# Contributions to natural and extended formulations for Combinatorial Optimization Problems

Viet Hung Nguyen

# Contributions to natural and extended formulations for Combinatorial Optimization Problems

## *Applications to exact and approximation algorithms*

by

Viet Hung Nguyen

Submitted to the University Pierre and Marie Curie
in partial fulfillment of the requirements for the degree of

Habilitation à Diriger des Recherches

at the

UNIVERSITY PIERRE AND MARIE CURIE

December 2016

defense before the comittee composed by:

| | | |
|---|---|---|
| Francisco Barahona | IBM J. Watson Research Center | reviewer |
| Gérard Cornuéjols | Carnegie Mellon University | reviewer |
| Alain Quilliot | University Blaise Pascal | reviewer |
| A. Ridha Mahjoub | University Paris Dauphine | chairman |
| Jean-François Maurras | University Aix-Marseille | examiner |
| Patrice Perny | University Pierre and Marie Curie | examiner |
| András Sebő | CNRS, Grenoble | examiner |
| Michel Minoux | University Pierre and Marie Curie | coordinator |

# Acknowledgments

I would like to thank all the members of the comittee for accepting to examine this HDR thesis. It is a great honor for me as I always admire their works and their humanity.

# Contents

# List of Figures

# List of Tables

# Introduction

Polyhedral combinatorics is one of most fascinating subject during the last fifty years in Combinatorial Optimization. Among the pioneering works, one can cite the seminal paper by Jack Edmonds in 1965 where he formulated the maximum matching problem as a linear program. More importantly, based on the latter, he gave a combinatorial algorithm to solve the problem. Two important concepts have been introduced by Edmonds in this paper:

- "good algorithm", i.e. polynomial time algorithms,
- "good characterization", i.e. polyhedral descriptions of combinatorial optimization on which we can establish a min-max relation.

These are fundamental concepts in polyhedral combinatorics. Ideally, when one want to solve a combinatorial optimization problem by a polyhedral approach, one would like to have a complete linear description together with a polynomial time algorithm. This is the case for many well known problems in $P$ like maximum matching, minimum weight perfect matching, shortest path in a digraph without negative cycle,... There are also other problems for which one know a polynomial time algorithm but not a complete description. On can cite for example, the minimum cut problem, the cardinality constraint shortest path problem. However, to the best of our knowledge, there is no problem for that we know a complete description but not a polynomial time algorithm. This prove that finding a complete description for a combinatorial optimization is usually more difficult than solving the problem itself. It is useful at this stage to specify more exactly what could be a complete description. When we refer to a complete description, we could refer to

- a description by facet defining inequalities of the convex hull of the incidence vectors of the feasible solutions in their natural space. By natural space, we mean a minimal space for describing the solutions. Such a description, called *a natural description* (or perfect description), that we will denote by

$$P = \{x \in \mathbb{Q}^n \mid Ax \geq b \text{ where } A \in \mathbb{Q}^{m \times n} \text{ and } b \in \mathbb{Q}^m\}.$$

- a description in an extended space by linear inequalities with additional variables not necessary for describing the solutions. Such a description called *extended formulation*, that we will denote by

$$EF = \{(x, y) \in \mathbb{Q}^{n+p} \mid Ax + By \geq b \text{ where } A \in \mathbb{Q}^{m \times n},\ B \in \mathbb{Q}^{m \times p} \text{ and } b \in \mathbb{Q}^m\}.$$

  The projection of an extended formulation on the natural space should give a natural formulation of the problem.

For more details on the importance of these notions in combinatorial optimization, one can refer to the excellent survey by Conforti et al. [16]. For $NP$-hard problems, it is very hard to find natural or extended (complete) formulations. These formulations should be very complex and probably contain an exponential number of inequalities (in term of the dimension of the space). In this case, one tries to establish some *(mixed) integer formulation* $IP = \{x \in \mathbb{Z}^n \mid Ax \geq b\}$ or *extended (mixed) integer formulations* $IEF = \{x \in \mathbb{Z}^n,\ y \in \mathbb{R}^p \mid Ax + By \geq b\}$, i.e. formulations with linear and integrity constraints (the integrity constraints are simply 0/1 constraints for combinatorial optimization problem).

For a given problem, different formulations (linear or integer) could exist but they are not equivalent when being used for solving the problem. Thus, beyond the work of elaborating formulations, we are also interested in the efficiency of using them in linear programming or (mixed) integer programming techniques for solving the problem. There exists several criteria to measure this efficiency like the number of variables, the number of constraints, the quality of the linear programming relaxation, the adapta-

tion of the formulation in an Branch-and-Bound/Cut algorithm, etc. The two first criteria can be estimated without numerical experiments. In particular, a formulation is *compact* if the number of constraints is polynomial in terms of the number of variables and vice-versa, i.e. $P$ is compact if $m$ is a polynomial of $n$ and vice-versa. It is clear that one prefers compact formulations to non-compact ones unless if the linear programming relaxation of the non-compact one is of very good quality. Note that for given problem, extended formulations are usually "more compact" than natural ones. The reason is that an extended formulation contains additional variables allowing to express more properties of the solutions than a natural formulation. Consequently, this implies simplifications in the constraints. More precisely, given a combinatorial optimization problem for which we know two respectively natural and extended formulations, it usually happens that the first is compact and the second is non-compact.

As we have explained previously, the compactness is not the only criterion for efficiency of a formulation. In particular, for $NP$-hard problems, we may only have integer formulations (strengthened or not). Among them we may prefer non-compact formulations to compact ones if the firsts give linear programming relaxations of good quality which can be exploited in a branch-and-bound frame work or in a approximation algorithm with guarantee of performance. However, the use of a non-compact formulation implies cutting-planes algorithms and consequently one should solve the separation problem for the inequalities belonging to the formulation. This problem properly is an optimization problem and could be very difficult to solve, even $NP$-hard. In some cases, the separation problem of some class of inequalities in a natural formulation can be solved by using compact extended formulation if this class is resulted from the projection of the extended formulation in question to the natural space. Let us bring here more precisions of the use of the formulations in problem solving. Given a combinatorial optimization,

- for which there exists a natural formulation $P$ or an extended formulation $EF$. If $P$ is compact and/or $m$ and $n$ are reasonable values, we can directly solve $P$ by linear programming techniques. If $P$ is not compact and/or $m$ and $n$ are big,

theoretically we can solve the problem in polynomial time by ellipsoid method provided that the separation problem can be solved in polynomial time. We may also hope to design strongly polynomial time primal-dual algorithms like Edmonds did for matching problems. The same possibilities may be considered for the extended formulation $EF$.

- for which there is only an integer natural formulation $IP$ or an integer extended formulation $EIP$. In this case, exact solutions for the problem are based on implicit enumeration strategies like Branch-and-Bound, Branch-and-Cut, Branch-and-Price or Branch-and-Cut/Price algorithms. One may also use the formulations $IP$ or $EIP$ to design approximation algorithm with performance guarantee. In this case, one works essentially with the linear programming relaxation. One can enumerate here some usual approximation techniques using linear programming relaxations like

  - primal-dual approximation algorithms: We start with a feasible dual solution. The iterations of the algorithm consists in improving the current dual feasible solution and at the same time elaborating a partial solution to obtain at the end a feasible solution for the problem. The quality of this solution may be estimated by using the last feasible dual solution.
  - rounding techniques: We solve the linear programming to obtain an optimal solution and we try to make the latter integer by rounding its fractional components while keeping additional costs at least as possible.

For summary, the subjects for research in Combinatorial Optimization that we have discussed above are:

- the polyhedral studies of the convex hull of the solutions in their natural space which include, in particular, the linear description of the facet-defining inequalities and their separation problem.
- the studies on extended formulations, in particular for the cases when one does not know complete description of the convex hull or one known one but the

latter is not compact. The aim is to find compact extended formulations whose size is as small as possible.

- the application of the natural and extended formulations in finding exact or approximate solutions.

In this HDR thesis, we make contributions in these subjects for combinatorial optimization problems involving basic structures in graph theory like: cycles, trees, paths, star, vertex cover, edge cover, cut, multi-cuts, .... For certain problems, we characterize completely or partially the convex hull of their solutions in natural or extended spaces (star, star forest, Huffman trees). For others, we propose extended formulations of smaller size than the ones known in the literature (max cut, graph partitioning). For algorithmic side, based on known formulations, we give new exact algorithms that improve the best known time complexity for the problem (detecting negative cost cycles in undirected graphs). We also give first approximation algorithms for certain problems (directed tree cover, directed tour cover). We also consider some combinatorial problem with non convex constraints and discuss linearization/convexification techniques applied to these constraints (graph partitioning under capacity constraints). More precisely, we present the following works:

**Chapter 1. Natural formulations and combinatorial algorithms.**

- A dual algorithm for the Undirected Negative Cost Cycle Detecting (UNNCD) problem. Given an undirected graph $G$ whose edges are weighted by an arbitrarily cost, the problem is to detect the presence of an negative cost cycle in $G$. Our algorithm is based on a natural formulation of the UNNCD problem given by Seymour. This not only represents the first direct algorithm for UCCND but also improves the best known time complexity for UCCND.
  *Paper : V. H. Nguyen "A direct dual algorithm for detecting negative cost cycles in undirected graphs", submitted to SIAM Journal on Computing.*

- A partial description for the Huffman tree polytope. We show that this

partial description is sufficient for optimizing over the Huffman tree polytope when the optimal solution is an Huffman tree of maximum depth.

*Paper: J.-F. Maurras, T. H. Nguyen et V. H. Nguyen "On the linear description of the Huffman trees polytope" Discrete Applied Mathematics, Vol. 164:225-236 (2014).*

- A combinatorial approximation algorithm for the minimum directed tree cover problem in a directed graph $G$. A directed tree cover in $G$ is an rooted arborescence whose vertex set $S$ form a vertex cover, i.e. for any arc $(i, j)$ in $G$, at least $i$ or $j$ belongs to $S$. We give the first approximation algorithm with performance guarantee for the problem.

  *Paper: V. H. Nguyen "Approximation algorithm for the minimum directed tree cover", LNCS, Vol 6509, pp. 144–159, Springer-Verlag, (2010).*

- A combinatorial approximation algorithm for Asymmetric Prize Collecting Traveling Salesman Problem. This algorithm is the first combinatorial approximation algorithm for the problem and in addition, its approximation ratio is $\lceil \log(n) \rceil$ which is equal to the best known approximation ratio (where $n$ denote the number of the vertices).

  *Paper: V. H. Nguyen "A primal-dual approximation algorithm for the Asymmetric Prize-Collecting TSP", J. Comb. Optim. Vol. 25(2), pp. 265-278 (2013).*

**Chapter 2. Natural formulations and algorithms using linear programming.**

- We consider the Ring Star problem which is a network design problem. A ring star network is a network consisting of a ring (backbone network) and the connections of nodes not in the ring to one of the node in the ring (stars). Hence, given a set of site to be connected by a ring star network, we have to determine the subset of the sites forming the ring and the assignments of the remaining sites to the ring. The objective is to minimize the cost of ring (connection cost) + the assignment costs.

We propose for this problem a new integer formulation based on the *st*-path polytope which can present advantages comparing with the existing formulation based on cycle polytope.

*Paper: S. Kedad Sidhoum, V.H. Nguyen "An Exact Algorithm for Solving the Ring Star Problem", Optimization, vol. 59(1), pp. 125-140, (2010).*

- Given a directed graph $G$ whose arcs are weighted, similarly as tree directed covers, a directed cycle cover is a cycle whose vertices form a vertex cover of $G$. We give a 0/1 linear formulation for the minimum weight directed cycle cover problem. Based on this formulation, we derive an approximation algorithm with $\lceil 2\log_2(n) \rceil$ as approximation ratio (where $n$ denote the number of the vertices in $G$).

  *Paper: V.H. Nguyen : "Approximating the minimum tour cover of a digraph", Algorithms, vol. 4 (2), pp. 75-86, (2011).*

**Chapter 3. Extended formulations: compactness and separation.**

- Given an undirected graph $G$, a substar in $G$ is connected subgraph of diameter at most 2. We study the linear description of the substar polytope, i.e. the convex hull of the incidence vectors of the substars in $G$. The substars are related with the necessary colors for coloring the edges of $G$ so that two incident edges receive two different colors. In such a coloration, the edges of the same color form a matching. When $G$ is a bipartite graph, Konig's theorem tells us that the number of necessary colors is equal to the maximum degree of the vertices. Equivalently, we can find $\Delta(G)$ disjoint matchings which covers the edges of the graph. We give here a complete description of the substars polytope which give a generalized version of Konig's theorem for all graphs. We also give a compact extended formulation for the substars polytope.

  *Paper: D. Cornaz, V. H. Nguyen "Konig's edge colouring theorem for all graphs" Operations Research Letters, Vol 41, No 6, pp.592-596 (2013).*

- In the second part, we are interested in the star forests, i.e. the collection of disjoint substars in $G$. This structure is strongly related to the dominating sets and the edge dominating sets in $G$. In particular, "the complements" (taken in some sense) of a star forest are respectively a (vertex) dominating set and a edge dominating set. We study the star forest polytope which is the convex hull of the incidence vectors of the star forests. We show that in the case where the graph $G$ is a tree, there exists a very simple extended formulation whose projection on natural space gives a complex class of facet-defining inequalities. The inequalities in this class are valid for the star forest polytope in the general case where $G$ is any graph and under certain conditions they define facets. We show how to solve the separation problem for this class of valid inequalities by exploiting the extended formulation.

  *Paper: M. Aider, L. Aoudia, M. Baiou, A.R. Mahjoub and V.H. Nguyen "On the star forest polytope for trees and cycles", submitted to RAIRO-Operations Research*

- We are interested in the maxcut polytope which is associated with the maxcut problem. There exists a well known 0/1 linear formulation of this polytope by the triangle inequalities. These inequalities describe completely the maxcut polytope when $G$ is planar or more generally when $G$ does not accept $K_5$ as a minor [8]. The triangle inequalities are expressed over all the vertex triplets of $G$, hence their number is $O(n^3)$ (where $n$ denote the number of the vertices) independently from the value of $m$ the number of edges in $G$. Consequently, we have the same 0/1 formulation of maxcut polytope for all the graphs of order $n$ sparse or dense. We show that one can obtain an equivalent 0/1 formulation for maxcut using only $O(nm)$ triangle inequalities and this equivalence is also true for the linear programming relaxations. Moreover, we show that in the case of series-parallel graphs, we can obtain a linear size complete formulation for the maxcut polytope which contains only $O(n)$ triangle inequalities.

*Papers: V.H. Nguyen, M. Minoux, D.P. Nguyen "Improved compact formulations for metric and cut polyhedra", Electronic Notes in Discrete Mathematics, Vol. 52, pp. 125-132 (2016).*

*V.H. Nguyen, M. Minoux and D.P. Nguyen "Reduced-size formulations for metric and cut polyhedra in sparse graphs", submitted to Networks.*

*D.P. Nguyen, M. Minoux , V.H. Nguyen, T.H. Nguyen and R. Sirdey "Improved compact formulations for graph partitioning in sparse graphs" to appear in Discrete Optimization.*

**Chapter 4. Extended formulations: linearization and convexification.**

In this first two sections of this chapter, we consider a Graph Partitioning problem under Capacity Constraint (GPCC) where the clusters forming the partition have to satisfy some capacity constraint. For example, in the designing of optical rings networks (norm SONET/SDH):

> The vertices of the graph represent the sites to be connected and the edges represent the traffic volume between two specific sites. In this problem, the vertices should be partitioned into clusters and the capacity constraints express the fact that the total traffic incident to the vertices of each cluster is bounded by a constant.

We consider the Node-Node model for GPCC which is the non-convex 0/1 formulation consisting of triangle constraints and quadratic capacity constraints. The linearization or convexification of the latters is thus necessary for the solution of GPCC. Classically, the linearization results in an extended formulation with additional variables representing the product of two binary variables. In GPCC, this technique increases the number of variables from $O(n^2)$ to $O(n^3)$ (where $n$ denote the number of the vertices) which is considerable already for the values of $n$ approaching 100. To overcome this difficulty, in this chapter, we study linearization methods that keep the number of variables at $O(n^2)$. We call, *more compact linearized model*, the model produced by theses methods as constrast to the model produced by the classical linearization method. More

precisely,

- in the first section, we propose a new technique of linearization for the Node-Node model by projecting/cut generating which allows to stay on the space of original variables and to generate violated inequalities if the current solution is outside the projection of the extended formulation (the one obtained by classical linearization). We show by numerical results that the proposed methods of linearization outperform the classical linearization in branch-and-bound algorithms.

  *Paper: P. Bonami, V. H. Nguyen, M. Klein et M. Minoux "On the Solution of a Graph Partitioning Problem under Capacity Constraints", LNCS, Vol 7422, pp. 285-296, (2012).*

- In the second section, we consider a variant of the graph partitioning problem involving knapsack constraints with Gaussian random coefficients. We call the problem, SGP, for Stochastic Graph Partioning. Under this assumption of probability distribution, SGP can be traditionally formulated as a binary SOCP for which the continuous relaxation is convex. In this section, we reformulate SGP as a binary quadratic constrained program for which the continuous relaxation is not necessarily convex. We propose then a more compact linearized model of SGP obtained by using the bilinear linearization technique. Numerical results show that branch-and-bound algorithms using this more compact linearized model as continuous relaxations outperform the ones using SOCP or classical linearization of SGP as continuous relaxation.

**PhD and Master thesis supervision and related publications** I have advised 11 Master thesis. I have also co-advised 4 PhD thesis:

- Minh Tuan Ngo (co-advised with M. Minoux ) "Modeling and Optimizing

for traffic lights - Application of Bender's decomposition method" defended in October 2010 at the University Pierre and Marie Curie, Paris.

- Thanh Hai Nguyen (co-advised with J.-F. Maurras) "Convex hull of the finite Huffman codes" defended in December 2010 at the University of Mediterranean Sea, Marseille. *Paper: J.-F. Maurras, T. H. Nguyen et V. H. Nguyen "On the linear description of the Huffman trees polytope" Discrete Applied Mathematics, Vol. 164:225-236 (2014).*

- Lamia Aoudia (co-advised with M. Aider) "Connectivity in combinatorial polyhedra", in preparation. *Paper: M. Aider, L. Aoudia, M. Baiou, A.R. Mahjoub and V.H. Nguyen "On the star forest polytope for trees and cycles", submitted to RAIRO-Operations Research*

- Dang Phuong Nguyen (co-advised with M. Minoux, R. Sirdey) "Graph partitioning under non linear and stochastic constraints" defended in December 2016. *Paper: D.P. Nguyen, M. Minoux , V.H. Nguyen, T.H. Nguyen and R. Sirdey "Improved compact formulations for graph partitioning in sparse graphs" to appear in Discrete Optimization.*
  *D.P. Nguyen, M. Minoux , V.H. Nguyen, T.H. Nguyen and R. Sirdey "Stochastic Graph Partitioning : Quadratic versus SOCP formulations", Optimization Letters, vol. 10(7), pp. 1505-1518 (2016).*

**Notations.**

Let us define below the notations which will be used in the manuscript.

**Undirected graphs.** We consider the undirected graph $G = (V, E)$ where $V$ denotes the node set and $E$ denote the edge set. The edges in $E$ are weighted by a cost vector $c \in \mathbb{Q}^E$ which is not restricted to be nonnegative. Let $ij$ denote the edge between two nodes $i$ and $j$ in $G$. The notation is symmetric, i.e. $ji$ denotes the same edge as $ij$. For $S \subset V$, let

$$\delta(S) = \{ij \in E \mid \text{where } i \in S \text{ and } j \in V \setminus S,\}$$

be the cut associated to $S$ in $G$. Let $E(S)$ denote the set of the edges with both two end-nodes in $S$. Let $\mathbb{Q}^E$ be the rational space of dimension $|E|$ with the coordinates indexed by the edge set $E$. For a vector $x \in \mathbb{R}^E$ and a subset $F \subset E$, let $x(F)$ denote the sum $\sum_{e \in F} x_e$.

**Directed graphs.** Let $G = (V, A)$ be a digraph with vertex set $V$ and arc set $A$. If $x \in \mathbb{Q}^{|A|}$ is a vector indexed by the arc set $A$ and $F \subseteq A$ is a subset of arcs, we use $x(F)$ to denote the sum of values of $x$ on the arcs in $F$, $x(F) = \sum_{e \in F} x_e$. Similarly, for a vector $y \in \mathbb{Q}^{|V|}$ indexed by the vertices and $S \subseteq V$ is a subset of vertices, $y(S)$ denotes the sum of values of $y$ on the vertices in the set $S$. For a subset of vertices $S \subseteq V$, let $A(S)$ denote the set of the arcs having both end-nodes in $S$. Let $\delta^+(S)$ and $\delta^-(S)$ denote the set of the arcs having only the tail and head in $S$, respectively. Let $\delta(S) = \delta^+(S) \cup \delta^-(S)$. We will call $\delta^+(S)$ the *outgoing cut* associated to $S$, $\delta^-(S)$ the *ingoing cut* associated to $S$ and $\delta(S)$ simply the cut associated to $S$. For each vertex $v \in V$, let $d^+(v)$ be the *out-degree* of $v$ which is equal to the number of arcs in $G$ having $v$ as tail and let $d^-(v)$ be the *in-degree* of $v$ which is equal to the number of arcs in $G$ having $v$ as head. For two subset $U, W \subset V$ such that $U \cap W = \emptyset$, let $(U : W)$ be the set of the arcs having the tail in $U$ and the head in $W$. For $u \in V$, we specify $v$ as an *outneighbor* of $u$ if $(u, v) \in A$ and as an *inneighbor* of $u$ if $(v, u) \in A$. For any $u, v \in V$, let $p(u, v)$ be the number of arc-disjoint paths from $u$ to $v$. For the sake of simplicity, in clear contexts, the singleton $\{u\}$ will be denoted simply by $u$. When we work on more than one graph, we specify the graph in the index of the notation, e.g., $\delta^+_G(S)$ will denote $\delta^+(S)$ in the graph $G$.

# Chapter 1

# Natural formulations and combinatorial algorithm

## 1.1 Introduction

In this chapter, we will discuss methods for exploiting complete or partial linear formulations for combinatorial optimization problem.

In the first part of the chapter, we are interested in the following case: Given an "easy" problem A for that some complete linear formulation is known and also polynomial time algorithms are known. However, these algorithms are not direct in the sense that one should transform the problem into an instance of some other known "easy" problem B and solve the latter by some algorithm properly designed for problem B. Our aim is to design a direct algorithm for problem A using the knwon linear formulation and prove that it may allow to improve the time complexity of problem A. To illustrate this idea, we consider *the problem of detecting a negative cost cycle in a (weighted) undirected graph* (UNCCD). This problem can be solved in polynomial time by transforming it into an instance of *the minimum weight $T$-join* or *the maximum weight degree constraint problem*. The UNCCD problem can be viewed as the optimization problem associated with the circuit cone which has a complete linear formulation given by Seymour [58]. We present a dual algorithm based on the formulation given by Seymour and improve the time complexity of the UNCCD problem

from $O(\min(n^3, mn\log(n))$ to $O(mn + n^2\log(n))$.

In the second part of the chapter, we aim at designing combinatorial approximation algorithm for some $NP$-hard problem called A. Our idea is to decompose the solutions of A into substructures obtained by relaxing some constraints in A. Suppose that complete linear formulations for the convex hull of these substructures are known and moreover primal-dual algorithms (for solving the associated optimization problem) based on these formulations exist. As the union of the inequalities issued from these linear formulations form a (partial) linear formulation $P(A)$ for A, applying successively in some order (that is to be determined) the primal-dual algorithms for substructures may constitute a primal-dual (approximation) algorithm for A. More precisely, let $D(A)$ be the dual of $P(A)$, we design a primal-dual approximation algorithm based on $P(A)$ by applying successively $k$ primal-dual algorithms corresponding to substructures. In this application, the successor algorithm will work on the reduced cost left by the predecessor. At the end, we should obtain a feasible dual solution $y$ and there should be a subset of the variables of zero reduced cost that constitutes a solution called $T$ for the problem A. We will call *sucessive primal-dual approximation algorithms*, the algorithms applying this principle. Let us see how to compute the approximation ratio of such a successive approximation algorithm. Because of the primal-dual nature of the algorithms, we also obtain $k$ feasible dual solutions $y_1, \ldots, y_k$ where $y_i$ is the current dual feasible solution after the $i$-th application of primal-dual algorithms. Let $y = y_k$ the last and best dual feasible solution. Let $T_i$ be the subset of the variables (of zero reduced cost) in $T$ that was added to $T$ after the $i$-th application of primal-dual algorithms. We suggest two following methods for analyzing the approximation ratio.

- For every $1 \leq i \leq k$, we show that the cost of $T_i$ is at most $\alpha_i$ times the cost of $y_i$. This results that the cost of $T$ is at most $\sum_{i=1}^{k} \alpha_i$ times the cost of the optimal solution for problem A. We apply this principle for the Asymmetric Prize Collecting TSP in Section **??**.

- For each variable $v$ in $T$, we decompose the cost of $v$ into $k$ portions $z_i^v$ corre-

sponding to the value of dual variables active in the $i$-th application and involving in the constraint associated with $v$ in $D(A)$. We show that $\sum_{v \text{ variable in } P(A)} z_i^v$ is at most $\alpha_i$ times the cost of $y$. The overall approximation ratio will be then $\max_{i=1,\ldots,k} \alpha_i$. We apply this principle for the Minimum Directed Tree Cover Problem in Section 2.2.

## 1.2 Detection of negative cost cycle in undirected graphs

In this section, we consider the problem of checking whether an undirected $G = (V, E)$ whose edges are weighted by an arbitrary cost vector $c \in \mathbb{Q}^E$, contains a negative cost cycle (UNCCD) problem. The UNCCD problem has been extensively studied in the past, especially in the 1980s. There are several approach for solving the UNCCD problem:

**Reduction to the the maximum weight degree constraint subgraph problem.**

The maximum weight degree constraint subgraph problem consists in finding a maximum weight subgraph under degree constraints (lower bound and upper bound constraints) on each nodes. The UNCCD problem on $G$ can be formulated as a maximum weight degree constraint problem in a new graph $G'$ by setting the lower bound and upper bound on the degree of every node to exactly 2, negating the weights of all the edges in $G$ and finally adding self-loops of weight 0 to all nodes. An optimal solution of this problem is a maximum weight cycle cover over the nodes of $G'$. We can see that if all cycles in the original graph were positive, then a maximum weight cycle cover over the nodes consists of just the 0-weight self-loops. Otherwise, if there is a negative cycle, then this cycle is of positive weight in the new graph $G'$ and any maximum weight cycle cover is of positive weight. In [30], Gabow gives an algorithm for the maximum weight degree constraint subgraph problem which can find a maximum weight cycle cover in $G'$ in $O(\min(n^3, mn \log n))$ time. Hence, we can detect a negative

cycle in $G$ in $O\min(n^3, mn\log n)$ time.

Note that Gabow's algorithm for maximum weight cycle cover is not neither a direct algorithm on $G'$. It consists in reducing the maximum weight cycle cover problem to a maximum weight matching problem in a graph $G''$ which is built as follows. Let $W$ be the maximum absolute value of an edge weight in $G'$. For every node $v$ of degree $d(v)$ in $G'$, create in $G''$ two special nodes $v_1$ and $v_2$, $d(v)$ nodes $l_1^v, \ldots l_{d(v)}^v$ and $d(v)$ nodes $r_1^v, \ldots r_{d(v)}^v$. The nodes $v_1$ and $v_2$ are respectively connected to each of the nodes $l_i^v$ for $i = 1, \ldots, d(v)$ by an edge of weight $mW$. There is an edge of weight $mW$ between the nodes $l_i^v$ and $r_i^v$ for $i = 1, \ldots, d(v)$. Finally, if $uv$ is an edge in $G'$ of weight $c_{uv}$ such that $v$ is the $i^{\text{th}}$ neighbour of $u$ and $v$ is the $j^{\text{th}}$ neighbor of $u$ then there is an edge between $r_j^v$ and $r_i^u$ of weight $W + c_{uv}$ in $G''$. The graph $G''$ has $O(m)$ vertices and edges and any maximum weight cycle cover in $G'$ can be converted to a perfect matching in $G''$. In particular, the maximum weight perfect matching in $G''$ has weight $2m^2W + nW +$ maximum weight of cycle covers in $G'$. In [30], Gabow designs an $O(\min(n^3, mn\log n))$ time algorithm to find such a maximum weight perfect matching in $G''$.

**Reduction to the minimum weight $T$-join problem.** Let $T \subseteq V$ be a vertex subset of even cardinality. An edge subset $J \subseteq E$ is a $T$-join if $T$ is exactly the set of odd degree vertices of the subgraph induced by $J$. When $T = \emptyset$, a $\emptyset$-join is a collection of cycles in $G$. Hence, we can solve the UNCCD problem by finding a minimum weight $\emptyset$-join in $G$. For this, one needs to invoke one execution of an algorithm for solving the Undirected All Pairs Shortest Paths (UAPSP) problem and subsequently one execution of an algorithm for finding a minimum weight perfect matching on a metric (complete graph with $n$ vertices and $O(n^2)$ edges). As the best complexity time algorithm for UAPSP given by Gabow [30] runs in $O(\min(n^3, mn\log n))$ time and Gabow's algorithm[31] finds minimum weight perfect matching in complete graphs in $O(n^3)$ time, solving UNCCD problem by this approach will run in $O(n^3)$.

In [51], we present a dual algorithm for solving the UNCCD problem based on a polyhedral description given by Seymour [58]. It works directly on $G$ and improves the time complexity for UNCCD problem to $O(mn + n^2 \log(n))$.

Let $C$ be any cycle in $G$, let $\chi(C) \in \mathbb{Q}^E$ be the incidence vector associated to $C$ defined as follows:

$$\chi(C)_e = \begin{cases} 1 & \text{if } e \in C \\ 0 & \text{otherwise} \end{cases}$$

Let $S \subset V$ be any vertex subset and let $e$ be any edge belonging to $\delta(S)$, we can observe that any cycle taking the edge $e$ must crossing at least some other edge in $\delta(S)$. Then the following *circuit inequality*

$$x(\delta(S) \setminus \{e\}) - x_e \geq 0 \text{ for all } \emptyset \subset S \subset V \tag{1.1}$$

where $x \in \mathbb{Q}^E$ is verified by the incidence vector of every cycle in $G$. These inequalities were introduced by Seymour [58]. He proved the following theorem.

**Theorem 1.2.1** *[58] The inequalities* (1.1) *along with the non-negativity inequalities characterize completely the circuit cone of $G$ which is the cone of the incidence vectors of all the circuits (Seymour calls circuit a cycle in an undirected graph) in $G$.*

Let us consider the following linear programming problem which minimizes the cost function $c$ over the system consisting of the inequalities (1.1) and the non-negativity inequalities:

$$\begin{aligned} &(\mathrm{P}) \quad \min c^T x \\ x(\delta(S) \setminus \{e\}) - x_e \quad &\geq \quad 0 \text{ for all } \emptyset \subset S \subset V \text{ and } e \in \delta(S), \qquad (1.2) \\ x_e \quad &\geq \quad 0 \text{ for all } e \in E. \qquad (1.3) \end{aligned}$$

For any subset $S \subset V$ and any edge $e \in \delta(S)$, let $y_{S,e}$ be the dual variable associated

to the inequality (1.1) associated to $S$. Then the dual of $(P)$ can be written as follows:

$$\begin{aligned} & & & & & & (\mathrm{D}) \quad \max\ 0 & \\ \sum_{\substack{S\subset V \text{s.t.} \\ e\in\delta(S)}} \sum_{f\in\delta(S)\setminus\{e\}} y_{S,f} - \sum_{\substack{S\subset V \text{s.t.} \\ e\in\delta(S)}} & y_{S,e} & \leq c_e & \quad e\in E, & & (1.4) \\ & y_{S,e} & \geq 0 & \quad S\subset V \text{ and } e\in\delta(S). & & (1.5) \end{aligned}$$

Hence, by the theory of linear programming and by Theorem 1.2.1, we have the following corollary.

**Corollary 1.2.2** *If all the cycles of $G$ are non-negative with respect to $c$ then $(P)$ and $(D)$ have optimal solutions of value zero and any feasible solution of $(D)$ is optimal. Otherwise, i.e. there exists a negative cost cycle in $G$, $(P)$ is unbounded and $(D)$ is infeasible.*

In [51], we present an algorithm that outputs either a dual feasible solution of $(D)$ or a negative cost cycle in $G$. The algorithm starts with the set of dual multipliers for $(P)$ that are all null, i.e. $y = \mathbf{0}$ with $\mathbf{0}$ is the null vector in the real space of suitable dimension. For each edge $e \in E$, we define its reduced cost

$$\bar{c}_e = c_e - \sum_{\substack{S\subset V \text{s.t.} \\ e\in\delta(S)}} \sum_{f\in\delta(S)\setminus\{e\}} y_{S,f} + \sum_{\substack{S\subset V \text{s.t.} \\ e\in\delta(S)}} y_{S,e}.$$

For every edge $e \in E$, we say that $e$ is *correct* if $\bar{c}_e \geq 0$, otherwise $e$ is incorrect. Let us use $\bar{c}_e$ to denote *the degree of correctness* of $e$.

**Remark 1.2.3** *At the beginning, the reduced cost is equal to the original cost for all edges, the negative edges are incorrect and $y = \mathbf{0}$ is an infeasible solution for $(D)$*

The principle of the algorithm is to iteratively make $y$ feasible for $(D)$ and in the case of failure, i.e. $(D)$ is infeasible, it exhibits a negative cycle. For this, it will iteratively correct the negative (hence incorrect) edges in $G$ by changing $y$ (from the initial null vector) until

- either all negative edges are corrected, i.e. $y$ becomes dual feasible,

- or it cannot incorrect edges any more and exhibits a negative cycle.

The following lemma suggests how to correct an incorrect edge $e$,

**Lemma 1.2.4** *Given an incorrect edge $e$ and a subset $S \subset V$ such that $e \in \delta(S)$, if we increase value of the dual variable $y_{S,e}$ then the reduced cost of $e$ increases and the reduced cost of the other edges in $\delta(S) \setminus \{e\}$ decreases, i.e. the degree of correctness of $e$ increases and the degree of correctness of the other edges in $\delta(S) \setminus \{e\}$ decreases.*

**Proof** By definition, we have $\bar{c}_e = c_e + y_{S,e} + \ldots - \ldots$, hence increasing $y_{S,e}$ increases $\bar{c}_e$. For all $f \in \delta(S) \setminus \{e\}$, $\bar{c}_f = c_f - y_{S,e} - \ldots + \ldots$, hence increasing $y_{S,e}$ decreases $\bar{c}_f$.

The algorithm works as the same way as the Edmonds' blossom algorithm for the minimum perfect matching problem.

### 1.2.1 Cycles, matchings and $T$-joins

Note that the cycles and the perfect matchings are special case of $T$-joins with $T \subseteq V$ of even cardinality. A cycle is $\emptyset$-join and a perfect matching is a $V$-join. Given a $T \subseteq V$, let $T$-join polytope be the convex hull of the incidence vector of all the $T$-join in $G$. The following linear system defines the $T$-join polytope.

$$x(\delta(U) \setminus F) - x(F) \geq 1 - |F| \; U \subseteq V,\, F \subseteq \delta(U),\, |U \cap T| + |F| \text{ odd}, \quad (1.6)$$

$$0 \leq x_e \leq 1 \qquad e \in E. \quad (1.7)$$

Let us call, the perfect matching polytope, the up hull of the incidence vector of the perfect matchings in $G$.

$$x(\delta(U)) \geq 1 \qquad U \subset V \text{ with } |U| \text{ odd}, \quad (1.8)$$

$$x_e \geq 0 \qquad e \in E \quad (1.9)$$

As cycles are $\emptyset$-join, inequalities (1.2) are special cases of inequalities (1.6) when $T = \emptyset$, $U$ is any subset of $V$ and $F$ is any subset of $\delta(U)$ of cardinality 1. As perfect

matchings are $V$-joins, inequalities (1.8) are also special cases of inequalities (1.6) when $T = V$, $U$ is any odd subset of $V$ and $F$ is the emptyset.

### 1.2.2 Direct algorithm for UNCCD

**Recall of Edmonds' blossom algorithm for minimum perfect matching**

A *blossom* $U$ is a node subset of odd cardinality by which the induced subgraph is either 2-connected or a single node. Let $y_U$ denote the dual variable associated with inequality (1.8) involving $U$. For each edge $e \in E$, we define the reduced cost

$$\bar{c}_e = c_e - \sum_{\substack{U \subset V, |U| odd \\ e \in \delta(U)}} y_U.$$

In fact, the blossom algorithm is a primal-dual algorithm in which only inequalities (1.8) involving a blossom $U$ could be active, i.e., $y_U$ could be strictly positive. The blossom algorithm operates on two graphs derived from $G$:

- the graph $G'$ which is initially equal to $G$ and can evolve to a multi graph with pseudo-nodes created by the contraction of blossoms.
- the graph $G^0$ which is the subgraph of $G'$ induced by the edges of zero reduced cost.

The algorithm iteratively builds a perfect matching $M$. Let us recall the main stages of Edmonds' Blossom algorithm.

**Step 1.** Choose any $M$-exposed node $r$ to be the root of the alternating tree $T$ to be built.

**Step 2.** Check each edge $vw$ of zero reduced cost with at least one node, say $v \in Even(T)$.

- if $w \in Even(T)$ then shrink the odd cycle (i.e., a blossom) in $T$ containing $v$ and $w$,

- if $w$ is a $M$-exposed node, the path between $v$ and $w$ in $T$ is an augmenting path. Using the latter to augment $M$ and goto Step 1.
- if $w \in Odd(T)$ and $y(\{w\}) = 0$ then $w$ should be a pseudo-node. Deshrink $w$.
- if $w \in Odd(T)$ and $y(\{w\}) > 0$ then extend $T$ with $vw$.
- or find an augmenting path. Goto Step 4.

**Step 3.** Using $T$ for a dual adjustement and go to Step 2.

**Step 4.** Using the augmenting path to augment $M$. If $M$ is a perfect matching then output $M$ and stop. Otherwise, goto Step 1.

Thus, between two augmentations of the matching, one has to perform *a search* (by repeating Step 2. and Step 3.) which consists in doing repeatedly four operations:

- extending the alternating tree $T$ with an edge,
- shrinking an odd cycle to pseudo-node,
- deshrinking a pseudo-node with which the associated dual variable has the value decreased to 0,
- and dual adjustements.

On can prove that there are $O(n)$ such operations in a search. Gabow [31] has shown that one can perform a search in $O(m + n\log(n))$. Since there are $\frac{n}{2}$ matching augmentations, the blossom algorithm can be implemented with $O(mn + n^2\log(n)$ running time.

### Preprocessing

We can suppose that the subgraph of $G$ induced by the negative cost edges is a forest since otherwise we can detect easily a cycles containing uniquely negative cost edges. Hence, our algorithm work with a collection of negative trees.

**Equivalence of "blossom sets" in UNCCD problem**

Equivalence of "blossom sets" in UNCCD are the subsets $U \subset V$ with which associated inequalities (1.2) could be active, i.e. the associated dual algorithm could be strictly positive in our algorithm. These subsets, called *negative leaf sets*, are the subsets $U \subset V$ such that the subgraph induced by $U$ in $G$ is connected and there are exactly one negative reduced cost edge $e_U$ in $\delta(U)$. Moreover, among the inequalities (1.2) associated to $U$ (there are possibly many), only the inequality associated with $U$ and the edge $e_U$ could be active, i.e., the dual variable $y_{U,e_U}$ could be strictly positive in our algorithm. Hence, there is a one-to-one correspondance between a negative leaf $U$ and the associated inequality (1.2) that could be active. Thus, in the algorithm, when we have chosen a negative leaf set $U$, the dual variable to be increased is $y_{U,e_U}$ with $e_U$ is the unique negative reduced cost edge in $\delta(U)$.

We have also the same notion of augmentation as in the matching algorithm. An augmentation in our case is when the reduced cost of a (original) negative edge becomes zero. As the original negative edges form a forest, as in the matching algorithm, we have $O(n)$ augmentations. Hence, our algorithm is similar to the matching algorithm at the following points:

Step 1. At each iteration, we choose a node $r$ of $G'$ (the multigraph created by the shrinking and deshrinking operations in $G$) such that $\delta_{G'}(r)$ contains exactly one negative reduced cost edge. This is to the choice of an $M$-exposed node in the matching algorithm.

Step 2. Doing a search as in the matching algorithm (with blossom sets are replaced by negative leaf sets) until the condition for changing the root of the alternating tree is verified

Step 3. If all the edges have non-negative reduced cost then stop (the graph has no negative cost cycle). Otherwise, goto Step 1.

The difference between our algorithm and the matching algorithm is in the condition in Step 2. for changing the root of the alternating tree. In the matching algorithm, this

condition is verified if there is a matching augmentation. In the UNCCD problem, this condition is more complex, in order to ensure that the (pseudo)-nodes in $Even(T)$ (the dual variable associated to these nodes would be increased by dual adjustement) are negative leaf sets, sometimes we will have to change the root without augmentation. In [51], we show that in spite of this complexity the total number of changings root of the alternating tree is at most $O(n)$. Hence, the time complexity of our algorithm for UNCCD is $n$ times the complexity of a search, i.e. $O(mn + n^2 \log(n)$. Note that if $G$ contains a negative cost cycle then it will be detected during a search.

## 1.3 Approximation algorithm for Minimum Weight Directed Tree Cover

### 1.3.1 Introduction

Let $G = (V, A)$ be a directed graph with a (non negative) cost function $c : A \Rightarrow \mathbb{Q}_+$ defined on the arcs. Let $c(u, v)$ denote the cost of the arc $(u, v) \in A$. A *directed tree* cover is a weakly connected subgraph $T = (U, F)$ such that

1. for every $e \in A$, $F$ contains an arc $f$ intersecting $e$, i.e. $f$ and $e$ have a end vertex in common.

2. $T$ is a branching, i.e. an arborescence rooted in some node $r \in U$. For any node $u \in r$, there is an unique path from $r$ to $u$ in $T$.

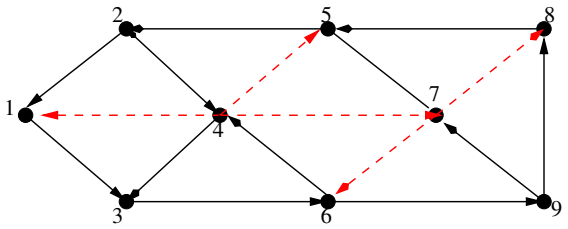


Figure 1-1: Example of a directed tree cover rooted in node 4 (red dashed line).

The *minimum directed tree cover problem* (DTCP) is to find a directed tree cover of minimum cost. In [52], we show that DTCP is $NP$-hard by proving that the minimum

weighted set cover is a special case of DTCP. To the best of our knowledge, there is no known approximation algorithm for DTCP. In the prospective section of [41] and [29] where the authors consider the undirected version of DTCP, they presented DTCP as a wide open problem for further research on the topic. In particular, Fujito [29] pointed out that his approach for TCP can be extended to give a 2-approximation algorithm for the unweighted case of DTCP but falls short once arbitrary costs are allowed.

In [52], we give the first logarithmic factor approximation algorithm for DTCP. We consider DTCP with a fixed root $r$, i.e. we find a minimum tree cover rooted at a specific node $r$. In particular, let $D^+$ be the maximum outgoing degree of the nodes in $G$, we design a primal-dual $\max(2, \ln(D^+))$-approximation algorithm for $r$-DTCP. Repeating $n$ times this algorithm yields obviously a $\max(2, \ln(D^+))$-approximation algorithm for DTCP which is thus somewhat best possible.

### 1.3.2 Integer programming formulation for $r$-DTCP

We use a formulation inspired from the one in [42] designed originally for the TCP. The formulation is as follows: for a fixed root $r$, define $\mathcal{F}$ to be the set of all subsets $S$ of $V \setminus \{r\}$ such that $S$ induces at least one arc of $A$,

$$\mathcal{F} = \{S \subseteq V \setminus \{r\} \mid A(S) \neq \emptyset\}.$$

Let $T$ be the arc set of a directed tree cover of $G$ containing $r$, $T$ is thus a branching rooted at $r$. Now for every $S \in \mathcal{F}$, at least one node, saying $v$, in $S$ should belong to $V(T)$. By definition of directed tree cover there is a path from $r$ to $v$ in $T$ and as $r \notin S$, this path should contain at least one arc in $\delta^-(S)$. This allows us to derive the following *cut* constraint which is valid for the DTCP:

$$\sum_{e \in \delta^-(S)} x_e \geq 1 \text{ for all } S \in \mathcal{F}$$

This leads to the following IP formulation for the minimum $r$-branching cover.

$$\min \sum_{e \in A} c(e) x_e$$

$$\sum_{e \in \delta^-(S)} x_e \geq 1 \text{ for all } S \in \mathcal{F}$$

$$x \in \{0,1\}^A.$$

Replacing the integrity constraints by

$$x \geq 0,$$

we obtain the linear programming relaxation, denoted by $DTC(\mathrm{G})$. We express below the dual of $DTC(\mathrm{G})$:

$$\max \sum_{S \in \mathcal{F}} y_S$$

$$\sum_{S \in \mathcal{F} \text{ s.t. } e \in \delta^-(S)} y_S \leq c(e) \text{ for all } e \in A$$

$$y_S \geq 0 \text{ for all } S \in \mathcal{F}$$

**Algorithm overview**

We sketch below a successive primal-dual approximation algorithm for $r$-DTCP. This algorithm begins with the zero dual feasible solution where $y_S = 0$ for all $S \in \mathcal{F}$. At each iteration of the algorithm, the dual variable $y_S$ associated to some $S \in \mathcal{F}$ will be increased until the reduced cost of some arc in $\delta^-(S)$ becomes 0. When this happends, we say that the subset $S$ is *covered*. During the algorithm, we keep a subset, denoted by $T_0$, of the set of the arcs of zero reduced cost that will grow progressively and contain a directed tree cover rooted at $r$ at the end. At initialization $T_0 = \emptyset$. For a node $u$ of $G$, we say that $u$ is *reachable* from $r$ if there is a path from $r$ to $u$ in $T_0$. The algorithm contains three phases,

- It determines first a node cover $U$ in Phase I by covering the sets $S \in \mathcal{F}$ such

that $|S| = 2$. Phase I outputs the set $A_0$ of the arcs of zero reduced cost. It outputs also a dual feasible solution $y$.

- Phase II works with the reduced costs issued from Phase I. In this phase , we determine in this phase a collection $\mathcal{S} \subset \mathcal{F}$ of node subsets in $\mathcal{F}$ to be covered in order to make all the nodes $v$ in $U$, either reachable from $r$ or connected to a strongly connected component of $A_0$ ($v$ is connected to a strong connected $B$ of $T_0$ if for every node $w \in B$, there is a path from $w$ to $v$ in $A_0$). We show that the covering of the node subsets in $\mathcal{S}$ can be done by solving an instance of Set Cover problem by the Chvatal's greedy algorithm. Phase II outputs the set of the zero reduced cost arcs $A_0$ and a dual feasible solution $y$ growing from the dual feasible solution given by Phase I.

- Phase III is executed only if $A_0$ does not contain a $r$-branching covering $U$, i.e. there still exists strong connected components of $A_0$ not reachable from $r$ such that some node $v \in U$ is connected to. We call such a components *a Edmonds subgraph*. Phase III makes the Edmons subgraphs reachable from $r$ by growing its associated dual variable. Finally, it extracts from $A_0$, the set of zero reduced cost arcs, a $r$-directed tree cover $T$. Output $T$.

We state now a theorem about the performance guarantee of the algorithm.

**Theorem 1.3.1** *The cost of $T$ is at most* $\max\{2, \ln(D^+)\}$ *times the cost of an optimal $r$-branching cover.*

**Proof** We consider the $r$-branching cover $T$ and the dual feasible solution $y$ output by the algorithm. We call every arc $e \in T$, *a choen arc* and every node subset $S \in \mathcal{F}$ such that $y_S > 0$, *an active subset*. For every chosen arc $e$, we divide the cost of $e$ into three parts: $c_i(e)$ the part of $c_e$ saturated by dual growing in Phase i for $i = I, II$ and III. We divide the active subsets $S \in \mathcal{F}$ into three parts: $\mathcal{F}_1$ that contains those covered in Phase I, $\mathcal{F}_2$ that contains those covered in Phase II and finally $\mathcal{F}_3$ that contains those covered in Phase III. We then show that for each chosen arc $e$:

- $c_1(e) \leq 2 \sum_{\substack{S \in \mathcal{F}_1 \\ e \in \delta^-(S)}} y_S$,

- $c_2(e) \leq \ln(D_r^+) \sum_{\substack{S \in \mathcal{F}_2 \\ e \in \delta^-(S)}} y_S$,
- $c_3(e) \leq \sum_{\substack{S \in \mathcal{F}_3 \\ e \in \delta^-(S)}} y_S$.

Hence, $c_e = c_1(e) + c_2(e) + c_3(e) \leq \max\{2, \ln(D^+)\} \sum_{\substack{S \in \mathcal{F} \\ e \in \delta^-(S)}} y_S$. This implies that $c(T) = \sum_{e \in T} c_e \leq \max\{2, \ln(D^+)\} \sum_{S \in \mathcal{F}} y_S$.

**Corollary 1.3.2** *We can approximate the DTCP within a* $\max\{2, \ln(D^+)\}$ *ratio.*

## 1.4 Combinatorial Approximation Algorithm for Asymmetric Prize Collecting TSP (N. [53])

### 1.4.1 Introduction

Let $G = (V, A)$ be a complete directed graph with the vertex set $V = \{1, 2, \ldots, n\}$ and the arc set $A$. We associate with each arc $e = (i, j)$ a cost $c_e$ and with each vertex $i \in V$ a nonnegative penalty $\pi_i$. The arc costs are assumed to satisfy the triangle inequality, that is, $c_{(i,j)} \leq c_{(i,k)} + c_{(k,j)}$ for all $i, j, k \in V$. In this paper, we consider a simplified version of the *Asymmetric Prize Collecting Traveling Salesman Problem* (APCTSP), namely, to find a tour that visits a subset of the vertices such that the length of the tour plus the sum of penalties of all vertices not in the tour is as small as possible. Note that in the general version of APCTSP, introduced by Balas [4], the arc costs are not assumed to satisfy the triangle inequality. Furthermore, in [4] associated with each vertex there is a certain reward or prize, and in the optimization problem one must choose a subset of vertices to be visited so that the total reward is at least a given a parameter $W_0$.

For APCTSP, though exact algorithms was developed in [20], there was no work on approximation algorithm until our work [56] in 2012. The work is based on the Held-Karp relaxation and heuristic methods such as the Frieze et al.'s heuristic [28] or the recent Asadpour et al.'s heuristic for the ATSP [2]. Depending on which of the two heuristics is used, it gives respectively $1 + \lceil \log(n) \rceil$ and $3 + 8\frac{\log(n)}{\log(\log(n))}$ as an approximation ratio. In [53], we present a primal-dual $\lceil \log(n) \rceil$-approximation

algorithm for APCTSP. This ratio obviously improves $1 + \lceil \log(n) \rceil$ in theory. It also improves the second ratio in practice since $3+8\frac{\log(n)}{\log(\log(n))}$ is asymptotically better than $\lceil \log(n) \rceil$ but for realistic values of $n$, $3+8\frac{\log(n)}{\log(\log(n))}$ is at least nearly $\frac{3}{2}$ times the value of $\lceil \log(n) \rceil$ (for example when $n = 10^{20}$, $3 + 8\frac{\log(n)}{\log(\log(n))} \approx 90$ and $\lceil \log(n) \rceil = 67$). Moreover, unlike the method in [56], the algorithm repsented in [53] is combinatorial. In this section, we focus on the algorithm presented in [53] which can be viewed as a successive primal-dual algorithm.

### 1.4.2 Integer formulation

Let $G = (V, A)$ be directed graph with $|V| = n$ and $|A| = m$. Each arc $a \in A$ is associated to a cost $c_a$. Each vertex $v \in V$ is associated to a penalty $\pi_v$. The arc cost $c$ is assumed to satisfy the triangle inequality. Our aim is to find a tour $T$ which minimizes $\sum_{a \in T} c_a + \sum_{v \notin T} \pi_v$. We consider the following integer formulation inspired from the undirected version in [10] for APCTSP$_j$, the subproblem of APCTSP when we impose a specific vertex $j$ to be in $T$. For every $i \in V$, for every arc $a \in A$, let

$$y_i = \begin{cases} 1 & \text{if } i \in T \\ 0 & \text{otherwise} \end{cases} \quad \text{and } x_a = \begin{cases} 1 & \text{if } a \in T \\ 0 & \text{otherwise} \end{cases}$$

For every subset $S$, let $\delta^+(S)$ be the set of arcs with tail in $S$ and head in $V \setminus S$ and $\delta^-(S)$ be the set of arcs with head in $S$ and tail in $V \setminus S$. Then APCTSP$_j$ can be

formulated as follows.

$$\min Z_j = \sum_{e \in A} c_e x_e + \sum_{i \in V} \pi_i (1 - y_i)$$

subject to

$$\begin{aligned}
x(\delta^+(S)) &= x(\delta^-(S)) \ \forall S \subset V, && (1.10)\\
x(\delta^+(i)) &= y_i \text{ for all } i \in V, && (1.11)\\
x(\delta^-(i)) &= y_i \text{ for all } i \in V, && (1.12)\\
x(\delta^+(S)) &\geq y_i \ \forall S \subseteq V \setminus \{j\} \text{ and } \forall i \in S, && (1.13)\\
x(\delta^-(S)) &\geq y_i \ \forall S \subseteq V \setminus \{j\} \text{ and } \forall i \in S, && (1.14)\\
y_j &= 1, && (1.15)
\end{aligned}$$

$$0 \leq x_e \leq 1 \text{ and integer}, \quad \text{and} \quad 0 \leq y_i \leq 1 \text{ and integer}.$$

The constraints (1.10) ensure that $T$ is Eulerian. Actually, it is only necessary to impose these constraints to the singletons but it will turn out useful to impose them to all the subsets of $V$ in our primal-dual algorithm. The constraints (1.11) et (1.12) are degree constraints which impose for any node $i$ that the tour must visit $i$ if the latter is included in the tour. The constraints (1.13) and (1.14) are the subtour elimination constraints. Note that for the two family of degree and respectively subtour elimination constraints, one of the two constraints (1.11), (1.12) and respectively (1.13), (1.14) is unnecessary if the Eulerian constraints (1.10) are respected because we have always $x(\delta^+(S)) = x(\delta^-(S))$ for all $S \subset V \setminus \{j\}$. But again we include nevertheless all thse constraints in the formulation because we need both of them in the primal-dual algorithm.

We relax the constraints (1.11) and (1.15) to

$$\begin{aligned}
x(\delta^+(i)) &\geq y_i \text{ for all } i \in V \setminus \{j\},\\
x(\delta^-(i)) &\geq y_i \text{ for all } i \in V \setminus \{j\},\\
x(\delta^+(j)) &\geq 1,\\
x(\delta^-(j)) &\geq 1.
\end{aligned}$$

We can then regroup the first two constraints with constraints (1.13) and (1.14) by allowing $|S| = 1$ in these constraints. Let $C = \sum_{i \in V \setminus \{j\}} \pi_i$ which is a constant, we can then write down the relaxation $(R)$ as follows:

$$
\begin{aligned}
(R) \qquad & \min Z_j = \sum_{e \in A} c_e x_e - \sum_{i \in V \setminus \{j\}} \pi_i y_i + C \\
\text{subject to } x(\delta^+(j)) &\geq 1, && (1.16)\\
x(\delta^-(j)) &\geq 1, && (1.17)\\
x(\delta^+(S)) &\geq y_i,\ \emptyset \neq S \subseteq V \setminus \{j\} \text{ and } \forall i \in S, && (1.18)\\
x(\delta^-(S)) &\geq y_i,\ \emptyset \neq S \subseteq V \setminus \{j\} \text{ and } \forall i \in S, && (1.19)\\
x(\delta^-(S)) - x(\delta^+(S)) &= 0,\ \emptyset \neq S \subset V, && (1.20)\\
y_i &\leq 1,\ i \in V \setminus \{j\} && (1.21)\\
y_i &\geq 0,\ i \in V \setminus \{j\} \\
x_e &\geq 0,\ e \in A
\end{aligned}
$$

Let us introduce the dual variable(s):

- The variables $z_j^+$ associated to the constraint (1.16),
- The variables $z_j^-$ associated to the constraint (1.17),
- The variables $z_{S,i}^+$ associated to the constraints (1.18),
- The variables $z_{S,i}^-$ associated to the constraints (1.19),
- The variables $p_S$ associated to the constraints (1.20),
- and finally, the variables $z_i$ associated to the constraints (1.21).

then the dual program $(D)$ of $(R)$ can be written as follows:

$$
\begin{aligned}
(D)\ \max C + z_j^+ + z_j^- - \sum_{i \in V \setminus \{j\}} z_i & & & \\
\text{subject to} \sum_{\substack{S \subseteq V \setminus \{j\} \\ \text{s.t. } i \in S}} (z_{S,i}^- + z_{S,i}^+) + z_i & \geq & \pi_i \ \forall i \in V \setminus \{j\} & \quad (1.22) \\
\sum_{\substack{S \subseteq V \setminus \{j\} \\ \text{s.t. } e \in \delta^-(S)}} (\sum_{i \in S} z_{S,i}^- + p_S) + \sum_{\substack{S \subseteq V \setminus \{j\} \\ \text{s.t. } e \in \delta^+(S)}} (\sum_{i \in S} z_{S,i}^+ - p_S) & \leq & c_e \ \forall e =\in A \text{ s.t/} & \quad (1.23) \\
z_j^+ - p_j + \sum_{\substack{S \subseteq V \setminus \{j\} \\ \text{s.t. } i \in S}} (\sum_{k \in S} z_{S,k}^- + p_S) & \leq & c_e \ \forall e = (j,i) \in A, & \quad (1.24) \\
z_j^- + p_j + \sum_{\substack{S \subseteq V \setminus \{j\} \\ \text{s.t. } i \in S}} (\sum_{k \in S} z_{S,k}^+ - p_S) & \leq & c_e \ \forall e = (i,j) \in A, & \quad (1.25) \\
z_{S,i}^+, z_{S,i}^- & \geq & 0 \ \forall S \subset V \setminus \{j\} \text{ and } \forall i \in S & \\
z_i & \geq & 0 \ \forall i \in V \setminus \{j\} & \\
z_j^+, z_j^- & \geq & 0 &
\end{aligned}
$$

In the sequel, we will design an approximation algorithm for APCTSP$_j$ based on $(R)$ and $(D)$. An approximation algorithm for APCTSP of the same ratio can be simply deduced from approximating APCTSP$_j$ for each $j \in V$.

### 1.4.3 General Idea of the algorithm

We will present a primal-dual algorithm that has at most $\lceil \log_2(n) \rceil$ iterations of dual augmentation. The algorithm starts with the following feasible dual solution of $(D)$:

- $z_{S,i}^+ = z_{S,i}^- = 0$ for all $i \in V \setminus \{j\}$ and for all $S \subset V \setminus \{j\}$ such that $i \in S$,
- $z_j^+ = z_j^- = 0$.
- $z_i = \pi_i$.

and applies at most $\lceil \log_2(n) \rceil$ dual augmentations. In the algorithm, we maintain an arc subset denoted by $T$ which contains the arcs that will constitute our solution for APCTSP$_j$ at the end of the algorithm. At initialization $T = \emptyset$, and at each iteration,

based on the current dual feasible solution of $(D)$, we add a set of vertex disjoint simple cycles to $T$. Hence, $T$ is always a collection of strongly connected Eulerian components. At each iteration of the algorithm, we consider the graph $\bar{G}$ obtained from $G$ by shrinking the vertex subsets of $G$ corresponding to strongly connected Eulerian components in $T$. We define the cost for the arcs in $\bar{G}$ with respect to the current reduced cost. From $\bar{G}$, we build a bipartite graph $B$ and transform the dual augmenting problem to a minimum cost assignment problem, called $(A)$, in $B$ with respect to the current reduced cost. We consider the classical linear programming formulation for $(A)$ and its dual. In particular, we make the correspondence from each dual variable of $(A)$ to some dual variable of $(D)$. We solve $(A)$ by any known primal-dual algorithm for the minimum cost assignment problem and assign the value of the dual optimal solution of $(A)$ to the corresponding dual variable of $(D)$. Note that each dual variable of $(D)$ will be augmented at most once and after that its value will not be changed until the end of the algorithm. After each iteration of the algorithm, we add to $T$ a set of arcs and may eliminate definitely some vertices from the solution tour $T$. Note that $T$ can contain a multiplicity of an arc and if a vertex $i$ is eliminated from $T$, then all the vertices that belong to the same connected component in the subgraph induced by $T$, will be also eliminated. Thus there will be no arc of $T$ such that one end-vertex is in $T$ and the other has been eliminated from $T$. The algorithm stops when $T$ becomes connected. As at each iteration, for each strongly connected Eulerian component $H$ of $T$, either $H$ is merged with some other strongly connected Eulerian component of $T$, or the vertices in $H$ are eliminated from $T$, at most $\lceil \log_2(n) \rceil$ iterations was performed. We prove that the cost of the arcs added to $T$ in each iteration is at most $C_j^*$ where $C_j^*$ is the optimal value of APCTSP$_j$. In particular, for the last iteration the cost of the arcs added to $T$ plus the total penalty associated to the vertices eliminated from $T$ (from the first iteration to the end) is at most $C_j^*$. As $T$ is Eulerian, $T$ is a solution of APCTSP$_j$ and the cost of $T$ is at most $\lceil \log_2(n) \rceil C_j^*$.

## 1.5 Linear description for the polytope of the Huffman trees (Maurras, Nguyen, N. [46])

In 1952, David Huffman discovered the concept of Huffman tree which offers the most efficient binary code for the characters of an alphabet $\Lambda = \{c_1, c_2, \ldots, c_n\}$ in the context of a language using $\Lambda$ as its alphabet. A Huffman tree is a full binary tree (i.e. every node in the tree has either 0 or 2 children) of $n$ leaves labeled by the characters in $\Lambda$. The *height* of the character $c_i$ ($i$=1,..., $n$), denoted by $l_i$, is the length of the path from the root to the leaf labeld by the character. Given a Huffman tree $H$, the Huffman point associated with $H$ is the point $(l_1, l_2, \ldots, l_n) \in \mathbb{Q}^n$. The Huffman polytope is the convex hull of the Huffman points associated with all the possible Huffman trees for the $n$ characters in $\Lambda$. Let $f_i$ for $i = 1, \ldots, n$ be the frequency

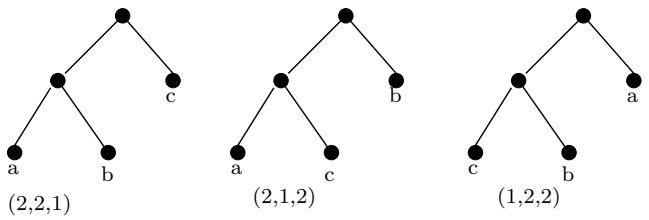


Figure 1-2: The Huffman trees and the associated Huffman point for the alphabet $\Lambda = \{a, b, c\}$.

of $c_i$ in a language having $\Lambda$ as alphabet, Huffman gives a $O(n \log n)$ time greedy algorithm to find the Huffman tree that minimizes the linear function $\sum_{i=1}^{n} f_i l_i$. One can derive from this tree an optimal binary code for the characters of $\Lambda$ with respect to the frequencies $f_i$. Hence, optimizing over the Huffman tree polytope can be done greedily in polynomial time. The fact seems to motivate J. Edmons and J.F. Maurras to introduce the Huffmantree polytope in 1974. They conjectured that in regards to the simplicity of the Huffman's algorithm, the Huffman polytope should have a simple and beautiful facial structure. In [46], we characterize all the facet-defining inequalities that define the highest Huffman trees (i.e. the Huffman trees that are the most unbalanced). The number of those inequalities is in $O(n!)$, what shows that $n!$ is lower bound for the number of facet-defining inequalities for the Huffman polytope.

This result implies that despite appearances, the Huffman polytope could be very complex. Note that following our result, Kaibel and Pashkovich [38] have given a $O(n \log n)$ size extended formulation for the Huffman polytope.

# Chapter 2

# Natural formulations and algorithms using linear programming

In this chapter, we consider combinatorial optimization problems for which natural integer formulations exists or is obvious. We show that more meticulous observations on these formulations can help to improve exact or approximation algorithms. In particular,

- the observation that if the cycles should contain a specific node, we can replace them by a $st$-chain (undirected path) in an extended graph. In spite of its simplicity, this observation is interesting polyhedrally since complet description for the $st$-chain polytope is known and no such thing is known for the cycle polytope. We apply this to the Ring Star problem in Section 2.1.

- The parsimonious property of directed Eulerian graphs that allows to remove nodes (by splitting-off operation) while preserving the connectivity and even the overall arc costs when the latters satisfy the triangle inequalities. We apply this to the Minimum Directed Tour problem in Section 2.2.

## 2.1 Chain-based formulation for Ring Star

### 2.1.1 Introduction

This section addresses a telecommunication network problem called the ring star problem. It consists in finding a simple cycle through a subset of vertices of a graph which minimizes the cost of the cycle and the assignment cost of the vertices not in the cycle to their closest vertices on the cycle. The problem has several applications in telecommunications. Indeed, the ring topology is chosen in many fiber optic communication networks to guarantee continuous communication service to the customers called terminals. These customers are connected to the concentrators of the ring by point-to-point links which results in a star topology. In other words, the problem consists in selecting a subset of user locations where concentrators will be installed, interconnecting them by a ring network and assigning the other customer locations to the concentrators.

The Ring Star Problem (RSP) was introduced by Labbé et al. [43] who derive a branch-and-cut method based on some polyhedral properties of the problem. It can be formally stated as follows. Let $G = (V, E \cup A)$ be a mixed graph, where $V = \{v_1, v_2, \cdots, v_n\}$ is the vertex set, $E = \{v_i v_j : v_i, v_j \in V\}$ is the edge set and $A = \{(v_i, v_j) : v_i, v_j \in V\}$ is the arc set. Vertex $v_1$ is referred to as a root (or depot). Edges in $E$ refer to the undirected concentrators links, and arcs in $A$ refer to the directed assignment between customers and concentrators. A nonnegative ring cost $c_{ij}$ is associated with each edge $v_i v_j$ and a nonnegative assignment cost $d_{ij}$ is associated with each arc $(v_i, v_j)$. A solution of the ring star problem is a simple cycle through a subset $C$ of $V$ including $v_1$. The objective is to determine a solution for which the sum of the ring and the assignment costs is minimized. The ring cost of a solution is the sum of the ring costs of all edges on the cycle. The assignment cost is defined as $\sum_{v_i \in V \setminus C} \min_{v_j \in C} d_{ij}$.

The problem is NP-hard since the special case in which the assignment costs are very large compared to the ring cost is the classical traveling salesman problem.

In [43], the authors propose the first branch-and-cut algorithm for RSP based on

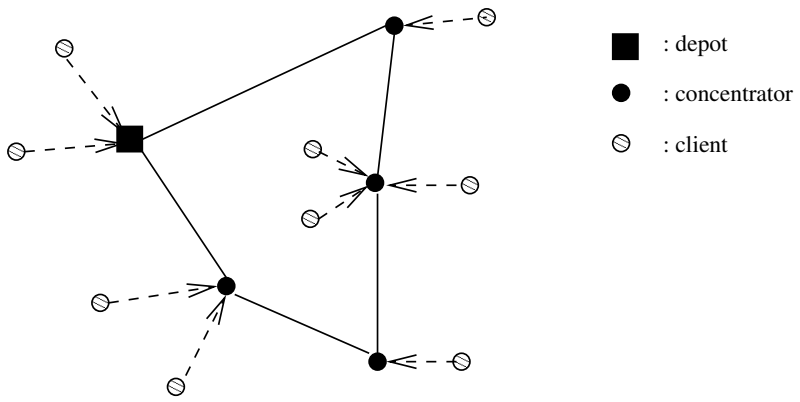


Figure 2-1: A solution of the ring star problem.

the circuit polytope. The latter studied by Bauer et al. [9] which is the convex hull of the incidence vectors of all the simple cycles in $G = (V, E)$. In [40], we observe that in a solution of RSP, the cycle must go through the node $v_1$. Hence, the set of the cycles beloing to a solution of RSP is a strict subset of the set of all the cycles in $G$. Consequently, the valid inequalities for RSP derived from the circuit polytope, the convex hull of the incidence vectors of all the cycles, may not always define facets for RSP. Our idea is to consider directly the convex hull of the incidence vectors of all the cycle containing $v_1$ which should give stronger valid inequalities for RSP than the circuit polytope. More precisely, we show that this convex hull has a complete extended formulation and we use it instead of the circuit polytope to derive a stronger class of facet-defining inequalities than the one having been derived using the circuit polytope. The extend formulation is as follows:

> We set $s = v_1$ and add a dummy node $t$ which is a clone of $s$ in $G$ to obtain a new graph $G' = (V', E' \cup A')$ where $V' = V \cup \{t\}$, $E' = E \cup \{tu |\ su \in E\}$ and $A' = A$. Let $n' = |V'|$, $m'_e = |E'|$, $m'_a = |A'|$ and $m' = m'_e + m'_a$. A $st$-chain in $G'$ is an simple undirected path between $s$ and $t$ whose edges are in $E'$. We can see that there is an one-one correspondence between the cycles containing $v_1$ in $G$ and the $st$-chain in $G'$. Hence, the $st$-chain polytope which is the convex hull of the incidence vectors of the $st$-chains of $G'$ is an extended formulation of the convex hull of the cycles containing $v_1$ in

$G$. As constrast to the circuit polytope for which no complete description is known, one knows a complete description of the $st$-chain polytope since a $st$-chain is a $T$-join with $T = \{s, t\}$.

We know that the blossom inequalities

$$x(\delta(U) \setminus F) - x(F) \leq 1 - |F| \; U \subset V', \; F \subset \delta(U) \text{ s.t. } |U \cap \{s, t\}| + |F| \text{ odd},$$

describe the $T$-join polytope with $T = \{s, t\}$ [24]. In [40], we derive from the blossom inequalities a class of valid inequalities for RSP called *st-chain blossom inequalities*:

$$x(E(U)) + x(F) \leq \sum_{v_i \in U \setminus \{s,t\}} y_{ii} - \lfloor \frac{|F| - 1}{2} \rfloor, \tag{2.1}$$

for $U \subset V'$, $F \subset \delta(U)$ s.t. $|U \cap \{s, t\}| + |F|$ odd. The $st$-chain blossom inequalities contain a subclass called *st-chain blossom even* ((2.1) with $|F|$ even) which defines facets for RSP and which is a new class of facet-defining inequalities comparing with the ones in [43]. The following table show the contribution of the new $st$-chain blossom even inequalities for improving the lower bound of RSP at the root node of the branch-and-cut tree. The instances are from the TSPLIB, $\alpha$ is a parameter for the estimation of the assignement cost in term of edge cost given by TSPLIB. The column CyB-BC denotes the cycle-based formulation given [43] and the column ChB-BC denotes our $st$-chain based formulation. The value in these column is the percentage of lower bound given at the root node over the best known solution value of RSP (which is not necessary the optimal value). The last column is the number of $st$-chain blossom even inequalities added by cutting-plane at the root node.

| Name | $\alpha$ | CyB-BC | ChB-BC | $st-$chain blossom even |
|---|---|---|---|---|
| kroA150 | 3 | 99.49 | 99.49 | 26 |
| kroB150 | 3 | 99.51 | 99.57 | 15 |
| pr152 | 3 | 99.51 | 99.55 | 3 |
| pr152 | 5 | 96.05 | 96.11 | 155 |
| pr152 | 7 | 96.66 | 96.76 | 308 |
| rat195 | 3 | 99.68 | 99.74 | 99 |
| kroA200 | 3 | 93.59 | 96.59 | 12 |
| kroA200 | 9 | 97.15 | 98.51 | 151 |
| kroB200 | 3 | 99.81 | 99.86 | 2 |
| kroB200 | 9 | 95.13 | 97.52 | 66 |

## 2.2 Approximation algorithm for the Minimum Directed Tour Cover Problem

### 2.2.1 Introduction

Let $G = (V, A)$ be a directed graph with a (non-negative) cost function $c : A \Rightarrow \mathbb{Q}_+$ defined on the arcs. A *directed tour cover* $T$ is a subgraph $T = (U, F)$ such that

1. For every $e \in A$, $F$ contains an arc $f$ intersecting $e$, *i.e.*, $f$ and $e$ have at least one end-node in common. In other words, $U$ is a vertex cover of the undirected version of $G$ where the orientation of the arcs is neglected.

2. $T$ is a closed directed walk.

We consider in this paper *the minimum directed tour cover (DToCP)* problem which is to find a directed tour cover of minimum cost. We can prove that DToCP is $NP$-hard by a reduction from the metric Asymmetric Traveling Salesman Problem (ATSP). We give here a concrete example of applications of DToCP, which is the following simplified version of the waste collection problem. A company is in charge of waste collection in an urban area. The area is supposed to be small enough so that one truck is sufficient to collect all of its daily wastes. The gather is done by the truck in early morning of each day. For each street in the area, waste should be regrouped at one of the end-points of the street in the night of the day before. When

the truck goes along a street, this induces some cost to the company. Hence, the aim of the company is to find a cheapest tour from its depot for the truck such that the waste collection is accomplished for every street of the area. Note that streets could be one-way or sloping, making the costs asymmetric. This problem suggests that the depot should belong to the tour. We shall see that our algorithm for DToCP will also work for the case when the tour should contain a fixed vertex. In this paper, we give the first approximation algorithm for the DToCP achieving a approximation ratio of $2\log_2(n)$. For this, we show the parsimonious property of Eulerian directed graphs, in particular we prove the equivalent directed version of Goemans and Bertsimas's theorem on splitting operations.

### 2.2.2 Integer Formulation

Let

$$\mathcal{F} = \{S \subseteq V \mid A(S) \neq \emptyset, A(V \setminus S) \neq \emptyset\}$$

Given any tour cover $TC$ of $G$, by definition, for any $S \in \mathcal{F}$ we have $TC \cap \delta^+(S) \geq 1$ and $TC \cap \delta^-(S) \geq 1$. Note that the condition $TC \cap \delta^-(S) \geq 1$ is equivalent to $TC \cap \delta^+(V \setminus S) \geq 1$, and by definition of $\mathcal{F}$, a vertex subset $S$ belongs to $\mathcal{F}$ if and only if $V \setminus S$ belongs to $\mathcal{F}$. This observation motivates our integer formulation for DToCP. For any $e \in A$, let $x_e$ indicate the number of copies of $e$ included in the tour cover. We minimize the total weight of arcs included, under the condition that every outgoing cut associated to some $S \in \mathcal{F}$ should be crossed at least one. In order to ensure that our solution is a tour we also need to specify that for any node $v \in V$, the number of arcs entering $v$ is equal to the number of arcs leaving $v$ in the tour.

This integer formulation can be stated as follows:

$$\begin{aligned}
&\min \sum_{e \in A} c_e x_e && \\
&\sum_{e \in \delta^+(v)} x_e = \sum_{e \in \delta^-(v)} x_e && \text{for all } v \in V \\
&\sum_{e \in \delta^+(S)} x_e \geq 1 && \text{for all } S \in \mathcal{F} \\
&x_e \text{ integer} && \text{for all } e \in A
\end{aligned}$$

If one specific vertex $u$ should belong to the optimal tour, we just need to set $\sum_{e \in \delta^+(u)} x_e = \sum_{e \in \delta^-(u)} x_e \geq 1$ in the formulation. Replacing the integrality constraints by

$$x_e \geq 0 \text{ for all } e \in A$$

we obtain the linear programming relaxation. We use DToC($G$) to denote the convex hull of all vectors $x$ satisfying the above constraints (those of the linear programming relaxation). Clearly minimizing the linear cost function $\sum_{e \in A} c_e x_e$ over DToC($G$) can be done in polynomial time since the separation problem of the cut constraint can be solved in polynomial time. Indeed, given a candidate solution $x$ and for every $e \in A$, let us consider $x_e$ as the capacity on arc $e$. For each pair of arcs $e_1,\ e_2 \in E$, we compute the minimum capacity cuts $\delta^+(U)$ separating them.

### 2.2.3 Algorithm's sketch

We state here our algorithm for directed tour cover which is heavily inspired from the one for ToCP in [41].

**(1)** Let $x^*$ be the vector minimizing $cx$ over DToC($G$);

**(2)** Let $U \leftarrow \{v \in V | x^*(\delta^+(\{v\})) \geq \frac{1}{2}\}$;

**(3)** Let $G_U = (V_U, A_U)$ be a graph with vertex set $V_U = U$ and the arc set $A_U$ is built as follows: for each pair of nodes $i, j \in U$, if there exists a path from $i$ to $j$ in $G$, create an arc $(i, j)$ in $G_U$ with the cost $c^U_{ij}$ being equal to the cost of the shortest

path from $i$ to $j$ in $G$;

**(4)** Run the Frieze, Galbiati and Maffioli heuristic [28] to find an approximate minimum traveling salesman directed tour on $G_U$.

Note that the linear program in step **(1)** can be solved in polynomial time by using the ellipsoid method with a minimum cut computation as separation oracle. The algorithm outputs a directed tour which spans $U$. We can see that $U$ is vertex cover of the undirected version of $G$ where the arcs becomes edges. Since for any arc $e = (u, v) \in A$, $x^*(\delta^+(\{u, v\})) \geq 1$ and $x^*(\delta^+(\{u, v\})) = x^*(\delta^+(u)) + x^*(\delta^+(v)) - 2x^*_e$, at least $x^*(\delta^+(u))$ or $x^*(\delta^+(v))$ is greater or equal to $\frac{1}{2}$, *i.e.*, at least $u$ or $v$ should belong to $U$. Therefore, the algorithm outputs a directed tour cover of $G$.

### 2.2.4 Held-Karp Relaxation for ATSP and the Parsimonious Property

Let us consider the Asymmetric Traveling Salesman Problem (ATSP) on $G_U$. The Held-Karp bound for this problem can be computed by solving the following linear program:

$$\min \sum_{e \in A_U} c_e x_e \tag{2.2}$$

subject to

$$\begin{aligned}
x(\delta^+(S)) &\geq 1 && \text{for all } S \subset V_U \text{ such that } |S| \geq 2 \\
x(\delta^+(v)) &= 1 && \text{for all } v \in V_U \\
x(\delta^-(v)) &= 1 && \text{for all } v \in V_U \\
x_e &\geq 0 && \text{for all } e \in A_U
\end{aligned}$$

Note that a directed traveling salesman tour can be viewed as a directed Eulerian subgraph where the in-degree and out-degree of every node are exactly 1. Such a tour is called 1-degree directed Eulerian subgraph. In [52], we show the parsimonious

property for k-degree directed Eulerian subgraph, i.e., if the costs $c$ satisfy the triangle inequality, we can relaxed the degree constraints in the problem of finding minimum degree-k-Eulerian directed graph. That implies the following theorem for the program 2.2.

**Theorem 2.2.1** *[N. [52] If the costs* c *satisfy the triangle inequality then the optimum of* (2.2) *is equal to the optimum of:*

$$\min \sum_{e \in A_U} c_e x_e \tag{2.3}$$

*subject to*

$$\begin{aligned} x(\delta^+(S)) &\geq 1 && \text{for all } \emptyset \neq S \subset V_U \text{ (included } S \text{ singleton)} \\ x(\delta^+(v)) &= x(\delta^-(v)) && \text{for all } v \in V_U \\ x_e &\geq 0 && \text{for all } e \in A_U \end{aligned}$$

### 2.2.5 Analysis of the algorithm in Section 2.2.3

Let us consider the point $x^* \in \mathbb{Q}^A$ obtained after step **(1)**. By the parsimonious property proved in [52], we can transform $x^*$ to a point $x_U \in \mathbb{Q}^{A_U}$ such that $c_U^t x_U \leq c^t x^*$ and $2x_U$ is a point satisfying the linear program (2.3). Let $T$ be the directed tour spanning $U$ output by step **(4)** of the algorithm. Let $x_U^*$ be an optimal of the linear program (2.3). Williamson [61] shows that $c(T) \leq \log_2(n) \times c_U^t x_U^*$. Hence, we have $c(T) \leq \log_2(n) \times c_U^t x_U \leq 2\log_2(n) \times c^t x^*$. The algorithm in Section 2.2.3 is then a $2\log_2(n)$-approximation algorithm for the directed tour cover problem.

# Chapter 3

# Extended formulations: compactness and separation

In this chapter, we investigate extended formulations under the following aspects:

- Given a problem with a natural formulation of exponential size, we aim at finding an extended formulation of compact size.

- Given a problem with a compact size extended formulation which is integral for some special case, we aim at projecting the latter to obtain classes of facet-defining inequalities in natural space. The latters define a complete description for the natural formulation for the special case in question. Moreover, we show that the separation problem of theses classes of facet-defining inequalities can be solved by optimizing over the extended formulation.

- Given a problem for which compact size extended formulations are known, we aim at reducing per se the size of these extended formulation or at finding the minimal size for them in special cases.

## 3.1 The substar polytope and extensions

### 3.1.1 Natural formulation and a generalization of Kőnig's edge-coloring theorem

Let $G = (V, E)$ be an arbitrary undirected graph without loop but with multiple edges allowed. A *substar of* $G$ is an edge subset $F \subseteq E$ such that $F \subseteq \delta(v)$ for some $v \in V$. We define the *substar polytope*, denoted by SSP, as the convex hull of the incidence vectors in $\mathbb{Q}^E$ of all the substars of $G$.

A *matching of* $G$ is a set of pairwise disjoint edges.

A *fractional matching of* $G$ is a (not necessarily integer) vector $\mu \in \mathbb{Q}^E$ satisfying:

$$(\text{FMATCH}) \begin{cases} x_e \geq 0 & \text{for all } e \in E \\ x(\delta(v)) \leq 1 & \text{for all vertex } v \in V. \end{cases}$$

**Theorem 3.1.1 (Cornaz and N. [17])** *The following system describes the substar polytope of all graphs:*

$$(A(\textit{FMATCH})) \begin{cases} x_e \geq 0 & \textit{for all } e \in E \\ \mu^\top x \leq 1 & \textit{for all fractional matching } \mu \textit{ of } G. \end{cases}$$

**Proof** Indeed, given any $c \in \mathbb{Z}^E$, restrict the set of fractional matchings to the finite subset of fractional matchings $\mu$ of the form $\mu_e = \frac{p}{q}$ with $p \in \mathbb{Z}$ and $q \in \{1, \ldots, \sigma^* := \max_{v \in V} c(\delta(v))\}$, then

$$\psi_{LP} = \begin{cases} & \min \mathbf{1}^\top y & \\ \text{s.t.} & & \\ & y_\mu \geq 0 & \text{for all fractional matching } \mu \text{ of } G \\ \sum_{\text{fractional matching } \mu} & \mu_e y_\mu \geq c_e & \text{for all } e \in E, \end{cases}$$

is the dual of maximizing $c^\top x$ over $x$ satisfying $(A(\text{FMATCH}))$, and a solution of $\psi_{LP}$

with value $\sigma^*$ is obtained with $\bar{y}$, defined as follows:

$$\bar{y}_{\bar{\mu}} := \begin{cases} \sigma^* & \text{if } \mu = \bar{\mu} \\ 0 & \text{otherwise} \end{cases} \qquad \text{with} \quad \bar{\mu}_e := \frac{c_e}{\sigma^*} \quad \forall e \in E.$$

Here $\bar{\mu} \geq 0$ is well a fractional matching since

$$\bar{\mu}(\delta(u)) = \frac{c(\delta(u))}{\sigma^*} \leq 1 \quad \text{for all } u \in V,$$

and $\bar{y}$ is well feasible since

$$\bar{\mu}_e \times \bar{y}_{\bar{\mu}} = \frac{c_e}{\sigma^*} \times \sigma^* = c_e \quad \text{for all } e \in E.$$

□

As the number of fractional matching are infinite, the system $(A(\text{FMATCH}))$ is not a minimal system describing the substar polytope. To derive a such system, we will make use the follwing result of Balinski. An *(odd cycles,matching) set* $(C, M)$ *of* $G$ is a collection of vertex-disjoint edge subsets $C_1, \ldots, C_k$ such that $C_i$ is either an odd cycle or a singleton; we denote by $C$ the union of the odd cycles and we denote by $M$ the union of singletons (so $M$ is a matching). For short, (odd cycles,matching) is abbreviated as *ocm*. Balinski [5] showed that a vector $x \in \mathbb{Q}^E$ is an extreme point of (FMATCH) if and only if

$$x = \frac{1}{2}\chi^C + \chi^M \quad \text{for some ocm set } (C, M) \text{ of } G.$$

Hence, we have

**Corollary 3.1.2 (Cornaz and N. [17])** *The system*

$$(A(\mathit{FMATCH}^*)) \begin{cases} x_e \geq 0 & \textit{for all } e \in E \\ \frac{1}{2}x(C) + x(M) \leq 1 & \textit{for all ocm set } (C, M) \textit{ of } G, \end{cases}$$

*is a minimal system describing the substar polytope provided that the ocm sets are*

*inclusionwise.*

Moreover, we show the following theorem.

**Theorem 3.1.3 (Cornaz and N. [17])** *The system* $(A(\mathit{FMATCH}^*))$ *is TDI.*

An *ocm covering of* $G$ is a collection of ocm sets $(C_1, M_1), \ldots, (C_k, M_k)$ such that each edge is covered by one matching or by two (elementary) odd cycles, formally: For each edge $e \in E$, either there exists $i$ such that $e \in M_i$, or there exists $i \neq j$ such that $e \in C_i$ and $e \in C_j$, for some $i, j \in \{1, \ldots, k\}$. Let $\chi''(G)$ denote the minimum $k$ such that there is an ocm covering of $G$ with $k$ ocm sets. Let $\Delta(G)$ denote the maximum degree of $G$, that is $\Delta(G) := \max_{v \in V} |\delta(v)|$. The TDI-ness of $(A(\text{FMATCH}^*))$ implies the following theorem.

**Theorem 3.1.4** $\chi''(G) = \Delta(G)$ *for all graph* $G$*.*

We derive from the fact that when $G$ is a bipartite graph, every ocm covering is matching covering, that Theorem 3.1.4 is a generalization of Kőnig's edge-colouring theorem.

### 3.1.2 Compact extended formulation for the substar polytope

In this section, we will give an compact extended formulation for the substar polytope whose projection on $\mathbb{Q}^E$ gives $A(\text{FMATCH}(G))$. In particular, we will add addtional variable $y_v$ for each $v \in V$ to express the fact that $y_v = 1$ if $v$ is the center (the *center* of a substar is the only node of degree strictly greater than 1 if the substar is not an edge. If the substar is an edge, any of the latter could be the center.) of the substar and 0 otherwise. Then the constraints we should formulate are:

- there is at most one center,
- for every edge $uv$, if $uv$ belong to the substar, then either $u$ or $v$ is the center.

Let

$$(Q)\begin{cases} \sum_{v\in V} y_v \leq 1 & \\ x_{uv} \leq y_u + y_v & \text{for all } uv \in E \\ y_v \geq 0 & \text{for all } v \in V \\ x_e \geq 0 & \text{for all } e \in E \end{cases}$$

be the system expressing these above constraints.

**Theorem 3.1.5 (Cornaz and N. [17])** *$(Q)$ is TDI and the projection of $Q$ on $\mathbb{Q}^E$ is equal to $A(\textsc{fmatch}(G))$*

**Corollary 3.1.6 (Cornaz and N. [17])** *$(Q)$ is an compact extended formulation of the substar polytope.*

## 3.2 Facets and extended formulations for the substar forest polytope

### 3.2.1 Introduction

The previous section consider the substars individually, in this section, we will consider them collectivelly. Given an undirected graph $G = (V, E)$ where $n = |V|$ and $m = |E|$, a *substar* in $G$ is either a single node of $G$ or a subgraph of $G$ where every edge shares one common end-node. The latter is called the *center* of the substar when the substar is not reduced to a single node. If the substar is a single edge, then any of its end-nodes can be designated as the center. A *substar forest* is a collection of vertex-disjoint substars in $G$. We suppose that the edges in $G$ have non-negative

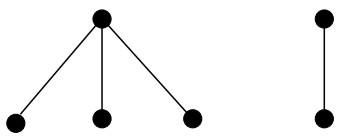

Figure 3-1: A substar forest of weight 4 with weights 1 on the edges.

weights (note that the unweighted case can be seen as a special weighted case when

weights are 0 or 1), then the weight of a substar forest or an edge dominating set is the sum of the weights of its edges. The *Maximum Weight spanning Star Forest Problem* (MWSFP) is to find a substar forest spanning the nodes of $G$ of maximum weight. The MWSFP is a recent problem which has been introduced by Nguyen et al. in [48]. It has applications in several areas, especially in computational biology [48] and automobile industry [1]. In [48], the authors show the $NP$-hardness of MWSFP by observing that in a *maximal substar forest* $F$ (a maximal substar forest is a substar forest to which no more edge can be added), the set of the centers of the substars in $F$ is a dominating set of $G$. Conversely, for any dominating set $S$ we can build a maximal substar forest with centers as the nodes belonging to $S$. Thus, given a maximal substar forest $F$, there exists a dominating set $S$ such that $|S| = |V| - |F|$ and vice versa.

Let $SFP(G)$ be the convex hull of the incidence vectors of the substar forests in $G$.

We call a 3-*path* a simple path having 3 edges in $G$ and a 3-*cycle* a triangle in $G$. Let $\mathcal{P}4$ (respectively $\mathcal{C}_3$ ) denote the collection of the 3-paths (resp. 3-cycles ) in $G$.

### 3.2.2 Integer formulations for the MWSFP

### 3.2.3 An integer formulation in natural space

In this subsection, we give an integer programming formulation for MWSFP in $\mathbb{Q}^E$. First we state the following lemma.

**Lemma 3.2.1** *A substar forest is a graph without 3-paths and 3-cycles, and vice-versa.*

Let us consider the following integer program.

$$
\begin{array}{lll}
 & \max\ c^T x & \\
(IP) & \text{s.t.} & \\
 & x(P) \leq 2 \qquad \text{for all } P \in \mathcal{P}_4 & (3.1) \\
 & x(C) \leq 2 \qquad \text{for all } P \in \mathcal{C}_3 & (3.2) \\
 & 0 \leq x_e \leq 1 \qquad \text{for all } e \in E & (3.3) \\
 & x \text{ integer} &
\end{array}
$$

Inequalities (3.1), called the *3-path inequalities*, state the fact that a substar forest can only take at most 2 edges in a 3-path. Similarly, inequalities (3.2), called the *3-cycle inequalities*, state the fact that a substar forest can only take at most 2 edges in a 3-cycle. Inequalities (3.3) are the *trivial inequalities*.

**Theorem 3.2.2** *$(IP)$ is equivalent to the MWSFP.*

**Proof** It is clear that by inequalities (3.1) and (3.2) in a solution of $(IP)$ there is neither 3-paths and nor 3-cycles. By Lemma 3.2.1, this solution represents a substar forest.

### 3.2.4 An extended integer formulation and valid inequalities for $SFP(G)$

$\vec{G} = (V, \vec{E})$ be the *bidirected graph* obtained by replacing every edge $ij$ of $G$ by two arcs $(i, j)$ and $(j, i)$. Hence, $\vec{G}$ has the same vertex set as $G$ and the number of arcs in $\vec{E}$ is two times the number of edges in $E$. We consider the *uncapacitated facility location problem* (UFLP) [18] defined on $\vec{G}$ where each vertex $i$ could be either a facility or a client, and each arc $(i, j) \in \vec{E}$ represents the assignment of client $i$ to the facility $j$. In a solution of the uncapacitated facility location problem each vertex $i$ should be determined to be either a facility or a client. If in a solution $i$ is a facility, then it should be opened with a cost $w(i)$ and $i$ is called a *center* in this case. Otherwise,

i.e. $i$ is a client, it should be assigned to an opened facility $j$ with a cost $c(i,j)$. Note that there can be centers in a solution to which no client is assigned. We consider here the *symmetric uncapacitated facility location problem*, SUFLP, a special version of UFLP on $\vec{T}$ in which the assignment costs are symmetric, i.e. $c(i,j) = c(j,i)$ for all $ij \in E$, and there is no opening cost for facilities, i.e. $w(i) = 0$ for all $i \in V$. Let us associate with each solution $F$ of UFLP, the incident vector $(x,y) \in \mathbb{R}^{|\vec{E}|+|V|}$ defined as follows:

- for all $(i,j) \in \vec{E}$,
$$\vec{x}(i,j) = \begin{cases} 1 & \text{if } (i,j) \in F, \\ 0 & \text{otherwise,} \end{cases}$$
where $\vec{x}(i,j)$ is the component of $x$ corresponding to the arc $(i,j) \in \vec{E}$, and

- for all $i \in V$,
$$y(i) = \begin{cases} 1 & \text{if } i \text{ is a center in } F, \\ 0 & \text{otherwise,} \end{cases}$$
where $y(i)$ is the component of $y$ corresponding to the vertex $i \in V$.

The following system, called $ESFP(\vec{G})$, derived from the integer formulation in [19] is an integer formution for UFLP in $\vec{G}$:

$$ESFP(\vec{G}) \quad \sum_{(i,j)\in\vec{E}} \vec{x}(i,j) + y(i) \leq 1 \qquad \text{for all } i \in V \tag{3.4}$$

$$\vec{x}(i,j) \leq y(j) \qquad \text{for all } (i,j) \in \vec{E} \tag{3.5}$$

$$0 \leq y(i) \leq 1 \qquad \text{for all } i \in V \tag{3.6}$$

$$\vec{x}(i,j) \in \{0,1\} \qquad \text{for all } (i,j) \in \vec{E} \tag{3.7}$$

**Lemma 3.2.3** *$ESFP(\vec{G})$ is an extended integer formulation for $SFP(G)$.*

**Proof** The proof can be done by simply using the correspondence $x_{ij} = \vec{x}(i,j) + \vec{x}(j,i)$ between the assignments $\vec{x}(i,j)$ and $\vec{x}(j,i)$ in a solution of UFLP($\vec{G}$) and the value of $x_{ij}$ in a solution of $SFP(G)$.

Let $LP\_ESFP(\vec{G})$ be the linear programming relaxation of $ESFP(\vec{G})$. We will consider the projection of $ESFP(\vec{G})$ on the natural space $\mathbb{Q}^E$ of $SFP(G)$ by using the correspondences $x_{ij} = \vec{x}(i,j)) + \vec{x}(j,i)$. For that, let us first define in $\mathbb{Q}^E$, a class of inequalities called the *perfect b-matching inequalities*. Given a vertex subset $S \subseteq V$, let $G_S = (S, E_S)$ be a subgraph whose node set is $S$ and edge set $E_S$ is a subset of $E(S)$ (recall that $E(S)$ is the set of the edges of $G$ with both end-nodes in $S$). We will suppose that $G_S$ is a connected graph. A *perfect b-matching* in $G_S$ is a subgraph (which is not necessarily simple) in which the degree of each vertex in $S$ is $b$.

Given a vertex subset $S \subset V$ such that $G_S$ is connected and a perfect $b$-matching $B$ in $G_S$. Let $S_2 \subset S$ be the set of the vertices of degree at least 2 in $G_S$ and $E_S^2 \subset E_S$ is the subset of the edges in $E_S$ with both end-nodes in $S_2$. For each edge $ij \in E_S$, let $m_{ij}$ denote the multiplicity of $ij$ in $B$ ($m_{ij} = 0$ if $ij \notin B$). Then the perfect $b$-matching subgraph inequalities can be defined as follows:

$$\sum_{ij \in E_S \setminus E_S^2)} m_{ij} x_{ij} + \sum_{ij \in E_S^2)} (m_{ij} + b) x_{ij} \leq b|S_2| \qquad (3.8)$$

$\forall S \subseteq V$ and $\forall$ connected subgraph $G_S$ and $\forall$ perfect $b$-matchings $B$ in $G_S$.

**Theorem 3.2.4** *The projection of $LP\_ESFP(\vec{G})$ on $\mathbb{Q}^E$ of $SFP(G)$ gives the perfect b-matching inequalities.*

**Corollary 3.2.5** *The perfect b-matching subgraph inequalities are valid for $SFP(G)$.*

We can remark that the 3-path inequalities is a perfect b-matching subgraph inequalities with $S$ is a set of 4 nodes $\{v_1, v_2, v_3, v_4\}$ such that $G$ contains the 3-path $v_1v_2v_3v_4$, with subgraph $G_S$ equal to the latter and with $B = \{v_1v_2, v_3v_4\}$. The trivial inequality $x_{ij} \leq 1$ for $ij \in E$ is also a perfect inequalities with $S = \{i, j\}$, $G_S = B =$ the singleton containing the edge $ij$.

### 3.2.5 Complete description of SFP(G) in trees

From now on and throughout this section, $G$ will be a tree denoted by $T$.

**Theorem 3.2.6** *[3] $LP\_ESFP(\vec{T})$ completely defines $UFLP(\vec{T})$.*

Actually, in [3], the authors characterize all the graphs $G$ such that $LP\_ESFP(\vec{T})$ completely defines $UFLP(\vec{G})$. Such graphs include bidirected trees.
Let us now consider the MWSFP on $T$ in which for each edge $ij \in T$, the cost $c_{ij}$ is equal to $c(i,j)$.

**Theorem 3.2.7** *The projection of $LP\_ESFP(\vec{T})$ on the variables $x_{ij}$ gives a system consisting of the perfect b-matching subgraph inequalities defined on $T$ and the nonnegativity inequalities.*

We can derive from Lemma 3.2.3 and Theorem 3.2.7 the following corollary.

**Corollary 3.2.8** *The perfect b-matching subgraph inequalities together with the nonnegativity inequalities completely describe $SFP(G)$ when $G$ is a tree.*

### 3.2.6 Separation of perfect b-Matching subgraph inequalities

Let us consider the separation problem of the perfect b-matching subgraph inequalities, i.e. given a solution $x^* \in [0,1]^E$, we want to know if $x^*$ violates some perfect $b$-matching subgraph inequalities, and if yes, then exhibit at least one. We will solve it via the integer extended formulation given in Section 3.2.4. Precisely, we solve the following linear program (SEP):

$$\text{(SEP)} \min \sum_{ij \in E} \delta_{ij}$$

$$\sum_{\substack{(i,k) \in \vec{E} \\ k \neq j}} \vec{x}(i,k) - \delta_{ij} \leq 1 - x^*_{ij} \qquad \text{for all } i \in V \text{ and } ij \in E$$

$$\vec{x}(i,j) + \vec{x}(j,i) = x^*_{ij} \qquad \text{for all } ij \in E$$

$$\vec{x}(i,j),\ \vec{x}(j,i),\ \delta_{ij} \geq 0 \qquad \text{for all } ij \in E$$

**Theorem 3.2.9** *There is a violated perfect b-matching subgraph inequality by $x^*$ if and only if the value of the objective of (SEP) is strictly positive.*

## 3.3 Extended formulations for metric polyhedra

### 3.3.1 Natural formulations

Given an undirected graph $G = (V, E)$, we consider in this section, two polyhedra defined on $\mathbb{Q}^E$. The first is the *metric polytope* , denoted by METP($G$), associated with $G$ , which can be defined as follows:

$$x(F) - x(C \setminus F) \leq |F| - 1$$
$$\forall C \in \mathcal{C} \text{ and } F \subseteq C \text{ with } |F| \text{ odd}, \tag{3.9}$$
$$0 \leq x_e \leq 1$$
$$\forall e \in E \text{ such that } e \text{ does not belong to any triangle} \tag{3.10}$$

Note that inequalities (3.9) are called *cycle inequalities*. The inequalities (3.10) are applied only for the edges in $G$ which do not belong to any triangle as those for the other edges can be derived from the cycle inequalities. These inequalities were introduced in the seminal paper by [8] on the cut polytope. The second polyhedron is *the metric cone* MET($G$) which consists of the cycle inequalities with sets $F$ such that $|F| = 1$ (the homogenuous cycle inequalities) and the "trivial" inequalities (3.10). More precisely,
MET($G$) = $\{x \in \mathbb{R}^E$ such that

$$x_e - x(C \setminus \{e\}) \leq 0 \ \forall C \in \mathcal{C} \text{ and } e \in C, \tag{3.11}$$

$$0 \leq x_e \leq 1 \ \forall e \in E.\}$$

Note that MET($G$) is a polytope, not a cone. However, we use here the standard terminology used by [21] which was proposed in a context where the basic space considered was the hypercube $[0, 1]^n$.

### 3.3.2 Compact extended formulations

Note that since there is a priori no known polynomial upper bound (in terms of $n$ and $m$) on the number of chordless cycles and there may be also an exponential number of choices for the set $F$ given a chordless cycle $C$, the above formulations of MET($G$) and METP($G$) have a priori an exponential number of inequalities. Nevertheless, when $G = K_n$, the complete graph of $n$ nodes, MET($K_n$) and METP($K_n$) are of polynomial size since in this case $\mathcal{C}$ reduces to the set of the triples $\{i \neq j \neq k \in V\}$ and $F$ can have only 1 or 3 edges. Concretely, let $\mathcal{T}$ be the set of all the (unordered) triples of distinct nodes $i, j, k \in V$, the following system:

$$x_{ij} + x_{ik} + x_{jk} \leq 2 \text{ for all } i, j, k \in \mathcal{T}. \tag{3.12}$$

$$\begin{aligned} &x_{ij} - x_{ik} - x_{jk} \leq 0, \\ &x_{ik} - x_{ij} - x_{jk} \leq 0, \\ &x_{jk} - x_{ij} - x_{ik} \leq 0 \text{ for all } i, j, k \in \mathcal{T}. \end{aligned} \tag{3.13}$$

defines METP($K_n$). Inequalities (3.12) are called *the non-homogeneous triangle inequalities* and the ones in (3.13) are called *the homogenous triangle inequalities*. They are all commonly called *the triangle inequalities*. The cone MET($K_n$) is defined only by the homogeneous inequalities (3.13) and the trivial inequalites (3.10). The number of inequalities in MET($K_n$) and in METP($K_n$) is clearly in $O(n^3)$, and thus polynomial in terms of $n$. In fact, [7] showed that the projections of MET($K_n$) and METP($K_n$) on $\mathbb{R}^E$ are exactly MET($G$) and METP($G$). Hence, MET($K_n$) and METP($K_n$) respectively represent compact extended formulations for MET($G$) and METP($G$).

### 3.3.3 Applications of metric and related polyhedra

The two polyhedra MET($G$) and METP($G$) are strongly related to *the maximum cut problem* which is one of the basic problems in combinatorial optimization. Actually,

the metric cone MET($G$) is a relaxation of *the cut cone* CUT($G$), the cone generated by all the cut vectors $\delta(S)$ for $S \subset V$ (with abuse of notation, by $\delta(S)$ we denote both the edge set of the cut defined by the node set $S$ and its incidence vector). Similarly, the metric polytope is a relaxation of *the cut polytope* CUTP($G$), the convex hull of all the cut vectors $\delta(S)$ for $S \subset V$. If we replace the trivial inequalities by the 0/1 constraints $x \in \{0,1\}^E$ in the formulation of the two polyhedra, we obtain respectively integer formulations for CUT($G$) and CUTP($G$). METP($G$) is an interesting relaxation of CUTP($G$) both theoretically and practically. In particular, the cycle inequalities (3.9) define facets for CUTP($G$). Moreover, Deza et al. [22] have shown that among a large class of facet-defining inequalities for CUTP($G$) , the cycle inequalities are the closest ones to the barycentrum of CUTP($G$) and they conjectured that this holds in general. Boros et al. [12] have shown that METP($G$) is the Chvatal closure of the linearized quadratic formulation for Max-Cut. Pratically, cycle inequalities appear in most 0/1 formulations intended for solving the Max Cut problem exactly. They can even be used to strengthen the semi-definite relaxation for Max-Cut [57] which is known to provide a very good upper bound for Max-Cut [33]. Moreover, when $G$ is sparse, i.e. $m = O(n)$, the upper-bound of Max-Cut given by the relaxation METP($G$) is very strong [35], [45].

Note that the applications of MET($G$) and METP($G$) are not limited to the Max-Cut problem. They appear in linear programming relaxations for Graph Partitioning Problems which has applications in various domains such as large-scale parallel computing, imagery, electronics,? etc. They are also used to characterize the feasibility of multicommodity flow which is a fundamental notion in telecommunications networks. A large number of network design problems make use of cycle inequalities in their mathematical models [47]. METP($G$) is also present in formulations for the boolean quadric polytope which has important applications especially on quadratic 0/1 programming [13].

### 3.3.4 Reduced size extended formulations for metric polyhedra

In pratical applications, optimizing a linear function over MET($G$) and METP($G$) usually appears as a subproblem and thus the latter has to be solved repeatedly. In this situation, the compact formulations MET($K_n$) and METP($K_n$) are usually preferred to the non-compact ones for optimizing over MET($G$) and METP($G$) since they can be directly transmitted to a linear programming solver. However, the number of triangle inequalities in MET($K_n$) and METP($K_n$), which is in $O(n^3)$, can be huge even for medium values of $n$ making the optimization over compact formulations computationally difficult ([27] is a typical reference reporting such computational problem).

In [55], we show that one can reduce the number of triangle inequalities to $O(nm)$ while preserving equivalence with MET($G$) and METP($G$). The idea is that instead of express the triangles inequalities for all the triangles of $K_n$ completed from $G$, we only do it for those containing at least one original edge of $G$. The number of such triangles obviously is $(n-2)m$, hence the number of triangle inequalities is in $O(nm)$. More precisely, let

$$\mathcal{T}' = \{(i,j,k) \in \mathcal{T} | \text{ at least one of } ij,\ ik \text{ or } jk \in E\}$$

Let us define RMETP($K_n$) as the polytope defined by the following "reduced" system,

$$x_{ij} + x_{ik} + x_{jk} \leq 2 \text{ for all } i,j,k \in \mathcal{T}'. \tag{3.14}$$

$$\begin{aligned} x_{ij} - x_{ik} - x_{jk} &\leq 0, \\ x_{ik} - x_{ij} - x_{jk} &\leq 0, \\ x_{jk} - x_{ij} - x_{ik} &\leq 0 \text{ for all } i,j,k \in \mathcal{T}'. \end{aligned} \tag{3.15}$$

together with the trivial inequalities (3.10). We define the reduced metric cone RMET($K_n$) as the one defined by inequalities (3.15) and the trivial inequalites (3.10). In [55], we prove the following theorem,

**Theorem 3.3.1** *The projections of* RMETP($K_n$) *and* RMET($K_n$) *on* $\mathbb{Q}^E$ *give respectively* METP($G$) *and* MET($G$).

which implies that RMETP($K_n$) and RMET($K_n$) are extended formulations for respectively METP($G$) and MET($G$) We can see that this result is of particular interest for the case of sparse graphs, when $m = O(n)$, since this yields much more compact formulations of size $O(n^2)$ variables and constraints. Clearly such reduction in problem size can be exploited computationally e.g. in the solution of the max-cut problem, due to the induced reduction in computational effort devoted to solving the linear relaxations in each node of the Branch-and-Bound tree.

However, beyond its computational interest, this result raises the natural and challenging new question of whether it is possible to further reduce the size of a linear formulation for MET($G$) and METP($G$) in sparse graphs, or at least some subclasses of sparse graphs. And, since $\Omega(m)$ is a lower bound to the size (number of variables and constraints) of any linear formulation (just considering the non negativity constraints, assuming connectivity), is it possible to achieve linear size $O(m) = O(n)$, at least for some subclasses of sparse graphs.

As a first step towards answering such polyhedral issues, in [54], we provides a positive answer to this last question by showing that for the subclass of series-parallel graphs (for which the max-cut problem can be solved in linear time, see [6]), it is possible to refine the reduced formulations to come up with linear-size formulations for MET($G$) and METP($G$). To the best of our knowledge, this is the first subclass of graphs enjoying linear-size representations for the associated metric polyhedra.

### 3.3.5 Extension of the result for graph partitioning

The Graph Partitioning problem (GP) is a fundamental problem in combinatorial optimization. We consider the basic version of the problem defined in Garey and

Johnson's book [32] (problem ND14) which can be stated as follows. Given an undirected graph $G = (V, E)$ with node set $V = \{1, \ldots, n\}$, weights $w_i \in \mathbb{Z}_+$ for each node $i \in V$, lengths $l_e \in \mathbb{Z}_+$ for each edge $e \in E$ and a positive integer $K$, find a partition of $V$ into disjoint sets (or clusters) $V_1, \ldots, V_k$ such that $\sum_{i \in V_j} w_i \leq W$ for $j = 1, \ldots, k$ minimizing the sum of the lengths of the edges whose endpoints are in different clusters (i.e. the $k$-Cut defined by the partition).

It was shown in [37] that the problem is NP-hard. As there are many variants of graph partitioning, let us call this version of graph partitioning problem, *the basic graph partitioning problem*. Note that the number of the clusters $k$ is not an input data in the basic version in Garey and Johnson's book, i.e. $k$ is not fixed. It is interesting to see that the extended formulation of metric cone $\mathrm{MET}(K_n)$ defined on the complete graph together with 0/1 constraints form an integer formulation for all the possible partitions of the node set of $G$, i.e.

$$
\begin{aligned}
x_{ij} - x_{ik} - x_{jk} &\leq 0, \\
x_{ik} - x_{ij} - x_{jk} &\leq 0, \\
x_{jk} - x_{ij} - x_{ik} &\leq 0 \text{ for all } i, j, k \in \mathcal{T} \\
x_{ij} &\in \{0, 1\} \text{ for all edge } ij \in E_n.
\end{aligned}
$$

In fact, the variables $x_{ij}$ represent the relation: $x_{ij} = 0$ meaning that that two nodes $i$ and $j$ belong to same cluster and $x_{ij} = 1$ sinon. The constraints $x_{ij} - x_{ik} - x_{jk} \leq 0$ represent the transitive relation: if $i$ and $k$ belong to the same cluster (i.e. $x_{ik} = 0$) and so do $j$ and $k$ (i.e. $x_{jk} = 0$) then $i$ and $j$ must belong to the same cluster (i.e. $x_{ij} = 0$). In particular, Chopra [15] has shown that the above integer formulation coincides with its linear programming relaxation when $G$ is a series-parallel graph.

To have a complete integer formulation for (GP), we should model the knapsack constraints on the clusters, i.e. $\sum_{i \in V_j} w_i \leq W$ for $j = 1, \ldots, k$. For this, we express

the $n$ following constraints

$$\sum_{i \in V \setminus \{j\}} v_i(1 - x_{ij}) + v_j \leq W,$$

which express the knapsack constraint for the cluster containing $j$ for each node $j \in V$. Hence, we obtain the following integer formulation, called IGP, for (GP).

$$
\begin{array}{lll}
 & \min \limits_{e \in E} l_e x_e & \\
 & x_{ij} - x_{ik} - x_{jk} \leq 0, & \\
IGP & x_{ik} - x_{ij} - x_{jk} \leq 0, & \\
 & x_{jk} - x_{ij} - x_{ik} \leq 0 & \text{for all } i,j,k \in \mathcal{T}, \\
 & \sum \limits_{i \in V \setminus \{j\}} w_i(1 - x_{ij}) + w_j \leq W & \text{for all } j \in V, \\
 & x_{ij} \in \{0,1\} & \text{for all edge } ij \in E_n.
\end{array}
$$

This model for Graph Partitioning has been used in several works in the litterature [60], [44]. Other works strengthen this model with additional variables modelling the relation node-cluster [36], [25], [39]. In [49], we study the question if our reduction [55] for MET($K_n$) can be still applied to IGP even in the presence of additional knapsack constraints? Our answer is "yes" and moreover the reduction can be applied for a generic class of additional constraints that inludes the knapsack constraints. Precisely, the addtional constraints can be of the generic form $f(\chi(C)) \leq 0$ for each cluster $C \subset V$ in the partition where $\chi(C) \in \{0,1\}^n$ is the incidence vector associated to $C$ and $f : \{0,1\}^n \Rightarrow \mathbb{Q}$ is a monotone nondecreasig pseudoboolean function ($f(\chi(C_1)) \leq f(\chi(C_2))$ if $C_1 \subseteq C_2$). For instance, the knapsack constraint $\sum_{i \in V_j} w_i \leq K$ can be viewed as $f(\chi(V_j)) \leq 0$ with $f(\chi(C)) = \sum_{i \in C} w_i - K$ for any $C \subset V$.

Our result in [49] can be summarized as follows. Let GIGP be the integer program IGP where the knapsack constraints are replaced by the generic constraints described above. Let RGIGP be the reduced size program which constraints the same con-

straints as GIGP except that the triangle constraints are defined on $\mathcal{T}'$ instead of $\mathcal{T}$. We prove the following theorem.

**Theorem 3.3.2** *GIGP coincides with RGIGP and more strongly, the continuous relaxations of IGP and RGIGP coincide.*

We have conducted numerical results solving IGP and RGIP the integer formulations and the reduced integer formulations of the basic version of graph partitioning problem. We focus on instances of random sparse graphs on which the positive effect of the reduction is very clear. Note that in Table 3.1, LGP and RLGP denote respec-

Table 3.1: Computation results of (IGP) and (RGIP) for random sparse graphs.

| graph types | n,m | IGP | LGP | RIGP | LRGP | Cont.Rlx |
|---|---|---|---|---|---|---|
| | | CPU | CPU | CPU | CPU | GAP(%) |
| random graph | 22, 120 | 58.8 | 0.54 | 16.75 | 0.15 | 9.3 |
| random graph | 25, 150 | 120.2 | 1.13 | 34.34 | 0.7 | 12.5 |
| random graph | 27, 150 | 379.5 | 1.25 | 139.7 | 0.49 | 10.8 |
| random graph | 30, 200 | 1436.8 | 2.79 | 458.5 | 0.98 | 8.4 |
| random graph | 35, 250 | - | 7.97 | 945.6 | 2.65 | 10.5 |
| random graph | 40, 280 | - | 14.5 | 3326.9 | 4.97 | 10.2 |
| random graph | 45, 200 | - | 19.0 | 3884.2 | 1.47 | 12.7 |
| random graph | 50, 200 | - | 35.88 | 9024.8 | 2.32 | 11.4 |

tively the linear programming relaxations of IGP and RIGP. We can see in Table 3.1 that the reduction allows to reduce the solving time of both integer formulation and its linear programming relaxation at least three times.

Let us now consider another instance of the generic constraint with applications in network design [34], [11] where the function $f$ is quadratic and not linear as for knapsack constraints. Suppose now that the nodes in $V$ represent the sites to be connected by a network and for every edge $ij \in E$, the length $l_{ij}$ is replaced by $t_{ij}$ the estimated traffic volume between the site $i$ and the site $j$. The *graph partitioning problem under capacity constraints* aims at partitionning the set of sites $V$ into local networks (i.e. clusters) in order to maximize the traffics inside these local networks. This objective is equivalent to minimize the traffic inter-clusters and hence this is indeed a graph partitioning problem. Usually, each local network $C$ is equiped a multiplexer for multiplexing the *incident traffic of* $C$ which is the sum of the traffics inside $C$ and the

traffics between $C$ and the rest of the network. As the capacity to handle traffics of the multiplexer is limited, the volume of the incident traffic of $C$ should be bounded by a constant $W$. Given a node $h \in V$, we can express this *capacity constraint* on $C_h$, the cluster containing $h$ as follows:

$$\sum_{ij \in E} t_{ij} - \sum_{\substack{ij \in E: \\ i \neq h, j \neq h}} t_{ij} x_{ih} x_{jh} \leq W.$$

The first term of the left-hand side is a constant representing the total volume of traffic over the network, the second term is a quadratic term which represents the volume of traffic not incident to $C_h$ ($x_{ih} = x_{jh} = 1$ means that both $i$ and $j$ do not belong to $C_h$, or equivalently the traffic between $i$ and $j$ is not an incident traffic of $C_h$). We can verify that the function $f(\chi(C_h)) = \sum_{ij \in E} t_{ij} - \sum_{\substack{ij \in E: \\ i \neq h, j \neq h}} t_{ij} x_{ih} x_{jh} - W$ is monotone decreasing. Hence, we can formulate the graph partitioning under capacity constraints by an 0/1 formulation containing the triangle inequalies and the capacity constraints. Let QIP denote the latter and let QP be its continuous relaxation. As in this case, our reduction applies, let RQIP be the reduced 0/1 formulation expressing the triangle inequalities on $\mathcal{T}'$ instead of $\mathcal{T}$ and let RQP its continuous relaxation. We have conducted numerical experiments solving the graph partitioning under capacity constraints using QIP and RQIP. Note that as the capacity constraints are quadratic non-convex, we have had to linearize them using the classical Fortet's linearization method [26]. It may not the best method to linearize the models but it is sufficient for the comparison of QIP to RQIP. As shown in Table 3.2, we obtain similar results as for the basic graph partitioning problem, i.e. the reduction allows to reduce the solving time of both integer formulation and its continuous relaxation at least three times.

Table 3.2: Computation results of (QIP) and (RQIP) for random graphs.

| n,m | (QIP) | (QP) | (RQIP) | (RQP) | Cont.Rlx |
|---|---|---|---|---|---|
| | CPU | CPU | CPU | CPU | GAP(%) |
| 22, 120 | 68.3 | 0.64 | 17.8 | 0.15 | 15.0 |
| 25, 125 | 156.9 | 1.1 | 37.3 | 0.32 | 10.4 |
| 27, 150 | 344.5 | 1.4 | 107.3 | 0.55 | 13.7 |
| 30, 200 | 1944.2 | 3.64 | 438.5 | 1.23 | 14.8 |
| 35, 200 | - | 9.56 | 1025.9 | 3.65 | 16.2 |
| 40, 250 | - | 20.67 | 3853.4 | 6.73 | 18.1 |
| 45, 200 | - | 28.39 | 3328.5 | 1.81 | 16.7 |
| 50, 200 | - | 45.31 | 11038.6 | 2.88 | 15.2 |
| 60, 150 | - | 327.63 | 5291.4 | 4.36 | 12.3 |
| 70, 140 | - | - | 10038.2 | 5.05 | 11.8 |
| 80, 120 | - | - | 8615.7 | 9.49 | 14.1 |

# Chapter 4

# Extended formulations: improved compactness and linearization

In the last section of the previous chapter, we have reduced of the number of triangle inequalities in extended formulations for metric polyhedra from $O(n^3)$ to $O(nm)$. In particular, for sparse graphs with $m = O(n)$, we have then a model of $O(n^2)$ variables and constraints. This is interesting since it allows to consider exact solutions for instances with thousand nodes, i.e. $n \approx 1000$ which is not possible currently with $O(n^3)$ variables or $O(n^3)$ constraints.

In this chapter, we will consider some 0/1 non-convex quadratic formulations (the quadratic constraints are not convex) with $O(n^2)$ variables and study methods to linearize the quadratic constraints without increasing asymptotically the number of variables. In particular, we will consider two variants of graph partitioning problems:

- the graph partitioning under capacity constraints problem which has been defined and considered in Section 3.3.5,

- the basic graph partitioning problem (as defined in Section 3.3.5) with uncertainty on the knapsack constraint.

## 4.1 Graph partitioning under capacity constraints (GPCC)

We consider the GPCC problem defined in Section 3.3.5. In particular, in constrast with the numerical experiments conducted in Section 3.3.5 on random sparse graphs, in this section, we focus on instances on $K_n$, the complete graph of order $n$. In this case, the reduction of the triangle constraints in Section 3.3.5 does not applied since $O(nm) = O(n^3)$ for $K_n$. Hence we will recall here the 0/1 quadratic formulation for GPCC with triangle inequalities defined for $\mathcal{T}$ the set of all triples of $V_n$:

$$
\begin{aligned}
(QIP - GPCC \min & \sum_{ij \in E_n} t_{ij} x_{ij} && && (4.1)\\
\text{s. t.: } & x_{ij} + x_{ik} - x_{jk} \geq 0 && (i,j,k) \in \mathcal{T} && (4.2)\\
& x_{ij)} + x_{jk} \geq x_{ik} && (i,j,k) \in \mathcal{T} && (4.3)\\
& x_{ik} + x_{jk} \geq x_{ij} && (i,j,k) \in \mathcal{T} && (4.4)\\
& \sum_{ij \in E} t_{ij} - \sum_{\substack{ij \in E: \\ i \neq h, j \neq h}} t_{ij} x_{ih} x_{jh} \leq W && h \in V_n && (4.5)\\
& x_e \in \{0,1\} && e \in E_n. && (4.6)
\end{aligned}
$$

This formulation contains $O(n^2)$ 0/1 variables, $O(n^3)$ triangle inequalities (4.2), (4.3), (4.4) and $n$ capacity constraints (4.5). The latters are quadratic and a priori non-convex. In the next section, we consider two methods for the linearization of these constraints:

- the Fortet's classical linearization method [26] making use of $O(n^3)$ addtional variables and constraints,

- and our method of linearization by projection in [11] which keeps working only with original binary variables $x$ and generating inequalties for the linearization by cutting-plane.

### 4.1.1 More Compact Linearization by Projection

To expose the techniques, we consider a problem in the standard form

$$\begin{aligned} & \min\ c^T x \\ \text{s. t.:}\ & Ax \leq b, \\ & \sum_{i=1}^{p} \sum_{j=i+1}^{p} q_{ij}^k x_i x_j + {d^k}^T x \leq q^k, \quad k = 1, \ldots, m, \\ & x \in \{0,1\}^p, \end{aligned} \tag{4.7}$$

Clearly, QIGPCC can be cast into this form. We consider here two solution techniques for QIGPCC based on reformulating (4.7) as a mixed-integer linear program. The first one is the classical linearization of (4.7) in an extended space obtained by introducing a variable $y_{ij}$ to represent each product $x_i x_j$:

$$\begin{aligned} & \min\ c^T x \\ \text{s. t.:}\ & Ax \leq b, \\ & \sum_{i=1}^{p} \sum_{j=i+1}^{p} q_{ij}^k y_{ij} + {d^k}^T x \leq q^k, \quad k = 1, \ldots, m, \\ & y_{ij}^-(x_i, x_j) \leq y_{ij} \leq y_{ij}^+(x_i, x_j), \quad 1 \leq i < j \leq p, \\ & (x, y) \in \mathbb{Z}^p \times \mathbb{R}^{\frac{p(p-1)}{2}}, \end{aligned} \tag{4.8}$$

where for $1 \leq i < j \leq p$, $y_{ij}^-(x_i, x_j) = \max\{0, x_i + x_j - 1\}$ and $y_{ij}^+(x_i, x_j) = \min\{x_i, x_j\}$.

A drawback of this extended formulation is the large number of variables it requires. Note that in our graph partitioning problem, for a complete graph with $n$ vertices the quadratic formulations we study already have $\mathcal{O}(n^2)$ variables and the extended formulation $\mathcal{O}(n^3)$ variables. This can become rapidly unpractical. The second technique that we detail below consists in projecting the extended formulation back in the space of $x$-variables. The technique was originally proposed in [**?**] but it is particularly simple to perform in our case.

**Projection Technique**

Based on the fact that if $x_i, x_j \in \{0,1\}$, $x_i x_j = \min\{x_i, x_j\} = \max\{x_i + x_j - 1, 0\}$, the left-hand-side of the quadratic constraints in (4.7) can be rewritten as:

$$\phi^k(x) := \sum_{\substack{i=1,\dots,p \\ j=i,\dots,p \\ q_{ij}^k<0}} q_{ij}^k \min\{x_i, x_j\} + \sum_{\substack{i=1,\dots,p \\ j=i,\dots,p \\ q_{ij}^k>0}} q_{ij}^k \max\{x_i + x_j - 1, 0\} + {d^k}^T x. \tag{4.9}$$

Note that $\phi^k(x)$ is a piecewise linear function with at most $2^{\frac{p(p-1)}{2}}$ pieces; furthermore, it is convex. The program (4.7) can be reformulated using $\phi^k(x)$ as:

$$\begin{aligned} &\min\ c^T x \\ \text{s. t.:}\ &Ax \le b, \\ &\phi^k(x) \le q^k, \quad k = 1, \dots, m, \\ &x \in \{0,1\}^p. \end{aligned} \tag{4.10}$$

Using the convexity of $\phi^k(x)$, (4.10) can be cast as a mixed integer linear program by expanding each constraint $\phi^k(x) \le q^k$ into $2^{\frac{p(p-1)}{2}}$ linear constraints (enumerating all possible values for each minimum and maximum). We denote by $P$ the set of feasible solutions to the continuous relaxation of (4.10), and by $Q$ the set of feasible solutions to the continuous relaxations of (4.8).

Next, we show that if we assume some monotonicity properties on the coefficients of the quadratic form, the set $P$ is the projection onto the $x$-space of $Q$. The monotonicity assumption that we make is that for given indices $i$ and $j$ with $1 \le i < j \le p$ all coefficients $q_{ij}^k$ have the same sign for $k = 1, \dots, m$. The projection of $Q$ onto the $x$-space is $\{x \in \mathbb{R}^p : \exists z \in \mathbb{R}^{\frac{p(p-1)}{2}}$, such that $(x, z) \in Q\}$, we denote it by $\text{proj}_x(Q)$. Note that this assumption is in particular verified by all the models we presented for GPCC.

**Theorem 4.1.1** *Suppose that for $k = 1, \dots, m$, $q_{ij}^k$ is either non-negative for all $1 \le i < j \le p$ or non-positive for all $1 \le i < j \le p$, then $P = \text{proj}_x(Q)$.*

Of course a difficulty of formulation (4.10) is the potential number of linear constraint: $m2^{\frac{p(p-1)}{2}}$. Nevertheless, these constraints are very easy to separate and we can use cut generation techniques to optimize (4.10). Given a point $x^* \in \mathbb{R}^n$, we can compute $\phi^k(x^*)$ in a time that is linear in the number of non-zero coefficients $q^k_{ij}$, and deduce from it the most violated linear cut that can be obtained from $\phi^k(x) \leq q^k$. We can then embed this separation procedure in a cutting plane framework to optimize the continuous relaxation of (4.10).

### 4.1.2 Computational comparisions

We present in this section the comparision of our linearization by projection and the Fortet's linearization method. We also include the CPLEX default method for solving QIGPCC in the comparision. The test set is composed of 48 randomly generated instances of graph partitionning problem with capacity constraints with a number of nodes $n$ between 16 and 25. For each values of $n$, 6 instances are generated with randomly chosen nodes: 4 instances where the traffics between two nodes are randomly generated following an uniform distribution, 2 instances where the traffics between two nodes are randomly generated using the rules devised in [**?**] (the traffic is inversly proportional to the distance between two nodes). The former instances tends to be more difficult in practice in our experiment. The traffic limit of the partitions is randomly chosen so that the instance is feasible (it always allow for the solution with each node in a separate partition to be feasible). In Table 4.1, summarized the result of the experiment. For each value of $n$ we report the number of instance solved by each algorithm, the average CPU time in seconds (we count 3 hours for unsolved instances) and the average number of nodes. We excluded from the results the instances that are not solved by any of the two methods. It can be seen from the table that the default algorithm in CPLEX is the less competitive, it can solve only 4 instances out of 37. CPLEX using the extended formulation can solve 34 instances and the branch-and-cut based on the projection can solve all 37 instances. For the 34 instances that can be solved both by the classical linearized and the projected formulation the number of nodes is significantly larger with the projected formulation (20737 on average vs. 6850

| | | projection | | | classical | | | CPLEX | | |
|---|---|---|---|---|---|---|---|---|---|---|
| n | #inst | # sol | CPU | Nodes | # sol | CPU | Nodes | # sol | CPU | Nodes |
| 16 | 6 | 6 | 52 | 6060 | 6 | 243 | 3178 | 2 | 7979 | 75233 |
| 17 | 6 | 6 | 82 | 6137 | 6 | 412 | 5827 | 1 | 9011 | 244233 |
| 18 | 6 | 6 | 117 | 10993 | 6 | 594 | 4246 | 1 | 10800 | 41646 |
| 19 | 6 | 6 | 900 | 66267 | 6 | 4885 | 29860 | 2 | 10800 | 30388 |
| 20 | 5 | 5 | 336 | 4542 | 5 | 1142 | 2080 | 0 | 10800 | 8404 |
| 21 | 4 | 4 | 1388 | 5047 | 3 | 5083 | 2435 | 0 | 10800 | 27079 |
| 22 | 2 | 2 | 6175 | 128823 | 1 | 8683 | 6584 | 0 | 10800 | 42341 |
| 25 | 2 | 2 | 2869 | 23982 | 1 | 9254 | 1342 | 0 | 10800 | 16921 |

Table 4.1: Statistics on complete solution by branch-and-cut.

with the classical linnearized formulation) but the CPU time is on average 3.5 times faster (it is always faster and more than 5 times faster for 16 of the 37 problems).

## 4.2 Stochastic Basic Graph Partitioning Problem

### 4.2.1 Binary Second Order Cone Formulation

We consider a variant of the basic graph partitioning problem involving knapsack constraints with Gaussian random coefficients. Let us restate first the integer formulation IGP for the basic graph partitioning problem described in Section 3.3.5. For the sake of clarity, in this section, the meaning of the variables $x_{ij}$ for $ij \in E_n$ will the opposite of it's meaning in Sections 3.3.5 and 4.1. Precisely, in this section,$x_{ij} = 1$ if $i$ and $j$ belong to the same cluster and $x_{ij} = 0$, otherwise. Hence, IGP in 3.3.5 can

be restated equivalently as follows.

$$
(\mathrm{I})\begin{cases}
\min & \displaystyle\sum_{(i,j)\in E} t_{ij}(1-x_{ij}) & \\
\text{s. t.:} & x_{ij}+x_{ik}\le 1+x_{jk}, & \forall (i,j,k)\in\mathcal{T}\\
& x_{ij}+x_{jk}\le 1+x_{ik}, & \forall (i,j,k)\in\mathcal{T}\\
& x_{ik}+x_{jk}\le 1+x_{ij}, & \forall (i,j,k)\in\mathcal{T}\\
& \displaystyle\sum_{i\in V\setminus\{j\}} w_i x_{ij}+w_j\le W & \forall j\in V_n\\
& x_{ij}\in\{0,1\} & (i,j)\in E_n
\end{cases}
$$

We consider a Stochastic version of the Basic Graph Partitioning (SBGP) where the node weights $w_i$ for $i \in V_n$ (see Section 3.3.5 for defintion)are uncertain and follow a multivariate Gaussian distribution with given mean and covariance matrix. Give a probability level $\varepsilon \in [0,1]$, the knapsack constraints in (i) can be formulated as chance constraints of the form $P(\sum_{v\in V_i} w_v \le W) \ge 1-\varepsilon$ for each cluster $V_i$ with $i = 1,\ldots,k$. This method was introduced in [14] which is one of the standard methods for handling uncertainty in optimization. Moreover, we assume that the chance constraints are individual and the probability level $\varepsilon$ is less than 0.5. In this case, the chance constraints can be reformulated as *Second Order Cone Constraints*(SOCC) as follows. Let us suppose that the probability distribution of node weights is a multivariate Gaussian law with given means $(\bar{w}_i)_{1\le i\le n}$ and covariance matrix $(\sigma_{ij})_{1\le i,j\le n}$. We therefore reformulate the chance constraints by the SOCCs as follows. For all clusters $i = 1,\ldots,n$,

$$
\sum_{u=1}^{n} x_{ui}\bar{w}_u + \gamma\sqrt{\sum_{u=1}^{n}\sigma_{uu}x_{ui}^2 + 2\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sigma_{uv}x_{ui}x_{vi}} \le W \tag{4.11}
$$

where $\gamma = \mathcal{F}^{-1}(1-\varepsilon)$, $\mathcal{F}$ denoting the cumulative distribution function of $\mathcal{N}(0,1)$ (e.g $\gamma \simeq 1.685$ for $\varepsilon = 0.05$). In the proposed model, the various chance constraints on the various clusters are considered as individual chance constraints.

Then the resulting model for SBGP is the following Binary SOCP program :

$$\begin{cases} \min & \displaystyle\sum_{(i,j)\in E} t_{ij}(1-x_{ij}) & \\ \text{s. t.:} & x_{ij}+x_{ik}\le 1+x_{jk}, & \forall (i,j,k)\in\mathcal{T} \\ & x_{ij}+x_{jk}\le 1+x_{ik}, & \forall (i,j,k)\in\mathcal{T} \\ & x_{ik}+x_{jk}\le 1+x_{ij}, & \forall (i,j,k)\in\mathcal{T} \\ & \displaystyle\sum_{u=1}^{n} x_{ui}\bar{w}_u+\gamma\sqrt{\sum_{u=1}^{n}\sigma_{uu}x_{ui}^2+2\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sigma_{uv}x_{ui}x_{vi}}\le W & i=1,\dots,n \\ & x_{ij}\in\{0,1\} & (i,j)\in E_n \end{cases}$$

As the probablity function $P(\sum_{v\in V_i} w_v \le W)$ is monotone and nonincreasing, i.e. it's opposite is monotone and nondecreasing, our reduction of triangle inequalities in Section 3.3.5 can be applied and we can express the triangle inequalities for the triples of $\mathcal{T}'$ instead for those in $\mathcal{T}$. Hence, we obtain the following binary SOCP for SBGP, denoted by BSOCP:

$$\text{BSOCP}\begin{cases} \min & \displaystyle\sum_{(i,j)\in E} t_{ij}(1-x_{ij}) & \\ \text{s. t.:} & x_{ij}+x_{ik}\le 1+x_{jk}, & \forall (i,j,k)\in\mathcal{T}' \\ & x_{ij}+x_{jk}\le 1+x_{ik}, & \forall (i,j,k)\in\mathcal{T}' \\ & x_{ik}+x_{jk}\le 1+x_{ij}, & \forall (i,j,k)\in\mathcal{T}' \\ & \displaystyle\sum_{u=1}^{n} x_{ui}\bar{w}_u+\gamma\sqrt{\sum_{u=1}^{n}\sigma_{uu}x_{ui}^2+2\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sigma_{uv}x_{ui}x_{vi}}\le W & i=1,\dots,n \\ & x_{uu}=1 & u=1,\dots,n \\ & x_{ij}\in\{0,1\} & (i,j)\in E_n \end{cases}$$

### 4.2.2 More compact linearization using bilinear terms

The continuous relaxation of BSOCP is convex as the SOCCs are convex. Hence, the standard approach to solve BSOCP is a branch-and-bound algorithm using its con-

tinuous relaxation for lower bound estimation at each node of the branch-and-bound tree. However, branch-and-bound algorithms may not be efficient since algorithms for solving continuous SOCP programs should be interior algorithms which prevent any warm-starting. Thus, we also consider the approach that consists of two steps:

- replacing the SOCCs with their quadratic form, i.e.

$$\begin{cases} -2\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}(\bar{w}_u\bar{w}_v - \gamma^2\sigma_{uv})x_{ui}x_{vi} + \sum_{u=1}^{n}(2W\bar{w}_u + \gamma^2\sigma_{uu} - \bar{w}_u^2)x_{ui} \leq W^2 \\ \sum_{u=1}^{n} x_{ui}\bar{w}_u \leq W \end{cases} \tag{4.12}$$

- and linearizing the quadratic constraints.

Note that the convexity is lost in the first step but the linearization in the second step allows to get it back. For the linearization in the second step, we are faced with the same problem as for the graph partitioning under capacity constraint in Section 4.1 that is the number of variables will increase from $O(n^2)$ to $O(n^3)$ if the Fortet's classical linearization is used. We do not use either the projection method as for graph partitioning under capacity constraint in Section 4.1 for the following reasons:

- we want to avoid the use of cutting-plane and focus on instances of SBGP on sparse graphs with $m = O(n)$. In this case, the number of constraints in BSCOP is in $O(n^2)$ that can be sent directly to solvers without using cutting-plane. Using the projection method will force the use of cutting-plane.

- Moreover, the coefficients of the quadratic terms are homogenous in sign. Thus, we do not have te guarantee that the linear programming relaxation given by the projection method has the same quality of the one given by the Fortet's classical method.

In [50], we give a method of linearization based on the one proposed by Sherali et Smith in [59] that keeps the number of variables and cosntraints within $O(n^2)$. The idea is to use an additional variable $\lambda_{ui} = \sum_{v=u+1}^{n}(\bar{w}_u\bar{w}_v - \gamma^2\sigma_{uv})x_{vi}$ for that the

quadratic term becomes $\lambda_{ui}x_{ui}$ which is a bilinear term. The latter is then linearized by using one more additional $z_{ui} = \lambda_{ui}x_{ui}$ with lower and upper bound constraints. We can see that the number of additional variables remains in $O(n^2)$. The strength of the linear programming relaxation depend on the one of lower and upper bound constraints for the variables $z_{ui}$. In [50], we propose two methods to compute upper bounds: the simple method (SS) which take into account only the binary constraints on the variables $x$'s, the improved method (ISS) which in addition taken into account the SOCC's constraints.

Table 4.2.2 presents computational results on instances of random series-parallel graphs (SP) and sparse graphs (RG) comparing four methods to solve BSCOP using the solver CPLEX: BSOCP is the method that sends directly BSOCP to CPLEX, CL is the classical linearization, SS is our method with simple bounds, ISS is our method with improved bound. We can see that ISS the most efficient method which is at least two times faster than BSOCP.

Table 4.2: Comparison of the various solution techniques. Nopt indicates that the exact optimal solution could not be found within the imposed time limit (7200s). In such cases, the value of the relative residual gap is shown in parenthesis.

| Instances | | BSOCP | | CL | | SS | | ISS | |
|---|---|---|---|---|---|---|---|---|---|
| types | n,m | CPU | GAP (Nodes) | CPU | GAP (Nodes) | CPU | GAP (Nodes) | CPU | GAP (Nodes) |
| SP | 25, 40 | 4.3 | 16.2 (6) | 12.6 | 17.0 (11) | 4.4 | 18.3 (22) | 2.2 | 18.3 (15) |
| SP | 30, 50 | 30.8 | 11.4 (17) | 100.9 | 13.1 (142) | 27.5 | 15.6 (366) | 15.6 | 15.6 (254) |
| SP | 35, 60 | 40.4 | 10.5 (32) | 177.3 | 15.4 (336) | 43.5 | 15.9 (829) | 29.3 | 15.9 (599) |
| SP | 40, 65 | 126.3 | 9.8 (59) | 513.4 | 11.3 (573) | 122.4 | 12.7 (1136) | 63.8 | 12.7 (729) |
| SP | 45, 75 | 665.2 | 13.2 (88) | 1539 | 14.6 (612) | 595.0 | 17.2 (1387) | 336.5 | 17.2 (874) |
| SP | 50, 80 | 862.8 | 9.1 (117) | 2238 | 10.4 (935) | 978.7 | 13.3 (1843) | 517.2 | 13.3 (1311) |
| SP | 60, 90 | 3127 | 10.6 (156) | Nopt (6.8%) | 12.3 (1033) | 1844 | 16.5 (2552) | 699.3 | 16.5 (1566) |
| SP | 80, 130 | Nopt (2.4%) | 12.7 (187) | Nopt (7.3%) | 14.8 (1426) | Nopt (2.8%) | 17.3 (3136) | 5273 | 17.3 (3629) |
| | | | (55) | | (474) | | (973) | | (666) |
| RG | 25, 150 | 442.3 | 15.0 (72) | 945.2 | 16.2 (553) | 489.1 | 17.1 (1296) | 226.4 | 17.1 (924) |
| RG | 30, 200 | Nopt (0.2%) | 15.5 (127) | Nopt (4.1%) | 16.1 (1183) | 7111 | 17.9 (6421) | 3123 | 17.9 (4318) |
| RG | 40, 120 | 3612 | 13.2 (196) | Nopt (3.3%) | 14.8 (974) | 3828 | 16.4 (3689) | 1805 | 16.4 (3150) |
| RG | 50, 120 | Nopt (2.3%) | 14.7 (188) | Nopt (8.6%) | 15.6 (2158) | Nopt (2.6%) | 17.2 (4523) | 6006 | 17.2 (5298) |
| RG | 60, 100 | Nopt (3.8%) | 16.2 (209) | Nopt (9.9%) | 17.2 (1542) | Nopt (3.6%) | 19.4 (4175) | 6419 | 19.4 (4941) |

# Conclusions and Perspectives

We have investigated in this thesis on (integer) linear programming formulations for combinatorial optimization problems involving basic structures of graphs like cycles, paths, stars, arbres, forests, cuts, multicuts,..etc. Our efforts focused on the following tasks:

- Using existing natural formulations to derive improved exact or approximation combinatorial algorithms,
- Describing a natural formulation and showing its complexity
- Analyzing more sharply existing natural formulations to improved exact or approximation based on linear programming,
- Studying the TDI-ness of existing natural formulations and derive a compact extended formulation,
- Reducing the size of existing compact extended formulations,
- Designing more compact extended formulation for binary quadratic constraints.

Our research projects for the three next years are the pursuit of the above topics, especially

- We aim at designning a direct combinatorial algorithm for minimum weight $T$-join problem for any $T$. To the best of our knowledge, finding a minimum $T$-join by now need to solve all-pair shortest path and perfect matching problems in auxillary graphs [23]. Note that there are direct algorithms for two special

cases: $T = V$, i.e. the minimum perfect matching problem and $T = \emptyset$, i.e. the problem of detecting a negative cost cycle (our algorithm in Section 1.2).

- We would like to chracterize classes of graph where a linear size extended formulation of the metric polyhedra is possible. This represents extension of our result for series-parallel graphs. We focus on sparse graphs such as grid, planar and nearly planar graphs. Another and alternative research direction is to show that linear size is impossible for these classes of graphs.

- We are also interested in designing efficient algorithms for binary SCOP programs. As we have seen in Section 4.1, although the continuous relaxation of binary SOCP is convexe and strong, it is not efficient when to be used in general branch-and-bound framework. It may be interesting to design a special-purpose branch-and-bound algorithm for binary SOCP.

# Bibliography


[1] *A spanning star forest model for the diversity problem in automobile industry*, 2005.

[2] Arash Asadpour, Michel X. Goemans, Aleksander Madry, Shayan Oveis Gharan, and Amin Saberi. An O(log n/ log log n)-approximation Algorithm for the Asymmetric Traveling Salesman Problem. In Moses Charikar, editor, *SODA*, pages 379–389. SIAM, 2010.

[3] Mourad Baiou and F. Barahona. On the integrality of some facility location polytopes. *SIAM J. Discrete Math.*, 23(2):665–679, 2009.

[4] Egon Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[5] Michel Balinski. Integer Programming: Methods, Uses, Computation. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming*, pages 133–197. Springer, 2010.

[6] F. Barahona. A solvable case of quadratic 0-1 programming. *Discrete Applied Mathematics*, 13(1):23–26, 1986.

[7] F. Barahona. On cuts and matchings in planar graphs. *Mathematical Programming*, 60(1-3):53–68, 1993.

[8] F. Barahona and A. R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36(2):157–173, 1986.

[9] Petra Bauer. The Circuit Polytope: Facets. *Math. Oper. Res.*, 22(1):110–145, 1997.

[10] Daniel Bienstock, Michel X. Goemans, David Simchi-Levi, and David P. Williamson. A note on the prize collecting traveling salesman problem. *Math. Program.*, 59:413–420, 1993.

[11] Pierre Bonami, Viet Hung Nguyen, Michel Klein, and Michel Minoux. On the Solution of a Graph Partitioning Problem under Capacity Constraints. In Ali Ridha Mahjoub, Vangelis Markakis, Ioannis Milis, and Vangelis Th. Paschos,

editors, *ISCO*, volume 7422 of *Lecture Notes in Computer Science*, pages 285–296. Springer, 2012.

[12] Endre Boros, Yves Crama, and Peter L. Hammer. Chvátal Cuts and ODD Cycle Inequalities in Quadratic 0 - 1 Optimization. *SIAM J. Discrete Math.*, 5(2):163–177, 1992.

[13] Endre Boros and Peter L. Hammer. Cut-Polytopes, Boolean Quadric Polytopes and Nonnegative Quadratic Pseudo-Boolean Functions. *Math. Oper. Res.*, 18(1):245–253, 1993.

[14] A. Charnes and W.W. Cooper. Chance-constrained programming. *Management Science*, 6(1):73–79, 1959.

[15] Sunil Chopra. The Graph Partitioning Polytope on Series-Parallel and4-Wheel Free Graphs. *SIAM J. Discrete Math.*, 7(1):16–31, 1994.

[16] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Extended formulations in combinatorial optimization. *Annals OR*, 204(1):97–143, 2013.

[17] Denis Cornaz and V. H. Nguyen. Kőnig's edge-colouring theorem for all graphs. *Oper. Res. Lett.*, 41(6):592–596, 2013.

[18] Gérard Cornuéjols, George Nemhauser, and Laurence A. Wolsey. The Uncapacitated Facility Location Problem . In Pitu Mirchandani and Richard Francis, editors, *Discrete Location Theory*, pages 119–172. Wiley, 1990.

[19] Gérard Cornuéjols and Jean-Michel Thizy. Some facets of the simple plant location polytope. *Math. Program.*, 23(1):50–74, 1982.

[20] M. Dell'Amico, Francesco Maffioli, and P. Varbrand. On prize-collecting tours and the asymmetric travelling salesman problem. *Int. Trans. in Operational Research*, 2(3):297–308, 1995.

[21] M. Deza and M. Laurent. Applications of cut polyhedra — I. *Journal of Computational and Applied Mathematics*, 55(2):191–216, 1994.

[22] Michel Deza, Monique Laurent, and Svatopluk Poljak. The cut cone III: On the role of triangle facets. *Graphs and Combinatorics*, 9(2-4):135–152, 1993.

[23] Jack Edmonds. The Chinese Postman problem. *The Bulletin of the Operatons Research of America*, 13:B–73, 1965.

[24] Jack Edmonds and Ellis L. Johnson. Matching, Euler tours and the Chinese postman. *Math. Program.*, 5(1):88–124, 1973.

[25] Carlos Eduardo Ferreira, Alexander Martin, C. Carvalho de Souza, Robert Weismantel, and Laurence A. Wolsey. The node capacitated graph partitioning problem: A computational study. *Math. Program.*, 81:229–256, 1998.

[26] R. Fortet. L´algebre de Boole et ses applications en recherche operationnelle. *Trabajos de Estadistica*, 11(2):111–118, 1960.

[27] Antonio Frangioni, Andrea Lodi, and Giovanni Rinaldi. New approaches for optimizing over the semimetric polytope. *Math. Program.*, 104(2-3):375–388, 2005.

[28] Alan M. Frieze, Giulia Galbiati, and Francesco Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[29] Toshihiro Fujito. How to trim a MST: A 2-Approximation algorithm for minimum cost-tree cover. *ACM Trans. Algorithms*, 8(2):16, 2012.

[30] Harold N. Gabow. An Efficient Reduction Technique for Degree-Constrained Subgraph and Bidirected Network Flow Problems. In David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas, editors, *STOC*, pages 448–456. ACM, 1983.

[31] Harold N. Gabow. Data Structures for Weighted Matching and Nearest Common Ancestors with Linking. In David S. Johnson, editor, *SODA*, pages 434–443. SIAM, 1990.

[32] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, 1979.

[33] Michel X. Goemans and David P. Williamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. ACM*, 42(6):1115–1145, 1995.

[34] Olivier Goldschmidt, Alexandre Laugier, and Eli V. Olinick. SONET/SDH Ring Assignment with Capacity Constraints. *Discrete Applied Mathematics*, 129(1):99–128, 2003.

[35] C. Helmberg. A Cutting Plane Algorithm for Large Scale Semidefinite Relaxations. pages 223–256, 2004.

[36] Søren Holm and Michael Malmros Sørensen. The optimal graph partitioning problem. *Operations-Research-Spektrum*, 15(1):1–8, 1993.

[37] Laurent Hyafil and Ronald L. Rivest. Graph Partitioning and Constructing Optimal Decision Trees are Polynomial Complete Problems. Technical Report Rapport de Recherche no. 33, IRIA – Laboratoire de Recherche en Informatique et Automatique, Domaine de Voluceau, Rocquencourt 78150 - Le Chesnay, France, 1973.

[38] Volker Kaibel and Kanstantsin Pashkovich. Constructing Extended Formulations from Reflection Relations. In Oktay Günlük and Gerhard J. Woeginger, editors, *IPCO*, volume 6655 of *Lecture Notes in Computer Science*, pages 287–300. Springer, 2011.

[39] Volker Kaibel, Matthias Peinhardt, and Marc E. Pfetsch. Orbitopal fixing. *Discrete Optimization*, 8(4):595–610, 2011.

[40] Safia Kedad-Sidhoum and Viet Hung Nguyen. An exact algorithm for Solving the Ring Star Problem. *Optimization*, 59(1):125–140, 2010.

[41] Jochen Könemann, Goran Konjevod, Ojas Parekh, and Amitabh Sinha. Improved Approximations for Tour and Tree Covers. *Algorithmica*, 38(3):441–449, 2003.

[42] Jochen Könemann, Goran Konjevod, Ojas Parekh, and Amitabh Sinha. Improved Approximations for Tour and Tree Covers. *Algorithmica*, 38(3):441–449, 2003.

[43] Martine Labbé, Gilbert Laporte, Inmaculada Rodríguez Martín, and Juan José Salazar González. The Ring Star Problem: Polyhedral analysis and exact algorithm. *Networks*, 43(3):177–189, 2004.

[44] Martine Labbé and F. Aykut Özsoy. Size-constrained graph partitioning polytopes. *Discrete Mathematics*, 310(24):3473–3493, 2010.

[45] Frauke Liers, Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. Computing Exact Ground States of Hard Ising Spin Glass Problems by Branch-and-Cut. In *New Optimization Algorithms in Physics*, pages 47–69. Wiley-VCH Verlag, 2005.

[46] Jean François Maurras, Thanh Hai Nguyen, and Viet Hung Nguyen. On the linear description of the Huffman trees polytope. *Discrete Applied Mathematics*, 164:225–236, 2014.

[47] Michel Minoux. Discrete Cost Multicommodity Network Optimization Problems and Exact Solution Methods. *Annals OR*, 106(1-4):19–46, 2001.

[48] C. Thach Nguyen, Jian Shen, Minmei Hou, Li Sheng, Webb Miller, and Louxin Zhang. Approximating the spanning star forest problem and its applications to genomic sequence alignment. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 645–654. SIAM, 2007.

[49] Dang Phuong Nguyen, Michel Minoux, Viet Hung Nguyen, Thanh Hai Nguyen, and Renaud Sirdey. Improved compact formulations for a wide class of graph partitioning problems in sparse graphs. *Discrete Optimization*, pages –, 2016.

[50] Dang Phuong Nguyen, Michel Minoux, Viet Hung Nguyen, Thanh Hai Nguyen, and Renaud Sirdey. Stochastic graph partitioning: quadratic versus SOCP formulations. *Optimization Letters*, 10(7):1505–1518, 2016.

[51] Viet Hung Nguyen. A direct dual algorithm for detecting negative cost cycles in undirected graphs. *SIAM J. on Computing*, submitted.

[52] Viet Hung Nguyen. Approximating the Minimum Tour Cover of a Digraph. *Algorithms*, 4(2):75–86, 2011.

[53] Viet Hung Nguyen. A primal-dual approximation algorithm for the Asymmetric Prize-Collecting TSP. *J. Comb. Optim.*, 25(2):265–278, 2013.

[54] Viet Hung Nguyen, Michel Minoux, and Dang Phuong Nguyen. Reduced-size formulations for metric and cut polyhedra in sparse graphs. *Networks*, submitted.

[55] Viet Hung Nguyen, Michel Minoux, and Dang Phuong Nguyen. Improved compact formulations for metric and cut polyhedra. *Electronic Notes in Discrete Mathematics*, 52:125–132, 2016.

[56] Viet Hung Nguyen and Thi Thu Thuy Nguyen. Approximating the asymmetric profitable tour. *IJMOR*, 4(3):294–301, 2012.

[57] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Program.*, 121(2):307–335, 2010.

[58] Paul Seymour. Sum of circuits. In J.A. Bondy and U.S.R. Murty, editors, *Graph Theory and Related Topics*, pages 341–355. Academic Press, 1979.

[59] Hanif D. Sherali and Jonathan Cole Smith. An improved linearization strategy for zero-one quadratic programming problems. *Optimization Letters*, 1(1):33–47, 2007.

[60] Michael M. Sørensen. Facet-defining inequalities for the simple graph partitioning polytope. *Discrete Optimization*, 4(2):221–231, 2007.

[61] David P. Williamson. Analysis of the Held-Karp heuristic for the traveling salesman problem. Technical report, 1990.