UNIVERSITÀ DI PISA

Dipartimento di Matematica
Corso di Laurea Triennale in Matematica

TESI DI LAUREA TRIENNALE

Lagrangian Relaxation and Dantzig-Wolfe Decomposition in Network Optimization Problems

Relatore:

Prof.ssa
Maria Grazia Scutellà

Candidato:

Irene Dovichi

ANNO ACCADEMICO 2021/2022

Contents

Introduction **5**

1 Lagrangian Relaxation **7**

1.1 Elements of Linear Programming . . . 7

1.2 Lagrangian Duality . . . 9

1.2.1 Lagrangian relaxation and inequality constraints . . . 11

1.3 Strength of the Lagrangian Dual . . . 11

1.4 Solving the Lagrangian Dual . . . 14

1.4.1 Subgradient Optimization Technique . . . 15

1.4.2 Subgradient Optimization and Inequality Constraints . . . 16

2 Applications of Lagrangian Relaxation **19**

2.1 Uncapacitated Facility Location (UFL) . . . 19

2.2 Symmetric Traveling Salesman Problem (STSP) . . . 20

2.3 Constrained Shortest Path (CSP) . . . 23

3 Dantzig-Wolfe Decomposition **25**

3.1 Dantzig-Wolfe reformulation of an Integer Linear Problem . . . 25

3.1.1 Uncapacitated Facility Location (UFL) . . . 26

3.2 Solving the Linear Programming Master Problem . . . 29

3.2.1 Symmetric Traveling Salesman Problem (STSP) . . . 34

3.3 Branch and Price Algorithm for 0-1 Problems . . . 37

3.3.1 Branching Rule . . . 38

4 Multicommodity Flows **39**

4.1 Formulation with Arc Flows . . . 39

4.1.1 Optimality Conditions . . . 40

4.1.2 Lagrangian Relaxation . . . 42

4.2 Formulation with Path Flows . . . 44

4.2.1 Optimality Conditions . . . 45

4.2.2 Column Generation Solution Approach . . . 48

5 Relevance of Lagrangian Relaxation and Dantzig-Wolfe Decomposition in Network Design: the Capacitated Facility Location Problem (CFL) 51
5.1 Problem formulation 52
5.2 Lagrangian relaxation of (CFL) 53
5.2.1 Reformulation of the Lagrangian Dual problem 55
5.3 Primal Master Problem of (CFL) 55

Bibliography 57

Introduction

In this thesis we present two general solution methodologies for addressing Integer Linear Programming problems, both from a theoretical and a practical point of view, showing how the methods introduced apply to particular network optimization problems.

The first methodology presented is the Lagrangian relaxation, which consists in dropping some of the constraints of the original problem and including them in the objective function with suitable dual variables, also referred to as Lagrangian multipliers. We will describe the Lagrangian relaxation technique for an Integer Linear Programming problem stated in minimization form, both in the case of equality and inequality constraints. We will observe that, in both cases, the Lagrangian relaxation provides a lower bound on the optimal objective function value of the problem. The problem of finding the best lower bound as we vary the Lagrangian multipliers is known as the Lagrangian Dual problem. We will prove that the optimal objective function value of the Lagrangian Dual problem is as sharp as the lower bound obtained by the Linear Programming relaxation, which is why this methodology has become popular in applications, since it is often easier to obtain the Lagrangian Dual bound instead of the Linear Programming bound. Moreover, in some cases the bound returned by the Lagrangian Dual is better than the one returned by the Linear Programming relaxation. This lower bound can then be exploited within exact approaches to Integer Linear Programming problems, such as Branch and Bound and Branch and Price. In the second chapter, we present three problems of location analysis and network optimization to which Lagrangian relaxation can be efficiently applied: the Uncapacitated Facility Location Problem, the Symmetric Traveling Salesman Problem and the Constrained Shortest Path Problem.

The second methodology presented to approach Integer Linear Programming problems is via decomposition, and consists in reformulating the original problem in an equivalent way, known as the Master Problem; this procedure is known as the Dantzig-Wolfe decomposition of the given problem. In particular, we present the Dantzig-Wolfe reformulation of the Uncapacitated Facility Location Problem. Afterwards, we present an algorithm, known as the Column Generation algorithm, that solves the Linear Programming relaxation of the Master Problem. In general, the Master Problem and its Linear Programming relaxation have fewer constraints than the original problem, but a very large number of variables. The idea behind the Column Generation algorithm is to avoid the explicit enumeration of all the variables of the problem, starting from a subset of variables and adding at each iteration only those that can improve the objective function. Then, we show how this algorithm applies to the case of the Symmetric Traveling Salesman Problem.

In the fourth chapter we focus on a specific class of network flow problems, whose general model is known as the Minimum Cost Multicommodity Flow problem. This model formalizes the situation, very frequent in applications, in which distinct commodities, governed by their own network flow constraints, share the same network and are linked by some joint constraints. We will present two formulations of the Minimum Cost Multicommodity Flow problem, using arc flows and path flows, and we will illustrate how to solve the problem by applying the Lagrangian relaxation in the first case, and using a Column Generation algorithm in the second case.

In the last chapter, we present a formulation of the Capacitated Facility Location Problem, which is an optimization problem with applications in many contexts, and we show how the Lagrangian relaxation and the Column Generation technique can be useful in addressing such a difficult problem.

Chapter 1

Lagrangian Relaxation

Lagrangian relaxation is a general solution strategy for solving optimization problems. It consists in dropping some of the constraints of the problem and including them in the objective function with suitable dual variables, also referred to as Lagrangian multipliers.
In the first section we present some basic results of linear algebra and linear programming that serve as support for the next topics. The second section contains the main definitions of the chapter and the first results on the relaxation methodology. In the last two sections we analyze how sharp is the bound obtained with the Lagrangian relaxation and we introduce an algorithm to compute such a bound by solving an associated Lagrangian Dual Problem.

1.1 Elements of Linear Programming

We assume that the *primal problem* is formulated as:

$$(P) \quad min\Big\{\sum_{j=1}^{q} c_j x_j : \sum_{j=1}^{q} a_{ij} x_j \geq b_i \;\; \forall i = 1, \dots, p, \;\; x_j \geq 0 \;\; \forall j = 1, \dots, q\Big\}$$

where $x = (x_1, \dots, x_q)$ is the vector of decision variables, $c = (c_1, \dots, c_q)$ is the cost vector, $A = (a_{ij})_{\substack{i \in \{1,\dots,p\} \\ j \in \{1,\dots,q\}}}$ and $b = (b_1, \dots, b_p)$ are the matrix of the coefficient constraints and the right-hand-side vector, respectively.
The *dual problem* associated with (P) according to the Linear Programming Duality is:

$$(D) \quad max\Big\{\sum_{i=1}^{p} b_i y_i : \sum_{i=1}^{p} a_{ij} y_i \leq \; c_j \;\; \forall j = 1, \dots, q, \;\; y_i \geq 0 \;\; \forall i = 1, \dots, p\Big\}.$$

We refer to these problems as the *symmetric form* of duality. If some constraint of the primal problem is in the equality form, the associated dual variable must be unrestricted in sign. Similarly, if some primal variable is unrestricted in sign, the

associated dual constraint must be in the equality form.
We state two relevant results about the primal-dual pair without reporting the proof [2].

Theorem 1.1.1. (Weak Duality Theorem)
Suppose that (P) and (D) admit feasible solutions. For any feasible solution x of the primal problem and any feasible solution y of the dual problem, we have: $\sum_{i=1}^{p} b_i y_i \leq \sum_{j=1}^{q} c_j x_j$.

Theorem 1.1.2. (Strong Duality Theorem)
If one between the primal and the dual problem has a finite optimal solution, so does the other one and the pair has the same optimal objective function values.

Definition 1.1.1. A pair (x, y) of feasible solutions for the primal and dual problems, respectively, is said to satisfy the *Complementary Slackness Property* if

$$\begin{cases} y_i\Big(\sum_{j=1}^{q} a_{ij}x_j - b_i\Big) = 0 \quad \forall i = 1, \dots, p \\ x_j\Big(c_j - \sum_{i=1}^{p} a_{ij}y_i\Big) = 0 \quad \forall j = 1, \dots, q \end{cases}$$

In other words, if a primal/dual constraint has a positive slack, the associated dual/primal variable must have value zero. Symmetrically, if a primal/dual variable has a positive value, the dual/primal solution must satisfy the corresponding constraint as an equality.

Theorem 1.1.3. (Complementary Slackness Optimality Conditions)
A pair (x, y) of feasible solutions for the primal and dual problems, respectively, is optimal if and only if it satisfies the complementary slackness property.

Proof. Assume that x and y are optimal primal and dual solutions, respectively. Multiplying the ith primal constraint by y_i and adding the p constraints side by side yields: $\sum_{i=1}^{p} y_i b_i \leq \sum_{i=1}^{p} y_i\Big(\sum_{j=1}^{q} a_{ij}x_j\Big)$. Multiplying the jth dual constraint by x_j and adding the q constraints side by side yields: $\sum_{j=1}^{q} x_j\Big(\sum_{i=1}^{p} a_{ij}y_i\Big) \leq \sum_{j=1}^{q} c_j x_j$. Therefore we have that: $\sum_{i=1}^{p} y_i b_i \leq \sum_{i=1}^{p}\sum_{j=1}^{q} a_{ij}y_i x_j \leq \sum_{j=1}^{q} c_j x_j$. The strong duality theorem implies that these inequalities must be equalities. We can rewrite the first equality as: $\sum_{i=1}^{p} y_i\Big(\sum_{j=1}^{q} a_{ij}x_j - b_i\Big) = 0$, and we notice that each term of this sum is nonnegative since x and y are feasible solutions. We conclude that each term must be zero, hence the first condition of the complementary slackness property is satisfied. Similarly, the second equality leads to the satisfaction of the second condition.

Assume now that x and y satisfy the complementary slackness property. Adding the first conditions of the complementary slackness property yields: $\sum_{j=1}^{q}\sum_{i=1}^{p} a_{ij}y_ix_j = \sum_{i=1}^{p} b_iy_i$ and adding the second conditions yields: $\sum_{j=1}^{q}\sum_{i=1}^{p} a_{ij}y_ix_j = \sum_{j=1}^{q} c_jx_j$. Therefore $\sum_{i=1}^{p} b_iy_i = \sum_{j=1}^{q} c_jx_j$, and the weak duality theorem ensures that the pair (x, y) is optimal. $\square$

1.2 Lagrangian Duality

In order to describe the Lagrangian relaxation methodology, we consider an optimization problem formulated as:

$$\begin{aligned} (P) \quad & min\ c^tx \\ & Ax = b \\ & x \in X \end{aligned}$$

where the objective function is linear, there are a certain number of complicating constraints, i.e., $Ax = b$, and the vector of decision variables x lies in a set X, which in this thesis will be a subset of $\mathbb{Z}^q$, where q is the dimension of vector x.

Definition 1.2.1. The *Lagrangian relaxation* of the problem (P) with respect to the constraints $Ax = b$ is:

$$\begin{aligned} (P(u)) \quad & min\ c^tx + u^t(Ax - b) \\ & x \in X \end{aligned}$$

where u is a real vector unrestricted in sign with as many components as the number of complicating constraints. The components of u are called *Lagrangian multipliers.* We refer to the function $L(u) = min\{c^tx + u^t(Ax - b) : x \in X\}$ as the *Lagrangian function.*

Now, we recall a general definition:

Definition 1.2.2. A problem (R) $min\{f(x) : x \in T \subseteq \mathbb{R}^n\}$ is a *relaxation* of (P) $min\{g(x) : x \in U \subseteq \mathbb{R}^n\}$ if:

- $\diamond$ $U \subseteq T$
- $\diamond$ $f(x) \leq g(x) \quad \forall x \in U$

Proposition 1.2.1. $(P(u))$ is a relaxation of (P) for all u.

Proof. For each u the feasible region of $(P(u))$ is X, which contains the feasible region of (P): $\{x : Ax = b,\, x \in X\}$. The objective value of $(P(u))$ is lower than or equal to the one of (P) for all feasible solutions to (P): since $Ax = b$, for any vector u we have $min\{c^t x : Ax = b,\, x \in X\} = min\{c^t x + u^t(Ax - b) : Ax = b,\, x \in X\}$. Now, note that $min\{c^t x + u^t(Ax - b) : Ax = b,\, x \in X\} \geq min\{c^t x + u^t(Ax - b) : x \in X\}$ since we are taking the minimum of the same function over X, which is a wider set than $\{x : Ax = b,\, x \in X\}$. □

Corollary 1.2.1.1. The value $L(u)$ is a lower bound on the optimal objective function value of (P) for all u.

Definition 1.2.3. The *Lagrangian Dual problem* associated with the problem (P) is:

$$(LD) \quad L^* = \max_u L(u)$$

and it is used to find the sharpest Lagrangian lower bound on the optimal objective value of (P).

Corollary 1.2.1.2. (Weak Duality)
The optimal objective function value L^* of (LD) is always a lower bound on the optimal objective function value of (P).

We are ready to compare the objective function values of (LD) and (P) for any choices of the Lagrangian multipliers u and any feasible solution x of (P).

Theorem 1.2.2. (Optimality Test)

1. Suppose that x satisfies the condition $L(u) = c^t x$ for some u. Then u is an optimal solution of (LD) and x is an optimal solution of (P).

2. If for some choice of u the solution x^* of the Lagrangian relaxation $P(u)$ is feasible for (P), then u is an optimal solution of (LD) and x^* is an optimal solution of (P).

Proof.
1. Since $L(u) \leq L^* \leq min\{c^t x : Ax = b,\, x \in X\} \leq c^t x$, if $L(u) = c^t x$ all these inequalities are equalities. In particular $L^* = L(u)$ and $min\{c^t x : Ax = b, x \in X\} = c^t x$.

2. Since x^* is a solution of $P(u)$ we have that $L(u) = c^t x^* + u^t(Ax^* - b)$ and since it is feasible for (P) we have that $Ax^* = b$. Therefore, $L(u) = c^t x^*$ and 1. implies the thesis.

□

In other words, we have a certificate of optimality whenever, for some feasible solution x of (P) and some Lagrangian multipliers u, it is $L(u) = c^t x$ and this makes the Lagrangian relaxation useful.

1.2.1 Lagrangian relaxation and inequality constraints

If the constraints of the optimization problem (P) are formulated in inequality form $Ax \leq b$, the Lagrangian Dual problem becomes $(LD)\ L^* = \underset{u \geq 0}{max}\ L(u)$. In addition to requiring the multipliers to be nonnegative, however, there is another substantial difference:

Theorem 1.2.3. (Optimality Test with inequality constraints)
Suppose that we apply Lagrangian relaxation to the optimization problem

$$(P) \quad min\{c^t x : Ax \leq b,\ x \in X\}$$

and that for some choice of $u \geq 0$ the solution x^* of the Lagrangian relaxation is feasible for (P). If x^* satisfies the complementary slackness property $u^t(b - Ax^*) = 0$, then x^* is an optimal solution of (P).

Proof. Since x^* is a solution of $P(u)$ we have that $L(u) = c^t x^* + u^t(Ax^* - b)$ and since $u^t(Ax^* - b) = 0$ we have that $L(u) = c^t x^*$. Since x^* is feasible for (P) by Theorem 1.2.2 the thesis follows. $\square$

1.3 Strength of the Lagrangian Dual

In this section we will show that the lower bound obtained by the Lagrangian relaxation technique is at least as sharp as that obtained by using a Linear Programming relaxation. As a result this methodology has become very popular in applications, since in some cases it is easier to obtain the Lagrangian Dual bound instead of the Linear Programming bound. Moreover, for some problems the bound returned by the Lagrangian Dual can be better than the one provided by the Linear Programming relaxation.

Theorem 1.3.1. Suppose that we apply the Lagrangian relaxation technique to a Linear Programming problem

$$(P') \quad min\Big\{c^t x : Ax = b,\ Dx \leq q,\ x \geq 0\Big\}$$

by relaxing the constraints $Ax = b$. Then, the optimal value L^* of the Lagrangian Dual problem equals the optimal objective function value of (P').

Proof. We rewrite (P') as: $(P') \quad min\Big\{c^t x : -Ax = -b,\ -Dx \geq -q,\ x \geq 0\Big\}$ so that its Linear Programming dual problem is: $(D') \quad max\Big\{-\pi^t b - \gamma^t q : -\pi^t A - \gamma^t D \leq c^t,\ \gamma \geq 0\Big\}$. The complementary slackness property for this pair of problems is

$$\begin{cases} \pi^t\big(-Ax + b\big) + \gamma^t\big(-Dx + q\big) = 0 \\ \big(c^t + \pi^t A + \gamma^t D\big)x = 0 \end{cases}.$$

Suppose that x^* is an optimal solution of (P') and π^*, γ^* are optimal solutions of (D') associated, respectively, with the constraints $Ax = b$ and $Dx \leq q$. By Theorem 1.1.3 the solutions x^*, π^*, γ^* satisfy the complementary slackness property:

(i) $\gamma^{*t}\big(-Dx^* + q\big) = 0$, since $Ax^* = b$ (x^* is feasible for (P'))

(ii) $\big(c^t + \pi^{*t}A + \gamma^{*t}D\big)x^* = 0$

in addition to the dual feasibility condition: (iii) $-\pi^{*t}A - \gamma^{*t}D \leq c^t$. We have that $L(\pi^*) = min\{c^t x + \pi^{*t}(Ax - b) : Dx \leq q,\, x \geq 0\}$ and we notice that x^* is feasible for this problem because it is feasible for (P'), whose feasible region is wider. We also notice that a certain vector x achieves the value $L(\pi^*)$ if and only if it achieves the optimal value of the Linear Programming problem:

$$(P'') \quad min\Big\{(c^t + \pi^{*t}A)x : -Dx \geq -q,\, x \geq 0\Big\}$$

in fact $L(\pi^*) = min\{(c^t + \pi^{*t}A)x - \pi^{*t}b : Dx \leq q,\, x \geq 0\}$ and since $\pi^{*t}b$ is constant for $L(\pi^*)$, it can be ignored when optimizing. The dual problem of (P'') is

$$(D'') \quad max\Big\{-\gamma^t q : \gamma \geq 0,\, -\gamma^t D \leq c^t + \pi^{*t}A\Big\}$$

and by Theorem 1.1.3 we know that given x'' solution of (P'') and γ'' solution of (D'') such that they satisfy the complementary slackness property, they are optimal. The complementary slackness property for the pair $(P'') - (D'')$ is:

$$\begin{cases} \gamma^t\big(-Dx + q\big) = 0 \\ \big(c^t + \pi^{*t}A + \gamma^t D\big)x = 0 \end{cases}$$

and we notice that $x'' = x^*$ is feasible for (P'') because it is feasible for (P'), whose feasible region is wider. Similarly, $\gamma'' = \gamma^*$ is feasible for (D'') because of the condition (iii) $-\pi^{*t}A - \gamma^{*t}D \leq c^t$. Moreover, the pair (x^*, γ^*) satisfies the complementary slackness property for $(P'') - (D'')$ since it is exactly the conditions (i) and (ii). We conclude that x^* is optimal for (P''), which is equivalent to say that it achieves $L(\pi^*)$: $L(\pi^*) = (c^t + \pi^{*t}A)x^* - \pi^{*t}b = c^t x^* + \pi^{*t}(Ax^* - b) = c^t x^*$. Consequently, Theorem 1.2.2 implies that $L^* = L(\pi^*) = c^t x^*$, that is the optimal objective function value of (P').

$\square$

From now to the end of the section, suppose that we apply the Lagrangian relaxation to a discrete optimization problem (P) $min\Big\{c^t x : Ax = b,\, x \in X\Big\}$ and that the discrete set X is defined as $X = \Big\{x : Dx \leq q,\, x \geq 0 \text{ and integer}\Big\}$.
More precisely, we assume that $X = \{x_1, \ldots, x_N\}$ is a finite non empty set and we recall that $x = \sum_{n=1}^{N} \lambda_n x_n$ is a *convex combination* of $x_1, \ldots, x_N$ if $\lambda_n \geq 0 \quad \forall n = 1, \ldots, N$ and $\sum_{n=1}^{N} \lambda_n = 1$. Let $Conv(X)$ denote the *convex hull* of X, that is the set of all convex combinations of X. Now, we state some properties of $Conv(X)$ without reporting the proof:

Remark.

1. $Conv(X)$ is a polyhedron.
2. The extreme points of $Conv(X)$ all lie in X and if we optimize a linear objective function over $Conv(X)$, there will be some optimal solution in X.
3. $Conv(X) \subseteq \big\{x : Dx \leq q,\, x \geq 0\big\}$

Theorem 1.3.2. The optimal objective function value L^* of the Lagrangian Dual problem equals the optimal objective function value of the linear program

$$min\big\{c^t x : Ax = b,\, x \in Conv(X)\big\}.$$

Proof. We notice that, for each u, the Lagrangian relaxation problem $L(u) = min\{c^t x + u^t(Ax - b) : x \in X\}$ is equivalent to the problem $L(u) = min\{c^t x + u^t(Ax - b) : x \in Conv(X)\}$ because the latter is a relaxation of the former and because of 2. in the previous remark. Furthermore, 1. in the previous remark tells us that $L(u) = min\{c^t x + u^t(Ax - b) : x \in Conv(X)\}$ is a Linear Programming problem and we can interpret this problem as a Lagrangian relaxation of (CP) $min\{c^t x : Ax = b,\, x \in Conv(X)\}$. By Theorem 1.3.1 the optimal value L^* of the Lagrangian Dual problem equals the optimal objective function value of the linear program (CP) $min\{c^t x : Ax = b,\, x \in Conv(X)\}$. □

Theorem 1.3.3. When applied to an Integer Linear Programming problem stated in minimization form, the lower bound obtained by the Lagrangian relaxation technique is as sharp as the lower bound obtained by the Linear Programming relaxation, that is:

$$\mathring{z} = min\{c^t x : Ax = b,\, Dx \leq q,\, x \geq 0\} \leq L^*$$

Proof. The problem is stated as (P) $min\big\{c^t x : Ax = b,\, x \in X\big\}$, with $X = \big\{x : Dx \leq q,\, x \geq 0,\, x \in \mathbb{Z}\big\}$. By 3. in the previous remark, we deduce that the set of the feasible solutions of (CP) $min\{c^t x : Ax = b,\, x \in Conv(X)\}$ is contained in the set of the feasible solutions of (LP) $min\{c^t x : Ax = b,\, Dx \leq q,\, x \geq 0\}$. Since optimizing the same objective function over a smaller set cannot improve the objective function value, we get the thesis. □

Definition 1.3.1. We refer to the problem (CP) $min\{c^t x : Ax = b,\, x \in Conv(X)\}$ as the *Convexified Relaxation* of the problem (P).

We show that, under certain conditions, the Lagrangian bound will equal the Linear Programming bound.

Definition 1.3.2. The Lagrangian relaxation is said to satisfy the *integrality property* if, in case of finite optimal value, it has an integer optimal solution for every choice of the objective function coefficients, even if we relax the condition that x must be integer.

Observation 1.3.4. When the Lagrangian relaxation of the problem (P) satisfies the integrality property, the problems $min\{c^tx + u^t(Ax - b) : Dx \leq q,\, x \geq 0 \,\text{and integer}\}$ and $min\{c^tx + u^t(Ax - b) : Dx \leq q,\, x \geq 0\}$ have the same optimal objective function values for every choice of the Lagrangian multipliers u.

Theorem 1.3.5. If the Lagrangian relaxation of the problem (P) satisfies the integrality property, then $\mathring{z} = L^*$.

Proof. By definition of integrality property, the problem $min\{c^tx+u^t(Ax-b) : Dx \leq q,\, x \geq 0\}$ has an integer optimal solution for every choice of the objective function coefficients. Therefore, every extreme point of the region $\{x : Dx \leq q,\, x \geq 0\}$ must be integer, otherwise we could find some coefficients for which the problem has a noninteger optimal solution. We deduce that $\{x : Dx \leq q,\, x \geq 0\} = Conv(\{x : Dx \leq q,\, x \geq 0 \,\text{and integer}\}) = Conv(X)$. Then, $\big\{x : Ax = b,\, Dx \leq q,\, x \geq 0\big\} = \big\{x : Ax = b,\, x \in Conv(X)\big\}$. The former is the set of feasible solutions of the Linear Programming relaxation (LP) and the latter is the set of feasible solutions of the Convexified Relaxation (CP). Since we optimize the same objective function over the same region, we have the thesis. □

1.4 Solving the Lagrangian Dual

We consider a discrete optimization problem in the form $(P)\ \ min\{c^tx : Ax = b,\, x \in X\}$ and again we assume that the set $X = \{x_1, \dots, x_N\}$ is non empty and finite. By relaxing the constraints $Ax = b$, we obtain the Lagrangian function $L(u) = min\{c^tx + u^t(Ax - b) : x \in X\}$ and we have that $L(u) \leq c^tx_n + u^t(Ax_n - b) \quad \forall n = 1, \dots, N$ and for each u.
Therefore, the Lagrangian Dual problem can be rewritten as:

$$\begin{aligned}(LD)\quad L^* &= \max_u L(u)\\ &= \max_u\Big\{\min_{x\in X}\big\{c^tx + u^t(Ax - b)\big\}\Big\}\\ &= \max_u\Big\{\min_{n=1\dots N}\big\{c^tx_n + u^t(Ax_n - b)\big\}\Big\}\end{aligned}$$

which is equivalent to the Linear Programming problem:

$$(LD)\quad max\big\{w : w \leq c^tx_n + u^t(Ax_n - b) \quad \forall n = 1, \dots, N\big\}$$

where u is unrestricted in sign and $w \in \mathbb{R}$ is an auxiliary variable.

This rewriting of the problem suggests that the Lagrangian Dual consists in maximizing the piecewise linear concave function $L(u)$, as exemplified in Figure 1.1 for the case of single Lagrangian multiplier:

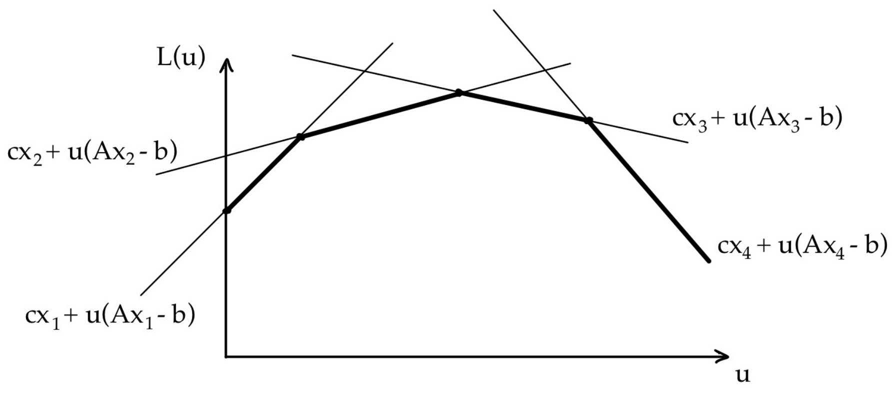

Figure 1.1: Lagrangian function for $X = \{x_1, \ldots, x_4\}$ and single multiplier u

1.4.1 Subgradient Optimization Technique

In order to solve the Lagrangian Dual problem, we present in this section the Subgradient Algorithm. Normally, the subgradient is defined for convex functions, while in the case of concave functions the supergradient is defined. Anyway, in order to simplify the presentation, we will talk about subgradient in the following.

Definition 1.4.1. A *subgradient* at u of a concave function $f : \mathbb{R}^m \to \mathbb{R}$ is a vector $\gamma(u) \in \mathbb{R}^m$ such that $f(x) \leq f(u) + \gamma(u)^t(x - u) \quad \forall x \in \mathbb{R}^m$.

Remark. The function f is differentiable at u if and only if f admits a unique subgradient at u.

In general $L(u)$ is not everywhere differentiable because it is a piecewise linear function, but since it is concave we can use the concept of subgradient in place of the concept of gradient in order to maximize it.
The subgradient algorithm consists in computing a sequence of $(u_k)_{k\in\mathbb{N}}$ such that $L(u_k)$ converges to an optimal solution. The algorithm scheme is:

1. Initialization: $k = 0$, we set $u = u_0$ where u_0 is a given input vector
2. Compute $L(u_k)$ and let $x_k \in X$ be a vector that achieves $L(u_k)$
3. Let $\gamma(u_k) = Ax_k - b$ be a subgradient at u_k of $L(u)$
4. If $\gamma(u_k) = 0$ then STOP, u_k is an optimal solution to the Lagrangian Dual
5. Else compute $u_{k+1} = u_k + \theta_k\gamma(u_k)$, where θ_k is the step length at the kth iteration
6. $k = k + 1$ and go back to step 2.

Lemma 1.4.1. At step 3. the vector $\gamma(u_k) = Ax_k - b$ is a subgradient at u_k of $L(u)$.

Proof. For any u, $L(u) = min\{c^t x + u^t(Ax - b) : x \in X\} \leq c^t x_k + u^t(Ax_k - b)$ since $x_k \in X$, and $L(u_k) = min\{c^t x + u_k^t(Ax - b) : x \in X\} = c^t x_k + u_k^t(Ax_k - b)$ for how we chose x_k, i.e., x_k is a minimizer of $L(u_k)$. Then $L(u) - L(u_k) \leq (u - u_k)^t(Ax_k - b) \quad \forall u$. □

The difficulty lies in choosing the step lengths $\{\theta_k\}_k$: if we choose them too small, the algorithm wouldn't move from the current point and if we choose them too large we could overtake the optimal solution.
Picking lengths such that $\theta_k \to 0$, for $k \to +\infty$, and $\sum_{k=1}^{\infty} \theta_k \to \infty$ ensures the convergence of the method. For example, we could choose $\theta_k = \frac{1}{k}$, but the convergence is too slow to be of practical interest.

Therefore, we can proceed as follows: we assume that $x_k \in X$ achieves $L(u_k)$ and that x_k continues to solve the Lagrangian relaxation as we vary u, meaning that we make a linear approximation to $L(u)$: $r(u) = c^t x_k + u^t(Ax_k - b)$. We also assume to know the optimal value L^* of the Lagrangian Dual problem, then we move in the subgradient direction $\gamma(u_k)$ until the value of the linear approximation equals L^*. We set the step length θ_k so that: $r(u_{k+1}) = c^t x_k + u_{k+1}^t(Ax_k - b) = L^*$, and since we have that: $u_{k+1} = u_k + \theta_k(Ax_k - b)$, we get:

$$\begin{aligned} r(u_{k+1}) &= c^t x_k + u_k^t(Ax_k - b) + \theta_k \|Ax_k - b\|^2 \\ &= L(u_k) + \theta_k \|Ax_k - b\|^2 \\ &= L^* \end{aligned}$$

We deduce that:

$$\theta_k = \frac{L^* - L(u_k)}{\|Ax_k - b\|^2}$$

however, since we do not know the value L^* we usually use the following expression:

$$\theta_k = \frac{\lambda_k(UB - L(u_k))}{\|Ax_k - b\|^2}$$

where UB is an upper bound on the optimal objective function value of the problem (P) and λ_k is a scalar strictly between 0 and 2.

1.4.2 Subgradient Optimization and Inequality Constraints

If the optimization problem that we consider is in the form (P) $min\{c^t x : Ax \leq b,\ x \in X\}$ we have seen in Section 1.2.1 that the Lagrangian multipliers must be nonnegative. We note that the formula $u_{k+1} = u_k + \theta_k(Ax_k - b)$ might cause one or more of the components of u_{k+1} to be negative. Therefore, we modify the formula as: $u_{k+1} = \left[u_k + \theta_k(Ax_k - b)\right]^+$. All other steps of the subgradient procedure remain unchanged compared to problems with equality constraints. For problems with both equality and inequality constraints, we use a mixture of the two versions of the algorithm: for the components of u_k corresponding to the inequality constraints we check if the formula for u_{k+1} would give a negative value and in that case we set 0 as value.

Chapter 2

Applications of Lagrangian Relaxation

In this chapter we illustrate some applications of the Lagrangian relaxation technique in location analysis and network optimization.

2.1 Uncapacitated Facility Location (UFL)

Suppose that $N = \{1, \dots, n\}$ is a set of depots and $M = \{1, \dots, m\}$ is a set of clients. Suppose also that there is a fixed cost f_j associated with the use of depot j and that the unit transportation from depot j of the client i's order costs c_{ij}. We study the problem of deciding which depots to open and organizing which depot serves each client in order to minimize the sum of the fixed and transportation costs. We introduce the binary variables y_j, $\forall j \in N$, that assume value 1 if the jth depot is used and 0 otherwise. The fraction of the client i's order satisfied from depot j is represented by the variable x_{ij}, $\forall i \in M$, $j \in N$. The following Mixed Integer Linear Programming model formulates the considered problem:

$$(UFL) \quad min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n} f_j y_j$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in M \tag{2.1}$$

$$x_{ij} \le y_j \qquad \forall i \in M,\ j \in N \tag{2.2}$$

$$x_{ij} \ge 0 \qquad \forall i \in M,\ j \in N \tag{2.3}$$

$$y_j \in \{0, 1\} \qquad \forall j \in N \tag{2.4}$$

Constraints (2.1) represent the satisfaction of the demand of each client. Constraints (2.2) and (2.3) express the fact that if the depot j isn't used, then it can't satisfy any fraction of client i's order, while if it's used it could satisfy a certain fraction of the demand.

It has been proved that (UFL) is a $\mathcal{NP}-hard$ problem, therefore, in practice, it may be convenient to address this problem with the Lagrangian relaxation methodology. If we dualize the constraints (2.1), since they are equality constraints, the Lagrangian multipliers vector u is unrestricted in sign. The corresponding Lagrangian relaxation is:

$$(UFL(u)) \quad min \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+u_i)x_{ij}+\sum_{j=1}^{n}f_jy_j-\sum_{i=1}^{m}u_i$$

$$x_{ij} \le y_j \qquad \forall i \in M,\, j \in N \tag{2.5}$$

$$x_{ij} \ge 0 \qquad \forall i \in M,\, j \in N \tag{2.6}$$

$$y_j \in \{0,1\} \qquad \forall j \in N \tag{2.7}$$

We observe that, for each u, $(UFL(u))$ decomposes in n subproblems, one for each depot, i.e., for each $j \in N$:

$$\begin{aligned}(UFL_j(u)) \quad z_j(u) = min \sum_{i=1}^{m}(c_{ij}+u_i)x_{ij}+f_jy_j\\ x_{ij} \le y_j \qquad \forall i \in M\\ x_{ij} \ge 0 \qquad \forall i \in M\\ y_j \in \{0,1\}\end{aligned}$$

Each $(UFL_j(u))$ can be solved by inspection:

- ◇ if $y_j = 0$ then $x_{ij} = 0\ \forall i \in M$, therefore $z'_j(u) = 0$
- ◇ if $y_j = 1$ then $x_{ij} = \begin{cases} 1 & if \quad c_{ij}+u_i < 0 \\ 0 & otherwise \end{cases}$ $\forall i \in M$ (this means that all profitable clients are served), therefore $z''_j(u) = \sum_{i=1}^{m} min\{c_{ij}+u_i,\, 0\} + f_j$.

We deduce that $z_j(u) = min\{z'_j(u),\, z''_j(u)\} = min\big\{0,\, \sum_{i=1}^{m} min\{c_{ij}+u_i,\, 0\} + f_j\big\}$ for each $j \in N$. Then, we conclude that $z(u) = \sum_{j=1}^{n} z_j(u) - \sum_{i=1}^{m} u_i$. And ultimately: $(LD) \quad L^* = \max_u z(u)$.

2.2 Symmetric Traveling Salesman Problem (STSP)

Suppose that $N = \{1,\dots,n\}$ is a set of cities and a salesman must visit each city exactly once and then return to the starting point. The time taken to travel from

city i to city j is c_{ij}. The distance between two cities is the same in each opposite direction, hence it is said to be a symmetric problem. We want to find the order in which the salesman should visit the cities, i.e., his tour, with the aim of minimizing the time required.
In order to formulate the problem, we introduce the binary variables x_{ij}, $\forall i, j \in N$, $i \neq j$, that assume value 1 if the salesman goes directly from city i to city j and 0 otherwise. We define E as the set of edges that connect the cities, $E(S)$ is the set of edges with both endpoints in $S \subseteq N$, $\delta(k)$ is the set of edges incident to node k and $\delta(S, N \setminus S)$ is the set of edges with one endpoint in S and the other in $N \setminus S$. $(STSP)$ can then be modeled by means of the following Integer Linear Programming formulation:

$$(STSP) \quad min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\sum_{(i,j)\in\delta(k)} x_{ij} = 2 \qquad \forall k \in N \tag{2.8}$$

$$\sum_{(i,j)\in E(S)} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq N,\ 2 \leq |S| \leq |N| - 1 \tag{2.9}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{2.10}$$

The constraints (2.8) are called *degree constraints* and represent the arrival and the departure of the salesman from each city. The constraints (2.9) are called *subtour elimination constraints* because they assure that no more that one cycle will be present in any solution.

Proposition 2.2.1. The constraints (2.8) can be replaced by:

$$\begin{cases} \sum\limits_{(i,j)\in\delta(k)} x_{ij} = 2 & \forall k \in N \setminus \{1\} \\ \sum\limits_{(i,j)\in E} x_{ij} = n \end{cases}$$

Proof. It is evident that $\forall k \in N \setminus \{1\}$ the constraints are unchanged, whereas for the node 1 we have:

$$\begin{aligned} \sum_{(i,j)\in E} x_{ij} &= \frac{1}{2} \sum_{k\in N} \sum_{(i,j)\in\delta(k)} x_{ij} \\ &= \frac{1}{2} \sum_{k\in N\setminus\{1\}} \sum_{(i,j)\in\delta(k)} x_{ij} + \frac{1}{2} \sum_{(i,j)\in\delta(1)} x_{ij} \\ &= n - 1 + \frac{1}{2} \sum_{(i,j)\in\delta(1)} x_{ij} \end{aligned}$$

where the last equality is due to the fact that $\sum\limits_{(i,j)\in\delta(k)} x_{ij} = 2$ for each $k \in N \setminus \{1\}$, and $N \setminus \{1\}$ has $n-1$ terms.

Then: $\sum\limits_{(i,j)\in E} x_{ij} = n \iff \frac{1}{2}\sum\limits_{(i,j)\in\delta(1)} x_{ij} = 1 \iff \sum\limits_{(i,j)\in\delta(1)} x_{ij} = 2.$ □

Proposition 2.2.2. Half of the constraints in (2.9) are redundant.

Proof. We notice that for any feasible solution we have:

$$\begin{aligned}
|S| - \sum_{(i,j)\in E(S)} x_{ij} &= \frac{1}{2}\sum_{s\in S}\sum_{(i,j)\in\delta(s)} x_{ij} - \sum_{(i,j)\in E(S)} x_{ij} \\
&= \frac{1}{2}\sum_{\substack{(i\in S\wedge j\in N\setminus S)\\ \vee(j\in S\wedge i\in N\setminus S)\\ (i,j)\in E}} x_{ij} \\
&= \frac{1}{2}\sum_{(i,j)\in\delta(S,N\setminus S)} x_{ij} \qquad (*)
\end{aligned}$$

where the first equality is explained by the constraints $\sum\limits_{(i,j)\in\delta(k)} x_{ij} = 2 \quad \forall k \in N.$
For the second equality we observe that: $\delta(s) = \{(i,j) \in E : i = s \vee j = s\}$ and if $(i,j) \in E(S)$, then $(i,j) \in \{(i,j) \in \delta(s),\, s \in S\}$.
Moreover, since $\delta(S, N \setminus S) = \delta(N \setminus S, S)$ we get:

$$|S| - \sum_{(i,j)\in E(S)} x_{ij} = \frac{1}{2}\sum_{(i,j)\in\delta(S,N\setminus S)} x_{ij} = \frac{1}{2}\sum_{(i,j)\in\delta(N\setminus S,S)} x_{ij} = |N\setminus S| - \sum_{(i,j)\in E(N\setminus S)} x_{ij}$$

where the last equality can be explained by reasoning backwards in $(*)$:

$$\begin{aligned}
\frac{1}{2}\sum_{(i,j)\in\delta(N\setminus S,S)} x_{ij} &= \frac{1}{2}\sum_{\substack{(i\in N\setminus S\wedge j\in S)\\ \vee(j\in N\setminus S\wedge i\in S)\\ (i,j)\in E}} x_{ij} \\
&= \frac{1}{2}\sum_{s\in N\setminus S}\sum_{(i,j)\in\delta(s)} x_{ij} - \sum_{(i,j)\in E(N\setminus S)} x_{ij} \\
&= |N \setminus S| - \sum_{(i,j)\in E(N\setminus S)} x_{ij}
\end{aligned}$$

Then:

$$|S| + \sum_{(i,j)\in E(N\setminus S)} x_{ij} = |N \setminus S| + \sum_{(i,j)\in E(S)} x_{ij} \le |N \setminus S| + |S| - 1$$

where the last inequality is due to constraints (2.9).
So, we have found that: $\sum\limits_{(i,j)\in E(N\setminus S)} x_{ij} \le |N\setminus S| - 1.$ This is exactly what we wanted

to show, in fact if $S \subseteq N,\ 1 \leq |S| \leq |N|-1$, then $N \setminus S \subseteq N,\ |N|-1 \geq |N \setminus S| \geq 1$, so $N \setminus S$ is itself a feasible set for constraints (2.9) and it's sufficient to control that the constraint is valid for S or $N \setminus S$. □

In light of the last proposition, we can consider only the node sets S such that $1 \notin S$. We dualize the degree constraints for the nodes different than 1. Since they are equality constraints, the vector u is unrestricted in sign.
Using the first proposition we rewrite the problem as:

$$(STSP) \quad min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\sum_{(i,j)\in\delta(k)} x_{ij} = 2 \qquad \forall k \in N \setminus \{1\} \tag{2.11}$$

$$\sum_{(i,j)\in E} x_{ij} = n \tag{2.12}$$

$$\sum_{(i,j)\in E(S)} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq N \setminus \{1\},\ |S| \geq 2 \tag{2.13}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{2.14}$$

The corresponding Lagrangian relaxation is:

$$(STSP(u)) \quad min \sum_{(i,j)\in E} (c_{ij} + u_i + u_j)x_{ij} - 2\sum_{k=2}^{n} u_k$$

$$\sum_{(i,j)\in E} x_{ij} = n \tag{2.15}$$

$$\sum_{(i,j)\in E(S)} x_{ij} \leq |S| - 1 \qquad \forall S \subseteq N \setminus \{1\},\ |S| \geq 2 \tag{2.16}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{2.17}$$

We define:

Definition 2.2.1. A *1-tree* is a subgraph consisting of two edges that incide in node 1, and the edges of a tree on nodes $\{2, \ldots, n\}$.

It is possible to observe that the feasible solutions of $(STSP(u))$ are the 1-trees. Finding a minimum weight 1-tree is an easy problem, then the presented Lagrangian relaxation methodology is of interest in addressing $(STSP)$.

2.3 Constrained Shortest Path (CSP)

Suppose that a directed graph $G = (N, E)$ is given. Assume that $c_{ij} \geq 0$ is the cost associated with the arc (i, j), and l_{ij} is the length of the arc (i, j), $\forall (i, j) \in E$.

There is a *source node* s and a *destination node* t. Finally, L is a given upper bound. We study the problem of finding the minimum cost path from s to t such that its length is lower than or equal to L.
We introduce the binary variables x_{ij}, $\forall (i,j) \in E$, that assume value 1 if (i,j) belongs to the path and 0 otherwise. Using such variables the problem can be formulated via the following ILP model:

$$(CSP) \quad min \sum_{(i,j)\in E} c_{ij}x_{ij}$$

$$\sum_{(i,j)\in FS(i)} x_{ij} - \sum_{(j,i)\in BS(i)} x_{ji} = \begin{cases} 1 & if\ i = s \\ -1 & if\ i = t \\ 0 & otherwise \end{cases} \qquad \forall i \in N \tag{2.18}$$

$$\sum_{(i,j)\in E} l_{ij}x_{ij} \leq L \tag{2.19}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{2.20}$$

We dualize the constraint (2.19) and since it is an inequality constraint, the vector u (that has a single component) must be nonnegative. The corresponding Lagrangian relaxation is:

$$(CSP(u)) \quad min \sum_{(i,j)\in E} (c_{ij} + u\, l_{ij})x_{ij} - Lu$$

$$\sum_{(i,j)\in FS(i)} x_{ij} - \sum_{(j,i)\in BS(i)} x_{ji} = \begin{cases} 1 & if\ i = s \\ -1 & if\ i = t \\ 0 & otherwise \end{cases} \qquad \forall i \in N \tag{2.21}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{2.22}$$

We can interpret $(CSP(u))$ as a shortest path problem with modified costs compared to the originals. In fact we now associate the new cost $(c_{ij} + u\, l_{ij})$ with each arc (i,j), which is as we had given a price to the length of the edges. Since the shortest path problem can be solved efficiently, the Lagrangian relaxation is useful in this application.

Chapter 3

Dantzig-Wolfe Decomposition

Some combinatorial optimization problems, such as crew scheduling, vehicle routing, and the cutting stock problem, can be intractable due to their computational complexity. In this chapter we present a way of approaching integer linear problems via decomposition, which consists in reformulating the problem considered in an equivalent way, known as the *Master Problem*, and solving its Linear Programming relaxation. In particular, we exemplify such methodologies by showing their application to two of the optimization problems previously reviewed, i.e., the Uncapacitated Facility Location, and the Symmetric Traveling Salesman, respectively. The procedure adopted for solving the Linear Programming relaxation of the Master Problem is called *Column Generation algorithm*, and represents a general methodology for solving large-scale linear programs which are characterized by a very large number of variables. The peculiarity of this procedure is that it avoids the explicit enumeration of all the variables, starting from a subset of variables and adding at each iteration those that can improve the objective function. Finally, in the last section we address the case where the Linear Programming solution is not integral, which leads to a *Branch and Price* algorithm for 0-1 problems.

We assume that the Integer Linear Programming problem considered has a feasible region X that can be written as a subset of the intersection of certain sets X^k: $X = \cap_{k=1}^{K} X^k$. The constraints that define the sets X^k are independent; there will be, however, a certain number of *joint constraints* that link together the different sets of variables.

3.1 Dantzig-Wolfe reformulation of an Integer Linear Problem

In this section, we consider an Integer Linear Programming problem formulated as:

$$
\begin{aligned}
(P) \quad max \; & \sum_{k=1}^{K} c^k x^k \\
& \sum_{k=1}^{K} A^k x^k = b \\
& x^k \in X^k \qquad \forall\, k = 1, \dots, K
\end{aligned}
$$

where the objective function is the sum of K linear functions, and there are K sets of variables defined as $X^k = \{x^k \in \mathbb{Z}_+^{n_k} : D^k x^k \leq d_k\}$[1], for each $k = 1, \dots, K$. Moreover, the different sets of variables are linked by a certain number of *joint constraints*, i.e., $\sum\limits_{k=1}^{K} A^k x^k = b$.

We assume that the sets X^k are finite, namely $X^k = \{x^{kt}\}_{t=1}^{T_k}$ for some integer value T_k, and we notice that the elements of X^k can be rewritten as:

$$
X^k = \Big\{x^k \in \mathbb{R}^{n_k} : x^k = \sum_{t=1}^{T_k} \lambda_{kt}\, x^{kt},\ \sum_{t=1}^{T_k} \lambda_{kt} = 1,\ \lambda_{kt} \in \{0,1\}\ \forall\, t = 1, \dots, T_k\Big\}.
$$

Definition 3.1.1. Substituting each x^k, written as a binary convex combination of the elements x^{kt}, $k = 1, \dots, K$, in the original problem (P) we obtain an equivalent problem:

$$
\begin{aligned}
(PM) \quad max \; & \sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{kt})\, \lambda_{kt} \\
& \sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{kt})\, \lambda_{kt} = b \\
& \sum_{t=1}^{T_k} \lambda_{kt} = 1 \qquad \forall\, k = 1, \dots, K \\
& \lambda_{kt} \in \{0,1\} \qquad \forall\, t = 1, \dots, T_k,\ \ k = 1, \dots, K
\end{aligned}
$$

which is called *Master Problem.*

3.1.1 Uncapacitated Facility Location (UFL)

We present the Dantzig-Wolfe reformulation of the Uncapacitated Facility Location Problem. We keep the same notation of Chapter 2: $N = \{1, \dots, n\}$ is a set of depots and $M = \{1, \dots, m\}$ is a set of clients, there is a fixed cost f_j associated with the use of depot j, and c_{ij} is the unit transportation cost from depot j of client i's order.

[1] In order to ease the notation, we omit the transpose symbol in the objective function, i.e., we write $c^k x^k$ instead of $(c^k)^t x^k$, for each $k = 1, \dots, K$.

We introduce the binary variables y_j, $\forall j \in N$, that assume value 1 if the jth depot is used, and 0 otherwise. For simplicity, here we assume that a depot j either satisfies all of client i's request or does not satisfy it at all. Therefore, we introduce the binary variables x_{ij}, $\forall i \in M$, $j \in N$, which assume value 1 if the jth depot satisfies the client i, and 0 otherwise.
The problem can be formulated by the following Integer Linear Programming model, which represents the *single-source* variant of the model presented in Chapter 2:

$$(UFL) \quad min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j y_j$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in M \tag{3.1}$$

$$\sum_{i=1}^{m} x_{ij} \le m\, y_j \qquad \forall j \in N \tag{3.2}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in M,\ j \in N \tag{3.3}$$

$$y_j \in \{0,1\} \qquad \forall j \in N \tag{3.4}$$

Constraints (3.1) represent the satisfaction of the demand of each client. Constraints (3.2) express the fact that if the depot j isn't used, i.e., $y_j = 0$, then it can't satisfy any clients, i.e., all the related x_{ij} variables are forced to zero, while if it's used it could satisfy all the demand.

We can decompose the problem in n subproblems, i.e., one for each depot. Therefore the indexes $j \in N$ correspond to $k = 1, \ldots, K$ of the previous general presentation. Moreover, here we consider (3.1) as the joint constraints. Specifically, for each depot j we define:

$$X^j = \Big\{(x_{1j} \ldots x_{mj}, y_j) : \sum_{i=1}^{m} x_{ij} \le m\, y_j,\ x_{ij} \in \{0,1\}\ \forall i \in M,\ y_j \in \{0,1\}\Big\}. \tag{3.5}$$

Consider the case where the depot j is used, and let $S \subseteq M$ denote the subset of the clients which are served by depot j. We notice that, if S is nonempty, $\forall i \in M$, $x_{ij} = 1$ if and only if $i \in S$.

Definition 3.1.2. For each $j \in N$, the *incidence vector* of $S_j \subseteq M$ is denoted by $(x^j_{S_j}, 1)$, and it represents the fact that, if $y_j = 1$, the depot j will supply a certain set of clients S_j. The vector $x^j_{S_j}$ has m components x_{ij}, where $x_{ij} = 1$ if and only if $i \in S_j$.

Considering the last definition, we can rewrite:

$$X^j = \{(x^j_{S_j}, 1)\}_{S_j \subseteq M} \cup \{(\underline{0}, 0)\}$$

where $\underline{0}$ is the vector of m components all zero. We associate the following variables with the points of X^j, $\forall j \in N$:

- ◇ $\lambda^j_{S_j} \in \{0,1\}$ is associated with $(x^j_{S_j}, 1)$, for each $S_j \subseteq M$
- ◇ ν^j is associated with $(\underline{0}, 0)$

which correspond to the variables λ_{kt} of the Dantzig-Wolfe reformulation previously introduced. Therefore, equivalently:

$$X^j = \Big\{ x^j = \sum_{S_j \subseteq M} \lambda^j_{S_j}\,(x^j_{S_j}, 1) + \nu^j\,(\underline{0}, 0) : \sum_{S_j \subseteq M} \lambda^j_{S_j} + \nu^j = 1,\ \lambda^j_{S_j} \in \{0,1\}$$
$$\forall S_j \subseteq M,\ \nu^j \in \{0,1\} \Big\}. \tag{3.6}$$

Furthermore, for each $j \in N$ let us define: $A^j = \big(Id_m \,\big|\, 0 \big) \in \mathcal{M}_{m,m+1}$, where Id_m is the $m \times m$ identity matrix.
By using the matrices A^j, $j = 1, \dots, n$, in case of UFL the joint constraints can be rewritten as: $\sum_{j=1}^{n} A^j x^j = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} := b \in \mathbb{R}^m$, and substituting for x^j, written as an element of (3.6), we get:

$$\begin{aligned}
\sum_{j=1}^{n} A^j x^j &= \sum_{j=1}^{n} \Big[\Big(\sum_{S_j \subseteq M} A^j\,(x^j_{S_j}, 1)\,\lambda^j_{S_j} \Big) + A^j\,(\underline{0}, 0)\,\nu^j \Big] \\
&\underset{(*)}{=} \sum_{j=1}^{n} \sum_{\substack{S_j \subseteq M \\ S_j \neq \varnothing}} x^j_{S_j}\,\lambda^j_{S_j} \\
&= \sum_{j=1}^{n} \sum_{\substack{S_j \subseteq M \\ S_j \neq \varnothing}} \lambda^j_{S_j} \begin{pmatrix} x_{1j} \\ \vdots \\ x_{mj} \end{pmatrix}.
\end{aligned}$$

where $(*)$ is explained by noticing that, for $S_j = \varnothing$, $x^j_{\varnothing} = 0$, therefore the associated term is zero. It follows that $\sum_{j=1}^{n} A^j x^j = b$ if and only if $\sum_{j=1}^{n} \sum_{\substack{S_j \subseteq M \\ S_j \neq \varnothing}} \lambda^j_{S_j} = 1$.

Moreover, the objective function can be written as:

$$\begin{aligned}
\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n} f_j y_j &= \sum_{j=1}^{n} c_{1j} x_{1j} + \cdots + c_{mj} x_{mj} + f_j y_j \\
&= \sum_{j=1}^{n} (c_{1j} \dots c_{mj}\ f_j) \cdot \begin{pmatrix} x_{1j} \\ \vdots \\ x_{mj} \\ y_j \end{pmatrix} \\
&\underset{(\circ)}{=} \sum_{j=1}^{n} (c_{1j} \dots c_{mj}\ f_j) \cdot \Big(\sum_{S_j \subseteq M} \lambda^j_{S_j}\,(x^j_{S_j}, 1) + \nu^j\,(\underline{0}, 0) \Big)
\end{aligned}$$

$$= \sum_{j=1}^{n} (c_{1j} \dots c_{mj}\ f_j) \cdot \Big(\sum_{\substack{S_j \subseteq M \\ S_j \neq \varnothing}} \lambda^j_{S_j} \, (x^j_{S_j}, 1) + \lambda^j_{\varnothing} \, (x^j_{\varnothing}, 1) \Big)$$

$$= \sum_{j=1}^{n} (c_{1j} \dots c_{mj}\ f_j) \cdot \Big(\sum_{\substack{S_j \subseteq M \\ S_j \neq \varnothing}} \lambda^j_{S_j} \, (x^j_{S_j}, 1) + \lambda^j_{\varnothing} \, (\underline{0}, 1) \Big)$$

$$= \sum_{j=1}^{n} \Big[\sum_{\substack{S_j \subseteq M \\ S_j \neq \varnothing}} \big(\sum_{i \in S_j} c_{ij} + f_j \big) \lambda^j_{S_j} \, + \, f_j \lambda^j_{\varnothing} \Big]$$

where the equality ($\circ$) holds for certain $\lambda^j_{S_j}$ and ν^j, and it is explained by the fact that (3.5) and (3.6) are equivalent.
In conclusion, the Master Problem of UFL can be formulated as:

$$(PM) \quad min \sum_{j=1}^{n} \Big[\sum_{\substack{S_j \subseteq M \\ S_j \neq \varnothing}} \big(\sum_{i \in S_j} c_{ij} + f_j \big) \lambda^j_{S_j} \, + \, f_j \lambda^j_{\varnothing} \Big]$$

$$\sum_{j=1}^{n} \sum_{\substack{S_j \subseteq M \\ S_j \neq \varnothing}} \lambda^j_{S_j} = 1 \tag{3.7}$$

$$\sum_{S_j \subseteq M} \lambda^j_{S_j} \, + \nu^j = 1 \qquad \forall j \in N \tag{3.8}$$

$$\lambda^j_{S_j} \in \{0, 1\} \qquad \forall S_j \subseteq M, \, j \in N \tag{3.9}$$

$$\nu^j \in \{0, 1\} \qquad \forall j \in N \tag{3.10}$$

3.2 Solving the Linear Programming Master Problem

In this section we present an algorithm that solves the Linear Programming relaxation of the Master Problem. We assume that the problem has m joint constraints, i.e., $b \in \mathbb{R}^m$ in the following formulation, and K convexity constraints. Moreover, we remember that, for each $t = 1, \dots, T_k$, we have $x^{kt} \in \mathbb{R}^{n_k}$, as $k = 1, \dots, K$ varies. Consequently, we deduce that, in the following formulation, A^k is a $m \times n_k$ matrix for $k = 1, \dots, K$. The *Linear Programming Master Problem* differs from that of the Master Problem only for the fact that the λ_{kt} are no more binary variables:

$$\begin{aligned}(LPM)\quad max\ &\sum_{k=1}^{K}\sum_{t=1}^{T_k}(c^k x^{kt})\,\lambda_{kt}\\ &\sum_{k=1}^{K}\sum_{t=1}^{T_k}(A^k x^{kt})\,\lambda_{kt}=b\\ &\sum_{t=1}^{T_k}\lambda_{kt}=1 \qquad \forall\, k=1,\dots,K\\ &\lambda_{kt}\geq 0 \qquad \forall\, k=1,\dots,K,\ t=1,\dots,T_k\end{aligned} \tag{3.11}$$

Proposition 3.2.1.

$$z_{LPM}=max\Big\{\sum_{k=1}^{K}c^k x^k : \sum_{k=1}^{K}A^k x^k=b,\ x^k\in Conv(X^k)\ \forall k=1,\dots,K\Big\}.$$

Proof. (LPM) can be obtained from problem (P) substituting $x^k=\sum_{t=1}^{T_k}\lambda_{kt}\,x^{kt}$, with $\sum_{t=1}^{T_k}\lambda_{kt}=1$ and $\lambda_{kt}\geq 0\ \forall t=1,\dots,T_k$, and it is obvious that this is equivalent to substitute by $x^k\in Conv(X^k)$. □

We are interested in writing (LPM) in matrix form, and making explicit its dual problem.

$$\begin{aligned}(LPM)\quad max\ &c^t\,\lambda\\ &A\,\lambda=\begin{pmatrix}b\\1\end{pmatrix}\\ &\lambda\geq 0\end{aligned} \tag{3.12}$$

where:

- ◇ $\lambda=\big(\lambda_{kt}\big)_{\substack{k=1\dots K\\ t=1\dots T_k}}=\begin{pmatrix}\lambda_{11}\\ \vdots\\ \lambda_{1T_1}\\ \vdots\\ \lambda_{K1}\\ \vdots\\ \lambda_{KT_K}\end{pmatrix}\in\mathbb{R}^{T_1+\dots+T_K}$ is the vector of the variables
- ◇ $c^t=\big(c^1x^{11}\dots c^1x^{1T_1}\ \dots\ c^Kx^{K1}\dots c^Kx^{KT_K}\big)\in\mathbb{R}^{T_1+\dots+T_K}$
- ◇ $A=\left(\begin{array}{c|c|c|c|c|c|c} A^1x^{11} & \dots & A^1x^{1T_1} & \dots & A^Kx^{K1} & \dots & A^Kx^{KT_K}\\ e_1 & \dots & e_1 & \dots & e_K & \dots & e_K\end{array}\right)$ is an element of $\mathcal{M}_{m+K,\,T_1+\dots+T_K}$, and e_k is the kth element of the canonical basis of $\mathbb{R}^K$

◇ $\begin{pmatrix} b \\ 1 \end{pmatrix} \in \mathbb{R}^{m+K}$

and we define $z_{\text{LPM}} = max\left\{ c^t\,\lambda \,:\, A\lambda = \begin{pmatrix} b \\ 1 \end{pmatrix},\ \lambda \geq 0 \right\}$.
In order to formulate the dual problem of (3.12), we associate the variables $(\pi_i)_{i=1}^m$ with the joint constraints, and the variables $(\mu_i)_{i=1}^K$ with the convexity constraints. Setting $y^t = \big(\pi_1 \ldots \pi_m\ \ldots\ \mu_1 \ldots \mu_K\big)$, we get:

$$\begin{aligned} (D) \quad & min\ y^t \cdot \begin{pmatrix} b \\ 1 \end{pmatrix} \\ & y^t\,A \geq c^t \end{aligned}$$

which can be expanded as:

$$\begin{aligned} (D) \quad min \sum_{i=1}^{m} \pi_i b_i + \sum_{k=1}^{K} \mu_k & \\ \pi^t A^1 x^{11} + \mu_1 \geq c^1 x^{11} & \\ \ldots & \\ \pi^t A^1 x^{1T_1} + \mu_1 \geq c^1 x^{1T_1} & \\ \ldots & \\ \pi^t A^K x^{K1} + \mu_K \geq c^K x^{K1} & \\ \ldots & \\ \pi^t A^K x^{KT_K} + \mu_K \geq c^K x^{KT_K} & \end{aligned}$$

Remembering that $X^k = \{x^{kt}\}_{t=1}^{T_k}$, for each $k = 1, \ldots, K$, the T_k constraints of the dual problem related to the kth subproblem can be written as:

$$\pi^t A^k x + \mu_k \geq c^k x \quad \forall x \in X^k$$

or equivalently:

$$c^k x - \pi^t A^k x - \mu_k \leq 0 \quad \forall x \in X^k.$$

The quantities $c^k x - \pi^t A^k x - \mu_k$ are called *reduced prices.*

Observation 3.2.2. The Master Problem (PM), and analogously (LPM), has the advantage of having few constraints with respect to the original problem (P), but the defect of having a huge number of variables. The idea behind the procedure, which we are about to present, for solving (LPM) is to avoid the explicit listing of all the variables of the problem, but rather to generate them only if necessary.

We are therefore ready to present the steps of the algorithm that solves (LPM), considered in the formulation (3.12). In the following we will refer to this procedure as the *Column Generation algorithm*:

1. **<u>Initialization</u>** We consider a matrix $\tilde{A}$ made of a certain number of columns of A, in which we assume there is at least a column $\begin{pmatrix} A^k x^{kt} \\ e_k \end{pmatrix}$ for each k. The vectors $\tilde{c}$ and $\tilde{\lambda}$ will denote the corresponding costs and variables. Therefore, we consider the *Restricted Linear Programming Master Problem*, which we assume to be nonempty:

$$(RLPM) \quad max\ \tilde{c}^{\,t}\,\tilde{\lambda}$$
$$\tilde{A}\,\tilde{\lambda} = \begin{pmatrix} b \\ 1 \end{pmatrix}$$
$$\tilde{\lambda} \geq 0$$

and we define $\tilde{z}_{\text{LPM}} = max\Big\{ \tilde{c}^{\,t}\,\tilde{\lambda} \ :\ \tilde{A}\,\tilde{\lambda} = \begin{pmatrix} b \\ 1 \end{pmatrix},\ \tilde{\lambda} \geq 0 \Big\}$.
The corresponding dual problem is:

$$(RD) \quad min\ (\pi^t, \mu^t) \cdot \begin{pmatrix} b \\ 1 \end{pmatrix}$$
$$(\pi^t, \mu^t) \cdot \tilde{A} \geq \tilde{c}^{\,t}$$

We solve (RLPM) and we obtain an optimal primal solution $\tilde{\lambda}^*$ and an optimal dual solution (π^*, μ^*).

2. **<u>Primal Feasibility</u>** We notice that $\lambda = (\tilde{\lambda}^*, 0)$ is a feasible solution of (LPM), therefore: $\tilde{z}_{\text{LPM}} = \tilde{c}^{\,t}\,\tilde{\lambda}^* \leq z_{\text{LPM}}$.

3. **<u>Optimality Check</u>** We also notice that from the Strong Duality Theorem in Linear Programming it follows that: $\tilde{c}^{\,t}\,\tilde{\lambda}^* = \sum_{i=1}^{m} \pi_i^* b_i + \sum_{k=1}^{K} \mu_k^*$. We have to check whether the dual solution (π^*, μ^*) is feasible for (D). By definition of the feasible region of (D), we have to control if the reduced prices are nonpositive:

$$c^k x - \pi^{*\,t} A^k x - \mu_k^* \leq 0 \quad \forall x \in X^k,\ \forall k = 1, \ldots, K.$$

For the sake of efficiency, we can consider all the points of X^k at once, in an implicit way, instead of examining each point separately. More precisely, for each k we solve the optimization subproblem:

$$\zeta_k = max\left\{(c^k - \pi^{*\,t} A^k)x - \mu_k^*,\ x \in X^k\right\} \tag{3.13}$$

which is called *pricing problem.*

4. **Stopping Criterion** If $\zeta_k \leq 0 \;\forall k = 1, \dots, K$, then the solution (π^*, μ^*) is feasible for (D). From the Weak Duality Theorem it follows that: $z_{LPM} \leq \sum_{i=1}^{m} \pi_i^* b_i + \sum_{k=1}^{K} \mu_k^*$. Combining this estimation with the bound found at Step 2. we get:

$$\tilde{z}_{\text{LPM}} = \tilde{c}^{\,t}\,\tilde{\lambda}^* = \sum_{i=1}^{m} \pi_i^* b_i + \sum_{k=1}^{K} \mu_k^* \leq z_{\text{LPM}} \leq \sum_{i=1}^{m} \pi_i^* b_i + \sum_{k=1}^{K} \mu_k^*$$

   therefore $z_{\text{LPM}} = \tilde{c}^{\,t}\,\tilde{\lambda}^*$, which means that $(\tilde{\lambda}^*, 0)$ is optimal for (LPM).

5. **Generating a New Column** If $\zeta_k > 0$ for some k, there is a certain $\tilde{x}_k \in X^k$ that has a positive reduced price. Introducing the column $\begin{pmatrix} c^k \tilde{x}_k \\ A^k \tilde{x}_k \\ e_k \end{pmatrix}$ in the restricted problem (RLPM) leads to an enhanced Restricted Linear Programming Master Problem. We solve the new (RLPM) and go back to Step 2.

Observation 3.2.3. We notice that the algorithm ends in a finite number of iterations, in fact it will add at most a column for each element of X^k, for each $k = 1, \dots, K$.

We present an alternative stopping criterion:

4'. **Alternative Stopping Criterion** We notice that $\zeta_k \geq (c^k - \pi^{*\,t} A^k)x - \mu_k^*$ for each $x \in X^k$, by definition of ζ_k. It follows that: $(c^k - \pi^{*\,t} A^k)x - \mu_k^* - \zeta_k \leq 0$ for each $x \in X^k$, which means that $(\pi^*, \mu^* + \zeta)$ is feasible for (D), where $\zeta = (\zeta_1 \dots \zeta_K)$. We can apply the Weak Duality Theorem to $(\pi^*, \mu^* + \zeta)$ and get the bound:

$$z_{LPM} \leq \sum_{i=1}^{m} \pi_i^* b_i + \sum_{k=1}^{K} \mu_k^* + \sum_{k=1}^{K} \zeta_k.$$

   We consider the vector $(\tilde{x}^1, \dots, \tilde{x}^K)$, whose components are the optimal solutions of the subproblems (3.13) for each $k = 1, \dots, K$. If $(\tilde{x}^1, \dots, \tilde{x}^K)$ is feasible for (LPM), meaning that it satisfies the joint constraints $\sum_{k=1}^{K} A^k x^k = b$, then it is optimal for (LPM), and the algorithm stops. This follows from the following observations:

   - ◇ $\sum_{k=1}^{K} c^k \tilde{x}^k \leq z_{LPM}$ since it is feasible
   - ◇ $\zeta_k = (c^k - \pi^{*\,t} A^k)\tilde{x}^k - \mu_k^*$ implies that $\sum_{k=1}^{K} c^k \tilde{x}^k = \sum_{k=1}^{K} \pi^{*\,t} A^k \tilde{x}^k + \sum_{k=1}^{K} \mu_k^* + \sum_{k=1}^{K} \zeta_k = \sum_{i=1}^{m} \pi_i^* b_i + \sum_{k=1}^{K} \mu_k^* + \sum_{k=1}^{K} \zeta_k$, where the last equality is due to the hypothesis that $(\tilde{x}^1, \dots, \tilde{x}^K)$ satisfies the joint constraints.

Therefore:

$$z_{LPM} \leq \sum_{i=1}^{m} \pi_i^* b_i + \sum_{k=1}^{K} \mu_k^* + \sum_{k=1}^{K} \zeta_k = \sum_{k=1}^{K} c^k \tilde{x}^k \leq z_{LPM}$$

from which we get that $\sum_{k=1}^{K} c^k \tilde{x}^k = z_{LPM}$.

3.2.1 Symmetric Traveling Salesman Problem (STSP)

We show how the Column Generation algorithm applies to the case of the Symmetric Traveling Salesman Problem. We recall that $N = \{1, \dots, n\}$ is a set of cities and c_{ij} is the time it takes to travel between city i and city j. We introduce the binary variables x_{ij}, $\forall i,\, j \in N$, $i \neq j$, that assume value 1 if the salesman goes directly from city i to city j and 0 otherwise. We define E as the set of edges that connect the cities, $E(S)$ is the set of edges with both endpoints in $S \subseteq N$, and $\delta(k)$ is the set of edges incident to node k. As seen in Chapter 2, the problem can be formulated as:

$$(STSP) \quad min \sum_{(i,j)\in E} c_{ij} x_{ij}$$

$$\sum_{(i,j)\in\delta(k)} x_{ij} = 2 \qquad \forall k \in N \tag{3.14}$$

$$\sum_{(i,j)\in E} x_{ij} = n \tag{3.15}$$

$$\sum_{(i,j)\in E(S)} x_{ij} \leq |S| - 1 \qquad \forall \varnothing \neq S \subseteq N \setminus \{1\} \tag{3.16}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \tag{3.17}$$

where the constraints (3.14) are called *degree constraints* and the constraints (3.16) are the *subtour elimination constraints*. We remember that a *1-tree* is a subgraph consisting of two edges that incide in node 1, and the edges of a tree on nodes $\{2, \dots, n\}$. The feasible solutions to (3.15), (3.16) and (3.17) are the 1-trees of (STSP).

We start by writing the Dantzig-Wolfe formulation of the problem. We assume that there are T_1 1-trees, and we define X^1 as the set of incidence vectors of the 1-trees, i.e., the set of the vectors $x = (x_{ij})_{(i,j)\in E}$ that satisfy the constraints (3.15), (3.16) and (3.17). Therefore, there is just a single subproblem, i.e., $k = 1$ in terms of the general presentation made in Section 3.1. We use the notation $G^t = (N, E^t)$ to refer to the tth 1-tree, for each $t = 1, \dots, T_1$, and, for each arc $(i,j) \in E$, we write:

$$x_{ij} = \sum_{\substack{t \text{ s.t.} \\ (i,j)\in E^t}} \lambda_t\,, \text{ with } \sum_{t=1}^{T_1} \lambda_t = 1 \text{ and } \lambda_t \in \{0,1\} \;\; \forall t = 1,\dots,T_1. \tag{3.18}$$

Replacing each x_{ij} by (3.18), the degree constraints become:

$$\sum_{(i,j)\in\delta(k)} x_{ij} = \sum_{(i,j)\in\delta(k)} \sum_{\substack{t \text{ s.t.} \\ (i,j)\in E^t}} \lambda_t = \sum_{t=1}^{T_1} d_k^t \lambda_t = 2 \qquad \forall k \in N$$

where d_k^t is the degree of the node k in G^t, which is the number of connections in G^t the node k has to other nodes.
Moreover, the objective function can be rewritten as follows. Let $c = (c_{ij})_{(i,j)\in E}$ be the vector of the costs, and let $x = (x_{ij})_{(i,j)\in E}$ be the vector of the variables. We define $x^t = (x_{ij}^t)_{(i,j)\in E}$, where

$$x_{ij}^t = \begin{cases} 1 & if\ (i,j) \in E^t \\ 0 & otherwise \end{cases}.$$

Therefore:

$$\sum_{(i,j)\in E} c_{ij} x_{ij} = \sum_{(i,j)\in E} c_{ij} \sum_{\substack{t \text{ s.t.} \\ (i,j)\in E^t}} \lambda_t = \sum_{t=1}^{T_1} (cx^t)\lambda_t$$

In conclusion, the Master Problem of STSP can be formulated as:

$$(PM) \quad min \sum_{t=1}^{T_1} (cx^t)\lambda_t$$

$$\sum_{t=1}^{T_1} d_k^t \lambda_t = 2 \qquad \forall k \in N \tag{3.19}$$

$$\sum_{t=1}^{T_1} \lambda_t = 1 \tag{3.20}$$

$$\lambda_t \in \{0,1\} \qquad \forall t = 1,\dots,T_1 \tag{3.21}$$

The corresponding Linear Programming Master is:

$$(LPM) \quad min \sum_{t=1}^{T_1} (cx^t)\lambda_t$$

$$\sum_{t=1}^{T_1} d_k^t \lambda_t = 2 \qquad \forall k \in N \tag{3.22}$$

$$\sum_{t=1}^{T_1} \lambda_t = 1 \tag{3.23}$$

$$\lambda \in \mathbb{R}_+^{T_1} \tag{3.24}$$

which can be written in matrix form as:

$$\begin{aligned} (LPM) \quad min & \sum_{t=1}^{T_1} (cx^t)\lambda_t \\ & A\,\lambda = \begin{pmatrix} b \\ 1 \end{pmatrix} \\ & \lambda \in \mathbb{R}_+^{T_1} \end{aligned}$$

where $A = \begin{pmatrix} d_1^1 & \dots & d_1^{T_1} \\ & \ddots & \\ d_n^1 & \dots & d_n^{T_1} \\ 1 & \dots & 1 \end{pmatrix}$ and $b = \begin{pmatrix} 2 \\ \vdots \\ 2 \end{pmatrix} \in \mathbb{R}^n$.

We associate the variables $(u_i)_{i=1}^n$ with the degree constraints, and the variable μ with the convexity constraint. The dual problem of (LPM) is:

$$\begin{aligned} (D) \quad max & \sum_{k=1}^{n} u_k b_k + \mu \\ & \sum_{k=1}^{n} u_k d_k^t + \mu \leq cx^t \qquad \forall t = 1, \dots, T_1 \end{aligned}$$

Therefore, the reduced price associated with the 1-tree G^t is:

$$\begin{aligned} cx^t - \sum_{k=1}^{n} u_k d_k^t - \mu &= cx^t - \sum_{k=1}^{n} u_k \Bigg(\sum_{(i,j)\in\delta(k)} x_{ij}^t \Bigg) - \mu \\ &= \sum_{(i,j)\in E} (c_{ij} - u_i - u_j) x_{ij}^t - \mu. \end{aligned}$$

The corresponding optimization subproblem is:

$$\zeta_1 = min \Big\{ \sum_{(i,j)\in E} (c_{ij} - u_i - u_j) x_{ij} - \mu, \; x \in X^1 \Big\}$$

Observation 3.2.4. In the case of the (STSP) problem, the pricing problem ζ_1 consists in finding a 1-tree of minimum cost, which is a problem that can be solved very efficiently.

Observation 3.2.5. In this context the pricing problem ζ_1 is a minimization problem, and not a maximization problem as in the general presentation of the procedure, since (STSP) is in turn a minimization problem. Therefore, if $\zeta_1 \geq 0$ the algorithm stops and returns an optimal solution; conversely, if $\zeta_1 < 0$, the subproblem suggests one or more columns to add to the Restricted Linear Programming Master Problem.

3.3 Branch and Price Algorithm for 0-1 Problems

In this section we keep the same notation as in the rest of the chapter, but we restrict to the 0-1 problems. Therefore, the problem that we consider is formulated as:

$$\begin{aligned} (P) \quad max \ & \sum_{k=1}^{K} c^k x^k \\ & \sum_{k=1}^{K} A^k x^k = b \\ & x^k \in X^k \qquad \forall\, k = 1, \dots, K \end{aligned}$$

where $X^k = \{x^k \in \{0,1\}^{n_k} : D^k x^k \leq d_k\}$, and its Dantzig-Wolfe reformulation is:

$$\begin{aligned} (PM) \quad max \ & \sum_{k=1}^{K} \sum_{t=1}^{T_k} (c^k x^{kt})\, \lambda_{kt} \\ & \sum_{k=1}^{K} \sum_{t=1}^{T_k} (A^k x^{kt})\, \lambda_{kt} = b \\ & \sum_{t=1}^{T_k} \lambda_{kt} = 1 \qquad \forall\, k = 1, \dots, K \\ & \lambda_{kt} \in \{0,1\} \qquad \forall\, t = 1, \dots, T_k,\ k = 1, \dots, K \end{aligned}$$

We assume that the Column Generation algorithm presented in Section 3.2 has returned an optimal solution $\tilde{\lambda}$ of the Linear Programming Master Problem, i.e, (LPM). If $\tilde{\lambda}$ is not integer, then it is not solution of the Master Problem, which is the problem we would like to solve. However, since (LPM) is a relaxation of (PM), we have that $z_{\text{LPM}} \geq z$, where z is the optimal value of (PM). This bound suggests that we could use a branch and bound algorithm to solve (PM) [2][4], by efficiently exploiting the Dantzig-Wolfe algorithm via a suitable branching rule.

3.3.1 Branching Rule

We notice that, for each $k = 1, \dots, K$, the point $\tilde{x}^k = \sum_{t=1}^{T_k} \tilde{\lambda}_{kt}\, x^{kt}$ is an element of $\{0,1\}^{n_k}$ if and only if $\tilde{\lambda}$ is integer, since $x^{kt} \in \{0,1\}^{n_k}$ $\forall t = 1, \dots, T_k$. If $\tilde{\lambda}$ is not integer, there exist a certain $\mathtt{k} \in \{1, \dots, K\}$ and $j \in \{1, \dots, n_{\mathtt{k}}\}$ such that the value $\tilde{x}_j^{\mathtt{k}}$ is not integer. We can branch the problem according to the value assumed by the variable $x_j^{\mathtt{k}}$, that is: we split the set S of all feasible solutions of (P) in $S_0 = \{x \in S : x_j^{\mathtt{k}} = 0\}$ and $S_1 = \{x \in S : x_j^{\mathtt{k}} = 1\}$.

In order to check if we can solve the Linear Programming Master Problems resulting from the partition $S = S_0 \cup S_1$ efficiently, we have to define the Master Problems and the subproblems ζ_k, defined in (3.13), associated with S_0 and S_1. Assume that $x_j^{\mathtt{k}} = \mathtt{i} \in \{0,1\}$. We notice that:

$$x_j^{\mathtt{k}} = \sum_{t=1}^{T_k} \lambda_{\mathtt{k}t}\, x_j^{\mathtt{k}t}$$

therefore, for each $t = 1, \dots, T_{\mathtt{k}}$, we can have: $\lambda_{\mathtt{k}t} = 0$ and the tth factor does not appear in the sum, or $\lambda_{\mathtt{k}t} = 1$ and it must be that $x_j^{\mathtt{k}t} = \mathtt{i}$. As a result, for $k = \mathtt{k}$ we sum over the t such that $x_j^{\mathtt{k}t} = \mathtt{i}$ in the Master Problem at node $S_{\mathtt{i}}$:

$$\begin{aligned}
(PM(S_{\mathtt{i}})) \quad max \; & \sum_{\substack{k=1\\ k\neq \mathtt{k}}}^{K} \sum_{t=1}^{T_k} (c^k x^{kt})\, \lambda_{kt} + \sum_{\substack{t \text{ s.t.}\\ x_j^{\mathtt{k}t}=\mathtt{i}}} (c^{\mathtt{k}} x^{\mathtt{k}t})\, \lambda_{\mathtt{k}t} \\
& \sum_{\substack{k=1\\ k\neq \mathtt{k}}}^{K} \sum_{t=1}^{T_k} (A^k x^{kt})\, \lambda_{kt} + \sum_{\substack{t \text{ s.t.}\\ x_j^{\mathtt{k}t}=\mathtt{i}}} (A^{\mathtt{k}} x^{\mathtt{k}t})\, \lambda_{\mathtt{k}t} = b \\
& \sum_{t=1}^{T_k} \lambda_{kt} = 1 \qquad \forall\, k = 1, \dots, K,\ k \neq \mathtt{k} \\
& \sum_{\substack{t \text{ s.t.}\\ x_j^{\mathtt{k}t}=\mathtt{i}}} \lambda_{\mathtt{k}t} = 1 \\
& \lambda_{kt} \in \{0,1\} \qquad \forall\, t = 1, \dots, T_k,\ k = 1, \dots, K
\end{aligned}$$

We observe that $(PM(S_{\mathtt{i}}))$ has the same form of the original Master Problem, but a set of columns has been excluded. Therefore, we have obtained a Restricted Master Problem, and we proceed to solve its linear relaxation, i.e., (RLPM), with the algorithm presented in Section 3.2. The pricing problems that must be solved in the Column Generation algorithm are:

$$\begin{aligned}
\zeta_k(S_{\mathtt{i}}) &= max \left\{ (c^k - \pi^t A^k)x - \mu_k,\ x \in X^k \right\} \qquad \forall k = 1, \dots, K,\ k \neq \mathtt{k} \\
\zeta_{\mathtt{k}}(S_{\mathtt{i}}) &= max \left\{ (c^{\mathtt{k}} - \pi^t A^{\mathtt{k}})x - \mu_{\mathtt{k}},\ x \in X^{\mathtt{k}},\ x_j = \mathtt{i} \right\}.
\end{aligned}$$

Chapter 4

Multicommodity Flows

In many applications it may happen that distinct commodities, governed by their own network flow constraints, share the same network. In particular, we consider some commodities distinct if they correspond to different goods, or if they have different points of origin and destination. If the single commodity flow problems are independent, we can solve them separately and get a global solution of the problem. However, the commodities are more likely to interact with each other, i.e., to be linked by some joint constraints. Some application contexts where such a problem arises are Logistics, Health Care, and telecommunications. For example, in a communication network, nodes represent stations transmitting or receiving messages, and arcs represent the lines of communication; in this case, the messages between different pairs of nodes define distinct commodities. Another example emerges in distribution system planning from the need to ship different products, which represent commodities, from industries to retailers through various means of transport. In this chapter we present a network flow problem, known as the *Minimum Cost Multicommodity Flow* problem, in which the total flow of the commodities on each arc (i, j) is bounded by a capacity u_{ij}, and the aim is to satisfy the balances of the nodes for each commodity, at a minimum cost, by respecting the joint capacity constraints. This problem extends the Minimum Cost Flow problem, which is a classic problem of network flow optimization theory[2], [4]. We will formulate the Minimum Cost Multicommodity Flow problem in terms of arc flows and in terms of path flows, and we will illustrate how to solve the problem by applying Lagrangian relaxation in the first case, and using a Column Generation algorithm in the second case.

4.1 Formulation with Arc Flows

We assume that $G = (N, E)$ is a directed graph representing a flow network, where $N = \{1, \dots, n\}$ is the set of nodes, E is the set of the m directed arcs, and we have K commodities, e.g., K goods which must be sent along the network from some origins to some destinations, at a minimum cost. Suppose that c^k_{ij} denotes the cost, per unit, for commodity k on arc (i,j), and x^k_{ij} represents the flow of commodity k on arc (i,j), for each $(i,j) \in E$ and for each $k \in \{1,\dots,K\}$. We can formulate the *Minimum Cost Multicommodity Flow* problem as the following Linear Programming problem:

$$(MCF1) \quad min \sum_{k=1}^{K} \sum_{(i,j)\in E} c^k_{ij} x^k_{ij}$$

$$\sum_{(j,i)\in BS(i)} x^k_{ji} - \sum_{(i,j)\in FS(i)} x^k_{ij} = b^k_i \qquad \forall\, i \in N,\ k = 1,\dots,K \tag{4.1}$$

$$\sum_{k=1}^{K} x^k_{ij} \le u_{ij} \qquad \forall\,(i,j) \in E \tag{4.2}$$

$$0 \le x^k_{ij} \le u^k_{ij} \qquad \forall\,(i,j) \in E,\ k = 1,\dots,K \tag{4.3}$$

where $BS(i)$ is the *backward star* of node i, which is the set of arcs entering node i, and $FS(i)$ is the *forward star* of node i, which is the set of arcs coming out of node i, for each node $i \in N$. Furthermore, b^k_i is the balance of node i relating to the commodity k, for each $i \in N$ and for each $k \in \{1,\dots,K\}$; the nodes with a negative balance are called *source nodes*, those with a positive balance are called *sink nodes*, and those with null balance are called *transit nodes.* We have also imposed individual flow bounds u^k_{ij} on each arc and for each commodity, but in certain applications these bounds are $+\infty$.
In the following we will refer to the constraints (4.1) as the *balance constraints*, while the global capacity constraints (4.2) are the joint constraints, since they link together the K commodities.

Observation 4.1.1. We are making some assumptions in this formulation of the problem. For instance, we are assuming that, for each commodity, every unit flow along an arc uses a unit of capacity of that arc. In general, the unit flow of each commodity k could use a certain quantity ρ^k_{ij} of capacity along the arc (i,j), and the joint constraints would become: $\sum_{k=1}^{K} \rho^k_{ij} x^k_{ij} \le u_{ij}$, $\forall\,(i,j) \in E$. We are also supposing that the cost on each arc is linear in the flow on that arc, and that the flow variables, i.e., x^k_{ij}, can be fractional.

4.1.1 Optimality Conditions

We state the optimality conditions for the Minimum Cost Multicommodity Flow problem, so that we can characterize the optimal solutions of the problem. Since we are handling a Linear Programming problem, Theorem 1.1.3 allows us to say that the desired conditions are given by the Complementary Slackness Property that we presented in Chapter 1. Here we make explicit what special form they assume, in terms of graph structures, for the primal-dual pair of the Minimum Cost Multicommodity Flow problem.

For this analysis we assume that $u_{ij}^k = +\infty$ for each arc $(i,j) \in E$ and for each commodity $k \in \{1,\dots,K\}$. In order to formulate the dual problem of (MCF1), we associate the dual variables $(w_{ij})_{(i,j)\in E}$ with the joint constraints (4.2), and the variables $(\pi_i^k)_{\substack{i=1\dots n\\ k=1\dots K}}$ with the balance constraints (4.1). We notice that the balance constraints can be rewritten in matrix form as:

$$\mathcal{N} x^k = b^k \quad \forall\, k = 1,\dots,K$$

where $\mathcal{N} \in \mathcal{M}_{n,m}$ is the incidence matrix of the graph $G = (N,E)$, that is a matrix which is defined in the following way:

$$\mathcal{N}_{i,(k,l)} = \begin{cases} -1 & if\ k = i \\ 1 & if\ l = i \\ 0 & otherwise \end{cases} \qquad \forall i \in N,\ \forall (k,l) \in E.$$

This means that in each column (k,l) of $\mathcal{N}$ there is a 1 in the row corresponding to the node where the arc enters, i.e., the head node l, and a -1 in the row corresponding to the node from where the arc exits, i.e., the tail node k.
The matrix form of (MCF1) is thus:

$$\begin{aligned} (MCF1) \quad & min\ c^t x \\ & Ax = b \\ & -Ix \geq -u \\ & x \geq 0 \end{aligned}$$

where:

- ◇ $c = (c_{ij}^k)_{\substack{(i,j)\in E\\ k=1\dots K}}$ is the cost vector, having mK components
- ◇ $x = (x_{ij}^k)_{\substack{(i,j)\in E\\ k=1\dots K}}$ is the flow vector, having mK components
- ◇ $u = (u_{ij})_{(i,j)\in E}$ is the capacity vector, having m components
- ◇ $b = (b_i^k)_{\substack{i=1\dots n\\ k=1\dots K}}$ is the balance vector, having nK components
- ◇ $A = \begin{pmatrix} \mathcal{N} & & \\ & \ddots & \\ & & \mathcal{N} \end{pmatrix} \in \mathcal{M}_{nK,\, mK}$
- ◇ $I = \left(\; Id_m \;\middle|\; \dots \;\middle|\; Id_m \; \right) \in \mathcal{M}_{m,\, mK}$.

Therefore, if we set $w = \big(w_{ij}\big)_{(i,j)\in E}$ and $\pi = \big(\pi_i^k\big)_{\substack{i=1\dots n\\ k=1\dots K}}$ as the vectors of the dual variables, the dual problem of (MCF1) is:

$$\begin{aligned} (D1) \quad max\ & \pi^t b - w^t u \\ & \pi^t A - w^t I \le c \\ & w \ge 0 \end{aligned}$$

which can be expanded as:

$$\begin{aligned} (D1) \quad max\ & \sum_{k=1}^{K}\sum_{i=1}^{n} \pi_i^k b_i^k - \sum_{(i,j)\in E} w_{ij}u_{ij} \\ & \pi_j^k - \pi_i^k - w_{ij} \le c_{ij}^k \qquad \forall (i,j)\in E,\ k=1,\dots,K \\ & w_{ij} \ge 0 \qquad \forall (i,j) \in E \end{aligned}$$

At this point we can write the complementary slackness conditions for the pair (MCF1), (D1), recalling that we are assuming $u_{ij}^k = +\infty$ for each arc $(i,j) \in E$ and for each commodity $k \in \{1,\dots,K\}$:

Theorem 4.1.2. (Arc flow Complementary Slackness Conditions)
A pair $\big(x,(w,\pi)\big)$ of feasible solutions for the problem (MCF1) and its dual (D1), respectively, is optimal if and only if it satisfies the following conditions:

- ◇ $w_{ij}\Big(u_{ij} - \sum_{k=1}^{K} x_{ij}^k\Big) = 0 \qquad \forall (i,j) \in E$
- ◇ $x_{ij}^k \Big(c_{ij}^k + w_{ij} + \pi_i^k - \pi_j^k\Big) = 0 \qquad \forall (i,j) \in E,\ k=1,\dots,K$

4.1.2 Lagrangian Relaxation

We apply the Lagrangian relaxation presented in Chapter 1 to the Minimum Cost Multicommodity Flow problem (MCF1) with each $u_{ij}^k = +\infty$ by dualizing the joint constraints (4.2). Since (4.2) are inequality constraints, we require the Lagrangian multipliers $w = (w_{ij})_{(i,j)\in E}$ to be nonnegative, i.e., $w_{ij} \ge 0 \ \ \forall (i,j) \in E$. The corresponding Lagrangian relaxation is:

$$\begin{aligned} (MCF1(w)) \quad min\ & \sum_{k=1}^{K}\sum_{(i,j)\in E} (c_{ij}^k + w_{ij})x_{ij}^k - \sum_{(i,j)\in E} w_{ij}u_{ij} \\ & \sum_{(j,i)\in BS(i)} x_{ji}^k - \sum_{(i,j)\in FS(i)} x_{ij}^k = b_i^k \qquad \forall\, i \in N,\ k=1,\dots,K \\ & x_{ij}^k \ge 0 \qquad \forall\,(i,j)\in E,\ k=1,\dots,K \end{aligned}$$

For any given choice of the Lagrangian multipliers, the term $-\sum_{(i,j)\in E} w_{ij}u_{ij}$ in the objective function is constant, therefore it can be omitted during the optimization

process. The resulting problem decomposes into K separate uncapacitated minimum cost flow problems, one for each commodity, with costs $(c^k_{ij} + w_{ij})$, for $k = 1, \dots, K$, which can be efficiently solved by means of algorithms working directly on the flow network [1][2].
By specializing the results presented in Chapter 1, and using an equivalent matrix formulation, we deduce that the Lagrangian function is:

$$L(w) = min\Big\{\sum_{k=1}^{K}\big((c^k)^t + w^t\big)x^k : \mathcal{N}x^k = b^k \ \forall k = 1, \dots, K, \ x \geq 0\Big\} \tag{4.4}$$

where $c^k = (c^k_{ij})_{(i,j)\in E}$ for each $k \in \{1, \dots, K\}$. The Lagrangian Dual problem can be stated as:

$$(LD) \quad L^* = \max_{w \geq 0} L(w)$$

We can solve (LD) by applying the subgradient optimization procedure presented in Section 1.4. In particular, at a generic iteration s we have to compute $L(w^s)$, for the current vector w^s of the Lagrangian multipliers. Let $y^{ks} = (y^{ks}_{ij})_{(i,j)\in E}$, for each $k \in \{1, \dots, K\}$, be such that it reaches the value $L(w^s)$, that is:

$$L(w^s) = \sum_{k=1}^{K}\big((c^k)^t + w^t\big)y^{ks}.$$

Moreover, specializing again the results presented in Chapter 1, the vector $\gamma^s = (\gamma^s_{ij})_{(i,j)\in E}$, whose components are defined as $\gamma^s_{ij} = \sum\limits_{k=1}^{K} y^{ks}_{ij} - u_{ij}$, is a subgradient at w^s of $L(w)$. If $\gamma^s_{ij} = 0 \ \ \forall (i,j) \in E$, the algorithm stops and w^s is an optimal solution to the Lagrangian Dual. Otherwise, for each $(i,j) \in E$, w^s_{ij} is updated to:

$$\begin{aligned} w^{s+1}_{ij} &= \big[w^s_{ij} + \theta_s\, \gamma^s_{ij}\big]^+ \\ &= \Big[w^s_{ij} + \theta_s\Big(\sum_{k=1}^{K} y^{ks}_{ij} - u_{ij}\Big)\Big]^+ \end{aligned}$$

where θ_s is the step length at iteration s.

Observation 4.1.3. Since the Minimum Cost Multicommodity Flow problem is a Linear Programming problem, the optimal value L^* of the Lagrangian Dual problem equals the optimal objective function value of (MCF1), as stated by Theorem 1.3.1.[1] The advantage is that the Lagrangian Dual problem can be usually solved more efficiently via the presented techniques, especially when (MCF1) is defined on a very large graph, as normally happens in the application context.

[1]Even if (MCF1) is not expressed in the same form as the statement of Theorem 1.3.1, the thesis still holds.

4.2 Formulation with Path Flows

The purpose of this section is to model the Minimum Cost Multicommodity Flow problem using path flows, instead of arc flows, and then solve it with the Column Generation algorithm presented in Section 3.2. Again, the motivation is to suitably decompose the (MCF1) model, in order to be able to solve it more efficiently.

We start by considering the case of a single commodity, and then we extend the results obtained to the multicommodity case. Therefore, we consider the Minimum Cost Flow problem, which is the special case of the Minimum Cost Multicommodity Flow problem managing a single commodity, whose formulation is obtained by setting $K = 1$ in (MCF1). Let $G = (N, E)$ be the directed graph modeling the considered flow network, as introduced in Section 4.1. We denote the collection of all directed paths between any pair of nodes as $\mathscr{P}$, and the collection of all directed cycles as $\mathscr{C}$. The decision variables in the path and cycle formulation will be the flows on each path and cycle, that is: $f(P)$ and $f(C)$ for $P \in \mathscr{P}$, $C \in \mathscr{C}$. On the contrary, the decision variables in the arc formulation presented in Section 4.1 are the flows along the individual arcs: x_{ij} for each $(i, j) \in E$.

Observation 4.2.1. Each set of path and cycle flows is uniquely associated with the arc flow variables defined as:

$$x_{ij} = \sum_{P \in \mathscr{P}} \delta_{ij}(P) f(P) + \sum_{C \in \mathscr{C}} \delta_{ij}(C) f(C)$$

where $\delta_{ij}(P) = \begin{cases} 1 & if\ (i,j) \in P \\ 0 & otherwise \end{cases}$ and, similarly, $\delta_{ij}(C) = \begin{cases} 1 & if\ (i,j) \in C \\ 0 & otherwise \end{cases}$ for each $(i, j) \in E$.

Furthermore, it is possible to prove that also the converse holds, i.e., that we can represent any arc flow as a path and cycle flow. Here we will just state this result, whose proof can be found in [1]:

Theorem 4.2.2. (Flow Decomposition Theorem)
Every path and cycle flow has a unique representation in terms of nonnegative arc flows. Conversely, every nonnegative arc flow can be represented in terms of a path and cycle flow, but not necessarily in a unique way.

Let us consider now again the more general Minimum Cost Multicommodity Flow problem, handling K commodities which interact through the joint capacity constraints. We start by making some assumptions:

◇ each commodity k has a single source node s^k and a single sink node t^k

◇ d^k is the amount of units that must be sent from s^k to t^k

◇ we set $u_{ij}^k = +\infty$ for each arc and for each commodity, as previously assumed

- ◇ $c_{ij}^k \geq 0$ for each arc and for each commodity

We notice that the condition that all arc costs must be nonnegative, i.e., $c_{ij}^k \geq 0$ for each $(i,j) \in E$ and $k \in \{1,\dots,K\}$, implies that all the directed cycles have nonnegative costs for each commodity. Therefore, there exist some minimum cost multicommodity flows such that, for each commodity, the flow on every cycle is zero, and for that reason we can eliminate the cycle flow variables.[1]

For each commodity k, we denote the collection of all directed paths from the source s^k to the sink t^k as $\mathbf{P}^k$. The decision variables in the path formulation will be the flows $f(P)$ on each path P of $\mathbf{P}^k$, for each $k \in \{1,\dots,K\}$. By Theorem 4.2.2 we can decompose each optimal arc flow x_{ij}^k into path flows $f(P)$ as:

$$x_{ij}^k = \sum_{P \in \mathbf{P}^k} \delta_{ij}(P)\, f(P). \tag{4.5}$$

For each $P \in \mathbf{P}^k$ we define the per unit cost of flow on P, related to commodity k, as:

$$c^k(P) = \sum_{(i,j)\in E} c_{ij}^k\, \delta_{ij}(P) = \sum_{(i,j)\in P} c_{ij}^k.$$

We can thus proceed to replace (4.5) in the formulation (MCF1), and we get:

$$(MCF2) \quad min \sum_{k=1}^{K} \sum_{P \in \mathbf{P}^k} c^k(P) f(P)$$

$$\sum_{P \in \mathbf{P}^k} f(P) = d^k \qquad \forall\, k = 1,\dots,K \tag{4.6}$$

$$\sum_{k=1}^{K} \sum_{P \in \mathbf{P}^k} \delta_{ij}(P) f(P) \leq u_{ij} \qquad \forall\,(i,j) \in E \tag{4.7}$$

$$f(P) \geq 0 \qquad \forall k = 1,\dots,K,\ P \in \mathbf{P}^k \tag{4.8}$$

Observation 4.2.3. We notice that the path flows formulation has a simple constraints structure: for a network with n nodes, m arcs, and K commodities, it has $m+K$ constraints. On the contrary, the arc flow formulation (MCF1) has $m+nK$ constraints. The path formulation has a huge number of variables, i.e., one for path in $\mathbf{P}^k$ for each commodity k, which, however, can be efficiently managed via the Dantzig-Wolfe reformulation and the Column Generation algorithm, as shown in the following sections.

4.2.1 Optimality Conditions

Similarly to what we did in the previous section, we state the optimality conditions for the path flow formulation of the Minimum Cost Multicommodity Flow problem.

We start by writing the matrix form of (MCF2) and its corresponding dual problem, then we deduce how the complementary slackness conditions are written in this context.

In order to formulate the dual problem of (MCF2), we associate the dual variables $(\sigma^k)_{k=1\dots K}$ with the constraints (4.6), and the variables $(w_{ij})_{(i,j)\in E}$ with the constraints (4.7). We assume that $\mathbf{P}^k$ has T_k elements and we denote its elements as: $P_1^k, \dots, P_{T_k}^k$, for each $k = 1, \dots, K$. Consequently, (MCF2) has $T_1 + \dots + T_K$ variables: $\big(f(P)\big)_{P\in\mathbf{P}^k,\, k=1\dots K}$.

The matrix form of (MCF2) is:

$$\begin{aligned} (MCF2) \quad & min\; c^t f(P) \\ & A\, f(P) = d \\ & -D\, f(P) \geq -u \\ & f(P) \geq 0 \end{aligned}$$

where:

- ◇ $c^t = \big(c^1(P_1^1)\dots c^1(P_{T_1}^1)\; \dots\; c^K(P_1^K)\dots c^K(P_{T_K}^K)\big) \in \mathbb{R}_+^{T_1+\dots+T_K}$
- ◇ $f(P) = \begin{pmatrix} f(P_1^1) \\ \vdots \\ f(P_{T_1}^1) \\ \vdots \\ f(P_1^K) \\ \vdots \\ f(P_{T_K}^K) \end{pmatrix} \in \mathbb{R}_+^{T_1+\dots+T_K}$
- ◇ $A = \begin{pmatrix} 1\dots1 & & & \\ & 1\dots1 & & \\ & & \ddots & \\ & & & 1\dots1 \end{pmatrix} = \Big(\; e_1 \,\big|\, \dots \,\big|\, e_1 \,\big|\, \dots \,\big|\, e_K \,\big|\, \dots \,\big|\, e_K \; \Big)$
  is an element of $\mathcal{M}_{K,\,T_1+\dots+T_K}$, and e_k is the kth element of the canonical basis of $\mathbb{R}^K$
- ◇ $d = \begin{pmatrix} d^1 \\ \vdots \\ d^K \end{pmatrix} \in \mathbb{R}^K$
- ◇ $D \in \mathcal{M}_{m,\,T_1+\dots+T_K}$ the matrix whose (i,j)th row is:

$$\big(\delta_{ij}(P_1^1)\dots\delta_{ij}(P_{T_1}^1)\;\dots\;\delta_{ij}(P_1^K)\dots\delta_{ij}(P_{T_K}^K)\big)$$

- ◇ $u = (u_{ij})_{(i,j)\in E}$

Therefore, if we set $w = (w_{ij})_{(i,j)\in E}$ and $\sigma = (\sigma^k)_{k=1\ldots K}$ as the vectors of the dual variables, the dual problem of (MCF2) is:

$$\begin{aligned}(D2) \quad & max\ \sigma^t d - w^t u \\ & \sigma^t A - w^t D \leq c^t \\ & w \geq 0\end{aligned}$$

which can be expanded as:

$$(D2) \quad max \sum_{k=1}^{K} \sigma^k d^k - \sum_{(i,j)\in E} w_{ij} u_{ij}$$

$$\sigma^k - \sum_{(i,j)\in E} w_{ij}\, \delta_{ij}(P) \leq c^k(P) \qquad \forall P \in \mathbf{P}^k,\ k = 1, \ldots, K \tag{4.9}$$

$$w_{ij} \geq 0 \qquad \forall (i,j) \in E \tag{4.10}$$

Theorem 4.2.4. (Path flow Complementary Slackness Conditions)
A pair $\big(f(P), (w,\sigma)\big)$ of feasible solutions for the problem (MCF2) and its dual (D2), respectively, is optimal if and only if it satisfies the following conditions:

- ◇ $$w_{ij}\Big(u_{ij} - \sum_{k=1}^{K} \sum_{P\in\mathbf{P}^k} \delta_{ij}(P) f(P)\Big) = 0 \qquad \forall (i,j) \in E \tag{4.11}$$
- ◇ $$f(P)\Big(c^k(P) + \sum_{(i,j)\in E} w_{ij}\delta_{ij}(P) - \sigma^k\Big) = 0 \qquad \forall P \in \mathbf{P}^k,\ k = 1, \ldots, K \tag{4.12}$$

Definition 4.2.1. For each path flow variable $f(P)$, with $P \in \mathbf{P}^k$ for some k, we define the *reduced cost of* P as:

$$\begin{aligned} c_P^{w,\sigma} &= c^k(P) + \sum_{(i,j)\in P} w_{ij} - \sigma^k \\ &= \sum_{(i,j)\in P} c_{ij}^k + \sum_{(i,j)\in P} w_{ij} - \sigma^k \\ &= \sum_{(i,j)\in P} (c_{ij}^k + w_{ij}) - \sigma^k. \end{aligned}$$

In other words, $c_P^{w,\sigma}$ is the per unit cost of flow on P, related to commodity k, with respect to the modified costs $(c_{ij}^k + w_{ij})$, from which it is subtracted σ^k.

Observation 4.2.5. We notice that the conditions $c_P^{w,\sigma} \geq 0 \quad \forall P \in \mathbf{P}^k,\ k = 1, \ldots, K$, are equivalent to the constraints (4.9), therefore they hold for any feasible solution of (D2). The complementary slackness conditions (4.12) imply that the reduced cost $c_P^{w,\sigma}$ must be zero if $f(P) > 0$, i.e., if the path P carries flow in the optimal solution of (MCF2). Further observe that $c_P^{w,\sigma} = 0$ if and only if $\sum_{(i,j)\in P} (c_{ij}^k + w_{ij}) = \sigma^k$.

4.2.2 Column Generation Solution Approach

In this section we show how to solve the Minimum Cost Multicommodity Flow problem using the Column Generation algorithm presented in Section 3.2; we notice that this is legitimate, since (MCF2) is a Linear Programming problem. The choice of this procedure is justified by the fact that we want to manage the exponential number of variables of (MCF2) efficiently, in fact, we recall that we have a variable for each path in $\mathbf{P}^1 \cup \ldots \cup \mathbf{P}^K$.

1. **Initialization** We start by taking a subset of paths in $\mathbf{P}^1 \cup \ldots \cup \mathbf{P}^K$, and we consider the problem (MCF2) restricted to the variables $f(P)$, for each P in the subset. We will refer to this problem as the *Restricted Minimum Cost Multicommodity Flow* problem, which can be formulated using $\tilde{c}$ and $\widetilde{f(P)}$ as costs and variables, and $\tilde{A}$, $-\tilde{D}$ as constraints matrices.

2. **Primal Feasibility** Solving the restricted problem, if it is feasible, we obtain an optimal primal solution $\widetilde{f(P)}^*$, and an optimal dual solution (w^*, σ^*). We notice that $f(P) = \big(\widetilde{f(P)}^*, 0\big)$ is a feasible solution of (MCF2), whereas (w^*, σ^*) is not necessarily feasible for (D2).

3. **Optimality Check** In order to verify the feasibility of the dual solution, we have to check whether, for each commodity k, the reduced costs of all the paths of $\mathbf{P}^k$ are nonnegative:

$$c_P^{w^*,\sigma^*} = \sum_{(i,j)\in P} (c_{ij}^k + w_{ij}^*) - \sigma^{*k} \geq 0 \quad \forall P \in \mathbf{P}^k$$

   which is equivalent to check if:

$$\sum_{(i,j)\in P} (c_{ij}^k + w_{ij}^*) \geq \sigma^{*k} \quad \forall P \in \mathbf{P}^k.$$

   For the sake of efficiency, the algorithm considers all the elements of $\mathbf{P}^k$ at once, therefore we solve the following optimization subproblem:

$$\zeta_k = min\Big\{ \sum_{(i,j)\in P} (c_{ij}^k + w_{ij}^*)\,,\ P \in \mathbf{P}^k \Big\}. \tag{4.13}$$

   We notice that, for each $k = 1, \ldots, K$, ζ_k is the length of the shortest path from the source node s^k to the sink node t^k with respect to the modified costs $(c_{ij}^k + w_{ij}^*)$. Therefore, the feasibility check of the dual solution consists in solving K shortest path problems, i.e., network optimization problems which can be solved very efficiently.[1] [2]

4. **Stopping Criterion** If $\zeta_k \geq \sigma^{*k}$ for each $k = 1, \ldots, K$, then the solution (w^*, σ^*) is feasible for (D2), and the algorithm stops. In fact, we can conclude that $f(P) = \big(\widetilde{f(P)}^*, 0\big)$ is optimal for (MCF2) since the pair $\big(f(P), (w^*, \sigma^*)\big)$ satisfies the complementary slackness conditions (4.11), (4.12).

5. **Generating a New Column** Conversely, if $\zeta_k < \sigma^{*k}$ for some k, there exists a $t \in \{1, \dots, T_k\}$ such that the length of the path P_t^k, with respect to the modified costs $(c_{ij}^k + w_{ij}^*)$, is less than σ^{*k}. In this case we enhance the restricted problem by adding to $\tilde{A}$ and $-\tilde{D}$, respectively, the column of A and $-D$ corresponding to P_t^k, and we go back to Step 2.

Observation 4.2.6. The procedure ends in a finite number of iterations, in fact it will add at most a column for each element of $\mathbf{P}^k$, for each $k \in \{1, \dots, K\}$.

Observation 4.2.7. At each iteration of the Column Generation algorithm we have both an upper bound and a lower bound of the optimal objective function value of the Multicommodity Flow problem. Let z denote the optimal objective function value of (MCF2), and assume that $\tilde{z}$ is the objective function value corresponding to the primal feasible solution $f(P) = \big(\widetilde{f(P)^*}, 0\big)$ obtained at Step 2. We notice that $z \leq \tilde{z}$ by the feasibility of $f(P)$. We recall that (MCF1) and (MCF2) are different formulations of the same problem, therefore the optimal value of the objective function of these two problems is the same. In Section 4.1.2 we presented the Lagrangian relaxation technique for (MCF1), and Corollary 1.2.1.1 assures that the value

$$L(w^*) = min\Big\{\sum_{k=1}^{K}\big((c^k)^t + w^{*t}\big)x^k - w^{*t}u : \mathcal{N}x^k = b^k \ \forall k = 1, \dots, K, \ \ x \geq 0\Big\} \tag{4.14}$$

is a lower bound of z. We notice that (4.14) differs from (4.4) for the subtraction of the term $w^t u$; this is due to the fact that, even though during the optimization process that term was irrelevant, it must be considered to define the lower bound. In conclusion at each iteration of the Column Generation algorithm we have:

$$L(w^*) \leq z \leq \tilde{z}$$

and therefore $\tilde{z} - L(w^*)$ constitutes an optimality gap for z, which can be used to terminate the procedure if it is smaller than a set value.

Chapter 5

Relevance of Lagrangian Relaxation and Dantzig-Wolfe Decomposition in Network Design: the Capacitated Facility Location Problem (CFL)

In this last chapter we refer to the Capacitated Facility Location problem[1], with the aim of showing that the two methodologies previously presented, i.e., the Lagrangian relaxation and the Column Generation technique, are relevant for solving important Network Design problems. The Capacitated Facility Location problem is in fact a combinatorial optimization problem with applications in distribution and production planning, and in telecommunication network design, among the others. By considering the scientific paper [3], in the first section we formulate the problem as a Mixed Integer Linear Programming model, drawing attention to the differences with respect to the Uncapacitated Facility Location problem. In the second section we present a Lagrangian relaxation of the Capacitated Facility Location problem, and we formulate the corresponding Lagrangian Dual problem as a Linear Programming problem, which we will call (LD$'$). The results in [3] show how, instead of solving the Lagrangian Dual using the subgradient algorithm presented in Section 1.4, we can consider the dual problem of (LD$'$), which constitutes the Linear Programming relaxation of an equivalent formulation of (CFL), and solve it efficiently using the Column Generation algorithm. The advantage is that, in this way, it is

[1]The uncapacitated version of this problem was introduced in Section 2.1.

possible to calculate exactly the lower bound given by the Lagrangian Dual problem, while the subgradient algorithm often gives only an approximation of this lower bound. Moreover, we are able to obtain the optimal primal solution of a Linear Programming relaxation, which, together with the Lagrangian Dual lower bound, can be employed within a Branch and Price algorithm for (CFL).

5.1 Problem formulation

Keeping the same notation of Chapter 2, we assume that $N = \{1, \dots, n\}$ is a set of depots and $M = \{1, \dots, m\}$ is a set of clients. We suppose that there is a fixed cost f_j associated with the use of depot j, and c_{ij} is the unit transportation cost to send client i's order from depot j. We study the problem of deciding which depots to open and how to assign the customers to those depots, with the aim of minimizing the fixed and the shipping costs. Moreover, we assume that each depot j must respect a capacity s_j if it is open. We introduce the binary variables y_j, $\forall j \in N$, that assume value 1 if depot j is used and 0 otherwise, and the nonnegative variables x_{ij}, $\forall i \in M,\ j \in N$, which represent the fraction of client i's order satisfied from depot j. We let d_i be the demand of client i, $\forall i \in M$, and $d_i x_{ij}$ is thus the quantity of client i's demand satisfied by depot j, $\forall i \in M,\ j \in N$. Finally, we define: $d(M) = \sum\limits_{i=1}^{m} d_i$.
The following Mixed Integer Linear Programming model formulates the considered problem:

$$(CFL) \quad min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{j=1}^{n} f_j y_j$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \in M \tag{5.1}$$

$$\sum_{i=1}^{m} d_i x_{ij} \le s_j y_j \qquad \forall j \in N \tag{5.2}$$

$$\sum_{j=1}^{n} s_j y_j \ge d(M) \tag{5.3}$$

$$x_{ij} \le y_j \qquad \forall i \in M, j \in N \tag{5.4}$$

$$x_{ij} \ge 0 \qquad \forall i \in M, j \in N \tag{5.5}$$

$$y_j \in \{0,1\} \qquad \forall j \in N \tag{5.6}$$

Constraints (5.1) represent the satisfaction of the demand of each client. Constraints (5.2) are the *capacity constraints*, and express the fact that, if depot j is open, the total demand to which it can supply can not exceed the value s_j. Moreover, if depot j is not open, they force the corresponding assignment variables x_{ij} to 0, i.e., no client can be assigned to j.

Observation 5.1.1. We notice that (CFL) and (UFL)[2] have the same objective function and share the same constraints, but (CFL) has two additional sets of constraints, i.e., constraints (5.2) and (5.3).

Observation 5.1.2. We observe that the constraints (5.3) and (5.4) are redundant. In fact, (5.3) can be obtained by adding over j the constraints (5.2), and using (5.1), while (5.4) follow from (5.1), (5.2) and (5.5). However, they are added in order to simplify the following discussion.

Remark. We will assume in the following that:

- ◇ $c_{ij} \geq 0 \quad \forall i \in M,\, j \in N$
- ◇ $f_j \geq 0 \quad \forall j \in N$
- ◇ $s_j > 0 \quad \forall j \in N$
- ◇ $d_i \geq 0 \quad \forall i \in M$

5.2 Lagrangian relaxation of (CFL)

In this section we present the Lagrangian relaxation of the Capacitated Facility Location problem obtained from the dualization of the constraints (5.1), analogously to what was done for the Uncapacitated Facility Location problem in Chapter 2. We recall that, as seen in Section 1.3, solving the Lagrangian Dual corresponding to this relaxation, we obtain a lower bound that is at least as sharp as that obtained by the Linear Programming relaxation. We associate the Lagrangian multipliers u_i, $\forall i \in M$, with the constraints (5.1), which are equality constraints, and therefore the multipliers are unrestricted in sign. Therefore, we get[3]:

$$(CFL(u)) \quad min \sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+u_i)x_{ij} + \sum_{j=1}^{n} f_j y_j - \sum_{i=1}^{m} u_i$$

$$\sum_{i=1}^{m} d_i x_{ij} \leq s_j y_j \qquad \forall j \in N \tag{5.7}$$

$$\sum_{j=1}^{n} s_j y_j \geq d(M) \tag{5.8}$$

$$x_{ij} \leq y_j \qquad \forall i \in M,\, j \in N \tag{5.9}$$

$$x_{ij} \geq 0 \qquad \forall i \in M,\, j \in N \tag{5.10}$$

$$y_j \in \{0,1\} \qquad \forall j \in N \tag{5.11}$$

[2](UFL) was formulated in Section 2.1.

[3]The formulation presented here is slightly different than the one presented in [3] because we wanted to keep the same notation of the previous chapters.

Consequently, the Lagrangian function is:

$$L(u) = min\Big\{\sum_{i=1}^{m}\sum_{j=1}^{n}(c_{ij}+u_i)x_{ij} + \sum_{j=1}^{n} f_j y_j - \sum_{i=1}^{m} u_i : (5.7) - (5.11)\Big\} \tag{5.12}$$

where the minimum is calculated with respect to the variables x_{ij} and y_j. According to the results in [3], we are interested in reformulating (5.12) in a more compact way. To do so, we first calculate the minimum with respect to the variables x_{ij}, and then with respect to the variables y_j.

For each $j \in N$, by considering the case where $y_j = 1$, i.e., we open the depot j, we define:

$$v_j = min\Big\{\sum_{i=1}^{m}(c_{ij}+u_i)x_{ij} : \sum_{i=1}^{m} d_i x_{ij} \le s_j,\ 0 \le x_{ij} \le 1 \quad \forall i \in M\Big\} \tag{5.13}$$

where the minimum is calculated with respect to the variables x_{ij}. Then, we calculate the minimum with respect to the variables y_j:

$$u_0 = min\Big\{\sum_{j=1}^{n}(f_j+v_j)y_j : \sum_{j=1}^{n} s_j y_j \ge d(M),\ y_j \in \{0,1\} \quad \forall j \in N\Big\} \tag{5.14}$$

Therefore, we obtain:

$$L(u) = u_0 - \sum_{i=1}^{m} u_i. \tag{5.15}$$

with u_0 defined by (5.14), and v_j defined by (5.13). At this point, we study the Lagrangian Dual problem:

$$(LD) \quad L^* = \max_{u} L(u)$$

Observation 5.2.1. Let $c_{ij_i} = min\{c_{ij} : j \in N\}$ be the minimum unit transportation cost of client i, for each $i \in M$. In [3] it is proved that the optimal Lagrangian multipliers $\overline{u}$ can be found in the interval $[u^{min}, u^{max}]$, where:

- ◇ $u_i^{min} = min\{-c_{ij} : j \in N\}$
- ◇ $u_i^{max} = max\{-c_{ij} : j \in N \setminus \{j_i\}\}$

for each $i \in M$.

As a consequence, the Lagrangian Dual problem can be rewritten as:

$$(LD) \quad L^* = max\Big\{L(u) : u \in [u^{min}, u^{max}]\Big\} \tag{5.16}$$

5.2.1 Reformulation of the Lagrangian Dual problem

We are interested in formulating the Lagrangian Dual problem (5.16) as a suitable Linear Programming problem. We start by defining the following sets:

- ◇ $\{x_{ij}^{t_j} : t_j \in T_j\}$ is the set of the vertices of the feasible region of the Linear Programming problems (5.13), for each $j \in N$
- ◇ $\{y^q = (y_j^q)_{j\in N} : q \in Q\}$ is the set of the feasible solutions of (5.14)

Moreover, we define:

- ◇ $F_q = \sum_{j\in N} f_j y_j^q$ for each $q \in Q$; notice that this is the fixed cost of the feasible solution y^q of (5.14).
- ◇ $C_{t_j j} = \sum_{i\in M} c_{ij} x_{ij}^{t_j}$ for each $t_j \in T_j$ and for each $j \in N$; notice that, for each j, this represents, in a partial way, the cost of the vertex $x_{ij}^{t_j}$ of the feasible region of (5.13).

At this point, exploiting (5.15) to express the objective function, we can reformulate (5.16) as a Linear Programming problem as follows:

$$(LD') \quad max\ u_0 - \sum_{i=1}^{m} u_i$$

$$u_0 \le F_q + \sum_{j=1}^{n} v_j y_j^q \qquad \forall q \in Q \tag{5.17}$$

$$v_j \le C_{t_j j} + \sum_{i=1}^{m} u_i x_{ij}^{t_j} \qquad \forall t_j \in T_j,\ j \in N \tag{5.18}$$

$$u_i^{min} \le u_i \le u_i^{max} \qquad \forall i \in M \tag{5.19}$$

where (5.17) and (5.18) follow by the definition of u_0, i.e., (5,14) and v_j, for each $j \in N$, i.e, (5.13). The variables of (LD′) are: u_0, $(u_i)_{i\in M}$ and $(v_j)_{j\in N}$. We call (LD′) the *Dual Master Problem*, and in the following section we present its corresponding dual problem, which we call the *Primal Master Problem*.

5.3 Primal Master Problem of (CFL)

In this section we present the formulation of the dual problem of the Dual Master Problem (LD′), which we will refer to as the *Primal Master Problem*. Then, we mention that a Column Generation procedure can be employed in order to find a solution to this problem.

We start by associating the dual variables with the constraints of (LD′). In particular:

- ◇ the variables α_q, $\forall q \in Q$, will be related to the constraints (5.17)
- ◇ the variables $\beta_{t_j j}$, $\forall t_j \in T_j$, $j \in N$, will be related to the constraints (5.18)
- ◇ the variables $\underline{p}_i$ and $\overline{p}_i$, $\forall i \in M$, will be related to the constraints (5.19).

Using the Duality Theory of Linear Programming [2], we obtain[4]:

$$(LCFL') \quad min \sum_{q\in Q} \alpha_q F_q + \sum_{j=1}^{n} \sum_{t_j \in T_j} \beta_{t_j j} C_{t_j j} + \sum_{i=1}^{m} (\overline{p}_i u_i^{max} - \underline{p}_i u_i^{min})$$

$$\sum_{q\in Q} \alpha_q = 1 \tag{5.20}$$

$$-\sum_{j=1}^{n} \sum_{t_j \in T_j} \beta_{t_j j} x_{ij}^{t_j} + \overline{p}_i - \underline{p}_i = -1 \qquad \forall i \in M \tag{5.21}$$

$$-\sum_{q\in Q} \alpha_q y_j^q + \sum_{t_j \in T_j} \beta_{t_j j} = 0 \qquad \forall j \in N \tag{5.22}$$

$$\alpha_q \geq 0 \qquad \forall q \in Q \tag{5.23}$$

$$\beta_{t_j j} \geq 0 \qquad \forall t_j \in T_j,\ j \in N \tag{5.24}$$

$$\underline{p}_i, \overline{p}_i \geq 0 \qquad \forall i \in M \tag{5.25}$$

Observation 5.3.1. We notice that (LCFL′) is the Linear Programming relaxation of a formulation of (CFL) equivalent to the one presented in Section 5.1. In particular, the equivalent formulation we are talking about consists of studying the problem of selecting a subset of depots $S^q = \{j \in N : y_j^q = 1\}$, for a certain $q \in Q$, of total capacity at least $d(M)$, and choosing for each depot $j \in S^q$ some feasible client assignment $\{x_{ij}^{t_j} : t_j \in T_j\}$, with the aim of minimizing the total costs, and ensuring that every client's demand is satisfied (constraints (5.21)) and that there are no assignments from closed depots (constraints (5.22)).

Observation 5.3.2. The number of the variables of the formulation (LCFL′) grows exponentially with the number of depots and clients. However, in Section 3.2, we have noticed that in these situations it is useful to adopt a Column Generation procedure, as it avoids the explicit listing of all the columns, which are instead generated only if needed. Details of the method employed for addressing this particular problem can be found in [3].

[4]As well as for the Lagrangian relaxation of (CFL), the formulation presented here is slightly different from the one presented in [3].

Bibliography

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, 1993.

[2] G. Bigi, A. Frangioni, G. Gallo, S. Pallottino, and M. G. Scutellà, *Appunti di Ricerca Operativa*, SEU, 2014.

[3] S. Görtz and A. Klose, *A branch-and-price algorithm for the capacitated facility location problem*, European Journal of Operational Research, 179 (2007), pp. 1109–1125.

[4] L. A. Wolsey, *Integer Programming*, Interscience Series in Discrete Mathematics and Optimization, Wiley-Interscience, 1998.