and Cooper (1962) show that linear fractional functionals can be equivalently solved using linear programming under mild conditions, namely a nonempty and bounded feasible region such that the denominator is either always positive or always negative. Assume the following linear fractional functional program $\{\min \mathbf{d}^\intercal\mathbf{x}/\mathbf{1}^\intercal\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{0}, \mathbf{x} \geq \mathbf{0}, \mathbf{1}^\intercal\mathbf{x} > 0\}$ satisfies all conditions such that an equivalent linear program writes as $\{\min \mathbf{d}^\intercal\mathbf{y} \mid \mathbf{A}\mathbf{y} = \mathbf{0}, \mathbf{1}^\intercal\mathbf{y} = 1, \mathbf{y} \geq \mathbf{0}\}$ where the change of variables $\mathbf{y} := \mathbf{x}/\mathbf{1}^\intercal\mathbf{x}$ is performed. Since the convexity constraint $\mathbf{1}^\intercal\mathbf{y} = 1$ implies $\mathbf{1}^\intercal\mathbf{x} > 0$ the transformation holds. Flow conservation implies that the solution vector $\mathbf{y}^*$ sports only two values: 0 or $1/w$, where $w = |W_H|$ when solving (5.94). Assuming $\mathbf{x} = w\mathbf{y}$, the change of variables is obviously verified since $\mathbf{1}^\intercal\mathbf{x} = w$. Finally, the pricing problem (5.94) optimizes over variable coefficients $\bar{\mathbf{d}}^\intercal = [d_{ij} - \pi_i + \pi_j]_{(i,j)\in V_P(\mathbf{x}^0)}$. The straightforward telescopic sum idea used to show the cost and reduced cost of a cycle are the same applies in the same fashion. □

Recall that by using the arc cost transfer policy described in Section 5.3.3, the dual variables associated with the root nodes are null. Solving (5.94) not only provides an optimal cycle W_H^0 of minimum average reduced cost μ_H^0 on the contracted network but also a new set of dual variable values for these root nodes. The following is a direct adaptation of Gauthier et al. (2015b, Proposition 3):

Proposition 24. *When solving the pricing problem (5.94) at iteration $k \geq 0$, there exists some dual variable values $\pi_\ell^k, \ell \in N_P(\mathbf{x}^k)$ such that $\bar{d}_{ij} - \pi^k_{\mathcal{R}(i)} + \pi^k_{\mathcal{R}(j)} \geq \mu_H^k, \forall (i,j) \in V_P(\mathbf{x}^k)$. By complementary slackness conditions, the latter in fact holds at equality for all positive variables, i.e., $\forall (i,j) \in W_H^k$.*

If the residual network contains a negative cycle, then by Proposition 22, so does the pricing problem (5.94). By construction, the current solution $\mathbf{x}^k, k \geq 0$ is then optimal when $\mu_H^k \geq 0$, that is, when the contracted network does not contain any negative cycle. Otherwise, given the identified contracted cycle W_H^k, one computes the nonnegative step size

$$\rho^k := \min_{(i,j)\in W^k_{H:G}} r^k_{ij} \geq 0, \tag{5.95}$$

which is zero only if an arc of the extended cycle $W^k_{H:G}$ has a null residual capacity. We then obtain the solution $\mathbf{x}^{k+1}$ where the flow update, if any, is only performed on the arcs of $W^k_{H:G}$,

that is,

$$x_{ij}^{k+1} := \begin{cases} x_{ij}^k + \rho^k, & \forall (i,j) \in W_{H:G}^k \mid (i,j) \in A \\ x_{ij}^k - \rho^k, & \forall (i,j) \in W_{H:G}^k \mid (j,i) \in A \\ x_{ij}^k, & \text{otherwise} \end{cases} \tag{5.96}$$
$$z^{k+1} := z^k + \rho^k \ |W_H^k| \ \mu_H^k.$$

Observe that when $\rho^k = 0$, the set P^{k+1} must be different from P^k otherwise a dead-end is encountered and the algorithm does not terminate. Degeneracy and cycling phenomenons from the primal simplex come to mind. Indeed, unless rules for null step size cycles are included, the same minimum mean cycle is identified in the next iteration. A pseudo-code is elaborated in Figure 5.27 where this process is repeated until optimality is reached.

Initialization : Iteration $k := 0$;
Feasible solution $\mathbf{x}^0$;
Define the set $P^k \subset A$;
1 Derive the contracted network $H(P^k, \mathbf{x}^k)$;
2 Solve the pricing problem (5.94) for μ_H^k, W_H^k;
3 If $\mu_H^k \geq 0$, **terminate** with an optimality certificate for $\mathbf{x}^k$;
4 Extract the extended cycle $W_{H:G}^k$;
5 Compute the step size ρ^k (5.95);
6 Update to solution $\mathbf{x}^{k+1}$ using the system (5.96);
7 Update the set $P^{k+1} \subset A$;
8 Iterate $k := k + 1$;

Fig. 5.27: Generic contraction algorithm for network flow problems

The nature of the optimality certificate in the generic Contraction-Expansion algorithm goes hand in hand with the nondegenerate guarantee provided (or not) by the selected set P and the induced contracted network. Let us analyze this property.

5.3.5 Optimality conditions

Given that any contracted cycle is uniquely extended, the underlying expectation is that it is possible to travel on the hidden arcs, i.e., the extended cycle is directed on $G(\mathbf{x}^0)$. Depending on the actual status of the arcs in set P, this expectation could be challenged. Verifying that any set P which consists of only free arcs always provides a contracted cycle W_H on which at least one flow unit can always take place on the extended cycle $W_{H:G}$ is trivial since all arcs unaccounted for can be used in either directions. As supporting evidence, the first

three extended cycle extractions illustrated in Figure 5.28 are directed in $G(\mathbf{x}^0)$. They come from contracted cycles obtained on a contracted network using only free arcs (Figure 5.21b or 5.22b). The fourth extended cycle extraction comes from Figure 5.23b and yields an undirected cycle inducing a zero step size.

Figure 5.28a is obtained from a contracted cycle defined by the arc loop on node 1 in the contracted network of Figure 5.21b. Figure 5.28b comes from the same contracted network but the contracted cycle uses more arcs: $\{(1,3),(3,9),(8,5)\}$. Figure 5.28c identifies part of the previous cycle although it is based on the contracted network of Figure 5.22b where more visible arcs are available. Finally, Figure 5.28d is based on Figure 5.23b used with the primal network simplex algorithm. The arc $(7,4)$ is basic degenerate and one of the nonbasic arcs corresponds to the visible arc $(2,6)$, a loop on node 1. The corresponding extended cycle is defined by $\{(\mathbf{2},\mathbf{6}),\{\},\{(6,5),(5,4),(4,7),(7,2)\}\}$. Unfortunately, the arc $(4,7)$ is not a residual arc and reduces the possible flow on the identified extended cycle to zero. In simplex terms, pivoting on arc variable x_{26} induces a degenerate pivot.

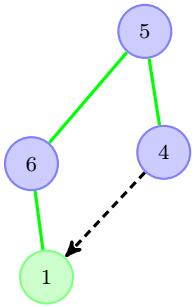
The contracted network $H(P,\mathbf{x}^0)$ is guaranteed to identify a directed extended cycle when $P \subseteq F^0$, that is, when the hidden arcs are free, any contracted cycle has a positive step size on $G(\mathbf{x}^0)$. On the other hand, when $P \nsubseteq F^0$ there is no such guarantee since the extended cycle could use an arc with a null residual capacity. Such is the proof of Proposition 25.

Proposition 25. *Given the set $P \subset A$ of linearly independent arcs, if*

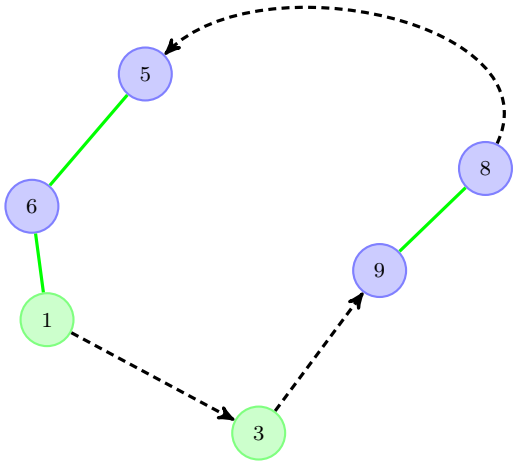
- *$P \subseteq F^0$, the extended cycle $W_{H:G}$ is directed on $G(\mathbf{x}^0)$ and the step size is positive. Therefore, the oracle (5.94) provides necessary and sufficient optimality conditions for (5.86).*
- *$P \nsubseteq F^0$, the extended cycle $W_{H:G}$ may be undirected on $G(\mathbf{x}^0)$ and the step size is greater than or equal to zero. In that case, the oracle (5.94) provides sufficient optimality conditions for (5.86).*

Regardless of the choice of the set P, a noteworthy observation is that arcs $(i,j) \in A$ can be categorized in two classes: those arcs that link nodes of the same tree, i.e., the head and tail of the residual arc have the same root node $\mathcal{R}(i) = \mathcal{R}(j)$ and those that link different trees, i.e., $\mathcal{R}(i) \neq \mathcal{R}(j)$. The former class is said to be *compatible* with P.

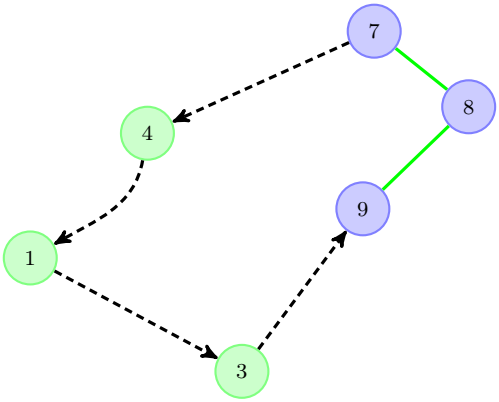
Definition 7. *An arc $(i,j) \in A$ is* compatible *with P if and only if the head and tail refer to the same root node in the contracted network, i.e., $\mathcal{R}(i) = \mathcal{R}(j)$. Otherwise, it is* incompatible.



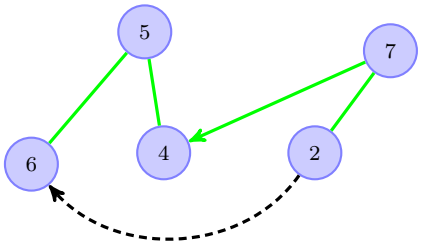
(a) Extended cycle from the contracted cycle $\{(4,1)\}$ on Figure 5.21



(b) Extended cycle from the contracted cycle $\{(1,3),\ (3,9),\ (8,5)\}$ on Figure 5.21



(c) Extended cycle from the contracted cycle $\{(1,3),\ (3,9),\ (7,4),\ (4,1)\}$ on Figure 5.22



(d) Extended cycle from the contracted cycle $\{(2,6)\}$ on Figure 5.23

Fig. 5.28: Extended cycle extractions

Examples of compatible arcs can be seen in Figure 5.21a as arcs $(4,1)$ and $(9,10)$ or in Figure 5.21b as contracted arc loops on root nodes 1 and 2. Compatible arcs are sometimes favored because these loops are easy to identify on the contracted network thus forming directed cycles on their own.

This also implies that the extended cycle associated with a compatible arc $(i,j) \in V_P(\mathbf{x}^0)$ is not influenced by the content of other trees. The associated cost is therefore the actual reduced cost of the arc. For incompatible arcs, rooted paths have to be combined to form cycles in $H(P,\mathbf{x}^0)$. In this respect, we like to think of the reduced costs $\bar{d}_{ij}, \forall (i,j) \in V_P(\mathbf{x}^0)$ as *rooted costs*. Also recall that the proposed arc cost transfer strategy is indeed used in the primal network simplex algorithm where a spanning tree and only one root node are

used. The reader may want to reread the leading statement of this section in light of the compatibility concept and the primal network simplex reconciliation.

5.3.6 Extremal point solution space

It is straightforward to verify that both extreme cases, $P = \emptyset$ and $P = B^0$ respectively correspond to the minimum mean cycle-canceling algorithm and the primal network simplex algorithm. In the former case, the contracted network $H(\emptyset, \mathbf{x}^0)$ has exactly the same structure as the residual network $G(\mathbf{x}^0)$ thus yielding a pricing problem equivalent to MMCC's. In the latter case, consider the contracted network of Figure 5.23 where the arcs of P are basic arcs (indeed forming the primal network simplex spanning tree) and all visible arcs are nonbasic. As each contracted arc is compatible, applying cost dominance trivially results in a single contender for the pricing problem to identify. Recall that rooted costs are equal to reduced costs for all compatible arcs such that the entering variable identified by Dantzig's pivot rule would be the same.

Let $\mathcal{C}_P$ be the set of all directed cycles obtainable as extreme points of the pricing problem (5.94). The cardinality of this set can be measured in two parts, that is, the compatible and incompatible portions. The first portion is insignificant and reduces to a single element regardless of the set P while the second grows exponentially as the set P reduces in size.

Proposition 26. *The cardinality of the extremal solution space $|\mathcal{C}_P|$ of the pricing problem (5.94) on the contracted network $H(P, \mathbf{x}^0)$ is $O\left(2^{(n-|P|+1)^2}\right)$.*

Proof. For the sake of simplicity, the argument is carried with a dominance scheme applied on the contracted arcs. Furthermore, we use a worst case type analysis on the density of these sets. As an extreme point of (5.94) can only identify a single compatible arc, dominance can be applied across all compatible arcs regardless of their root node association.

The more interesting portion thus concerns incompatible combinations which of course require at least two incompatible arcs. Using combinatorial enumeration, let us count the cycles in rough numbers by randomly selecting arcs within the available possibilities $V_P(\mathbf{x}^0)$ thus forming directed cycles of different sizes ranging from 2 to $n - |P| + 1$, the number of nodes in the contracted network, i.e., $|\mathcal{C}_P| = \sum_{k=2}^{n-|P|+1} \binom{|V_P(\mathbf{x}^0)|}{k}$. Then again, once dominance is applied, the set of contracted arcs vastly overestimates the actual number of arcs remaining

in the contracted network which for a complete graph amounts to $(n - |P| + 1)^2$. Since basic calculus reduces $\sum_{k=1}^{n} \binom{n^2}{k}$ to the dominant term 2^{n^2}, we obtain $|\mathcal{C}_P| \equiv O\left(2^{(n-|P|+1)^2}\right)$. □

Granted it is possible to order the cardinalities, $|\mathcal{C}_\emptyset| \gg |\mathcal{C}_F| \gg |\mathcal{C}_B|$, the same cannot be done for the actual sets. Indeed, hidden arcs and dominance rules means that directed cycles present in a smaller set are not necessarily present in a bigger one. Consider for instance Figures 5.28b and 5.28c where the cycle uses the nondominated arc $(8, 5)$ for the former and $(7, 4)$ for the latter.

While this computation tremendously dramatizes the size of the extreme point solution set of the pricing problem (5.94), it gives the general intuition that some balance can be achieved between the workload offset transferred to the oracle and the simplicity of obtaining undirected cycles that may induce null step size. Let us put this in perspective of the contracted networks seen in Figures 5.21b, 5.22b and 5.23b. The formulation of the pricing problem (5.94) is always the same, yet the contracted network reminiscing of the primal network simplex (5.23b) has $|\mathcal{C}_B| = 1$ whereas many more cycles are present in the contracted network using all free variables (5.21b), a number that increases exponentially as the contracted network gets larger. We are thus expecting that the resolution of the pricing problem (5.94) be harder as $|P|$ gets smaller.

5.3.7 Root selection

Since the selection of root nodes is arbitrary, the attentive reader might wonder what impact, if any, a different set of root nodes would have on the oracle and thus the algorithm's course. As shown in the proof of Proposition 27, it turns out very little. Indeed, the oracle's content is modified on a per component basis yet as a whole it is completely unaffected.

Proposition 27. *Root selection has no influence on the compatible set, the extreme point solution set of the pricing problem* (5.94)*, the average cost evaluation or the nature of the optimality certificate.*

Proof. On the one hand, it is no surprise that, for all nodes $i \in N$, the paths $\mathcal{P}(i)$ and their costs π_i are modified when an alternative set of root nodes is used. This means that while each contracted arc (i, j) of $V_P(\mathbf{x}^0)$ continues to exist on $H(P, \mathbf{x}^0)$, it is now associated with a different reduced or rooted cost $\bar{d}_{ij} = d_{ij} - \pi_i + \pi_j$. The first observation is that the set of extreme point solutions $\mathcal{C}_P$ corresponding to directed cycles obtained by linear combination of contracted arcs is unaltered. Furthermore, by Proposition 22 the original cost on each cycle is

maintained regardless of the root selection. As such, the average cost of every extended cycle is also maintained which obviously means as much for the nature of the optimality certificate. Finally, observe that not only is the compatibility status of any arc (i, j) persistent, the reduced or rooted cost of a compatible arc is also immune to change. □

Furthermore, the proof of Proposition 22 makes the same assumption as in Ahuja et al. (1993, Chapter 11.4), that is, a root node dual variable value is arbitrarily fixed to 0 since any constant θ can be added to the vector of dual variables without modifying the reduced costs. More generally, each root node dual variable value can be fixed to a different constant without any impact on the outcome of the pricing problem (5.94).

Proposition 28. *The dual variable of each root node $\pi_i,\ i \in N_P(\mathbf{x}^0)$, can be initialized to an arbitrary but fixed value. The remaining dual variable values $\pi_i,\ i \in N \setminus N_P(\mathbf{x}^0)$ are then the cost of the path $\mathcal{P}(i)$ plus the root node dual variable value $\pi_{\mathcal{R}(i)}$.*

5.4 Behavioral study

Supported with numerical results from a minimum cost flow problem containing 1,025 nodes and 92,550 arcs referred to as Instance 1, Section 5.4.1 analyzes the behavior of a specific variant of the contraction algorithm opposite MMCC's established behavior. This behavioral study not only serves to grasp some mechanical aspects of these algorithms, but more importantly draws attention on key points of the theoretical minimum mean cycle-canceling complexity. Section 5.4.2 connects the dots with Cancel-and-Tighten and digs into more advanced aspects.

5.4.1 A lower bound on the minimum mean cost

Our analysis opposes the mean cost of negative cycles obtained on the contracted network to those that would be obtained in the framework of MMCC. At a given iteration k, let μ_G^k be the minimum mean cost of the directed cycle W_G^k on the residual network as obtained by MMCC. We then denote by μ_H^k the minimum mean cost of the directed cycle W_H^k fetched on the contracted network and

$$\mu_{H:G}^k = \mu_H^k \frac{|W_H^k|}{|W_{H:G}^k|} \tag{5.97}$$

the mean cost of the extended cycle where hidden arcs are accounted for. The following proposition is verified by construction.

Proposition 29. *Let* $\mathbf{x}^k$ *be a nonoptimal solution. Given the set* $P \subseteq F^k$*, the following ordering of the minimum mean costs always holds:*

$$\mu_H^k \leq \mu_G^k \leq \mu_{H:G}^k. \tag{5.98}$$

Proof. The cycle $W_{H:G}^k$ is visible as is on the residual network $G(\mathbf{x}^k)$ which means $\mu_G^k \leq \mu_{H:G}^k$ trivially holds by the nature of minimizing the mean cost in $G(\mathbf{x}^k)$. Furthermore, although contracted, any cycle W_G^k that also appears in $H(P, \mathbf{x}^k)$ is evaluated with a different mean cycle cost which eventually uses less arcs such that $\mu_H^k \leq \mu_G^k$. □

Since we aim to establish complexity results in light of MMCC's analysis, it is quite natural to think of W_G^k as the reference minimum mean cost cycle. The inequalities can be interpreted in a straightforward manner: The contracted cycle appears to provide a lower minimum mean cost μ_H^k yet once the hidden arcs are accounted for, this illusion disappears and indeed shows that μ_G^k is overestimated by $\mu_{H:G}^k$. Observe that the ordering (5.98) is equal throughout if there are no hidden arcs in the contracted cycle, i.e., $|W_H^k| = |W_{H:G}^k|$.

The notion of estimation might be better understood by observing the evolution of the various μ during the resolution process. For illustrative purposes, we systematically use the contracted network $H(F^k, \mathbf{x}^k)$ at every iteration, where all and only the free arcs are hidden. The value of μ_H^k and its counterpart $\mu_{H:G}^k$ are naturally obtained on top of which we also poll for the value μ_G^k as if to look for the minimum mean cycle. Figure 5.29 shows the evolution of these measures for Instance 1.

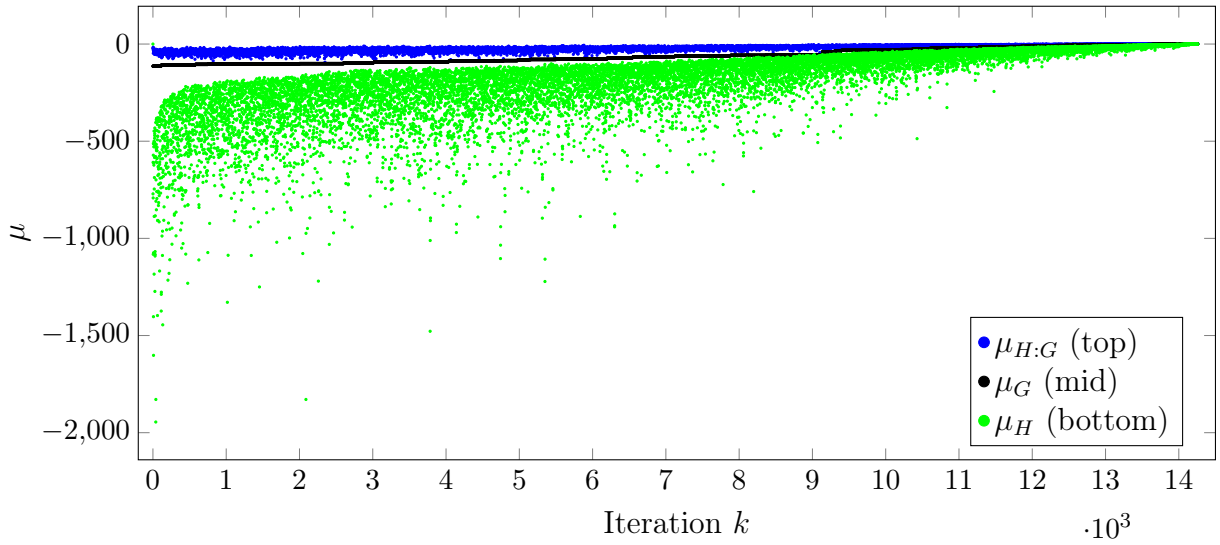


Fig. 5.29: Comparison of μ_H^k, μ_G^k and $\mu_{H:G}^k$

As expected, the plot verifies the aforementioned ordering of Proposition 29. While all three plots show a general increasing tendency, what is striking is how different the inequality $\mu_H^k \leq \mu_G^k$ is from $\mu_G^k \leq \mu_{H:G}^k$. The first gap is fairly intuitive and goes back to basic mathematics: the density of the contracted network produces contracted cycles which use relatively few arcs compared to their extended counterparts. As the cycle costs remain the same, the denominator strongly influences the mean evaluation.

The second gap is much more tight and deserves more attention. In this matter, Figure 5.30 zooms on the evolution of the extended mean cycle cost $\mu_{H:G}^k$ and minimum mean cycle cost μ_G^k. For the record, MMCC both searches and applies the minimum mean cycle at each iteration yielding an algorithm which features a nondecreasing property on μ_G^k, see Goldberg and Tarjan (1989, Lemma 3.5) or Gauthier et al. (2015b, Proposition 4). Take a close look around the 10,000$^{\text{th}}$ iteration. This is not a graphical aberration showing that this property is indeed lost when negative cycles are not canceled in the order suggested by MMCC. What we think is surprising for this particular instance is how little this phenomenon occurs, only 21 times within 14,258 iterations to be exact.

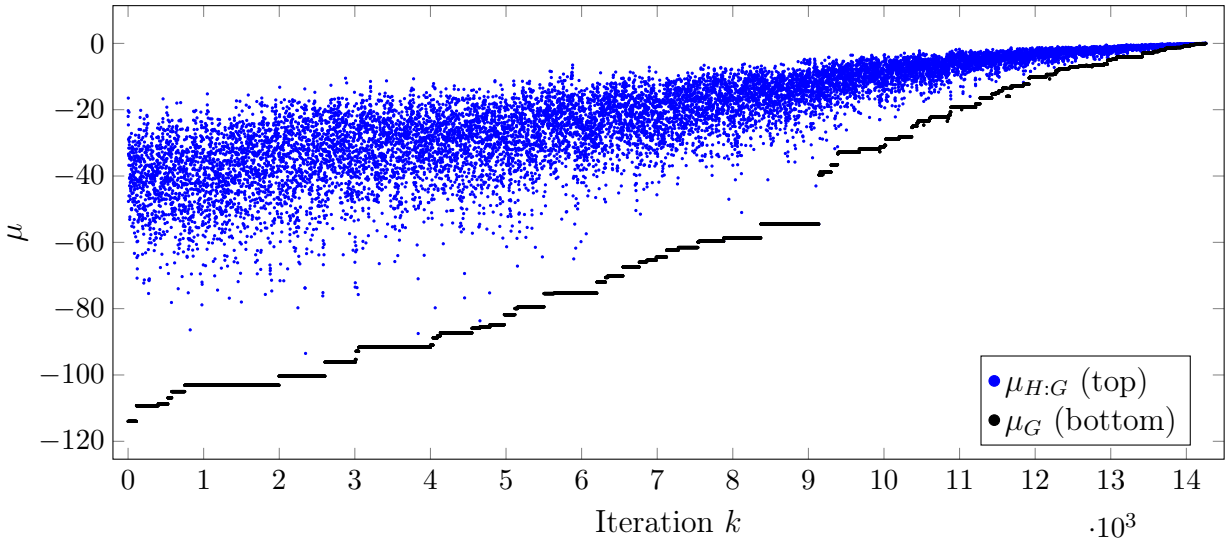


Fig. 5.30: Zoom on comparison of μ_G^k and $\mu_{H:G}^k$

Convergence. As the contraction algorithm identifies a negative cycle at every iteration, it is evident that convergence of the objective function to optimality is guaranteed. By default, there are then at most $O(mCU)$ such negative cycles, a weakly polynomial result referring to the largest absolute cost ($C := \max_{(i,j)\in A} |c_{ij}|$) and interval bound values ($U := \max_{(i,j)\in A} u_{ij} - \ell_{ij}$).

Nevertheless, comparing the canceled cycle average cost to that of the minimum mean cycle cost is an enlightening exercise. Indeed, the chaotic behavior also appears in a strongly polynomial algorithm, namely Cancel-and-Tighten. Section 5.4.2 recalls some concepts from the minimum mean cycle-canceling algorithm (Gauthier et al. 2015b) for which strongly polynomial properties are established through the analysis of the so-called *optimality parameter*.

5.4.2 Optimality parameter analysis

Strongly polynomial runtime complexity certifies that an algorithm performs a bounded number of operations measured solely by the size of its input (see Cormen et al. (2009) for further inquiries). For network flow problems, this bound should be a function of m and n only. In introducing the minimum mean cycle-canceling algorithm, Goldberg and Tarjan (1989) also provide such a complexity proof which holds in two parts. First, an oracle capable of producing the *minimum mean cost directed cycle* in $O(mn)$ time. Second, an optimal solution is reached by canceling at most $O(m^2 n \log n)$ such negative cycles. The first part can be seen as the inner loop while the second the outer loop. Their product then yields a strongly polynomial global complexity. Radzik and Goldberg (1994) refine the complexity analysis, reducing the number of cycle cancellations to $O(m^2 n)$. Although they also introduce the concept of *phases* to analyze the behavior of the algorithm, Gauthier et al. (2015b) strongly insist on the latter to further improve the complexity result by combining the Cancel-and-Tighten strategy introduced in Goldberg and Tarjan (1989) with the original algorithm.

Type 2 (negative) cycles. Even though the nondecreasing property of μ_G^k across iterations in the minimum mean cycle-canceling algorithm is interesting, it has been argued by Gauthier et al. (2015b) that the strictly increasing behavior observed across *phases* is more enlightening. The phase definition goes hand in hand with the proof of Gauthier et al. (2015b, Proposition 5) which distinguishes between Type 1 and Type 2 cycles. It shows that a Type 2 cycle is attained within m cancellations or optimality is achieved, where a Type 2 cycle is obtained when there exists at least one variable in the cycle with a nonnegative reduced cost computed with respect to a set of dual variables established at the beginning of a phase. Given μ_G^h at the beginning of phase $h \geq 0$, such a Type 2 cycle W^l then serves to imply a strict increase on μ_G^h as

$$\mu_G^l \geq (1 - 1/|W^l|)\, \mu_G^h \geq (1 - 1/n)\, \mu_G^h \tag{5.99}$$

thus marking the end of the phase, i.e., the sequence of iterations leading to this *measurable jump factor*.

At the end of the day, since each phase contains at most m cancellations, the number of cancellations can be interpreted as at most $O(mn \log n)$ or $O(mn)$ phases depending on the complexity point of view (Gauthier et al. 2015b), respectively Theorem 2 using the minimal increasing factor $(1 - 1/n)$ and Theorem 3 rather exploiting the stronger factor $(1 - 1/|W^l|)$. Truth be told, while the concept of phases is useful for the complexity analysis of MMCC, it is not transparent at all in the implementation. Indeed, in MMCC, the phase is purely a question of theoretical existence; dual variables are never required to advance such that the resolution process cares not about these cycle Types. The enlightenment comes from the inclusion of Cancel-and-Tighten (CT) in the analysis where phases are observed *in actu*. The latter fixes dual variables and depletes the residual network of Type 1 cycles (those formed with negative reduced cost arcs only). Once this Cancel-step is terminated, one must conclude that a Type 2 cycle comes next thus implying a jump with respect to some lower bound $\hat{\mu}^h_{CT}$ on the minimum mean cycle on μ^h_G at the beginning of phase h.

Figure 5.31 opposes the mean value μ^k_{CT} at iteration k of the Type 1 cycles canceled in the Cancel-and-Tighten implementation with the lower bound value $\hat{\mu}^k_{CT} \leq \mu^k_G$ proposed by Gauthier et al. (2015b, Proposition 14). While the chaotic behavior of μ^k_{CT} can clearly be observed throughout the resolution, a general pattern of increase can be noted across the phases. The minimum (i.e., optimal) mean cycle value μ^k_G is once again fetched as background information. The 4,900$^{\text{th}}$ iteration is worth a look: again, a small decrease for μ^k_G. In total, four such occurrences within 7,294 cancellations contained in 357 phases. Furthermore, Cancel-and-Tighten maintains strongly polynomial properties despite the usage of Type 1 cycles going against the nondecreasing property of μ^k_G. The *strictly increasing lower bounds $\hat{\mu}^h_{CT}$ between phases* obtained with the existence of a Type 2 cycle marking the end of phase h is indeed where the properties are established (Gauthier et al. 2015b, Theorem 6).

Arc fixing. Strongly polynomial time complexity is achieved by keeping track of the number of phases through the concept of *arc fixing* as seen in the minimum mean cycle-canceling algorithm (Gauthier et al. 2015b, Propositions 9 and 11). A relaxation of the arc fixing condition is used in Proposition 30, where the two different arc fixing conditions are in line with the complexity point of views, that is, $O(mn \log n)$ or $O(mn)$ phases. The proof is straightforward and recuperates the Cancel-and-Tighten proof adaptation where the lower bound values $\hat{\mu}^h_{CT}$ are shown to mimic the behavior of the true value μ^h_G. Indeed, as mentioned

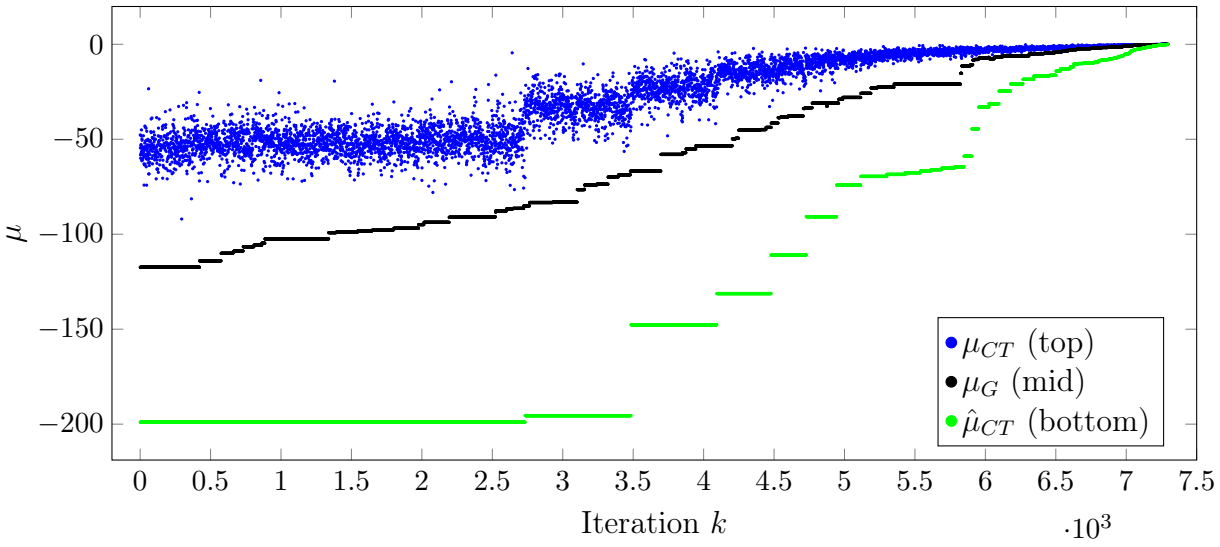


Fig. 5.31: Comparison of μ^k_{CT}, μ^k_G and $\hat{\mu}^k_{CT}$

in Gauthier et al. (2015b, p. 131), it is conceivable to rewrite the arc fixing conditions using a lower bound $\hat{\mu}^h$ on μ^h_G instead.

Proposition 30. *Assuming $\hat{\mu}^h$ is a lower bound for μ^h_G at the beginning of phase h.*

- *Arc fixing for arc $(i,j) \in A$ occurs or has already occurred if $\left|\bar{c}^h_{ij}\right| \geq -2n\hat{\mu}^h$.*
- *Implicit arc fixing for arc $(i,j) \in A$ occurs or has already occurred if $\left|\bar{c}^*_{ij}\right| > -n\hat{\mu}^h$, where $\bar{c}^*_{ij}$ is the reduced cost of arc (i,j) computed with an optimal set of dual variables.*

Optimality parameter $\hat{\mu}^h$. It is at this point important to create separation between μ^k_G at iteration k and strongly polynomial properties. In MMCC, μ^k_G is coined as the *optimality parameter*: it converges without oscillations to 0 from below in strongly polynomial time. It might however be more appropriate to reserve this name for another value as illustrated by the Cancel-and-Tighten strategy in the above example. Indeed, whether one looks at μ^k_H or μ^k_{CT} or any other cycle-canceling scheme, the minimum mean cycle μ^k_G can always be fetched as secondary information (recall that $\mu^k_H \leq \mu^k_G \leq \mu^k_{H:G}$ and $\hat{\mu}^k_{CT} \leq \mu^k_G \leq \mu^k_{CT}$). If strongly polynomial properties are to be established on oscillating values from one iteration to the next, it appears mandatory to divert the analysis to a lower bound $\hat{\mu}^h$ over the phases instead. In other words, while the portion above μ^k_G (blue) may be unpredictable, it is the portion below μ^k_G (green) that should be well-behaved. We are ready to propose an adaptation of the contraction algorithm.

5.5 Contraction-Expansion algorithm

Section 5.5.1 introduces a flexible phase definition based on so-called Type 3 cycles that serve as end-phase markers. This is followed in Section 5.5.2 by a discussion of an expansion scheme which controls the visible and hidden arc sets, that is the content of the set P, for the proposed algorithm. Section 5.5.3 argues that applying the contraction on the residual network and expanding the contracted network using that specific expansion scheme as the algorithm progresses produces a strongly polynomial algorithm. In Section 5.5.4, we show that the expansion scheme is not unique such that different strategies can be used to improve the algorithm. Finally, computational results are presented in Section 5.5.5.

5.5.1 End-phase markers

Our contribution is to revisit the phase definition in order to extract the true pertinent information which allows convergence in strongly polynomial time. In this respect, the current phase definition is built upon a weak jump condition which waits for the identification of a Type 2 cycle as the minimum mean cycle to confirm a jump on μ_G^h at the end of phase h. Let us propose a more flexible definition.

Definition 8. *A* phase $h \geq 0$ *is a sequence of cycle cancellations terminated whenever a* measurable jump *is observed in strongly polynomial metrics, that is, both the factor and the time required to obtain it are expressed in strongly polynomial time. A phase solution* $\mathbf{x}^h$ *is understood as the solution at the beginning of phase* h*.*

The factor proposed in the following cycle Type obviously satisfies the strongly polynomial requirement whereas the time requirement is shown in Proposition 31.

Definition 9. *Let* $\mathbf{x}^h$ *be a nonoptimal phase solution. At an iteration* t *within the phase* h*, let us call a cycle* W^t *on* $G(\mathbf{x}^t)$ *a* Type 3 (negative) cycle *if its underestimated mean cost* $\hat{\mu}^t$ *(with* $\hat{\mu}^t \leq \mu_G^t \leq \mu^t$*) produces the measurable jump*

$$\hat{\mu}^t \geq \left(1 - \frac{1}{|W^t|}\right) \mu_G^h. \tag{5.100}$$

Proposition 31. *In MMCC, a Type 3 cycle is identified within at most* m *cancellations.*

Proof. Assume the phase h. The proof is trivial and connects with the existence of a Type 2 cycle, say W^l, which must have $\mu_G^l = \hat{\mu}^l \geq (1 - 1/|W^l|)\, \mu_G^h$. Therefore, the iteration t either happens simultaneously as l or earlier. □

Observe that whether the jump factor is obtained using an actual Type 2 cycle or not is irrelevant: a phase is completed in accordance with Definition 8. In other words, the only important aspect of the Type 2 cycle is the measurable jump it procures on μ_G^h, a tactic which can incidentally also be verified against a lower bound according to Proposition 30, that is,

$$\hat{\mu}^t \geq \left(1 - \frac{1}{|W^t|}\right) \hat{\mu}^h. \tag{5.101}$$

This is true for $\hat{\mu}^h = \hat{\mu}_{CT}^h$ in Cancel-and-Tighten (Gauthier et al. 2015b, Theorem 3) and is also used with $\hat{\mu}^h = \mu_H^h$ in the proposed Contraction-Expansion algorithm.

5.5.2 Expansion scheme

The focus on phases should by now be realized by the reader. Figure 5.32 presents a pseudo-code for the Contraction-Expansion algorithm whereas the following paragraphs explain how the proposed expansion scheme constrains the latter into producing such phases.

Let $\mathbf{x}^0$ be a nonoptimal solution at iteration $k = 0$. Note that $\mathbf{x}^0$ is also a nonoptimal phase solution ensuring a lower bound $\hat{\mu}^0 \leq \mu_G^0$. Let the set $P^0 = F^0$. By Proposition 25, the extended cycle $W_{H:G}^0$ is directed on the residual network $G(\mathbf{x}^0)$ and the step size is positive.

When $W_{H:G}^0$ is canceled, the aftermath is hard to predict but one thing is for certain: only the arcs part of $W_{H:G}^0$ are affected. Some arcs that were free in $\mathbf{x}^0$ remain free in $\mathbf{x}^1$ in the next iteration while all the other arcs have changed status either from restricted lower to restricted upper and vice versa, from free to restricted or from restricted to free for a total of six possibilities. Four of these end up with a new status restricted whereas two end up with a free status. Intuitively speaking, since the contraction happens around the free variables and the nondecreasing μ_G^k property is lost when the contraction is systematically applied, let us concentrate on controlling these newly freed variables in $\mathbf{x}^1$, i.e., $F^1 \setminus F^0$. By applying coerced degeneracy on these specific variables, the hidden set then contains only those variables that were already free in $\mathbf{x}^0$ and are still free in $\mathbf{x}^1$. This amounts to selecting the set $P^1 = F^1 \cap F^0$ at iteration $k = 1$.

From there, the idea is simple, repeatedly apply cycle cancellation with the extended cycle $W_{H:G}^k$ at iteration k and expand the contracted network using the intersection of variables free in both the previous and new solution until the algorithm reaches a Type 3 cycle, that is, a negative cycle producing a jump factor of at least $(1 - 1/|W_{H:G}^k|)$. The phase $h = 0$ is then terminated and a new phase begins. Since we may at this point reapply the full contraction,

Initialization : Iteration $k := 0$;
Phase $h := 0$, $\hat{\mu}^0 := -\max_{(i,j)\in A} |c_{ij}|$, *new phase* := true;
Feasible solution $\mathbf{x}^0$;
if *new phase* **then**
  new phase := false;
  counter := 0;
  Eliminate cycles of free arcs from $\mathbf{x}^k$;
  Build partition F^k, L^k, U^k;
  Define the set $P^k := F^k$;
Derive the contracted network $H(P^k, \mathbf{x}^k)$;
Solve the pricing problem (5.94) for μ_H^k, W_H^k;
If $\mu_H^k \geq 0$, **terminate** with an optimality certificate for $\mathbf{x}^k$;
Extract the extended cycle $W_{H:G}^k$;
Compute the positive step size ρ^k from (5.95);
Update to solution $\mathbf{x}^{k+1}$ using the system (5.96);
if $\mu_H^k \geq (1 - 1/|W_{H:G}^k|)\, \hat{\mu}^h$ **then**
  $\hat{\mu}^{h+1} := \mu_H^k$;
  $h := h + 1$;
  new phase := true;
else
  counter := counter + 1;
  if *counter* $< n$ **then**
    Expand the contracted network with $P^{k+1} := F^{k+1} \cap F^k$;
  else
    $P^{k+1} := \emptyset$;
Iterate $k := k + 1$;

Fig. 5.32: Contraction-Expansion algorithm

it is worthwhile to eliminate any cycle of free arcs from the phase solution to maximize the contraction benefit.

5.5.3 *Complexity analysis*

The complexity analysis of the Contraction-Expansion algorithm revolves around bringing the resolution process of the latter to terms with MMCC's behavior. As testified by Cancel-and-Tighten, these terms refer to the strictly increasing phase markers. Figure 5.33 might help to get a feel for this endeavor. It displays the value of μ_H^k using the minimal expansion scheme $P^{k+1} = F^{k+1} \cap F^k$ where each trail corresponds to a phase during the resolution of Instance 1.

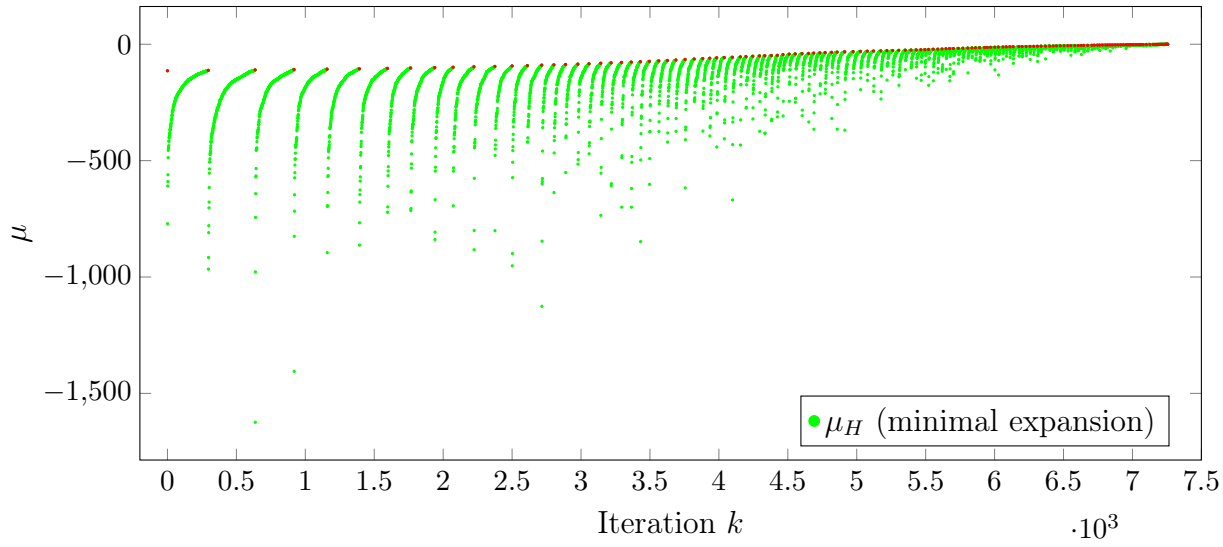


Fig. 5.33: Evolution of μ_H^k with the Contraction-Expansion algorithm

The upcoming analysis focuses on the outer loop, that is, we examine the time required to reach the end of a phase and the total number of such phases.

Proposition 32. *The Contraction-Expansion algorithm completes a phase in $O(m^2 n)$ time.*

Proof. The proof is threefold with respect to the expansion scheme portion of the algorithm. We prove that 1) the nondecreasing property of μ_H^k during the phase is maintained, 2) at most $O(n)$ contracted cycle cancellations are required to reach a contracted network equivalent to the residual network, and 3) finding the minimum average reduced cost cycle on the contracted network requires $O(mn)$ time.

Gauthier et al. (2015b, Proposition 4) states that a cycle cancellation on the residual network $G(\mathbf{x}^k)$ cannot introduce a minimum mean cycle in $G(\mathbf{x}^{k+1})$ yielding $\mu_G^{k+1} < \mu_G^k$ in the following iteration. Here is an adaptation for the expansion scheme. Consider the contracted network $H(P^k, \mathbf{x}^k)$ and an optimal contracted cycle W_H^k of average reduced cost μ_H^k along with an optimal set of dual variables $\pi_\ell^k, \ell \in N_P(\mathbf{x}^k)$ on the root nodes (Proposition 24). From there, fix the root nodes dual variables to these new values and update the remaining dual variables accordingly (Proposition 28). The residual network $G(\mathbf{x}^k)$ hence satisfies $\bar{d}_{ij} - \pi^k_{\mathcal{R}(i)} + \pi^k_{\mathcal{R}(j)} \geq \mu_H^k, \forall (i,j) \in A(\mathbf{x}^k)$ and in particular $\bar{d}_{ij} = 0, \forall (i,j) \in H_P(\mathbf{x}^k)$. Upon canceling the expanded cycle $W_{H:G}^k$ and obtaining $\mathbf{x}^{k+1}$, every new residual arcs in $G(\mathbf{x}^{k+1})$ either has a reduced cost of 0 or $-\mu_H^k$, i.e., $\bar{d}_{ij} - \pi^k_{\mathcal{R}(i)} + \pi^k_{\mathcal{R}(j)} \geq \mu_H^k, \forall (i,j) \in A(\mathbf{x}^{k+1})$. Observe that every arc in $F^k \cap F^{k+1}$ already has a reduced cost of 0 such that the arc cost transfer policy already holds. The contraction is readily available with every remaining visible arc

evaluated at a reduced cost greater than or equal to μ_H^k. The average cost of the next contracted cycle is then at least μ_H^k.

Recall that the pricing problem in MMCC can be derived from the pricing problem (5.94) by making visible all residual arcs, that is, by setting $P = \emptyset$. Then, observe that the set P is updated by intersecting the sets of free variables of the previous solution with the current one such that its size either stays the same or decreases at each cancellation. Assuming the initial phase solution $\mathbf{x}^0$ is a basic solution, at most $|F^0| \equiv O(n)$ cancellations yielding a decrease are then obviously required to reach $P = \emptyset$. Furthermore, when a cycle is canceled without modifying the set of free arc variables, it means that some restricted variables changed from one bound to another. Unfortunately, this kind of pathological phenomenon is what nightmarish instances are made of. An iteration counter limiting the number of cancellations prior to reaching the residual network to at most n is put in place to make the move directly should it be necessary. Trivially, at most n cycle cancellations allows the expansion scheme to reach a contracted network equivalent to the residual network. From there, a Type 3 cycle is ensured within an additional m cycle cancellations (Proposition 31). To sum up, at most $n + m \equiv O(m)$ cycle cancellations are required to meet the required jump.

Finally, solving the pricing problem (5.94), which can still be accomplished in $O(mn)$ time using dynamic programming (Karp 1978), dominates all the other operations performed at every iteration. Indeed, the data structure is maintained in $O(m)$ time while the extended cycle extraction, step size computation and solution update is done in $O(n)$ time. A phase is ultimately completed in at most $O(m)$ iterations each one requiring at most $O(mn)$ for a total phase runtime of $O(m^2n)$. □

Theorem 11. *The Contraction-Expansion algorithm runs in $O(m^3n^2)$ strongly polynomial time.*

Proof. We show that at most $O(mn)$ phases can occur in accordance with Gauthier et al. (2015b, Theorem 3). Since $\mu_H^k \leq \mu_G^k$ is indeed a lower bound for the true minimum mean cycle cost value by Proposition 29, this is also true for any phase solution. As soon as $\mu_H^k \geq \hat{\mu}^h \left(1 - 1/|W_{H:G}^k|\right)$, the phase h ends and the lower bound is increased to $\hat{\mu}^{h+1} := \mu_H^k$. By Proposition 30, arc fixing occurs on the lower bound value $\hat{\mu}^h$ as well such that the phase time complexities are valid. By Proposition 32, since each phase runs in $O(m^2n)$, the compound time is obtained. An initial valid lower bound for μ_G^0 can be trivially obtained with $\hat{\mu}^0 := -\max_{(i,j)\in A} |c_{ij}|$. □

The proposed phase definition along with the Type 3 cycle not only satisfy theoretical properties of the strongly polynomial complexity of MMCC, they also express very practical

observations. The hope is that not only the Type 3 cycle occurs much faster than m cancellations, it also happens while the phase still exploits the contracted network.

5.5.4 Alternative end-phase markers and expansion schemes

So long as strongly polynomial phase time is maintained, alternative expansion schemes may be used. Figures 5.34-5.35 show the evolution of μ_H^k for two simple variations. The first variation (cycle expansion) updates the set P with a more aggressive reduction, i.e., a *faster* expansion. The update writes as

$$P^{k+1} = \left(F^{k+1} \cap F^k\right) \setminus \{(i,j) \in A \mid (i,j) \in W_{H:G}^k \text{ or } (j,i) \in W_{H:G}^k\} \tag{5.102}$$

such that all arcs of the expanded cycle $W_{H:G}^k$ that are still free (thus common to F^k and F^{k+1}) are also removed from P^{k+1}. The second variation (cycle expansion & loops) uses the same update and also uses a post cycle-cancellation heuristic which cancels loops derived from all compatible variables with negative rooted costs as well.

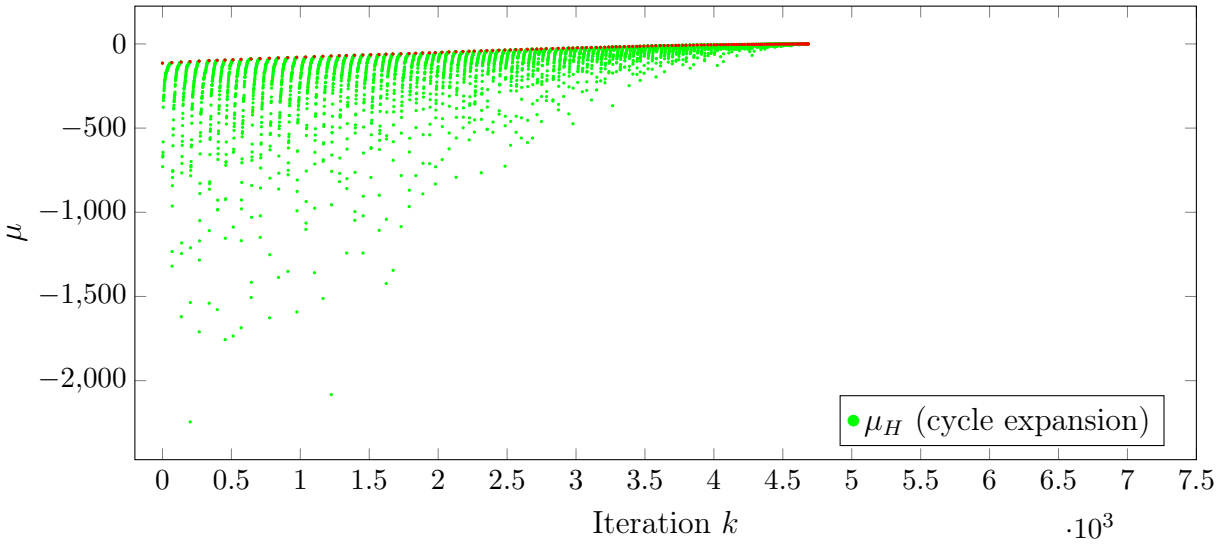


Fig. 5.34: Evolution of μ_H^k with cycle expansion

For anyone familiar with successful divide-and-conquer methods, the resolution speed typically benefits from the decomposition at a higher rate then the cost of the latter. While postponing the end of a phase with a less aggressive expansion scheme appears to agree with this statement, expanding the contracted network faster also means that end-phase markers are reached faster at which point the full contraction is reapplied. It seems that reaching said end-phase markers as fast as possible is of particular interest.

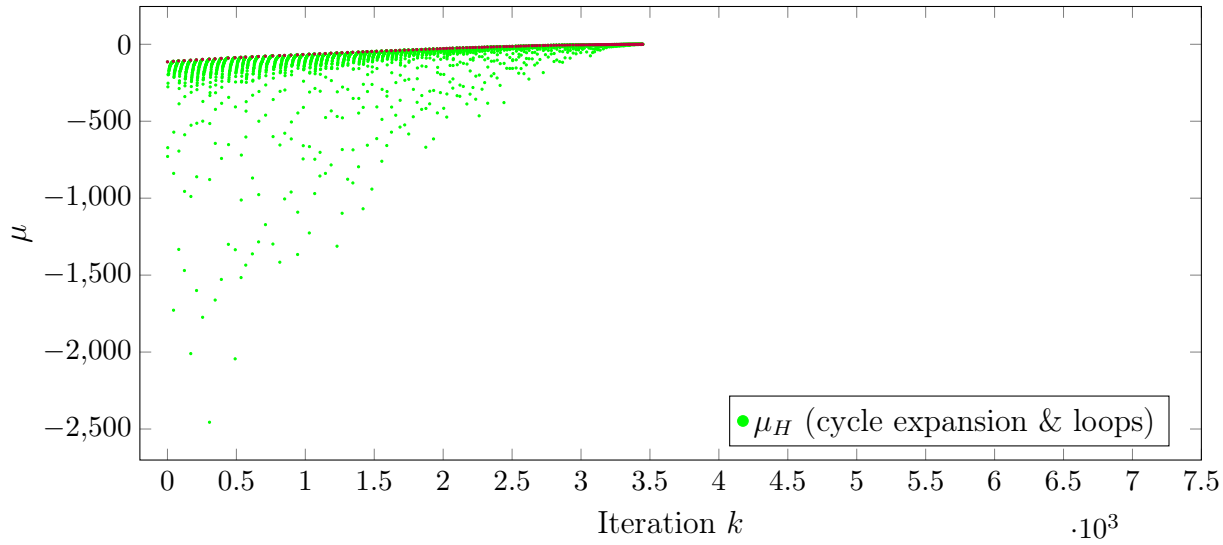


Fig. 5.35: Evolution of μ_H^k with cycle expansion and cancellation of negative loops

Speaking of end-phase markers, a measurable jump could be established using alternative Type 3 cycle definitions. Consider for instance the modification of (5.101) as follows:

$$\hat{\mu}^t \geq \left(1 - \frac{1}{|W^t|}\right)\hat{\mu}^h \geq \left(1 - \frac{1}{\max_{k\in h}|W^k|}\right)\hat{\mu}^h \geq \left(1 - \frac{1}{n}\right)\hat{\mu}^h. \tag{5.103}$$

where $k \in h$ is understood as an iteration k within the phase h. While the last criterion is reminiscing of Theorem 2 which contents itself with the same jump every time thus obtaining the $O(mn \log n)$ phases, the second criterion also tracks a cycle length and compromises on the desired jump. The latter is in fact the criterion used in all plots of the Contraction-Expansion algorithm.

5.5.5 *Computational experiments*

This section is separated in two parts. The first details computational results with respect to three specific instances and serves to show CE indeed behaves as expected when identifying smaller cycles on the contracted network. The second complements intuitive assertions made in the first part with a computational profile on benchmark instances from DIMACS (1990–1991). For the record, these results are obtained using an `Intel` i7-4770K@3.50GHz with 16GB of RAM.

Table 5.7 resumes the content of plots displayed in Figures 5.33–5.35 along with their computational times as well as averages of cycle sizes and induced step sizes. The performances

of MMCC and CT are added for order comparison purposes whereas results regarding CE also incorporate an average contraction level given by $\overline{\alpha}$ where $\alpha^k = |N_P(\mathbf{x}^k)|/n, \forall k \geq 0$.

Resolution	CPU (sec)	k / cycles	h	$\overline{\lvert W \rvert}$	$\overline{\rho}$	$\overline{\alpha}$
Instance 1						
MMCC	91.52	4,296	90	27.11	1.30	–
CE - minimal expansion	87.59	7,256	151	5.46	1.30	0.32
CE - cycle expansion	63.98	4,686	144	5.19	1.70	0.47
CE - cycle expansion/loops	51.84	3,449/9,499	142	7.40/3.32	1.61/2.23	0.58
CT	0.82	7,294	357	21.62	1.37	–
gridgen_sr_13a						
MMCC	15,629.40	20,893	289	72.85	7.45	–
CE - minimal expansion	6,709.96	25,291	301	5.93	9.70	0.56
CE - cycle expansion	2,435.97	9,202	236	4.70	19.18	0.62
CE - cycle expansion/loops	1,906.97	6,034/23,429	224	7.13/2.58	18.53/36.01	0.70
CT	36.17	27,473	1,482	49.80	7.28	–
road_paths_05_WI_a						
MMCC	9,552.52	1,119	300	82.27	1.00	–
CE - cycle expansion	10,355.30	1,122	348	82.20	1.00	1.00
CT	2,389.51	3,248	18,647	34.85	1.00	–

Tab. 5.7: Computational results for variations of CE opposite MMCC and CT

First of all, there is no denying that Cancel-and-Tighten is orders of magnitude faster than MMCC. The proposed Contraction-Expansion algorithm sits somewhere in between although it is clear that the contraction boosts the speed of fetching negative cycles by a significant amount even more so as the size of the instance gets larger. The same feel is palpable across other benchmark instances such as the significantly larger problem gridgen_sr_13a from the `GRIDGEN` family which contains 8,191 nodes and 745,381 arcs. It is important to understand that the benefit of the contraction comes from the degeneracy observed in encountered solutions. For instance, the problem road_paths_05_WI_a ($n = 519,157$ and $m = 1,266,876$) from the `ROAD` family is structured with uniform capacities at 1 such that there are never any free variables, hence no contraction. We therefore just get penalized with the contraction computational overhead.

The remainder of this section looks at the `GRIDGRAPH` family and brings further testimonies to the previous claims. For each of the three categories of problems (**long**, **square**, **wide**) ranging from sizes 2^8 to 2^{17}, Table 5.8 displays the average contraction level and CPU time CPU_{CE} for CE. As expected the latter increases with the instance size yet the average contraction level of about 60% appears stable across this family. The middle columns indicate the relative CPU time calculated with respect to MMCC's as $\beta = CPU_{CE}/CPU_{MMCC}$. For instance, CE requires 2,619 seconds to solve grid_square_16 meaning that MMCC took

2,619/0.21 ≈ 12,471 seconds to terminate. The relative computing times β are plotted in Figure 5.36. Once again, the relative performance of CE increases with the instance size.

	long			**square**			**wide**		
n	CPU_{CE} (sec)	β	$\overline{\alpha}$	CPU_{CE} (sec)	β	$\overline{\alpha}$	CPU_{CE} (sec)	β	$\overline{\alpha}$
2^8	0.01	0.65	0.64	0.01	0.60	0.64	0.01	0.60	0.64
2^9	0.02	0.44	0.59	0.04	0.43	0.56	0.05	0.50	0.59
2^{10}	0.08	0.31	0.56	0.16	0.36	0.58	0.21	0.41	0.61
2^{11}	0.18	0.26	0.61	0.64	0.30	0.56	0.93	0.44	0.64
2^{12}	0.59	0.19	0.56	2.95	0.29	0.58	4.43	0.38	0.64
2^{13}	1.51	0.12	0.50	11.96	0.26	0.57	19.72	0.39	0.64
2^{14}	5.45	0.16	0.45	59.18	0.26	0.58	89.83	0.41	0.63
2^{15}	8.51	0.08	0.51	325.44	0.22	0.60	476.19	0.38	0.63
2^{16}	23.15	0.04	0.45	2,619.07	0.21	0.60	2,911.39	0.31	0.63
2^{17}	402.98	0.13	0.45	17,890.40	0.27	0.61	15,944.70	0.34	0.63

Tab. 5.8: Computational results for CE on the `GRIDGRAPH` family, version `A`

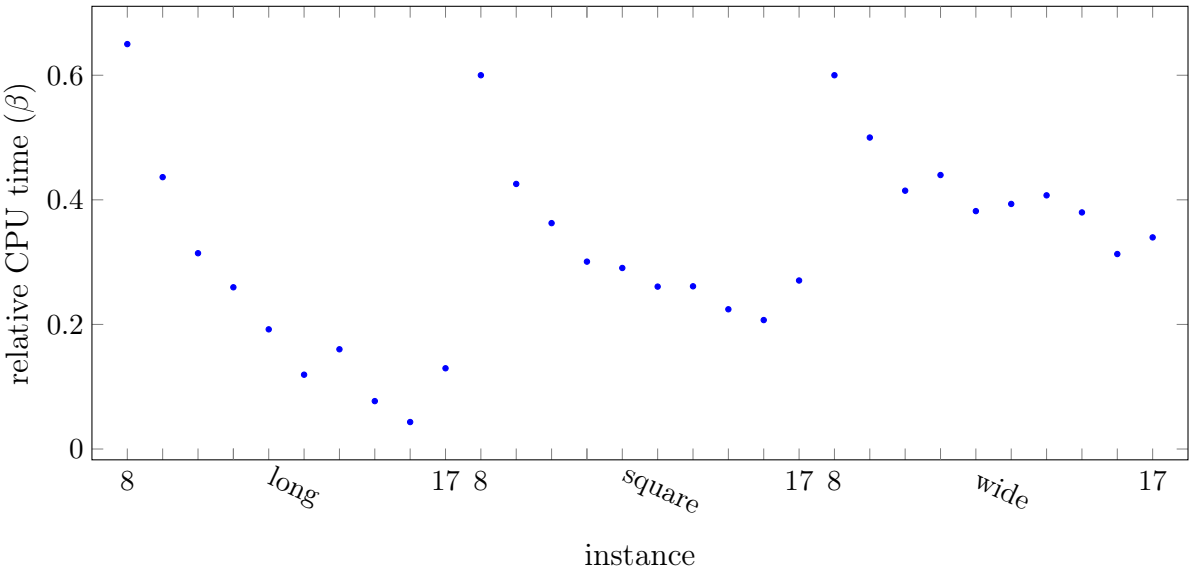


Fig. 5.36: Relative CPU time for CE when solving the `GRIDGRAPH` family, version `A`

While we certainly did not cover the complete benchmark suite, this paper does not pretend to a competitive algorithm just yet. Case in point, even larger instances up to 2^{20} nodes are available but MMCC's resolution proved to be far too demanding. We also believe the omitted instance versions and/or more results from other families would not significantly contribute to this framework.

5.6 Conclusion

We start with a note addressed to users of the primal network simplex algorithm. Observe that the spirit of the minimum mean cycle-canceling algorithm is tangibly similar to that of PS. The pricing step is home to the optimality certificate whereby the latter is acquired unless an improving direction is identified. It turns out this is not all that surprising since it has been shown that PS and MMCC belong to a more generic contraction algorithm, indeed extending to both ends of its spectrum.

A variety of special cases inducing positive step sizes is also identified. By combining these with results from MMCC, a strongly polynomial Contraction-Expansion algorithm which behaves much better than the former, especially as the instance's size increases, comes to life. The reader is carried around this birth process by opposing the behavior of MMCC and Cancel-and-Tighten in a computational study. Both iteration and phase bases are illustrated, although a strong emphasis on the latter concept is systematically done thus providing an alternative way of showing strongly polynomial properties.

Such a property can also be observed in this framework when using partial contraction. The latter is obtained by modifying the choice of hidden arcs as the algorithm progresses. The selection is made in such a way that it actually corresponds to an *expansion* of the contracted network. Furthermore, the proposed Contraction-Expansion algorithm sticks to practical observations otherwise overlooked in MMCC. As such, phase markers are verified algorithmically rather than just existing for theoretical purposes. It even recuperates the idea that not all jumps are created equal thus underlining the important aspect of Type 2 cycles, namely the measurable jump. The Type 3 cycle is born. The strongly polynomial argument uses both phases and Type 3 cycles on top of which the convergence of the original *optimality parameter* is neglected in favor of a lower bound.

There seems to be some arbitrage to be done between trying to meet optimality conditions in a more aggressive manner and the work required to do so. By contenting itself with a sufficient condition, a significant proportion of cancellations performed in PS are degenerate whereas MMCC uses a rule whose computational footprint is too high. That being said, the contracted network is closer to the spirit of an oracle than the residual network is. By this, we mean that it matters not to see all directed negative cost cycles so long as at least one can be detected thus allowing to improve and repeat.

Acknowledgements

Jacques Desrosiers acknowledges the Natural Sciences and Engineering Research Council of Canada for its financial support.

Vector space decomposition for linear and network flow problems

Jean Bertrand Gauthier and Jacques Desrosiers

GERAD & HEC Montréal, Canada

3000, chemin de la Côte-Sainte-Catherine

Montréal (Québec) Canada, H3T 2A7

<jean-bertrand.gauthier, jacques.desrosiers>@hec.ca

Marco E. Lübbecke

RWTH Aachen University, Germany

Kackertstraße 7

D-52072 Aachen, Germany

marco.luebbecke@rwth-aachen.de

ABSTRACT

This paper describes a vector space decomposition algorithmic framework for linear programming guided by dual feasibility considerations. The resolution process moves from one solution to the next according to an exchange mechanism which is defined by a direction and a post-evaluated step size. A framework for a family of primal algorithms if you will. The core component of this direction is obtained via a pricing problem devised in primal and dual forms. From the dual perspective, one maximizes the minimum reduced cost that can be achieved upon dividing the set of dual variables in two subsets: one being fixed while the other is optimized. From the primal perspective, one selects a convex combination of the variables part of this direction which turns out to be incomplete with respect to the original problem. This direction is however uniquely completed by identifying affected variables, if any. The degeneracy status of the current solution can be exploited in specific variants.

This unified framework is presented in a generic format although it borrows several concepts from network flow problems. Different variants can be extracted several of which corresponding to well known algorithms. The most known special case is the primal simplex algorithm where all dual variables are fixed: this results in the choice of a single entering variable commonly leading to degenerate pivots. At the other extreme, we find the so-called minimum mean *weighted* cycle-canceling algorithm, a perfect example of network flow analogies introduced in this paper, for which all dual variables are optimized at every iteration. Somewhere in between these two extremes lies another special case, the improved primal simplex algorithm for which one fixes the dual variables associated with the nondegenerate basic variables and optimizes the remaining dual variables. The two last variants both bestow a pricing problem providing necessary and sufficient optimality conditions. As a result, directions yielding positive step sizes are also issued from these pricing steps. These directions move on the edges of the polyhedron for the latter while the former can identify interior directions by combining edge directions.

Keywords: primal simplex algorithm, degeneracy, optimized reduced costs, combination of variables, positive step size algorithms, vector space.

6.1 Introduction

While Dantzig's primal simplex algorithm (PS) dates back to the summer of 1947 (Dantzig and Thapa 1997, p. xxvi), it still commands attention today. The race for shaving seconds off the resolution process started soon after its birth and indeed remains strong today. Although researchers keep breaking records, the original simplicity of the algorithm has been traded for it. In other words, it is hard to deny that PS has stood the test of time, but then again each one of its steps is invested by mathematical details which make an efficient implementation nontrivial. Think of the basis matrix factorization maintenance, partial pricing strategies or primal-dual exploits.

One of the most critical phenomena encountered in practice is degeneracy. The latter introduces stalling in the resolution process which may even lead to cycling. An important survey on anti-cycling schemes and pivot-selection rules can be found in Terlaky and Zhang (1993) where a very large number of these are examined. In the words of Dantzig and Thapa (2003, p. 167), *whether any degeneracy avoiding scheme is needed in practice has never been settled. It has been observed, however, that even when there is no degeneracy, there is a high probability of near-degeneracy. This suggests that pivot-selection criteria should be designed to seek feasible solutions in directions away from degenerate and near-degenerate basic feasible solutions, or better yet, driven by dual feasibility considerations.*

The whole paragraph is the perfect embodiment of a rhetorical question. Although there is a lot of evidence that suggests accounting for degeneracy one way or another can be profitable, in no way does it positively asserts the need for such schemes. We therefore acknowledge these observations and address their final remark whereby pivot-selection rules guided by dual information seem favorable.

Of course, at the end of the day, all known primal algorithms base their stopping criteria on the optimality conditions provided by dual variables. Since no two algorithms share the same iterative process, there must be different levels of exposure that can be harvested from the realm of dual feasibility. Klotz (1988) extensively studies oracle based pricing methods for linear programs in an effort to side step degeneracy. The author is keen to observe that a trade-off between the number of iterations and the computational cost of the pricing requires significant additional work even when comparing to Dantzig's pivot-selection rule. As unfortunate as this may seem, one of the most forthcoming idea is that of *variable screening*. By discarding variables from the pricing problem, one hopes to dramatically reduce the difficulty of solving it. While many of the proposed screening rules try to get a handle on the problem structure and transfer the information to the pricing problem, we present a design

for the latter that inherently and dynamically exploits the problem structure. Such concerns also already appeared within column generation, the extension of PS for linear programs with a large number of variables, see Lübbecke and Desrosiers (2005). Degeneracy in column generation has been dealt with using certain dual variable stabilization approaches, see for example du Merle et al. (1999) and Ben Amor et al. (2009) for a stabilized column generation framework and the many references therein.

In this paper, we propose an algorithmic framework which, given a feasible basic solution, *fixes the values of a subset of dual variables* and optimizes the remaining ones for maximizing the minimum reduced cost (until one reaches an optimal solution). It turns out that the dual formulation of this pricing problem selects a convex combination of variables entering the basis. The way to divide these two subsets of dual variables relies on the choice of a vector subspace basis and opens a wide spectrum of possibilities thus paving the way for the paper at hand. Since each dual variable is associated with a constraint, this division also amounts to a partition of the rows.

Gauthier et al. (2016) propose a generic *contraction algorithm* for network flow problems whose elaboration is based on constructive arguments. This generic algorithm can be seen as a framework for primal network algorithms. At one extreme, the most known special case, Dantzig's pivot-selection rule for PS, considers all dual variables fixed and selects a single entering variable, a myopic strategy commonly resulting in degenerate pivots. At the other extreme, when none of the dual variables are fixed, we stumble upon the minimum mean cycle-canceling algorithm (MMCC) devised and shown to be strongly polynomial by Goldberg and Tarjan (1989). A contraction based on the set of free arcs (i.e., those for which the associated flow variables are strictly between the bounds) is also presented without an algorithm correspondence mention. This paper addresses this omission by associating this specialization with the improved primal algorithm (IPS) (Elhallaoui et al. 2011) and generalizes that framework to linear programs.

Introduced in Gauthier et al. (2015c), we base this paper on the vector space decomposition useful to close the theoretical gap between IPS and the dynamic constraint aggregation method (DCA) of Elhallaoui et al. (2005, 2008). The latter is specifically designed to overcome degeneracy in the context of solving the linear relaxation of set partitioning models by column generation. Although DCA is a precursor of IPS, its methodology slightly differs in that the row partition is deduced using a heuristic design. It works well because the row partition is in line with the purpose of set partitioning problems. By recovering notations from Gauthier et al. (2016), the proposed vector space decomposition creates network flow analogies to support the framework. The latter still extends from PS to MMCC while notably

including IPS and DCA in the process. With its generalization, simplified notation, and structure, this paper replaces an earlier version of this research only available as a working paper (Gauthier et al. 2015a).

The paper is organized as follows. Section 6.2 takes a close look at the essential components of the framework. Several concepts that partake (or not) in the resolution process of a linear program are examined such as nondegerate pivots, directions and the so-called *residual problem*. Each of these is presented in a separate manner whereas the last subsection ties everything together. Section 6.3 builds upon these ties and gives birth to the generic algorithm. In Section 6.4, we demonstrate a few properties, determine a family of algorithms with nondegenerate pivots at every iteration, and show that some directions are interior rather than along edges. Moreover, we examine well known special cases. Indeed, by using vector space decomposition, the paper at hand unifies within the same generic algorithm a variety of specialized ones for linear and network programs. We conclude in Section 6.5 with our contribution and some research perspectives.

6.2 The problem

Consider the linear program (LP) with lower and upper bounded variables:

$$\begin{aligned} z^\star := \min \quad & \mathbf{c}^\intercal \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \quad [\boldsymbol{\pi}] \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned} \tag{6.104}$$

where $\mathbf{x}, \mathbf{c}, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $m < n$. We assume that the matrix $\mathbf{A}$ is of full row rank and that LP (6.104) is feasible such that z^* is obviously finite. The vector of dual variables $\boldsymbol{\pi} \in \mathbb{R}^m$ associated with the equality constraints appears within brackets on the right-hand side.

Notation. Vectors and matrices are written in bold face. We denote by $\mathbf{I}_r$ the $r \times r$ identity matrix and by $\mathbf{0}$ (resp. $\mathbf{1}$) a vector/matrix with all zeros (resp. ones) entries of contextually appropriate dimension. For an ordered subset $R \subseteq \{1, \ldots, m\}$ of row indices and an ordered subset $P \subseteq \{1, \ldots, n\}$ of column indices, we denote by $\mathbf{A}_{RP}$ the sub-matrix of $\mathbf{A}$ containing the rows and columns indexed by R and P, respectively. We further use standard linear programming notation like $\mathbf{A}_B\mathbf{x}_B$, the subset of basic columns of $\mathbf{A}$ indexed by B multiplied by the corresponding vector of basic variables $\mathbf{x}_B$. The set of nonbasic columns N is used

analogously. The lower case notation is reserved for vectors and uses the same subset index rules. In particular, the matrix $\mathbf{A} := [\mathbf{a}_j]_{j\in\{1,\dots,n\}}$ contains n column vectors.

In Section 6.2.1, we formulate the so-called *residual problem* which allows the construction of an oracle generating feasible directions in Section 6.2.2. The latter also provides two alternative primal and dual conditions characterizing optimality for linear programs. Finally, let us embark upon this generic algorithm in Section 6.2.3 by analyzing a linear transformation, the goal being to structure the technological constraints.

6.2.1 The residual problem

It is common practice in developing network flow algorithms to use a *residual network* to improve upon some intermediate solution by identifying incremental flows, see Ahuja et al. (1993). In this paper, we do the same with linear programs and recover primal-dual optimality conditions on the residual problem.

We define the *residual problem* $LP(\mathbf{x}^k)$ with respect to a given solution $\mathbf{x}^k$ at iteration $k \geq 0$ as follows. Each variable $x_j,\ j \in \{1,\dots,n\}$, in the original LP (6.104) is replaced by two directed variables: the forward variable y_j of cost $d_j := c_j$ represents the possible increase $r_j^k := u_j - x_j^k$ of x_j relatively to x_j^k while the backward variable y_{j+n} of cost $d_{j+n} := -c_j$ represents its possible decrease $r_{j+n}^k := x^k - \ell_j$; moreover, only one can be used with a positive value, i.e., the complementarity condition $y_j\, y_{j+n} = 0$ holds, $\forall j \in \{1,\dots,n\}$, see Figure 6.37.

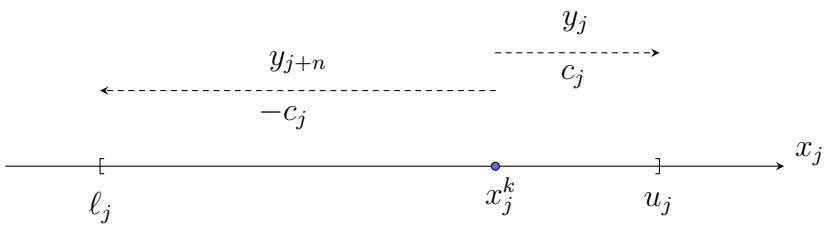


Fig. 6.37: Forward and backward variables for the residual problem

Equivalent to LP (6.104), a formulation for the residual problem $LP(\mathbf{x}^k)$ is given below:

$$\begin{aligned} z^\star = z^k \ + \quad \min \quad & \mathbf{d}^\intercal \mathbf{y} \\ \text{s.t.} \quad & \mathbf{K}\mathbf{y} = \mathbf{0} \quad [\boldsymbol{\pi}] \\ & \mathbf{0} \leq \mathbf{y} \leq \mathbf{r}^k, \end{aligned} \tag{6.105}$$

where $z^k := \mathbf{c}^\intercal \mathbf{x}^k$, $\mathbf{d} := [d_j]_{j\in\{1,\dots,2n\}}$ is the cost vector, $\mathbf{y} := [y_j]_{j\in\{1,\dots,2n\}} \in \mathbb{R}^{2n}_+$ contains the forward and backward variables, their residual upper bounds are given by $\mathbf{r}^k := [r^k_j]_{j\in\{1,\dots,2n\}}$, and the matrix $\mathbf{K} := [\mathbf{A}, -\mathbf{A}] \equiv [\mathbf{k}]_{j\in\{1,\dots,2n\}}$ stands to remind us that the kernel (or nullspace) of this matrix is the set of solutions to $\mathbf{Ky} = \mathbf{0}$. Since a variable bounded under and above by 0 is useless, the residual problem $LP(\mathbf{x}^k)$ may be written using only *residual* variables, that is, y-variables with positive residual upper bounds within the set $J^k := \{j \in \{1,\dots,2n\} \mid r^k_j > 0\}$.

Network flow analogies. Let us propose some linear programming vocabulary in the spirit of network flows. By neglecting the residual upper bounds from (6.105), one obtains a cone of extreme rays.

Definition 10. *An extreme ray* $\mathbf{y} \in \{\mathbf{Ky} = \mathbf{0}, \mathbf{y} \geq \mathbf{0}\}$ *induces an* n*-dimensional* direction $\vec{\mathbf{v}} := [\vec{v}_j]_{j\in\{1,\dots,n\}}$, *not necessarily feasible for LP* (6.104)*, by computing the difference*

$$\vec{v}_j = y_j - y_{j+n}, \quad \forall j \in \{1,\dots,n\}. \tag{6.106}$$

In order to find an improving direction, it suffices to look within this set of directions using any comparative measure. Then again, since optimizing in a cone proves to be delicate, let us enlist a normalization constraint $\mathbf{w}^\intercal \mathbf{y} = 1$, where $\mathbf{w} > \mathbf{0}$ is a vector of arbitrary positive weights associated with the y-variables. This results in a cut cone where every nonnull extreme point corresponds to an extreme ray and thus captures any scaled direction. This correspondence is clearly visible on Figure 6.38. Imagine it represents the cone of extreme rays at $\mathbf{x}^k$ as expressed in Definition 10 and then consider the normalization constraint with weights of one all around. Furthermore, by considering a different weight vector $\mathbf{w}$, the cutting plane of the cone would be slanted differently thus resulting in modified extreme points. Nevertheless, each of these extreme point would remain associated with the same extreme ray. Definition 11 introduces a normalized extreme ray which can then be manipulated with any scalar.

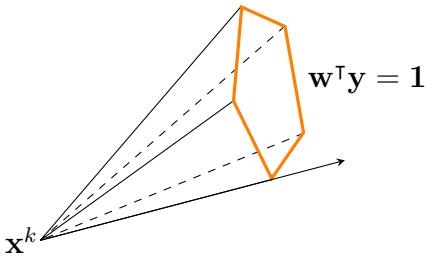


Fig. 6.38: At $\mathbf{x}^k$, the cone $\{\mathbf{y} \geq \mathbf{0} \mid \mathbf{Ky} = \mathbf{0}\}$ cut by $\mathbf{w}^\intercal\mathbf{y} = \mathbf{1}$

Definition 11. *Let a* weighted cycle *be a normalized extreme ray* $\mathbf{y}$ *satisfying*

$$\mathbf{y} \in \mathcal{N}^k := \{\mathbf{K}\mathbf{y} = \mathbf{0}, \mathbf{y} \geq \mathbf{0}, \mathbf{w}^\intercal \mathbf{y} = 1\} \subset \mathbb{R}^{2n}_+. \tag{6.107}$$

Observe that the y-values cannot be rid of even if $\mathbf{w} = \mathbf{1}$ as they are inherently dependent on the structure of the technological constraints. Nevertheless, since these values are unique for a given weighted cycle, it is often simpler to work solely with the variable support denoted by $W := \{j \in \{1, \ldots, 2n\} \mid y_j > 0, \mathbf{y} \in \mathcal{N}^k\}$. The cost of a weighted cycle W is then computed as $d_W := \sum_{j \in W} d_j y_j$.

Definition 12. *A weighted cycle* W *is* directed *if and only if all its composing* y*-variables can increase from zero in the suggested direction, i.e.,*

$$r^k_j > 0, \ \forall j \in W \quad \textit{or equivalently} \quad W \subseteq J^k. \tag{6.108}$$

Definition 13. *A* negative weighted cycle *is a directed weighted cycle with a negative cost.*

Necessary and sufficient optimality conditions. Under the above nomenclature, necessary and sufficient optimality conditions come together in a straightforward manner. The reduced cost of variable $x_j, \ j \in \{1, \ldots, n\}$ is defined as $\bar{c}_j := c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j$ while those of variables y_j and y_{j+n} are respectively

$$\bar{d}_j := c_j - \boldsymbol{\pi}^\intercal \mathbf{a}_j = \bar{c}_j \quad \text{and} \quad \bar{d}_{j+n} := -c_j - \boldsymbol{\pi}^\intercal(-\mathbf{a}_j) = -\bar{c}_j. \tag{6.109}$$

In addition to the complementary slackness optimality conditions on LP (6.104) based on the reduced cost of the original x-variables (see Schrijver 1986), we provide two alternative conditions characterizing optimality for linear programs.

Proposition 33. (Gauthier et al. 2014, Theorem 4) *A feasible solution* $\mathbf{x}^k$ *to* LP (6.104) *is optimal if and only if the following equivalent conditions are satisfied:*

Complementary slackness: $\exists \boldsymbol{\pi}$ *such that*

$$\bar{c}_j > 0 \Rightarrow x^k_j = \ell_j; \qquad \bar{c}_j < 0 \Rightarrow x^k_j = u_j; \qquad \ell_j < x^k_j < u_j \Rightarrow \bar{c}_j = 0. \tag{6.110}$$

Primal: $LP(\mathbf{x}^k)$ *contains no negative weighted cycle, that is,*

$$d_W \geq 0, \quad \forall W \subseteq J^k. \tag{6.111}$$

Dual: $\exists \boldsymbol{\pi}$ *such that the reduced cost of every residual variable of* $LP(\mathbf{x}^k)$ *is nonnegative, that is,*

$$\bar{d}_j \geq 0, \quad \forall j \in J^k. \tag{6.112}$$

Proof. Assuming the reader accepts complementary slackness, the equivalence with the dual conditions is trivial. Depending on the value of x_j^k, the residual variables are either y_j (when $x_j^k = \ell_j$), y_{j+n} (when $x_j^k = u_j$) or both y_j and y_{j+n} (when $\ell_j < x_j^k < u_j$). Recall the signed reduced costs of directed variables y_j and y_{j+n} in (6.109) such that all residual variables have a nonnegative reduced cost if and only if complementary slackness is met.

With respect to the primal condition claim, analogously to the cost and reduced cost of a network cycle being equal, the cost d_W and reduced cost $\bar{d}_W$ of a weighted cycle W are also the same. Indeed, the latter satisfies (6.107), hence $\sum_{j \in W} \mathbf{k}_j y_j = \mathbf{0}$ and

$$\bar{d}_W = \sum_{j \in W} (d_j - \boldsymbol{\pi}^\intercal \mathbf{k}_j) y_j = d_W - \boldsymbol{\pi}^\intercal \sum_{j \in W} \mathbf{k}_j y_j = d_W. \tag{6.113}$$

When all residual variables have nonnegative reduced costs, then $d_W = \bar{d}_W \geq 0$, $\forall W \subseteq J^k$ and a negative weighted cycle cannot exist. □

6.2.2 An oracle

In order to prove the optimality of $\mathbf{x}^k$, one can derive an oracle relying on the identification of weighted cycles. It is derived automatically from the domain (6.107) and an objective function which effectively computes the cost (or the reduced cost) of each weighted cycle properly as follows

$$\min_{\mathbf{y} \in \mathcal{N}^k} \ \mathbf{d}^\intercal \mathbf{y}. \tag{6.114}$$

By definition of a negative weighted cycle (Definition 13), the oracle (6.114) honors the necessary portion of the conditions if and only if it uses only residual y-variables, i.e., the y-variables with null residual capacity are explicitly discarded before removing the upper bounds. Keeping track of residual variables can also be done by partitioning the x-variables according to their values. In order to achieve this, let $\mathbf{x}^k$ be represented by $(\mathbf{x}_F^k, \mathbf{x}_L^k, \mathbf{x}_U^k)$, where the three sub-vectors respectively refer to the set of free variables $F := \{j \in \{1, \ldots, n\} \mid \ell_j < x_j^k < u_j\}$, at their lower bounds $L := \{j \in \{1, \ldots, n\} \mid x_j^k = \ell_j\}$, and at their upper bounds $U := \{j \in \{1, \ldots, n\} \mid x_j^k = u_j\}$. Let there be $f := |F|$ such free variables, $0 \leq f \leq n$. Observe that if $\mathbf{x}^k$ is basic then $0 \leq f \leq m$. Controlling the presence

of residual variables can then alternatively be achieved by imposing

$$y_j = 0,\ \forall j \in U \quad \text{and} \quad y_{j+n} = 0,\ \forall j \in L. \tag{6.115}$$

It hardly takes any convincing to accept that it is possible to improve intermediate solutions using negative weighted cycles until an optimal solution is reached. In this respect, the step size ρ associated with the negative weighted cycle W must satisfy $\rho\, y_j \leq r_j^k,\ \forall j \in W$ and this cycle is *canceled* when the step size is equal to

$$\rho := \min_{j \in W} \frac{r_j^k}{y_j} > 0. \tag{6.116}$$

Primal simplex algorithm. Consider a basic solution $\mathbf{x}^0$ and the index set of basic variables B within the primal simplex algorithm. A pivot operation tries to improve the current solution using a nonbasic entering variable, say $x_\ell,\ \ell \in N$. The aftermath of this operation is simplified to a properly selected exiting variable and the associated step size ρ is determined by the ratio-test. The ratio-test is useful on two counts. It maximizes the exchange potential of the entering variable and it maintains a basic solution for $\mathbf{x}^1$. The mechanic is incredibly simple although it might sometimes render the linear algebra aspect of the pivot nebulous, especially in the context of degeneracy. In this respect, when PS performs a nondegenerate pivot at iteration $k \geq 0$, it goes from vertex $\mathbf{x}^k$ represented by a nonoptimal basis to vertex $\mathbf{x}^{k+1}$ by moving along an edge (Dantzig and Thapa 2003, Theorem 1.7), a direct consequence of the entering/exiting variable mechanism. In the case of a degenerate pivot, the basis is modified, but the geometrical solution vertex remains the same. In other words, the n-dimensional direction (see Definition 10)

$$\vec{v}_j = \begin{cases} y_j - y_{j+n}, & \forall j \in B \cup \{\ell\} \\ 0, & \forall j \in N \setminus \{\ell\} \end{cases} \tag{6.117}$$

induced by the selected negative reduced cost entering variable x_ℓ leads outside the domain of LP (6.104) and we do not move. One may want to consider the column $\mathbf{a}_\ell$ of the entering variable as part of the linear span of $\mathbf{A}_B$, that is, $\mathbf{V}(\mathbf{A}_B) = \mathbb{R}^m$. By definition, any m-dimensional column belongs to $\mathbf{V}(\mathbf{A}_B)$ meaning in particular that for any nonbasic entering variable

$$\exists!\boldsymbol{\lambda} \in \mathbb{R}^m \text{ such that } \sum_{j\in B} \mathbf{a}_j \lambda_j = \mathbf{a}_\ell \text{ which works out to } \boldsymbol{\lambda} = \mathbf{A}_B^{-1}\mathbf{a}_\ell. \tag{6.118}$$

It is settled then, the simplex pivot follows a direction which is certainly not the sole entering variable, nor is it limited to the entering/exiting variable couple. By thinking of the simplex pivot as a direction (and the associated weighted cycle), we fall back upon the oracle seen in the residual problem which provides a way to identify these, and many others, in a general framework. Since the linear combination scalars $\boldsymbol{\lambda}$ can take any sign, every column of $\mathbf{A}_B$ is implicitly expected to have freedom to move in either direction. This *could* unfortunately be proven false when the pivot exercise arrives. This possibility can only arise when a nonbasic variable is defined by a linear combination containing at least one degenerate variable, that is, a basic variable at one of its bounds. Indeed, the associated weighted cycle to such an entering variable *might* include a y_j-variable, $j \in B$ with a residual upper bound of 0, i.e., a forward variable $y_j > 0 \mid x_j^k = u_j$ or a backward variable $y_{j+n} > 0 \mid x_j^k = \ell_j$.

The reader may want to compare this with (6.115) to realize that the degeneracy phenomenon takes an equivalent form in the oracle as well. As degenerate pivots can only occur when the current solution is degenerate, the following section derives a so-called *transformation matrix* which capitalizes on the over representation notion attached to degenerate solutions.

6.2.3 Linear algebra

Recall that the presence of degeneracy in a basic solution $\mathbf{x}^k$ to LP (6.104) is detected when only a strict subset F of the basic variables are free, say $\emptyset \subseteq F \subset B$. Applying the inverse of an arbitrary nonsingular matrix $\mathbf{T}$ on the equality constraints of LP yields an equivalent system, i.e., $\mathbf{T}^{-1}\mathbf{A}\mathbf{x} = \mathbf{T}^{-1}\mathbf{b} \Leftrightarrow \mathbf{A}\mathbf{x} = \mathbf{b}$. The goal of the following linear transformation $\mathbf{T}_P^{-1}$ is to structure the technological constraints.

Consider a set P of *observed* or *assumed* free variables forming a *subspace basis* $\mathbf{A}_P$ with dimension $p := |P|$. In that case, a subset of p rows within $\mathbf{A}_P$ are independent. There must then exist a row partition of $\mathbf{A}_P$ such that $\mathbf{A}_{RP}$ is a nonsingular matrix of size $p \times p$. For instance, an optimal basic solution to the following *restricted* phase I problem (see Gauthier et al. 2015c)

$$\min\{\mathbf{1}^\intercal\boldsymbol{\theta} \mid \mathbf{A}_P\mathbf{x}_P + \mathbf{I}_m\boldsymbol{\theta} = \mathbf{b}, \boldsymbol{\theta} \geq \mathbf{0}\} \tag{6.119}$$

identifies a set S of rows by associating $(m-p)$ $\boldsymbol{\theta}$-variables with the used rows of the identity matrix yielding the simplex basis $\begin{bmatrix} \mathbf{A}_{RP} & \mathbf{0} \\ \mathbf{A}_{SP} & \mathbf{I}_{m-p} \end{bmatrix}$, hence the row partition $\begin{bmatrix} \mathbf{A}_{RP} \\ \mathbf{A}_{SP} \end{bmatrix}$ of $\mathbf{A}_P$.

By definition of a subspace basis, this can be equivalently expressed by $\left[\begin{smallmatrix}\mathbf{I}_p\\ \mathbf{M}\end{smallmatrix}\right]$, where $\mathbf{M} := \mathbf{A}_{SP}\mathbf{A}_{RP}^{-1}$. Since *completing the basis* can be done arbitrarily using artificial variables, one can in general let $\mathbf{T}_P$ and $\mathbf{T}_P^{-1}$ be given by

$$\mathbf{T}_P = \begin{bmatrix}\mathbf{I}_p & \mathbf{0}\\ \mathbf{M} & \mathbf{I}_{m-p}\end{bmatrix} \qquad \text{and} \qquad \mathbf{T}_P^{-1} = \begin{bmatrix}\mathbf{I}_p & \mathbf{0}\\ -\mathbf{M} & \mathbf{I}_{m-p}\end{bmatrix}, \tag{6.120}$$

such that applying the linear transformation on the system $\mathbf{Ax} = \mathbf{b}$ results in $\bar{\mathbf{A}} := \mathbf{T}_P^{-1}\mathbf{A}$ and $\bar{\mathbf{b}} := \mathbf{T}_P^{-1}\mathbf{b}$ as follows:

$$\bar{\mathbf{A}} = \begin{bmatrix}\mathbf{A}_R\\ \mathbf{A}_S - \mathbf{M}\mathbf{A}_R\end{bmatrix} \qquad \text{and} \qquad \bar{\mathbf{b}} = \begin{bmatrix}\mathbf{b}_R\\ \mathbf{b}_S - \mathbf{M}\mathbf{b}_R\end{bmatrix}. \tag{6.121}$$

The point of the transformation all comes together in the following definition by reminiscing on the outcome of Gauss-Jordan elimination on linear dependent systems.

Definition 14. (Gauthier et al. 2015c, Proposition 3) *A vector* $\mathbf{a} \in \mathbb{R}^m$ *(and the associated variable, if any) is* compatible with $\mathbf{A}_P$ *if and only if* $\bar{\mathbf{a}}_S := \mathbf{a}_S - \mathbf{M}\mathbf{a}_R = \mathbf{0}$ *or, equivalently, it belongs to* $\mathbf{V}(\mathbf{A}_P)$*, the linear span of the vector subspace basis* $\mathbf{A}_P$*.*

Recall that $\mathbf{V}(\mathbf{A}_P) := \{\mathbf{A}_P\boldsymbol{\lambda} \mid \boldsymbol{\lambda} \in \mathbb{R}^p\}$ such that verifying the equivalence is straightforward when decomposing $\left[\begin{smallmatrix}\mathbf{a}_R\\ \mathbf{a}_S\end{smallmatrix}\right] = \left[\begin{smallmatrix}\mathbf{A}_{RP}\\ \mathbf{A}_{SP}\end{smallmatrix}\right]\boldsymbol{\lambda}$. The compatibility status of a column vector can therefore be determined using any and all methods available from the linear algebra arsenal. Some are more efficient than others depending on the content of matrix $\mathbf{A}$, the most probing known cases being the network and set partitioning problems which easily permit the verification of the definition, see the *transformation matrix insight* paragraph at the end of this section.

In fact, let $Q := \{1, \ldots, n\} \setminus P$ contain all the variables outside the set P. This column partition is represented by the matrix $\mathbf{A} = \begin{bmatrix}\mathbf{A}_P, \mathbf{A}_Q\end{bmatrix}$. Altogether, we have the row/column partition

$$\mathbf{A} = \begin{bmatrix}\mathbf{A}_{RP} & \mathbf{A}_{RQ}\\ \mathbf{A}_{SP} & \mathbf{A}_{SQ}\end{bmatrix},$$

where the nonsingular $p \times p$ matrix $\mathbf{A}_{RP}$ is called the *working basis*. Applying $\mathbf{T}_P^{-1}$ on $\mathbf{A}$ yields

$$\bar{\mathbf{A}} = \begin{bmatrix}\mathbf{I}_p & \mathbf{0}\\ -\mathbf{M} & \mathbf{I}_{m-p}\end{bmatrix}\begin{bmatrix}\mathbf{A}_{RP} & \mathbf{A}_{RQ}\\ \mathbf{A}_{SP} & \mathbf{A}_{SQ}\end{bmatrix} = \begin{bmatrix}\mathbf{A}_{RP} & \mathbf{A}_{RQ}\\ \mathbf{0} & \bar{\mathbf{A}}_{SQ}\end{bmatrix}. \tag{6.122}$$

In primal simplex terms, if one thinks of $\mathbf{T}_P$ as the current basis, the ratio-test of an entering variable x_ℓ with null entries in $\bar{\mathbf{a}}_{S\ell}$ would be performed only on the positive coefficients of $\bar{\mathbf{a}}_{R\ell}$ and thus only influence variables related to $\mathbf{A}_P$. This means that all variables associated with $\mathbf{A}_P$ and the row set R are *assumed* to be free whereas all variables associated with $\left[\begin{smallmatrix}\mathbf{0}\\ \mathbf{I}_{m-p}\end{smallmatrix}\right]$ and the row set S are *assumed* to be at their bounds. If P indeed corresponds to free variables only ($\mathbf{A}_P = \mathbf{A}_F$), the resulting step size would be positive for sure. In this spirit, the purpose of $\mathbf{T}_P^{-1}$ is to induce a partition in $\bar{\mathbf{A}}$ to help look for so-called compatible column vectors.

For the record, compatibility can also be determined using *Positive Edge* (PE). The latter is an alternative rule described in Raymond et al. (2010a), Towhidi et al. (2014), and Gauthier et al. (2015c) which uses a stochastic argument to reduce the matrix multiplication computational penalty. The rule uses a random vector $\boldsymbol{\alpha} \neq \mathbf{0} \in \mathbb{R}^{m-p}$ and considers a column vector $\mathbf{a}$ compatible with $\mathbf{A}_P$ if $\boldsymbol{\alpha}^\intercal[-\mathbf{M} \quad \mathbf{I}_{m-p}]\mathbf{a} = \boldsymbol{\alpha}^\intercal\mathbf{T}_S^{-1}\mathbf{a} = \boldsymbol{\beta}^\intercal\mathbf{a} = 0$, where the *premultiplication* of vector $\boldsymbol{\beta}^\intercal = \boldsymbol{\alpha}^\intercal\mathbf{T}_S^{-1} \in \mathbb{R}^m$ procures the computational savings. Testing for compatibility over basis $\mathbf{A}_B$ can be done in $O(m^2)$.

Transformation matrix insight. Ultimately, the transformation matrix produces row and column partitions intimately binded together. Depending on the application, the row partition can even be obtained in the midst of selecting the set P by trying to capture the linear dependence of the technological constraints. Network flow and set partitioning problems are such applications, see Figures 6.39 and 6.40 respectively.

In network flows, the free arcs forming $\mathbf{A}_F$ can be visually separated in trees forming a forest. The latter is expressed in matrix form in Figure 6.39a. One can then associate a root node to each tree. Each of these root nodes corresponds to a linear dependent row in $\mathbf{A}_F$ thus forming the row partition presented in Figure 6.39b. A constructive approach leading to a contracted network for the identification of negative cycles is presented in Gauthier et al. (2016).

In set partitioning problems, and more specifically when using DCA, the subspace basis $\left[\begin{smallmatrix}\mathbf{I}_p\\ \mathbf{M}\end{smallmatrix}\right]$ is obtained on-the-fly while *heuristically* trying to establish linear independent rows within $\mathbf{A}_F$. This process is sketched in Figure 6.40. In Figure 6.40a, the original matrix $\mathbf{A}_F$ is presented whereas Figure 6.40b reorganizes the duplicated rows on the bottom (and this reorganization then applies to the original system). By associating a unique identifier to each singled out row in the top portion and replicating these identifiers in the bottom portion, Figure 6.40b obtains five rows in the set R and three in the set S. Figure 6.40c uses these identifiers by replacing the matrix content with trivial unity references for each identifier,

	x_{54}	x_{56}	x_{16}	x_{78}	x_{98}	$x_{8,10}$	x_{72}
5	1	1					
4	-1						
6		-1	-1				
1			1				
7				1			1
8				-1	-1	1	
9					1		
10						-1	
2							-1
3							

(a) Matrix form of the forest induced by $\mathbf{A}_F$

		x_{54}	x_{56}	x_{16}	x_{78}	x_{98}	$x_{8,10}$	x_{72}
	5	1	1					
	4	-1						
	6		-1	-1				
R	7				1			1
	8				-1	-1	1	
	9					1		
	10						-1	
	1			1				
S	2							-1
	3							

(b) Row partition $\{R, S\}$ of $\mathbf{A}_F$

Fig. 6.39: Network flow problem row partition

thus obtaining the subspace basis $\left[\begin{smallmatrix}\mathbf{I}_5\\ \mathbf{M}\end{smallmatrix}\right]$. One can easily verify that the four columns of $\mathbf{A}_F$ in Figure 6.40b belong to the span of that subspace basis. This is also true for another column from the simplex basis $\mathbf{A}_B$, its actual content being irrelevant.

$$\begin{pmatrix} 1 & 1 & & \\ & 1 & 1 & \\ 1 & 1 & & \\ & & 1 & 1 \\ 1 & & 1 & \\ 1 & & & 1 \\ & 1 & 1 & \\ 1 & & & 1 \end{pmatrix}$$

(a) Original matrix $\mathbf{A}_F$

$$\left(\begin{array}{cccc} 1 & 1 & & \\ & 1 & 1 & \\ & & 1 & 1 \\ 1 & & 1 & \\ 1 & & & 1 \\ \hline 1 & 1 & & \\ & 1 & 1 & \\ 1 & & & 1 \end{array}\right) \begin{matrix} a \\ b \\ c \\ d \\ e \\ \hline a \\ b \\ e \end{matrix}$$

(b) Row partition $\left[\begin{smallmatrix}\mathbf{A}_{RF}\\ \mathbf{A}_{SF}\end{smallmatrix}\right]$

$$\left(\begin{array}{cccc} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \\ \hline 1 & & & \\ & 1 & & \\ & & & 1 \end{array}\right)$$

(c) Subspace basis $\left[\begin{smallmatrix}\mathbf{I}_5\\ \mathbf{M}\end{smallmatrix}\right]$

Fig. 6.40: Set partitioning problem row partition

6.3 Vector space decomposition framework

In this section, we look at the essential components of the proposed framework. The algorithm relies on an oracle to iterate. The latter is dynamically updated with respect to the values of the current solution $\mathbf{x}^k$. As such, we already stated the residual problem which gives a great deal of insight to the oracle's solutions. We also define a row/column partition, based on the transformation matrix, that generates the content of the oracle. In a nutshell, the portions obtained from this partition communicate with each other in the same way a master/subproblem would. In practice, we capitalize on the partition by treating its content like a Dantzig-Wolfe decomposition, see Dantzig and Wolfe (1960).

The generic algorithm is broken down into eight main steps aside from the initialization. Figure 6.41 provides an overview of these steps, while the following subsections detail their content.

Initialization : Iteration $k := 0$;
Feasible solution $\mathbf{x}^0$;
Select the subspace basis $\mathbf{A}_P$;
1 Derive the transformation matrix $\mathbf{T}_P^{-1}$ and the row/column partitions $\{R, S\}/\{P, Q\}$;
2 Solve the pricing problem (6.129) for μ_V^k, W_V^k;
3 If $\mu_V^k \geq 0$, **terminate** with an optimality certificate for $\mathbf{x}^k$;
4 Extract the extended weighted cycle $W_{V\cup H}^k$ from (6.130)-(6.131) and the direction $\vec{\mathbf{v}}^k$;
5 Compute the step size ρ^k (6.132);
6 Update to solution $\mathbf{x}^{k+1} := \mathbf{x}^k + \rho^k \vec{\mathbf{v}}^k$ using the system (6.133);
7 Update the subspace basis $\mathbf{A}_P$;
8 Iterate $k := k + 1$;

Fig. 6.41: Generic vector space decomposition algorithm for linear programs

Initialization. It all starts at iteration counter $k = 0$ with some basic feasible solution $\mathbf{x}^0$ and column partition $\{F, L, U\}$. The construction of the residual problem $LP(\mathbf{x}^k)$ (6.105) calls for a change of variables: $y_j - y_{j+n} := x_j - x_j^k,\ y_j\, y_{j+n} = 0,\ \forall j \in \{1, \ldots, n\}$. These directed variables are bounded by $0 \leq y_j \leq r_j^k,\ \forall j \in \{1, \ldots, 2n\}$.

6.3.1 The structured residual problem

Once an arbitrary subspace basis $\mathbf{A}_P$ is selected, the induced transformation matrix $\mathbf{T}_P^{-1}$ is derived along with a row/column partition $\{R, S\}$ and $\{P, Q\}$. The only point we shall insist on is the structuring effect of the transformation matrix. The same structure can obviously be observed in the residual problem. Let us divide the y-variables according to the column partition $\{P, Q\}$ with the sets

$$H_P \ \equiv\ H_P(\mathbf{x}^k) := \bigcup_{j\in P} \{j, j+n\} \tag{6.123}$$

$$V_P \ \equiv\ V_P(\mathbf{x}^k) := J^k \setminus H_P\,, \tag{6.124}$$

where the residual y-variables are obviously exhaustively considered. However, observe that while V_P contains only residual variables, H_P may or may not. That is, if there exists a $j \in P$ such that either $j \in L$ or $j \in U$, the variable y_{j+n} or respectively y_j has a null residual

capacity. The residual problem formulation (6.105) can then be rewritten as

$$\begin{aligned} z^\star = z^k \; + \quad \min \quad & \sum_{j\in H_P} d_j y_j \; + \; \sum_{j\in V_P} d_j y_j & & \\ \text{s.t.} \quad & \sum_{j\in H_P} \mathbf{k}_{Rj} y_j \; + \; \sum_{j\in V_P} \mathbf{k}_{Rj} y_j & = \; \mathbf{0} \quad & [\boldsymbol{\psi}_R] \\ & \sum_{j\in V_P} \bar{\mathbf{k}}_{Sj} y_j & = \; \mathbf{0} \quad & [\boldsymbol{\psi}_S] \\ & 0 \le y_j \le r_j^k,\ \forall j \in H_P, \qquad 0 \le y_j \le r_j^k,\ \forall j \in V_P, & & \end{aligned} \tag{6.125}$$

where $\mathbf{k}_j$ is the j-th column vector of $\mathbf{K}$, $\bar{\mathbf{K}} := \mathbf{T}_P^{-1}\mathbf{K}$, and $\boldsymbol{\psi}^\intercal = [\boldsymbol{\psi}_R^\intercal,\ \boldsymbol{\psi}_S^\intercal]$ is the vector of dual variables of the transformed system. The original dual vector $\boldsymbol{\pi}$ can be retrieved from $\boldsymbol{\psi}$ using the expression $\boldsymbol{\pi}^\intercal = \boldsymbol{\psi}^\intercal \mathbf{T}_P^{-1}$, that is,

$$[\boldsymbol{\pi}_R^\intercal,\ \boldsymbol{\pi}_S^\intercal] = [\boldsymbol{\psi}_R^\intercal - \boldsymbol{\psi}_S^\intercal \mathbf{M},\ \boldsymbol{\psi}_S^\intercal]. \tag{6.126}$$

6.3.2 The pricing problem

The pricing problem exploits the structure in (6.125) and derives an oracle based on the resulting transformation. The oracle is presented in both primal and dual forms, each having its own interpretation. Let us start with the dual form which is derived by trying to meet the necessary and sufficient optimality conditions. That is, if the assumed free variables in the set P are at an optimal value, the dual variables of $\boldsymbol{\pi}$, or those of $\boldsymbol{\psi}$, must impose a null reduced cost on these variables [complementary slackness conditions]:

$$\mathbf{0} = \bar{\mathbf{c}}_P^\intercal = \mathbf{c}_P^\intercal - \boldsymbol{\psi}_R^\intercal \mathbf{A}_{RP} \quad (= \mathbf{c}_P^\intercal - (\boldsymbol{\pi}_R^\intercal + \boldsymbol{\pi}_S^\intercal \mathbf{M})\mathbf{A}_{RP} = \mathbf{c}_P^\intercal - \boldsymbol{\pi}_R^\intercal \mathbf{A}_{RP} - \boldsymbol{\pi}_S^\intercal \mathbf{A}_{SP}). \tag{6.127}$$

This is equivalent to imposing $\bar{d}_j = d_j - \boldsymbol{\psi}_R^\intercal \mathbf{k}_{Rj} \ge 0,\ \forall j \in H_P$, that is, $\bar{d}_j = \bar{d}_{j+n} = 0,\ \forall j \in P$. Furthermore, if the current solution $\mathbf{x}^k$ is also optimal, there should exist a dual vector $\boldsymbol{\psi}_S$ such that the smallest reduced cost of the remaining variables in V_P, say μ_V, is nonnegative [dual condition]. Given the scalars $w_j > 0,\ \forall j \in V_P$, this verification can be done with the program

$$\begin{aligned} \max \ & \mu_V \\ \text{s.t.} \quad & \mu_V \le \frac{\bar{d}_j}{w_j} = \frac{1}{w_j}(d_j - \boldsymbol{\psi}_R^\intercal \mathbf{k}_{Rj} - \boldsymbol{\psi}_S^\intercal \bar{\mathbf{k}}_{Sj}) \quad [y_j] \quad \forall j \in V_P, \end{aligned} \tag{6.128}$$

where the vector $\boldsymbol{\psi}_R^\intercal = \mathbf{c}_P^\intercal \mathbf{A}_{RP}^{-1}$ is fixed by (6.127) whereas the vector $\boldsymbol{\psi}_S^\intercal$ is part of the optimization so as to maximize the minimum reduced cost.

Dualizing (6.128), we obtain the primal form of the oracle which comprises $m - p + 1$ constraints and writes as

$$\begin{aligned} \min \sum_{j \in V_P} \tilde{d}_j y_j & \\ \text{s.t.} \quad \sum_{j \in V_P} \bar{\mathbf{k}}_{Sj} y_j = \mathbf{0} & \qquad [\boldsymbol{\psi}_S] \\ \sum_{j \in V_P} w_j \, y_j = 1 & \qquad [\mu_V] \\ y_j \geq 0, \quad \forall j \in V_P. & \end{aligned} \tag{6.129}$$

where $\tilde{d}_j := d_j - \boldsymbol{\psi}_R^\intercal \mathbf{k}_{Rj} = d_j - \mathbf{c}_P^\intercal \mathbf{A}_{RP}^{-1} \mathbf{k}_{Rj}, \forall j \in V_P$. The oracle interpretation is done through the primal/dual pair (6.128)/(6.129). It brings together the negative weighted cycles and the transformation matrix $\mathbf{T}_P^{-1}$.

Oracle interpretation. First of all, the formulation (6.129) is always feasible unless $\mathbf{x}^0$ is the only feasible solution of LP (6.104) in which case the former is infeasible. Reciprocally, the formulation (6.128) is always feasible although unbounded in the exception case. Note that we can ensure that the primal/dual pricing system is feasible/bounded if the normalization constraint is written as a less-than-or-equal inequality or equivalently one imposes $\mu_V \leq 0$. Furthermore, weight values of $\mathbf{w} := [w_j]_{j \in \{1,\ldots,2n\}}$ used for the normalization constraint can be set in stone or updated dynamically. In the former case, think of the simple one vector typically used in network flows (see Gauthier et al. 2015b) or the norm based weights such as $w_j = w_{j+n} = \|\mathbf{a}_j\|, \ \forall j \in \{1, \ldots, n\}$, which makes the ratio $\bar{d}_j / w_j$ impervious to the scaling of variable x_j. In the latter case, dynamic weight choices can also be made to help steer the pricing problem towards or away from certain solutions, see (Rosat et al. 2016a,b) for several alternatives which have been particularly successful for solving set partitioning problems using the *integral simplex using decomposition* algorithm (Zaghrouti et al. 2014). Finally, it can also be noted that all other things being equal, a smaller value of w_j relatively to the others favors the selection of variable y_j in the pricing problem.

Let μ_V^k and $\mathbf{y}_V^k := [y_j^k]_{j \in V_P}$ denote an optimal solution to the primal/dual system at iteration $k \geq 0$. If $\mu_V^k \geq 0$, the dual optimality conditions are satisfied and the algorithm terminates with an optimal solution $\mathbf{x}^k$. Otherwise, $\mu_V^k < 0$ and the current solution might be *improved* by following a direction.

The dual form takes on a very simple form and maximizes the minimum reduced cost by pricing the listed y-variables. As explained in the next paragraphs, the primal form measures the minimum mean cost of the directed weighted cycles available using the y-variables in V_P.

Observe that an optimal weighted cycle, say W_V^k, derived from (6.129) is directed since only residual variables $y_j,\ j \in V_P$ are considered. The fact that $\mathbf{y}_V^k$ is built omitting variables $\mathbf{y}_H$ means that it only provides a *portion* of $\mathbf{y}^k$. However, by construction of the linear transformation, these missing components are uniquely determined within (6.125) in the system of row set R where the free nature of variables in P is assumed, that is,

$$\begin{array}{cc} \displaystyle\sum_{j\in H_P} \mathbf{k}_{Rj} y_j & + \quad \displaystyle\sum_{j\in V_P} \mathbf{k}_{Rj} y_j^k \quad = \mathbf{0} \\ 0 \le y_j \le r_j^k,\ \forall j \in H_P, & \qquad 0 \le y_j^k \le r_j^k,\ \forall j \in V_P. \end{array} \tag{6.130}$$

The solution $\mathbf{y}_H^k$ of this system is determined alike (6.118) by $\boldsymbol{\lambda}_P := \mathbf{A}_{RP}^{-1}\left(-\sum_{j\in V_P} \mathbf{k}_{Rj} y_j^k\right)$ as follows

$$y_j^k = \begin{cases} -\lambda_j, & \text{if } \lambda_j < 0 \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad y_{j+n}^k = \begin{cases} \lambda_j, & \text{if } \lambda_j > 0 \\ 0, & \text{otherwise} \end{cases} \quad \text{for all } j \in P\,, \tag{6.131}$$

and the direction $\vec{\mathbf{v}}^k$ follows from Definition 10. Observe that the complementarity condition $y_j\, y_{j+n} = 0,\ \forall j \in \{1,\ldots,n\}$ is taken into account at every stage. First off, by looking for extreme point solutions to the pricing problem, any negative weighted cycle W_V^k cannot contain both y_j and y_{j+n} variables simultaneously. Secondly, $\mathbf{y}_H^k$ is established in (6.131) by dichotomy on the signs of $\boldsymbol{\lambda}_P$.

All in all, the weighted cycle W_V^k found in the pricing problem is the support of incomplete information about the direction yet, once W_V^k is identified, the complete cycle is always uniquely determined. In this respect, let W_V^k be called a *contracted weighted cycle* whereas the weighted cycle produced with $\mathbf{y}^k = [\mathbf{y}_H^k,\ \mathbf{y}_V^k]$ is named the *extended weighted cycle* and is denoted by $W_{V\cup H}^k$. One can then think of the variables in the sets H_P and V_P as those variables that are *hidden* and *visible* in the *contracted* linear system (6.129). Recall that the contracted weighted cycle W_V^k is always directed. Whether or not its extension $W_{V\cup H}^k$ is itself guaranteed to be directed over the set of residual variables in J^k is directly related to the free nature of hidden variables in H_P. In other words, whether the step size computed next is certainly positive depends on the content of P, see Proposition 34.

6.3.3 Step size and updates

The step size ρ^k of the extended weighted cycle $W^k_{V\cup H}$ is computed with respect to the residual capacities of the directed variables forming it divided by their respective contribution as

$$\rho^k := \min_{j\in W^k_{V\cup H}} \left\{ \frac{r^k_j}{y^k_j} \right\} \geq 0\,. \tag{6.132}$$

A new primal solution $\mathbf{x}^{k+1} := \mathbf{x}^k + \rho^k \vec{\mathbf{v}}^k$ with cost z^{k+1} is obtained by computing

$$\forall j \in \{1,\ldots,n\} \qquad x^{k+1}_j := \begin{cases} x^k_j + \rho^k\, y^k_j\,, & \text{if } j \in W^k_{V\cup H} \\ x^k_j - \rho^k\, y^k_j\,, & \text{if } j+n \in W^k_{V\cup H} \\ x^k_j, & \text{otherwise} \end{cases} \tag{6.133}$$

$$z^{k+1} := z^k + \rho^k\, \mu^k_V. \tag{6.134}$$

Depending on the choice of the subspace basis $\mathbf{A}_P$, $\mathbf{x}^{k+1}$ represented by $[\mathbf{x}^{k+1}_F, \mathbf{x}^{k+1}_L, \mathbf{x}^{k+1}_U]$ could be nonbasic. Section 6.4.2 explains how and when this can happen with the conceptualization of *interior directions*. We simply mention that any nonbasic solution $\mathbf{x}^{k+1}$ can be rendered basic by solving a restricted problem over the set of free variables:

$$\begin{aligned} z^{k+1} = \min \quad & \mathbf{c}_F^\intercal \mathbf{x}_F + \mathbf{c}_L^\intercal \mathbf{x}_L + \mathbf{c}_U^\intercal \mathbf{x}_U \\ \text{s.t.} \quad & \mathbf{A}_F \mathbf{x}_F + \mathbf{A}_L \mathbf{x}_L + \mathbf{A}_U \mathbf{x}_U = \mathbf{b} \\ & \mathbf{l}_F \leq \mathbf{x}_F \leq \mathbf{u}_F, \quad \mathbf{x}_L = \mathbf{x}^{k+1}_L, \quad \mathbf{x}_U = \mathbf{x}^{k+1}_U. \end{aligned} \tag{6.135}$$

This problem identifies directed weighted cycles comprising free variables only, and increases or decreases the value of these variables until some lower and upper bounds are reached while possibly improving the overall solution cost. In network flows terminology, one obtains a *cycle free solution*, that is, a network solution containing no cycle composed of free arcs only, see Ahuja et al. (1993). Nonbasic network solutions are handled analogously in Gauthier et al. (2016, Section 3.1). This small manipulation only reflects that the set P is often selected with respect to F. Since the former must by definition be a linear independent set of variables in $\mathbf{A}$, having a basic solution means that no further verification on free variables must be done.

There only remains to update the residual problem $LP(\mathbf{x}^{k+1})$ with residual capacities $\mathbf{r}^{k+1}$ and column partition $\{F, L, U\}$ and to select a new subspace basis $\mathbf{A}_P$. Another iteration $k \to k+1$ then starts in Step 1.

6.4 Properties

This generic algorithm ultimately depends on a single parameter, that is, the selection of the set P. Section 6.4 derives two propositions revolving around this selection. In Section 6.4.1, we underline particular well known variants of this generic framework whereas Section 6.4.2 qualifies the kinds of directions found with the pricing problem.

6.4.1 Special cases

Let us start with a family of variants which perform a positive step size at every iteration. This section is completed with four specific variants found in the linear programming literature.

Proposition 34. *Let $\mathbf{x}^k$, $k \geq 0$ be a nonoptimal solution to LP (6.104). Given $P \subseteq F$, the step size of $\vec{\mathbf{v}}^k$ is guaranteed to be positive.*

Proof. If $\emptyset \subseteq P \subseteq F$ or equivalently $P \cap \{L \cup U\} = \emptyset$, then all y-variables in H_P are residual as well as those in V_P. Therefore, the primal/dual pair of the pricing problem (6.128)/(6.129) respectively match the necessary and sufficient primal/dual optimality conditions of Proposition 33. Indeed, the *a posteriori* extended cycle $W^k_{V \cup H}$ obtained from W^k_V trivially only uses variables in J^k and is as such a negative weighted cycle in $LP(\mathbf{x}^k)$ meaning that the associated step size is positive. □

Remark. Consider the case $P \nsubseteq F$ from the dual perspective. If there exists a $j \in P$ such that $j \notin F$, then the null reduced cost imposed on both y_j and y_{j+n} is too stringent with respect to the dual optimality condition. The reduced cost of the y-variable with a null residual capacity is irrelevant which would incidentally have granted more freedom to $\boldsymbol{\psi}_R$. In other words, this overly restrictive observation is also echoed in the primal form where a weighted cycle using such a y-variable is made possible, i.e., the additional column in the primal form comes from the additional constraint in the dual form.

Case $P = \emptyset$. When choosing $P = \emptyset$, it amounts to a subspace basis $\mathbf{A}_\emptyset$ of dimension zero which in turn means that $V(\mathbf{A}_\emptyset) = \{\mathbf{0}\}$. Since the vector subspace contains only the null vector, there are no compatible variables basic or otherwise. There is no linear transformation, that is, $\mathbf{T}_\emptyset = \mathbf{T}_\emptyset^{-1} = \left[\begin{smallmatrix} \mathbf{I}_0 & \emptyset \\ \emptyset & \mathbf{I}_m \end{smallmatrix}\right] = \mathbf{I}_m$. From a dual point of view, the m-dimensional dual vector $\boldsymbol{\pi}$ is fully optimized to maximize the minimum reduced cost μ. From a dual point of

view, the pricing problem contains only but all the residual variables and guarantees a positive step size (otherwise the current solution is optimal). When $\mathbf{A}$ is a network flow incidence matrix, this particular case corresponds to the remarkable strongly polynomial minimum mean cycle-canceling algorithm of Goldberg and Tarjan (1989) devised for capacitated minimum cost flow problems. With respect to arbitrary linear programs, it appears natural to think of yet another analogy: *minimum mean weighted cycle-canceling* algorithm. From a mechanical point of view, the adaptation is straightforward. However, the extent to which time complexity properties of MMCC are also portrayed in the latter is left for another paper.

Case $P = F$. When choosing $P = F$, it corresponds to the strategy developed by Elhallaoui et al. (2011) in IPS. The vector subspace $\mathbf{V}(\mathbf{A}_F)$ includes all columns of $\mathbf{A}_F = \left[\begin{smallmatrix}\mathbf{A}_{RF}\\ \mathbf{A}_{SF}\end{smallmatrix}\right]$ but none associated with the degenerate basic variables: $\forall j \in B,\ \mathbf{a}_j \in \mathbf{V}(\mathbf{A}_F) \Leftrightarrow j \in F$. The linear transformation is given by $\mathbf{T}_F^{-1} = \left[\begin{smallmatrix}\mathbf{I}_f & \mathbf{0}\\ -\mathbf{M} & \mathbf{I}_{m-f}\end{smallmatrix}\right]$, where $\mathbf{M} = \mathbf{A}_{SF}\mathbf{A}_{RF}^{-1}$. As a special case of Proposition 34, a positive step size is guaranteed.

Case $P = B$. When choosing $P = B$, we have m linearly independent column vectors with $\mathbf{A}_B$ and $\mathbf{V}(\mathbf{A}_B) = \mathbb{R}^m$. All variables are compatible whereas the sets P and Q respectively correspond to all basic and nonbasic variables. The subspace basis induces $\mathbf{T}_B = \mathbf{T}_B^{-1} = \left[\begin{smallmatrix}\mathbf{I}_m & \emptyset\\ \emptyset & \mathbf{I}_0\end{smallmatrix}\right] = \mathbf{I}_m$ and there is no transformation. Most importantly, it fixes $\boldsymbol{\pi}^\intercal = \mathbf{c}_B^\intercal \mathbf{A}_B^{-1}$ and the generic algorithm becomes PS with Dantzig's pivot-selection rule. When $B \cap \{L \cup U\} \neq \emptyset$, that is, when at least one basic degenerate variable is present, the set H_P contains y-variables with null residual capacities and a null step size can occur, i.e., a degenerate pivot.

Case $P \supseteq F$. When choosing $P \supseteq F$, we have $f \leq p \leq m$ linearly independent column vectors in $\mathbf{A}_P$. This is the strategy used in Elhallaoui et al. (2005, 2008) for solving set partitioning problems by column generation, see Gauthier et al. (2015c). While the equivalent form with the columns of $\mathbf{A}_B$ exists, the subspace basis $\left[\begin{smallmatrix}\mathbf{I}\\ \mathbf{M}\end{smallmatrix}\right]$ is obtained by design, more precisely heuristic row clustering as seen in Figure 6.40. If $P \supset F$, then the set H_P contains $(p - f)$ y-variables with null residual capacities. This possibly larger than necessary subspace basis gives a lot of freedom in the implementation of DCA, a method steered by practical imperatives. Indeed, in typical applications (Desrosiers et al. 1995, Desaulniers et al. 1998), a vehicle route or a crew schedule covers several tasks, say on average $\bar{m}$, which implies that the number of variables assuming value one in the basis is of the order $m/\bar{m}$. DCA thus capitalizes on the characterization of set partitioning optimal binary solutions which

are usually highly degenerate. The compatibility interpretation follows by design, a column $\mathbf{a}_j,\ j \in \{1,\ldots,n\}$, of $\mathbf{A}$ is compatible with the row clustering if and only if the itinerary or schedule respects that clustering into multi-task activities. Computational results also reveal that there is no inevitable correlation between an algorithm's inefficiency and degeneracy, contradicting common belief. See for example Benchimol et al. (2012) for the implementation of a stabilized DCA on highly degenerate multi-depot vehicle scheduling problems.

6.4.2 Interior directions

Since this paper backtracks on the edge movement induced by a pivot by first considering the direction of travel, let us add a layer of definition on the resulting impact of this direction.

Definition 15. *Let C be a convex polyhedron. Given a vertex $\mathbf{x} \in C$ and a direction $\vec{\mathbf{v}} \neq \mathbf{0}$, let $\mathbf{x} + \rho\vec{\mathbf{v}} \in C$ for some $\rho > 0$. The vector $\vec{\mathbf{v}}$ is called an* edge direction *originating from $\mathbf{x}$ if for $0 < \delta < \rho$, the vector $\mathbf{x} + \delta\vec{\mathbf{v}}$ belongs to an edge of C. Otherwise, a nonedge direction is called an* interior direction *originating from $\mathbf{x}$.*

An important property of the IPS algorithm is its movement on an edge of the polyhedron defined by the set of constraints of LP (6.104) at every iteration (Elhallaoui et al. 2011). The main idea of the proof is as follows. Consider a generic basis composed of the columns of $\mathbf{A}_F$ completed with $m - f$ artificial variables at zero. Because the index set of the visible y-variables is $V_P^k = L \cup U$ at iteration k, either a single compatible variable or a combination of at most $m - f + 1$ incompatible variables at their lower or upper bounds is selected in the weighted cycle W_V^k. In the former case, it acts as in PS with a nondegenerate pivot, hence a movement along an edge. In the latter case, the selected incompatible variables can enter the current basis one by one with degenerate pivots, each pivot removing an artificial variable thus maintaining the basis status of the solution, whereas the last pivot is nondegenerate, hence the movement is made along an edge. Let us formalize this result.

Proposition 35. (Elhallaoui et al. 2011, Proposition 4) *Let $\mathbf{x}^k,\ k \geq 0$, be a nonoptimal basic solution to LP (6.104). For $P = F$, the direction $\vec{\mathbf{v}}^k$ is an edge direction.*

Proposition 34 shows that the family of algorithms with $P \cap \{L \cup U\} = \emptyset$, or equivalently $P \subseteq F$, ensures a positive step size at every iteration. This also means that the oracle associated with any of these variants is able to verify the necessary and sufficient optimality conditions. While one might rest uneasy about equivalent necessary and sufficient optimality conditions provided by two different oracles, the following proposition sheds light on their

content and characterizes improving interior directions originating from a nonoptimal basic solution $\mathbf{x}^k$. In a nut shell, since the case $P = F$ only identifies edge directions by Proposition 35, it provides the smallest contracted linear system $\bar{K}_{SV}\mathbf{y}_V = \mathbf{0}$ able to exhaustively identify the set of feasible edge directions. Variants using $P \subset F$ must then *intuitively* contain these edge directions or combinations of these, i.e., interior directions.

For $P \subseteq F$, any extreme point solution $\mathbf{y}^k$ to the pricing problem (6.129) is in a one-to-one correspondence with an extreme ray of the cone defined by removing the normalization constraint, hence in a one-to-one correspondence with the direction $\vec{\mathbf{v}}^k$, see Definition 10 and Figure 6.38. Let Ω_P^k be the set of these directions, where Ω_F^k hence corresponds to the set of edge directions (Proposition 35).

Proposition 36. *Let $\mathbf{x}^k$, $k \geq 0$, be a nonoptimal basic solution to LP (6.104). For $P \subset F$, if $\vec{\mathbf{v}}^k \notin \Omega_F^k$, then it is an interior direction.*

Proof. For $P \subset F$, the pricing problem involves more visible y-variables compared to the case with $P = F$ because $V_P \supset V_F$. At the same time, it contains $(f - p)$ more constraints since $m - p + 1 > m - f + 1$. Therefore, if $\vec{\mathbf{v}}^k \in \Omega_P^k \setminus \Omega_F^k$, it can be expressed as a nonnegative combination of the edge directions of Ω_F^k by the representation theorems of Minkowski and Weyl (see Schrijver 1986), and as such, it is an interior direction originating from $\mathbf{x}^k$ (Definition 15). □

Note that a direction leading to a nonbasic solution can only happen with an interior direction. As such, observe that the case $P = \emptyset$ is one of the variant susceptible to lead to nonbasic solutions. However, since $\mathbf{T}_\emptyset^{-1} = \mathbf{I}_m$ and all dual variables are optimized in the pricing problem, the simplifications applicable to the different steps of the generic algorithm in Figure 6.41 imply that maintaining the basic nature of the solutions is irrelevant. For all other cases, fetching a linearly independent set P can be made simple by solving (6.135).

Illustrative example. For illustrative purposes, consider the following linear program with three variables and four inequality constraints. Let $s_1, \ldots, s_4$ be the slack variables associated with each constraint at $\mathbf{x}^0 = (x_1^0, x_2^0, x_3^0) = (0, 0, 0)$ and assume the initial basic solution uses these slack variables at values $s_1 = 21, s_2 = 8, s_3 = 15$ and $s_4 = 32$ for a total cost of $z^0 = 0$. This basic solution is nondegenerate and there are three edge directions according to the

selected entering variable x_1, x_2, or x_3.

$$\begin{array}{rrrrrl} \max & 130x_1 & +80x_2 & +60x_3 & & \\ \text{s.t.} & 2x_1 & - x_2 & + 2x_3 & \leq 21 \\ & - x_1 & + x_2 & - x_3 & \leq 8 \\ & 2x_1 & - x_2 & - x_3 & \leq 15 \\ & - x_1 & - x_2 & + 2x_3 & \leq 32 \\ & x_1, & x_2, & x_3 & \geq 0 \end{array}$$

Figure 6.42 details the content of the direction $\vec{\mathbf{v}}^0$ found using the pricing problem (6.129) under the suggested set P and weight vector $\mathbf{w} = \mathbf{1}$. The minimum mean cost of the associated weighted cycle W_V^0 is given by μ_V^0 and the step size ρ^0 can be recovered with (6.132). Finally, the new solution $\mathbf{x}^1$ is obtained with (6.133) whereas the updated objective function z^1 comes from (6.134). As expected, the direction found with the improved primal simplex algorithm ($P = F$) follows an edge and yields an extreme point solution. Notice that since the current solution is not degenerate, we have $F = B$ and would have found the same direction upon selecting x_1 as the entering variable in the primal simplex algorithm.

However, when $P = \emptyset$, the direction so happens to be interior and yields a feasible point which is not an extreme point (six variables take positive values). The reader can verify that it is indeed the combination of the three edge directions with weights $\{1/6, 1/4, 1/12\}$ (see Figure 6.43). By no means do we imply the set $P = \emptyset$ provides all around better directions than with $P = F$. In fact, it suffices to modify the coefficients of x_1 and x_2 in the third constraint to 1 and 3 to get the opposite effect when using the uniform one weight vector. The last example still is with the set $P = \emptyset$ but uses a weight vector whose every element w_j is determined by computing the squared norm $\|\mathbf{a}_j\|^2$ of each column, i.e., $w_{x_1} = 2^2 + (-1)^2 + 2^2 + (-1)^2 = 10$. The pricing problem finds a different interior direction which happens to be within the x_1x_2-face.

6.5 Conclusion

This paper unites under one generic framework several known algorithms with a broad spectrum of possibilities whereby both extreme cases correspond to the primal simplex and minimum mean weighted cycle-canceling algorithms. Several properties are established for different family members. In particular, a family that provides positive step sizes at every iteration and another one for which the pricing problems provide edge directions only. The

Variant	c_j coefficient Direction & Solution	130 x_1	80 x_2	60 x_3	s_1	s_2	s_3	s_4	z^1	μ_V^0	ρ^0
$P = F$	$\vec{\mathbf{v}}^0$ (edge)	1	0	0	-2	1	-2	1		130	7.5
	$\mathbf{x}^1$	7.5	0	0	6	15.5	0	39.5	975		
$P = \emptyset$	$\vec{\mathbf{v}}^0$ (interior) $w_j = 1$	$1/6$	$1/4$	$1/12$	$-1/4$	0	0	$1/4$		$140/3$	84
	$\mathbf{x}^1$	14	21	7	0	8	15	53	3920		
$P = \emptyset$	$\vec{\mathbf{v}}^0$ (face) $w_j = \|\mathbf{a}_j\|^2$	$1/22$	$1/11$	0	0	$-1/22$	0	$3/22$		$145/11$	176
	$\mathbf{x}^1$	8	16	0	21	0	15	56	2320		

Fig. 6.42: Directions found at $\mathbf{x}^0$ in pricing for $P = F$ and $P = \emptyset$

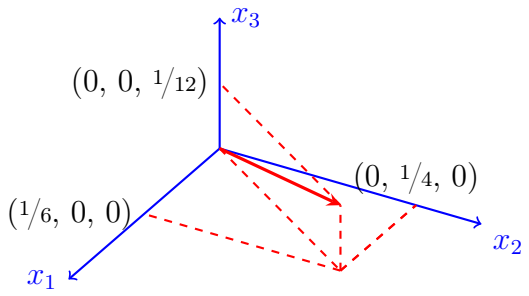


Fig. 6.43: Three-dimensional interior direction $(x_1, x_2, x_3) = (1/6, 1/4, 1/12)$ with $P = \emptyset$ and $\mathbf{w} = \mathbf{1}$

improved primal simplex algorithm is remarkably the only variant which qualifies for both features.

While interior directions are certainly usual in the realm of nonlinear optimization, it is not so often that one thinks about such possibilities for simplex-like algorithms. The oracle for such findings does not require fancy derivatives and in fact remains linear.

It is often noted in column generation that giving more structure to the pricing problem allows faster retrieval of improving columns. A potentially beneficial avenue might therefore be the use of *dual-optimal inequalities*. The idea is to add to the primal formulation some additional variables and columns for which the corresponding dual inequalities are always satisfied by an optimal dual solution. These have been used in several applications such as the cutting stock and the bin packing problems, see Valério de Carvalho (2005), Ben Amor et al. (2006) and Gschwind and Irnich (2016). Dual-optimal inequalities in these papers must verify the necessary and sufficient dual optimality conditions of Proposition 33, but we hope the latter also allows the retrieval of additional such inequalities.

All in all, the implementation of this generic framework is a chapter of its own. While this paper has no computational study pretension, several ideas have already been suggested as promising assets. The transformation matrix $\mathbf{T}_P^{-1}$ induces structure in the technological matrix whereas the residual problem automates degeneracy screening. By combining both

constructions, extensive variable screening in the pricing problem can be done on several fronts such as compatibility and partial reduced costs to residual upper bounds ratios. The first kind being easy performed pivots whereas the second kind aims at detecting good cost to step size pivots.

Acknowledgements

Jacques Desrosiers acknowledges the National Science and Engineering Research Council of Canada for its financial support.

7. CONCLUSION (FR)

Plaçons-nous cinq ans plus tôt. *Dynamic Constraint Aggregation* traduit une préoccupation pratique : accélérer la vitesse de résolution des problèmes de partitionnement en génération de colonnes. La fondation est mise en place tant pour l'algorithme du *Improved Primal Simplex* que pour la règle du *Positive Edge* qui apparaîssent peu de temps après. Chacun de ces trois outils repose sur une transformation linéaire pour mieux capturer les directions améliorantes. C'est sans doute ce qui se dégage le mieux de notre revue de littérature, mais nous croyons également que la valeur théorique apporte des réponses quant au succès (ou non) de certains choix d'implémentation.

Qu'en est-il aujourd'hui ? La présentation d'IPS nous apparaît plus accessible ne serait-ce que pour l'interprétation visuelle de la dégénérescence qui transparaît dans les exemples réseaux. L'algorithme du simplexe s'appuie sur la convexité des problèmes linéaires, à tel point qu'il souffre d'un manque de considérations géométriques induites par les contraintes technologiques. En effet, du point de vue visuel, la dégénérescence est hors de propos. Le lecteur est invité à penser aux méthodes de points intérieurs qui sont immunisées en traversant le polyèdre à l'intérieur. IPS tente de faire un compromis entre la considération explicite de la dégénérescence et un des avantages non négligeable du simplexe primal qui voyage uniquement sur des arêtes.

En appliquant IPS aux problèmes de flot, l'étude met en lumière des propriétés intéressantes qui passent en large partie par l'importance accordée aux phases. Il en découle que des améliorations peuvent être apportées à des algorithmes comparables en forçant ceux-ci à épouser le principe des phases. C'est ainsi, qu'en mélangeant une gymnastique de contraction avec le comportement de l'algorithme du *Minimum Mean Cycle-Canceling*, une complexité fortement polynomiale est exposée. Alors qu'une implémentation efficace d'un simplexe primal repose sur une panoplie de trucs et astuces dont une bonne gestion de la tarification partielle, l'algorithme de Contraction-Expansion n'utilise pour l'instant aucune technique d'accélération. Malgré qu'il ne soit donc pas en mesure de compétitionner avec les outils disponibles à ce jour, la gestion du problème contracté fait en sorte qu'il est possible d'effectuer un arbitrage à la fois sur la longueur des pas et l'envergure du cycle de coût moyen en écartant temporairement

des arcs du problème de tarification. Le premier biais insiste sur une longueur de pas désirée tandis que le second oppose les *rooted costs* au paramètre d'optimalité.

Sur une autre note, *Cancel-and-Tighten* est devenu une partie intégrante de la thèse à force d'afficher des comportements particulièrement fascinants. Notamment, son fonctionnnement faisant en sorte que les variables duales sont amenées à se fixer à des valeurs optimales de façon non-décroissante. Celui-ci hérite également intrinsèquement d'une forme de tarification partielle. Enfin, une des plus jolies contributions renvoie au nouveau résultat de complexité théorique proposé en combinant MMCC avec *Cancel-and-Tighten*. Rappelons que nous introduisons une mesure heuristique qui influence par ailleurs la vitesse de résolution.

Comme la section des sujets divers de l'introduction le suggère, IPS a ouvert le chemin vers beaucoup d'études. Plusieurs questions se dressent au fur et à mesure que notre compréhension s'élargit. À titre d'exemples, pensons à la possibilité d'intégrer la stabilisation des variables duales ou encore un algorithme de type MMCC pour les problèmes linéaires avec sa propre analyse de complexité. Par ailleurs, une des pistes attrayantes consiste à porter d'autant plus d'attention au problème de tarification afin d'extraire des directions qui maintiennent l'intégralité. Voilà le propre du *Integral Simplex Using Decomposition* dont le prochain tournant consiste à coordonner son fonctionnement avec la génération de colonnes.

En parlant de génération de colonnes, le problème de cycle moyen minimal *(minimum mean cycle)* étant à la base de plusieurs problématiques industrielles, de nombreux algorithmes dédiés à sa résolution sont disponibles, tout comme des études approfondies sur le sujet. Ces dernières concluent que malgré sa complexité exponentielle, l'algorithme de Howard domine en pratique les alternatives. Notre compréhension de MMCC indique cependant qu'un algorithme capable de dénicher beaucoup de cycles est plus important que sa capacité de résoudre à l'optimalité rapidement. La méthode de résolution du problème de tarification mérite donc elle-même un examen autant dans l'esprit du *Vector Space Decomposition* qu'en génération de colonnes où cette observation est une des clés de toute bonne implémentation.

7. CONCLUSION (EN)

Journey back five years ago. Dynamic Constraint Aggregation is driven by practical concerns: improve the column generation resolution speed of set partitioning problems. The foundation is laid for the Improved Primal Simplex algorithm and the Positive Edge rule which surface shortly thereafter. All three tools rely on a linear transformation to better capture the improving directions. This much is certainly made clear in our literature review but we believe it also adds theoretical value by enlightening the reason why certain implementation choices prove successful while others less so.

Where do we stand today? The presentation of IPS seems more accessible and the reason behind that might go hand in hand with the visual interpretation of degeneracy. The simplex algorithm relies on the convex nature of linear programs, so much so it suffers from the lack of geometrical consideration induced by the technological constraints. Indeed, from a visual perspective, degeneracy is irrelevant. Consider for instance interior point methods which are more or less naturally immune to degeneracy by traveling inside the polyhedron. IPS compromises on one of the most interesting asset of the simplex algorithm which is to travel around the polyhedron although degeneracy is also taken into account.

By applying IPS to network flow problems, the study shows that interesting properties can be obtained, and that there is room for improvement on comparable algorithms. Indeed, largely relying on phases, a strongly polynomial algorithm is devised by mixing the contraction scheme with the Minimum Mean Cycle-Canceling algorithm. While the most efficient primal simplex code today rely in very large measure on good partial pricing methods, the Contraction-Expansion algorithm has yet to incorporate any such acceleration technique. Despite not being as competitive as existing algorithms, possible partial pricing arbitrages include guaranteed step sizes and mean cycle cost potential by temporarily discarding contracted arcs from the pricing problem. The first bias insists on the minimal desired step size while the second opposes rooted costs to the optimality parameter.

On another note, Cancel-and-Tighten has become an integral part of this dissertation by recurringly appearing in our discussions with fascinating behaviors. Notably, the dual variables are guided to optimal values from below. It also inherently benefits from a kind

of partial pricing. Furthermore, one of the most interesting highlight is the new strongly polynomial time complexity proposed by combining MMCC with Cancel-and-Tighten. Recall that we introduce a heuristic scheme to make this possible which incidentally also improves the performance.

As the miscellaneous subjects section of our introduction suggests, IPS has opened a very broad field of study. Numerous questions are raised as our understanding deepens. For instance, whether dual variables stabilization can be integrated or the possibility of an MMCC like algorithm for linear programs with its own complexity study. Moreover, one of the promising side product pays more attention to the pricing step in order to maintain integrality. Such is the purpose of the Integral Simplex Using Decomposition for which the next milestone is the integration in a column generation framework.

Speaking of column generation, it turns out that several algorithms dedicated to solving the minimum mean cycle problem are available, and have been studied in-depth because of their presence in many industrial challenges. The conclusion is that Howard's algorithm seems to be most efficient despite its theoretical exponential complexity. However, our understanding of MMCC tells us that an algorithm capable of providing more cycles might be more important than the speed at which it finds the optimal one. The resolution method of the pricing problem thus merits examination in the Vector Space Decomposition framework as much as in column generation where this observation is paramount to any well designed implementation.

REFERENCES

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Upper Saddle River, NJ, USA, 1993.

Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998. doi:10.1287/opre.46.3.316.

Hatem M. T. Ben Amor, Jacques Desrosiers, and José M. Valério de Carvalho. Dual-Optimal Inequalities for Stabilized Column Generation. *Operations Research*, 54(3):454–463, 2006. doi:10.1287/opre.1060.0278.

Hatem M. T. Ben Amor, Jacques Desrosiers, and Antonio Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167–1184, 2009. doi:10.1016/j.dam.2008.06.021.

Pascal Benchimol, Guy Desaulniers, and Jacques Desrosiers. Stabilized dynamic constraint aggregation for solving set partitioning problems. *European Journal of Operational Research*, 223(2): 360–371, 2012. doi:10.1016/j.ejor.2012.07.004.

Dimitri Panteli Bertsekas. A distributed algorithm for the assignment problem. Technical report, Working paper, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, USA, 1979.

Robert Gary Bland. New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2(2):103–107, May 1977. doi:10.1287/moor.2.2.103.

Abraham Charnes. Optimality and degeneracy in linear programming. *Econometrica*, 20(2):160–170, April 1952. doi:10.2307/1907845.

Abraham Charnes and William W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4):181–186, 1962. doi:10.1002/nav.3800090303.

Jean Cochet-Terrasson, Guy Cohen, Stéphane Gaubert, Michael Mc Gettrick, and Jean-Pierre Quadrat. Numerical computation of spectral elements in max-plus algebra. In *IFAC Conference on System Structure and Control*, Nantes, France, July 8–10 1998.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, USA and London, England, 3rd edition, 2009.

George B. Dantzig. *Linear programming and extensions*. Princeton University Press, Princeton, NJ, USA, 1963.

George B. Dantzig and Mukund N. Thapa. *Linear Programming 1: Introduction*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, USA, 1997.

George B. Dantzig and Mukund N. Thapa. *Linear Programming 2: Theory and Extensions*. Springer Series in Operations Research and Financial Engineering (Book 2). Springer, New York, NY, USA, 2003.

George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960. doi:10.1287/opre.8.1.101.

Ali Dasdan. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Transactions on Design Automation of Electronic Systems*, 9(4):385–418, October 2004. doi:10.1145/1027084.1027085.

Ali Dasdan, Sandy S. Irani, and Rajesh K. Gupta. Efficient algorithms for optimum cycle mean and optimum cost to time ratio problems. In *Design Automation Conference, 1999. Proceedings. 36th*, pages 37–42, New Orleans, LA, USA, 1999. IEEE. doi:10.1109/DAC.1999.781227.

Guy Desaulniers, Jacques Desrosiers, Irina Ioachim, Marius M. Solomon, François Soumis, and Daniel Villeneuve. A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In Teodor Gabriel Crainic and Gilbert Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Springer, New York, NY, USA, 1998. doi:10.1007/978-1-4615-5755-5_3.

Jacques Desrosiers and Marco E. Lübbecke. Branch-price-and-cut algorithms. In James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., Chichester, West Sussex, England, January 2011. doi:10.1002/9780470400531.eorms0118.

Jacques Desrosiers, Yvan Dumas, Marius M. Solomon, and François Soumis. Time constrained routing and scheduling. In Michael Ball, Tom Magnanti, Clyde Monma, and George Nemhauser, editors, *Handbooks in Operations Research and Management Science, Vol. 8: Network Routing*, volume 8, chapter 2, pages 35–139. Elsevier, Maryland Heights, MO, USA, October 1995. doi:10.1016/S0927-0507(05)80106-9.

Jacques Desrosiers, Jean Bertrand Gauthier, and Marco E. Lübbecke. A contraction-expansion algorithm for the capacitated minimum cost flow problem. Southampton, England, 2013. Presentation at VeRoLog 2013.

Jacques Desrosiers, Jean Bertrand Gauthier, and Marco E. Lübbecke. Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, 236(2):453–460, 2014. doi:10.1016/j.ejor.2013.12.016.

DIMACS. Network Flows and Matching: First DIMACS Implementation Challenge, 1990–1991. URL `ftp://dimacs.rutgers.edu/pub/netflow`.

Olivier du Merle, Daniel Villeneuve, Jacques Desrosiers, and Pierre Hansen. Stabilized column generation. *Discrete Mathematics*, 194:229–237, 1999. doi:10.1016/S0012-365X(98)00213-1.

Jack Edmonds and Richard Manning Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, April 1972. doi:10.1145/321694.321699.

Issmail Elhallaoui, Daniel Villeneuve, François Soumis, and Guy Desaulniers. Dynamic aggregation of set partitioning constraints in column generation. *Operations Research*, 53(4):632–645, 2005. doi:10.1287/opre.1050.0222.

Issmail Elhallaoui, Guy Desaulniers, Abdelmoutalib Metrane, and François Soumis. Bi-dynamic constraint aggregation and subproblem reduction. *Computers & Operations Research*, 35(5): 1713–1724, 2008. doi:10.1016/j.cor.2006.10.007.

Issmail Elhallaoui, Abdelmoutalib Metrane, François Soumis, and Guy Desaulniers. Multi-phase dynamic constraint aggregation for set partitioning type problems. *Mathematical Programming*, 123(2):345–370, 2010. doi:10.1007/s10107-008-0254-5.

Issmail Elhallaoui, Abdelmoutalib Metrane, Guy Desaulniers, and François Soumis. An Improved Primal Simplex algorithm for degenerate linear programs. *INFORMS Journal on Computing*, 23:569–577, 2011. doi:10.1287/ijoc.1100.0425.

Lester Randolph Ford, Jr. and Delbert Ray Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956. doi:10.4153/CJM-1956-045-5.

Lester Randolph Ford, Jr. and Delbert Ray Fulkerson. *Flows in networks*. Princeton University Press, Princeton, NJ, USA, 1962.

Satoru Fujishige. A capacity-rounding algorithm for the minimum cost circulation problem: A dual framework of Tardos' algorithm. *Mathematical Programming*, 35:298–308, 1986. doi:10.1007/BF01580882.

Komei Fukuda. *Oriented Matroid Programming*. PhD thesis, University of Waterloo, Waterloo, ON, Canada, July 1982.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. Decomposition theorems for linear programs. *Operations Research Letters*, 42(8):553–557, December 2014. doi:10.1016/j.orl.2014.10.001.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. Vector space decomposition for linear programs. Les Cahiers du GERAD G-2015-26, HEC Montréal, Montreal, QC, Canada, March 2015a.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. About the minimum mean cycle-canceling algorithm. *Discrete Applied Mathematics*, 196:115–134, 2015b. doi:10.1016/j.dam.2014.07.005. Advances in Combinatorial Optimization.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. Tools for primal degenerate linear programs: IPS, DCA, and PE. *EURO Journal on Transportation and Logistics*, pages 1–44, 2015c. doi:10.1007/s13676-015-0077-5.

Jean Bertrand Gauthier, Jacques Desrosiers, and Marco E. Lübbecke. A strongly polynomial Contraction-Expansion algorithm for network flow problems. Les Cahiers du GERAD G-2016-18, HEC Montréal, Montreal, QC, Canada, March 2016.

Loukas Georgiadis, Andrew V. Goldberg, Robert Endre Tarjan, and Renato F. Werneck. An experimental study of minimum mean cycle algorithms. In Irene Finocchi and John Hershberger, editors, *2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 1–13, New York, NY, USA, January 2009. SIAM. doi:10.1137/1.9781611972894.1.

Andrew V. Goldberg and Robert Endre Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989. doi:10.1145/76359.76368.

Timo Gschwind and Stefan Irnich. Dual inequalities for stabilized column generation revisited. *INFORMS Journal on Computing*, 28(1):175–194, 2016. doi:10.1287/ijoc.2015.0670.

Paula M. J. Harris. Pivot selection methods of the Devex LP code. *Mathematical Programming*, 5 (1):1–28, 1973. doi:10.1007/BF01580108.

Ronald Arthur Howard. *Dynamic Programming and Markov Processes*. The MIT Press, Cambridge, MA, USA, 1960.

Alon Itai. Two-commodity flow. *Journal of the ACM*, 25(4):596–611, October 1978. doi:10.1145/322092.322100.

Richard Manning Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3):309–311, 1978. doi:10.1016/0012-365X(78)90011-0.

Victor Klee and George J. Minty. How good is the simplex algorithm? In Oved Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, NY, USA, 1972.

Morton Klein. A primal method for minimal cost flows with applications to the assignment and transportation problems. *Management Science*, 14(3):205–220, November 1967.

Edward Seymour Klotz. *Dynamic Pricing Criteria in Linear Programming*. PhD thesis, Stanford University, Stanford, CA, USA, July 1988.

Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, Emilie Danna, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Andrea Lodi, Hans Mittelmann, Ted Ralphs, Domenico Salvagnin, Daniel E. Steffy, and Kati Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011. doi:10.1007/s12532-011-0025-9.

Péter Kovács. Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127, 2015. doi:10.1080/10556788.2014.895828.

Andreas Löbel. Vehicle scheduling in public transit and Lagrangean pricing. *Management Science*, 44(12):1637–1649, 1998. doi:10.1287/mnsc.44.12.1637.

Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi:10.1287/opre.1050.0234.

John W. Mamer and Richard D. McBride. A decomposition-based pricing procedure for large-scale linear programs – An application to the linear multicommodity flow problem. *Management Science*, 46(5):693–709, 2000. doi:10.1287/mnsc.46.5.693.12042.

Roy E. Marsten, Matthew J. Saltzman, David F. Shanno, George S. Pierce, and J. F. Ballintijn. Implementation of a dual affine interior point algorithm for linear programming. *ORSA Journal on Computing*, 1989. doi:10.1287/ijoc.1.4.287.

Abdelmoutalib Metrane, François Soumis, and Issmail Elhallaoui. Column generation decomposition with the degenerate constraints in the subproblem. *European Journal of Operational Research*, 207(1):37–44, 2010. doi:10.1016/j.ejor.2010.05.002.

Jérémy Omer, Samuel Rosat, Vincent Raymond, and François Soumis. Improved Primal Simplex: A More General Theoretical Framework and an Extended Experimental Analysis. Les Cahiers du GERAD G-2014-13, HEC Montréal, Montreal, QC, Canada, March 2014.

Amar Oukil, Hatem M. T. Ben Amor, Jacques Desrosiers, and Hicham El Gueddari. Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. *Computers & Operations Research*, 34(3):817–834, 2007. doi:10.1016/j.cor.2005.05.011.

Ping-Qi Pan. A basis deficiency-allowing variation of the simplex method for linear programming. *Computers & Mathematics with Applications*, 36(3):33–53, 1998. doi:10.1016/S0898-1221(98)00127-8.

André F. Perold. A degeneracy exploiting LU factorization for the simplex method. *Mathematical Programming*, 19(1):239–254, 1980. doi:10.1007/BF01581646.

Tomasz Radzik and Andrew V. Goldberg. Tight bounds on the number of minimum-mean cycle cancellations and related results. *Algorithmica*, 11(3):226–242, 1994. doi:10.1007/BF01240734.

Vincent Raymond, François Soumis, and Abdelmoutalib Metrane. Improved primal simplex version 3: Cold start, generalization for bounded variable problems and a new implementation. Les Cahiers du GERAD G-2009-15, HEC Montréal, Montreal, QC, Canada, 2009.

Vincent Raymond, François Soumis, Abdelmoutalib Metrane, and Jacques Desrosiers. Positive edge: A pricing criterion for the identification of non-degenerate simplex pivots. Les Cahiers du GERAD G-2010-61, HEC Montréal, Montreal, QC, Canada, 2010a.

Vincent Raymond, François Soumis, and Dominique Orban. A new version of the Improved Primal Simplex for degenerate linear programs. *Computers & Operations Research*, 37(1):91–98, 2010b. doi:10.1016/j.cor.2009.03.020.

Hans Röck. Scaling techniques for minimal cost network flows. In Uwe Pape, editor, *Discrete Structures and Algorithms*, pages 181–191. Carl Hanser, Munich, Germany, 1980.

Samuel Rosat, Issmail Elhallaoui, François Soumis, and Andrea Lodi. Integral Simplex Using Decomposition with Primal Cuts. In Joachim Gudmundsson and Jyrki Katajainen, editors, *Experimental Algorithms*, volume 8504 of *Lecture Notes in Computer Science*, pages 22–33. Springer International Publishing, Switzerland, 2014. doi:10.1007/978-3-319-07959-2_3.

Samuel Rosat, Issmail Elhallaoui, François Soumis, and Driss Chakour. Influence of the normalization constraint on the integral simplex using decomposition. *Discrete Applied Mathematics*, 2016a. doi:10.1016/j.dam.2015.12.015.

Samuel Rosat, Frédéric Quesnel, Issmail El Hallaoui, and François Soumis. Dynamic penalization of fractional directions in the integral simplex using decomposition: Application to aircrew scheduling. Les Cahiers du GERAD G-2016-01, HEC Montréal, Montreal, QC, Canada, January 2016b.

David M. Ryan and Michael R. Osborne. On the solution of highly degenerate linear programmes. *Mathematical Programming*, 41(1-3):385–392, May 1988. doi:10.1007/BF01580776.

Rusian Sadykov and François Vanderbeck. Column generation for extended formulations. *EURO Journal on Computational Optimization*, 1(1-2):81–115, 2013. doi:10.1007/s13675-013-0009-9.

Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., Chichester, West Sussex, England, 1986.

Daniel Dominic Kaplan Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26(3):362–391, June 1983. doi:10.1016/0022-0000(83)90006-5.

Éva Tardos. A strongly polynomial minimum cost circulation algorithm. *Combinatorica*, 5(3): 247–255, 1985. doi:10.1007/BF02579369.

Éva Tardos. Strongly polynomial and combinatorial algorithms in optimization. volume 1 of *Proceedings of the International Congress of Mathematicians.*, pages 1467–1478, Kyoto, Japan, 1990.

Tamás Terlaky and Shuzhong Zhang. Pivot rules for linear programming: A survey on recent theoretical developments. *Annals of Operations Research - Annals OR*, 46-47(1):203–233, 1993. doi:10.1007/BF02096264.

Mehdi Towhidi, Jacques Desrosiers, and François Soumis. The positive edge criterion within COIN-OR's CLP. *Computers & Operations Research*, 49(0):41–46, 2014. doi:10.1016/j.cor.2014.03.020.

José M. Valério de Carvalho. Exact solution of bin-packing problems using column generation and branch-and-bound. *Annals of Operations Research*, 86(0):629–659, 1999. doi:10.1023/A:1018952112615.

José M. Valério de Carvalho. LP models for bin-packing and cutting stock problems. *European Journal of Operational Research*, 141(2):253–273, 2002. doi:10.1016/S0377-2217(02)00124-8.

José M. Valério de Carvalho. Using extra dual cuts to accelerate convergence in column generation. *INFORMS Journal on Computing*, 17(2):175–182, 2005. doi:10.1287/ijoc.1030.0060.

Philip Wolfe. A technique for resolving degeneracy in linear programming. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):205–211, 1963. doi:10.1137/0111016.

Neal E. Young, Robert Endre Tarjan, and James B. Orlin. Faster parametric shortest path and minimum balance algorithms. *Networks*, 21(2):205–221, March 1991. doi:10.1002/net.3230210206.

Abdelouahab Zaghrouti, François Soumis, and Issmail El Hallaoui. Integral Simplex Using Decomposition for the Set Partitioning Problem. *Operations Research*, 62(2):435–449, 2014. doi:10.1287/opre.2013.1247.