Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Dottorato di Ricerca in Ingegneria dell'Informazione

Mixed-integer programming models and methods for bilevel fair network optimization and energy cogeneration planning

Doctoral Dissertation of:
Leonardo Taccari

Advisor:
Prof. Edoardo Amaldi

Tutor:
Prof. Federico Malucelli

The Chair of the Doctoral Program:
Prof. Carlo Fiorini

2015 – XXVII

Summary

This thesis addresses two relevant mixed-integer optimization problems, with application in telecommunications and energy systems, using mathematical programming techniques.

Part I focuses on a bilevel multi-commodity flow problem subject to max-min fair flow allocation, which arises in telecommunication networks with elastic demands. The problem is motivated by routing in Internet Protocol (IP) networks, where there are no prescribed demands to be satisfied, since the network provides a best-effort service. The network operator aims at maximizing a utility function (total throughput), by selecting the routing paths, while the bandwidth is allocated *fairly* by the transport protocol. Accordingly, we define the maximum-throughput Unsplittable Flow Problem subject to Max-Min Fair flow allocation (UFP-MMF) as the bilevel problem where, at the upper level, the routing paths maximizing the total throughput are sought, while, at the lower level, the flow is allocated to each origin-destination pair maximizing a fairness measure. In this work, we analyze the problem from a theoretical standpoint, and investigate various MIP-based solution approaches.

In Chapter 1, we recall some basic concepts and provide some background on max-min fairness and on bilevel programming. In Chapter 2 we discuss the motivation, summarize related work and investigate theoretical properties of UFP-MMF, including complexity results and its price of fairness. In Chapter 3, we discuss how the bilevel problem can be cast as a single-level mixed-integer programming problem exploiting the concept of the bottleneck arcs. The single-level problem turns out to be very challenging, since the origin-destination paths have to be elementary (without repeated nodes), and the bottleneck conditions are, in essence, equilibrium constraints. In particular, we develop two different approaches: a branch-and-cut algorithm based on an arc formulation, where subtour elimination constraints are separated, and a branch-and-price algorithm based on a path formulation. We also propose a heuristic based on local search that provides good feasible solutions in a short computing time. In Chapter 4 we discuss relaxations and variants of UFP-MMF, involving alternative fairness criteria. Since in both the branch-and-price and the branch-and-cut algorithm eliminating subtours is an essential aspect, in Chapter 5 we investigate integer programming formulations to prevent subtours in elementary path problems. We describe several extended formulations, and one based on generalized cutset inequalities (GCS). We prove an equivalence result, for elementary longest path problems, between the GCS-based formulation and the strongest known extended formulation, and summarize computational experiments to compare the performance of the formulations in a full branch-and-cut framework. Finally, in Chapter 6 we report computational results, on different topologies taken from the SNDlib [OWPT10], with the described approaches for UFP-MMF and its relaxations. We also discuss the structure of the solutions, and draw some concluding remarks. From the computational experiments, the best exact method appears to be the branch-and-cut algorithm with separation of the GCS inequalities, although optimality remains out of reach on a good fraction of the instances. The heuristics are crucial in obtaining close-to-optimal solutions for the hardest instances in short computing times (below 10 minutes), while the proposed relaxations are able to provide tight dual bounds.

Part II addresses an operational planning problem in energy cogeneration system with thermal storage, where one has to determine the operations of a set of (co)generation units (i.e., on/off status and production level), over a given time horizon, in order to satisfy users demands and minimize the operating costs. Chapter 7 includes a brief survey on previous and related work, and its relationship with two well-known problems in operations research: unit commitment and production planning. Chapter 8 discusses mixed-integer programming approaches, in the spirit of the techniques used for production planning [PW06], focusing in particular on some basic variants of the problem. For the variant with constant production upper and lower bounds, a dynamic programming-based polynomial algorithm is described. In Chapter 9, we introduce a revised Γ-robust formulation, inspired by the one by Bertsimas and Thiele in [BT06], that we extend to the operational planning problem. Finally, in Chapter 10 we discuss real-world cases, where the problem is solved as a MINLP or as a MILP, exploiting a piecewise-linear approximation. The problem can typically be solved with exact methods for time horizons between one day and a few weeks. When annual economic incentives have to be taken into account, we also propose a MILP-based rolling-horizon heuristic that can be applied to larger-scale problems provided by an Italian energy company.

Contents

Part I

Bilevel unsplittable flow problems subject to fair flow allocation

CHAPTER 1

Preliminaries

The first part of this thesis addresses a bilevel network optimization problem involving fair allocation of the bandwidth. This chapter introduces background material and previous work on the notion of fairness and, specifically, the max-min fairness criterion in network optimization problems. Then, we give a brief summary and some pointers to the literature of bilevel programming.

1.1 Fairness

Let us consider a system where a set of resources has to be shared among n users, and let $X \subseteq \mathbb{R}^n$ be the set of feasible allocation vectors $\underline{x} \in X$. Let us also assume there is a vector of real-valued utility functions $(g_1, \ldots, g_n)$ that maps a resource allocation vector $\underline{x}$ to the utility level u_i of each user $i = 1, \ldots, n$, and let $U \subseteq \mathbb{R}^n$ the set of feasible utility vectors, i.e., $U = \{\underline{u} = (g_1(\underline{x}), \ldots, g_n(\underline{x})) \mid \underline{x} \in X\}$.

Since there are multiple subjects, each with its own utility function, there can be multiple conflicting objectives. Then, we can consider the multi-objective optimization problem defined as:

$$\max\{(u_1, \ldots, u_n) \mid \underline{u} \in U\},$$

for which a set of Pareto-optimal solutions exists.

Among those, each rational, but selfish, user i would like to select the one maximizing its own utility u_i. On the other hand, a central decision maker might want to determine the allocation maximizing the total utility in the system $\sum_{i=1}^{n} u_i$, that is, the so-called *social optimum*. An alternative could be determining the most *efficient* solution, defined as the one where the usage of the available resources $\sum_{i=1}^{n} x_i$ is maximized. If the utility level of each player is proportional to their resource allocation, the two objectives are equivalent.

In many settings, no priorities among the users exist, but a solution that maximizes the overall utility might not be desirable, if it is considered *unfair* by some of the involved players.

The concept of fairness is rather subjective, and it has no universally accepted definition, but its importance has been widely acknowledged in several fields, ranging from economic theory to communication networks. The foundational concepts of fairness and equity have been studied since the beginning of political economics and, more recently, in social choice theory. As an example, classic *utilitarianism*, according to the principle of utility which dates back to the philosopher Jeremy Bentham [Ben79], prescribes an allocation which maximizes the sum of utilities (the social optimum). On the contrary, *Rawlsian justice* gives priority to the players that are the least well off [Raw71]:

> A scheme is unjust when the higher expectations, one or more of them, are excessive. If these expectations were decreased, the situation of the least favored would be improved. [...] Social and economic inequalities are to be arranged so that they are both (a) to the greatest expected benefit of the least advantaged and (b) attached to offices and positions open to all under conditions of fair equality of opportunity.

The point (a) in the above passage is referred to by Rawls as the *difference principle*, which posits:

> [I]n a basic structure with n relevant representatives, first maximize the welfare of the worst off representative man; second, for equal welfare of the worst-off representative, maximize the welfare of the second worst-off representative man, and so on until the last case which is, for equal welfare of all the preceding $n-1$ representatives, maximize the welfare of the best-off representative man. We may think of this as the lexical difference principle.

A criterion which is essentially equivalent to the Rawlsian concept of equality and has been widely studied in the operations research literature is *max-min fairness*, where the minimum resource allocation is maximized, then the second-worst, the third-worst, and so on. Another informal definition states that an allocation is max-min fair if there is no way to increase the utility of any element i without decreasing the utility of an element j with a smaller or equal utility[1].

As an example, borrowed from [NP08], consider the problem of distributing fairly 1 liter of beer to three customers with different glass capacities. The allo-

[1]Note that these definitions are only equivalent if U is convex.

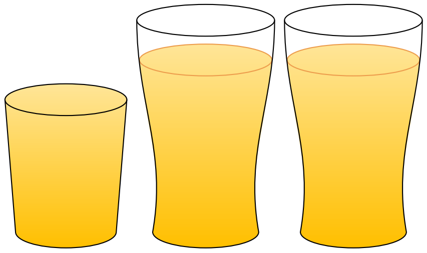


Figure 1.1: *Example of max-min fair allocation. The allocation is fair in the sense that the resource is allocated equally to all players as far as their capacity allows it.*

cation depicted in Figure 1.1 is max-min fair, because no customer can rationally claim more beer that what has been allocated. Customer 1 has reached its maximum capacity, while Customer 2 and 3 have the same level of beer.

1.1.1 Max-min fairness (MMF)

From now on, we will assume that the utility level for each user is equivalent to their allocation level, i.e., $u_i = x_i$ for $i = 1, \dots, n$. Then, we will always speak of fairness in the allocation space X, rather than in the utility space U.

Let us define max-min fairness formally. First of all, we need to define a preference relation over the set X of feasible allocation vectors.

Definition 1.1. The lexicographic order $\succeq_{lex}$ on a set $X \subseteq \mathbb{R}^n$ is the preference relation defined as $\underline{x} \succeq_{lex} \underline{x}'$ if either $\underline{x} = \underline{x}'$, or there exists an integer l, with $1 \leq l \leq n$, such that $x_q = x'_q$ for all $q < l$ and $x_l > x'_l$.

Let $\sigma : X \to \mathbb{R}^n$ be the sorting operator permuting the components of $\underline{x}$ in nondecreasing order, i.e., such that $\sigma_i(\underline{x}) \leq \sigma_j(\underline{x})$ whenever $i < j$.

Definition 1.2 (Max-min fairness)**.** An allocation vector $\underline{x}^* \in X$ is *max-min fair* (MMF) if, for any other vector $\underline{x} \in X$, $\sigma(\underline{x}^*) \succeq_{lex} \sigma(\underline{x})$, i.e., it is a solution to the sorted lexicographic maximization problem:

$$\underline{x}^* \in \text{arglexmax}\{\sigma(\underline{x}) \mid \underline{x} \in X\}. \tag{1.1}$$

This is as general a definition as possible. In the literature, several alternative definitions exist – usually, restrictions that are valid only under some additional assumptions on X. The following definition, slightly more restrictive, is used, e.g., in [BG92], and we refer to it as strong max-min fairness as in [Nil06].

Definition 1.3 (Strong max-min fairness)**.** An allocation vector $\underline{x}^* \in X$ is *strongly max-min fair* if, for any other vector $\underline{x} \in X$, if $\exists i$ such that $x_i > x_i^*$, then $\exists j$ such that $x_j^* \leq x_i^*$ and $x_j^* > x_j$.

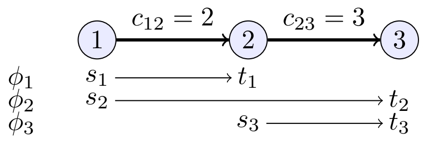
In other words, an allocation vector is strongly-MMF if there is no way to increase the allocation of any element i without decreasing the allocation of an element j with a smaller or equal allocation, as in Figure 1.1. A strongly-MMF vector is easily shown to be unique, and if a vector $\underline{x}$ is strongly-MMF, then it is also MMF. The two definitions are equivalent when the allocation set X is convex.

Concerning the algorithms to achieve max-min fairness, the main distinction is between problems where the allocation set X is convex and those where it is not. In the convex case, it is possible to find the max-min fair solution using an algorithm based on a sequence of, at most, n convex optimization problems (linear programs if X is a polyhedron). This algorithm is described in various forms in [Tom05], [OPT05], [Nil06], [RB07] and [NP08]. If the feasible allocation set X is not convex (a notable example: if it is discrete), the problem of finding a max-min fair solution is harder, but can be solved in a similar way, with algorithms based on a sequence of, at most, n non-convex subproblems – e.g., mixed-integer linear programs if X is described by linear constraints (except for the integrality ones). This technique is described, e.g., in [OW04] and [Nil06].

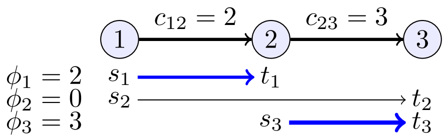
1.1.2 Max-min fair network flow allocation

Let us consider max-min fair allocation in the context of network flow problems. Consider a directed graph $G = (V, A)$ with arc capacities $c_{ij} \in \mathbb{R}^+$, and a set $K = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of origin-destination (OD) pairs for which a flow has to be routed over the arcs. The pairs (i.e., the network users) have elastic demands, in the sense that there is not a predetermined flow demand to satisfy. We start with the example of single-path routing, or unsplittable flow problem. The resources to be allocated are the arc capacities (e.g., bandwidth), and the allocation vector is the vector $\underline{\phi}$ of the flow values, where ϕ_i is the flow allocated to the i-th pair (s_i, t_i).

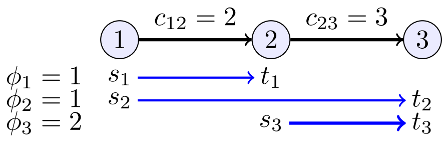
As an example of max-min fair allocation in a network flow problem, consider the following linear graph with three nodes $V = \{1, 2, 3\}$, two arcs $A = \{(1, 2), (2, 3)\}$, three origin-destination pairs $K = \{(1, 2), (1, 3), (2, 3)\}$, and fixed routing paths as shown in the following figure:



The flow for pair (s_1, t_1) is routed from node 1 to node 2 on the arc $(1, 2)$; the flow for pair (s_2, t_2) is routed from node 1 to node 3 on the path containing the consecutive arcs $(1, 2)$ and $(2, 3)$; the flow for pair (s_3, t_3) is routed from node 2 to node 3 on the arc $(2, 3)$. Maximizing the throughput over the network yields the allocation vector $\underline{\phi} = (2, 0, 3)$, with max total throughput $\tau = 5$, as follows:



The allocation is not fair, in the sense that the pair (s_2, t_2) has an allocation $\phi_2 = 0$ that could be increased by reducing *larger* allocations, as follows:



The flow allocation vector $\underline{\phi} = (1, 1, 2)$ satisfies the definition of MMF. Observe, however, that it has a smaller total throughput $\tau = 4$. In the next paragraph we will discuss how this allocation can be computed.

Computing the max-min fair flow allocation on fixed paths

Consider the case where, for each pair, each origin and destination is connected via a single routing path which is known a priori. Let us denote with $D \in \mathbb{R}^{|A|\times|K|}$ the arc-pair incidence matrix, where the entry $d_{ij}^{st} = 1$ if the arc (i, j) belongs to the s-t path, and 0 otherwise. The problem of finding a max-min fair flow allocation over the fixed paths can be written as:

$$\text{lexmax} \quad \sigma(\underline{\phi}) \tag{1.2}$$

$$s.t. \quad D\underline{\phi} \leq \underline{c} \tag{1.3}$$

$$\underline{\phi} \geq 0, \tag{1.4}$$

where σ is the sorting operator previously mentioned, and the Constraints (1.3)–(1.4) define the set of feasible flow allocations according to the arc capacities and the given paths. Since the feasible region is a (bounded) polytope, this is a convex max-min fairness problem. As such, there exists a unique solution, that can be found by the so-called *water filling* algorithm, which was introduced in [BG92].

Let $\mathcal{N} \subseteq K$ be the set of non-blocked pairs, i.e., those whose flow allocation can still be increased starting from the current flow allocation $\underline{\phi}$, and $\mathcal{E} \subseteq A$ be the subset of arcs which are not saturated by the current flow allocation vector $\underline{\phi}$. Let us denote by $\mathcal{U}(i, j) \subseteq K$ the set of pairs whose path contains the arc (i, j), i.e., the (s, t) such that $d_{ij}^{st} = 1$.

Algorithm 1.4 (Water filling, [BG92])**.**

Step 0: *Initialize* $\underline{\phi} \leftarrow \underline{0}$, $\mathcal{N} \leftarrow K$, $\mathcal{E} \leftarrow A$

Step 1: *Find a saturating arc and its allocation value:*

$$\begin{aligned} \eta^* \leftarrow \max \quad & \eta \\ s.t. \quad & \eta \left(\sum_{(s,t)\in\mathcal{N}} d_{ij}^{st} \right) \le c_{ij} \qquad\qquad (i,j) \in \mathcal{E}, \end{aligned}$$

where $\sum_{(s,t)\in\mathcal{N}} d_{ij}^{st}$ is the number of non-blocked pairs sharing an arc.

Step 2: *For all $(s,t) \in \mathcal{N}$:*

$$\phi_{st} \leftarrow \phi_{st} + \eta^*.$$

For all $(i,j) \in \mathcal{E}$:

$$c_{ij} \leftarrow c_{ij} - \eta \left(\sum_{(s,t)\in\mathcal{N}} d_{ij}^{st} \right).$$

For all $(i,j) \in \mathcal{E}$ such that $c_{ij} = 0$:

$$\mathcal{E} \leftarrow \mathcal{E} \setminus \{(i,j)\},$$

$$\mathcal{N} \leftarrow \mathcal{N} \setminus \mathcal{U}(i,j).$$

Step 3: *If $\mathcal{N} = \emptyset$,* **stop**. *Otherwise, go to* **Step 1**.

The maximization problem in Step 1 does not require solving a linear program, as it can be solved by inspection, its solution being:

$$\eta^* = \min \left\{ \frac{c_{ij}}{\sum_{(s,t)\in\mathcal{N}} d_{ij}^{st}} \mid (i,j) \in \mathcal{E} \right\}.$$

The idea of the algorithm, as implied by the name, is to increase all the flow allocations simultaneously by a quantity η, until an arc is saturated. The flow value for pairs sharing the saturated arc cannot increase further, hence their value is fixed and they are removed from the set $\mathcal{N}$. The procedure is then repeated on the residual graph, where the arc capacities are reduced by the amount of flow already allocated, until all the pairs cannot increase anymore. It is worth noting that the water filling algorithm can be seen as a specialization of more general algorithms to achieve the max-min fair allocation on convex sets.

1.1.3 Max-min fair network optimization

If the paths are not fixed, but can be optimized, i.e., selected during the optimization, the routing paths and the flow allocations have to be determined jointly so to have a solution which is as (max-min) fair as possible. Several variants of this problem have been studied in the literature.

Optimized paths with single source: splittable (Megiddo, 1974) and unspittable flows (Kleinberg et al., 2001)

In [Meg74], Megiddo studies the case of max-min fair splittable flows with a single source s and multiple sinks $T = \{t_1, \ldots, t_n\}$, where the MMF condition is imposed on the vector containing the values of the flows entering each sink t_i. The problem is convex and a polynomial-time algorithm, close in spirit to the one that can be used for general convex MMF problems, is shown to be correct.

Following Megiddo's work, Kleinberg, Rabani and Tardos [KRT01] consider the unsplittable-flow variant of the single-source problem, where, for each pair, the routing path connecting the origin and the destination can be selected among all possible elementary paths[2]. Each origin-destination flow is unsplittable, and must be routed along a single path. In this unsplittable case, the problem is not convex, and already $\mathcal{NP}$-hard. Nevertheless, the authors describe an approximation algorithm by considering a problem where the MMF condition is relaxed by a multiplicative factor, and the flows are restricted to be powers of 2.

Optimized paths with unsplittable flows

In the general case where there are multiple sources and sinks, one considers a set of origin-destination pairs $(s, t) \in K$, for which a single routing path and a flow allocation have to be determined so to yield an allocation which is as fair as possible.

The problem is nonconvex and $\mathcal{NP}$-complete. Algorithms for the problem consist in a sequence of at most $|K|$ mixed-integer programming problems: at iteration l, the l-th smallest flow allocation *value* is determined, but not which origin-destination pair will take that allocation in the optimal solution. For the interested reader, a detailed description of such algorithms can be found in [Nil06, Chap. 4], [OŚ06], [OPT05].

1.2 Bilevel programming

Classical mathematical programming problems typically assume a single objective function and a single decision maker (DM) that controls all the decision variables.

A bilevel optimization problem accounts for two decision makers that control only a subset of the decision variables. The so-called *leader* is the DM at the upper level, that makes his decision first, on a subset of variables. The *follower* (lower level) reacts optimally to the decision of the leader, according to his own set of constraints and his objective function. The leader has perfect knowledge of the follower's behavior, hence the decision of the leader has to take into account the reaction of the follower.

[2]We say a path is elementary if no vertices appear more than once in it. Paths with this property are sometimes called *simple*, (see e.g. [CLRS01]).

Accordingly, a bilevel program is a mathematical programming problem where one or more mathematical programs appear in the constraints, and can be written as follows:

$$\begin{aligned}
\min_{x,y} \quad & F(\underline{x}, \underline{y}) && (1.5)\\
s.t. \quad & G(\underline{x}, \underline{y}) \leq 0 && (1.6)\\
& \underline{x} \in X && (1.7)\\
& \underline{y} \in \operatorname*{argmin}_{y} f(\underline{x}, \underline{y}) && (1.8)\\
& \qquad s.t.\ g(\underline{x}, \underline{y}) \leq 0 && (1.9)\\
& \qquad \quad \underline{y} \in Y. && (1.10)
\end{aligned}$$

The upper and lower-level variables are, respectively, $\underline{x} \in X$ and $\underline{y} \in Y$. Similarly, F and f are the upper and lower-level objective functions, while G and g are the upper and lower level constraints (notice that they may involve both sets of variables). We remark that this formulation refers to the optimistic scenario, where we assume that, among equivalent optimal solutions for the follower, the leader can select the one which is best for its objective function. Viceversa, in the pessimistic setting, the follower is fully adversarial, in the sense that it picks the (lower-level) optimal solution which is worst for the leader.

Bilevel problems are a special case of general multilevel problems, where a whole hierarchy of decision makers is considered [VC94]. The concept of bilevel problem can be traced all the way back to the work of Stackelberg [VS52] in economic game theory, which introduced a class of games involving agents at two levels, the leaders and the followers, with a hierarchical structure analogous to the one found in bilevel programming. Stackelberg considered, at the lower level, a Nash equilibrium rather than an optimization problem, much in the spirit of what are known today as mathematical programs with equilibrium constraints (MPEC) [LPR96]. This class of problems can be seen as a special case of bilevel programming, since MPECs are essentially equivalent to bilevel programming problems where the lower level is convex and differentiable, and any MPEC can be formulated as a bilevel problem.

The simplest and most studied case, where both the problem at the upper and the lower level are linear, is known as linear bilevel programming and was shown to be $\mathcal{NP}$-hard by Jeroslow in [Jer85], while Hansen et al. [HJS92] proved it is strongly $\mathcal{NP}$-hard. In this case, one can exploit strong duality or the complementary slackness conditions to reformulate the problem as a single-level problem with nonconvex constraints, which, in turn, can be linearized obtaining a mixed-integer linear program. This combinatorial structure makes possible, e.g., approaches based on branch-and-bound algorithms [BM90]. An analogous reformulation is possible when the lower level is convex, so that it can be replaced with its Karush Kuhn Tucker (KKT) conditions. Both for the linear and convex case several approaches have been explored, such as methods based on vertex enumeration, gradient descent or penalty functions. For surveys and further details on bilevel programing, especially the linear case, plenty of resources are available. As a starting point, we refer to [CMS07], [VC94], [Dem03] and [BA93].

When the second level problem also controls integer variables, most reformulations and approaches that can be used for the convex variants cannot be easily extended, since there are no compact optimality conditions. In general, bilevel problems with integer variables at the second level cannot be expressed as a single-level mixed-integer program [Jer85], although in some special cases of practical relevance, this is possible (e.g., [ABF11, LSO12]).

Recently, the authors in [CCLW14] show that several known bilevel variants of the knapsack problem, which is one of the most basic and fundamental problem in combinatorial optimization, and whose standard version is $\mathcal{NP}$-hard[3], are complete for the Σ_2^p complexity class of the polynomial hierarchy [Pap03]. Nevertheless, there have been a few attempts at computational algorithms for general mixed-integer bilevel problems. The first is probably the one of Moore and Bard [MB90], that propose a basic branch-and-bound algorithm for pure integer bilevel problems. In [BM92], the same authors propose a specialized variant of the algorithm for pure binary bilevel problems. In [WY90], heuristic algorithms for general mixed-integer bilevel programs are described. The algorithm of [MB90] is extended by DeNegre in his PhD thesis [DeN11], where he also proposes a family of cutting planes for the pure integer case, which are, however, not generalizable to mixed-integer problems. To the best of our knowledge, no efficient algorithm for general mixed-integer bilevel problems has been proposed to this date, remaining an open challenge.

[3]Although it has a pseudopolynomial time algorithm and full polynomial time approximation scheme.

CHAPTER 2

Bilevel unsplittable flows subject to max-min fairness

In this chapter, the problem of bilevel maximum-throughput Unsplittable Flow subject to Max-Min Fair flow allocation (UFP-MMF) is introduced. Specifically, after describing the motivation for this work, we give a formal definition of the problem, provide some examples and discuss several theoretical properties, including complexity results, other structural properties, its relationship with respect to other network routing problems and establish its *price of fairness*.

2.1 Motivation

Best-effort delivery describes a network service in which the network does not provide any guarantees that data is delivered or that a user is given a certain quality of service level or priority. In a best-effort network, all the users obtain best-effort service, meaning that they obtain unspecified variable bit rate and delivery time, depending on the current traffic load. In this setting, user demands are said to be *elastic*: they are specified by an origin-destination (OD) pair without a prescribed flow value.

Let us consider the routing problems for best-effort service in Internet Protocol (IP) networks where, for instance, several users simultaneously download data between different hosts with no guaranteed rate, but wish to do so as fast as possible. The aim of the IP network operator is maximizing a utility function such as the total throughput. The operator can select the routing paths; however, it has no direct control over the underlying transport protocol (e.g., TCP) that actually allocates the bandwidth. The flow of each origin-destination pair is adapted by TCP based on the available capacity (which depends on the current traffic load) and the distributed congestion control mechanism is expected to allocate the flows in a fair way, that is, without privileging any user. Congestion avoidance algorithms in practice are complex [Jac88, CJ89], especially because of their highly dynamic and distributed nature, and the analysis of their behavior is not trivial [DKS89, KMT98, MR02]. A widely used notion of fairness in networks is that of Max-Min Fairness (MMF): indeed, in IP networks common congestion avoidance mechanisms aim at realizing a fair allocation of the flows over the routing paths provided by the IP layer, see e.g. [MR02, CJ89, Flo91] and the references therein, and the average transmission rates allocated by TCP can be reasonably approximated by max-min fair flows over the routing paths provided by the IP layer.

Hence, maximizing the throughput in a network under congestion avoidance mechanisms can be viewed as a bilevel problem where the network operator selects the routing paths for each user, and the transport protocol allocates the resources (available bandwidth) according to a fairness principle (MMF).

2.2 Balancing throughput and fairness

So far, most of the attention in the literature has been devoted to the problem of finding the fairest solution, which we have discussed in Section 1.1. When the routing paths are fixed, the water-filling algorithm finds the max-min fair solution in polynomial time. When routing paths are not given *a priori*, the problem consists in determining jointly a network routing (i.e., selecting a path for each origin-destination pair) and a flow allocation such that the allocation is as max-min fair as possible. An important distinction is between the cases with unsplittable or splittable routing, that is, where the flow between each origin-destination pair can be routed over one or many (not necessarily disjoint) paths. In the splittable case, the problem is convex and can be solved via a sequence of at most one LP problem per origin-destination pair, while, in the unsplittable case, the subproblems to solve are mixed-integer programs with binary variables. As we mentioned in Chapter 1, the first to study max-min fair problems in directed graphs, focusing on the single-source variant, were Megiddo [Meg74] for the splittable case, and Kleinberg, Rabani and Tardos [KRT01] for the unsplittable one.

More than one author have raised the concern that such fair solutions might not be sufficiently *efficient* and investigated the loss of efficiency due to imposing fairness (see, e.g., [BFT11]). Different, or relaxed, definitions of fairness have been investigated to make up for this inefficiency. In [SB08], an approach similar to that of traditional algorithms for convex max-min fairness is adopted to derive fair flow allocations balancing fairness and efficiency (throughput). In [AS04], the splittable case with a weighted MMF criterion is considered, so that it is possible to give different priorities to the users. The authors of [DHK+12] propose a relaxation of the notion of MMF with the definition of Upward Max-Min Fairness, where a locally max-min fair solution is obtained instead of the overall fairest one. In [DMS12] a practical algorithm to balance throughput and fairness for splittable routing is presented.

To the best of our knowledge, the work in [ACCG13], [ACT14] and in this thesis is the first to consider the fair flow allocation as a constraint of a more general network routing problem, rather than the optimization objective.

2.3 The bilevel problem: UFP-MMF

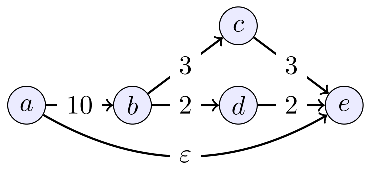
Let us formally define the bilevel problem of Maximum Unsplittable Flow subject to Max-Min Fair flow allocation (UFP-MMF).

Definition 2.1 (UFP-MMF). *Given a directed graph $G = (V, A)$ with capacities $c_{ij} \in \mathbb{R}^+$ and a set $K = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of origin-destination pairs, select a single routing path for each pair so as to maximize the total throughput $\sum_{(s,t)\in K} \phi_{st}$, subject to the constraint that the amount of flow allocated to the origin-destination pairs is Max-Min Fair with respect to the chosen paths.*

As motivated by the application, UFP-MMF is a bilevel problem where, at the upper level, the leader (network operator) selects a routing path for each origin-destination pair maximizing the total throughput, i.e., the sum of all the flows allocated to the pairs. The allocation is done, at the lower level, by the follower (the congestion avoidance protocol), that allocates the flows to the paths chosen by the leaders according to the Max-Min Fairness principle.

Example 2.2. Consider the graph with the arc capacities and the six OD pairs reported in Figure 2.1, where $c_{ae} = \varepsilon$ is a positive value smaller than 1. Note that for $i = 2, \ldots, 5$ there is a unique path to route (s_i, t_i), while there are two paths for (s_1, t_1) and three paths for (s_6, t_6).

In the optimal UFP-MMF solution, the flow of pair (s_6, t_6) is routed over the arc (a, e), avoiding congestion that would result in a decrease of the total throughput. The flow of pair (s_1, t_1) can be routed either through c or d. In case it is routed through c, the max-min fair flow allocation results in $\underline{\phi} = (1.5, 1.5, 1.5, 2, 2, \varepsilon)$ with a (suboptimal) total throughput of $8.5 + \varepsilon$. In case it is routed through d, the allocation vector is $\underline{\phi} = (1, 3, 3, 1, 1, \varepsilon)$, with $\tau = 9 + \varepsilon$, which is the optimal solution.



$(s_1 = b, t_1 = e)$
$(s_2 = b, t_2 = c)$
$(s_3 = c, t_3 = e)$
$(s_4 = b, t_4 = d)$
$(s_5 = d, t_5 = e)$
$(s_6 = a, t_6 = e)$

Figure 2.1: *Graph with six origin-destination pairs used in Example 2.2.*

The bilevel problem UFP-MMF has a natural interpretation from the perspective of game theory. The problem can be seen as a 2-player static Stackelberg (or leader-follower) game [VS52], with the leader (the network operator) that plays first, maximizing the objective function $\sum_{(s,t)\in K} \phi_{st}$, and one non-cooperative Nash follower (the transport protocol) that reacts maximizing a sorted lexicographical objective function. The pure Stackelberg equilibria, i.e., solutions where none of the players has an incentive to change unilaterally its (pure) strategy, are represented by the optimal solutions to UFP-MMF. The equilibria are not unique, and it is easy to verify that more than one equivalent optimal (leader-maximal) solutions can exist, as shown in Figure 2.2. Also other problems related to max-min fairness

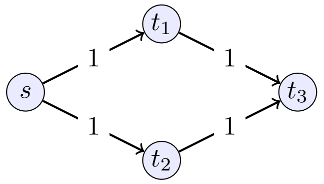


Figure 2.2: *Graph with 3 origin-destination pairs* $K = \{(s, t_1), (s, t_2), (s, t_3)\}$ *and multiple optimal solutions for UFP-MMF. Pairs* (s, t_1) *and* (s, t_2) *only have one path. For* (s, t_3)*, both path choices result in the same overall throughput* $\tau = 2$.

have been studied in the context of game theory; for sake of readability, we refer the reader to the end of the chapter (Section 2.8) for a brief discussion of them.

2.4 Difference with related problems in network optimization

If the leader has total control on both the path selection and the flow allocation, thus dropping the fairness constraint, UFP-MMF becomes a single-level maximum-throughput unsplittable flow problem. The maximum-throughput Unsplittable Flow Problem (UFP) is well known to be $\mathcal{NP}$-hard on arbitrary graphs, but we will see in Chapter 6 that, in practice, it appears to be significantly easier than UFP-MMF. Note also that UFP is a relaxation of UFP-MMF, as both problems have the same objective function and the feasible region of UFP-MMF is strictly contained into the one of UFP.

Viceversa, suppose that the follower has control on both paths and flow allocations. This is equivalent to replacing the max-throughput objective function of the leader with the MMF objective function. The problem would become a single-level non-convex max-min fair problem, where the overall fairest solution is sought over the set of all unsplittable flows (we denote this problem by UMMF).

Example 2.3. Consider again the graph in Figure 2.1. We have discussed in Example 2.2 that the bilevel UFP-MMF optimal vector is $\underline{\phi} = (1, 3, 3, 1, 1, \varepsilon)$, with a total throughput of $\tau = 9 + \varepsilon$, obtained routing (s_6, t_6) over the arc (a, e), and (s_1, t_1) through d.

It is easy to verify that the optimal value of UFP can be obtained by allocating a flow $\phi_1 = 0$ to the pair (s_1, t_1), and by routing a flow $\phi_6 = \varepsilon$ over the arc (a, e). The resulting flow allocation vector is $\underline{\phi} = (0, 3, 3, 2, 2, \varepsilon)$, with a total throughput $\tau = 10 + \varepsilon$, which is strictly greater than the optimal solution for UFP-MMF. Observe that, in UFP, the optimal solution often has flow allocations of value 0 corresponding to paths with high congestion on several arcs.

Consider, viceversa, the case where paths and flow allocations are optimized jointly, and the overall max-min fair solution is sought (UMMF). Assume that $\varepsilon < \frac{1}{3}$. The flow for (s_6, t_6) will not be routed over the arc (a, e), since ε is smaller than any allocation that would be granted to (s_6, t_6) using one of the other two paths, for any choice of (s_1, t_1). Therefore, both (s_1, t_1) and (s_6, t_6) will go through either c or d. If both are routed through d, the allocation vector reads $\underline{\phi} = (\frac{1}{3}, 3, 3, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, with total throughput $7 + \frac{1}{3}$. If both are routed through c, the allocation vector reads $\underline{\phi} = (1, 1, 1, 2, 2, 1)$, with total throughput 8. If one is routed through d and one through c, both top and bottom arcs would have congestion 2, and the allocation vector would read either $\underline{\phi} = (1, \frac{3}{2}, \frac{3}{2}, 1, 1, \frac{3}{2})$ or $\underline{\phi} = (\frac{3}{2}, \frac{3}{2}, \frac{3}{2}, 1, 1, 1)$, both yielding the sorted lexicographically maximal solution, that has total throughput $7 + \frac{1}{2}$. Notice that two equivalent solutions exists, as uniqueness of the max-min fair solution is not guaranteed when the utility set is not convex.

At this point, it should be clear that:

- on the one hand, maximizing the total throughput does not guarantee fairness. Indeed, the optimal UFP solution usually contains several allocations with value 0, which cannot appear in a max-min fair solution by definition[1];

- on the other hand, maximizing fairness according to the MMF principle may lead to a solution which is far from optimal with respect to the total throughput.

[1]Assuming the network contains no arcs with capacity 0.

It is possible to quantify what is the worst-case gap between the optimal value of the bilevel UFP-MMF and the single-level variants we have just described. Unfortunately, this quantity can be arbitrarily large in both cases.

Proposition 2.4. *The gap in terms of total throughput between an optimal solution to UFP-MMF and an optimal solution to UFP can be arbitrarily large.*

Proof. Consider the example of the linear graph in Figure (2.3), with m arcs of unit capacity and the set of origin-destination pairs $K = \{(s_1, t_1), \ldots, (s_k, t_k)\}$. Each pair (s_i, t_i) has a corresponding fixed path p_i, thus the leader has actually no choice on the routing paths.

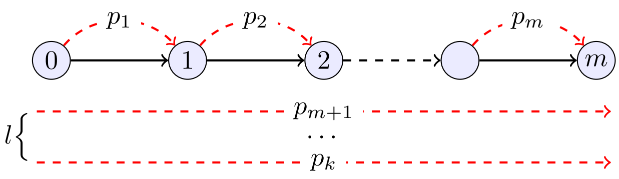


Figure 2.3: *Linear graph used in Proposition 2.4.*

Each of the pairs (s_i, t_i) with $i = 1, \ldots, m$ has origin $s_i = i - 1$ and destination $t_i = i$. The remaining l pairs, with $i = m+1, \ldots, k$, where $l = k - m$, have origin in the node 0 and destination in m, so they use all the m arcs simultaneously, causing a $l + 1$ congestion on every arc.

The maximum total throughput for UFP has value m. In the optimal solution, the pairs $i = 1 \ldots m$ will be allocated a flow of value 1, while the remaining pairs will have an allocated flow of value 0.

On the other hand, the total throughput of UFP-MMF is given by:

$$\frac{m}{l+1} + \frac{l}{l+1} = \frac{m+l}{l+1} = \frac{k}{k-m+1}, \tag{2.1}$$

since each arc is shared by $l + 1$ pairs and, according to the notion of max-min fairness, they are all allocated the same amount of capacity. As the number of pairs l grows, the total throughput approaches 1:

$$\lim_{l \to \infty} \frac{m+l}{(l+1)} = 1, \tag{2.2}$$

while, if l is constant and $m \to \infty$, the limit is divergent:

$$\lim_{m \to \infty} \frac{m+l}{(l+1)} = \infty. \tag{2.3}$$

Then, for any constant $\varepsilon > 0$, it is possible to build an example where the ratio between UFP-MMF and UFP is smaller than ε. As an example, take $l_\varepsilon > \frac{1}{\varepsilon} - 1$.

Then the ratio tends to a quantity which is smaller than ε:

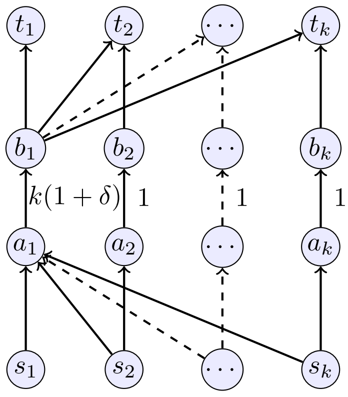
$$\lim_{m\to\infty} \frac{m + l_\varepsilon}{m(l_\varepsilon + 1)} = \frac{1}{l_\varepsilon + 1} < \varepsilon, \tag{2.4}$$

and by the very definition of limit, there exists a $m_\varepsilon > 0$ such that $\frac{m_\varepsilon + l_\varepsilon}{m_\varepsilon(l_\varepsilon+1)} < \varepsilon$. □

Let us now consider the problem of finding the max-min fair solution over the (non-convex) feasible region of flow allocation vectors corresponding to all possible unsplittable flows (UMMF).

Proposition 2.5. *The gap in terms of throughput or flow allocation between an optimal solution to UFP-MMF and an optimal solution to UMMF can be arbitrarily large.*

Proof. Consider the example in the figure below, with k origin-destination pairs, capacity 1 for the arcs (a_i, b_i), $i = 2, \ldots, k$, capacity $k(1+\delta)$ for the arc (a_1, b_1), with $\delta > 0$, and an arbitrarily large capacity for the other arcs.



It is easy to verify that $\sigma(\underline{\phi}) = (1, \ldots, 1, k(1+\delta))$, with a total throughput of $k(1+\delta)+k-1$, is an optimal solution to UFP-MMF, while $\sigma(\underline{\phi}) = (1+\delta, \ldots, 1+\delta)$, with a total throughput of $k(1+\delta)$, is an optimal solution to the UMMF. For a fixed δ and an arbitrarily large k, these solutions differ by an additive factor of $k-1$ in terms of throughput. For a fixed k and an arbitrarily large δ, these solutions differ by an additive factor of δ in terms of smallest flow allocation. □

2.5 Complexity and approximability

Let us analyze the computational complexity of UFP-MMF. As in most unsplittable flow problems, the notion of edge-disjoint paths in a graph plays a central role.

Definition 2.6 (Edge-disjont paths)**.** Two paths p and p' in a directed graph $G = (V, A)$ are said to be (pairwise) edge-disjoint if there is no edge in A that belongs to both paths.

Complexity and inapproximability results for UFP-MMF can be derived exploiting the connection of the UFP-MMF variant where all the arc capacities are equal to 1, with the $\mathcal{NP}$-complete k-EDP problem, that consists of deciding whether a directed graph G with k origin-destination pairs admits k edge-disjoint paths.

Proposition 2.7. *UFP-MMF is $\mathcal{NP}$-hard even when $c_{ij} = 1,\ \forall\,(i,j) \in A$*

Proof. By reduction from k-EDP. It suffices to observe that G admits k edge-disjoint paths if and only if the UFP-MMF instance on G, with the same k pairs and all $c_{ij} = 1$, admits an optimal UFP-MMF flow allocation with total throughput of value k. If this is the case, since each origin-destination pair achieves the maximum flow of value 1, k edge-disjoint paths are used. □

Proposition 2.8 (2-inapproximability)**.** *UFP-MMF is $\mathcal{NP}$-hard to approximate within a factor smaller than 2.*

Proof. By reduction from the $\mathcal{NP}$-hard problem 2DIRPATH [FHW80], that, given distinct vertices s_1, s_2, t_1, t_2, consists of deciding whether there exist two edge-disjoint paths, one from s_1 to t_2 and the other from s_2 to t_2. G admits two edge-disjoint paths if and only if the UFP-MMF instance with the same graph G, the same two pairs and all $c_{ij} = 1$ admits an MMF flow allocation with total throughput of value 2. Thus, we can map a YES instance for 2DIRPATH to a UFP-MMF instance whose optimal flow allocation has total throughput of value 2. We can map a NO instance to a UFP-MMF instance whose optimal flow allocation has total throughput of value not greater than 1. Thus, unless $\mathcal{P}$=$\mathcal{NP}$, a 2-approximation algorithm for UFP-MMF does not exist. □

Theorem 2.9. *UFP-MMF is $\mathcal{NP}$-hard to approximate within a factor smaller than $m^{1/2-\epsilon}$ for any $\varepsilon > 0$.*

Proof. In [GKR+03] the authors provide the following strong inapproximability result for k-EDP:

Theorem 2.10 ([GKR+03])**.** *Consider a directed graph $G = (V, E)$ with $|A| = m$, and a set K of k origin-destination pairs. Then, for any $\varepsilon > 0$, it is $\mathcal{NP}$-hard to distinguish whether all k pairs in K can be connected by edge-disjoint paths or at most a fraction $\frac{1}{m^{1/2-\varepsilon}}$ of the k pairs can be connected.*

We now show that, if UFP-MMF can be approximated within any constant factor, then we can distinguish in polynomial time between YES and NO instances of EDP. Let us define a YES instance of k-EDP as an instance where all k pairs can be connected by edge-disjoint path, and NO instance one where at most a fraction $\frac{1}{m^{1/2-\varepsilon}}$ of the k pairs can be connected. For any instance I_1 of k-EDP, let us consider the instance I_2 of UFP-MMF with the same graph G, the same set of origin-destination pairs, and $c_{ij} = 1$ for all arcs $(i, j) \in A$.

For any YES instance I_1, the corresponding instance I_2 has a throughput of k, since all k origin-destination pairs can be connected via edge-disjoint paths and $c_{ij} = 1$ for all arcs.

Let us now turn to the instances I_2 corresponding to any NO instance I_1 of k-EDP. We know that, in any such instance I_1, at most $\frac{k}{m^{1/2-\varepsilon}}$ origin-destination pairs can be routed with edge-disjoint paths. With $c_{ij} = 1$ for all edges $(i, j) \in A$, the total flow for any instance I_2 is bounded above by:

$$\sum_{(s,t)\in K} \phi_{st} \leq \frac{k}{m^{1/2-\varepsilon}}, \tag{2.5}$$

since the maximum number of edge-disjoint paths is a valid upper bound for the optimal objective value of UFP-MMF.

Suppose that UFP-MMF can be approximated within a factor $\rho < m^{1/2-\varepsilon}$, and let $\mathcal{A}$ be the value of the approximate solution. For a YES instance I_1, we would have

$$\mathcal{A}(I_2) \geq OPT(I_2)/\rho \geq k/\rho,$$

where $OPT(I_2)$ is the optimal value of the corresponding UFP-MMF instance I_2, while, for a NO instance, from (2.5):

$$\mathcal{A}(I_2) \leq OPT(I_2) \leq \frac{k}{m^{1/2-\varepsilon}} < \frac{k}{\rho}.$$

We would then be able to distinguish YES and NO instances, contradicting Theorem 2.10. It follows that, unless $\mathcal{P} = \mathcal{NP}$, UFP-MMF cannot be approximated within a factor smaller than $m^{1/2-\varepsilon}$ for any ε. □

Note that UFP-MMF is hard not only due to the number of alternative paths. Indeed, even if the set of possible s-t paths were to be restricted to a subset $\bar{P}^{st} \subset P^{st}$, we show that UFP-MMF is already hard when $|\bar{P}^{st}| = 2$. We use the same construction used in a proof in [Kle96] and [Nil06].

Proposition 2.11. *UFP-MMF is $\mathcal{NP}$-hard even when $c_{ij} = 1$, $\forall\,(i, j) \in A$ and only two alternative paths exist for each origin-destination pair.*

Proof. From the $\mathcal{NP}$-complete PARTITION problem of deciding, given a set S of positive integers $a_1, \dots, a_k$, whether there is a set $S' \subset S$ such that

$$\sum_{j:a_j\in S'} a_j = \frac{1}{2}\sum_{i}^{k} a_i.$$

An instance of PARTITION can be reduced to the instance of UFP-MMF in Figure 2.4. where the two parallel arcs connecting s to v have capacity $c_u = c_l = \frac{1}{2}\sum_i^k a_i$, and the (v, t_i) arcs have capacity a_i. YES instances of PARTITION are mapped to UFP-MMF instance whose optimal flow allocation has total value $\sum_i^k a_i$, while we can map a NO instance to a UFP-MMF instance whose optimal flow allocation has value not greater than $\sum_i^k a_i - 1$. □

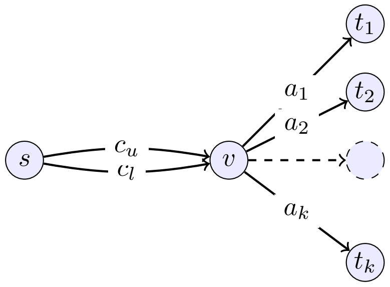


Figure 2.4: *Graph used in the proof of Proposition 2.11*

2.6 Other structural properties of UFP-MMF optimal flows

In this section, some further structural properties for the optimal flow allocations of UFP-MMF are discussed. We will derive a simple bound for the flow allocations, and show that, in general, no stronger bounds can be obtained. Interestingly, we also show that solutions with a larger congestion are in general not preferrable from the point of view of the leader.

Given a routing path for each (s, t) in K, let us define *congestion* of the solution the maximal number of overlapping (s, t)-paths on any arc of G. As a first observation, since only a single path can be used by each origin-destination pair, an arc can be used by at most k pairs (solution with congestion equal to k). In the worst case, the arc with the smallest capacity is shared among all k pairs and, in order to be MMF, its capacity is divided equally among the pairs. Then, for each pair $(s, t) \in K$, the optimal flow ϕ_{st} allocated to the pair is bounded below as $\phi_{st} \geq \frac{c_{min}}{k}$, where $c_{min} := \min_{(i,j)\in A}\{c_{ij}\}$ is the minimum capacity in the graph. The bound is tight in the somewhat pathological case where all the flows are bound to share the same arc, i.e., there is a *bridge* in the graph G between the component $S \subset V$ that contains all the origins $\{s_i\}_{i=1,\dots,k}$ and the component $T \subset V$ containing the destinations $\{t_i\}_{i=1,\dots,k}$, with $S \cap T = \emptyset$. With an additional assumption, that is satisfied in all non-trivial cases, a slightly stronger bound can be derived.

Proposition 2.12 (Valid lower bound)**.** *If there is at least a solution (i.e., a path selection) such that not all the flows share the same arc (i, j), the flows ϕ_{st} in an optimal solution of UFP-MMF are bounded below as follows:*

$$\phi_{st} \geq \frac{c_{min}}{k-1} \ \forall (s, t) \in K.$$

Proof. Shown by contradiction. Suppose that, in an optimal solution, a flow ϕ_{st} has value strictly smaller than $\frac{c_{min}}{k-1}$. By the properties of MMF, this can only happen if an arc with minimum capacity has congestion greater than $k - 1$, i.e., all the k pairs share such arc. Let us call $\underline{x}$ such solution, where the arc (i, j) with

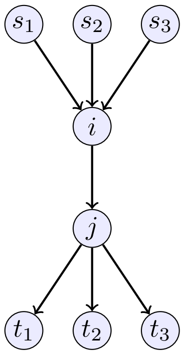


Figure 2.5: *Example of a* bridge. *The only possible solution has congestion $k = 3$.*

smallest capacity is shared among all k pairs and, in order to be MMF, the capacity is divided equally among the pairs. All the pairs are assigned a flow $\frac{c_{min}}{k}$, and the total throughput takes value $k\frac{c_{min}}{k} = c_{min}$.

By hypothesis, there is at least one feasible path p' for a pair (s, t) which does not contain any arc that is shared with *all* the other paths selected in the solution $\underline{x}$. Consider then the solution $\underline{x}'$ where (s, t) is assigned the path p', and all other pairs use the same path as in $\underline{x}$: we are decreasing the congestion of the solution to $k - 1$. Such move results in a flow allocation where:

- the path p' does not contain any arc where more than $k - 1$ paths overlap,
- all the other pairs have value $\frac{c_{min}}{k-1}$.

The total flow value is at least $\frac{c_{min}}{k-1} + (k-1)\frac{c_{min}}{k-1} = c_{min}\frac{k}{k-1}$ which is greater than c_{min}, thus $\underline{x}$ is not optimal, a contradiction. Hence $\frac{c_{min}}{k-1}$ is a valid lower bound under the given assumption. □

The assumption is usually fulfilled in practice, and the existence of a bridge can be verified in linear time [Tar74]. The bound is easily shown to be tight for any $k \geq 2$.

Proposition 2.12 implies that, in unit capacity graphs, a solution with congestion $k - 1$ is always preferred to one with congestion k. However, this property cannot be extended to graphs with arbitrary congestion. Assume one could determine a solution with minimum congestion on G^2, and let us indicate by q its congestion. Unfortunately, such solution is not necessarily optimal for UFP-MMF, and the value $\frac{c_{min}}{q}$ is *not*, in general, a valid lower bound on the flow allocation for an optimal solution of UFP-MMF: this is because a solution with smaller congestion does not necessarily yield an UFP-MMF allocation with larger total throughput than one with higher congestion. This is shown in the following example.

[2]The Congestion Minimization problem is known to be $\mathcal{NP}$-complete [ACG+10].

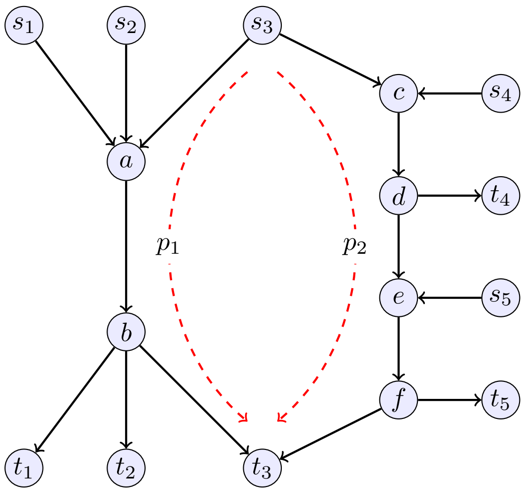


Figure 2.6: *Example where a solution with higher congestion is preferred to one with lower congestion.*

Example 2.13. Consider the graph with unit arc capacities and the 5 origin-destination pairs reported in Figure 2.6. Note that the path for $\phi_1, \phi_2, \phi_4, \phi_5$ is unique. The flow ϕ_3 can be routed over the path p_1 or p_2. Routing over p_1, the congestion of the solution is 3. The resulting MMF flow allocation vector is $\underline{\phi} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 1, 1)$ with a total throughput $\tau = 3$, and is optimal for UFP-MMF. If ϕ_3 is routed over the path p_2, the congestion of the solution is only $q = 2$, which is the minimum possible congestion. However, the resulting MMF flow allocation vector is $\underline{\phi}' = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$, with a smaller total throughput $\tau = 2.5$. Observe that $\sigma(\underline{\phi}') \succ_{lex} \sigma(\underline{\phi})$, and indeed $\underline{\phi}'$ is the overall max-min fair solution (UMMF). Note also that the value $\frac{c_{min}}{q} = \frac{1}{2}$ is not a valid lower bound for the flow allocation values, since the optimal allocation vector contains values equal to $\frac{1}{3}$.

Similar results show that, although one would like, in principle, to avoid solutions with a high congestion, it may pay off for the leader to have a high congestion for a few origin-destination pairs – in an attempt to move as far from the overall max-min fairness as possible, towards the maximization of the total throughput.

2.7 Price of fairness

Proposition 2.4 states that the gap between the value of the optimal solution of UFP, where the max-min fairness constraint is dropped, and the UFP-MMF one can be arbitrary large. It is possible to derive a tight lower bound on this gap, which is also known as price of fairness. The price of fairness, as defined in [BFT11], is the relative efficiency loss under a *fair* allocation compared to the one that maximizes the sum of the allocations (social optimum), i.e., the optimal solution to UFP. We will adopt in this section a notation which is close to the one in [BFT11].

Let X be the set of feasible allocations, and $\mathcal{S}(X) \in X$ the fairest allocation vector according to the fairness scheme in consideration. Let us denote with $FAIR(X;\mathcal{S})$ the sum of allocations of the fair solution $\mathcal{S}(X)$:

$$FAIR(X;\mathcal{S}) = \underline{1}^T\mathcal{S}(X). \tag{2.6}$$

The price of fairness, denoted by $PoF(X;\mathcal{S})$, is defined as the relative reduction in the sum of allocations under the considered fairness scheme $\mathcal{S}$, compared to the value $EFF(X)$ of the efficient, or *utilitarian*, solution, that maximizes the sum of the allocations:

$$PoF(X;\mathcal{S}) = \frac{EFF(X) - FAIR(X;\mathcal{S})}{EFF(X)}. \tag{2.7}$$

Values closer to 0 are preferable for the price of fairness, meaning that the fairness criterion is able to combine high system efficiency and fairness.

Let us indicate with $\mathcal{S}^{UFP-MMF}$ the fairness scheme based on the bilevel UFP-MMF, where fairness is enforced only on the second level of the problem (flow allocation). $EFF(X)$ is the value of the optimal solution of the unsplittable maximum multicommodity flow problem UFP. The PoF measures exactly what we lose when we impose fairness on the second level.

Proposition 2.14 (Price of fairness)**.** *The price of $\mathcal{S}^{UFP-MMF}$ for UFP with $k \geq 2$ users and unit arc capacities is bounded by:*

$$PoF(X;\mathcal{S}^{UFP-MMF}) \leq \begin{cases} 1 - \frac{4k}{(k+1)^2} & \text{if } k \text{ is odd} \\ 1 - \frac{4k}{k(k+2)} & \text{if } k \text{ is even.} \end{cases} \tag{2.8}$$

The bound is tight for all $k \geq 2$.

Proof. Assume that UFP has optimal value:

$$EFF(X) = m < k.$$

We have seen that UFP with unit capacities is equivalent to the problem of finding edge-disjoint paths in a graph. Then, there exists at least one optimal solution where m users have pairwise edge-disjoint paths with flow allocation of value exactly 1, while there are no edge-disjoint paths for the remaining $l = k - m$ users, whose allocated flow value is 0.

Now, consider the paths of such utilitarian solution. Clearly, the largest possible congestion for each of the k paths is $l+1$, since m paths are pairwise edge-disjoint, and only the other l paths can intersect. This implies that, under max-min fairness (i.e., applying the water-filling algorithm over the selected paths), the flow allocation for each user is not smaller than $\frac{1}{l+1}$. Then, the total utility under $\mathcal{S}^{UFP-MMF}$ is:

$$FAIR(X;\mathcal{S}^{UFP-MMF}) \geq \frac{k}{l+1} = \frac{k}{k-m+1}. \tag{2.9}$$

This is actually a tight bound, in the case where all of the remaining l users can only choose paths that intersect with all the other k selected paths, e.g., see Figure 2.3. The price of fairness is then bounded by:

$$PoF(X; \mathcal{S}^{UFP-MMF}) \leq 1 - \frac{k}{(k-m+1)m} \qquad \forall\, 1 \leq m \leq k. \tag{2.10}$$

Minimizing $\frac{k}{(k-m+1)m}$ over $1 \leq m \leq k$, for $m = \frac{k+1}{2}$ gives, if k is odd, the lower bound:

$$PoF(X; \mathcal{S}^{UFP-MMF}) \leq 1 - \frac{4k}{(k+1)^2}. \tag{2.11}$$

and, in case k is even, for either $m = \left\lceil \frac{k+1}{2} \right\rceil$ or $m = \left\lfloor \frac{k+1}{2} \right\rfloor$

$$PoF(X; \mathcal{S}^{UFP-MMF}) \leq 1 - \frac{4k}{k(k+2)}. \tag{2.12}$$

The bounds for PoF are tight, and can be attained by considering a linear graph as in Figure 2.3, with k users and m arcs, with $m = \left\lceil \frac{k+1}{2} \right\rceil$. As a nontrivial example, take $k = 4$ and $m = \lceil 5/2 \rceil = 3$. The utilitarian solution has value $EFF(U) = m = 3$, while the optimal solution of UFP-MMF has total throughput of value 2, matching the $\frac{1}{3}$ bound. $\square$

An equivalent result can also be independently derived from Theorem 1 in [BFT11], that, although being valid only for a convex and compact utility set U, can be adapted to UFP-MMF. Figure 2.7 displays the bound as k grows. Already with $k = 20$, the optimal solution for UFP-MMF can be, in theory, 80% less efficient than the optimal solution of UFP.

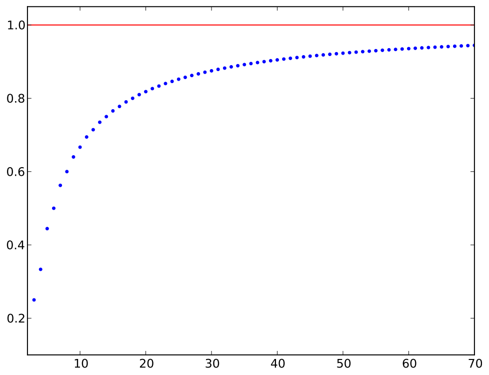


Figure 2.7: *PoF bound as the number of users k increases.*

2.8 Related problems in game theory

From a game theoretic point of view, games subject to max-min fair flow allocation have been studied in the last few years. Let us consider the sequential game MMF-

SEQ, first introduced in [YXF$^+$13] with the name MAXBAR, with a set of players corresponding to the set of origin-destination pairs K. Assume the game starts with a path assigned to each player. The strategy of a player corresponds to the selection of its path. At each turn, a player i selects a single (s_i, t_i) path, either the current or a different one, such that its own utility, i.e., its flow allocation, is maximized (*best response*). Once the new path is selected, the flow is immediately re-allocated over the paths according to the MMF principle (i.e., applying the water-filling algorithm). The computation of the (selfish) best-response consists of determining the widest *s*-*t* path over the graph where the arc capacities have been modified according to the available capacity-to-be, i.e., the capacity that a player would observe were that arc selected. The authors in [YXF$^+$13] show that this quantity can be computed in polynomial time, and that MMF-SEQ is guaranteed to converge to a pure Nash equilibrium. In this case, the equilibrium obtained by non-cooperative users is, in general, not efficient – i.e., it is not the solution with the optimal social utility, incurring what is usually called *price of anarchy* (PoA) [KP99]. The problem of finding the *social optimum* in the MMF-SEQ setting is exactly equivalent to UFP-MMF, although the authors do not attempt to solve the problem. Notice that the social optimum is, in general, not a Nash equilibrium for MMF-SEQ: there can be at least a user that might have an interest to change selfishly its path, thereby increasing its flow allocation but decreasing the total throughput (social utility).

An extension of this work is discussed by Harks et al. in [HHSS13, HHSS14], where the more general class of *progressive filling games* is introduced and analyzed. This class accounts also for games where the water filling algorithm is applied raising players' allocations at different rates. Along with further results on the PoA, the authors show that, for this class of games, strong equilibria always exist. A strong equilibrium is a stricter and more robust notion of Nash equilibrium, where no coalition of the players can change their routes and increase the allocation of each of its members.

CHAPTER 3

Mixed-integer programming approaches for UFP-MMF

UFP-MMF is a mixed-integer bilevel problem, and, as such, it is very challenging to solve. The lower-level problem, for fixed paths, is a sorted lexicographical maximization problem, where it is difficult to use primal-dual relations to characterize its optimal solutions. However, it has two crucial properties that stem from the definition of max-min fairness applied to network flows: *i*) once the paths have been selected, the feasible set of the second-level problem is convex, hence the MMF flow allocation is unique; *ii*) the problem of finding the max-min fair solution can be solved efficiently over fixed paths via the *water filling* algorithm. These properties suggest that, in a way, the second-level problem is *easy* on the set defined by the upper-level.

This chapter describes how, from such algorithm, it is possible to derive an explicit characterization of the max-min fair allocations. This allows us to recast the bilevel problem as a single-level problem, where the second-level max-min fair problem is replaced by its optimality conditions. Nevertheless, the resulting single-level mixed-integer linear problem is still challenging to solve, as it is, in essence, an unspittable multicommodity flow with additional global constraints. Two MIP-based exact approaches are proposed: a branch-and-cut algorithm for an arc formulation of the problem, and a branch-and-price for a path formulation. Since the problem is very challenging to tackle with pure exact methods, we also describe heuristics to find good feasible solutions.

For all the approaches described in this chapter, computational results will be reported and discussed in Chapter 6.

3.1 Characterization of the max-min fair allocation

In this section, we will characterize the optimality conditions of the lower level MMF-allocation problem starting from the definition of bottleneck arc.

Definition 3.1. Given a flow allocation, an arc $(i, j) \in A$ is *bottleneck* for the origin-destination pair $(s, t) \in K$ if:

1. the arc capacity c_{ij} is saturated,
2. the flow allocated to (s, t) is greater than or equal to the value of the flow allocated to any other (o, d)-pair on the arc (i, j).

On given routing paths, the max-min fair allocation can be determined via the water filling algorithm (Alg. 1.4). At each iteration, the water filling algorithm finds at least a bottleneck arc for an origin-destination pair, suggesting a simple characterization of the max-min fair flow allocation on given paths that was first described by Bertsekas and Gallager in [BG92, Chap. 6].

Proposition 3.2. *Given a directed graph $G = (V, A)$, a set K of origin-destination pairs and a simple path p^{st} for each $(s, t) \in K$, a feasible flow allocation vector $\underline{\phi}$ is max-min fair if and only if there is at least a bottleneck arc for each pair $(s, t) \in K$.*

For completeness, since in [BG92] a formal proof is not included, one is easily given.

Proof. Consider an allocation vector $\underline{\phi}$, and suppose that at least one (s, t) pair has no bottleneck arcs. If its routing path p^{st} contains no saturated arc, then the flow allocated to (s, t) can be increased, contradicting the fact that the solution is optimal. If p^{st} contains a saturated arc where ϕ_{st} does not have the largest value, then it is possible to decrease the value of another flow ϕ_{od}, larger than ϕ_{st}, to increase ϕ_{st} yielding a fairer solution.

Viceversa, consider a solution with allocation $\underline{\phi}'$, where every origin-destination pair has a bottleneck arc, and assume it is not max-min fair. Then, there must be another solution with allocation $\underline{\phi}''$ such that $\sigma(\underline{\phi}'') \succ \sigma(\underline{\phi}')$. For this to happen, at least one allocation ϕ'_{st} must be increased. Since all pairs have a bottleneck arc, increasing the allocation of (s, t) implies decreasing at least an allocation to another pair whose path contains a bottleneck arc of (s, t). But, on a bottleneck arc, $\phi'_{st} \geq \phi'_{od}$ for any $(o,d) \neq (s,t)$, thus the resulting sorted flow allocation vector would be lexicographically smaller than the previous, $\sigma(\underline{\phi}'') \prec \sigma(\underline{\phi}')$. Hence, $\underline{\phi}'$ is max-min fair. $\square$

The MMF solution is unique, since the set of feasible flow allocation vectors with fixed paths is convex. According to Proposition 3.2, the second level problem can be replaced with a set of linear constraints and binary variables that ensure the existence of a bottleneck arc for each (s,t) pair, as first noted in [Tom05] and later in [ACCG13] and [ACT14].

Let us indicate with f^{st}_{ij} the variables corresponding to the value of the (s,t)-flow allocation on the arc $(i,j) \in A$. For each $(s,t) \in K$, we denote with y^{st}_{ij} the binary variables such that, if $y^{st}_{ij} = 1$, then (i,j) is bottleneck. Then, for each (s,t), at least one y^{st}_{ij} must be equal to 1:

$$\sum_{(i,j)\in A} y^{st}_{ij} \geq 1. \tag{3.1}$$

We can characterize a bottleneck arc for pair (s,t) writing the bottleneck properties for an arc (i,j), with capacity $c_{ij} \geq 0$, as the following complementarity conditions:

$$\Big(\sum_{(o,d)\in K} f^{od}_{ij} - c_{ij}\Big) y^{st}_{ij} = 0 \tag{3.2}$$

$$(f^{st}_{ij} - f^{od}_{ij}) y^{st}_{ij} \geq 0 \qquad (o,d) \in K, (o,d) \neq (s,t), \tag{3.3}$$

where (3.2) ensures that, if an arc is bottleneck, it is saturated, and (3.3) impose that, if (i,j) is bottleneck for the pair (s,t), then the flow allocated to (s,t) must be larger or equal to that allocated to all other pairs using arc (i,j). Bilinear Constraints (3.2)–(3.3) can be linearized using the upper bound c_{ij} on the value f^{st}_{ij}, obtaining the following polyhedral characterization of the MMF allocation, which is valid for unsplittable flows over given selected paths:

$$\sum_{(i,j)\in A} y^{st}_{ij} \geq 1 \qquad (s,t) \in K \tag{3.4}$$

$$\sum_{(o,d)\in K} f^{od}_{ij} \geq c_{ij} y^{st}_{ij} \qquad (s,t) \in K \tag{3.5}$$

$$f^{st}_{ij} \geq f^{od}_{ij} + c_{ij}(y^{st}_{ij} - 1) \qquad (s,t) \in K, (o,d) \in K, (s,t) \neq (o,d). \tag{3.6}$$

The number of necessary constraints can be reduced introducing a variable $u_{ij} := \max\{f^{st}_{ij}\}$ representing the maximum flow allocation on the arc (i,j). Then, Constraints (3.6) can be replaced by:

$$u_{ij} \geq f^{st}_{ij} \qquad (s,t) \in K \tag{3.7}$$

$$f^{st}_{ij} \geq u_{ij} + c_{ij}(y^{st}_{ij} - 1) \qquad (s,t) \in K, \tag{3.8}$$

where the first inequality imposes that u_{ij} is an upper bound, and the second ensures that $f^{st}_{ij} = u_{ij}$ when the arc (i,j) is bottleneck for the pair (s,t).

3.2 (Partial) single-level arc formulation

A single-level mixed-integer programming formulation can be obtained by formulating a standard unsplittable multi-commodity flow model, where the objective function is the total throughput over the graph. Constraints (3.4)–(3.6) are added to the program in order to impose max-min fairness on the flow allocations.

Let us denote with x_{ij}^{st} the binary arc variables that take value 1 if the arc belongs to the selected s-t path, and 0 otherwise. Then, we can formulate the problem as follows:

$$\max \sum_{(s,t)\in K} \phi_{st} \tag{3.9}$$

s.t.

$$\sum_{(i,j)\in\delta^+(i)} f_{ij}^{st} - \sum_{(j,i)\in\delta^-(i)} f_{ji}^{st} = \begin{cases} \phi_{st} & \text{if } i = s \\ -\phi_{st} & \text{if } i = t \\ 0 & \text{else} \end{cases} \qquad i \in V, (s,t) \in K \tag{3.10}$$

$$\sum_{(s,t)\in K} f_{ij}^{st} \le c_{ij} \qquad (i,j) \in A \tag{3.11}$$

$$f_{ij}^{st} \le c_{ij} x_{ij}^{st} \qquad (i,j) \in A, (s,t) \in K \tag{3.12}$$

$$\sum_{(h,j)\in\delta^+(h)} x_{hj}^{st} \le 1 \qquad h \in V, (s,t) \in K \tag{3.13}$$

$$\sum_{(i,j)\in A} y_{ij}^{st} \ge 1 \qquad (s,t) \in K \tag{3.14}$$

$$\sum_{(o,d)\in K} f_{ij}^{od} \ge c_{ij} y_{ij}^{st} \qquad (i,j) \in A, (s,t) \in K \tag{3.15}$$

$$u_{ij} \ge f_{ij}^{st} \qquad (i,j) \in A, (s,t) \in K \tag{3.16}$$

$$u_{ij} \le f_{ij}^{st} + c_{ij}(1 - y_{ij}^{st}) \qquad (i,j) \in A, (s,t) \in K \tag{3.17}$$

$$f_{ij}^{st}, \phi_{st}, u_{ij} \ge 0 \qquad (i,j) \in A, (s,t) \in K \tag{3.18}$$

$$y_{ij}^{st}, x_{ij}^{st} \in \{0,1\} \qquad (i,j) \in A, (s,t) \in K. \tag{3.19}$$

Constraints (3.10)–(3.11) are standard flow conservation and capacity constraints. Constraints (3.12) ensure that the flow is 0 if the arc is not selected. Constraints (3.13) impose that only one path is chosen. Constraints (3.14)–(3.17) impose that the flow vector is MMF for the selected paths, according to Proposition 3.2. More specifically, Constraints (3.14) guarantee that at least an arc is bottleneck for each (s,t). Constraints (3.15) ensure that arc (i,j) is saturated if it is bottleneck for some pair (s,t). Constraints (3.16) make sure that u_{ij} is equivalent to the largest flow allocated over arc (i,j). Constraints (3.17) impose that the flow of a pair (s,t) through its bottleneck arc (i,j) is as large as the largest flow through (i,j) for all the other pairs.

Remarks:

1. Although all the ingredients of UFP-MMF are included, it must be noted that Formulation (3.9)–(3.19) is not yet sufficient to guarantee a correct solution for the problem, as it does not prevent subtours that can arise in a solution. The issue will be discussed in the following section.

2. A variable $y_{ij}^{st} = 1$ indicates a sufficient, but not necessary, condition for an arc (i,j) to be bottleneck for (s,t). In other words, in a feasible solution, an arc in a selected path p^{st} might satisfy the bottleneck conditions, but its corresponding variable y_{ij}^{st} might have value 0 – provided there is at least another bottleneck arc in p_{st} with $y_{ij}^{st} = 1$. This does not invalidate the correctness of the MMF conditions: it is not necessary to count the number of bottlenecks in a (s,t) path, it is sufficient to impose that one exists. However, this introduces a degree of symmetry in the formulation.

3. The bottleneck Constraints (3.14)–(3.17) are *global*: even when the paths have been fixed, it is not possible to consider each origin-destination pair independently, since each flow allocation cannot be determined without considering the allocations of all the other flows.

4. The bottleneck Constraints (3.14)–(3.17) have a very poor linear relaxation. Consider a solution $(\underline{\phi}, \underline{x}, \underline{f}, \underline{u})$ belonging to the set defined by (3.10)–(3.13) and (3.18)–(3.19), with $\underline{x}$ integer, and let us denote by p^{st} the path for each pair (s,t) induced by the arc incidence vector $\underline{x}$. Then, if the paths are sufficiently long, it is always possible to construct a bottleneck vector $\underline{y} \in \mathbb{R}_{+}^{|A|\times|K|}$ which satisfies the bottleneck constraints simply by taking:

$$y_{ij}^{st} := \begin{cases} \frac{1}{|p^{st}|} & \text{if } x_{ij}^{st} > 0 \\ 0 & \text{otherwise,} \end{cases}$$

where $|p^{st}|$ is the length of the (s,t)-path. Constraints (3.14) are clearly satisfied, since $\sum_{(i,j)\in A} y_{ij}^{st} = |p^{st}|\frac{1}{|p^{st}|} = 1$, while Constraints (3.15) are satisfied if $\sum_{(o,d)\in K} f_{ij}^{od} \geq c_{ij}\frac{1}{|p^{st}|}$, and Constraints (3.16) are satisfied if $u_{ij} \leq f_{ij}^{st} + c_{ij}(1 - \frac{1}{|p^{st}|})$. Both conditions hold true if $\frac{1}{|p^{st}|}$ is sufficiently close to 0, that is, if the path p^{st} is sufficiently long.

3.3 Existence of subtours

The formulation presented in the previous section is not sufficient to correctly describe the feasible region of UFP-MMF, as it does not prevent the formation of subtours (or cycles). In a classic maximum-throughput unsplittable multicommodity flow problem (UFP), subtours do not affect the optimal value of the objective function, and can be removed *a posteriori* from any optimal solution yielding an equivalent solution, so it is not necessary to account for them. However, subtours may (and do) arise when dealing with the MMF allocation at the lower level.

We remark again that the solutions with maximum throughput, if fairness on the flow allocation is not imposed (UFP), are generally *not* fair, and involve several allocations with value 0. Then, the reason a solution for Formulation (3.9)–(3.19) might have a subtour, unless explicitly prevented, is to bypass the max-min fairness constraints and allow (almost) arbitrary flow allocations to a subset of origin-destination pairs.

We have observed that this can be done mainly in two ways: *i*) setting an artificial bottleneck arc on a disconnected subtour: once a bottleneck arc has been imposed on a disconnected subtour, the flow allocation on the main s-t path is free; *ii*) using subtours with high congestion to decrease artificially the flow allocated to pairs which would "hurt" the total throughput.

Consider the example in Figure 3.1. The leader might want to select the a–b–c cycle for the origin-destination pair (s,t), and select the arc (a,b) as its bottleneck arc. Doing so, the flow ϕ_{st}, i.e., the one leaving s and entering t, is free from fairness constraints along the path s–e–t, because the bottleneck constraints are already fulfilled in the subtour.

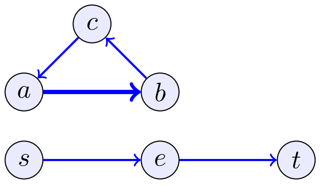


Figure 3.1: *The subtour a–b–c creates an artificial bottleneck arc (a,b), so that the flow allocated on the main s-t path is free from fairness constraints.*

In a similar fashion, see Figure 3.2, a subtour might be used to drive down a flow ϕ_{st}, if it is desirable to have as small an allocation to (s,t) as possible. To see why, suppose that there are "virtuous" origin-destination pairs[1], using either arc (a,c) or (c,t), whose flow allocation the leader would like to increase. Then, the leader might want to decrease the flow allocated to the pair (s,t), which uses simultaneously arcs (s,a), (a,c) and (c,t), overlapping with the virtuous flows. One way to do so, preserving the bottleneck constraint, is to route other flows (dashed line in the figure) along the subtour s–a–b, so that the arc (s,a) becomes a bottleneck for the pair (s,t) with an artificially inflated congestion, resulting in a small flow allocation ϕ_{st} and, potentially, allowing for a higher allocation to the other flows using the arcs (a,c) and (c,t).

Unfortunately, this means that it is necessary to introduce subtour elimination constraints for each $(s,t) \in K$. One way to do so is to prevent the subtours in the

[1]Intuitively: origin-destination pairs for which there exist paths with low congestion and high capacity.

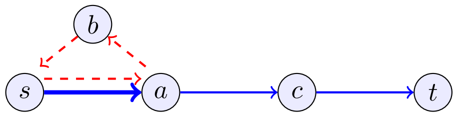


Figure 3.2: *The subtour s-a-b is used to drive down artificially the flow ϕ_{st}. Then, the allocation of flows, e.g., passing through (c,t) but not through (s,a), can be increased.*

arc formulation itself. We will then see that an alternative approach is to rewrite the MILP with a path formulation and use a column generation approach.

3.3.1 Subtour elimination in the arc formulation

To prevent subtours in the arc formulation, we can introduce subtour elimination constraints for each $(s,t) \in K$. While the s-t paths are generally not Hamiltonian, techniques similar to the ones applied for the Travelling Salesman Problem can be adapted to our case. More precisely, it is possible to adopt all the methods that are valid to model the elementary longest (or shortest) path problem as a mixed-integer program. The discussion on integer programming formulations for elementary-path problems deserves a chapter on its own, and we refer the reader to Chapter 5 for the details. For sake of completeness, we give here a short summary of the two main approaches that can be used.

Extended formulations

One way to prevent subtours entails the use of extended formulations with a polynomial number of auxiliary variables and constraints.

The strongest known extended formulation is based on multicommodity flows, and is obtained considering, for each (s,t) pair, an auxiliary unit flow q^{hst} to be delivered to each node h belonging to the s-t path:

$$q_{ij}^{hst} \leq x_{ij}^{st} \qquad h \in V_s,\ (i,j) \in A, (s,t) \in K$$

$$\sum_{(i,j)\in\delta^+(i)} q_{ij}^{hst} - \sum_{(j,i)\in\delta^-(i)} q_{ji}^{h} = \begin{cases} z_h^{st} & \text{if } i = s \\ -z_h^{st} & \text{if } i = h \\ 0 & \text{else} \end{cases} \qquad i \in V,\ h \in V_s,\ (s,t) \in K$$

$$\sum_{(i,h)\in\delta^-(h)} x_{ih}^{st} = z_h^{st} \qquad h \in V_s, (s,t) \in K$$

$$q_{ij}^{hst}, z_h^{st} \geq 0,$$

thus guaranteeing connectedness and elementarity. The formulation is very strong, although at the cost of $O(|V||A|)$ variables and constraints for each $(s,t) \in K$. A similar idea is used in the classic extended formulation for the asymmetric travelling salesman problem of Wong [Won80] and Claus [Cla84]. In [IMM09], an equivalent flow-based formulation for the elementary shortest path problem is discussed.

Cutting planes

An alternative is to use a cutting plane approach to prevent subtours: the experience of the TSP teaches that, in a branch-and-cut setting, dynamic separation of subtour elimination constraints (SEC) is likely to be computationally more attractive. The main idea is to strengthen the classic cutset inequalities for the ATSP, obtaining the following *Generalized Cutset* inequalities, as also noted in [DI14]:

$$\sum_{(i,j)\in\delta^+(S)} x_{ij}^{st} \geq \sum_{(h,j)\in\delta^+(h)} x_{hj}^{st} \qquad h \in S \subseteq V \setminus \{s,t\}, |S| \geq 2. \tag{3.20}$$

The inequalities impose that the cutset induced by any subset S reached by a path must have cardinality greater than 1. The cutting planes can be generated either on incumbents, identifying strongly connected components in the subgraph induced by the variables x_{ij}^{st}, and on fractional solutions, solving a sequence of Min-Cut problems.

l-cycle elimination

It is also possible to explicitly add a subset of the subtour elimination constraints for subtours of bounded length. Specifically, it is sometimes convenient to add elimination constraints for 2-cycles (antisymmetric arcs),

$$x_{ij}^{st} + x_{ji}^{st} \leq 1 \qquad \forall\ i,j : (i,j),(j,i) \in A, (s,t) \in K, \tag{3.21}$$

and for 3-cycles (triangles),

$$x_{ij}^{st} + x_{jl}^{st} + x_{li}^{st} \leq 2 \qquad \forall\ i,j,l : (i,j),(j,l),(l,j) \in A, (s,t) \in K. \tag{3.22}$$

The number of 2-cycles in a graph is $O(|A|)$, exactly $\frac{|A|}{2}$ if each arc has an antisymmetric arc, and sometimes it appears convenient to insert them in the formulation. The number of 3-cycles is $O(|A|^2)$ or $O(|V|^3)$ – exactly $\frac{2n(n-1)(n-2)}{6}$ in a complete digraph – although on small-density graphs it is clearly much smaller. The number of additional constraints in UFP-MMF will be, respectively, of the order of $O(|A|k)$ and $O(|A|^2k)$.

3.4 Valid bounds and inequalities

We describe here valid bounds and inequalities that can be applied to tighten the arc formulation of UFP-MMF. As we will see in the section devoted to computational results, only some of them are effective in improving the performance of the branch-and-bound algorithm. Nevertheless, we believe all the explored attempts are worth being briefly discussed.

Flow upper and lower bounds

A valid upper bound on the flow values ϕ_{st} is given by the value of the maximum unsplittable flow between node s and t. This is also known as the *widest path* or *bottleneck shortest path* problem (see, e.g., [Sch03, Chapter 8.6] and [KP06]), as it is equivalent to finding the path whose smallest-capacity arc is the largest, that is:

$$\bar{\phi}_{st} = \max_{p \in P^{st}} \min_{(i,j) \in p} c_{ij}, \tag{3.23}$$

where P^{st} is the set of all s-t elementary paths. The problem can be solved in polynomial time with an adaptation of Dijikstra's algorithm for shortest paths, replacing the label update formula for a node i by the rule:

$$l[i] := \max_{i \in S} \{\min\{l[j], c_{ji}\}\}, \tag{3.24}$$

where S is the set of visited nodes so far. The bound can be used to restrict the domain of the ϕ_{st} variables, and also to strengthen some of the constraints. Specifically, for the activation constraints (3.12) we can use, in place of c_{ij}, the tighter upper bound $\min\{c_{ij}, \bar{\phi}_{st}\}$, obtaining the following inequality:

$$f_{ij}^{st} \leq \min\{c_{ij}, \bar{\phi}_{st}\} x_{ij}^{st} \qquad (i,j) \in A, (s,t) \in K. \tag{3.25}$$

These inequalities are similar to the so-called "strong inequalities" in network design [CCG09], where, however, the demand d_{st} for an (s,t) pair is fixed (which is clearly an extremely tight bound, as $\phi_{st} = d_{st}$ in any feasible solution). In our case, $\bar{\phi}_{st}$ is, in general, significantly weaker.

On the other hand, as we mentioned, there exists a simple valid lower bound on the value of the flow allocations, which is tight in the case where all the flows are routed over the same arc with minimum capacity:

$$\phi_{st} \geq \frac{c_{min}}{k} \qquad (s,t) \in K, \tag{3.26}$$

where $c_{min} := \min_{(i,j) \in A} c_{ij}$. If the instance has no $S-T$ bridges (see Section 2.6), the slightly stronger lower bound is valid:

$$\phi_{st} \geq \frac{c_{min}}{k-1} \qquad (s,t) \in K. \tag{3.27}$$

It is worth noting that these are, in general, the tightest upper and lower bounds for the optimal flow values, unless some further assumptions on the structure of the graph and the set K are made. The lower bound can also be seen as a cover inequality, $\sum_{(s,t) \in K} x_{ij}^{st} \leq k - 1$ for each arc $(i,j) \in A$.

Bottleneck consistency inequality

An arc can be bottleneck only if it is selected, thus the following inequality that links the y and x variables is valid:

$$y_{ij}^{st} \leq x_{ij}^{st} \qquad (i,j) \in A, (s,t) \in K. \tag{3.28}$$

Although there is a polynomial number of such inequalities, separating them appears to be the best option from a computational perspective.

Arc variables balance

In an optimal solution, the conservation of the value of the x variables is implied by their integrality and Constraints (3.10)–(3.13). However, it is useful to state explicitly the balance constraints on x:

$$\sum_{(i,j)\in\delta^+(i)} x_{ij}^{st} - \sum_{(j,i)\in\delta^-(i)} x_{ji}^{st} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{else} \end{cases} \qquad i \in V, (s,t) \in K, \tag{3.29}$$

in order to tighten the linear programming relaxation.

Flow constraints

Due to the (unsplittable) flow conservation, in an optimal solution the flow on an arc (i,j) for an origin-destination pair (s,t) is either 0 or ϕ_{st}. The following bilinear constraints are, therefore, valid:

$$f_{ij}^{st} = \phi_{st} x_{ij}^{st}. \tag{3.30}$$

The constraints can be linearized by McCormick's envelope [McC76] as follows:

$$f_{ij}^{st} \le \min\{\phi_{st}, \bar{\phi}_{st} x_{ij}^{st}\} \qquad (i,j) \in A, (s,t) \in K \tag{3.31}$$

$$f_{ij}^{st} \ge \max\{0, \phi_{st} - \bar{\phi}_{st}(1 - x_{ij}^{st})\} \qquad (i,j) \in A, (s,t) \in K \tag{3.32}$$

where $\bar{\phi}_{st}$ is a valid upper bound on ϕ_{st}, e.g., the one obtained computing the (s,t)-widest path. In (3.31), it is possible to replace $\bar{\phi}_{st}$ with $\min\{c_{ij},\ \bar{\phi}_{st}\}$, obtaining (3.25).

Nonlinear MMF flow expression

Let us denote with $\mathcal{U}(i,j) \subseteq K$ the set of origin-destination pairs that share the arc (i,j), and with $\mathcal{B}(i,j) \subseteq \mathcal{U}(i,j)$ the set of pairs for which the arc (i,j) is bottleneck. By the definition of MMF and bottleneck arc, the flow quantity f_{ij}^{st} for any $(s,t) \in \mathcal{U}(i,j)$ is bounded above by the value of the flows of the pairs in $\mathcal{B}(i,j)$. Such value is obtained by equally dividing among the pairs in $\mathcal{B}(i,j)$ the capacity of (i,j) decreased by the flows not in $\mathcal{B}(i,j)$, as follows:

$$(s,t) \in \mathcal{B}(i,j) \Rightarrow f_{ij}^{st} = \frac{c_{ij} - \sum_{(o,d)\notin\mathcal{B}(i,j)} f_{ij}^{od}}{|\mathcal{B}(i,j)|}. \tag{3.33}$$

that can be rewritten with the y variables as:

$$y_{ij}^{st} = 1 \Rightarrow f_{ij}^{st} = \frac{c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od}(1 - y_{ij}^{od})}{\sum_{(o,d)\in K} y_{ij}^{od}}. \tag{3.34}$$

This equation implies two valid inequalities. First of all, it provides an upper bound for all $(s,t) \in \mathcal{U}(i,j)$:

$$f_{ij}^{st} \le \frac{c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od}(1-y_{ij}^{od})}{\sum_{(o,d)\in K} y_{ij}^{od}}, \tag{3.35}$$

that can be rewritten as:

$$f_{ij}^{st} \sum_{(o,d)\in K} y_{ij}^{od} \le c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od} + \sum_{(o,d)\in K} f_{ij}^{od} y_{ij}^{od}. \tag{3.36}$$

In order to linearize the expression, let us introduce the continuous variable $f_{ij}^{stod} \ge 0$, defined as

$$f_{ij}^{stod} = y_{ij}^{st} f_{ij}^{od}. \tag{3.37}$$

We can then write:

$$\sum_{(o,d)\in K} f_{ij}^{stod} \le c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od} + \sum_{(o,d)\in K} f_{ij}^{odod}, \tag{3.38}$$

with the additional linearization constraints:

$$f_{ij}^{stod} \le \min\{f_{ij}^{od}, c_{ij} y_{ij}^{st}\} \qquad (i,j)\in A, (s,t)\in K, (o,d)\in K \tag{3.39}$$

$$f_{ij}^{stod} \ge \max\{0, f_{ij}^{od} - c_{ij}(1-y_{ij}^{st})\} \qquad (i,j)\in A, (s,t)\in K, (o,d)\in K. \tag{3.40}$$

Another valid inequality is the following, providing a lower bound for the pairs $(s,t) \in \mathcal{B}(i,j)$:

$$f_{ij}^{st} \ge y_{ij}^{st} \frac{c_{ij} - \sum_{(o,d)\in K} f_{ij}^{od}(1-y_{ij}^{od})}{\sum_{(o,d)\in K} y_{ij}^{od}}, \tag{3.41}$$

that can be rewritten as:

$$f_{ij}^{st} \sum_{(o,d)\in K} y_{ij}^{od} \ge y_{ij}^{st} c_{ij} - y_{ij}^{st} \sum_{(o,d)\in K} f_{ij}^{od} + y_{ij}^{st} \sum_{(o,d)\in K} f_{ij}^{od} y_{ij}^{od} - k c_{ij}(1-y_{ij}^{st}). \tag{3.42}$$

In order to linearize the expression, let us introduce the continuous variable $f_{ij}^{stodbe} \ge 0$, defined as

$$f_{ij}^{stodbe} = y_{ij}^{be} f_{ij}^{stod}$$

We can then write:

$$\sum_{(o,d)\in K} f_{ij}^{stod} \ge y_{ij}^{st} c_{ij} - \sum_{(o,d)\in K} f_{ij}^{stod} + \sum_{(o,d)\in K} f_{ij}^{ododst} - k c_{ij}(1-y_{ij}^{st}), \tag{3.43}$$

with the additional linearization constraints:

$$f_{ij}^{stodbe} \le \min\{f_{ij}^{stod}, c_{ij} y_{ij}^{be}\} \qquad (i,j)\in A, (s,t),(o,d),(b,e)\in K \tag{3.44}$$

$$f_{ij}^{stodbe} \ge \max\{0, f_{ij}^{stod} - c_{ij}(1-y_{ij}^{be})\} \qquad (i,j)\in A, (s,t),(o,d),(b,e)\in K \tag{3.45}$$

Both families of valid inequalities (3.38) and (3.43), alongside with the auxiliary variables and constraints necessary for the linearization, can be added in order to strengthen the formulation.

Slack inequalities

In a max-min fair solution, each bottleneck arc is saturated by the flows that share the arc. The following condition is thus valid for each arc that is bottleneck for at least one flow:

$$\sum_{(s,t)\in K} f_{ij}^{st} = c_{ij} \qquad \forall (i,j) \in A \qquad s.t. \sum_{(s,t)\in K} y_{ij}^{st} \geq 1. \tag{3.46}$$

Conversely, for the arcs which are not bottleneck for any origin-destination pair it holds:

$$\sum_{(s,t)\in K} f_{ij}^{st} < c_{ij} \qquad \forall (i,j) \in A \qquad s.t. \sum_{(s,t)\in K} y_{ij}^{st} = 0. \tag{3.47}$$

Let us introduce the nonnegative slack variables s_{ij} that represent the slack on an arc (i,j) between its capacity c_{ij} and the sum of the flow values over (i,j), i.e., the available capacity on (i,j). Equation (3.46) and (3.47) can be rewritten as follows:

$$\sum_{(s,t)\in K} f_{ij}^{st} + s_{ij} = c_{ij} \qquad \forall (i,j) \in A, \tag{3.48}$$

where the slack variable s_{ij} needs to be linked with the bottleneck variables. Specifically, it is necessary to impose that:

$$s_{ij} > 0 \implies y_{ij}^{st} = 0 \qquad \forall (i,j) \in A, (s,t) \in K, \tag{3.49}$$

which can be written equivalently as:

$$s_{ij} \sum_{(s,t)\in K} y_{ij}^{st} = 0 \qquad \forall (i,j) \in A. \tag{3.50}$$

Constraint (3.50) can also be disaggregated into the constraints:

$$s_{ij} y_{ij}^{st} = 0 \qquad \forall (i,j) \in A, (s,t) \in K. \tag{3.51}$$

Let us linearize the equality. We can express s_{ij} as:

$$s_{ij} = c_{ij} - \sum_{(s,t)\in K} f_{ij}^{st} \qquad \forall (i,j) \in A, \tag{3.52}$$

that can be substituted into (3.51), yielding the constraints:

$$y_{ij}^{st} c_{ij} - y_{ij}^{st} \sum_{(o,d)\in K} f_{ij}^{od} = 0 \qquad \forall (i,j) \in A. \tag{3.53}$$

This allows us to use carry out the linearization using the same variables f_{ij}^{stod} defined in (3.37).

Symmetry

The formulation for UFP-MMF is symmetric. If, in a solution, an (s,t) path has more than one arc fulfilling the bottleneck conditions, all the solutions with at least one of those bottleneck arcs with $y_{ij}^{st} = 1$ are equivalent. Breaking symmetry is generally useful in mixed-integer programming, see e.g. [Mar10]. In order to do so, it is necessary to establish a strict implication between the value of the y variables and the bottleneck conditions.

This is made difficult by the fact that the bottleneck conditions involve a comparison of fractional values. Specifically, we need to distinguish whether two flows have the same or different values. This means using a value $\varepsilon > 0$ and introducing the constraint:

$$f_{ij}^{st} \leq u_{ij} - \varepsilon(1 - y_{ij}^{st}), \tag{3.54}$$

imposing that the (s,t) flow is *strictly* smaller than the largest one on (i,j) if it is not bottleneck for (s,t). However, it remains to be seen if there exists such an ε that guarantees the exactness of the solution. It is necessary to find a bound on the difference between two possible flow values sharing the same arc. By the MMF conditions, the largest flow value in a saturated arc takes value

$$\frac{c_{ij} - \sum_{(o,d)\notin\mathcal{B}(i,j)} f_{ij}^{od}}{|\mathcal{B}(i,j)|},$$

where $\mathcal{B}(i,j)$ is the set of pairs for which (i,j) is bottleneck. Since we are dealing with unsplittable flows and every pair has a bottleneck, the k flow values in a feasible solution are univocally determined by the path selection. The number of possible path selections is finite, so is the set of feasible flow values. Then, in principle, by taking ε smaller than the difference of any two feasible flow values, the exactness of the formulation is guaranteed. However, obtaining a non-trivial smaller bound on ε is hard, and, in practice, we have observed optimal flow vectors where the smallest difference between two allocations is smaller $10^{-6}c_{min}$.

Other valid inequalities

To the best of our knowledge, the literature does not contain much work on *elastic* unsplittable multicommodity flow problems, that is, problems where the demands of the sink nodes are not given, but must be maximized. The demands are typically known, and a cost function is to be minimized: the presence of fixed demands allows to generate strong valid inequalities.

A class of lifted cover inequalities for unspittable multicommodity flow problems is proposed in [BJN$^+$98] and later extended in [Alv05]. They are based on the valid arc capacity inequality:

$$\sum_{(s,t)\in K} d_{st}x_{ij}^{st} \leq c_{ij}$$

where d_{st} is the demand for the pair $(s,t) \in K$, and can be replaced by any lower bound on the flow ϕ_{st}. However, in UFP-MMF we have only a valid lower bound for ϕ_{st} that is not strong enough to yield inequalities that are not trivially satisfied.

For some network design problems that share similarities with UFP-MMF, the authors of [CCG09] mention four classes of nontrivial valid inequalities: cover, minimum cardinality, flow cover and flow pack inequalities. All of them are based on the following valid cutset inequality:

$$\sum_{(i,j)\in\delta(S)} c_{ij} z_{ij} \geq d_{\delta(S)},$$

where z_{ij} is the binary variable that takes value 1 if the arc (i,j) is used by any flow and 0 otherwise, $\delta(S)$ is the cutset corresponding to the subset of nodes $S \subset V$, and $d_{\delta(S)}$ is a lower bound on the amount of flow that must circulate across the cutset in any feasible solution. Clearly, the right-hand side is easily computed when the demands d_{st} are given, but in our case we do not have good (and easy to compute) lower bounds on the flow allocations ϕ_{st}.

These cases indicate that the absence of given flow demands makes it difficult to generate strong knapsack or cutset-like valid inequalities. This was confirmed by preliminary experiments (whose details we omit) with the cover inequalities of [BJN$^+$98].

Attempts with reformulation-linearization technique

The set of MMF bottleneck constraints is another source of difficulty for MILP-based approaches, since it generally yields a linear relaxation solution where the bottleneck variables y_{ij}^{st} are highly fractional (see Remark 4 at page 33). To approximate the convex hull of the polyhedron defined by the MMF constraints (3.14),(3.15) and (3.17), we can apply Sherali-Adams' reformulation-linearization technique (RLT) for mixed-integer 0-1 programming problems [SA94]. The technique consists in first reformulating the problem by constructing redundant nonlinear constraints, obtained by multiplying the constraints by polynomial factors of the n binary variables and their complements. Then, the constraints of the reformulated problem are linearized by substituting a continuous variable for each nonlinear term. Depending on the degree of the polynomial factors used in the reformulation, different levels of RLT can be obtained: each level of the hierarchy provides a program whose continuous relaxation is at least as tight as the previous level, with the n-th level giving a convex hull representation.

In our attempt we consider a level-1 RLT formulation, applying the following two steps:
Reformulation: multiply each of the inequalities by each binary variable y_{ij}^{st} and its complement $(1 - y_{ij}^{st})$. Substitute $(y_{ij}^{st})^2 = y_{ij}^{st}$ throughout the constraints.
Linearization: linearize the resulting constraints by substituting, for every dis-

tinct nonlinear term, a continuous variable, and adding the appropriate linearization constraints.

Let us consider first Constraints (3.14), that we multiply by the variable y_{be}^{od} and its complement $1 - y_{be}^{od}$ for any $(o,d) \in K, (b,e) \in A$:

$$y_{be}^{od} \sum_{(i,j)\in A} y_{ij}^{st} \geq y_{be}^{od} \qquad (s,t),(o,d) \in K, (b,e) \in A \tag{3.55}$$

$$(1 - y_{be}^{od}) \sum_{(i,j)\in A} y_{ij}^{st} \geq (1 - y_{be}^{od}) \qquad (s,t),(o,d) \in K, (b,e) \in A. \tag{3.56}$$

Let us now consider (3.15) and (3.17). Both can be rewritten in the form:

$$y_{ij}^{st} \leq \square_{ij}^{st}$$

with

$$\square_{ij}^{st} := \begin{cases} \frac{\sum_{(o,d)\in K} f_{ij}^{od}}{c_{ij}} & \text{for (3.15)} \\ \frac{f_{ij}^{st} - u_{ij} + c_{ij}}{c_{ij}} & \text{for (3.17).} \end{cases}$$

After multiplying by y_{be}^{od} and $1 - y_{be}^{od}$, we obtain:

$$y_{be}^{od} y_{ij}^{st} \leq \square_{ij}^{st} y_{be}^{od} \qquad (s,t),(o,d) \in K, (b,e) \in A \tag{3.57}$$

$$(1 - y_{be}^{od}) y_{ij}^{st} \leq \square_{ij}^{st} (1 - y_{be}^{od}) \qquad (s,t),(o,d) \in K, (b,e) \in A \tag{3.58}$$

In order to linearize the resulting formulation, we replace all the terms $(y_{ij}^{st})^2$ with y_{ij}^{st}, and define the variables:

$$y_{beij}^{odst} := y_{be}^{od} y_{ij}^{st}$$

which can be linearized as:

$$y_{beij}^{odst} \leq \min\{y_{be}^{od}, y_{ij}^{st}\} \qquad (s,t),(o,d) \in K, (i,j),(b,e) \in A \tag{3.59}$$

$$y_{beij}^{odst} \geq \max\{0, y_{be}^{od} + y_{ij}^{st} - 1\} \qquad (s,t),(o,d) \in K, (i,j),(b,e) \in A \tag{3.60}$$

Observe that (3.57) and (3.58) contain bilinear terms also involving the continuous variables u_{ij} and f_{ij}^{st}, that have to be linearized in a similar way. The resulting system of inequalities provides a linear relaxation that is at least as strong as the original, but it involves a large (though polynomial) number of constraints. Eventually, we would like to exploit the RLT reformulation to identify valid inequalities that are helpful in tightening the formulation, and could possibly be separated in a cutting plane fashion. However, preliminary experiments were not promising, in the sense that even adding all level-1 RLT inequalities does not appear to strengthen the formulation noticeably, which, for the moment, discouraged us from further investigation in this direction.

3.5 Branch-and-price

Let us consider a formulation including variables representing all the elementary paths connecting the origin-destination pairs in the graph, instead of the arc variables $\underline{x}$. This allows us to impose the elementarity of the paths implicitly. Since the number of elementary paths in a graph connecting two nodes (s,t) may be exponential in the size of the graph, we describe a column generation algorithm to generate the path variables dynamically.

3.5.1 Path formulation

Let P^{st} be the set of all the elementary paths connecting the origin-destination pair $(s,t) \in K$. Let the parameter σ_{ij}^{pst} be 1 if the path $p \in P^{st}$ contains the arc (i,j) and 0 otherwise. Let the binary variable λ_p^{st} be 1 if the path of index p is selected for the pair (s,t) and 0 otherwise. For each pair $(s,t) \in K$, let $\phi^{st} \in \mathbb{R}_+$ be the flow allocated to it and let f_{ij}^{st} the amount of flow on the arc $(i,j) \in A$. Notice that f_{ij}^{st} is either ϕ^{st} or 0. Let u_{ij} be an upper bound on the flows over the arc $(i,j) \in A$. The binary variables y_{ij}^{st} indicate whether an arc (i,j) is a bottleneck for (s,t). We obtain the following MILP path-based formulation for UFP-MMF:

$$\max \sum_{(s,t)\in K} \phi^{st} \tag{3.61}$$

$$\text{s.t.} \quad \sum_{(i,j)\in\delta^+(i)} f_{ij}^{st} - \sum_{(j,i)\in\delta^-(i)} f_{ji}^{st} = \begin{cases} \phi^{st} & \text{if } i = s \\ -\phi^{st} & \text{if } i = t \\ 0 & \text{else} \end{cases} \qquad i \in V, (s,t) \in K \tag{3.62}$$

$$\sum_{(s,t)\in K} f_{ij}^{st} \le c_{ij} \qquad (i,j) \in A \tag{3.63}$$

$$\sum_{p\in P^{st}} \lambda_{st}^p = 1 \qquad (s,t) \in K \tag{3.64}$$

$$f_{ij}^{st} \le c_{ij} \sum_{p\in P^{st}} (\sigma_{ij}^{pst} \lambda_{st}^p) \qquad (i,j) \in A, (s,t) \in K \tag{3.65}$$

$$\sum_{(i,j)\in A} y_{ij}^{st} \ge 1 \qquad (s,t) \in K \tag{3.66}$$

$$\sum_{(o,d)\in K} f_{ij}^{od} \ge c_{ij} y_{ij}^{st} \qquad (i,j) \in A, (s,t) \in K \tag{3.67}$$

$$u_{ij} \ge f_{ij}^{st} \qquad (i,j) \in A, (s,t) \in K \tag{3.68}$$

$$u_{ij} \le f_{ij}^{st} + c_{ij}(1 - y_{ij}^{st}) \qquad (i,j) \in A, (s,t) \in K \tag{3.69}$$

$$\phi^{st}, f_{ij}^{st}, u_{ij} \ge 0 \qquad (i,j) \in A, (s,t) \in K \tag{3.70}$$

$$\lambda_{st}^p \in \{0,1\} \qquad (s,t) \in K, p \in P^{st} \tag{3.71}$$

$$y_{ij}^{st} \in \{0,1\} \qquad (i,j) \in A, (s,t) \in K. \tag{3.72}$$

Constraints (3.62)–(3.63) are standard flow conservation and capacity constraints. Constraints (3.64) guarantee that only one path is chosen for each $(s,t) \in K$, i.e., that the original arc variables x_{ij}^{st} are a convex combinations of the variables λ. Constraints (3.65) ensure that the flow f_{ij}^{st} on an arc is 0 if the selected path $p \in P^{st}$ does not contain the arc $(i,j) \in A$. Constraints (3.66)–(3.69) impose that the flow vector is MMF for the selected paths, according to Proposition 3.2. More specifically, Constraints (3.66) guarantee that at least an arc is bottleneck for each (s,t). Constraints (3.67) ensure that arc (i,j) is saturated if it is bottleneck for some pair (s,t). Constraints (3.68) make sure that u_{ij} is equivalent to the largest flow allocated over arc (i,j). Finally, Constraints (3.69) impose that the flow of a pair (s,t) through its bottleneck arc (i,j) is as large as the largest flow through (i,j) for all the other pairs.

3.5.2 Column generation

Formulation (3.61)–(3.72) has an exponential number of variables with respect to the size of the graph, but has a rather natural interpretation and a few practical advantages.

One advantage of a path formulation is that it is quite convenient to take into account restrictions on the structure of the routing paths when the path variables are generated, even when it is not easy to handle those constraints effectively in the arc formulation – for instance, the elementarity of the paths. In some cases, the set of feasible paths $\mathcal{P}^{st}$ for each (s,t) pair might be even entirely given a priori (e.g., by the network operator itself).

A second reason for choosing a path formulation is related to the the different solution approach and the additional insight that we gain. The path formulation can be viewed as a Dantzig-Wolfe decomposition [DW60] of the arc formulation we described in the previous section. The idea of a Dantzig-Wolfe decomposition is to reformulate the problem so to convexify a subset of the constraints. To do so, a set of variables is replaced by a convex combination of the extreme points of the polytope where they lie (possible due to Minkowski's theorem). The corresponding constraints, sometimes called *structural constraints*, are removed by the formulation. They will be used to generate the extreme points (each corresponding, in our case, to a different elementary s-t paths). What is left, i.e., the *linking constraints* (3.61)–(3.72) plus the original objective function, is usually referred to as the master problem.

The so-called *restricted* master problem is obtained by restricting each set P^{st} to a subset $\bar{P}^{st}$ of paths that have been generated. Then, it is possible to use a column generation approach, where new paths (equivalently, the λ_p^{st} variables), are dynamically generated via a pricing subproblem. The pricing subproblem has the aim of generating attractive paths considering their reduced cost, and the procedure stops when no more improving variables can be found. In essence, this can be seen as a generalization of the simplex method [DL05].

It is necessary to point out that this technique, as described up to now, can be used to solve only the LP relaxation of (3.61)–(3.72). The next step, then, is to combine the Dantzig-Wolfe decomposition with integer programming techniques aimed at finding integer solution. This means branching on the integer variables, and, possibly, generating valid cutting planes, while the relaxation in each node will be solved by column generation. This method is usually referred to as branch-and-price, or branch-and-price-and-cut. A similar approach has been proposed in the past for integer or unsplittable multicommodity flows, notably by Barnhart et al. [BJN+98], and later by Alvelos and Carvalho [AdC03, Alv05]. Note, however, that they solve a multicommodity flow problem where a total cost is to be minimized, and the demands are given.

3.5.3 Pricing

Let $\omega^{st} \in \mathbb{R}$ and $\pi_{ij}^{st} \geq 0$ be the dual variables associated to, respectively, Constraints (3.64) and (3.65). The reduced cost associated with a variable λ_p^{st} is $0 + \sum_{(i,j)\in A} \sigma_{ij}^{pst}(\pi_{ij}^{st} c_{ij}) - \omega^{st}$. In order to generate an improving column for a given $(s,t) \in K$, we need to generate an elementary path $p \in P^{st}$ whose associated λ_p^{st} variable has a positive reduced cost, i.e., such that

$$\sum_{(i,j)\in A} \sigma_{ij}^{pst}(\pi_{ij}^{st} c_{ij}) > \omega^{st}.$$

This amounts to finding, for each $(s,t) \in K$, a longest path $p \in P^{st}$ over the original graph G where the weight of each arc is given by $\pi_{ij}^{st} c_{ij} \geq 0$. The path p is described by the vector $\underline{\sigma}^{pst} \in \{0,1\}^{|A|}$, where $\sigma_{ij}^{pst} = 1$ if the arc $(i,j) \in A$ belongs to the path and equals 0 otherwise.

The pricing procedure requires the exact solution of the subproblem. Due to the possible presence of positive cycles, we must impose the elementarity of the path explicitly. This is consistent with the observation that subtours may arise in a solution of the UFP-MMF arc-based formulation, unless prevented.

An alternative that is often used similar problems, where the pricing problem is a shortest path, but branching constraints may give rise to negative cost cycles, is to introduce cycle variables — see, e.g., [Alv05]. This is possible when the structure of the problem guarantees that, in a optimal solution, those cycle variables will be null (or cycles can be safely removed from the solution). However, this does not hold true in our case; thus, we actually need to solve a longest elementary path problem.

The longest elementary path problem is known to be $\mathcal{NP}$-hard. Here, we cast it as a MILP which is based on the mathematical programming formulation of the elementary shortest path problem, with additional subtour elimination constraints. While similar pricing subproblems are often solved with ad-hoc algorithms, e.g., dynamic programming-based labeling algorithms, this is impractical in our case, since the absence of resource constraints make the search space very large. Moreover, using a MILP approach allows us to easily incorporate the branching information in the pricing subproblems.

Let $V_s := V \setminus \{s\}$ and $V_{st} := V \setminus \{s,t\}$, for a given $(s,t) \in K$. Let the variable $\sigma_{ij} \in \{0,1\}$ be 1 if the path that we are looking for contains arc (i,j) and 0 otherwise. The pricing subproblem can be cast as a MILP as follows:

$$\max \sum_{(i,j)\in A} (c_{ij}\pi_{ij})\sigma_{ij} \tag{3.73}$$

s.t.

$$\sum_{(i,j)\in\delta^+(i)} \sigma_{ij} - \sum_{(j,i)\in\delta^-(i)} \sigma_{ji} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{else} \end{cases} \qquad i \in V \tag{3.74}$$

$$\sum_{(i,j)\in\delta^+(i)} \sigma_{ij} \le 1 \qquad i \in V \tag{3.75}$$

$$\textit{Subtour elimination constraints} \tag{3.76}$$

$$\sigma_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{3.77}$$

Constraints (3.74) are standard flow balance constraints. Constraints (3.75) limit the outgoing degree to 1, hence guaranteeing that the flow is unsplittable. As indicated in (3.76), we also need to prevent subtours. In this regard, we can adopt the techniques that will be discussed in Chapter 5.

The column generation approach, at each node, requires the pricing subproblem to certify there are no improving columns. This means solving the problem exactly at least once. However, finding the variable with maximal reduced cost is not always necessary[2], thus, we can use a hierarchy of heuristics that try to find improving columns (or prove there are none). As an example, a first check consists of computing the sum of the outgoing arc with maximum weight for each vertex. Clearly, if it is smaller than ω^{st}, no improving column can exist. As a simple heuristic to find an improving column, we adopt a greedy longest path heuristic that selects the largest-cost outgoing arc, starting from s, until t is reached. We also check if the arc weights do not actually induce positive cycles. If this is the case, we can use a polynomial-time algorithm; otherwise, one eventually needs to solve the MILP problem. The pricing phase is summarized in Figure 3.3.

3.5.4 Branching

We have explained how we can use a Dantzig-Wolfe decomposition to obtain a lower bound of the optimal value of the original problem. If the solution we obtain with column generation does not satisfy the integrality constraints on the variables

[2]Indeed, the classic maximum reduced cost pivoting rule by Dantzig [Dan98] is often replaced, in modern simplex implementations, in favor of variants of the *steepest-edge* rule [FG92].

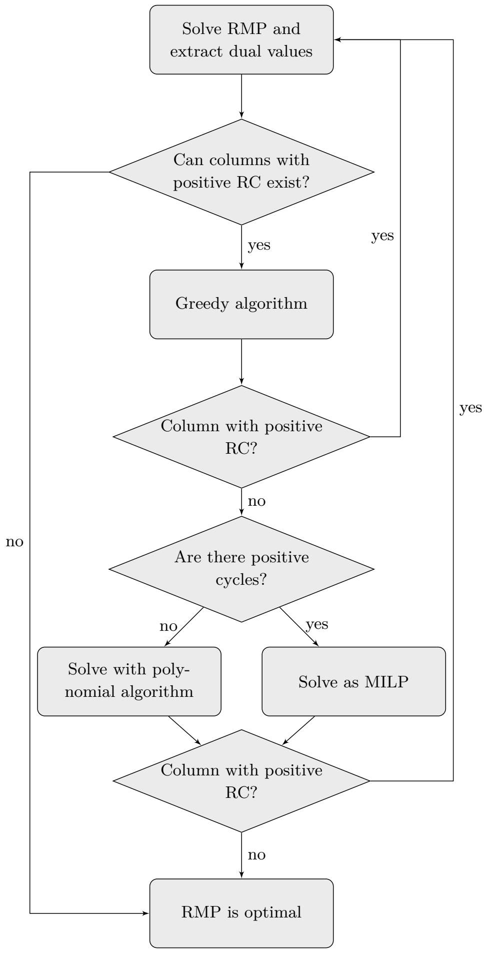

Solve RMP and extract dual values
Can columns with positive RC exist?
yes
Greedy algorithm
Column with positive RC?
yes
no
Are there positive cycles?
no
yes
Solve with polynomial algorithm
Solve as MILP
Column with positive RC?
yes
no
no
RMP is optimal

Figure 3.3: *Scheme of the pricing phase in the column generation algorithm.*

y_{ij}^{st} and λ_{st}^{p}, it is necessary to use a branch-and-bound scheme to reach an exact integer solution, in what is known as branch-and-price [BJN+98].

In particular, one needs to pay special attention when the branching involves the path selection. A possibility is branching on the variables in the master problem, i.e., the variables λ_{st}^{p} themselves, so that

$$\lambda_{st}^{p} \leq \lfloor \bar{\lambda}_{st}^{p} \rfloor = 0$$

and:

$$\lambda_{st}^{p} \geq \lceil \bar{\lambda}_{st}^{p} \rceil = 1,$$

where $\bar{\lambda}_{st}^{p}$ is the current (fractional) value of the path variable. This approach has some clear drawbacks, although it has been used in some early work on the branch-and-price. First, it generally yields a poorly balanced branch-and-bound tree, as the branching decision is highly asymmetrical: imposing the selection of a path is a very strong decision, while imposing a path is *not* taken is a very weak one. Second, in the case where a path is forbidden, we must take special care so that a variable that has been deleted is not generated again in a following node. This can be hard to do, and it might mean making the pricing subproblem significantly more difficult to solve.

A different approach arises from the observation that the integrality of the path variables is not really necessary. Indeed, for an arc variable x_{ij}^{st} to be integer, it suffices that the quantity $\sum_{p \in P^{st}} \lambda_p^{st} \sigma_{ij}^{pst}$ is integer. A natural idea is to branch on the original arc variables, whose value can be reconstructed using the coupling equation:

$$\bar{x}_{ij}^{st} = \sum_{p \in P^{st}} \bar{\lambda}_p^{st} \sigma_{ij}^{pst} \qquad (i,j) \in A,\ (s,t) \in K. \tag{3.78}$$

A valid branching rule consists of selecting an arc (i,j) and a pair (s,t) such that $\bar{x}_{ij}^{st}$ has fractional value. We create two subproblems, namely a node with the constraint:

$$\sum_{p \in P^{st}} \lambda_p^{st} \sigma_{ij}^{pst} \le \lfloor \bar{x}_{ij}^{st} \rfloor = 0 \tag{3.79}$$

and one with the constraint:

$$\sum_{p \in P^{st}} \lambda_p^{st} \sigma_{ij}^{pst} \ge \lceil \bar{x}_{ij}^{st} \rceil = 1. \tag{3.80}$$

A combination of the two approaches is also possible. As an example, Parker and Ryan [PR93] adopt a branching rule that generates $l+1$ children, where l is the length of the path where we are branching on. The idea is to enforce, on one branch, $\lambda_p^{st} = 1$, and, on each of the l other branches, $x_{p_i}^{st} = 0$, where p_i is the i-th arc contained in the path p.

Regarding the bottleneck variables y_{ij}^{st}, branching is straightforward. The branching decisions for variables y_{ij}^{st} can be left to the underlying solver, that can apply its highly-tuned branching rules, such as pseudocost branching, strong branching, reliability branching (see, e.g., [BGG+71] and [AKM05]).

A further option is to impose a different branching priority among the variables. In other words, branching decisions are imposed first only on one family of variables, until all of them have integer values. For UFP-MMF, one can branch first on the bottleneck variables y, and then on the arc variables x.

Note that, in a branch-and-price approach, the branching decision can be imposed in two different ways. One way is to add the branching constraints to the master problem, i.e., adding them to the *linking constraints*. This gives rise to additional dual variables that we have to keep into account in the pricing subproblem. In particular, the coefficients of the objective function will be augmented with the optimal dual values corresponding to the branching constraints. This approach has the advantage of leaving the structure of the pricing subproblem intact, except for the objective coefficients.

An alternative is to enforce the branching decisions in the pricing subproblems. The master problem is not augmented with additional constraints that might make it computationally heavier, and we do not need to consider other dual values. However, this means potentially modifying the structure of the subproblems. In some cases, imposing a branching decision is easy: think, for example, of the case where the pricing phase consists of solving knapsack problems. Then, both branching decisions (imposing an item is or is not selected) can be enforced easily. Viceversa, if the pricing phase involved a standard shortest path, imposing an arc is not in the path is trivial, while forcing its inclusion is not. This argument does not apply if the pricing subproblems are solved as generic MILPs: in this case, changing the lower and upper bounds of a variable is straightforward, and the resulting problem is, typically, not harder.

3.6 Heuristics

In preliminary experiments, we observed that finding feasible solutions for UFP-MMF with variants of a branch-and-bound algorithm is very hard, even for state-of-the-art MILP solvers such as CPLEX. The introduction of *ad hoc* primal heuristics is indeed crucial for the effectiveness of algorithms based on a branch-and-bound scheme. However, for the most challenging instances, standalone heuristics are sometimes the only option to find good feasible solutions within a reasonable time limit. In this section, we describe: simple rounding heuristics to be used within the branch-and-bound tree; a restricted path formulation, that can be used as a standalone heuristic or within the branch-and-price; a randomized greedy heuristic; and finally, a local search algorithm with variable neighborhood and tabu list.

3.6.1 Rounding heuristics

Shortest-path rounding

Starting from a feasible solution of the continuous relaxation, for each (s,t) we find a shortest path in G with weights $1 - x_{ij}^{st}$ for each arc $(i,j) \in A$. A variant is obtained by sampling the weight of each arc from a uniform distribution in $(0, 1 - x_{ij}^{st})$. In a branch-and-price algorithm, an interesting side product of this kind of heuristic is that, if the path that is found is not already in $\bar{P}^{st}$, the new column can be added to the pool.

Path-based rounding

This kind of heuristic can be used only within a branch-and-price. Starting from a feasible solution of the continuous relaxation, for each (s,t) we select the path $p \in \bar{P}^{st}$ with the largest λ_p^{st}. In an alternative, randomized variant, we pick a path $p \in \bar{P}^{st}$ with a probability equal to λ_p^{st}.

For both rounding heuristics, once a path p^{st} has been selected for each (s,t) pair, a complete feasible solution is constructed by running the water filling algorithm over the chosen paths.

3.6.2 Restricted path formulation

Let us consider a restriction of the path formulation, where for each (s,t), only a subset of the feasible elementary paths in the graph can be selected. Its optimal value is a lower bound with respect to the optimal value of the original problem, and the solutions will be feasible also for UFP-MMF.

If we are interested in using the restricted MILP as a standalone heuristic, the choice of the paths can be made according to a criterion that generates a set of attractive (and diverse) paths *a priori*.

An alternative option is to start the branch-and-price algorithm and stop the pricing at a certain depth of the search tree (*bounded depth*), or after a certain number of paths has been reached (*bounded cardinality*), and continue the branch-and-bound search until conclusion.

It is also possible to use this kind of heuristic within an exact branch-and-price: we can build and solve a restricted path subproblem, where we include only the columns generated so far, and impose the integrality of the λ_p^{st} variables. However, it must be noted that solving this MILP to optimality, even with a restricted set of variables, is challenging. In order to use this powerful but rather heavy heuristic, it is essential to adopt a short time limit and to run it with a low frequency.

3.6.3 Greedy multistart heuristic

Greedy algorithm

A deterministic greedy algorithm can be devised by considering K as an ordered list of (s,t) pairs. Then, in sequence, each of the flows is routed over the graph finding a path according to a given criterion. The water-filling algorithm is run at each iteration for all the paths selected so far, so as to provide a flow allocation which approximates the final MMF allocation with increasing accuracy and that can be used in the selection criterion.

Algorithm 3.1: Greedy algorithm.

```
   Data: $G = (V, A), K$
 1 $\bar{P} \leftarrow \varnothing$;
 2 for $(i, j) \in A$ do
 3     $w_{ij} \leftarrow \frac{1}{c_{ij}}$;
 4 end
 5 for $(s, t) \in K$ do
 6     $p^{st} \leftarrow$ shortest_path($G, s, t, \underline{w}$);
 7     $\bar{P} \leftarrow \bar{P} \cup p^{st}$;
 8     $\underline{\phi} \leftarrow$ water-filling($G, \bar{P}$);
 9     for $(i, j) \in A$ do
10         update($w_{ij}$);
11     end
12 end
```

The general scheme can be modified with respect to the order in which the OD pairs are considered, and what criterion is used to select the routing path for a given OD pair. Regarding the *a priori* order in which the commodities are considered, we have observed no criterion that consistently performs better, on average, than a random order. Then, we can exploit this idea to randomize the algorithm, as will be seen in the next subsection.

For a given (s, t), in order to select a routing path we find a shortest path over the graph G where each arc has a weight that is inversely proportional to its attractiveness. We have experimented with the following cost functions in the `update` step (line 10):

$$w_{ij} \leftarrow \frac{1}{c_{ij} - \gamma \sum_{(s,t) \in \mathcal{U}(i,j)} \phi^{st} + \epsilon} \qquad w_{ij} \leftarrow \frac{1 + |\mathcal{U}(i,j)|}{c_{ij} - \sum_{(s,t) \in \mathcal{U}(i,j)} \phi^{st} + \epsilon}$$

where ϕ^{st} is the MMF flow allocation computed over the partial solution built so far, $\mathcal{U}(i, j)$ represents the subset of paths using the arc (i, j), and γ is a value in the interval $(0, 1]$. The first cost function penalizes congested arcs, considering their residual capacity $\tilde{c}_{ij} = c_{ij} - \sum_{(s,t) \in \mathcal{U}(i,j)} \phi^{st}$ with a weight γ multiplying the value of the flows using that arc. Randomization can be introduced by considering the value γ to be sampled from a random distribution for each arc (i, j). The second cost function is an attempt to further penalize overlapping paths. while the third one introduces some randomization. The term ϵ is a small positive number (e.g., 0.001) in order to avoid division by 0 when the capacity of the arc is saturated.

An alternative approach to select the paths entails using the widest-path algorithm over the residual graph, instead of the shortest-path with the arc costs w_{ij}, that is, replacing line 6 with:

$$p^{st} \leftarrow \texttt{widest\_path}(G, s, t, \underline{\tilde{c}}).$$

This approach is quite effective in finding good paths when few OD pairs have been routed in the graph. However, the results are quite poor when the number of routed pairs grows, and the residual graph has many arcs with capacity equal to 0. Then, a good practical approach is to use a hybrid variant of Algorithm 3.1 where the widest-path algorithm is used as long as we find a widest path of nonzero value, and then the shortest-path update is adopted for the following iterations.

Multistart algorithm

Given an order of the OD pairs, the greedy algorithm is deterministic. It can be easily randomized considering sequences with a different ordering of the origin-destination pairs to be routed.

In the following simple multistart heuristic, the integer *MaxIt* represents the number of restarts. At each iteration, the sequence of OD pairs is shuffled randomly, and Algorithm 3.1 is run; after *MaxIt* attempts, the heuristic returns the best solution found. Let us denote by $\tau(sol)$ the objective value of the solution *sol*.

Algorithm 3.2: Multistart algorithm with randomized order of selection.

```
Data: $G = (V, A), K, MaxIt$
$best\_sol \leftarrow$ null;
for $t = 1, \ldots, MaxIt$ do
    $K \leftarrow$ shuffle($K$);
    $sol \leftarrow$ greedy($G, K$);
    if $\tau(sol) > \tau(best\_sol)$ then
        $best\_sol \leftarrow sol$;
    end
end
return $best\_sol$;
```

Refining a good solution

The multistart approach does not exploit previous good solutions to build better ones. We can randomize only *partially*, introducing the parameters:

- *Nreuse*: number of times a good solution is exploited;
- *Reuse_percentage*: percentage of the paths that are maintained.

When the current solution is better than the best found so far, we begin an intensification phase, where the next *Nreuse* restarts are close to that solution. To do so, we fix the first *Reuse_percentage* of the paths in the best solution, and we complete the solution with the randomized greedy approach.

Algorithm 3.3: Multistart algorithm with intensification phase.

```
Data: $G = (V, A), K, \textsc{MaxIt}$
$best\_sol \leftarrow$ null;
$reuse \leftarrow 0$;
for $t = 1, \dots, \textsc{MaxIt}$ do
    if $reuse > 0$ then
        $K \leftarrow$ partial_shuffle($K, \textsc{Reuse\_percentage}$);
    else
        $K \leftarrow$ shuffle($K$);
    end
    $sol \leftarrow$ greedy($G, K$);
    if $\tau(sol) > \tau(best\_sol)$ then
        $best\_sol \leftarrow sol$;
        $reuse \leftarrow \textsc{Nreuse}$;
    else
        if $reuse > 0$ then
            $reuse \leftarrow reuse - 1$;
        end
    end
end
return $best\_sol$;
```

The function `partial_shuffle` only randomizes the last $1 - \textsc{Reuse\_percentage}$ elements, while the first $\textsc{Reuse\_percentage}$ of them are fixed. If after $\textsc{Nreuse}$ times a new best solution has not been found, the variable *reuse* will reach 0 and a new solution will be built from scratch (complete restart), shuffling the whole list K, to guarantee diversification.

3.6.4 Local search with variable neighborhood and tabu list

The multistart algorithm described in the previous paragraphs quickly generates reasonably good solutions, as we will show in the computational experiments. A way to improve upon those solutions is to explore their neighborhoods moving towards a local optimum.

There are several ways to define a neighborhood of a UFP-MMF solution. The most natural approach consists in considering a solution as a collection of s-t paths, one for each OD pair[3]. Let us define the neighborhood $N_m(sol)$ of a solution as all the UFP-MMF feasible solutions where at most m paths differ from those of sol. Then, the cardinality of $N_m(sol)$ is $O\left(\sum_{i=1}^{m} \hat{P}^i \binom{k}{i}\right)$, where $k := |K|$ and $\hat{P}$ is an upper bound on the number of distinct routing paths for any OD pair. It is clear that even $N_1(sol)$ can be very large: there are k subsets of K with cardinality 1, and for each of them there are a number of alternative paths which is exponential with

[3]Note that a UFP-MMF solution is univocally determined by the selected routing paths, since the corresponding MMF flow allocation is unique.

the size of the arc-set. Clearly we can explore only a subset of a neighborhood. We would like to consider, instead of all possible paths for each $(s, t) \in K$, just a subset of attractive paths, selected based on an appropriate criterion: the question is how to choose alternative paths that would improve the solution. We could use a MIP formulation to find the paths that give the largest improvement in the objective function. However, preliminary experiments have showed that this is not a viable approach, since even redirecting 2 paths is far too computationally demanding to be done iteratively.

An alternative is to adopt a criterion which attempts to approximate such a locally optimal move. It appears essential that the selection of the new routing paths is done as efficiently as possible, even if we are sacrificing the quality of the move[4]. Given a subset $\hat{K} \subset K$ of OD pairs, with $|\hat{K}| = m$, we generate an alternative path for each pair $(s, t) \in \hat{K}$ solving a shortest-path problem where the weights of the arcs are determined so as to promote *i*) diversity with respect to the current path, and *ii*) a potential improvement in the objective function. It is easy to obtain *i*), while obtaining *ii*) is not trivial. A criterion that appears to work well in producing good paths is the one that we adopted in the greedy approach of Algorithm 3.1, i.e., using the inverse of the residual capacity on each arc, with a randomization factor.

In Algorithm 3.4 we report a high-level description of our local search method, where we move between neighboring solutions as long as we are able to improve the objective function, in the spirit of hill climbing techniques. The algorithm first looks for improving solutions by sampling the neighborhood $N_m(sol)$. The function `SP_Neigh`(sol, m, s) samples randomly a subset of size s from the neighborhood $N_m(sol)$, selecting the alternative paths according to the above-mentioned shortest-path criterion. If no improving solution is found, we consider larger neighborhoods, increasing the parameter m with a unitary step-size. We also simultaneously increase the number s of solutions sampled from the neighborhood, in an attempt to explore more thoroughly this larger neighborhood. The function `increment`(s, m) is responsible for increasing the size of the current neighborhood when no improving solution is found. We also adopt some tabu search features: when we have reached the maximum allowed size of the neighborhood, we accept also non-improving moves, and consider the best solution among those we have explored in the current neighborhood. To avoid cycling to the previous solutions, we keep a tabu list where we store the last 2 moves. A move is simply encoded as the list of (s, t) pairs whose path had been modified. Such short tabu list is sufficient, since longer cycles appear to be very unlikely in practice. The number of non-improving moves in a search is bounded by the parameter *MaxNonimprov*.

[4]Note, however, that a move which does not give the (locally) largest improvement is not necessarily worse in the context of a local search.

Algorithm 3.4: Local search with variable neighborhood and tabu list.

```
Data: $G = (V, A), K, initSol, s_0, m_0, S, M$
$curSol, s, m \leftarrow initSol, s_0, m_0$;
while $continue$ do
    $continue \leftarrow$ false;
    $localBest \leftarrow$ null;
    foreach $sol \in$ SP_Neigh$(curSol, m, s)$ do
        if $\tau(sol) > \tau(curSol)$ then
            $curSol \leftarrow sol$;
            $tabu.push(curSol)$;
            $continue \leftarrow$ true;
            break;
        end
        if $sol \notin tabu$ and $\tau(sol) > \tau(localBest)$ then
            $localBest \leftarrow sol$;
        end
    end
    if $continue =$ false then
        if $m < M$ then
            increment$(m, s)$;
            $continue \leftarrow$ true;
        else if $i <$ MaxNonimprov then
            $curSol \leftarrow localBest$;
            $tabu.push(curSol)$;
            $i \leftarrow i + 1$;
            $continue \leftarrow$ true;
        end
    end
end
return $curSol$;
```

Finally, to provides further diversification, the local search algorithm is embedded in the multistart scheme of Algorithm 3.2. At each restart, we first find a first solution via the greedy heuristic, then we use the local search with variable neighborhood and tabu list to reach a locally optimal solution.

CHAPTER 4

Relaxations and variants

Within an exact MIP-based method, or simply when evaluating a feasible solution for UFP-MMF, it is crucial to obtain valid upper bounds that give a guarantee on its quality. This can be done solving a relaxation of the original problem. The solutions obtained from a relaxation can also be used to build feasible UFP-MMF solutions heuristically.

A standard relaxation of a mixed-integer programming problem is the linear programming one, where the integrality requirement is dropped. UFP-MMF has a number of natural relaxations that are obtained by relaxing the max-min fairness criterion. These relaxations have various degrees of difficulty and effectiveness in providing good bounds and feasible solutions. In Sections 4.1–4.3 we describe three of them. In Section 4.4 we also discuss two alternative definitions of fairness which do not give a relaxation of UFP-MMF, but are of interest on their own.

4.1 Maximum-throughput unsplittable flow problem (UFP)

An obvious relaxation of UFP-MMF is obtained by discarding the max-min fairness constraints, and considering the single-level problem where the leader has control of both paths and flow allocation to maximize the total throughput. This is a maximum unsplittable multicommodity flow problem (UFP), which is $\mathcal{NP}$-hard but relatively easier than UFP-MMF, not only because of the lack of fairness constraints, but also because there is no need to explicitly prevent subtours (that cannot arise in optimal solutions). The UFP solution is typically not feasible for UFP-MMF. However, from an UFP optimal solution, a UFP-MMF feasible solution can be obtained recomputing the flow allocations over the selected paths according to the max-min fairness criterion.

4.2 Unsplittable flows subject to max-bottleneck fairness (UFP-MB)

Instead of MMF, suppose that at the second level of our problem the flows are allocated so that only the value of the smallest allocated flow is maximized. This is equivalent to relaxing the MMF constraints to consider only the first element of the sorted vector of allocation. In other words, we maximize the smallest allocation on the first bottleneck arc found by the water filling over the paths chosen by the leader. We denote this criterion by Max-Bottleneck fairness (MB).

Since, for any path selection, the set of feasible flow vectors for MMF is a subset of the ones which are feasible for MB, and the overall objective function is the same, UFP-MB is a relaxation of UFP-MMF and, hence, the optimal objective function value of the former is an upper bound on that of the latter.

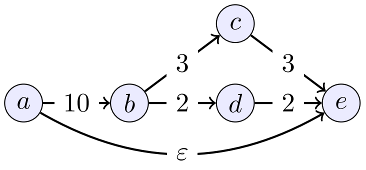
The follower problem of maximizing the smallest flow allocation is essentially a variant of the max-concurrent flow problem. Over fixed paths, it is easy to verify that the maximum smallest flow allocation can be obtained applying the first step of the water-filling algorithm. Note that, once the smallest allocation is maximized, the remaining allocations are free, and we assume an optimistic setting, where they will be allocated in a max-throughput fashion.

The following simple example shows that, in general, the optimal solution values of the unconstrained throughput maximization problem (UFP), that of UFP-MMF, and that of UFP-MB differ.

Example 4.1. Consider the same graph used in Example 2.2 and 2.3, that we report in Figure 4.1 for convenience.

We have discusses in Example 2.2 that the bilevel UFP-MMF optimal vector is $\underline{\phi} = (1, 3, 3, 1, 1, \varepsilon)$, with a total throughput of $\tau = 9 + \varepsilon$, obtained routing (s_6, t_6) over the arc (a, e), and (s_1, t_1) through d. We have also seen already that the optimal value of UFP can be obtained by allocating a flow $\phi_1 = 0$ to the pair (s_1, t_1), and by routing a flow $\phi_6 = \varepsilon$ over the arc (a, e), with the resulting flow allocation vector $\underline{\phi} = (0, 3, 3, 2, 2, \varepsilon)$ and a total throughput $\tau = 10 + \varepsilon$.

If the flow allocation is subject to Max-Bottleneck fairness, once the smallest allocation has been maximized and allocated to all the origin-destination pairs, the problem on the residual graph becomes a standard maximum unsplittable flow problem with no fairness constraints. It is easy to check that the maximum throughput under MB is obtained routing again ϕ_6 over the arc (a, e), with value ε, and guaranteeing a flow value of at least ε to all other pairs. Let us consider the OD pair (s_1, t_1). Since both paths where the flow ϕ_1 can be routed contain arcs which are used by other pairs, no more than ε should be allocated to it. If ϕ_1 is routed over the path through node d, the resulting allocation is $\underline{\phi} = (\varepsilon, 3, 3, 2-\varepsilon, 2-\varepsilon, \varepsilon)$, with $\tau = 10$. If that through node c is used, we obtain $\underline{\phi} = (\varepsilon, 3-\varepsilon, 3-\varepsilon, 2, 2, \varepsilon)$, with the same $\tau = 10$.



$(s_1 = b, t_1 = e)$
$(s_2 = b, t_2 = c)$
$(s_3 = c, t_3 = e)$
$(s_4 = b, t_4 = d)$
$(s_5 = d, t_5 = e)$
$(s_6 = a, t_6 = e)$

Figure 4.1: *Graph with six origin-destination pairs used in Example 4.1.*

We now derive a characterization of the optimal solution for the Max-Bottleneck version, similar to that of Proposition 3.2.

Proposition 4.2. *Given a directed graph $G = (V, A)$, a set K of origin-destination pairs and a simple path for each $(s, t) \in K$, a feasible flow allocation vector $\underline{\phi}$ is optimal for the problem of maximizing the minimum flow allocated to any pair if and only if there is at least an arc $(i, j) \in A$ (referred to as* global bottleneck*), satisfying the following properties:*

1. *the arc capacity is saturated,*
2. *the arc capacity is equally divided among all the origin-destination pairs that share the arc,*
3. *the flow allocated to the pairs that share the arc is the smallest among the flow values allocated to the pairs in K.*

Proof. Suppose that arc (i, j) is a global bottleneck and let $\eta := \min_{i=1,\dots,k}\{\phi_i\}$. Due to 3), all the flows sharing the global bottleneck have a value of η. Since, due to 1) and also to 2), the capacity c_{ij} is equally divided among the pairs, it is not possible to improve one of the allocations of value η without decreasing another one (thus, decreasing η). Since η is independent of the flow value for pairs not using (i, j), it follows that the solution is optimal.

Conversely, suppose that $\underline{\phi}$ is optimal. Consider again the smallest flow allocation η in $\underline{\phi}$. Assume that there is no global bottleneck. Then, for all (s, t) pairs with flow value η, either all the arcs in their path are nonsaturated, or the capacity is not equally shared. In both cases, the flow can be increased (in the latter case, by decreasing a larger flow). □

To obtain a formulation for UFP-MB, it suffices to remove the y_{ij}^{st} variables while introducing the binary variables b_{ij}, each of which equals 1 if the corresponding arc (i,j) is a global bottleneck. Then, we can replace the MMF constraints (3.14)–(3.17) and the variables y_{ij}^{st} with the following ones:

$$\sum_{(i,j)\in A} b_{ij} \geq 1 \tag{4.1}$$

$$\sum_{(s,t)\in K} f_{ij}^{st} \geq c_{ij} b_{ij} \qquad (i,j)\in A \tag{4.2}$$

$$\eta \leq \phi_{st} \qquad (s,t)\in K \tag{4.3}$$

$$\eta \geq f_{ij}^{st} - c_{ij}(2 - b_{ij} - x_{ij}^{st}) \qquad (i,j)\in A, (s,t)\in K \tag{4.4}$$

$$\eta \geq 0,\ b_{ij}\in\{0,1\} \qquad (i,j)\in A. \tag{4.5}$$

Constraint (4.1) ensures that there is at least one global bottleneck. Constraints (4.2) impose that the global bottleneck arc is saturated. Constraints (4.3)–(4.4) make sure that all pairs sharing the global bottleneck assume the smallest allocation.

An additional valid inequality for UFP-MB, which we will adopt to tighten the formulation in the computational experiments, is:

$$b_{ij} \leq \sum_{(s,t)\in K} x_{ij}^{st}. \tag{4.6}$$

Note that subtours must be prevented also for UFP-MB, and we can adopt the same branch-and-cut algorithm we have devised for UFP-MMF. In particular, we generate the GCS inequalities to eliminate subtours and strengthen the formulation. The majority of the valid bounds and inequalities for UFP-MMF can also be adapted easily to UFP-MB. The rounding heuristics can be adapted as follows. Once a path has been selected for each (s,t) pair, a simple 2-stage algorithm has to be applied: it consists of solving a bottleneck-maximization problem over fixed paths (carrying out the first step of the water-filling algorithm) followed by an LP that maximizes the total throughput over the residual graph.

4.2.1 Dual-based alternative formulation

An alternative to Constraints (4.1)–(4.4) can be obtained by considering an LP formulation for the problem of maximizing the smallest flow allocation, in the spirit of the max-concurrent flow problem. When a path p^{st} is given for all $(s,t)\in K$, the lower level problem can be written as the following LP:

$$\max \quad \eta \tag{4.7}$$
$$s.t. \quad \eta \leq \phi_{st} \qquad \forall (s,t) \in K \tag{4.8}$$
$$\sum_{(s,t)\in K:(i,j)\in p^{st}} \phi_{st} \leq c_{ij} \qquad \forall (i,j) \in A \tag{4.9}$$
$$\phi_{st} \geq 0 \qquad \forall (s,t) \in K \tag{4.10}$$
$$\eta \geq 0. \tag{4.11}$$

Its LP dual reads:

$$\min \quad \sum_{(i,j)\in A} c_{ij}\beta_{ij} \tag{4.12}$$
$$s.t. \quad \sum_{(s,t)\in K} \alpha_{st} \geq 1 \tag{4.13}$$
$$\sum_{(i,j)\in p^{st}} \beta_{ij} \geq \alpha_{st} \qquad \forall (s,t) \in K \tag{4.14}$$
$$\alpha_{st} \geq 0 \qquad \forall (s,t) \in K \tag{4.15}$$
$$\beta_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{4.16}$$

Proposition 4.3. *The original constraints*

$$\eta \leq \phi_{st} \quad \forall (s,t) \in K \tag{4.17}$$

which link first and second level problems can be replaced with

$$\sum_{(i,j)\in A} c_{ij}\beta_{ij} \leq \phi_{st} \quad \forall (s,t) \in K \tag{4.18}$$

subject to (4.13)–(4.16).

Proof. By weak LP duality, any solution to (4.13)–(4.16) yields an upper bound of $\sum_{ij\in A} c_{ij}\beta_{ij}$ to any feasible solution to (4.8)–(4.11). Hence, in any feasible solution Constraint (4.18) is at least as tight as Constraint (4.17). □

When the paths are not given, Constraint (4.14) reads:

$$\sum_{(i,j)\in A} x^{st}_{ij}\beta_{ij} \geq \alpha_{st} \quad \forall (s,t) \in K \tag{4.19}$$

We introduce a new variable $\gamma^{st}_{ij} := x^{st}_{ij}\beta_{ij}$. Due to the direction of the constraint, of the four constraints arising from the (exact) linearization *à la* McCormick of the bilinear product between a continuous and a binary variable, only two are needed, namely $\gamma^{st}_{ij} \leq x^{st}_{ij}\bar{\beta}_{ij}$ and $\gamma^{st}_{ij} \leq \beta_{ij}$, where $\bar{\beta}_{ij}$ is an upper bound on β_{ij}. We derive one as follows. Due to strong duality, we can assume

$\sum_{(i,j)\in A} c_{ij}\beta_{ij} = \eta \leq \phi_{st} \leq \bar{\phi}^{st}$ for all $(s,t) \in K$, from which we have: $c_{ij}\beta_{ij} \leq \bar{\phi}^{st}$ for all $(s,t) \in K$, where $\bar{\phi}^{st}$ is an upper bound on ϕ_{st}. Then we obtain:

$$\bar{\beta}_{ij} := \frac{\min_{(s,t)\in K}\{\bar{\phi}^{st}\}}{c_{ij}}.$$

Then, the max-bottleneck fairness can be imposed adding to the UFP formulation the following set of linear inequalities:

$$\sum_{ij\in A} c_{ij}\beta_{ij} \leq \phi_{st} \qquad \forall (s,t) \in K \tag{4.20}$$

$$\sum_{(s,t)\in K} \alpha_{st} \geq 1 \tag{4.21}$$

$$\sum_{(i,j)\in A} \gamma_{ij}^{st} \geq \alpha_{st} \qquad \forall (s,t) \in K \tag{4.22}$$

$$\gamma_{ij}^{st} \leq \bar{\beta}_{ij} x_{ij}^{st} \qquad \forall (s,t) \in K, (i,j) \in A \tag{4.23}$$

$$\gamma_{ij}^{st} \leq \beta_{ij} \qquad \forall (s,t) \in K, (i,j) \in A \tag{4.24}$$

$$\alpha_{st} \geq 0 \qquad \forall (s,t) \in K \tag{4.25}$$

$$\beta_{ij} \geq 0 \qquad \forall (i,j) \in A, \tag{4.26}$$

which are also valid inequalities for UFP-MMF.

4.3 Relaxed max-min fairness (r-MMF)

It is also possible to relax the Max-Min Fairness constraint in a different way. Instead of limiting the *depth* of the lexicographical optimization to the first element, as in MB, we can borrow an idea from [KRT99] and relax the bottleneck constraint by a factor $r \in (0,1]$. Then, in an r-bottleneck, a flow f_{ij}^{st} must be at least a factor r of the maximum flow (u_{ij}) allocated on the arc (i,j). In other words, this means imposing a softer notion of fairness, which tolerates "injustice" up to a factor r. To assume r-MMF, it is sufficient to replace the bottleneck Constraint (3.17) with the following inequality:

$$f_{ij}^{st} \geq r u_{ij} + c_{ij}(y_{ij}^{st} - 1) \qquad (i,j) \in A, (s,t) \in K, \tag{4.27}$$

where the relaxed equilibrium imposes that an allocation is r-MMF if it is not possible to increase the allocation of (s,t) unless we decrease the flow of an OD pair which is worse off by a factor r.

4.4 Alternative definitions of fairness

Let us consider UFP-MMF as a variant of a more general bilevel unsplittable flow problem subject to fair flow allocation (UFP-*fair*). Massoulié et al. showed in [MR02] that different congestion avoidance mechanisms in IP networks may produce different types of fairness. Then, it is possible to modify the lower level of the bilevel formulation to account for different fairness criteria that are implemented by the network protocols. Note that, in the two variants we will describe, the resulting problems are not a relaxation of UFP-MMF. Moreover, they give rise to mixed-integer nonlinear problems. We do not include experimental results in this thesis, although research on these variants is already underway.

Unsplittable flows subject to proportional fairness (UFP-PF)

A widely studied of fairness is proportional fairness [KMT98], that is known to well approximate FIFO fair queuing policies [MR02].

Definition 4.4. (Proportional Fairness). A flow allocation $\underline{\phi}$ is *proportionally fair* if and only if, for any other feasible allocation $\underline{\phi}'$, we have:

$$\sum_{(s,t)\in K} \frac{\phi'_{st} - \phi_{st}}{\phi_{st}} \leq 0$$

In other words, any change in the allocation must have a negative average relative change. If the set of feasible allocation vectors is convex, the proportionally fair allocation is unique and the following proposition is easily verified.

Proposition 4.5. *The proportionally fair allocation is the one maximizing the utility function*

$$J(\underline{\phi}) = \sum_{(s,t)\in K} \ln(\phi_{st})$$

over the set of feasible flow allocations.

Then, one can build the bilevel problem of finding the maximum throughput unsplittable flows subject to proportional fairness (UFP-PF) replacing the lower-level MMF problem with the maximization of the sum of the logarithms:

$$\underline{\phi} \in \operatorname{argmax} \left\{ \sum_{(s,t)\in K} \ln(\phi_{st}) \right\}. \tag{4.28}$$

The lower-level problem depends on the path selection of the upper level. It will be:

$$\max \quad \sum_{(s,t)\in K} \ln(\phi_{st}) \tag{4.29}$$

$$s.t. \quad \sum_{(s,t)\in K:(i,j)\in p^{st}} \phi_{st} \leq c_{ij} \qquad \forall (i,j)\in A \tag{4.30}$$

$$\phi_{st} \geq 0 \qquad \forall (s,t)\in K, \tag{4.31}$$

where p^{st} is the routing path for pair (s,t). Let $\pi_{ij} \geq 0$ be the dual variables corresponding to Constraints (4.30). Then, the KKT conditions read:

$$\left(\sum_{(s,t)\in K:(i,j)\in p^{st}} \phi_{st} - c_{ij} \right) \pi_{ij} = 0 \qquad \forall (i,j) \in A \tag{4.32}$$

$$\sum_{(s,t)\in K:(i,j)\in p^{st}} \phi_{st} \leq c_{ij} \qquad \forall (i,j) \in A \tag{4.33}$$

$$\frac{1}{\phi_{st}} = \sum_{(i,j)\in p^{st}} \pi_{ij} \qquad \forall (s,t) \in K \tag{4.34}$$

$$\pi_{ij} \geq 0 \qquad \forall (i,j) \in A. \tag{4.35}$$

The primal feasibility condition (4.33) is already implied by the constraints in the upper-level problem. Equations (4.32) and (4.34), when the paths are not fixed, become:

$$\left(\sum_{(s,t)\in K} \phi_{st} x_{ij}^{st} - c_{ij} \right) \pi_{ij} = 0 \qquad \forall (i,j) \in A \tag{4.36}$$

$$\frac{1}{\phi_{st}} = \sum_{(i,j)\in A} \pi_{ij} x_{ij}^{st} \qquad \forall (s,t) \in K, \tag{4.37}$$

Equation (4.36) can be rewritten as $\left(\sum_{(s,t)\in K} f_{ij}^{st} - c_{ij} \right) \pi_{ij} = 0$, where f_{ij}^{st} is the variable defined in the upper level as the s-t flow through the arc (i,j), and can be linearized introducing a binary variable that is 0 if $\pi_{ij} = 0$, and 1 otherwise. On the other hand, Equation (4.37) cannot be linearized in an exact way, since it is non homogeneous and it involves a product of continuous variables, so that the problem can be solved as a (nonconvex) mixed-integer nonlinear program with a spatial branch-and-bound [BLL$^+$09]. An alternative is to solve an approximation, either linearizing the KKT conditions or linearizing the objective function (4.29) and, then, using strong duality to impose its optimality conditions.

Unsplittable flows subject to minimum potential-delay fairness (UFP-MDPF)

Another definition of fairness cited in [MR02] is the minimum potential-delay fairness, where the fair solution is defined as the one maximizing the utility function:

$$J(\underline{\phi}) = \sum_{(s,t)\in K} \left(-\frac{1}{\phi_{st}} \right).$$

Similarly to the case of proportional fairness, the KKT conditions can be derived, yielding the following equations, when the paths are not given:

$$\left(\sum_{(s,t)\in K} f_{ij}^{st} - c_{ij}\right)\pi_{ij} = 0 \qquad \forall (i,j) \in A \tag{4.38}$$

$$\frac{1}{\phi_{st}^2} = \sum_{(i,j)\in A} \pi_{ij} x_{ij}^{st} \qquad \forall (s,t) \in K, \tag{4.39}$$

that give rise, even in this case, to a mixed-integer nonlinear program.

CHAPTER 5

Subtour elimination in elementary-path problems

In this chapter, we describe and analyze integer programming approaches for elementary-path problems. In particular, we focus on the elementary longest path problem (ELPP), that comes up in the pricing phase of the branch-and-price algorithm for UFP-MMF. Moreover, the same approaches can also be applied directly to the arc formulation of UFP-MMF and its relaxations to impose elementarity of the origin-destination paths.

Several mixed-integer formulations for the ELPP are described in detail in Section 5.2. In Section 5.3 we provide some analytical results, including a proof of equivalence between the polyhedra described by the two strongest formulations. Section 5.4 reports computational experiments where we compare the LP relaxation bounds and branch-and-cut results. Most of the results described in this chapter can be found in [Tac14].

5.1 Related work

The problem of finding an elementary longest path (ELPP) when the costs c_{ij} induce positive cycles on G, is clearly $\mathcal{NP}$-hard due to a simple reduction from the Hamiltonian path problem. The authors in [BHK03] prove that the LPP is hard to approximate on directed graphs within a $n^{1-\epsilon}$ for any ϵ, unless $\mathcal{P} = \mathcal{NP}$. Approximation algorithms have been proposed, e.g., in [AYZ95] and [Scu03].

The ELPP is equivalent to the elementary shortest path problem (ESPP), which is its minimization counterpart, where the costs induce negative cycles. Often, the pricing phase in Vehicle Routing Problems (VRP) involve resource-constrained variants of the elementary shortest path problem (ESPPRC). These problems are usually solved with fast dynamic programming-based labeling algorithms, e.g., see [RS06, FDGG04, BDD06]. It is possible to adapt this kind of approach to the unconstrained ELPP by considering an artificial resource for each node, and imposing that less that one unit of each resource is used, as already proposed in [BC89]. However, this is a very weak constraint, so that the typical approaches for the ESPPRC do not seem to carry over effectively to the ELPP [DI14]. Even when resource constraints are present, labeling algorithms are sometimes not applicable or inefficient, and in those cases branch-and-cut algorithms might be a better choice [JPS08].

5.2 Integer programming formulations

Let us denote with $\delta^+(S)$ and $\delta^-(S)$ the arcs leaving/entering the set $S \subseteq V$, and with $A(S)$ the set of arcs with both ends in $S \subseteq V$. Let us also define $V_i := V \setminus \{i\}$, and $x(S) := \sum_{i \in S} x_i$. In all the formulations, it is assumed w.l.o.g. that $|\delta^-(s)| = |\delta^+(t)| = 0$. A standard arc formulation to determine the longest path from node s to node t is the following:

$$\max \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{5.1}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} - \sum_{(j,i) \in \delta^-(i)} x_{ji} = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{else} \end{cases} \qquad i \in V \tag{5.2}$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} \leq 1 \qquad i \in V \tag{5.3}$$

$$x_{ij} \in \{0, 1\} \qquad (i,j) \in A, \tag{5.4}$$

where $c_{ij} \in \mathbb{R}$ are the arc costs, and x_{ij} are binary arc variables that take value 1 if the arc (i,j) belongs to the optimal path. Constraints (5.2) are balance equations imposing there is a path connecting s and t, while Constraints (5.3) ensure that the outgoing degree of each node is at most one. When the costs c_{ij} induce positive cycles on G, i.e., there is a subtour such that the total cost of its arcs is positive, this system of inequalities is not sufficient to guarantee the elementarity of the path. Thus, additional constraints (and, possibly, variables) are necessary to prevent subtours ensuring that no node is visited more than once.

Notice that the crucial difference with respect to problems where the path has to be Hamiltonian is the absence, in the ELPP, of the degree constraints:

$$\sum_{(i,j)\in\delta^+(i)} x_{ij} = \sum_{(j,i)\in\delta^-(i)} x_{ji} = 1 \qquad i \in V.$$

We now describe different set of constraints and variables that can be added to the Formulation (5.1)–(5.4) so as to obtain a valid integer programming formulation for the ELPP.

Dantzig-Fulkerson-Johnson (DFJ)

For the TSP, success has been achieved with strong formulations with exponentially many constraints. It is possible to write a formulation for the ELPP based on the classical Dantzig-Fulkerson-Johnson subtour elimination constraints [DFJ54], adding to the basic formulation (5.1)–(5.4) the following inequalities:

$$\sum_{(i,j)\in A(S)} x_{ij} \le |S| - 1 \qquad S \subseteq V_{st}, |S| \ge 2. \tag{5.5}$$

In each subset S, subtours are prevented ensuring that the number of arcs in S which are selected is smaller than the number of nodes in S. This formulation includes $O(m)$ variables and $O(2^n)$ constraints.

Observation For the Asymmetric TSP (ATSP), due to the degree constraints, the DFJ subtour eliminations constraints can be written in the equivalent form:

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \ge 1 \qquad S \subset V, |S| \ge 2. \tag{5.6}$$

For the ELPP, Constraints (5.6) are valid only for subsets S with $s \in S$ and $t \notin S$. Moreover, they are not equivalent to (5.5), and they are *not* sufficient to prevent subtours.

Example 5.1. Consider the solution reported in Figure 5.1, assuming it is a complete graph and that only the arcs with $x_{ij} = 1$ are drawn. The solution does not violate any inequality (5.6) for any set S containing s, since $x(\delta^+(S)) = 1$ for any such S, although it contains a (disconnected) subtour. On the other hand, notice that the inequality (5.6) is not valid, e.g., for the set S', not containing s and containing only nodes that are not part of the s-t path, where $x(\delta^+(S')) = 0$.

Generalized cutset inequalities (GCS)

DFJ Constraints (5.5) can be strengthened for the ELPP, by replacing the constant right-hand side with a variable expression, as follows:

$$\sum_{(i,j)\in A(S)} x_{ij} \le \sum_{i\in S\setminus\{k\}} \sum_{(i,j)\in\delta^+(i)} x_{ij} \qquad \forall k \in S \subseteq V_{st}, \; |S| \ge 2. \tag{5.7}$$

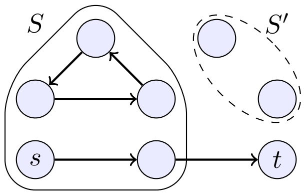


Figure 5.1: *An example where Constraints* (5.6) *are not sufficient to prevent subtours.*

These inequalities can be interpreted as imposing that the number of selected arcs in a subset S is strictly smaller than the number of nodes in S that belong to the s-t path.

An equivalent version of these inequalities can be obtained considering the cutset variant, which is more easily separated. This approach was used for a symmetric version of ESPPRC in [JPS08] and applied to the asymmetric ESPP in [DI14]. Similar subtour elimination constraints have also been used in branch-and-cut algorithms for the VRP [NR02] or variants of the TSP with profits [FGT98, JPSP14]. We refer to them as *generalized cutset inequalities* (*GCS*):

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq \sum_{(k,j)\in\delta^+(k)} x_{kj} \qquad k \in S \subseteq V_{st}, |S| \geq 2. \tag{5.8}$$

Constraints (5.8) prevent subtours by ensuring that, for each subset S, the number of selected arcs leaving S is greater or equal to the number of selected arcs outgoing from any node in S. In an integer solution, this means that the cut induced by S must contain at least one arc if at least one node in S belongs to the s-t path, while, if S does not contain any node in the s-t path, the constraint is the trivial inequality $0 \geq 0$. The number of variables in the formulation is $O(m)$, while the number of constraints is $O(n2^n)$.

Proposition 5.2. *Constraints* (5.7) *and* (5.8) *are equivalent.*

Proof. For all $k \in S$, $x(\delta^+(k)) \leq x(\delta^+(S)) = x(\delta^+(S)) + x(A(S)) - x(A(S)) = \sum_{i\in S} x(\delta^+(i)) - x(A(S))$, where the inequality follows from (5.8). □

Sequential formulations (MTZ)

To derive an extended formulation à la Miller, Tucker and Zemlin [MTZ60] (hereafter MTZ) it is enough to introduce, for each node, an auxiliary variable that can be viewed as the order in which the node is visited and a constraint for each arc:

$$t_j \geq t_i + 1 + (n-1)(x_{ij} - 1) \qquad \begin{array}{c}(i,j) \in A \\ i \neq s, j \neq t.\end{array} \tag{5.9}$$

For the ATSP, this formulation is well-known to give poor linear relaxation bounds, and we include it only for completeness. However, it is very compact, as it requires only $O(m)$ additional constraints and $O(n)$ auxiliary variables.

Reformulation-linearization based (RLT)

From the following nonlinear version of the MTZ formulation:

$$t_j x_{ij} = (t_i + 1)x_{ij} \qquad (i,j) \in A \tag{5.10}$$

$$t_j x_{sj} = x_{sj} \qquad (s,j) \in \delta^+(s), \tag{5.11}$$

the authors in [HMM13] use the Sherali-Adams *reformulation-linearization technique* to obtain the following stronger formulation for the ELPP:

$$\alpha_{ij} = \beta_{ij} + x_{ij} \qquad (i,j) \in A \tag{5.12}$$

$$x_{sj} + \sum_{\substack{(i,j)\in\delta^-(j)\\ i\neq s}} \alpha_{ij} - \sum_{(j,i)\in\delta^+(j)} \beta_{ji} = 0 \qquad \begin{array}{r}(s,j) \in \delta^+(s),\\ j \neq t\end{array} \tag{5.13}$$

$$\sum_{(i,j)\in\delta^-(j)} \alpha_{ij} - \sum_{(j,i)\in\delta^+(j)} \beta_{ji} = 0 \qquad \begin{array}{r}(s,j) \notin \delta^+(s)\\ j \neq s, j \neq t\end{array} \tag{5.14}$$

$$x_{ij} \leq \alpha_{ij} \qquad \begin{array}{r}(i,j) \in A,\\ i \neq s, j \neq s\end{array} \tag{5.15}$$

$$\alpha_{ij} \leq (n-1)x_{ij} \qquad (i,j) \in A, i \neq s \tag{5.16}$$

$$x_{ij} \leq \beta_{ij} \qquad \begin{array}{r}(i,j) \in A,\\ i \neq s, j \neq s\end{array} \tag{5.17}$$

$$\beta_{ij} \leq (n-1)x_{ij} \qquad (i,j) \in A : i \neq s \tag{5.18}$$

$$\beta_{ij} \leq \alpha_{ij} \qquad (i,j) \in A. \tag{5.19}$$

This extended formulation requires $O(m)$ constraints and $O(m)$ auxiliary variables.

Single-flow formulation (SF)

A formulation which is close in spirit to the single-flow ATSP formulation of [GG78], can be obtained introducing an auxiliary flow q to be delivered to the nodes belonging to the s-t path:

$$q_{ij} \leq (n-1)x_{ij} \qquad (i,j) \in A \tag{5.20}$$

$$\sum_{(s,j)\in\delta^+(s)} q_{sj} = \sum_{k\in V} z_k \tag{5.21}$$

$$\sum_{(i,k)\in\delta^-(k)} q_{ik} - \sum_{(k,j)\in\delta^+(k)} q_{kj} = z_k \qquad k \in V_s \tag{5.22}$$

$$\sum_{(i,k)\in\delta^-(k)} x_{ik} = z_k \qquad k \in V_s \tag{5.23}$$

$$q_{ij} \geq 0 \qquad (i,j) \in A. \tag{5.24}$$

Constraints (5.20) impose that the auxiliary flow is positive only over the arcs where $x_{ij} = 1$. The auxiliary flow leaving from the node s has value equal to the number of nodes that are reached by the s-t path. Constraints (5.22) ensure that the balance of the auxiliary flow on each node is equivalent to z_k, which, according to Constraint (5.23), is either 1, if node k is in the s-t path, or 0 otherwise.

Multicommodity-flow formulation (MCF)

An extension of the single-flow formulation is obtained by disaggregating the auxiliary flow into $n-1$ unitary flows. Subtours are prevented by enforcing, in the support graph, one unit of a distinct auxiliary flow from s to each node that belongs to the s-t path.

$$q^k_{ij} \leq x_{ij} \qquad \begin{array}{r} k \in V_s, \\ (i,j) \in A \end{array} \tag{5.25}$$

$$\sum_{(i,j)\in\delta^+(i)} q^k_{ij} - \sum_{(j,i)\in\delta^-(i)} q^k_{ji} = \begin{cases} z_k & \text{if } i = s \\ -z_k & \text{if } i = k \\ 0 & \text{else} \end{cases} \qquad \begin{array}{c} i \in V, \\ k \in V_s \end{array} \tag{5.26}$$

$$\sum_{(k,i)\in\delta^+(k)} x_{ki} = z_k \qquad k \in V_{st} \tag{5.27}$$

$$\sum_{(i,k)\in\delta^-(k)} x_{ik} = z_k \qquad k \in V_s \tag{5.28}$$

$$\sum_{(s,j)\in\delta^+(s)} x_{sj} = 1 \tag{5.29}$$

$$\sum_{(i,t)\in\delta^-(t)} x_{it} = 1 \tag{5.30}$$

$$q^k_{ij} \geq 0, z_k \in \{0,1\} \qquad \begin{array}{r} (i,j) \in A, \\ k \in V_s. \end{array} \tag{5.31}$$

The formulation includes $O(nm)$ additional variables and constraints. This extended formulation is introduced, for the ELPP, in [IMM09], and it is very similar to classic multi-commodity flow formulations for the ATSP proposed by Wong [Won80] and Claus [Cla84].

In Table 5.1 we summarize the considered formulations.

Table 5.1: *A summary of the considered formulations.*

	vars	cons	description
DFJ	m	2^n	Dantzig-Fulkerson-Johnson SECs(5.5)
GCS	m	$n2^n$	generalized cutsets (5.8)
MTZ	m	n	Miller-Tucker-Zemlin (5.9)
RLT	m	m	reformulation-linearization (5.12)–(5.19)
SF	m	m	single-flow (5.20)-(5.24)
MCF	nm	nm	multi-commodity flow (5.25)-(5.30)

5.3 Polyhedral results

Let us describe some analytical results for the considered ELPP formulations.

Proposition 5.3. *Formulation* MCF *is stronger than formulation* SF*.*

Proof. Constraints (5.20)–(5.22) can be obtained by *MCF* simply aggregating Constraints (5.25)–(5.26) over $k \in V_s$, and then substituting $\sum_{k\in V_s} q_{ij}^k$ with q_{ij}. The example in Figure 5.2 shows that the inclusion is strict. □

Proposition 5.4. *Formulation* GCS *is stronger than formulation* DFJ*.*

Proof. The result follows by considering *GCS* as stated in (5.7), whose right-hand side is obviously smaller or equal to $|S| - 1$, right-hand side in (5.5), and the inclusion is strict by the example in Figure 5.2. □

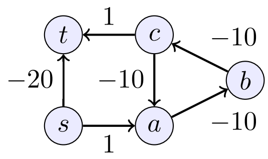


Figure 5.2: *Example proving strict inclusion for Proposition 5.3 and 5.4. With* GCS *and* MCF*, the LP optimal solution is the one with $x_{st} = 1$ and optimal value -20. With* SF*, the optimal solution has value -26, with $x_{st} = x_{ca} = \frac{1}{4}$, $x_{sa} = x_{cta} = \frac{3}{4}$ and $x_{ab} = x_{bc} = 1$. With* DFJ*, the solution has value -40, with $x_{st} = 1$ and a disconnected subtour with $x_{ab} = x_{bc} = x_{ca} = \frac{2}{3}$.*

We will now show that *GCS* is as tight as formulation *MCF*. This requires to calculate the projection of the *MCF* extended formulation into the space of the x

variables. We will make use of two strong results presented in [PS91], and follow a similar approach to the equivalence proofs therein.

Theorem 5.5. *The projection of the* MCF*-polytope onto the x-space is equivalent to the* GCS*-polytope.*

Proof. Let us rewrite formulation *MCF* by projecting out the z_k variables. Constraints (5.34) and (5.35) follow from Constraint (5.28), while Constraints (5.27) and (5.28). Assume again that the graph G does not contain the arcs $\delta^-(s)$ and $\delta^+(t)$. The compact formulation reads:

$$q_{ij}^k \leq x_{ij} \qquad k \in V_s, \ (i,j) \in A \tag{5.32}$$

$$\sum_{(i,j)\in\delta^+(i)} q_{ij}^k - \sum_{(j,i)\in\delta^-(i)} q_{ji}^k = 0 \qquad i \in V_s, i \neq k, \ k \in V_s \tag{5.33}$$

$$\sum_{(s,j)\in\delta^+(s)} q_{sj}^k = \sum_{(i,k)\in\delta^-(k)} x_{ik} \qquad k \in V_s \tag{5.34}$$

$$\sum_{(j,k)\in\delta^-(k)} q_{jk}^k = \sum_{(i,k)\in\delta^-(k)} x_{ik} \qquad k \in V_s \tag{5.35}$$

$$\sum_{(i,k)\in\delta^+(k)} x_{ik} = \sum_{(i,k)\in\delta^-(k)} x_{ik} \qquad k \in V_{st} \tag{5.36}$$

$$\sum_{(s,j)\in\delta^+(s)} x_{sj} = 1 \tag{5.37}$$

$$\sum_{(i,t)\in\delta^-(t)} x_{it} = 1 \tag{5.38}$$

$$q_{ij}^k \geq 0, x_{ij} \geq 0 \qquad (i,j) \in A, k \in V_s. \tag{5.39}$$

In order to compare *MCF* and *GCS*, we need to project out also the q-variables of the *MCF* formulation. Let us define the sets:

$$\begin{aligned}
&X = \{\underline{x} \in \mathbb{R}^m \mid \underline{x} \text{ satisfies (5.36), (5.37) and (5.38)}\},\\
&P_{GCS} = \{\underline{x} \in X \mid \underline{x} \text{ satisfies (5.8)}\},\\
&P_{MCF} = \{(\underline{x}, \underline{q}) \in \mathbb{R}^{mn} \mid (\underline{x}, \underline{q}) \text{ satisfies (5.32)–(5.39)}\},\\
&Proj_x(P_{MCF}) = \{\underline{x} \in X \mid \exists\ \underline{q} \text{ s.t. } (\underline{x}, \underline{q}) \in P_{MCF}\},
\end{aligned}$$

where P_{GCS} is the *GCS*-polytope, P_{MCF} is the *MCF*-polytope and $Proj_x(P_{MCF})$ is its projection onto the x-space. It is convenient to rewrite Constraints (5.32)–(5.35) in matrix form as follows:

$$B\underline{x} + M\underline{q} = \underline{0} \tag{5.40}$$

$$-D\underline{x} + I\underline{q} \leq \underline{0} \tag{5.41}$$

$$\underline{x}, \underline{q} \geq \underline{0}. \tag{5.42}$$

Equation (5.40) corresponds to (5.33)–(5.35), while (5.41) corresponds to (5.32). One can easily notice that, since the $n-1$ flows are independent, the system can be decomposed according to the index k. The matrix M is, in fact, block-diagonal, and can be decomposed into $n-1$ blocks M_k for all $k \in V_s$. Each block M_k represents the node-arc incidence matrix of the graph G. I is the identity matrix, that can be decomposed into submatrices I_k of dimension $m \times m$. In a similar fashion, we also decompose B and D into submatrices B_k and D_k. The rows and columns of B_k correspond, respectively, to the nodes and arcs of the graph G. A submatrix B_k has zeros everywhere, except for the row corresponding to node s, with entries of value -1 for each arc in $\delta^-(k)$, and the row corresponding to node k, with $+1$ entries for each arc in $\delta^-(k)$. Each row of D_k corresponds to a variable q_{ij}^k and has zeros everywhere, except for a $+1$ in the column associated with variable x_{ij}.

We now make use of Theorem 2 in [PS91], by which the projection onto the x-space of the polytope P_{MCF} defined by (5.40)–(5.41) can be obtained as:

$$\begin{aligned} Proj_x(P_{MCF}) = \{\underline{x} \in X \mid (\underline{u}B - \underline{v}D - \underline{w})\underline{x} \leq 0 \\ \forall(\underline{u}, \underline{v}, \underline{w}) \in C\}, \end{aligned} \tag{5.43}$$

where C is the cone defined as:

$$C = \{(\underline{u}, \underline{v}, \underline{w}) \mid \underline{u}M + \underline{v}I \geq 0, \underline{v} \geq 0, \underline{w} \geq 0\}.$$

The result allows us to carry out the comparison between P_{GCS} and $Proj_x(P_{MCF})$ simply by finding a system of generators for the cone C.

From the inequalities $\underline{w} \geq \underline{0}$ we obtain extreme rays of the form $\underline{u} = \underline{0}$, $\underline{v} = \underline{0}$, $\underline{w} = \underline{e}_i$, where $\underline{e}_i$ is the i-th standard basis vector of $\mathbb{R}^m$, that yield the nonnegativity constraints

$$x_{ij} \geq 0 \qquad \forall\ (i,j) \in A. \tag{5.44}$$

This allows us to restrict our following study to the cone C' defined as:

$$C' = \{(\underline{u}, \underline{v}) \mid \underline{u}M + \underline{v}I \geq 0, \underline{v} \geq 0\}.$$

Exploiting the decomposition of M, we can work on the even smaller cones:

$$C_k = \{(\underline{u}^k, \underline{v}^k) \mid \underline{u}^k M_k + \underline{v}^k \geq 0, \underline{v}^k \geq 0\}. \tag{5.45}$$

Due to (5.43), once we have the system of generators $(\underline{u}^k, \underline{v}^k)$ for each cone C_k, the constraints in the x-space are obtained by calculating $(\underline{u}^k B_k - \underline{v}^k D_k)\underline{x} \leq \underline{0}$ for each $k \in V_s$.

According to Proposition 6 in [PS91], a full system of generators of a cone C_k defined as in (5.45), where M_k is a node-arc incidence matrix of a digraph, is given by:
- a basis of its lineality space, of the form:

$$\underline{u}^k = \pm\underline{e},\ \underline{v}^k = \underline{0},$$

where $\underline{e}$ is the all-ones vector, that in our case translate to the trivial equality $\underline{0} = \underline{0}$, and
- the extreme rays, given by all the positive multiples of the vector $(\underline{u}^k, \underline{v}^k)$ such that:

$$(i)\;\; u_i^k = 0 \quad \forall i \in V, \qquad v_{ij}^k = \begin{cases} 1 & \text{for one } (i,j) \in A \\ 0 & \text{otherwise} \end{cases}$$

$$(ii)\; u_i^k = \begin{cases} 1 & \forall\, i \in S, \\ 0 & \text{otherwise} \end{cases} \qquad v_{ij}^k = \begin{cases} 1 & \forall\, i \in \bar{S}, j \in S, \\ 0 & \text{otherwise} \end{cases}$$

$$(iii)\; u_i^k = \begin{cases} -1 & \forall\, i \in S, \\ 0 & \text{otherwise} \end{cases} \qquad v_{ij}^k = \begin{cases} 1 & \forall\, i \in S, j \in \bar{S}, \\ 0 & \text{otherwise} \end{cases}$$

for any $S \subseteq V$, where $\bar{S} = V \setminus S$. The extreme rays of the form (i) give rise to nonnegativity constraints.

From the extreme rays given by (ii) and (iii) we obtain the inequalities:

$$-\,x(\delta^-(S)) + u_k x(\delta^-(k)) - u_s x(\delta^-(k)) \le 0 \tag{5.46}$$
$$-\,x(\delta^+(S)) - u_k x(\delta^-(k)) + u_s x(\delta^-(k)) \le 0 \tag{5.47}$$

where $u_i = 1$ if $i \in S$, and 0 otherwise. For both (5.46) and (5.47), we can distinguish four cases depending on whether s and k are in S, thus whether u_s, u_k are 0 or 1. If both s and k are in S, or neither of them is, the inequality is implied by the nonnegativity constraints (5.44). If only the coefficient with negative sign is nonzero, the corresponding inequality is, again, redundant. Therefore, the only meaningful cases are the following:

$$x(\delta^-(S)) \ge x(\delta^-(k)) \qquad \forall S \subseteq V, s \notin S, k \in S \tag{5.48}$$
$$x(\delta^+(S)) \ge x(\delta^-(k)) \qquad \forall S \subseteq V, s \in S, k \notin S. \tag{5.49}$$

We have established so far that $Proj_x(P_{MCF})$ is fully described by the nonnegativity constraints and Constraints (5.48)–(5.49). This set of inequalities can be shown to be equivalent to:

$$x(\delta^+(S)) \ge x(\delta^+(k)) \qquad \forall S \subseteq V_{st}, k \in S. \tag{5.50}$$

Constraint (5.48) and (5.49) are equivalent, due to the fact that $x(\delta^+(S)) = x(\delta^-(\bar{S}))$. Let us then consider only (5.48). For $k = t$, the inequality is trivially satisfied by all $x \in X$, thus redundant. For $k \neq t$ and $t \notin S$, we obtain exactly the inequalities in (5.50), since by (5.36)–(5.38), we have that $x(\delta^-(k)) = x(\delta^+(k))$ and $x(\delta^-(S)) = x(\delta^+(S))$ for any S containing neither s nor t. If $k \neq t$ and $t \in S$, it suffices to observe that, since $\delta^+(t) = 0$, the inequality $x(\delta^-(S)) \ge x(\delta^-(k))$ is implied by $x(\delta^-(S \setminus \{t\})) \ge x(\delta^-(k))$, which, again, can be rewritten in the form (5.50).

Hence, the projection of P_{MCF} onto the x-space is given by

$$Proj_x(P_{MCF}) = \{\underline{x} \in X \mid \underline{x} \text{ satisfies (5.36)–(5.38) and (5.50)}\},$$

and it follows that $Proj_x(P_{MCF}) = P_{GCS}$. $\square$

From this result and Proposition 5.4, it also follows that formulation *MCF* is stronger than formulation *DFJ*.

5.4 Computational comparison

To understand the behavior of the formulations, we report computational experiments not only with pricing subproblems of the path formulation of UFP-MMF, but also with other instances from the literature. We will compare the described formulations with respect to their LP relaxation bounds and their behavior within an exact branch-and-bound framework.

Three types of instances are considered for the tests. The first set consists of 180 instances from the pricing phase of UFP-MMF on small-sized networks from the SNDlib [OWPT10], namely, the topologies, `atlanta`, `france`, `geant`, `germany`, `nobel-us`, `polska`. The second set is a set of 240 small to medium-sized random-cost graphs, either sparse (`rnd-s`) or dense (`rnd-d`). The graphs for the instances in `rnd-s` are generated by building a connected component including all the nodes, and then randomly adding arcs until the desired sparsity is reached. The instances in `rnd-d` are the dense instances in [Dre13], with random arc costs on a complete graph. The third set (`prc`) contains 270 pricing instances from [Dre13]. It consists of small and medium-size pricing problems from a column generation algorithm for the asymmetric m-salesmen TSP at the first (`f`), penultimate (`p`) and last (`l`) pricing iterations. Table 5.2 summarizes the features of the test instances.

	n	m	range	#
`polska`	12	36	[−10000,10000]	30
`nobel-us`	14	42	[−10000,10000]	30
`atlanta`	15	44	[−10000,10000]	30
`geant`	22	72	[−10000,10000]	30
`france`	25	90	[−10000,10000]	30
`germany`	50	176	[−10000,10000]	30
`rnd-s`	50/100/200	164/660/2654	[−1000,1000]	90
`rnd-s`	500/1000	16634/66601	[−1000,1000]	60
`rnd-d`	25/50/100	600/2450/9900	[−1000,1000]	90
`prc-f`	27/52/102	702/2652/10302	$[-10^8, -9.48 \cdot 10^7]$	90
`prc-p`	27/52/102	702/2652/10302	$[-4 \cdot 10^4, 5.18 \cdot 10^6]$	90
`prc-l`	27/52/102	702/2652/10302	$[-4 \cdot 10^4, 5.18 \cdot 10^6]$	90

Table 5.2: *Description of the instances.*

5.4.1 Linear programming relaxation bounds

The LP relaxation bounds are computed constructing the complete model for the extended formulations and for formulation *DFJ*. We have not implemented a separation procedure for the DFJ SECs, since they are dominated by the *GCS* inequalities and there is no point in using them in practice. However, we include them here (when the size of the network allows it) to give an idea of difference of the bound with respect to the GCS inequalities. For formulation *GCS*, we use a Min Cut-based separation procedure (its implementation details are left to the next section). The tests are carried out with IBM ILOG Cplex 12.5 on an Intel Xeon E5645 @2.40GHz.

Table 5.3 reports the average gap of the LP relaxation bounds with respect to the optimal integer values, computed as $100\frac{|Bound_{LP}-Opt|}{|Bound_{LP}|}$. The results confirm that the LP bounds of *MCF* and *GCS* are equivalent, and show that they are by far the tightest formulations. Remarkably, *MCF* and *GCS* close the gap on a good fraction of instances (roughly 50%), while all the remaining formulations have a relaxation with 0 gap only in less than 10% of the considered instances. However, on the largest instances, the LP relaxation of the *MCF* formulation could not be solved within the time limit of 1200 seconds. Observe the considerable difference between *GCS* and classic *DFJ*, while formulations *RLT* and *SF* provide similar bounds.

For the `prc-f` instances, all the extended formulations provide good bounds. The reason is that, on the majority of those instances, all the arcs have very similar costs, hence the integrality gap is inherently small for all the formulations. It is worthwhile to point out that, even when the bounds are very good, weaker formulations find solutions with many more fractional values.

5.4.2 Branch-and-cut

Since we aim at integer solutions, let us compare the behavior of an off-the-shelf MIP solver with the considered formulations. The formulations and the separation procedures were implemented in C++ with IBM Ilog Cplex/Concert 12.5, using default settings.

For the polynomial-size extended formulations, the full model is built. Formulation *DFJ* is not included in these tests. For *GCS*, we report results obtained with two different separation routines, that we denote by *GCS-StrongComp* and *GCS-MinCut*. For *GCS-StrongComp*, the separation is carried out for fractional and integer solutions, identifying the strongly connected components in the support graph induced by the variables x_{ij}. This can be done in a $O(n+m)$ running time with Tarjan's algorithm. Once a strong component S has been found, it is enough to check if Constraint (5.8) is violated for any of the nodes in S. This separation procedure is efficient, but not guaranteed to find all the violated inequalities on fractional solutions. Correctness is preserved by the fact that the procedure is exact

	GCS	*DFJ*	*MTZ*	*RLT*	*SF*	*MCF*
`polska`	37.87	67.85	69.57	61.73	63.35	37.87
`nobel-us`	26.48	66.60	71.96	69.07	69.72	26.48
`atlanta`	9.91	35.25	42.80	36.46	37.53	9.91
`geant`	5.52	-	32.02	30.90	31.06	5.52
`france`	4.33	-	64.50	59.88	60.37	4.33
`germany`	2.03	-	31.47	31.29	31.31	2.03
`rnd-50-s`	0.43	-	11.38	10.91	10.94	0.43
`rnd-100-s`	0.22	-	1.57	1.52	1.53	0.22
`rnd-200-s`	0.02	-	0.42	0.42	0.42	-
`rnd-500-s`	0.00	-	0.06	0.06	0.06	-
`rnd-25-d`	0.19	-	0.56	0.46	0.47	0.19
`rnd-50-d`	0.04	-	0.14	0.12	0.12	0.04
`rnd-100-d`	0.01	-	0.03	0.03	0.03	-
`prc-25-f`	0.00	-	0.03	0.02	0.02	0.00
`prc-50-f`	0.00	-	0.01	0.01	0.01	0.00
`prc-100-f`	0.00	-	0.00	0.00	0.00	-
`prc-25-p`	9.72	-	91.68	87.41	87.65	9.72
`prc-50-p`	7.92	-	80.50	77.40	77.49	7.92
`prc-100-p`	1.96	-	60.70	42.66	42.93	-
`prc-25-l`	1.34	-	86.73	81.06	81.39	1.34
`prc-50-l`	2.18	-	54.75	48.99	49.17	2.18
`prc-100-l`	2.44	-	60.87	42.55	42.83	-

Table 5.3: *Average LP relaxation gaps (%) w.r.t. the optimal integer solution. Missing values are due to time (*MCF*) or memory limits (*DFJ*).*

for integer solutions. For *GCS-MinCut*, the separation is carried out on fractional solutions by solving a sequence of Min Cut (or Max Flow) problems between each node and t, with an overall worst-case complexity of $O(n^3\sqrt{n})$ using the highest-label preflow-push algorithm described, e.g., in [AMO93]. All violated inequalities are identified, although with higher computational cost. On the incumbents, the faster strong components-based procedure is used.

For the Min Cut problems and the identification of strongly connected components, we use the efficient implementations in the open-source LEMON Graph Library 1.3 [DJK11]. We refer the interested reader to [Dre13] for additional considerations on the separation of subtour elimination constraints for the ELPP.

In Table 5.4 we summarize the results over the test instances. Column "opt" reports the fraction of instances that are solved to optimality within the time limit of 1200 seconds. Column "time" reports the average computing time. Column "nodes" reports the average number of explored nodes in the branch-and-bound tree. Column "cuts" reports the average number of *GCS* inequalities added by the separation procedures.

Both *GCS* variants solve to optimality all the instances, except the ones in `rnd-1000-s`, and prove to be by far the best choice for the largest instances. Formulation *GCS-MinCut* usually explores fewer nodes in the search tree, while the number of cuts is similar. However, with respect to the average computing time, *GCS-StrongComp* has an advantage over *GCS-MinCut*, especially on the largest instances, where solving a sequence of Min Cut problems is computationally heavy. *GCS-StrongComp* is the only formulation able to solve one of the `rnd-1000-s` instances, with 1000 nodes and 66601 arcs, within the time limit.

Compared to the extended formulations, *GCS-StrongComp* is a clear winner on the `rnd` and `prc` instances, where it is often more than one order of magnitude faster. Formulation *MCF*, despite the good LP bounds, appears to be too heavy for large-sized instances.

Figure 5.3 summarizes the computational experiments. On the y-axis, we report the fraction of all the instances that are solved to optimality within the time on the x-axis. Note that the left part of the x-axis is in linear scale, while the right part is logarithmic. *GCS-StrongComp* is the topmost curve, solving more than 80% of the instances within 20 seconds, and 90% in 50. *GCS-MinCut* is not far behind, although it is generally slower. Both solve over 95% of the instances within 1200 seconds. Formulations *RLT*, *SF* and *MTZ* have similar results on the easiest instances, although, overall, *SF* is able to solve around 85% of the instances within the time limit, while *RLT* and *MTZ* solve, respectively, less then 78% and 72% of them. Formulation *MCF* (bottom curve) solves the smallest fraction of the instances (around 65%).

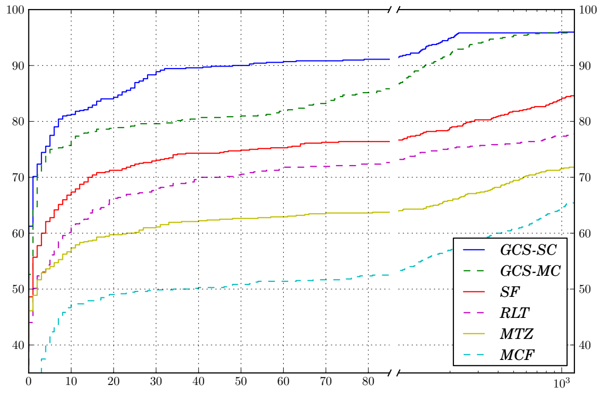


Figure 5.3: *Fraction of instances solved to optimality within a given time. The x-axis is in linear scale before the break (up to 85), logarithmic scale after the break (90 to 1200).*

Focusing on the UFP-MMF pricing instances, the results show that all formulations are close as long as the size is modest. On the `germany` instances, *MCF* and *MTZ* are clearly worse than the others. Then, in the pricing phase of the branch-and-price algorithm, we will use the *GCS* formulation with the strong component-based separation procedure, which seems to be, on average, the fastest and most stable approach. From these computational results, we expect that *GCS* should behave well also in the arc formulation of UFP-MMF.

	MCF			*GCS-StrongComp*				*GCS-MinCut*				*RLT*			*SF*			*MTZ*		
	%		B%B	%		B%B		%		B%B		%		B%B	%		B%B	%		B%B
	opt	time	nodes	opt	time	nodes	cuts	opt	time	nodes	cuts	opt	time	nodes	opt	time	nodes	opt	time	nodes
`polska`	100	0	0	100	0	3.5	20.7	100	0	2.6	20.2	100	0	7.2	100	0	0.3	100	0	32.2
`nobel-us`	100	0.1	0	100	0	2.8	21.8	100	0	2.1	21.8	100	0	8.1	100	0	2.9	100	0	23.7
`atlanta`	100	0.1	0	100	0	1.3	22.3	100	0	0.8	22	100	0	1.7	100	0	0.5	100	0	17.9
`geant`	100	0.2	0	100	0.1	3.6	45.8	100	0.1	2.6	41.1	100	0.1	6.7	100	0.1	1.3	100	0	22.1
`france`	100	0.4	0	100	0.1	1.7	44.5	100	0.1	1.1	46	100	0.1	30.8	100	0.2	8.2	100	0.1	279.8
`germany`	100	8.7	7.9	100	0.9	58.7	379.6	100	1.4	21.7	392.6	100	0.5	177.5	100	0.7	207.3	100	4.8	8189.5
`rnd-50-s`	100	4.7	9	100	0.2	40.2	152.3	100	0.3	12.1	144.7	100	0.3	119.1	100	0.3	48.9	100	0.2	118.7
`rnd-100-s`	30	886	17.2	100	1.1	67.2	221	100	2.2	33.9	239	100	5	529.2	100	3.7	326.9	100	2.4	652.5
`rnd-200-s`	0	1200	0	100	6.6	46.1	351.4	100	14.7	38.7	367.5	100	55.7	930.8	100	10.4	103.8	100	11.1	655.2
`rnd-500-s`	0	1200	0	100	156.3	79.3	777.7	100	238.1	52.3	707	5.6	1173.1	326.8	72.2	615.8	428.9	100	355	2606
`rnd-1000-s`	0	1200	0	3.3	1192.0	6.4	1417.6	0.0	1200	0.0	1275.5	0.0	1200	0.0	0.0	1200	0.0	0.0	1200	46.1
`rnd-25-d`	100	3.4	3	100	0.1	5.1	40.2	100	0.1	3.7	39.6	100	0.2	9.9	96.7	0.1	5.3	100	0.1	15.1
`rnd-50-d`	86.7	479.1	7.7	100	0.4	12.1	85.8	100	0.4	11.4	78.1	100	1.5	51	100	1.1	50.1	100	2.4	686.5
`rnd-100-d`	0	1200	0	100	3.1	26.2	159	100	3.6	20.5	147	100	37.2	219.8	100	27.8	188	100	11.4	147.7
`prc-25-f`	100	6.1	5.1	100	0.1	10	42.8	100	0.1	12	46.2	100	9.8	1348.5	100	0.4	17.1	100	9	13785
`prc-50-f`	70	748.9	13.3	100	0.6	37.5	100.2	100	1.3	20.9	123.5	96.7	100	5314.6	100	23.2	1601.5	80	339.8	$1.6 \cdot 10^5$
`prc-100-f`	0	1200	0	100	13.7	2061.3	276.7	100	100.6	2436.6	415.3	10	1156.8	16036.4	50	839	11586	6.7	1148	98779.5
`prc-25-p`	100	6	3.6	100	0.2	14	64.5	100	0.2	5.1	63.3	100	1.8	3444.7	100	1	358.9	53.3	668.7	$1.1 \cdot 10^6$
`prc-50-p`	93.3	406.1	3.8	100	1	29.4	128.6	100	1.7	7.2	137.1	100	24.5	23169.4	100	44.7	6617.4	40	937.1	$7.4 \cdot 10^5$
`prc-100-p`	0	1200	0	100	34.2	2242.9	549.2	100	115.8	730.6	727.8	0	1200	$2.8 \cdot 10^5$	13.3	1155.8	$2.3 \cdot 10^5$	0	1200	$2.5 \cdot 10^5$
`prc-25-l`	100	6.8	0.4	100	0.3	16.6	77.9	100	0.3	1.4	80.6	100	51.6	25478.5	100	1.5	119.7	33.3	820.8	$1.4 \cdot 10^6$
`prc-50-l`	86.7	415.7	9.6	100	1.8	85.7	224.4	100	2.6	12	184.1	50	790.4	79178.6	96.7	217.1	24884.5	10	1085.1	$6.8 \cdot 10^5$
`prc-100-l`	0	1200	0	100	38.2	3349.4	508.1	100	130.6	1489.1	746.9	0	1200	$2.9 \cdot 10^5$	6.7	1184.5	$2.3 \cdot 10^5$	0	1200	$2.6 \cdot 10^5$

Table 5.4: *Branch-and-cut results.*

CHAPTER 6

Computational experiments

In this chapter we report computational results of the MIP-based and heuristic approaches for UFP-MMF and its relaxations. We first describe the instances used in the computational experiments and some implementation details. Section 6.3 is devoted to the exact methods: we first discuss computational experiments with the branch-and-price algorithm, and then report the results obtained with the branch-and-cut approach. Section 6.4 discusses computational results of heuristic algorithms for UFP-MMF, which are essential in order to obtain good solutions for the most challenging instances. Section 6.5 includes computational experiments with the relaxations of UFP-MMF described in Chapter 4. Section 6.6 describes how the feasible solutions obtained with the heuristics and the upper bound obtained from the relaxations are effective in closing the gap for a large part of the instances. In Section 6.7 we analyze the impact of max-min fairness on the achievable throughput and the structure of the solutions for different variants of the problem. Finally, in Section 6.8 we give some concluding remarks.

6.1 Instances

The computational experiments have been carried out on a set of network topologies obtained from the Survivable Network Design library [OWPT10]. The library contains realistic backbone networks used as benchmarks for network design problems. For each topology, we consider a set of instances that differ for the arc capacities and for the set K of origin-destination pairs. The main features of the considered instances are summarized in Table 6.1, that reports the size of the graph (number of nodes and arcs), the cardinality of the considered sets K (let $k := |K|$), and the number of different UFP-MMF instances that are included for each topology. Since the library contains undirected graphs, the directed graphs

Table 6.1: *Summary of the instances used in the experiments.*

	nodes	arcs	k	number of instances
`polska`	12	36	{10, 21, 28, 36, 42, 50}	4 for each k (24)
`nobel-us`	14	42	{15, 21, 28, 36, 42, 50}	4 for each k (24)
`atlanta`	15	44	{12, 20, 30, 42}	4 for each k (16)
`france`	25	90	{10, 15, 21, 28, 36, 45}	4 for each k (24)
`geant`	22	72	{12, 20, 30, 42, 56, 72}	3 for each k (18)
`germany`	50	176	{10, 15, 20, 26, 34, 41}	3 for each k (18)

are obtained simply considering, for each original edge, the arcs in both directions. The sets $K \subset V \times V$ of origin-destination pairs are extracted randomly from a subset of candidate nodes. Specifically, for the topologies `polska`, `nobel-us`, `atlanta` and `france`, for a given cardinality k we consider two different sets K_1 and K_2, and for each of them we have an instance with uniform capacities and one with non-uniform ones (thus, a total of 4 different instances for each value of k). For the topologies `geant` and `germany`, for each cardinality k we consider a single set K, and we build three instances: one with uniform capacities, and two with non-uniform ones (thus, 3 different instances for each value of k). The arc capacities in the uniform instances are $c_{ij} = 1000\ \forall (i,j) \in A$, while in the non-uniform instances they are assigned randomly to each arc from a discrete set of predefined values, namely $c_{ij} \in \{1000, 2500, 5000, 10000\}$, with probability respectively $\{10\%, 20\%, 40\%, 30\%\}$. Subsets of these instances have already been used in [ACT14, ACCG13].

In Figure 6.1, we report, when available, the representation of the original network topologies.

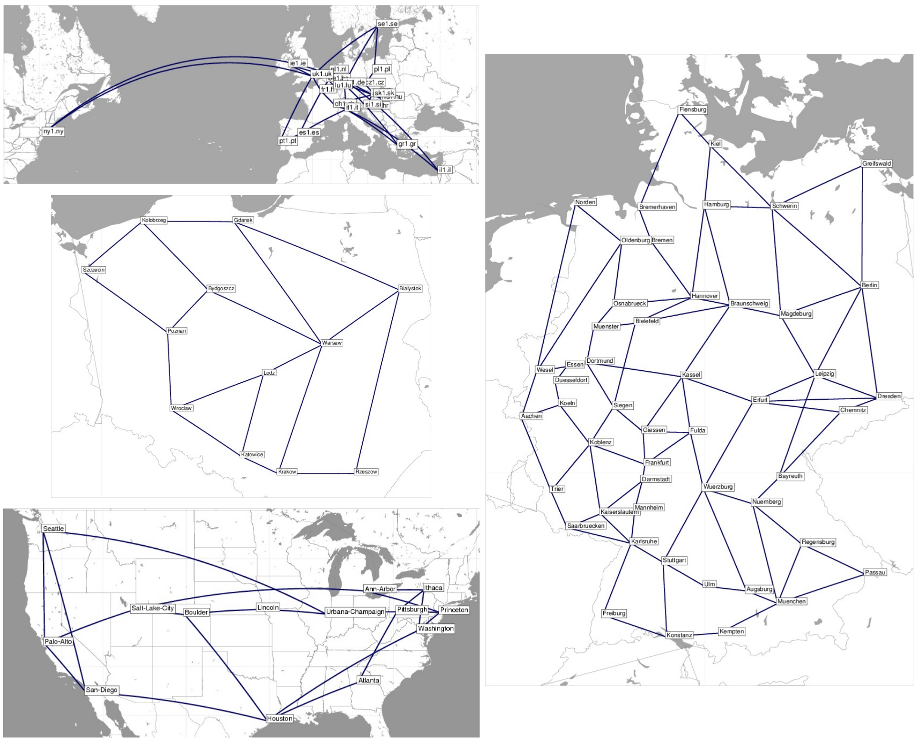


Figure 6.1: *Network topologies, in clockwise order:* `geant`*,* `germany`*,* `nobel-us` *and* `polska`*.*

6.2 Implementation

The branch-and-cut algorithm for the arc formulation of UFP-MMF (Section 3.2), its relaxations (Chapter 4), and the restricted path heuristic (Section 3.6) have been implemented in C++ using the Concert library of IBM ILOG CPLEX 12.5. In particular, the rounding heuristics and the cutting plane separation procedures were implemented as CPLEX callbacks.

The branch-and-price algorithm has been implemented using the open-source framework SCIP[1] 3.0.1 [Ach09] developed by ZIB, with IBM ILOG CPLEX 12.5 as the underlying LP solver.

For the separation of GCS inequalities, both in the pricing subproblems and in the branch-and-cut, we identify strongly connected components using the efficient implementation of Tarjan's algorithm in the open-source LEMON Graph Library 1.3 [DJK11]. More details on the separation procedure can be found in Chapter 5.

The heuristic algorithms described in Section 3.6 have been implemented in

[1]CPLEX does not allow column generation within its branch-and-bound, to this date.

C++11 compiled with GCC 4.6.4. It is worth noting that, in this case, we use custom implementations of all the graph algorithms that are involved (Dijikstra-based shortest and widest path, waterfilling).

All the tests have been carried out on an Intel Xeon E5645 @2.40GHz with 16GB of RAM. Each run was limited to a maximum time of 3600 seconds.

6.3 Exact methods

Let us focus first on the exact methods described in Chapter 3. We have seen that the path formulation allows us to impose elementarity of the paths in a rather natural way. Moreover, the column generation algorithm solves the LP relaxation efficiently. However, we will show that using a branch-and-cut algorithm with the separation of the GCS inequalities proves more effective.

6.3.1 Branch-and-price

The path formulation of UFP-MMF, solved via column generation, is guaranteed to provide relaxation bounds that are at least as good as those of the arc formulation, due to the properties of Dantzig-Wolfe reformulations. In Table 6.2, the computational results suggest that the improvement in the quality of the bounds is typically modest: only for some `france`, `geant` and `polska` instances the bound is different. Nevertheless, the column generation approach appears potentially attractive, as it is able to achieve a significant speedup in solving the LP relaxation.

Table 6.2: *Results for the LP relaxation of the arc formulation and the path formulation solved via column generation (% gap, computed w.r.t. the best known solution, and time to optimality in seconds).*

	Arc formulation		Path formulation	
	gap	time	gap	time
`polska`	1.94	1.0	**1.93**	**0.8**
`nobel-us`	2.42	1.4	2.42	**1.3**
`atlanta`	1.64	1.0	1.64	**1.0**
`france`	1.21	19.0	**1.20**	**6.9**
`geant`	2.89	13.4	**2.88**	**7.2**
`germany`	3.66	2519.0	3.66	**772.5**

Even for the instances where the gap is 0, the solution obtained for the linear relaxation at the root node typically contains many fractional values. Then, it is necessary to resort to the full branch-and-price algorithm. During the experiments, several variants have been evaluated. Let us briefly discuss them.

Branching rules: Branching on the λ_{st}^{p} variables is ineffective, as the pricing phase often proposes paths that have already been generated (and branched on), leading to a rapidly growing search tree. On the other hand, our results indicated that an effective branching strategy is to relax the integrality on the path variables λ_p^{st}, and branch on the value of the original arc variables, i.e., $\sum_{p \in P^{st}} \lambda_p^{st} \sigma_{ij}^{pst}$, that, in an optimal solution, must be equal to 0 or 1. Preliminary experiments also showed that imposing a high branching priority on the bottleneck variables y_{ij}^{st} leads to poorer performance; then, we let the solver (SCIP) balance branching on the arc or the bottleneck variables.

Pricing: We have described in Section 3.5.3 how, before solving the ELPP pricing problem as a MILP, we carry out some heuristic steps in an attempt to quickly identify a variable of positive reduced cost. If the heuristics fail, we resort to the MILP solver, where we use an elementary longest path formulation with dynamic generation of GCS subtour elimination constraints. The graphs in the pricing subproblems typically exhibit a fairly high level of sparsity, in the sense that most of the arc weights are 0 (see Table 6.3). For this reason, even when the MILP has

Table 6.3: *Average arc weight sparsity in the pricing subproblems.*

sparsity (% nonzeros)	
`polska`	4.59
`nobel-us`	4.30
`atlanta`	3.65
`france`	3.91
`geant`	2.60
`germany`	1.68

to be solved, the pricing problems are usually solved rather quickly (see computational results in Chapter 5). The convergence of the column generation algorithm is also quite steady, and stabilization was not deemed necessary. Experiments with a smoothing technique similar to the one described in [Nea00] proved ineffective: stabilization sometimes causes mispricing and generally yields harder (more dense) pricing subproblems, resulting in a larger overall computing time even when the convergence rate as a function of the number of iteration is improved (see Figure 6.2).

Column pool initialization: In order to initialize the column generation algorithm, we populate the initial pool of columns by generating a set of initial paths for each $(s, t) \in K$ that are as diverse as possible. This is achieved by repeatedly finding shortest paths from s to t in an appropriately weighted graph. We have observed that the column generation algorithm is not very sensitive to the number

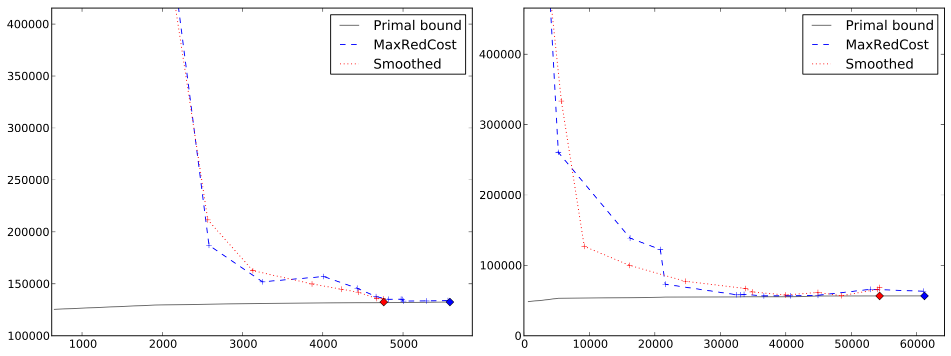


Figure 6.2: *Primal bound and Dantzig-Wolfe dual bound*[2] *vs. number of iterations solving the LP relaxation for instances* `france_7_1` *(left) and* `germany_8_2` *(right). Although the plots show that the convergence rate, in terms of iterations, is better with stabilization (reduced tailing-off effect), for both instances the computing time is larger with smoothing (more than twice for* `germany_8_2`*), since the pricing subproblems are significantly harder.*

of initial paths.

Heuristics: The choice of which heuristics to use and how to combine them requires a fine tuning which is not effortless. Modifying the frequency or the depth where a heuristic is invoked can greatly affect the efficiency of the branch-and-bound search, especially for computationally-heavy heuristics, such as solving the restricted path formulation. According to our experiments, the best configuration is to include the probabilistic path rounding heuristic and the probabilistic shortest-path based heuristic (Section 3.6.1) at each node of the branch-and-bound, while the restricted path formulation (Section 3.6.2) is solved with a small time limit (10 seconds) at the root node and every 50 explored nodes, up to a maximum depth of 9.

In Table 6.4, we summarize the results of the branch-and-price algorithm. We report the average gap at the time limit (3600 seconds), the average computing time to reach optimality and the fraction of solutions which have been certified optimal.

The experiments with the branch-and-price show that, despite solving the LP relaxation more efficiently, on a large fraction of the instances optimality remains out of reach. The algorithm has difficulties in particular on the instances in `geant` and `germany`, where it is often not able to find good solutions. Note that the

[2]The DW dual bound is computed at each iteration as the sum of the current solution value and the value of the maximum reduced cost column (in our case, $\sum_{(s,t)\in K}(\phi^{st} + OPT^{st} - \omega^{st})$, where OPT^{st} is the optimal value of the pricing subproblem for (s,t).

Table 6.4: *Results of the branch-and-price for UFP-MMF (average % gap at the time limit, average time for instances solved to optimality and fraction of optima found).*

	gap	time to opt	opt
`polska`	1.60	281.24	8/24
`nobel-us`	2.17	11.02	4/24
`atlanta`	1.17	36.03	6/16
`france`	1.87	444.12	11/24
`geant`	9.00	5.05	2/18
`germany`	15.15	5.45	2/18

average computing time to optimality is larger for the sets where more instances can be solved: this simply indicates that harder instances can be solved, while on `geant` and `germany` only the smallest-size instances are solved.

Clearly, the fact that the LP bound is not significantly stronger than the one of the arc formulation means that, in a way, a large part of the difficulty of the problem remains in the master, i.e., in the bottleneck constraints. We would have expected the column generation bound to be better, since the pricing subproblem is an elementary longest path problem, which is notoriously hard. However, as we have mentioned, it appears that in practice the UFP-MMF pricing subproblems are quite easy, even after branching. This might explain the LP bounds not being tighter. In the end, exploiting the path formulation structure, although it allows for solving the linear relaxations faster, is not sufficient to outperform a branch-and-cut solving the arc formulation, as we will see in the next paragraph.

6.3.2 Branch-and-cut

Since UFP-MMF is still challenging for the branch-and-price algorithm, let us now consider a branch-and-cut approach for the arc-based formulation.

The arc formulation of UFP-MMF (3.9)–(3.19) requires imposing explicitly the elementarity of the paths for each origin-destination pair (s,t). In Chapter 5, we have discussed mixed-integer programming formulations to prevent subtours in elementary paths problems. Our conclusion was that the extended formulations either give weak linear relaxations, or are too heavy to be competitive with the separation of generalized cutset inequalities (GCS) with the strong component-based separation procedure. Indeed, computational experiments with the different formulations confirm that the cutset inequalities are the best option also for UFP-MMF. Specifically, the extended formulations yield weaker LP relaxations — except for MCF, which gives the same bound of GCS, but is too large to be of practical interest.

A second crucial point of the formulation consists of the MMF conditions themselves. In Chapter 3, we have described how the arc formulation for UFP-MMF can be tightened with valid inequalities derived from the properties of max-min fairness. In Table 6.5 we compare the LP relaxation bound obtained from the basic formulation (3.9)–(3.19) and the ones with the additional valid bounds and inequalities (3.23), (3.27), (3.28), (3.29), (3.31),(3.32), which are all effective in tightening the formulation. The linearized inequalities (3.38), (3.43) and (3.53), along with their linearization constraints, yield some minor improvement in the bound, but do not appear to strengthen the formulation any more than using the inequalities (3.29), (3.31) and (3.32). Hence, the latter (which are fewer) are to be preferred. Table 6.5 also includes the LP bounds obtained adding Constraints (4.20)–(4.26) from the dual-based formulation of UFP-MB, which are valid also for UFP-MMF. The gaps are computed with respect to the best known feasible solution (optimal, when possible).

Table 6.5: *Results for the LP relaxation of UFP-MMF with and without valid inequalities (% gaps computed with the best known feasible solution).* †*Not all instances solved within the time limit.*

	Basic form.		Valid ineq.		Valid + MB dual	
	gap	time	gap	time	gap	time
`polska`	2.57	0.32	1.94	0.96	**1.89**	6.54
`nobel-us`	2.91	0.53	2.42	1.42	**2.40**	17.86
`atlanta`	2.02	0.42	1.64	1.01	**1.61**	13.11
`france`	2.16	8.67	1.21	19.00	**1.19**	162.14
`geant`	4.49	6.25	**2.89**	13.45	**2.89**	570.47
`germany`	11.51	874.49	3.66	2519.02	**2.09**†	2516.98†

The valid inequalities are clearly effective in providing better bounds: the difference is especially significant for `geant` and `germany` instances. Adding the MB dual-based constraints yields an even stronger bound, although with a rather steep increase in computing time.

We are especially interested in a tight formulation if the branch-and-bound search can benefit from it. The impact of the strengthened formulation in a full branch-and-cut algorithm (including the rounding heuristics described in Section 3.6) can be seen in the results summarized in Table 6.6, where we report the average gap and the fraction of instances certified optimal with the three formulations. The average gap at the time limit of the tightened formulation is better for all the topologies. Even more interestingly, the number of instances solved to optimality in 1 hour is significantly larger adopting the valid inequalities. The results with the UFP-MB dual-based valid inequalities are competitive only for the small-size topologies; however, on larger networks the size of the formulation is

Table 6.6: *Results of the branch-and-cut for UFP-MMF with and without valid inequalities.* [†] *We do not report the average gap, since we obtained a finite gap only for a small subset of the instance.*

	Basic form.		Valid ineq.		Valid + dual	
	gap	opt	gap	opt	gap	opt
`polska`	2.86	**8/24**	**1.76**	**8/24**	1.92	**8/24**
`nobel-us`	3.46	4/24	**2.08**	**6/24**	2.78	**6/24**
`atlanta`	2.03	5/16	**1.64**	**7/16**	2.20	**7/16**
`france`	1.75	11/24	**1.63**	**12/24**	7.37	9/24
`geant`	5.09	1/18	**5.06**	**3/18**	–[†]	2/18
`germany`	11.51	2/18	**9.28**	**3/18**	–[†]	2/18

too large to carry out an effective branch-and-bound (note that all the dual-based inequalities have to be added to the formulation, since we do not have a separation procedure for them).

We have observed that even the simple rounding heuristics are essential, since, in practice, off-the-shelf solvers have serious difficulties in finding feasible solutions for UFP-MMF. The effect of disabling them is summarized in Table 6.7, where we compare our branch-and-cut results with those obtained disabling the rounding heuristics, using only CPLEX's default ones. We report the fraction of instances where a feasible solution is found and the instances solved to optimality. Notice how, for a remarkable number of instances, CPLEX's default heuristics are not able to find a single feasible solution within the time limit of 1 hour.

Table 6.7: *Results of the branch-and-cut for UFP-MMF with and without rounding heuristics (fraction of instances where a feasible (optimal) solution was found). In both cases, all CPLEX's default heuristics are enabled.*

	CPLEX default		Rounding heuristics	
	feasible	opt	feasible	opt
`polska`	13/24	7/24	**24/24**	**8/24**
`nobel-us`	12/24	6/24	**24/24**	6/24
`atlanta`	12/16	7/16	**16/16**	7/16
`france`	14/24	11/24	**24/24**	**12/24**
`geant`	4/18	3/18	**18/18**	3/18
`germany`	3/18	1/18	**18/18**	**3/18**

The results summarized in Table 6.6 and 6.7 show that, even using GCS inequalities, strengthening the formulation with valid inequalities, and using ad-hoc rounding heuristics, it is not easy to close the gap. For a small-to-medium number of origin-destination pairs, UFP-MMF is typically solved to optimality in a few seconds for all the topologies. However, when the cardinality of K grows, in particular for the largest networks `geant` and `germany`, the size of the LP relaxation and the time spent by CPLEX generating cutting planes do not allow the exploration of more than a handful of nodes within the time limit (1 hour), with an obvious deterioration of the results, in particular concerning the quality of the solutions found.

Finally, let us compare side by side the results obtained with the branch-and-cut and the branch-and-price algorithm, summarized in Table 6.8. It is worth noting that the branch-and-price relies on the open-source solver SCIP. We have observed that the branch-and-cut of CPLEX is significantly more efficient than the one of SCIP[3], so we do not exclude that, if it were possible to integrate the column generation algorithm within CPLEX's branch-and-bound, the branch-and-price results could be significantly improved. So far, it is clear that the branch-and-price algorithm is not superior to the branch-and-cut approach with separation of GCS inequalities. Nevertheless, both approaches encounter difficulties especially in finding good solutions.

Table 6.8: *Comparison of the branch-and-price and the branch-and-cut results for UFP-MMF (from Table 6.4 and 6.6).*

	Branch-and-price		Branch-and-cut	
	gap	opt	gap	opt
`polska`	**1.60**	**8/24**	1.76	**8/24**
`nobel-us`	2.17	4/24	**2.08**	**6/24**
`atlanta`	**1.17**	6/16	1.64	**7/16**
`geant`	1.87	11/24	**1.63**	**12/24**
`france`	9.00	2/18	**5.06**	**3/18**
`germany`	15.15	2/18	**9.28**	**3/18**

6.4 Finding primal solutions: heuristics

As we have seen, UFP-MMF is very challenging to solve only with MIP-based exact methods. In particular, while we have observed that the bounds obtained with the branch-and-bound approaches are of good quality, finding the optimal solutions appears to be very difficult. Heuristics can play an essential role in providing good, and possibly optimal, solutions in reasonably short computing time. Let us summarize computational experiments with standalone heuristic approaches,

[3]According to the benchmarks by H. Mittelman (`http://plato.asu.edu/ftp/milpc.html`) on 87 instances of the MIPLIB2010, CPLEX 12.6 is, on average, roughly 5 times faster than SCIP 3.1.0

described in Section 3.6: the restricted path formulation, the randomized greedy heuristic and the local search with variable neighborhood exploration.

A remark is in order: all the gaps reported in this section are computed with respect to the optimal value, when available, or to the best known upper bound.

6.4.1 Restricted path formulation

A first, simple approach to UFP-MMF consists of solving the path formulation with a restricted set of path variables that are generated a priori. Clearly, this is only a heuristic, since there is no guarantee that a path not included in the formulation would not improve the solution. This approach was originally proposed in [ACCG13].

In order to cover the set of all possible origin-destination paths P^{st} in the best possible way, we try to construct a subset of paths that are as diverse as possible. For each (s,t), we generate paths by repeatedly finding shortest paths from s to t in a copy of the graph G where the arcs are assigned a weight $w_{ij} \geq 0$. The arc weights are initialized to a random nonnegative value; then, each time a new path is generated, the cost of the arcs therein contained is increased, thus promoting diversity among the paths. More specifically, for each (s,t) pair in K, the number of paths we generate is proportional (by a factor ω) to the minimum (s,t)-cut in the graph with unit arc capacities, i.e., the number of edge-disjoint paths connecting s and t. We adopt this value, that we indicate with γ^{st}, as a proxy for the total number of distinct paths between s and t: if a (s,t) pair has more connecting edge-disjoint paths, we assume that there will be more alternative paths — thus we generate more of them. In particular, for our tests we generate a number of paths for each OD pair which is equal to $\omega\gamma^{st}$, with $\omega = 2$. Then, the number of path variables in the formulation will be $\sum_{(s,t)\in K} \omega\gamma^{st}$. Table 6.9 reports the gap

Table 6.9: *Results for UFP-MMF with the restricted path formulation, with $\omega = 2$, solved with CPLEX (average % gap with respect to best known upper bound, average computing time and fraction of optimal solutions).*

	gap	time	opt
polska	1.64	1150.86	10/24
nobel-us	1.40	1140.94	9/24
atlanta	3.39	1073.72	3/16
france	5.81	987.60	6/24
geant	4.16	1637.35	2/18
germany	11.32	1146.00	2/18

with respect to the best known UFP-MMF upper bound, the average computing time (we set a time limit of 3600 seconds) and the number of solutions known to be optimal. The restricted path formulation gives rather good solutions when the topologies are small: on `nobel-us` and `polska`, that have a small number of arcs and, therefore, a small number of alternative (s,t)-paths, the solutions are less than 2% from the optimum. On the other hand, the approach seems ineffective on the largest topologies, in particular for `germany`, where the average gap is more than 10%. To give an idea of the size of the full set P^{st} for a given (s,t) pair, the number of all possible elementary paths connecting two nodes s and t in the smallest network (`polska`) is, on average, around 50, while, for `germany`, it can be well over 50 millions[4].

Note that the MILPs require a rather large computing time, even with $\omega = 2$, and on the hardest instances the convergence of the algorithm is often not reached within the time limit. For this reason, increasing the factor ω does not appear to be a viable option.

6.4.2 Randomized greedy heuristic

Let us consider the multistart randomized greedy algorithm described in Section 3.6 (Algorithm 3.2). The solutions obtained with this simple heuristic are typically around 5% from the optimal value in below 2 minutes of computing time; we do not report detailed results here for sake of brevity. Extending the algorithm with an intensification phase (Algorithm 3.3) further improves the quality of the solutions. Let us recall that the idea is to adopt a *partial* randomization when we find a solution which improves upon the global best. This phase requires tuning two parameters controlling the tradeoff between intensification and diversification: Reuse_percentage, the fraction of the paths from the best solution that are retained during the intensification phase (i.e., how many of them are fixed in the greedy algorithm), and Nreuse, the number of local search attempts in the neighborhood of such solution. Table 6.10 summarizes the procedure that was carried out to identify the best parameters (with 5000 iterations).

Table 6.10: *Results for the multistart randomized algorithm with intensification phase, for different values of the parameters* Reuse_percentage *and* Nreuse *(*MaxIt = 5000*).*

Reuse_perc.	0.3		0.5		0.7		0.9	
Nreuse	gap	opt	gap	opt	gap	opt	gap	opt
20	4.89	**27/124**	4.78	25/124	4.86	24/124	5.16	23/124
30	4.97	22/124	4.76	**27/124**	4.72	26/124	5.03	23/124
50	4.97	26/124	4.79	24/124	4.78	25/124	5.06	26/124
100	4.76	**27/124**	$\boxed{\textbf{4.66}}$	$\boxed{\textbf{27/124}}$	4.78	25/124	5.13	25/124

[4]Elementary paths computed with the function `all_simple_paths` of the Python library NetworkX [HSS08]. The computation, for `germany`, was stopped after 30 minutes.

In Table 6.11 we report additional results for the settings $\textsc{Reuse\_percentage} = 0.5$, $\textsc{Nreuse} = 100$, which are the best among those we tested.

Table 6.11: *Results for UFP-MMF with the multistart randomized algorithm and intensification phase with* $\textsc{Reuse\_percentage} = 0.5$, $\textsc{Nreuse} = 100$.

MaxIt	10000			15000		
	gap	time	opt	gap	time	opt
polska	4.42	40.8(88.2)	7/24	4.26	56.2(118.8)	7/24
nobel-us	4.18	65.8(106.9)	3/24	4.12	76.9(152.1)	3/24
atlanta	2.70	47.6(71.4)	6/16	2.57	56.1(92.6)	6/16
france	2.06	47.6(131.2)	11/24	2.03	52.7(143.2)	12/24
geant	6.79	148.6(221.9)	3/18	6.49	176.5(318.5)	3/18
germany	3.35	157.1(289.0)	4/18	3.21	169.8(376.9)	4/18
total	3.88	80.4(146.5)	34/124	3.76	93.5(193.0)	35/124

With 15000 iterations, the algorithm usually reaches solutions less than 4% from the optimum within 5 minutes, and roughly 25% of them are guaranteed to be optimal. On `france` instances the algorithm finds half of the known optima, while `geant` instances are the hardest ones. The heuristic appears to be useful even for the largest-size instances of `germany`, that we have seen to be extremely hard to solve with MILP-based methods.

Note that, compared to the restricted path formulation, the randomized algorithm is substantially more attractive from a computational point of view: it is able to find solutions of comparable quality (on some instances, even superior) in far shorter computing times.

6.4.3 Local search with variable neighborhood and tabu list

In Section 3.6.4 we have introduced a local search algorithms with variable neighborhood and tabu list. We recall that the algorithm consists in a hill climbing algorithm, where we also attempt to increase the size of the neighborhood (that depends on the parameters m and s) when we are stuck in a local optimum. When the neighborhood has reached the maximum allowed size, we accept also a number of non-improving moves while keeping a tabu list to avoid cycling.

The algorithm requires tuning a small number of parameters. The parameter m_0 represents the (initial) number of paths that can differ in neighboring solutions, while s_0 represents the (initial) number of solutions to be sampled from the neighborhood. The increase of the neighborhood size and of the number of sampled solutions is done with a fixed step of size 1 for the parameter m, and 200 for the parameter s. We use a tabu list of length 2, since longer cycles are unlikely to occur. We set the maximum number of non-improving moves to 10. We carried out experiments to guide the choice of the parameters m_0 and s_0, that control the initial size of the neighborhood, since they appear to be the ones having the larges impact. To determine the best combination of parameters, we first run the local search in a multistart algorithm with 100 iterations.

Table 6.12: *Results for the local search algorithm with variable neighborhood exploration, for different values of the parameters m_0 and s_0 ($\textsc{MaxIt} = 100$).*

m_0	0.3		0.5		0.7		0.9	
s_0	gap	opt	gap	opt	gap	opt	gap	opt
200	0.85	51	0.81	51	0.77	53	1.05	48
400	0.88	49	0.77	52	0.74	53	1.00	49
600	0.85	53	**0.74**	**55**	0.75	55	1.02	48

Table 6.12 shows that, even with a rather small number of restarts, that corresponds to very short overall computing times, the algorithm achieves quite good results: it is able to find solutions of good quality, on average below 1% from the optimal value, in less than 2 minutes. The best combination of parameters appears to be $m_0 = 4$ and $s_0 = 600$. Observe how the average computing time grows with m_0 and s_0 — since for each of the s neighboring solutions we have to compute m shortest paths — but the quality of the solutions is significantly worse for $m_0 = 10$. Let us remark that a larger m corresponds to larger neighborhoods, thus potentially better moves. However, this does not seem to pay off: indeed, it seems better to consider relatively smaller neighborhoods. Since we can explore them more thoroughly, we are able to find improving solutions with higher probability.

In Table 6.13 we report additional results for the best settings, running the algorithm with a time limit of 10 minutes. We report, for each topology, the average time where the best solution was found, the average gap, and the number of certified optima.

Table 6.13: *Results for UFP-MMF with the local search with variable neighborhood exploration for $m_0 = 4$, $s_0 = 600$ and a time limit of 600 seconds.*

	gap	time best	opt
`polska`	0.40	121.0	13/24
`nobel-us`	0.78	140.2	12/24
`atlanta`	0.40	62.0	8/16
`france`	0.57	82.3	14/24
`geant`	0.68	271.2	7/18
`germany`	0.97	151.7	5/18
`total`	**0.63**	**135.9**	**59/124**

In this short computing time, our heuristic find the optimal solution in almost half of the instances, with an average gap of about 0.6%. The maximum gap over all instances is below 5%, and it exceeds 2% in only 10 instances.

6.5 Computing dual bounds: relaxations

In Chapter 4, we have described relaxations of UFP-MMF where the max-min fairness conditions are either dropped or relaxed. We have already mentioned that the relaxations of UFP-MMF can be used to obtain tighter dual bounds than its LP relaxation: as we will see, some of the relaxed problems are considerably easier than the original one, and the bounds we obtain are of good quality.

Max-throughput unsplittable flows (UFP)

The relaxed problem where the max-min fairness constraints are discarded, i.e., what we call the maximum unsplittable flow problem (UFP), proves to be significantly easier to solve as a MILP. In this case, the bottleneck constraints are lifted, and the subtour elimination constraints are also not necessary. With an arc formulation, obtained considering Formulation (3.9)–(3.13), most instances can be solved to optimality in a matter of seconds with CPLEX. Only the set `germany` is still rather challenging, where 3 instances cannot be certified optimal within the time limit of 1 hour. Note that, for this variant, no customized heuristics nor valid inequalities have been developed.

Table 6.14: *Average % gap, computing time and fraction of optimal solutions when solving the arc formulation of UFP with CPLEX (time limit of 3600 seconds).*

UFP	gap	time	opt
`polska`	0.00	0.17	23/23
`nobel-us`	0.00	0.33	24/24
`atlanta`	0.00	0.21	16/16
`france`	0.00	1.61	24/24
`geant`	0.00	36.25	18/18
`germany`	0.21	217.10	15/18

Concerning the quality of the UFP bounds with respect to the optimal values of UFP-MMF, we can see in Table 6.15 that they are rather good, and often better than the LP bounds of Table 6.5, in particular for the instances of `france` and `geant`. Even if the total throughput value is similar, the solutions (i.e., routing paths and flow allocations) for UFP and UFP-MMF are, in general, quite different, as we will see in Section 6.7.

Table 6.15: *Quality of the bounds obtained with UFP for UFP-MMF (average % gap from the best known UFP-MMF solution).*

UFP/UFP-MMF	gap
`polska`	1.90
`nobel-us`	2.19
`atlanta`	1.48
`france`	0.70
`geant`	0.55
`germany`	1.19

Unsplittable flows subject to max-bottleneck fairness (UFP-MB)

UFP-MB is the relaxation where the fairness criterion at the lower level only imposes that the smallest flow allocation is maximized. To solve UFP-MB, in Section 4.2 we have proposed two different formulations: the global-bottleneck (GB) formulation, Eq. (4.1)–(4.5), and the dual-based (DB) one, Eq. (4.20)–(4.26). Computational experiments show that the LP relaxation of the dual-based formulation is always at least as strong as the global-bottleneck one, at the cost of larger computing times. In Table 6.16 we summarize the branch-and-cut results with the two formulations. In both cases, we use GCS inequalities to prevent subtours. The global-bottleneck formulation is typically faster than the dual-based one, and the average gap at the time limit (3600 seconds) is also smaller for GB, despite starting with a slightly weaker LP formulation. There also is a significant difference in the fraction of instances that are solved to optimality, with GB clearly outperforming DB on all datasets except `germany`.

It is clear that, compared to UFP-MMF, the UFP-MB relaxation is substantially more tractable with a branch-and-cut algorithm: UFP-MB can be solved for instances with more than twice as many OD pairs. Even when optimality is not reached within the time limit, the gaps are usually small, with the exception of `germany`, for which the exact methods exhibit the same weaknesses discussed for UFP-MMF.

Concerning the quality of the UFP-MB bounds with respect to the optimal values of UFP-MMF, we can see in Table 6.17 that they are very close. The UFP-MB/UFP-MMF gap is actually 0 for more than half of the instances. Indeed, despite the counter-example described in Section 4.2, we have certified a nonzero UFP-MB/UFP-MMF gap (that is, cases where UFP-MB is solved to optimality and we know a smaller valid upper bound for UFP-MMF) in only 11 of the instances. In several cases, the bound obtained by solving optimally UFP-MB is even better than what we are able to achieve after 1 hour of branch-and-cut on UFP-MMF.

Table 6.16: *Results of the branch-and-cut for UFP-MB with the global-bottleneck formulation (GB) or the dual-based one (DB) (average % gap at the time limit and fraction of optima).*

	GB		DB	
	gap	opt	gap	opt
`polska`	**0.03**	**23/24**	0.49	15/24
`nobel-us`	**0.45**	**20/24**	0.88	12/24
`atlanta`	**0.06**	**14/16**	0.49	12/16
`france`	**0.64**	**15/24**	0.64	13/24
`geant`	**0.73**	**10/18**	1.91	8/18
`germany`	**2.93**	**5/18**	8.11	4/18

Table 6.17: *Quality of the bounds obtained with UFP-MB for UFP-MMF (average % gap from the best known UFP-MMF solution).*

UFP-MB/UFP-MMF	gap
`polska`	0.46
`nobel-us`	0.94
`atlanta`	0.54
`france`	0.55
`geant`	0.59
`germany`	1.28

Unsplittable flows subject to r-MMF (UFP-r-MMF)

Adopting the relaxed max-min fairness r-MMF concept, the bottleneck condition is relaxed by a factor $0 < r < 1$. If the relaxed conditions yield an easier problem, we could solve UFP-r-MMF (possibly with an increasing value of r), so to obtain better bounds and approximate solutions which are closer to feasibility for the original problem. We formulate UFP-r-MMF as a single level MILP as in (3.9)–(3.19), replacing (3.17) with the relaxed Constraint (4.27). The problem appears to be somewhat easier than the original UFP-MMF, as summarized in Table 6.18. The fraction of instances solved to optimality is larger than for UFP-MMF (compare with Table 6.8), but the problem appears to be still rather challenging, even for $r = 0.25$. Note that we did not implement heuristics tailored specifically for UFP-r-MMF.

The quality of the UFP-r-MMF bounds, compared to the optimal values of UFP-MMF, as reported in Table 6.19, is rather good. However, the relaxation is almost as hard as the original problem, and often we are unable to find the optimal value within the time limit, so it has limited applicability.

Table 6.18: *Results of the branch-and-cut for UFP-r-MMF with* $r = 0.25, 0.5, 0.75$.

	$r = 0.25$		$r = 0.5$		$r = 0.75$	
	gap	opt	gap	opt	gap	opt
`polska`	0.69	8/24	1.11	9/24	1.46	8/24
`nobel-us`	0.86	8/24	1.41	8/24	1.81	8/24
`atlanta`	0.40	11/16	0.79	9/16	1.27	7/16
`france`	0.74	15/24	0.84	15/24	1.28	14/24
`geant`	2.67	8/18	2.85	9/18	4.47	5/18
`germany`	6.90	4/18	6.52	4/18	7.62	3/18

Table 6.19: *Quality of the bounds obtained with UFP-r-MMF for UFP-MMF.*

	UFP-r-MMF/UFP-MMF gap		
	$r = 0.25$	$r = 0.5$	$r = 0.75$
`polska`	1.68	1.54	1.39
`nobel-us`	1.99	1.90	1.66
`atlanta`	1.41	1.37	1.31
`france`	0.67	0.63	0.59
`geant`	0.64	0.63	0.64
`germany`	1.69	1.58	1.49

6.6 Bringing together heuristics and dual bounds

The results presented so far have showed that the standalone heuristics can be very effective, and the relaxations provide good dual bounds. Then, it is natural to consider a practical approach to UFP-MMF, that consists in using the local search heuristic to find good solutions, and solving a relaxation (specifically, UFP-MB) to get a tight good bound, in many cases closing the gap.

The detailed results for this approach, which we consider to be the most efficient, are reported in Table 6.20 in the following two pages. For the upper bounding procedure, we report the best upper bound (UB) obtained within the time limit of the branch-and-bound for UFP-MB, the time to optimality, the number of explored nodes and the number of GCS cuts that have been generated. For the local search heuristic, we report the best solution and the time where it has been found. In the rightmost columns, we report the gap computed using the upper and lower bound.

Overall, such upper and lower bounding procedure allows for solving more than 40% of the instances to optimality, with an average gap around 0.65%. In 114 out of 124 we are within 2% of the optimal value, and only 5 instances have a gap larger than 3%.

Notice how the computing time required to solve UFP-MB is highly sensitive to the size of the network and the cardinality of K. All `polska` and `atlanta` instances are solved well within the time limit, except for 3 cases; on the other hand, only five of the `germany` instances are solved to optimality. However, the computing time required to reach optimality can vary significantly even among instances of similar size. Take as an example `france_3_4`, which we have observed to be one of the most consistently hard instances: despite the fact that it only has 15 origin-destination pairs in a network of 25 nodes, we were not able to close the gap (even for UFP-MB) in any of our attempts. This is quite surprising, given that the instance `france_3_3` can be solved in a few seconds, and it only differs for the composition of the set K (both instances have uniform arc capacities).

Table 6.20: *Detailed results of the upper and lower bounding procedure for UFP-MMF.*

		UFP-MB relaxation (3600 s)				Local search (600 s)		
inst_name	k	UB	time	nodes	GCS cuts	LB	time best	gap
polska_2_1	10	42.50	1.4	0	0	42.50	0.35	0.00
polska_2_2	10	39.50	0.4	0	4	39.50	1.63	0.00
polska_2_3	10	9.00	0.1	0	0	9.00	0.11	0.00
polska_2_4	15	9.00	1.6	90	121	9.00	13.35	0.00
polska_3_1	21	90.00	0.6	0	0	90.00	64.29	0.00
polska_3_2	21	86.50	4.6	185	165	86.50	52.11	0.00
polska_3_3	21	13.00	0.7	0	5	13.00	0.49	0.00
polska_3_4	21	10.00	0.9	0	0	10.00	79.18	0.00
polska_4_1	28	71.50	89.6	2583	662	71.50	3.55	0.00
polska_4_2	28	69.00	72.0	3750	503	68.88	2.54	0.18
polska_4_3	28	13.75	314.7	8412	849	13.75	326.51	0.00
polska_4_4	28	11.80	143.6	5562	850	11.80	288.71	0.00

Table 6.20: *Detailed results of the upper and lower bounding procedure for UFP-MMF.*

		UFP-MB relaxation (3600 s)				Local search (600 s)		
`inst_name`	k	UB	time	nodes	GCS cuts	LB	time best	gap
`polska_5_1`	36	82.14	108.8	4986	522	81.86	360.66	0.34
`polska_5_2`	36	83.70	1216.0	35648	691	83.64	3.41	0.08
`polska_5_3`	36	15.56	758.8	23019	1271	15.41	356.00	0.98
`polska_5_4`	36	13.42	311.3	9510	840	13.18	503.54	1.78
`polska_6_1`	42	117.12	232.7	5900	922	116.47	89.49	0.56
`polska_6_2`	42	113.17	307.3	13249	963	112.25	229.64	0.82
`polska_6_3`	42	18.18	183.3	3685	1360	18.00	253.18	1.01
`polska_6_4`	42	14.39	25.6	554	868	14.25	92.97	0.94
`polska_9_1`	50	144.00	62.0	2607	636	141.88	64.35	1.49
`polska_9_2`	50	112.44	3600	75240	789	112.00	4.44	0.39
`polska_9_3`	50	21.80	795.0	11700	2099	21.59	101.79	0.98
`polska_9_4`	50	13.81	645.5	22050	1619	13.74	12.36	0.50
`nobel-us_3_1`	15	63.50	0.4	0	0	63.50	1.17	0.00
`nobel-us_3_2`	15	49.50	6.4	99	170	49.50	1.91	0.00
`nobel-us_3_3`	15	11.00	0.2	0	0	11.00	11.65	0.00
`nobel-us_3_4`	15	10.33	73.9	2935	611	10.33	5.16	0.00
`nobel-us_4_1`	21	75.17	100.7	4187	418	75.00	12.42	0.22
`nobel-us_4_2`	21	85.50	0.6	0	0	85.50	73.98	0.00
`nobel-us_4_3`	21	15.00	1.1	0	3	14.83	3.58	1.12
`nobel-us_4_4`	21	12.00	7.3	176	170	11.93	4.70	0.56
`nobel-us_5_1`	28	105.50	176.4	5151	854	105.25	3.27	0.24
`nobel-us_5_2`	28	100.30	472.1	11169	607	99.83	2.77	0.47
`nobel-us_5_3`	28	17.75	311.4	7684	1409	17.50	12.55	1.43
`nobel-us_5_4`	28	13.43	3080.2	53847	1805	13.33	73.79	0.71
`nobel-us_6_1`	36	91.50	97.7	813	826	90.78	13.64	0.80
`nobel-us_6_2`	36	116.28	3600	111828	1376	116.17	5.74	0.10
`nobel-us_6_3`	36	17.54	3600	29430	2639	16.71	542.16	4.99
`nobel-us_6_4`	36	14.58	3600	39043	2309	14.02	206.01	3.99
`nobel-us_7_1`	42	103.52	260.5	8573	1149	103.40	440.99	0.12
`nobel-us_7_2`	42	126.58	862.6	17815	1353	126.53	296.94	0.04
`nobel-us_7_3`	42	19.29	1629.4	38426	2027	19.24	288.31	0.28
`nobel-us_7_4`	42	15.58	3600	63263	2230	15.35	249.21	1.46
`nobel-us_9_1`	50	83.75	81.7	1540	1406	83.42	490.41	0.40
`nobel-us_9_2`	50	117.88	198.1	2860	989	117.81	75.99	0.05
`nobel-us_9_3`	50	18.03	3600	19876	3349	17.31	546.25	4.18
`nobel-us_9_4`	50	16.71	553.5	12137	2775	16.49	3.22	1.33
`atlanta_1_1`	12	48.50	0.7	0	3	48.50	1.97	0.00
`atlanta_1_2`	12	56.00	0.8	20	72	56.00	0.11	0.00
`atlanta_1_3`	12	9.00	0.5	0	1	9.00	0.10	0.00
`atlanta_1_4`	12	10.00	0.2	0	0	10.00	0.10	0.00
`atlanta_2_1`	20	62.50	0.5	0	0	62.50	24.31	0.00
`atlanta_2_2`	20	62.50	24.6	884	280	62.50	4.64	0.00
`atlanta_2_3`	20	12.00	0.7	0	5	12.00	79.06	0.00
`atlanta_2_4`	20	12.75	328.3	23036	444	12.67	5.10	0.66
`atlanta_3_1`	30	97.21	302.8	11624	575	96.08	115.07	1.18
`atlanta_3_2`	30	83.30	96.7	3607	397	83.00	203.25	0.36
`atlanta_3_3`	30	16.14	495.1	25474	783	16.02	45.58	0.74
`atlanta_3_4`	30	14.82	3600	111414	974	14.55	305.00	1.83
`atlanta_4_1`	42	75.67	144.1	4296	577	75.62	33.30	0.06
`atlanta_4_2`	42	119.50	2.0	0	1	119.20	17.64	0.25
`atlanta_4_3`	42	15.40	595.5	10210	1017	15.22	49.54	1.19
`atlanta_4_4`	42	16.88	3600	91109	922	16.65	107.44	1.37
`france_2_1`	10	52.50	0.8	0	1	52.50	0.30	0.00
`france_2_2`	10	48.50	2.8	0	9	48.50	0.17	0.00
`france_2_3`	10	9.00	0.9	0	8	9.00	0.15	0.00
`france_2_4`	10	10.00	0.1	0	0	10.00	0.18	0.00
`france_3_1`	15	56.50	0.9	0	0	56.50	0.29	0.00

Table 6.20: *Detailed results of the upper and lower bounding procedure for UFP-MMF.*

		UFP-MB relaxation (3600 s)				Local search (600 s)		
inst_name	k	UB	time	nodes	GCS cuts	LB	time best	gap
france_3_2	15	67.00	1.0	0	0	67.00	0.46	0.00
france_3_3	15	15.00	0.2	0	0	15.00	0.23	0.00
france_3_4	15	12.86	3600	43626	1060	12.33	2.88	4.25
france_4_1	21	77.00	1.9	0	8	77.00	8.62	0.00
france_4_2	21	72.00	2.2	0	8	72.00	0.94	0.00
france_4_3	21	15.95	3600	23089	1151	15.75	145.12	1.27
france_4_4	21	15.90	3600	31542	1599	15.60	0.31	1.92
france_5_1	28	113.50	3.8	0	0	113.50	428.46	0.00
france_5_2	28	101.00	2.0	0	0	101.00	5.30	0.00
france_5_3	28	20.00	85.0	361	1153	20.00	8.11	0.00
france_5_4	28	17.89	3600	11872	2143	17.62	43.57	1.50
france_6_1	36	116.00	12.3	0	23	115.88	234.76	0.11
france_6_2	36	95.50	7.3	0	5	95.50	1.31	0.00
france_6_3	36	21.00	12.3	0	6	21.00	26.90	0.00
france_6_4	36	21.97	3170.7	22759	2389	21.91	469.18	0.28
france_7_1	45	131.41	3600	8093	2200	130.01	81.63	1.07
france_7_2	45	109.93	3600	12284	2862	109.29	105.31	0.59
france_7_3	45	25.42	3600	10920	3914	24.62	242.04	3.23
france_7_4	45	23.95	3600	8309	3574	23.87	169.28	0.37
geant_4_1	12	9.00	0.6	0	0	9.00	0.36	0.00
geant_4_2	12	35.60	0.6	0	1	35.60	0.38	0.00
geant_4_3	12	42.00	3600	11967	1457	41.60	574.35	0.96
geant_5_1	20	11.00	91.2	1000	1235	11.00	28.20	0.00
geant_5_2	20	50.19	3600	12487	1970	49.40	209.18	1.60
geant_5_3	20	52.80	4.6	0	0	52.80	42.45	0.00
geant_6_1	30	14.00	6.3	0	0	14.00	281.74	0.00
geant_6_2	30	74.20	3600	10738	2725	72.71	255.56	2.04
geant_6_3	30	62.20	14.6	0	5	62.11	247.04	0.14
geant_7_1	42	19.00	17.4	0	2	19.00	548.51	0.00
geant_7_2	42	99.20	3600	6845	3034	98.66	336.20	0.55
geant_7_3	42	84.60	1200.2	7000	2494	84.52	58.86	0.09
geant_8_1	56	25.00	3600	7030	6189	24.78	154.98	0.89
geant_8_2	56	123.60	2402.4	9000	3540	123.20	179.88	0.32
geant_8_3	56	111.62	3600	6954	5445	109.31	541.09	2.11
geant_9_1	72	27.00	3600	5504	6566	26.78	471.01	0.81
geant_9_2	72	127.60	1685.5	3000	3951	127.40	349.52	0.16
geant_9_3	72	117.00	3600	4812	4897	113.99	601.65	2.64
germany_5_1	10	8.00	2.2	0	0	8.00	0.36	0.00
germany_5_2	10	31.80	4.3	0	5	31.80	0.76	0.00
germany_5_3	10	42.40	19.6	0	19	42.40	31.14	0.00
germany_6_1	15	10.86	3600	706	4913	10.50	0.63	3.40
germany_6_2	15	38.80	27.6	0	25	38.80	1.73	0.00
germany_6_3	15	54.80	3600	1741	5505	54.20	22.39	1.12
germany_7_1	20	11.84	3600	4941	12707	11.62	21.63	1.87
germany_7_2	20	43.80	40.4	0	75	43.80	4.80	0.00
germany_7_3	20	60.20	3600	1424	6305	59.94	449.60	0.43
germany_8_1	26	14.88	3600	1	2303	14.73	476.34	1.04
germany_8_2	26	56.20	3600	6	1396	55.80	200.00	0.72
germany_8_3	26	70.80	3600	1	1845	70.08	488.07	1.03
germany_9_1	34	17.88	3600	60	3624	17.69	66.31	1.05
germany_9_2	34	65.60	3600	1	2206	65.47	10.95	0.20
germany_9_3	34	80.66	3600	1	1997	78.15	15.88	3.22
germany_10_1	41	18.88	3600	1	1860	18.71	597.41	0.90
germany_10_2	41	70.00	3600	1	1585	69.60	275.18	0.57
germany_10_3	41	88.40	3600	1	1779	86.73	66.79	1.92

6.7 The impact of fair flow allocation

Let us consider the optimal solutions of UFP-MMF we have obtained. We can now answer a crucial question: how much of the maximal achievable throughput on a network can be reached, if fairness is imposed?

Proposition 2.14, on the price of fairness of UFP-MMF, states that having a fair flow allocation over the chosen routing paths can affect the efficiency of the solution in a significant way: with 20 origin-destination pairs, it is possible to lose even more than 80% of the efficiency with respect to the most efficient (utilitarian) solution. (i.e., the optimal solution of UFP). Despite this theoretical result, our experiments show that, on realistic networks, the total throughput that can be achieved under fair flow allocation is only marginally smaller with respect to the case where the upper-level decision maker can also decide on the flow allocations in the network (UFP). Table 6.21 shows how much we lose with respect to UFP when the flows are allocated fairly. We compare the optimal value of bilevel problems with fairness at the lower level, with the optimal throughput of the UFP solution as the baseline (considered with value 100).

Table 6.21: *Optimal values obtained with fair flow allocation at the lower level, compared to the UFP optimal values on the same instances, where $OPT_{UFP} = 100$.*

	UFP-MB	UFP-r-MMF			UFP-MMF
		$r = 0.25$	$r = 0.5$	$r = 0.75$	$(r = 1)$
`polska`	98.59	99.16	98.60	98.12	97.91
`nobel-us`	98.71	99.24	98.76	98.36	97.95
`atlanta`	99.16	99.63	99.40	98.84	98.67
`france`	99.43	99.47	99.53	98.97	98.91
`geant`	99.75	99.65	99.72	98.55	97.62
`germany`	98.46	99.61	98.89	98.80	98.58

The throughput that could be obtained in a given network is indeed larger if MMF fairness were not imposed, but only by a few percent points: the total throughput of UFP-MMF is, on average, less than 2% from the maximum achievable throughput on the network. The values obtained with the other fairness relaxations are even closer to UFP. In other words, our computational experiments suggest that, in practice, one incurs a small price of fairness. This is good news for the network operator: the message is that, even if fairness is imposed by the network protocols, by solving our bilevel problem it is possible to achieve very efficient solutions — almost as efficient as if fairness were not imposed. We will show that, even if the total throughput value is similar, the solutions (i.e., routing paths and flow allocations) for UFP and UFP-*fair* are not necessarily similar.

6.7.1 Relaxation solutions in networks subject to MMF

Since the optimal objective values for UFP, UFP-MMF and the other relaxations of UFP-MMF are so close to each other, a natural question is whether the solutions (i.e., the optimal routing paths) to the relaxed problems can be used successfully in the MMF setting. If this were the case, we could simply take the optimal routing paths obtained, e.g., with UFP, and use them in a network where flow is allocated according to MMF.

Let us verify, then, how the routing paths obtained for a relaxation of UFP-MMF behave in a network subject to max-min fair flow allocation. It is sufficient to apply the water-filling algorithm to these optimal routing paths, and we can compare the resulting throughput with the optimal value of UFP-MMF (or the tightest known upper bound). Table 6.22 summarizes this comparison.

Table 6.22: *Gap with respect to the optimal UFP-MMF value (or the tightest known upper bound) when MMF flow allocations are recomputed over the optimal routing paths found by solving the other problems.*

	UFP	UFP-MB	UFP-r-MMF		
			$r = 0.25$	$r = 0.5$	$r = 0.75$
`polska`	16.28	3.14	1.48	0.99	0.92
`nobel-us`	16.50	3.21	2.04	1.45	1.43
`atlanta`	17.21	2.43	2.22	1.11	0.69
`france`	10.43	1.66	1.43	1.32	1.44
`geant`	20.64	3.80	6.22	4.61	5.06
`germany`	16.04	7.47	9.56	7.43	7.73

The optimal routing paths for UFP are far from optimal if the flow allocation is imposed to be max-min fair. Indeed, when the flow is allocated fairly over those paths, the solution value is often more than 20% off from the UFP-MMF optimal value, and sometimes even more than 50% far.

The solutions for the other relaxations prove to be better. The routing paths obtained with UFP-MB are quite good also under MMF. This is also true for the solutions to UFP-r-MMF, which get closer to the UFP-MMF optima when r is increased. Notice that the solutions obtained with $r = 0.75$ are sometimes worse that those with smaller values of r because the problems is harder and fewer instances are solved to optimality (see Table 6.18), while, in general, we would expect the relaxation with larger r to provide solutions which are closer to the optimal solutions for UFP-MMF.

In conclusion, it is worth stressing the fact that the solutions of UFP, i.e., maximizing the throughput not considering fairness, are of poor quality in an MMF network: this provides a strong practical motivation for further work on bilevel problems subject to fair flow allocation, since taking into account fairness in the optimization is crucial for the quality of the solutions.

6.7.2 Structure of the solutions

To obtain more insight on the structure of the solutions, let us focus on the flow allocations in the UFP and UFP-MMF optimal solutions. Even if the values of the UFP optimal solutions are quite close to those of UFP-MMF, their structure is significantly different. In Figure 6.3 we show the distribution of the flow allocations in the best known solutions for UFP and UFP-MMF, as a histogram of flow allocation values expressed as percentage of the maximal allocation on each instance, i.e., we normalize the optimal flow allocation vector for each instance as:

$$\underline{\phi}_{\%} := 100 \frac{\underline{\phi}}{\max_{i \in K}\{\phi_i\}}.$$

As an example, the second leftmost bar corresponds to the fraction of OD pairs, over all the optimal solutions, whose flow allocation is larger than 0 and not larger than 10% of the maximal allocation in the solution where they belong. UFP solutions almost have a bimodal distribution, with the majority of the allocations either at 0 or at the maximum, while UFP-MMF solutions have a more balanced distribution, although the sub-maximal allocations tend to be less than 50% of the largest.

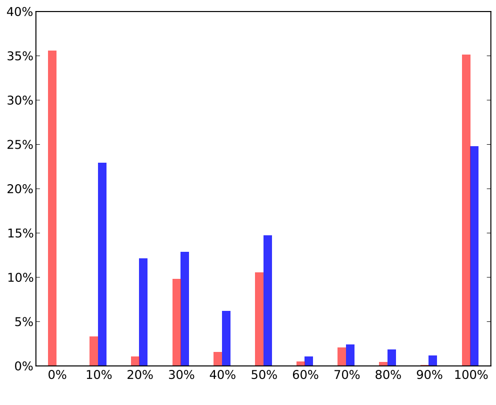


Figure 6.3: *Distribution of the flow allocations for all the origin-destination pairs in the best found (optimal) UFP solution (red) and UFP-MMF solution (blue).*

We have actually observed that on *all* the instances[5], in the optimal solutions of UFP, a subset of the origin-destination pairs are allocated a flow of value 0, and Table 6.23 reports such fraction. Then, in a UFP solution the pairs (s, t) with $\phi^{st} = 0$ are, in a way, irrelevant: they can be assigned an arbitrary routing path, since it does not affect the objective function in any way.

[5]Except those where all the flows can be routed via edge-disjoint paths.

This analysis indicates that the optimal routing paths for UFP can be very different from the optimal paths of UFP-MMF, as the results in the previous subsection also suggested.

Table 6.23: *Percentage of OD pairs with $\phi^{st} = 0$ in the optimal solutions of UFP.*

`polska`	41.18%
`nobel-us`	33.83%
`atlanta`	42.72%
`france`	21.54%
`geant`	28.96%
`germany`	19.55%

6.8 Concluding remarks

In the first part of the thesis, the maximum-throughput unsplittable flow problem subject to max-min fair flow allocation (UFP-MMF) has been studied. After discussing theoretical properties of the problem, we have described how the problem can be cast as a single-level MILP, and proposed several solution approaches.

We have investigated two exact methods: a branch-and-price algorithm, where path variables are dynamically generated, and a branch-and-cut algorithm, based on the separation of generalized cutset inequalities. The computational experiments show that tightening the MILP formulation with valid inequalities, and including rounding heuristics in the branch-and-bound, helps in improving the convergence of the MIP-based algorithms.

Since several instances (due to the size of the network or the number of origin-destination pairs) are very challenging for exact approaches, we have proposed a local search heuristic with variable neighborhood and tabu list that achieves solutions of very good quality (on average, below 0.65% from the optimal value) within 10 minutes.

Several relaxations of UFP-MMF have also been investigated. Solving the relaxation where we replace the MMF criterion with the max-bottleneck (MB) one gives very good dual bounds, that are sufficient to completely close the MIP gap for almost half the instances, exceeding 2% in only 10 instances.

Part II

Operational planning of energy cogeneration systems

CHAPTER 7

Short-term operational planning of cogeneration systems

The second part of the thesis addresses planning problems for energy cogeneration systems, that involve the production of thermal and electrical energy. In this chapter we provide the motivation for this work, introduce the problem of short-term operational planning of cogeneration systems, and include a brief review of previous work on the subject and its relationship with well-known related problems.

7.1 Motivation

Combined Heat and Power (CHP) plants, also called *cogeneration power plants*, are energy systems composed of a network of units that convert primary energy (e.g., fossil fuels) into electricity and useful heat so as to meet the demand of electrical power and heat at certain temperature levels of a set of users. As in a cascade process, primary energy is converted into electrical power through a thermodynamic cycle, and the heat discharged by the cycle is used to satisfy the users' heat demand. Thanks to the improved integration of these heat flows, CHP plants achieve remarkable savings in primary energy and in CO_2 emissions with respect to non-cogeneration plants at both large [Boy10] and small scales [Kol11]. Therefore, several European and North American countries have recently adopted incentive policies to strongly favor CHP plants, as well as Combined Cooling Heat and Power (CCHP) plants, that also cogenerate refrigeration power. As an example, in 2012 the White House issued an executive order on accelerating investment in industrial energy efficiency [EO12]:

> Instead of burning fuel in an on site boiler to produce thermal energy and also purchasing electricity from the grid, a manufacturing facility can use a CHP system to provide both types of energy in one energy efficient step. Accelerating these investments in our Nation's factories can improve the competitiveness of United States manufacturing, lower energy costs, free up future capital for businesses to invest, reduce air pollution, and create jobs.

Due to its practical relevance, the optimization of cogeneration systems has received a growing attention during the last decade. The problems addressed range from long and medium-term strategical and tactical planning, to short-term operational planning of the energy plants.

7.2 Operational planning of cogeneration systems

A cogeneration system is characterized by the presence of units that generate more than one commodity at the same time. A system involves:

- (C)CHP cogeneration units, that simultaneously generate electrical and thermal power,
- generation units, that generate only one commodity (e.g., either electrical or thermal power),
- and heat storage tanks.

The basic version of the short-term cogeneration system operational planning problem we consider is defined as follows. Given

- a cogeneration system, including (C)CHP cogeneration units, with possibly other generation units, and heat storage tanks with fixed capacity,
- time-dependent demands of thermal, cooling and electrical power,
- time-dependent fuel price and fixed costs,
- time-dependent ambient temperatures, which implies time-dependent efficiencies,

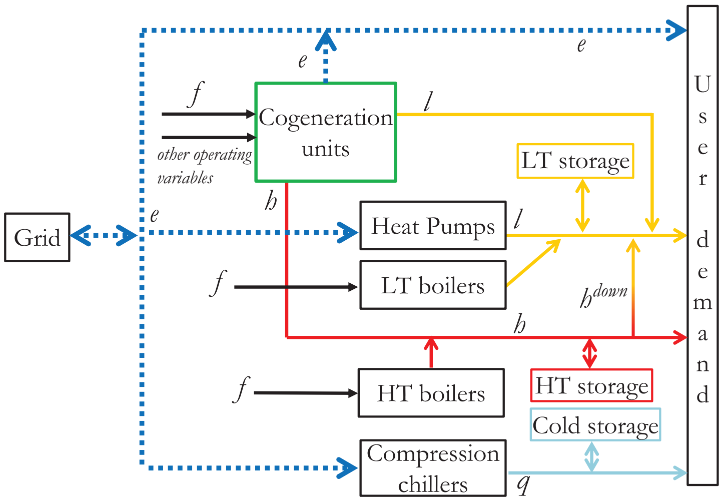


Figure 7.1: *Example of CCHP network connecting the (co)generation units with the storage tanks, the electric grid and the users. Further details will be discussed in Chapter 10, devoted to a real-world application.*

determine, for each time period $t \in \mathcal{T}$, the schedule that minimizes the total operating costs while satisfying the given demands.

In addition, electrical energy can be sold/purchased to/from the power grid: in deregulated markets, the price of electrical energy can vary hourly. Since the different cogeneration units (e.g., multiple CHP gas turbines) can be independently controlled (e.g., switched on/off) and the performance functions of many cogeneration units might be nonlinear, if they incur significant efficiency decrease at partial loads, the short-term operational planning of a cogeneration system can be seen as a mixed-integer nonlinear optimization problem.

We characterize each unit with a linear or nonlinear "performance function", that maps input variables (e.g., fuel or electricity) to output variables (the generated commodities, e.g., thermal and/or electrical power).

The unit performance functions can be time-varying, since the ambient temperature affects performance. We also account for the start-up phase of some units, that may incur a significant energy penalty due to warm-up costs. Heat can be stored in tanks, that may be subject to losses, and higher-temperature heat can be converted into lower-temperature heat. Heat in excess of the demand can be dissipated without additional costs.

Additional logical constraints, with a strong combinatorial aspect, can also be included, typically linking together different consecutive time periods. The most common are:

- Minimum/maximum up/down-time: a unit has to remain on/off for at least/most a given number of (consecutive) time periods.

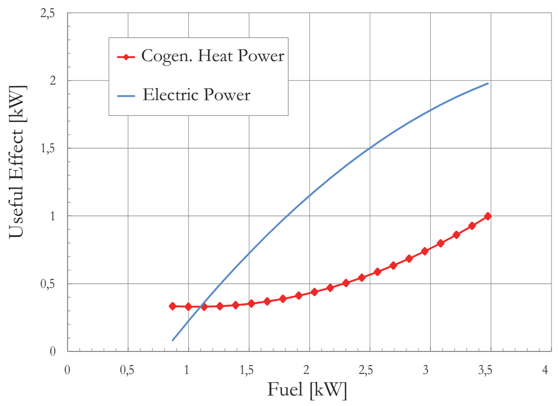


Figure 7.2: *Useful effect (electrical and thermal power) of a cogeneration unit as a function of consumed fuel. Note that the concavity of the curves is different: at larger loads the thermal efficiency increases, while the electrical efficiency decreases.*

- Ramp-up and ramp-down: the increment and decrement of the generation of a unit from a time period to another, is bounded by a constant or, sometimes, it has to follow a specific ramping profile.
- Maximum numbers of start-ups: the number of start-ups can be limited over a specific time horizon (also implicitly imposed by minimum up/down-time constraints).

7.3 Previous and related work

The problem of planning the operations of combined heat and power systems has received a lot of attention in the energy and power systems communities, as well as in the operations research one. The literature on the subject is vast and rather heterogeneous.

In planning the *operations* of a system, we refer to computing the optimal production schedule of the system within a short-term horizon (from a few hours to a few weeks). Typically, the composition of the system is given, and the aim is to operate the units minimizing the overall costs, while satisfying fine-grained technical constraints and users' demands. The optimization problems are often considered to be deterministic, since short-term forecasts tend to be rather accurate, and due to their stochastic counterparts being harder to solve. In recent years, however, the robust optimization approach has received considerable attention, as it allows one to consider uncertainties with a more reasonable tradeoff in computational complexity.

As opposed to the short-term (purely operational) planning, the medium and long-term problems (from months to years) may involve also tactical/strategical decisions on fuel purchase, failure risks, investments on new cogeneration units and the management of polluting emissions. In these longer-term horizons, it is essential to greatly simplify the technical constraints and consider the problem at a higher level. In this thesis, we will focus mainly on the short-term setting. However, some of the considerations can also be applied to a long-term scenario.

From the point of view of the cogeneration system modeling, two main approaches are adopted in the literature. We must point out that there often is no clear cut distinction, and a combination of the two approaches can be used.

In approaches based on thermodynamic first principles, the system is decomposed into small components whose behavior can be modeled accurately by imposing mass/energy balance equations. This requires a rather profound knowledge of the units comprising the cogeneration systems, and often involves the introduction of complex (possibly highly nonlinear) technical constraints. Examples are [DH12] and [MGPA12], where the behavior of each component is described starting from specific thermodynamic relations. In both articles, the nonlinearity in the model are approximated via piecewise linear functions.

In data-driven, or *black-box*, approaches, the behavior of the energy system is described with models obtained from experimental data or from data given by the manufacturer. This is a rather common approach that gives considerable flexibility with respect to the level of accuracy in the description of the system. It is possible to consider an explicit approximation of the performance function of the units in the system, see, e.g., [ZLL$^+$13] and [BTM$^+$14], that consider linear and nonlinear models. An alternative is to consider only the space of the output variables (heat and electrical power), projecting out the input variables (fuel, consumed electricity), either considering linear costs [AM03], or a convex-hull representation in the power-heat-cost space, as done in [LH03] and [CKPT12].

With regard to the mathematical techniques that are used to solve the planning problems, a variety of models and methods have been proposed over the years. A broad survey of related research (up to 2006) is [SP08], which contains a review of different formulations and solution methods, ranging from AI techniques, evolutionary or genetic algorithms, to dynamic programming and mixed-integer programming.

Our aim is to study the short-term operation planning problem adopting mathematical programming techniques. If the discrete unit operation modes are taken into account, one typically has to deal with mixed-integer programming formulations – possibly nonlinear, depending on the performance functions and the technical constraints that are included.

One of the first complete Mixed-Integer Linear Programming (MILP) formulations for the operational planning of cogeneration systems can be found in Seeger and Verstege [SV91]. Their model includes thermal storage with losses, minimum up and down-time constraints and start-up cost. The units are described by means of piecewise linear performance functions. Gardner and Scott Rogers in [GS97] discuss a MILP reformulation for the variant with no energy storage.

In [Dot97, Dot01, DHR99], Dotzauer studies a detailed MINLP model with quadratic costs, which includes thermal storage and explicit temperature/mass balance equations, and proposes approaches based on Lagrangian decomposition. Although his formulation is rather comprehensive, it does not account for electricity demands, and all the generated power is sold to the grid.

A prolific stream of research is the one by Rong, Lahdelma, Makkonen and coauthors, who choose to represent the feasible region as a convex combination of a set of working points. In their early work, they consider a Linear Programming model, where the on/off status of the units is relaxed and thermal storage is not included. For this variant, they propose an efficient revised simplex method [LH03]. Their model is extended to account for non-convex feasible regions (e.g., on/off status or unit modes) in [RL07a, RLG09], where they apply MILP-based heuristics, and in [RL07b], where a convex-envelope branch-and-bound is developed. In all these articles, thermal storage is neglected. In [RHL08], they also account for thermal storage (with losses) and refrigeration power, while the on/off status variables are relaxed, and propose a fast Lagrangian decomposition algorithm to solve the resulting large-scale LPs.

Among other related work, a MILP-based heuristic where the thermal and electrical parts are solved sequentially is proposed in [SFT+05]. Mitra and Grossman in [MGPA12] describe a generalized MILP model for units with more than two operation modes, obtained as a piecewise linear approximation of nonlinear units. Jüdes et al. in [JVT09] propose a MINLP model that includes both the design and the operation of a system, and an ad-hoc branch-and-cut algorithm. In [BTM+14], a complete MINLP formulation is proposed, and a piecewise linear approximation is used to solve it as an MILP.

It is important to observe that short-term operational planning of cogeneration systems can be seen as a combination of two well-known problems, whose literature is worthwhile to investigate.

On the one hand, the discrete on/off status of the cogeneration unit, the logical constraints (such as ramp-up/down rates, minimum up/down-time, etc.), and the production/demand dynamics are typical of the problem usually referred to as Unit Commitment (UC). On the other hand, the management of the storage and of the production level can also be seen as a production planning, or lot-sizing, problem.

7.3.1 Unit commitment

The operational planning of cogeneration systems can be considered as a variant of what is known in the power systems community as the Unit Commitment (UC) problem [Pad04]. UC consists of determining when to start up and shut down the power plants (*unit commitment*), and how much each committed unit should generate to meet the demand (*economic dispatch*), while minimizing a cost function which is typically linear or quadratic with respect of the production level. Successful approaches for UC include dynamic programming for simple subproblems (such as the single-unit variants, see e.g. [FG06]), and Lagrangian methods for more complex, large-scale problems [FGL11, RLL08], while, in recent years, mathematical programming is becoming the method of choice [CA06, FGL09] thanks to the great advances in mixed-integer programming theory and the flexibility it provides by using off-the-shelf MIP solvers as black-box tools.

Let us remark the main peculiarities of the cogeneration system planning problem with respect to UC. Compared to classic unit commitment problems, a CCHP system includes not only the generation of electrical power, but also of other commodities, such as thermal power (at different temperature levels) and refrigeration power. Cogeneration units produce both electrical power and thermal power simultaneously. Therefore, the cost cannot be considered simply as a quadratic function of the production level, as the commodities are interdependent in a nontrivial way. Their interdependence is important also because commodities can be converted (with some caveats): high-temperature heat can be easily converted to low-temperature heat (not the opposite), electricity can be used to generate thermal power via a heat pump, and so on.

A crucial feature of our problem is also the possibility of storing thermal energy from one time period to the following, which is not typically possible for electrical energy. We also remark that the thermal storage for each commodity is global, in the sense that it can be accessed by multiple units. The presence of the storage constraints, that couple the time periods and the units, make many approaches for UC not easily extended to our case.

7.3.2 Production planning

Production planning is a fundamental problem in operations research and management science. At the core of production planning is the so-called lot-sizing problem. Given an inventory and a sequence of demands for an item q over a finite planning horizon of n discrete periods, the deterministic lot-sizing problem consists of finding the optimal production sequence for q. For each time period t, the production incurs a fixed cost K_t and in costs that are proportional to the production level q_t. The amount produced in period t can be used to satisfy demands in that period or in the following ones, thanks to an inventory, which may have holding costs. The aim is to find a production plan that minimizes the overall production and holding costs, satisfying the demands.

Several variants of the problem exist and have been studied. For an extensive account on deterministic production planning problems and, in particular, on mixed-integer programming approaches, see the excellent book by Pochet and Wolsey [PW06], and the references therein. The basic uncapacitated lot-sizing model is introduced for the first time by Wagner and Whitin in [WW58], that propose a $O(n^2)$ dynamic programming algorithm, later improved to $O(n \log n)$ in [WVHK92]. In [BVRW84b] the authors prove that the (l, S) inequalities fully describe its convex hull. The problem with production upper bounds, usually called capacitated lot-sizing, is in general $\mathcal{NP}$-hard [BY82], but can be solved via dynamic programming in $O(n^3)$ if the capacity is constant [VHW96]. The uncapacitated lot-sizing problem with inventory bounds was shown to be polynomially solvable by Love [Lov73], and a $O(n^2)$ algorithm is described in [AK08]. In [BW01], mixed-integer programming formulations for general lot-sizing problems are discussed. Strong formulations and valid inequalities for the multi-item capacitated lot-sizing are proposed in [BVRW84a].

The core structure of the operation planning problem for cogeneration systems can be seen as a multi-item, multi-machine capacitated lot-sizing problem with bounded inventory and lower bounds on the production. Two main peculiarities with respect to a classic production planning problem are the strong interdependence between the production items and the presence of nonlinear terms. Moreover, energy systems planning may involve several additional constraints, which are common in UC problems but are not usually addressed in the literature on lot-sizing. However, due also to new possibilities of electrical power storage that are emerging, we envisage that more and more often the operation planning of energy systems will need to be addressed as a production planning with bounded inventory, and it is important to investigate exact optimization methods apt for the task.

CHAPTER 8

Mixed-integer programming approaches for operational planning of cogeneration systems

In this chapter, we discuss approaches for the operational planning of cogeneration systems based on mixed-integer programming. In particular, we study the building blocks of the problem from a computational and polyhedral point of view. Starting from known results for general production planning, we attempt to extend them to account for the specificities of our problem. We consider both non-linear and linear variants of the single-unit version, and we discuss polynomial algorithms (when possible), reformulations, and ways to derive valid inequalities. Note that this is meant to be a first step in the investigation of this interesting variant, and it does not contain extensive computational results.

Terminology and notation remarks

From now on, adopting the terminology used in the production planning literature, we will sometimes refer to the (co)generation units as *machines*, to the commodities as *items* and to the (heat) storage as *inventory* or *stock*. Moreover, we will speak interchangeably of generation or *production* and inventory or *holding* costs. We denote by n the cardinality of $T = \{1, \dots, n\}$, and also adopt the following notation to indicate the sum of a parameter d_t over an interval $[k, l]$:

$$d_{kl} = \sum_{i=k}^{l} d_t.$$

8.1 Single-item generation unit

Let us begin with a system containing one generation unit (single-machine problem) that generates a single commodity. We assume that the item can be stored between consecutive periods, e.g., it is some type of thermal energy (cooling or high/low temperature heat). The quantity of thermal energy exceeding the demand can be dissipated with no additional costs.

Let u_t be the inventory in stock at the end of time period t, f_t the input quantity that drives the production (e.g., fuel), q_t the amount of the generated item, s_t the amount that is dissipated, z_t the binary variable that has value 1 if the machine is active in period t, and 0 otherwise, s_t the dissipated quantity of item q, and $g_t(\cdot)$ the nonlinear, nondecreasing performance function that maps the input variable f_t to q_t. Let us denote by d_t the demand to be satisfied in period t. A mixed-integer nonlinear programming formulation is as follows:

$$\min \sum_{t \in T} (c_t f_t + K_t z_t) \tag{8.1}$$

$$u_t + d_t = u_{t-1} + q_t - s_t \qquad t \in T \tag{8.2}$$

$$u_t \leq U \qquad t \in T \tag{8.3}$$

$$z_t F_t^{min} \leq f_t \leq z_t F_t^{max} \qquad t \in T \tag{8.4}$$

$$q_t = z_t g_t(f_t) \qquad t \in T \tag{8.5}$$

$$z_t \in \{0, 1\} \qquad t \in T \tag{8.6}$$

$$q_t, s_t, f_t, u_t \geq 0 \qquad t \in T \tag{8.7}$$

Assume that $u_0 = 0 = u_n$. The objective coefficient $c_t > 0$ is the time-varying cost of the input variable f_t, while K_t represents the fixed cost of generation in time t. U is the maximum storage capacity, and F_t^{min}, F_t^{max} the lower and upper bound on f_t. Notice that all the parameters are time-dependent, except for the inventory upper bound U, which is constant.

Constraints (8.2) are the balance equations that model the inventory and demand dynamics. Constraints (8.4) impose that the consumed fuel lies within the operating range of the generation unit, if and only if the unit is on ($z_t = 1$). Constraints (8.5) link the input variables f_t to the output variable q_t via the performance function if $z_t = 1$. The binary variable in Constraints (8.5) is necessary if $g_t(0) \neq 0$. Since $f_t = 0$ if and only if $z_t = 0$, it is also possible to rewrite (8.5) with a big-M term, obtaining

$$q_t = g_t(f_t) - (1 - z_t) g_t(0),$$

where the rightmost term cancels out if $z_t = 1$.

Observe that the nonnegative slack variable s_t in Constraint (8.2) accounts for the possibility of dissipating part of the generated energy (without any penalty cost), which is one of the peculiarities of the problem. We assume that we can dissipate energy only in the *same* period where it is generated, thus $s_t \leq q_t$. This assumption can be made without loss of generality when the dissipation does not have a cost or its cost is constant over time.

If the functions g_t are nondecreasing and the costs c_t are positive for any $t \in T$, all the variables s_t are going to be 0 in most cases, since, in an optimal solution, there is no incentive to generate more than is strictly necessary. Indeed, a variable s_t can be positive only if the quantity that can be "absorbed" by the system (either via storage or item demand) is smaller than the technical minimum. This might happen if $u_{t-1} + q_t - d_t > U$, violating the stock upper bound, and neither q_t nor u_{t-1} can be decreased. Only in such a case, it may be necessary to generate $q_t = g_t(F_t^{min})$, and dissipate the rest through s_t.

Proposition 8.1. *Assume $c_t > 0$, g_t nondecreasing and $s_t \leq q_t$ $\forall t \in T$. For any $t \in T$ it holds that:*

$$s_t \leq g_t(F_t^{min}) \qquad \forall t \in T. \tag{8.8}$$

Moreover, in an optimal solution to (8.1)–(8.7), *$\underline{s}$ can have a nonzero element only if there is a $k \in T$ such that $g_k(F_k^{min}) > d_k$.*

Proof. Consider an optimal solution. If $s_t > 0$, then $u_{t-1} + q_t > u_t + d_t$ and $q_t > 0$ from the assumption that $s_t \leq q_t$. If $q_t > g_t(F_t^{min})$, q_t could be decreased, yielding a cheaper solution. Hence $q_t = g_t(F_t^{min})$, and (8.8) follows.

Assume w.l.o.g. k is the only $k \in T$ such that $s_k = \epsilon > 0$, and $q_k = g_k(F_k^{min})$. Let us consider the solution obtained moving the nonzero dissipation from k to the next period, so that $s_k = 0$ and $s_{k+1} = \epsilon$, by increasing u_k of the quantity ϵ. Even in period $k+1$, it must hold that $q_{k+1} = g_{k+1}(F_{k+1}^{min})$, otherwise one could decrease it obtaining a cheaper solution. Then, we can move the ϵ dissipation to $k+2$, and so on, until a period $k' \geq k$ is found where $q_{k'} = g_{k'}(F_{k'}^{min})$ and $u_{k'} = U$, so that we cannot move any further. For such k', since $s_{k'} = \epsilon$, $u_{k'-1} + q_{k'} > u_{k'} + d_{k'}$, therefore $g_{k'}(F_{k'}^{min}) > d_{k'} + U - u_{k'-1} \geq d_{k'}$. The same argument can be applied backwards, up to a $k' \leq k$ such that $u_{k'-1} = 0$, and yielding equivalently $g_{k'}(F_{k'}^{min}) > d_{k'} + u_{k'} - 0 \geq d_{k'}$. □

Observe that, if the variables s_t do not appear in other constraints[1], they can be removed, obtaining a formulation where the balance equation is replaced by the inequality:

$$u_{t-1} + q_t \geq d_t + u_t \qquad t \in T. \tag{8.9}$$

[1]This might happen when there are efficiency/environmental constraints.

In analogy to the formulations usually considered for lot-sizing and unit commitment problems, it is possible to project out the f_t variables and work only in the space of the production variables q_t, moving the nonlinearity to the objective function. Let $Q_t^{min} := g_t(F_t^{min})$ and $Q_t^{max} := g_t(F_t^{max})$, let us assume that the functions g_t are invertible in $[F_t^{min}, F_t^{max}]$ and, for simplicity, that $g_t(0) = 0 \; \forall t \in T$. By defining $\phi_t(\cdot) = c_t g_t^{-1}(\cdot)$, we can write:

$$\min \sum_{t \in T} \phi_t(q_t) + K_t z_t \tag{8.10}$$

$$\begin{aligned}
& u_t + d_t = u_{t-1} + q_t - s_t && t \in T \\
& u_t \leq U && t \in T \\
& z_t Q_t^{min} \leq q_t \leq z_t Q_t^{max} && t \in T \\
& z_t \in \{0, 1\} && t \in T \\
& q_t, s_t, u_t \geq 0 && t \in T \qquad (8.11)
\end{aligned}$$

All the constraints are linear. Thus, if the functions ϕ_t are convex, the problem is a so-called convex mixed-integer nonlinear program. In this case, tighter formulations that are known for quadratic Unit Commitment problems can be used, for instance exploiting perspective formulations cuts [GL10, FGL09]. On the other hand, if the functions ϕ_t are concave, we will see in Section 8.1.3 that the variant with constant-capacity production bounds can be solved via a dynamic programming algorithm.

Unit Commitment-like constraints

In addition to the constraints related to the balance of the generated quantities, other technical constraints are often included to model the behavior and the limitations of the cogeneration units. Let us introduce, for each $t \in T$, the binary start-up variable δ_t, that takes value 1 if $z_{t-1} = 0$ and $z_t = 1$, indicating the time periods where the unit is started. We model them with the following linear constraints:

$$\delta_t \geq z_t - z_{t-1} \qquad t \in T \tag{8.12}$$

$$\delta_t \leq z_t \qquad t \in T \tag{8.13}$$

$$\delta_t \leq 1 - z_{t-1} \qquad t \in T \tag{8.14}$$

$$\delta_t \in \{0, 1\} \qquad t \in T \tag{8.15}$$

Then, common Unit Commitment-like logical constraints are the following:

$$\begin{aligned}
& L\delta_t \leq \sum_{i \in [t, t+L-1]} z_t && t \in T \text{ (min uptime)} \\
& \sum_{t \in T} \delta_t \leq \bar{\Delta} && \text{(max start-ups)} \\
& q_{t+k} \leq \delta_t M_k && t \in T, k \in [0, K] \text{ (ramp-up profile)}
\end{aligned}$$

where L is the minimum uptime, Δ the maximum start-ups, and M_k for $k = 0, \dots, K$ is the bound on the production level at the k-th period after a start-up. Similarly, one can also model the shut-down of a machine and the minimum/maximum downtime constraints.

8.1.1 Linear variant

In practice, it is sometimes preferable to work with linear approximations of the real performance functions[2]. The resulting problem is naturally simpler to solve, and the loss of accuracy may be acceptable, if the nonlinearities are mild or if one is dealing with longer-term planning problems.

Let g_t be a linear function for all t, that is, $g_t(f_t) = \alpha_t f_t$. A MILP formulation reads:

$$\begin{aligned}
\min \sum_{t \in T} & c_t f_t + K_t z_t && \\
& u_t + d_t = u_{t-1} + q_t - s_t && t \in T \\
& u_t \le U && t \in T \\
& z_t F_t^{min} \le f_t \le z_t F_t^{max} && t \in T \\
& q_t = \alpha_t f_t && t \in T \\
& z_t \in \{0,1\} && t \in T \\
& q_t, s_t, u_t \ge 0 && t \in T.
\end{aligned}$$

Let us project out f_t applying the substitution $f_t = \frac{q_t}{\alpha_t}$. The new objective function reads:

$$\min \sum_{t \in T} c_t \frac{q_t}{\alpha_t} + K_t z_t,$$

that can be rewritten defining $c'_t := \frac{c_t}{\alpha_t}$, so that without loss of generality we can focus on the case with $\alpha_t = 1$. Then, let $Q_t^{min} := \alpha_t F_t^{min}$ and $Q_t^{max} := \alpha_t F_t^{max}$. The formulation in the space of the production variables q_t is as follows:

$$\begin{aligned}
\min \sum_{t \in T} & c'_t q_t + K_t z_t && && (8.16) \\
& u_t + d_t = u_{t-1} + q_t - s_t && t \in T && (8.17) \\
& u_t \le U && t \in T && (8.18) \\
& z_t Q_t^{min} \le q_t \le z_t Q_t^{max} && t \in T && (8.19) \\
& z_t \in \{0,1\} && t \in T && (8.20) \\
& q_t, s_t, u_t \ge 0 && t \in T. && (8.21)
\end{aligned}$$

[2] Of course, even the nonlinear performance functions can only be an approximation of the real behavior of a unit.

This formulation is essentially identical to the standard model for the single-item capacitated lot-sizing problem with inventory upper bound U and production lower bounds Q_t^{min}, although there is the possibility of dissipating item q_t via the slack variable s_t. This allows us to adapt valid inequalities and extended formulations for standard lot-sizing problems.

8.1.2 An extended formulation and valid inequalities

A classic facility-location-based reformulation for the single-item lot-sizing problem [PW06] can be obtained considering the amount generated in each $k \leq t$ to satisfy the demand in period t, that we indicate with the nonnegative variable o_k^t. The extended formulation, adapted to our problem, reads:

$$\min \sum_{t \in T} c_t q_t + K_t z_t \tag{8.22}$$

$$u_t + d_t = u_{t-1} + q_t - s_t \qquad t \in T \tag{8.23}$$

$$\sum_{k=1}^{t} o_k^t = d_t \qquad t \in T \tag{8.24}$$

$$\sum_{t=k}^{n} o_k^t = q_k - s_k \qquad k \in T \tag{8.25}$$

$$o_k^t \leq d_t z_k \qquad k,t \in T, k \leq t \tag{8.26}$$

$$u_t \leq U \qquad t \in T \tag{8.27}$$

$$z_t Q_t^{min} \leq q_t \leq z_t Q_t^{max} \qquad t \in T \tag{8.28}$$

$$z_t \in \{0,1\} \qquad t \in T \tag{8.29}$$

$$q_t, s_t, u_t \geq 0 \qquad t \in T, \tag{8.30}$$

where the balance equation (8.23) can also be omitted.

Several valid inequalities can also be derived. Consider an interval $[k,t]$. Clearly, if no generation occurs in the interval, all the demand must fulfilled by the inventory. It follows that:

$$d_{kt} \leq M \sum_{i=k}^{t} z_i + U,$$

which is equivalent to imposing for each interval (k,t) such that $d_{kt} > U$:

$$\sum_{i=k}^{t} z_i \geq 1. \tag{8.31}$$

Similarly, it is easy to see that the amount which is generated and not dissipated, i.e., $q_t - s_t$, cannot be greater than the sum of the demand in the current period and the capacity of the inventory:

$$q_t - s_t \leq (d_t + U) z_t. \tag{8.32}$$

The (l, S) inequalities are a class of valid inequalities for the lot-sizing problem first described in [BVRW84b]. For the uncapacitated lot-sizing, they suffice to describe the convex hull of the polyhedron. They can be easily extended to the case we consider as follows:

Proposition 8.2. *Let* $1 \leq l \leq n$*,* $L = \{1, \dots, l\}$ *and* $S \subseteq L$*, then the following inequality is valid for Formulation* (8.16)–(8.21)*:*

$$\sum_{j \in S}(q_j - s_j) \leq \sum_{j \in S} d_{jl} z_j + u_l. \tag{8.33}$$

Proof. Consider a point $(\underline{u}, \underline{z})$. If $\sum_{j \in S} z_j = 0$, then $q_j = 0$ and $s_j = 0$ for $j \in S$, and the inequality is satisfied. Otherwise, let $b = \min\{j \in S : z_j = 1\}$. Then

$$\sum_{j \in S}(q_j - s_j) \leq \sum_{j=b}^{l}(q_j - s_j) \leq d_{bl} + u_l \leq \sum_{j \in S} d_{jl} z_j + u_l.$$

□

A similar straightforward adaptation of the two valid inequalities proposed in [AK05] (for lot-sizing with inventory bounds) leads to:

Proposition 8.3. *Given an interval* $L = [k, l]$ *and* $S \subseteq L$*, then the following inequalities:*

$$u_{k-1} + \sum_{j \in S}(q_j - s_j) \leq \sum_{j \in S} d_{kj} z_j + U \tag{8.34}$$

$$u_{k-1} + \sum_{j \in S}(q_j - s_j) \leq \sum_{j \in S} \min\{d_{kj}, d_{kl} - U, d_{jl}\} z_j + U + u_l \tag{8.35}$$

are valid for Formulation (8.16)–(8.21)*.*

One can also consider, for a $[k, l]$ interval, the set defined by:

$$u_{k-1} + \sum_{j=k}^{l} a_j z_j \geq d_{kl},$$

where $a_j = \min\{Q_j^{max}, d_{jl}\}$. Let us define the complemented variables $\bar{z}_t = 1 - z_t$. Then, we can work on the so-called continuous 0-1 knapsack set:

$$\sum_{j=k}^{l} a_j \bar{z}_j \leq \sum_{j=k}^{l} a_j - d_{kl} + u_{k-1},$$

that has been studied in [MW99] and [MNS00], where, given a cover $C \subseteq T$, facet-defining MIR inequalities are proposed.

In presence of Unit Commitment-like logical constraints, it is also possible to tighten the formulation with valid inequalities that have been studied for the Unit Commitment problem. For example, the work in [OAV12], [DKKRA13] and [MELR13] contains polyhedral studies of the ramping constraints, while [HOO09], [LLM04] and [RT05] investigate the minimum up/downtime polytopes.

Numerical example

To give an idea of the effectiveness of the reformulation and the valid inequalities, we report some computational results on a small single-item, single-machine instance with $n = 7$ time periods.

In Table 8.1, we report the linear programming bound and the percentage gap obtained with the nominal formulation (8.16)–(8.21), and the extended formulation (8.22)–(8.30), and activating different classes of valid inequalities, namely: the two simple valid inequalities (8.31) and (8.32), the (l, S) inequalities (8.33), and the valid inequalities (8.34) and (8.35). The (l, S) inequalities (8.33) and the inequalities (8.34)–(8.35) are exponentially many, but the size of the example makes possible generating all of them a priori. The numerical results suggest that the

nominal form.	extended form.	(8.31)+(8.32)	(8.33)	(8.34)+(8.35)	bound	gap
×					80.21	4.51
	×				81.26	3.26
×		×			80.43	4.25
×			×		81.26	3.26
×				×	80.43	4.25
	×	×			81.40	3.10
	×		×		81.26	3.26
	×			×	83.40	0.71
×		×	×	×	83.40	0.71
	×	×	×	×	83.40	0.71

Table 8.1: *Linear programming bound and relative gap of different formulations with respect to the optimal value (84.0).*

inequalities can be quite effective in strengthening the formulation, in particular in combination with the extended formulation. In particular, observe that, with the extended formulation, adding inequalities (8.34)+(8.35) is sufficient to obtain the best gap (0.71%), while to obtain the same bound with the original formulation all the valid inequalities have to be added.

8.1.3 Dynamic programming algorithm for problems with constant production bounds and concave costs

Let us consider the following formulation of a variant of the single-item operational planning problem in the space of the production variables $\underline{q}$:

$$\min \sum_{t \in T} \phi_t(q_t) + K_t z_t \tag{8.36}$$

$$u_t + d_t = u_{t-1} + q_t \qquad t \in T \tag{8.37}$$

$$u_t \leq U \qquad t \in T \tag{8.38}$$

$$q_t \leq z_t Q^{max} \qquad t \in T \tag{8.39}$$

$$z_t \in \{0,1\} \qquad t \in T \tag{8.40}$$

$$q_t, u_t \geq 0 \qquad t \in T, \tag{8.41}$$

where the lower production bound on q_t is dropped, and we assume that the upper bound Q^{max} is constant. If we take $Q^{max} := \max_{t \in T}\{Q_t^{max}\}$, this is clearly a relaxation of the complete problem (8.10)–(8.11). Note also that the slack variable s_t is no longer necessary in the balance constraint, since it will always be 0 in an optimal solution.

If ϕ is concave, the problem is shown to be polynomially solvable in [Wol06], due to its equivalence to lot-sizing with delivery time windows. In this section we first describe in detail a $O(n^4)$ dynamic programming algorithm for this variant, which combines the one in [AK08] for the uncapacited lot-sizing with inventory bounds and the algorithm described in [Wol06] for the capacitated lot-sizing without inventory bounds. Then, we show how this algorithm can be extended to account also for constant lower bounds on the production.

2-phase dynamic programming algorithm

To devise a correct dynamic programming algorithm, it is first necessary to describe some theoretical results on the structure of an optimal solution that will guarantee its correctness. Let us begin by extending the definition of regeneration interval, which is a concept widely used in lot-sizing theory.

Definition 8.4. The interval $[k,l]$ is a *U-regeneration interval* for a solution $(\underline{q}, \underline{u}, \underline{z})$ if $u_{k-1} \in \{0,U\}$, $u_l \in \{0,U\}$ and $0 < u_t < U$ for all $t \in [k, l-1]$.

A regeneration interval can be seen as the special case of U-regeneration interval with $u_{k-1} = 0 = u_l$. We represent the four types of U-regeneration intervals by $[k,l]_{ab}$, where $a = u_{k-1} \in \{0,U\}$, $b = u_l \in \{0,U\}$.

Since we can always consider an optimal solution as a sequence of U-regeneration intervals, the idea of the algorithm is, in the first phase, to find optimal solutions for each one of the possible intervals $[k,l]_{ab}$, while, in a second phase, we use these optimal sub-plans to construct the overall optimal value.

Suppose that $[k,l]_{ab}$ is a U-regeneration interval forming part of an optimal solution $(\underline{q},\underline{u},\underline{z})$. We can show that there exists an optimal solution where at most one period p of the U-regeneration interval has $q_p \notin \{0, Q^{max}\}$, and such period is called a *fractional period.* In other words, in all (but one) periods of the $[k,l]$ interval the production is either at 0 or at the upper bound Q^{max}. The proof is along the lines of the one provided in [FK71] for the case without inventory bounds.

Proposition 8.5. *If a feasible solution* $(\underline{q},\underline{u},\underline{z})$ *is an extreme point of the polyhedron defined by the system* (8.37)–(8.41)*, then it consists only of* U*-regeneration intervals such that, for each interval* $[k,l]_{ab}$*, there exists at most one fractional period* $p \in [k,l]$ *with* $0 < q_p < Q_p^{max}$.

Proof. Suppose $(\underline{q},\underline{u},\underline{z})$ is an extreme point, and let us consider a U-regeneration interval $[k,l]_{ab}$ that is part of that solution. Let $\underline{q}_{kl}$ be the vector of size $l-k+1$ containing the elements of $\underline{q}$ in the interval $[k,l]$. Assume that the production sequence $\underline{q}$ contains at least two periods r and v such that $0 < q_r < Q_r^{max}$ and $0 < q_v < Q_v^{max}$. We assume that there are just two such periods, without loss of generality. Let us define a value ϵ:

$$\epsilon = \frac{1}{2}\min\{q_r, Q_r^{max} - q_r, q_v, Q_v^{max} - q_v, \min_{k\le t\le l-1} u_t, \min_{k\le t\le l-1} U_t - u_t\},$$

where we multiply by a factor $\frac{1}{2} < 1$ so that $\epsilon > 0$ by the definition of U-regeneration interval. Let δ_i the vector with a 1 in i-th position and 0 elsewhere. Then, we can define the production sequences: $\underline{q}'_{kl} = \underline{q}_{kl} - \epsilon\delta_r + \epsilon\delta_v$ and $\underline{q}''_{kl} = \underline{q}_{kl} + \epsilon\delta_r - \epsilon\delta_v$. By construction of ϵ, it is easy to verify that $\underline{q}'_{kl}$ and $\underline{q}''_{kl}$ define feasible solutions. However, $\frac{1}{2}(\underline{q}'_{kl} + \underline{q}''_{kl}) = \underline{q}_{kl}$, contradicting the assumption that $\underline{q}$ is an extreme point. The result follows. $\square$

Note that the proposition is valid also for non-constant capacities and inventory bounds.

It is a well known fact that, in a linear program, there is at least an extreme point which is an optimal solution. This is true also if the production cost $c_t q_t$ is replaced by a concave function $\phi_t(q_t)$. Proposition 8.5 allows us to characterize an extreme point as a sequence of $[k,l]_{ab}$ U-regeneration intervals, with initial level a and final level b, so that production is either at 0 or at full capacity for all $t \in T$, except for, at most, one fractional periods in each U-regeneration interval.

More precisely, it is easy to see that the overall production in an optimal solution for an interval $[k,l]_{ab}$ with $a \in \{0,U\}$ and $b \in \{0,U\}$ will have the value $P_{kl}^{ab} = d_{kl} + b - a$. Then, in the constant-capacity case, the value assumed in the fractional period (if any) is the remainder of the integer division P_{kl}^{ab}/Q^{max}:

$$\rho_{kl}^{ab} = P_{kl}^{ab} - \left\lfloor \frac{P_{kl}^{ab}}{Q^{max}} \right\rfloor Q^{max},$$

with $0 \leq \rho_{kl}^{ab} < Q^{max}$, and there will be exactly $\lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor$ periods with production level Q^{max}.

Let us denote with $G_{kl}^{ab}(t,\tau,\delta)$ the value of a minimum cost solution, in a U-regeneration interval $[k,l]_{ab}$, for periods k up to t, during which production occurs τ times at full capacity and $\delta \in \{0,1\}$ times at level ρ_{kl}^{ab}.

To preserve the assumptions, the recursion has to ensure that the inventory lower and upper bounds (0 and U) are never reached within the U-regeneration interval. Hence, if $\tau Q^{max} + \delta\rho_{kl}^{ab} \leq d_{kt} - a$, it is impossible to have $u_t > 0$ for these (τ,δ) values and so we define $G_{kl}^{ab}(t,\tau,\delta) = \infty$. If $\tau Q^{max} + \delta\rho_{kl}^{ab} \geq d_{kt} - a + U$, it is impossible to have $u_t < U$ for these (τ,δ) values and so we define $G_{kl}^{ab}(t,\tau,\delta) = \infty$. If $\tau + \delta > t - k + 1$ for some (τ,δ) values, it is not possible to produce $\tau + \delta$ times in the interval $[k,t]$ and again $G_{kl}^{ab}(t,\tau,\delta) = \infty$. Moreover, for all $\tau < 0$, we set $G_{kl}^{ab}(t,\tau,\delta) = \infty$.

A forward recursion to compute $G_{kl}^{ab}(t,\tau,\delta)$ for the case $\rho_{kl}^{ab} > 0$ is:

$$G_{kl}^{ab}(t,\tau,0) = \begin{cases} \infty & \begin{array}{l}\text{if } \tau Q^{max} \leq d_{kt} - a \\ \text{or } \tau Q^{max} \geq d_{kt} - a + U \\ \text{or } \tau > t - k + 1\end{array} \\ \min \begin{cases} G_{kl}^{ab}(t-1,\tau,0), \\ G_{kl}^{ab}(t-1,\tau-1,0) + K_t + c_t Q^{max} \end{cases} & \text{otherwise} \end{cases}$$

for $t = k,\ldots,l-1$, $\tau = 0,\ldots,\left\lfloor \frac{P_{kl}^{ab}}{Q^{max}} \right\rfloor$, and

$$G_{kl}^{ab}(t,\tau,1) = \begin{cases} \infty & \begin{array}{l}\text{if } \tau Q^{max} + \rho_{kl}^{ab} \leq d_{kt} - a \\ \text{or } \tau Q^{max} + \rho_{kl}^{ab} \geq d_{kt} - a + U \\ \text{or } \tau > t - k\end{array} \\ \min \begin{cases} G_{kl}^{ab}(t-1,\tau,1), \\ G_{kl}^{ab}(t-1,\tau-1,1) + K_t + c_t Q^{max}, \\ G_{kl}^{ab}(t-1,\tau,0) + K_t + c_t \rho_{kl}^{ab} \end{cases} & \text{otherwise} \end{cases}$$

for $t = k,\ldots,l-1$, $\tau = 0,\ldots,\left\lfloor \frac{P_{kl}^{ab}}{Q^{max}} \right\rfloor$.

For $t = l$, the recursion is computed only for $\tau = \lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor$. If $\rho_{kl}^{ab} = 0$, the equation is as follows:

$$G_{kl}^{ab}(l, \lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor, 0) = \min \begin{cases} G_{kl}^{ab}(l-1, \lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor, 0), \\ G_{kl}^{ab}(l-1, \lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor - 1, 0) + K_l + c_l Q^{max} \end{cases}$$

while, if $\rho_{kl}^{ab} > 0$:

$$G_{kl}^{ab}(l, \lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor, 1) = \min \begin{cases} G_{kl}^{ab}(l-1, \lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor, 1), \\ G_{kl}^{ab}(l-1, \lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor - 1, 1) + K_l + c_l Q^{max}, \\ G_{kl}^{ab}(l-1, \lfloor \frac{P_{kl}^{ab}}{Q^{max}} \rfloor, 0) + K_l + c_l \rho_{kl}^{ab} \end{cases} .$$

Starting from:

$$\begin{aligned}
&G^{ab}_{kl}(k,1,0) = K_k + c_k Q^{max} && \text{if } 0 < Q^{max} + a - d_k < U,\\
&G^{ab}_{kl}(k,0,1) = K_k + c_k \rho^{ab}_{kl} && \text{if } 0 < \rho^{ab}_{kl} + a - d_k < U,\\
&G^{ab}_{kl}(k,\tau,\delta) = \infty && \text{otherwise,}
\end{aligned}$$

we evaluate the recursion for increasing values of t and all values of τ, thus computing $\alpha^{ab}_{kl} = G^{ab}_{kl}(l, \left\lfloor \frac{P^{ab}_{kl}}{Q^{max}} \right\rfloor, 1)$, which is the value of a minimum cost solution for the U-regeneration interval $[k,l]_{ab}$ with $\rho^{ab}_{kl} > 0$.

When $\rho^{ab}_{kl} = 0$ and $P^{ab}_{kl} > 0$, it suffices to use the same recursion for $G^{ab}_{kl}(t,\tau,0)$ to calculate $\alpha^{ab}_{kl} = G^{ab}_{kl}(l, \left\lfloor \frac{P^{ab}_{kl}}{Q^{max}} \right\rfloor, 0)$. If $P^{ab}_{kl} < 0$, then we set $\alpha^{ab}_{kl} = \infty$, and if $P^{ab}_{kl} = 0$, it holds trivially that $\alpha^{ab}_{kl} = 0$.

At this point, we have obtained the optimal solutions for all possible U-regeneration intervals $[k,l]_{ab}$. The second phase consists in combining these solutions to find the overall optimal solution. To do so, once the optimal values α^{ab}_{kl} have been computed, it is enough to find the sequence of matching U-regeneration intervals that minimize the overall cost. With matching intervals, we refer to intervals $[k,l]_{ab}$ and $[k',l']_{a'b'}$ such that $k' = l+1$ and $b = a'$, that is, the intervals are consecutive and the final inventory level of the first interval (u_l) is the same as the initial inventory level of the second interval (u_{k-1}). This can be formulated as a shortest-path problem in an directed acyclic graph.

Proposition 8.6. *Let $G = (V,A)$ be a directed graph with the set of nodes $V = \{1_0, 2_0, \dots, (n+1)_0\} \cup \{2_U, 3_U, \dots, n_U\}$. Each node with label t_0, $t \in [1, n+1]$ corresponds to a period t which immediately follows a U-regeneration interval $[k, t-1]_{a0}$ with empty final inventory. Each node with label t_U, $t \in [2,n]$ corresponds to a period t which immediately follows a U-regeneration interval $[k,t-1]_{aU}$ with full final inventory. Since we assume that $u_0 = 0 = u_n$, the graph does not containt the nodes 1_U and $(n+1)_U$. The arc set consists of all the forward arcs from a node k_0 or k_U to all the nodes l_0 and l_U such that $l > k$. The cost on an arc connecting nodes k_a and l_b has weight $\alpha^{ab}_{k,l-1}$ equivalent to the optimal solution for the interval $[k, l-1]_{ab}$. Then, the shortest path on graph G from node 1_0 to $(n+1)_0$ solves the problem.*

The graph is acyclic by construction, and a topological order is trivially obtained by considering the nodes sorted by their index. Then, a shortest path is found in linear time with respect to the cardinality of the arc set of the graph via a simple dynamic programming algorithm, going backwards from the ending node. Algorithm 8.1 describes a simple shortest-path algorithm over the topologically sorted nodes in the graph, where $\sigma(i)_a$ is the value of the shortest path from node i_a to the destination node $(T+1)_0$.

To summarize the overall complexity of the algorithm, in the first phase, the dynamic programming algorithm for each $[k,l]_{ab}$ is $O(T^2)$. There are $4\frac{T(T+1)}{2}$ −

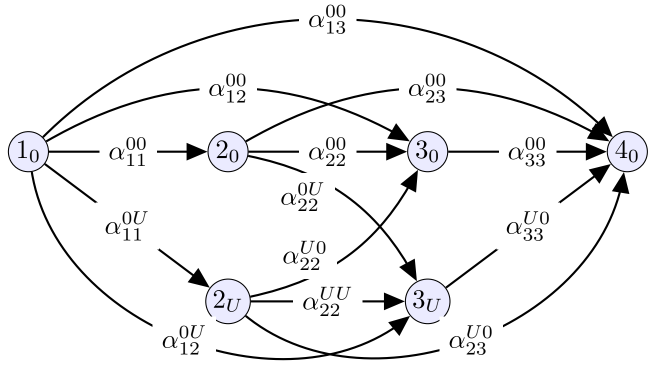


Figure 8.1: *Graph G used in the second phase for an instance with $n = 3$.*

Algorithm 8.1: Dynamic-programming algorithm to determine a shortest path in G.

```
Initialize all $\sigma(i)_a \leftarrow \infty$;
$\sigma(T+1)_0 = 0$;
for $i \leftarrow T \dots 1,\ a \in [0, U]$ do
    for $j \in [i+1, T+1]$ do
        $\sigma(i)_a \leftarrow \min \begin{cases} \sigma(i)_a \\ \sigma(j)_0 + \alpha_{ij-1}^{a0} \\ \sigma(j)_U + \alpha_{ij-1}^{aU} \end{cases}$
    end
end
```

$2T - 2(T-1) - 1$ intervals (arcs in the graph), so the first phase has an overall $O(T^4)$ running time. The second phase has a running time which is linear with the number of arcs. Thus, the overall optimization time to solve to optimality the problem defined in (8.36)–(8.41) is $O(T^4)$. While we have described only a method to find the optimal *value*, in both phases of the algorithm it is easy to augment the dynamic programming updates to also keep track of the structure of the optimal solution.

Finally, we point out that the algorithm can be easily verified to be correct also for the general case with non-constant inventory bounds. Essentially, it is enough to replace all occurrences of U with the appropriately indexed U_t. We have chosen to describe the version with constant inventory bounds to adopt a lighter notation.

Constant lower bounds on production

The algorithm can be extended to the case with constant lower bounds on the production:

$$z_t Q^{min} \leq q_t \leq z_t Q^{max}. \tag{8.42}$$

With the additional assumption that $u_n = 0$[3], Proposition 8.5 is still valid, in this slightly different form:

Proposition 8.7. *If a feasible solution* $(\underline{q}, \underline{u}, \underline{z})$ *is an extreme point of the polyhedron defined by the system* (8.37)–(8.41)*, and Constraint* (8.39) *is replaced with* (8.42)*, then it consists only of U-regeneration intervals such that, for each interval* $[k,l]_{ab}$*, there exists at most one* fractional period $p \in [k,l]$ *with* $Q_p^{min} < q_p < Q_p^{max}$.

The proof is basically the same, obtained by defining $\epsilon = \frac{1}{2}\min\{q_r - Q_r^{min}, Q_r^{max} - q_r, q_v - Q_v^{min}, Q_v^{max} - q_v, \min_{k+1 \leq t \leq l} u_t, \min_{k+1 \leq t \leq l} U_t - u_t\}$. Then, an optimal solution for a U-regeneration interval is characterized as τ_0 periods with production at full capacity, ν_0 periods with production at minimum capacity and 0 or 1 fractional periods. The dynamic programming algorithm can be modified considering the function $G_{kl}^{ab}(t, \tau, \nu, \delta; \rho_{kl}^0)$ that represents the value of a minimum cost solution for period k up to t during which production occurs τ times at Q^{max}, ν times at Q^{min} and δ times at level ρ_{kl}^0.

To compute the optimal solution for a U-regeneration interval α_{kl}^{ab}, one needs to compute $G_{kl}^{ab}(l, \tau_0, \nu_0, \delta_0; \rho_{kl}^0)$ for each feasible triplet $(\tau_0, \nu_0, \delta_0)$ that satisfies $\tau_0 Q^{max} + \nu_0 Q^{min} + \delta_0 \rho_k l = P_{kl}^{ab}$. The optimal value α_{kl}^{ab} will be the minimum among them.

The dynamic programming computation of G_{kl}^{ab} involves an additional variable ν in $[k,l]$, so it increases by a factor $O(n)$, and has to be repeated as many times as there are feasible triplets $(\tau_0, \nu_0, \delta_0)$.

[3]The algorithm can be generalized even to the case with $u_n > 0$, but we omit the details here.

Proposition 8.8. *Given a production quantity P^{ab}_{kl} in an interval $[k,l]$, the number of feasible triplets $(\tau_0, \nu_0, \delta_0)$ with $\tau_0 \in \mathbb{N}$, $\nu_0 \in \mathbb{N}$, $\delta_0 \in \{0,1\}$, such that:*

$$\tau_0 Q^{max} + \nu_0 Q^{min} + \delta_0 \rho = P^{ab}_{kl},$$
$$\tau_0 + \nu_0 + \delta_0 \leq l - k + 1$$

and either $\rho = 0$ or $Q^{min} < \rho < Q^{max}$, is bounded by $O(n)$.

Proof. For each value of ν_0 in $[0, l-k+1]$, consider the equation $\tau_0 Q^{max} + \rho = P^{ab}_{kl} - \nu_0 Q^{min}$. The only possible value for τ_0, with $\rho \leq Q^{max}$, is the result of the integer division:

$$\tau_0 = \left\lfloor \frac{P^{ab}_{kl} - \nu_0 Q^{min}}{Q^{max}} \right\rfloor,$$

and the corresponding production level in the fractional period will be its remainder:

$$\rho = P^{ab}_{kl} - \left\lfloor \frac{P^{ab}_{kl} - \nu_0 Q^{min}}{Q^{max}} \right\rfloor Q^{max}.$$

Then, if $\rho = 0$ ($\delta_0 = 0$) or $\rho \geq Q^{min}$ ($\delta_0 = 1$) and $\tau_0 + \nu_0 + \delta_0 \leq l - k + 1$, the triplet $(\tau_0, \nu_0, \delta_0)$ is feasible. Since there is at most a (τ_0, δ_0) pair for each possible value of ν_0, the number of possible feasible triplets is $O(n)$. □

The inner dynamic programming algorithm is $O(n^3)$, which has to be repeated $O(n)$ times (the number of triplets) for each of the $O(n^2)$ intervals. This leads to an overall computing time of $O(n^6)$ for the first phase. The second phase of the algorithm is unchanged, as the numbers of arcs in the graph is the same, so the bottleneck of the algorithm remains the first phase. Without a doubt, it is a rather large complexity, although polynomial; in practice, we have observed that a state-of-the-art MILP solver is, in most cases, more efficient that a simple implementation of this algorithm. Still, this allows us to state that:

Proposition 8.9. *The lot-sizing problem with concave production costs, inventory bounds and constant upper and lower bounds on production can be solved in polynomial time.*

8.2 Generation of a non-storable item (electrical energy)

The formulations discussed in the previous section model a single generation unit producing a quantity q_t that can be stored from a time period t to the following period $t+1$, such as thermal energy. However, some items typically cannot be stored, e.g., electrical energy, giving rise to a different problem, which, in the single-unit case, is essentially a standard Unit Commitment problem. In this variant, exchange with the power grid is allowed. The single-item problem where the machine only generates a non-storable item can be written as:

$$\min \sum_{t\in T} c_t f_t + K_t z_t + p_t^+ e_t^+ - p_t^- e_t^- \tag{8.43}$$

$$\begin{aligned}
& e_t + e_t^+ - e_t^- = w_t && t \in T\\
& z_t F_t^{min} \le f_t \le z_t F_t^{max} && t \in T\\
& e_t = z_t h_t(f_t) && t \in T\\
& z_t \in \{0,1\} && t \in T\\
& e_t, e_t^+, e_t^-, f_t \ge 0 && t \in T
\end{aligned}$$

The demand w_t can be satisfied by the amount e_t that is generated by the machine and/or by the amount e_t^+ that is purchased externally, e.g., electrical energy bought from the grid. If the production level e_t exceeds the internal demand w_t, it is possible to sell the exceeding amount e_t^- to the grid with price p_t^-. We make the reasonable assumption that the electricity selling price p_t^- is smaller than its purchase cost p_t^+ (essentially, we assume it is an arbitrage-free market). However, it is not true, in general, that the unit production cost is larger than the selling price: in that case, one can gain a profit by producing more than w_t.

Remark: It is possible to assume without loss of generality that

$$h_t(F_t^{min}) \le w_t \le h_t(F_t^{max}) \ \forall t \in T,$$

i.e., any demand w_t can be entirely satisfied (in principles) without selling to or purchasing from the grid. If this were not the case, it is always possible to transform the problem obtaining an equivalent one where the assumption is fulfilled. If $w_t > h_t(F_t^{max})$, one can take $w_t' := h_t(F_t^{max})$ and add to the objective function a constant term $p_t^+(w_t - h_t(F_t^{max}))$, while if $w_t < h_t(F_t^{min})$, one can take $w_t' := h_t(F_t^{min})$ and add to the objective function a constant term $-p_t^-(h_t(F_t^{min}) - w_t)$.

If we do not consider additional Unit Commitment-like constraints that couple consecutive time periods, each period can be solved independently. The optimal value for a given $t \in T$ is obtained either purchasing the whole demand from the grid, hence $z_t = 0$, $e_t = 0$, $e_t^+ = w_t$ and cost $p_t^+ w_t$, or determining the optimal solution to the nonlinear, continuous problem where $z_t = 1$. Assuming that h_t is invertible, let us take $\theta_t(\cdot) := c_t h_t^{-1}(\cdot)$. Let $E_t^{min} = h_t(F_t^{min})$ and $E_t^{max} = h_t(F_t^{max})$, and let us drop the subscript t for the moment. Then, the single-period subproblem, for $z = 1$, thanks to the assumption $p^+ > p^-$, can be written as:

$$\zeta = \min \quad \theta(e) + \max\{p^+(w-e), p^-(w-e)\} \tag{8.44}$$
$$E^{min} \le e \le E^{max} \tag{8.45}$$
$$e^+, e^- \ge 0. \tag{8.46}$$

If θ is concave, the optimal solution will be one of the extreme points of the feasible region, that are: $e = w, e^+ = 0, e^- = 0$; $e = E^{min}, e^+ = w - E^{min}, e^- = 0$; and $e = E^{max}, e^+ = 0, e^- = E^{max} - w$. If θ is convex, one can write KKT optimality conditions.

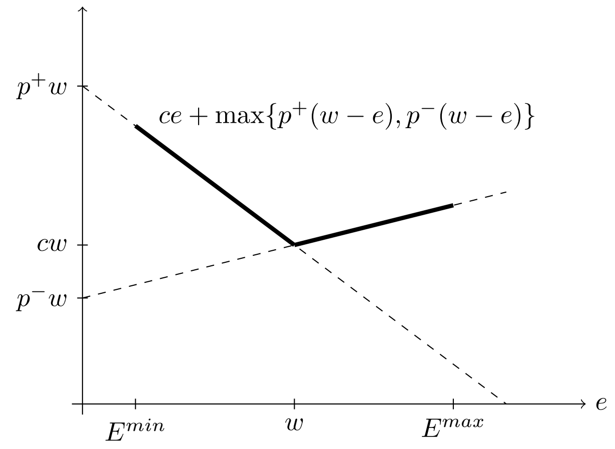


Figure 8.2: *Plot of the objective function of the single-period subproblem when θ is linear with production cost c, and $p^- < c < p^+$.*

The value of the optimal solution to the complete multi-period problem in Formulation (8.43) is then given by:

$$\sum_{t \in T} \min\{K + \zeta_t, p_t^+ w_t\}, \tag{8.47}$$

where ζ_t is the optimal value for the subproblem (8.44)–(8.46) in period t.

Note that, if the Unit Commitment-like logical constraints (minimum uptime, ramp-up rates, ...) are included, the time periods are no longer independent. Then, it is not possible to solve the overall problem by optimizing the single-period subproblems. In this case, we have a single-unit variant of the problems usually studied in the UC literature. Even if we do not discuss them, we point out that fast dynamic programming algorithms exist for several variants of the single-unit UC problem with logical constraints, see e.g. [FG06] for the case of ramp-up constraints.

8.3 Cogeneration unit with production of a storable and a non-storable item

Let us now consider a cogeneration unit, i.e. a machine that generates simultaneously more items. We start by considering the generation of only two different items: an item q (e.g., thermal energy), that can be stored in the inventory, and an item e (e.g., electrical energy), that cannot be stored, but can be purchased from the market (or sold, if in excess). We make the assumption that the electricity selling price p_t^- is smaller than its purchase cost p_t^+ (arbitrage-free market). A mathematical programming formulation is as follows:

$$\min \sum_{t \in T} c_t f_t + K_t z_t + p_t^+ e_t^+ - p_t^- e_t^- \tag{8.48}$$

$$u_t + d_t = u_{t-1} + q_t - s_t \qquad t \in T \tag{8.49}$$

$$e_t + e_t^+ - e_t^- = w_t \qquad t \in T \tag{8.50}$$

$$u_t \leq U \qquad t \in T \tag{8.51}$$

$$z_t F_t^{min} \leq f_t \leq z_t F_t^{max} \qquad t \in T \tag{8.52}$$

$$q_t = z_t g_t(f_t) \qquad t \in T \tag{8.53}$$

$$e_t = z_t h_t(f_t) \qquad t \in T \tag{8.54}$$

$$z_t \in \{0, 1\} \qquad t \in T. \tag{8.55}$$

For any $t \in T$, we assume that g_t and h_t are nondecreasing, and that $g_t(f_t) \geq 0$, $h_t(f_t) \geq 0$ for $f_t \in [F_t^{min}, F_t^{max}]$. The formulation is similar to the single-item models previously described. However, note that, in this case, dissipating a quantity of item q or selling the exceeding amount of e can be necessary more often: given a demand pair (d_t, w_t) to be fulfilled, there might not exist a value of f_t such that the demands are exactly satisfied without either dissipation or external purchase.

Assume that there is no external purchase, $e^+ = 0$, and the inventory levels u_{t-1}, u_t are fixed. To get a feasible solution for period t it is enough to find the smallest γ such that $q_t = g_t(\gamma) \geq d_t + u_t - u_{t-1}$ and $e_t = h_t(\gamma) \geq w_t$. In an optimal solution, at least one of the two inequalities will be tight. If γ is such that $e_t = h_t(\gamma) = w_t$ and $g_t(\gamma) > d_t + u_t - u_{t-1}$, then some q-energy has to be dissipated, and $s_t > 0$. If, viceversa, $q_t = g_t(\gamma) = d_t + u_t - u_{t-1}$ and $e_t = h_t(\gamma) > w_t$, it is possible to sell the exceeding amount $e_t^- > 0$ with a profit.

An alternative option, which might be convenient depending on the price p_t^+ and the marginal cost of generating q_t, is to consider a γ such that $q_t = g_t(\gamma) \geq d_t + u_t - u_{t-1}$ and $e_t = h_t(\gamma) < w_t$, and purchase the remaining quantity $e_t^+ = w_t - e_t > 0$ (with a per-unit cost p_t^+).

If we assume, w.l.o.g., that $h(F_t^{min}) \leq w_t \leq h(F_t^{max})$ (see Section 8.2), we can derive the following simple propositions that characterize optimal solutions under specific assumptions on the production cost of electrical energy. The idea is that, if the per-unit selling price of the electrical energy is larger than its marginal production cost, then it always pays off to generate as much as possible. Moreover, if the revenue obtained selling the amount exceeding the demand is even larger than its overall production cost (including the fixed cost), the net income is positive, thus it is always convenient to produce (in order to sell).

Proposition 8.10. *For a given $t \in T$, if $p_t^- \dfrac{\partial h_t}{\partial f_t}(x) > c_t \ \forall x \in [F_t^{min}, F_t^{max}]$, then in any optimal solution $z_t = 1 \implies f_t = F_t^{max}$.*

Proof. By hypothesis, $z_t = 1$. For $h_t(f_t) \geq w_t$, in any optimal solution $e_t^+ = 0$ and the objective function reduces to $K_t + c_t f_t - p_t^-(h_t(f_t) - w_t)$. Since $p_t^- \dfrac{\partial h_t}{\partial f_t}(x) > c_t \ \forall x \in [F_t^{min}, F_t^{max}]$, the objective function is monotonically decreasing in f_t, and the minimum over $[h_t^{-1}(w_t), F_t^{max}]$ is attained in F_t^{max}, with value $K_t + c_t F_t^{max} - p_t^-(h_t(F_t^{max}) - w_t)$. We need to show that the value for $f_t = F_t^{max}$ is also as good as any solution obtained in the interval $[F_t^{min}, h_t^{-1}(w_t)]$. For $h_t(f_t) \leq w_t$, $e_t^- = 0$ and the objective function will be $K_t + c_t f_t - p_t^+(h_t(f_t) - w_t)$. For any $f_t \in [F_t^{min}, h_t^{-1}(w_t)]$, we have that $K_t + c_t F_t^{max} - p_t^-(h_t(F_t^{max}) - w_t) \leq K_t + c_t f_t - p_t^-(h_t(f_t) - w_t) \leq K_t + c_t f_t - p_t^+(h_t(f_t) - w_t)$, since $p_t^+ > p_t^-$.

It follows that the minimum for $z_t = 1$ is obtained for $f_t = F_t^{max}$. $\square$

Proposition 8.11. *For a given $t \in T$, if $p_t^- \dfrac{\partial h_t}{\partial f_t}(x) > c_t \ \forall x \in [F_t^{min}, F_t^{max}]$ and $p_t^-(h(F_t^{max}) - w_t) > K_t + c_t F_t^{max}$, then in any optimal solution $z_t = 1$ (and $f_t = F_t^{max}$).*

Proof. The minimum for $z_t = 1$ is in $f_t = F_t^{max}$ by Proposition 8.10. Then, if $p_t^-(h(F_t^{max}) - w_t) > K_t + c_t F_t^{max}$, the cost in t is negative if $z_t = 1$. Since if $z_t = 0$ the cost would be $p_t^+ h_t^{-1}(w) \geq 0$, and we assume there are no additional constraints on the binary variables, it is always convenient to produce in t ($z_t = 1$). $\square$

Proposition 8.12. *For a given $t \in T$, if $p_t^+ \dfrac{\partial h_t}{\partial f_t}(x) > c_t \ \forall x \in [F_t^{min}, F_t^{max}]$, then in any optimal solution $z_t = 1 \implies f_t \geq h_t^{-1}(w_t)$.*

Proof. By hypothesis, $z_t = 1$. For $h_t(f_t) \leq w_t$, in any optimal solution $e_t^- = 0$ and the objective function will be $K_t + c_t f_t - p_t^+(h_t(f_t) - w_t)$. Since $p_t^+ \dfrac{\partial h_t}{\partial f_t}(x) > c_t \ \forall x \in [F_t^{min}, F_t^{max}]$, the objective function is monotonically decreasing in f_t, and the minimum over $[F_t^{max}, h_t^{-1}(w_t)]$ is attained in $f_t = h_t^{-1}(w_t)$. It follows that, if $z_t = 1$, $f_t \geq h_t^{-1}(w_t)$. $\square$

Proposition 8.13. *For a given $t \in T$, if $p_t^+ \dfrac{\partial h_t}{\partial f_t}(x) > c_t \ \forall x \in [F_t^{min}, F_t^{max}]$ and $p_t^+ w_t > K_t + c_t h^{-1}(w_t)$, then in any optimal solution $z_t = 1$ (and $f_t \geq h_t^{-1}(w_t)$).*

Proof. The hypothesis implies that producing the whole demand (including the fixed cost K_t), with $z_t = 1$, is cheaper then buying the total demand from the grid, with $z_t = 0$ and cost $p_t^+ w_t$. □

For the linear variant of the two-item case, it is possible to obtain a result similar to the one that allowed us to derive a polynomial dynamic programming algorithm for the constant capacity single-item variant.

Proposition 8.14. *If a feasible solution* $(\underline{q}, \underline{u}, \underline{s}, \underline{e}^+, \underline{e}^-, \underline{z})$ *is an extreme point of the polyhedron defined by the system* (8.49)–(8.55) *(where g and h are linear), then it consists only of U-regeneration intervals such that, for each interval $[k, l]_{ab}$, there exists at most one fractional period $p \in [k, l]$ with $Q_p^{min} < q_p < Q_p^{max}$ and $q_p \neq w_p$. If $p_t^- > c_t$ $\forall t \in [k, l]$, then there are no fractional periods.*

Proof. Similar argument to the one in the proof of Proposition 8.5, with:

$$\epsilon = \frac{1}{2} \min\{q_r - Q_r^{min}, Q_r^{max} - q_r, q_v - Q_v^{min}, Q_v^{max} - q_v, \min_{k+1 \le t \le l} u_t, \min_{k+1 \le t \le l} U_t - u_t, |q_t - w_t|\}.$$

□

However, this characterization is not sufficient, in general, to extend the dynamic programming algorithm: one would need to have not only constant capacities, but also constant demands w_t, which appears to be a rather restrictive assumption. Even if this were the case, we would incur an even larger computational complexity.

8.4 Cogeneration unit with multiple storable and non-storable items

Let us now generalize the formulation to account for a cogeneration unit generating a set I of items with storage (e.g., thermal and refrigeration energy at different temperature levels), and a set J of non-storable items with market exchange. In this section, we discuss some future possible lines of research, namely, a class of valid inequalities and a Lagrangian relaxation.

A complete formulation for the operational planning problem of a cogeneration unit with multiple items can be written as follows:

$$\min \sum_{t \in T} [c_t f_t + K_t z_t + \sum_{j \in J} (p_t^+ e_t^{j+} - p_t^- e_t^{j-})] \tag{8.56}$$

$$u_t^i + d_t^i = u_{t-1}^i + q_t^i - s_t^i \qquad i \in I, t \in T \tag{8.57}$$

$$e_t^j + e_t^{j+} - e_t^{j-} = w_t^j \qquad j \in J, t \in T \tag{8.58}$$

$$u_t^i \le U^i \qquad i \in I, t \in T \tag{8.59}$$

$$z_t F_t^{min} \le f_t \le z_t F_t^{max} \qquad t \in T \tag{8.60}$$

$$q_t^i = z_t g_t^i(f_t) \qquad i \in I, t \in T \tag{8.61}$$

$$e_t^j = z_t h_t^j(f_t) \qquad j \in J, t \in T \tag{8.62}$$

$$z_t \in \{0, 1\} \qquad t \in T, \tag{8.63}$$

where each item has its own inventory u_t^i, but the quantity which is generated in period t is driven by the same variable f_t for all of them, and there is only one global on/off decision z_t. While the assumption of one degree-of-freedom for a simple cogeneration backpressure unit is often fulfilled, to have a model which is as inclusive as possible one can replace Constraints (8.61)–(8.62) with a single nonlinear linking constraint for each t:

$$f_t = z_t \psi_t(q_t^1, q_t^2, \dots, e_t^1, e_t^2, \dots) \qquad t \in T, \tag{8.64}$$

which in the linear case becomes:

$$f_t = \alpha_t^1 q_t^1 + \alpha_t^2 q_t^2 + \dots + \beta_t^1 e_t^1 + \beta_t^2 e_t^2 + \dots \qquad t \in T, \tag{8.65}$$

to account for the fact that it is possible to have units with more than one degree of freedom. Indeed, in general there can be ways to regulate a unit so as to modify the ratio between the production levels of the items. In that case, instead of univariate performance functions that determine a 1-dimensional curve in the space of the output variables, the feasible vectors $(q_t^1, q_t^2, \dots, e_t^1, e_t^2, \dots)$ would belong to a higher-dimensional subspace.

Deriving valid inequalities

Since it is an intersection of single-item polyhedral sets (with additional linking constraints), all the valid inequalities for a single-item unit can be adopted in a straightforward way to this case.

However, one can try to exploit the fact that a solution must fulfill all the balance equations simultaneously. Then, let us consider, for a $[k, l]$ interval, the set defined by:

$$u^1_{k-1} + \sum_{j=k}^{l} a^1_j z_j \geq d^1_{kl}$$
$$u^2_{k-1} + \sum_{j=k}^{l} a^2_j z_j \geq d^2_{kl}$$

where $a^i_j = \min\{Q^{max,i}_j, d^i_{jl}\}$, and for simplicity of notation we assume I has cardinality 2. By considering the complemented variables $\bar{z}_t = 1 - z_t$, one can define a continuous 0–1 multidimensional knapsack set:

$$\sum_{j=k}^{l} a^1_j \bar{z}_j \leq (\sum_{j=k}^{l} a^1_j - d^1_{kl}) + u^1_{k-1}$$
$$\sum_{j=k}^{l} a^2_j \bar{z}_j \leq (\sum_{j=k}^{l} a^2_j - d^2_{kl}) + u^2_{k-2}$$

in analogy to the continuous 0-1 knapsack set of [MW99], for which MIR inequalities can be derived. Lifted cover inequalities that attempt to exploit the global structure of the problem for a pure 0-1 multidimensional knapsack problem have been studied, e.g., by Kaparis and Letchford in [KL08]. However, to the best of our knowledge, no work has appeared so far on multidimensional knapsack problems with a continuous variable per constraint, that might be worth investigating.

Lagrangian relaxation

In order to compute good lower bounds, it is possible to build a Lagrangian relaxation obtained by decoupling the generated items and solving single-unit subproblems. It is necessary to keep in the formulation the variables q^i_t for each $i \in I$ and e^j_t for each $j \in J$, subject to the coupling constraints $q^i_t = \alpha^i_t f_t$, $e^j_t = \beta^j_t f_t$, and the copy variables $x^i_t, y^i_t \in \{0, 1\}$ such that $z_t = x^i_t, z_t = y^j_t$. Then, dualizing the coupling equalities, we end up with the dual function defined as:

$$\begin{aligned}
\omega(\underline{\mu},\underline{\nu},\underline{\pi},\underline{\rho}) = \min \sum_{t\in T}(c_t f_t + K_t z_t &+ \sum_{j\in J}(p_t^{j+}e_t^{j+} - p_t^{j-}e_t^{j-}) \\
&+ \sum_{i\in I}\mu_t^i(\alpha_t^i f_t - q_t^i) + \sum_{j\in J}\nu_t^j(\beta_t^i f_t - e_t^j)) \\
&+ \sum_{i\in I}\pi_t^i(z_t - x_t^i) + \sum_{j\in J}\rho_t^j(z_t - y_t^j))
\end{aligned}$$

$$\begin{aligned}
& u_t^i + d_t^i = u_{t-1}^i + q_t^i - s_t^i && i \in I, t \in T \\
& e_t^j + e_t^{j+} - e_t^{j-} = w_t^j && j \in J, t \in T \\
& u_t^i \le U^i && i \in I, t \in T \\
& z_t F_t^{min} \le f_t \le z_t F_t^{max} && t \in T \\
& x_t^i F_t^{min} \le q_t^i \le x_t^i F_t^{max} && i \in I, t \in T \\
& y_t^j F_t^{min} \le e_t^j \le y_t^j F_t^{max} && j \in J, t \in T \\
& z_t, x_t^i, y_t^j \in \{0,1\} && t \in T,
\end{aligned}$$

The dual function is separable, and can be decomposed by item. The decomposition yields a subproblem of the form:

$$\min \sum_{t\in T}[f_t(c_t + \sum_{i\in I}\alpha_t^i\mu_t^i + \sum_{j\in J}\beta_t^j\nu_t^j) + (K_t + \sum_{i\in I}\pi_t^i + \sum_{j\in J}\rho_t^j)z_t] \tag{8.66}$$

$$\begin{aligned}
& z_t F_t^{min} \le f_t \le z_t F_t^{max} && t \in T \\
& z_t \in \{0,1\} && t \in T,
\end{aligned}$$

the following subproblem for each item $i \in I$:

$$\min \sum_{t\in T}(-\mu_t^i q_t^i - \pi_t^i x_t^i) \tag{8.67}$$

$$\begin{aligned}
& u_t^i + d_t^i = u_{t-1}^i + q_t^i - s_t^i && t \in T \\
& u_t^i \le U^i && t \in T \\
& x_t^i F_t^{min} \le q_t^i \le x_t^i F_t^{max} && t \in T \\
& q_t^i, u_t^i \ge 0 && t \in T \\
& x_t^i \in \{0,1\} && t \in T,
\end{aligned}$$

and the following for each item $j \in J$:

$$\min \sum_{t\in T}(p_t^{j+}e_t^{j+} - p_t^{j-}e_t^{j-} - \nu_t^j e_t^j - \rho_t^j y_t^j) \tag{8.68}$$

$$\begin{aligned}
& e_t^j + e_t^{j+} - e_t^{j-} = w_t^j && t \in T \\
& y_t^j F_t^{min} \le e_t^j \le y_t^j F_t^{max} && t \in T \\
& e_t^j, e_t^{j+}, e_t^{j-} \ge 0 && t \in T \\
& y_t^j \in \{0,1\} && t \in T.
\end{aligned}$$

A Lagrangian relaxation yields solutions with a bound which is at least as good as the one of the LP relaxation. Preliminary computations, with a simple implementation of the subgradient method, show that this Lagrangian relaxation provides bounds of good quality, strictly better than the LP bounds.

The crucial point is whether the Lagrangian dual problem $\max_{\underline{\lambda}} \omega(\underline{\lambda})$, where $\underline{\lambda}$ is the vector of Lagrange multipliers, can be solved efficiently. This is true insofar as the subproblems are easier than the original problem, as even the evaluation of the function $\omega(\underline{\lambda})$ entails solving the subproblems exactly.

For the subproblem (8.66), this is certainly true, as it is trivially solved by inspection. The subproblems (8.68) corresponding to the items in J are typically easy, too, as they are non-storable single-item planning problems that can be solved for each t independently. The feasible region of a single-period subproblem for an item in J is shown in Figure 8.3. Observe that the polyhedron is unbounded –

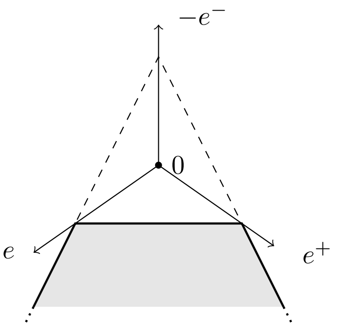


Figure 8.3: *Feasible region of the single-item, single-period subproblem obtained in the Lagrangian decomposition for non-storable items. The lower and upper bound on e are not shown.*

given a e_t^j, it is possible to arbitrarily increase e_t^{j+} and e_t^{j-} – but the direction of the objective function (with the usual assumption that $p_t^{j+} \geq p_t^{j-}$) and the bounds on e_t^j make the optimal value always finite.

Proposition 8.15. *The Lagrangian subproblem* (8.68) *for an item $j \in J$ has a finite optimal value that can be computed as the sum of the optimal values for each period $t \in T$. The optimal solution is, for each t, the one with minimum value* *among the following:*

$$\begin{cases} y_t = 0, e_t = 0, e_t^+ = w_t, e_t^- = 0 \\ y_t = 1, e_t = F_t^{max}, e_t^+ = w_t - F_t^{max}, e_t^- = 0 \\ y_t = 1, e_t = F_t^{max}, e_t^+ = 0, e_t^- = F_t^{max} - w_t \\ y_t = 1, e_t = F_t^{min}, e_t^+ = w_t - F_t^{min}, e_t^- = 0 \\ y_t = 1, e_t = w_t, e_t^+ = 0, e_t^- = 0 \\ y_t = 1, e_t = F_t^{max}, e_t^+ = w_t - F_t^{max} \\ y_t = 1, e_t = w_t, e_t^+ = 0, e_t^- = 0 \end{cases} \tag{8.69}$$

Proof. Since the problem is separable, we can consider the single-period problem for machine $j \in J$ and a fixed t. We drop for simplicity the j and t indices. The objective function to be minimized is:

$$p^+e^+ - p^-e^- - \nu e - \rho y.$$

Let us project the feasible region on the subspace (e^+, e^-), parametrized by the value of e, according to the equation $e^+ - e^- = w - e$. Although the feasible region is unbounded, since by assumption $p^+ > p^-$, the optimal value will always be such that either $e^+ = 0$ or $e^- = 0$.

For $y = 0$, $e = 0$ and the solution is $e = 0, e^+ = w, e^- = 0$, with value p^+w. For $y = 1$, e belongs to the interval $[F^{min}, F^{max}]$. The Lagrangian multiplier ν is free in sign, so we have two cases. If $\nu > 0$, there is an incentive to increase e as much as possible. Thus, $e = F^{max}$. If $F^{max} \leq w$, the solution is $e = F^{max}, e^+ = w - F^{max}, e^- = 0$. If $F^{max} > w$, the solution is $e = F^{max}, e^+ = 0, e^- = F^{max} - w$, with a negative objective function. If $\nu < 0$, the optimization direction goes towards decreasing e. If $p^+ \leq -\nu$, that is, e^+ is cheaper than e, the optimal solution will be $e = F^{min}, e^+ = w - F^{min}, e^- = 0$ or $e = w, e^+ = 0, e^- = 0$ if $F^{min} > w$. If $p^+ > -\nu$, that is, e^+ is more expensive than e, the optimal solution will be either $e = F^{max}, e^+ = w - F^{max}, e^- = 0$, or $e = w, e^+ = 0, e^- = 0$ if $F^{max} > w$.

□

It remains to be seen whether the Lagrangian subproblem (8.67), for the items in I, can be solved significantly faster than the complete multiple-item problem. If the production capacities are constant, one can use the dynamic programming algorithm described in Section 8.1.3, and solve the subproblems in polynomial time. In the general case, where bounds are not constant or additional UC-like logical constraints are considered, the subproblems (8.67) have to be solved as mixed-integer programs. Thanks to the decomposition, they can be solved in parallel, and valid inequalities for single-item lot-sizing problems with bounded inventory can be used. Computational experiments suggest that the relaxation is (not yet) computationally attractive with respect to a MILP approach. However, it must be noted that the choice of more sophisticated techniques for the optimization of the non-differentiable dual problem, such as bundle methods, could be of help, as well as improving the efficiency of solving the Lagrangian subproblems.

8.5 Concluding remarks

In this chapter, we have discussed the main building blocks of the operational planning problem for cogeneration systems, i.e., the single-unit problems with generation or cogeneration of multiple storable and non-storable items. For the single-item problem with constant upper and lower production bounds, we have described a dynamic programming-based polynomial-time algorithm. For problems with nonconstant bounds and multiple-items, valid inequalities from classic production planning problems can be extended. In Chapter 10, we will discuss a detailed MI(N)LP formulation for a real-world application, with systems including multiple cogeneration units and multiple items.

As previously mentioned, this is a first step in investigating MIP approaches for operational planning problems of cogeneration systems. A thorough polyhedral study of the single-unit variants would allow us to deal also with cogeneration systems containing multiple (co)generation units, for instance, by extending the single-unit valid inequalities, or decoupling the problem into single-unit subproblems by means of a Lagrangian relaxation.

CHAPTER 9

Robust optimization for production and operational planning

The parameters appearing in some classes of optimization problems are naturally prone to uncertainty. As an example, in the operational planning of cogeneration systems, the actual demands of the items over the time horizon can deviate significantly from the forecasts. In these cases, uncertainty plays a crucial role in determining the practical effectiveness of a solution: an optimal plan is of little use, if any small variation in the parameters heavily affects its cost – or even threatens its feasibility. Then, it is of the utmost importance to protect against the deviations that may occur in practice.

In this chapter, we investigate robust optimization approaches for the basic block of our operational planning problem, i.e., the single-unit, single-item variant, starting from existing approaches for production planning. Section 9.1 gives a brief introduction to optimization under uncertainty and, in particular, robust optimization approaches for production planning. In Section 9.2 we describe a Γ-robust optimization model for production planning, first proposed in [BT06], and we point out how it can be revised to be slightly less conservative. We also derive a class of (l, S) inequalities that are valid for the Γ-robust version of general production planning problems. In Section 9.3 we show how the revised formulation can be extended to the operational planning of energy systems, and we show how we can determine the robust solution just by solving the nominal problem with modified parameters. Finally, Section 9.4 reports some computational experiments comparing the robust formulations with various degrees of protection.

9.1 Robust optimization

Several ways to deal with uncertainty in optimization problems have been proposed during the years.

Stochastic approaches use *a priori* knowledge of the probability distributions of the parameters to obtain solutions where the minimum expected cost (or risk) is sought. One issue is that it can be hard to estimate probability distributions in an accurate way. Moreover, stochastic techniques usually require dealing with classes of problems that are computationally heavier than the original (deterministic) one (sometimes to the point of being intractable), since these approaches often incur the so-called *curse of dimensionality.*

For these reasons, distribution-free approaches have been proposed and studied especially in the last decade. In the Robust Optimization (RO) approach, the idea is to protect the solution against all possible realizations belonging to a so-called *uncertainty set*, in a min-max fashion. Determining the worst-case realization, given a solution, requires solving an adversarial problem. Clearly, the choice of an uncertainty set is crucial from the point of view of both the uncertainty modeling (one must avoid begin too conservative while guaranteeing sufficient protection) and the tractability of the resulting formulation.

The work of Soyster [Soy73] is among the first to formulate a robust problem to obtain a solution which is feasible for all parameters belonging to a given convex set. The core of Robust Optimization theory has been developed starting from the 90s, in particular in the work of Ben-Tal and Nemirovski [BTN00, BTN98], that show how most convex optimization problems subject to ellipsoidal uncertainty can be cast as second-order cone programs or SDP, that can be solved in polynomial time. A RO framework that has gained considerable attention in recent years is the so-called Γ-robustness, that originates from the work of Bertsimas and Sim [BS04]. This approach consists in characterizing the uncertainty set via a parameter Γ, the so-called budget of uncertainty, which bounds the number of deviations of the parameters from their nominal values. An attractive feature of Γ-robustness is that the robust counterpart of a problem maintains, broadly speaking, the same computational class of the deterministic formulation, in the sense that the robust counterpart of a linear program will still be a linear program, and so on. This is especially nice for problems with integer variables, for which the notions of robustness of Ben-Tal and Nemirovski yield problems that, in practice, are often too hard to deal with. Moreover, if the uncertainty set is constructed in a specific way, one can derive probabilistic guarantees on the feasibility of the obtained plans, akin to what is done in chance-constrained optimization.

Another body of research on RO focuses on less conservative models, in what is called adjustable or recoverable robustness, where (a part of) the decisions can be modified *after* the realization of (a part of) the parameters. The adjustable robust counterpart of an LP is, in general, already $\mathcal{NP}$-hard, but it is possible to restrict the adjustable variables yielding more tractable problems, as in affinely-adjustable robustness [BTGGN04]. In combinatorial problems, the idea of recoverable robustness has been explored in, e.g., [Büs11, BKK11a], and in [BKK11b] where a discrete set of scenarios is considered. A first attempt to solve general two-stage RO models with integer variables via a column-and-constraint generation algorithm is made in [ZZ13], although the problem still appears intractable for all but the simplest cases.

For an extensive treatment of the theory of RO, we refer to the book by Ben-Tal, El Ghaoui and Nemirovski [BTEGN09], while a recent survey of robust optimization approaches and applications can be found in [BBC11].

9.1.1 Right-hand-side uncertainty in production planning

In production planning, given a time horizon T, the result of the optimization is a *plan* $\underline{q}$ (or policy), that consists in a sequence of production decisions q_t for each $t \in T$. A plan is said to be *robust* with respect to uncertainty in the right-hand side if, for any realization of the demand vector $\underline{d}$, the plan is still feasible, in the sense that no constraints are violated, and its cost is within a certain upper bound.

Traditional work on production planning under uncertainty involves the use of (approximate) dynamic programming [Por02, Ber95] approaches or stochastic programming techniques [BL11] to derive policies that are optimal in expected value or with respect to some risk measure. These approaches have been very successful throughout the years, but have the drawbacks that we mentioned in the previous section: namely, accurately estimating the distributions is hard, and for some variants – especially with discrete variables – solving large-scale problems is very challenging.

Even for robust optimization models, most of the work has been on problems without discrete variables, which are typically more tractable. This is the case for the affinely-adjustable robust approaches proposed by Ben-Tal et. al. in [BTGS09, BTGNV05], and for the robust formulation proposed by See and Sim in [SS10], that leads to a second-order cone program. Bienstock and Özbay in [BÖ08] study the problem of robust production planning (without fixed costs) where the solutions have to be basestock (i.e., the optimal production level at time t can be determined as a function of the current demand and stock), and adopt a Benders like algorithm to solve it efficiently.

Bertsimas and Thiele [BT06, BT04] apply the framework of Γ-robustness to lot-sizing problems with backlogging and fixed costs, for the case of constant costs of production, holding and shortage. This approach will be described more in detail later. A practical application of the Γ-robustness approach of [BT06] to furniture manufacturing can be found in [JAM12].

9.2 Robust production planning

In order to be as self-contained as possible, in this section we discuss a robust formulation for a general production planning problem that we will later adapt to the operational planning problem. We start by considering a lot-sizing problem with backlogging, where the inventory variables can also be negative, since this is the case that is also considered in [BT06, BT04].

A standard mixed-integer programming formulation for production planning with backlogging reads:

$$\min \sum_{t\in T} c_t q_t + K_t z_t + \max\{h_t u_t, -p_t u_t\} \tag{9.1}$$

$$u_{t-1} + q_t = u_t + d_t \qquad t \in T \tag{9.2}$$

$$0 \le q_t \le Q_t z_t \qquad t \in T \tag{9.3}$$

$$z_t \in \{0,1\} \qquad t \in T \tag{9.4}$$

where q_t are the production variables with per-unit cost c_t and fixed costs K_t, u_t is the stock level at the end of period t, and h_t and p_t are, respectively, nonnegative holding and shortage costs. Note that the inventory u_t does not have any lower or upper bounds.

Let us suppose that data uncertainty affects the demand parameters d_t (right-hand side of Constraints (9.2)). In order to adopt a robust optimization approach, it is convenient to reformulate the problem in a different way. We project out each u_t variable by summing the balance equations up to period t, that is, we replace them in the objective function with their closed-form expression $u_t = \left(u_0 + \sum_{i=1}^{t}(q_i - d_i)\right)$. Then, the piecewise linear holding/shortage cost function can be rewritten leading to the following MILP:

$$\min \sum_{t\in T} c_t q_t + K_t z_t + y_t \tag{9.5}$$

$$y_t \ge h_t \left(u_0 + \sum_{i=1}^{t}(q_i - d_i)\right) \qquad t \in T \tag{9.6}$$

$$y_t \ge -p_t \left(u_0 + \sum_{i=1}^{t}(q_i - d_i)\right) \qquad t \in T \tag{9.7}$$

$$0 \le q_t \le Q_t z_t \qquad t \in T \tag{9.8}$$

$$z_t \in \{0,1\} \qquad t \in T \tag{9.9}$$

This formulation is more suitable for a static robust optimization approach, where we consider the worst possible realization for each row in the model. Each one of the

Constraints (9.6)–(9.7), for a period t, now contains all the uncertain parameters d_i with $i = 1, \ldots, t$. Their probability distribution is unknown, but we assume that each parameter belongs to the support interval $[\bar{d}_t - \widehat{d}_t, \bar{d}_t + \widehat{d}_t]$, in what is often called box uncertainty model. The uncertain demands can then be expressed as $d_t = \bar{d}_t + \widehat{d}_t\zeta_t$, where $\bar{d}_t$ is the nominal value, $\widehat{d}_t$ the maximum deviation and $\zeta_t \in [-1, 1]$ is the scaled deviation of the demand in period t.

If the number of deviations is not bounded, for the t-th holding cost in (9.6), the worst-case occurs when every demand in $i = 1, \ldots, t$ reaches its minimum (i.e., the realizations such that $\zeta_i = -1 \ \forall i \leq t$); while, for the t-th shortage cost in (9.7), the worst-case is obtained with the maximum demand ($\zeta_i = 1 \ \forall i \leq t$). Indeed, we could simply replace Constraints (9.6)–(9.7) with the following linear constraints:

$$y_t \geq h_t \left(u_0 + \sum_{i=1}^{t} (q_i - (\bar{d}_i - \widehat{d}_i)) \right) \qquad t \in T \qquad (9.10)$$

$$y_t \geq -p_t \left(u_0 + \sum_{i=1}^{t} (q_i - (\bar{d}_i + \widehat{d}_i)) \right) \qquad t \in T \qquad (9.11)$$

A concern about this kind of pure worst-case approach is that it seems highly unlikely that all the parameters are going to reach their worst-case deviation simultaneously. To mitigate this drawback, it is possible to define a less conservative uncertainty set in several ways. In the budget-of-uncertainty approach, the idea is to restrict the uncertainty set by imposing a budget Γ that acts as a threshold on the sum of absolute deviations. Let us then restrict our attention only to the scaled deviation vectors $\underline{\zeta}$ belonging to the uncertainty set defined as:

$$\mathcal{P} = \{\underline{\zeta} : |\zeta_t| \leq 1 \ \forall t \in T, \sum_{i=1}^{t} |\zeta_i| \leq \Gamma_t \ \forall t \in T\},$$

that is, all the deviation vectors such that, for every interval $[1, t]$, no more than Γ_t parameters can reach their worst-case value (either the upper or lower bound) simultaneously. A reasonable assumption for this approach is that $\Gamma_t \leq \Gamma_{t+1}$: having distinct and nondecreasing budget values avoids overprotecting the first periods. It does not seem realistic that a larger amount of deviation can occur in the first $k \ll n$ periods as in the whole time horizon $T = 1, \ldots, n$. Another assumption that we can safely make is that $\Gamma_{t+1} - \Gamma_t \leq 1$ for all $t \in T$, since, if only Γ_t deviations could occur in the first t periods, then no more that $\Gamma_t + 1$ should occur in the first $t + 1$.

The uncertainty set $\mathcal{P}$ is the one also considered by Bertsimas and Thiele in

[BT06]. Having defined $\mathcal{P}$, we can write the following robust formulation:

$$\min \sum_{t\in T} c_t q_t + K_t z_t + y_t \tag{9.12}$$

$$y_t \geq \max_{\underline{\zeta}\in\mathcal{P}} \left[h_t \left(u_0 + \sum_{i=1}^{t} (q_i - (\bar{d}_i + \widehat{d}_i \zeta_i)) \right) \right] \qquad t \in T \tag{9.13}$$

$$y_t \geq \max_{\underline{\zeta}\in\mathcal{P}} \left[-p_t \left(u_0 + \sum_{i=1}^{t} (q_i - (\bar{d}_i + \widehat{d}_i \zeta_i)) \right) \right] \qquad t \in T \tag{9.14}$$

$$0 \leq q_t \leq Q_t z_t \qquad t \in T \tag{9.15}$$

$$z_t \in \{0,1\} \qquad t \in T, \tag{9.16}$$

where Constraints (9.13)–(9.14) ensure that y_t is an upper bound to the cost obtained for any $\underline{\zeta} \in \mathcal{P}$.

Observe that the problem is well-posed and has finite value, since the nominal problem is feasible for any $\underline{\zeta} \in \mathcal{P}$. Given any feasible solution to the nominal problem, since backlogging is allowed, then the occurrence of a demand smaller or larger than expected cannot yield an infeasibility: given a production plan $\underline{q}$, an increase (decrease) in a demand d_t can always be translated into a decrease (increase) of the inventory u_t, since u_t can assume negative values. Therefore, the uncertainty of the problem only affects the objective function value: the aim of the robust counterpart is to minimize the worst-case cost, so as to provide an upper bound on the cost for any realization of the parameters within the uncertainty set.

Note that both an increase or a decrease with respect to the nominal demand may cause an increase in the overall cost. Since we have assumed symmetric deviations, it is easy to see that the worst-case cumulative deviation of the demands is the same both for the t-th holding cost and for the t-th shortage cost, and can be expressed as $\max_{\underline{\zeta}\in\mathcal{P}} \sum_{i=1}^{t} \widehat{d}_i \zeta_i$. Then, in order to compute such maximum cumulative deviation, Bertsimas and Thiele in [BT06] propose the following auxiliary LP:

$$A_t = \max \sum_{i=1}^{t} \widehat{d}_i \zeta_i \tag{9.17}$$

$$\sum_{i=1}^{t} \zeta_i \leq \Gamma_t \tag{9.18}$$

$$0 \leq \zeta_i \leq 1 \qquad \forall i \leq t \tag{9.19}$$

The problem is feasible and bounded, hence by strong duality its optimal cost is equivalent to the optimal value of its dual:

$$\begin{aligned} A_t = \min\ & \nu_t\Gamma_t + \sum_{i=1}^{t} r_{it} & & (9.20)\\ & \nu_t + r_{it} \geq \widehat{d}_i & \forall i \leq t &\\ & \nu_t \geq 0 & &\\ & r_{it} \geq 0 & \forall i \leq t. & \end{aligned}$$

Due to the direction of the inequalities, the objective function of the dual problem can be inserted directly in the holding/shortage constraints yielding the complete formulation, which we will denote, from now on, by BT:

$$\min \sum_{t \in T} (c_t q_t + K_t z_t + y_t) \tag{9.21}$$

$$y_t \geq h_t \left(u_0 + \sum_{i=1}^{t} (q_i - \bar{d}_i) + \nu_t\Gamma_t + \sum_{i=1}^{t} r_{it} \right) \qquad t \in T \tag{9.22}$$

$$y_t \geq -p_t \left(u_0 + \sum_{i=1}^{t} (q_i - \bar{d}_i) - \nu_t\Gamma_t - \sum_{i=1}^{t} r_{it} \right) \qquad t \in T \tag{9.23}$$

$$\nu_t + r_{it} \geq \widehat{d}_i \qquad \forall t, i \leq t \tag{9.24}$$

$$\nu_t \geq 0, r_{it} \geq 0 \qquad \forall t, i \leq t \tag{9.25}$$

$$0 \leq q_t \leq Q_t z_t \qquad t \in T \tag{9.26}$$

$$z_t \in \{0,1\} \qquad t \in T. \tag{9.27}$$

This MILP formulation protects row-wise against all possible realizations in the uncertainty set $\mathcal{P}$. The optimal values $A_t = \nu_t^*\Gamma_t + \sum_{i=1}^{t} r_{it}^*$ represent the worst-case deviation of the cumulative demand from its nominal value, subject to the budget of uncertainty Γ_t.

Let us give a few remarks on this formulation. It is quite evident that the optimal values A_t can be computed a priori as n linear programs (9.17)–(9.19), hence no additional variables and constraints are needed. It is sufficient to replace Constraint (9.22)–(9.23) with:

$$y_t \geq h_t \left(u_0 + \sum_{i=1}^{t} (q_i - d_i) + A_t \right) \qquad t \in T \tag{9.28}$$

$$y_t \geq -p_t \left(u_0 + \sum_{i=1}^{t} (q_i - d_i) - A_t \right) \qquad t \in T. \tag{9.29}$$

The robust problem we obtain is still a mixed-integer linear program. Nevertheless, in general the problem is different from the original nominal formulation (9.1)-(9.4), since we have inequalities with an extra term A_t that, in general, cannot be transformed back into the original balance constraints. Bertsimas and Thiele show that, for the special case with uniform costs $p_t = p$, $h_t = h$, the robust problem is equivalent to the original production planning problem with the modified demand values:

$$d'_t = \bar{d}_t + \frac{p-h}{p+h}(A_t - A_{t-1}), \tag{9.30}$$

where $A_0 = 0$, and an additional constant term $\frac{2ph}{p+h}\sum_{t\in T} A_t$ in the objective function. Hence, in this case the complexity of the robust problem is exactly equivalent to the nominal one. If the inventory also has a constant upper bound U, they show that the parameters can be transformed into

$$U_t = U - \frac{2p}{p+h} A_t, \tag{9.31}$$

although, in this case, the problem is not equivalent to the original, since the upper bound is no longer constant. Let us remark that these transformations are not valid with non-uniform holding/shortage costs or inventory bounds.

9.2.1 A revised Γ-formulation

The formulation of Bertsimas and Thiele is unnecessarily conservative with respect to $\mathcal{P}$. The definition of the uncertainty set:

$$\mathcal{P} = \{\underline{\zeta} : |\zeta_i| \le 1 \ \forall i \ge 0, \sum_{i=1}^{t} |\zeta_i| \le \Gamma_t \ \forall t \ge 0\}$$

implies that any realization of the vector $\underline{\zeta}$ will satisfy the budget constraint $\sum_{i=1}^{t} |\zeta_i| \le \Gamma_t$ for all $t \in T$. Formulation (9.17)–(9.19) only imposes the budget Γ_t for a given t, but it does not impose the budget Γ_i for all the subintervals $[1, i]$ with $0 \le i < t$. This does not invalidate the correctness of the formulation, in the sense that we are still protecting against all possible realizations of $\underline{\zeta} \in \mathcal{P}$ – however, this observation implies that the formulation is overprotecting, including also scaled deviation vectors $\underline{\zeta}$ which are not in $\mathcal{P}$.

To see why this may happen, consider a period $t > 1$. Assume that Γ_t and Γ_{t-1} are both integer, and $\Gamma_t = \Gamma_{t-1} + 1$. If $\hat{d}_t$ is smaller than the Γ_t-th largest deviation $\hat{d}_j$ with $j \in [1, t-1]$, then the cumulative deviation A_t, according to (9.17)–(9.19), will be $A_{t-1} + \hat{d}_j$. This means that, on the t-th row, we are protecting against a deviation vector where Γ_t demands in the interval $[1, t-1]$ reach their worst-case value, exceeding the threshold Γ_{t-1}.

A revised robust formulation, which does not overprotect the first periods, follows quite naturally by simply applying the definition of $\mathcal{P}$. We can write a revised primal-dual pair for the maximum deviation up to a given period t as follows:

$$H_t = \max \sum_{i=1}^{t} \widehat{d}_i \zeta_i \tag{9.32}$$
$$\sum_{j=1}^{i} \zeta_j \le \Gamma_i \qquad \forall i \le t$$
$$0 \le \zeta_i \le 1 \qquad \forall i \le t$$

$$H_t = \min \sum_{i=1}^{t} (\nu_{it}\Gamma_i + r_{it}) \tag{9.33}$$
$$\sum_{j=i}^{t} \nu_{jt} + r_{it} \ge \widehat{d}_i \qquad \forall i \le t$$
$$\nu_{it} \ge 0, r_{it} \ge 0 \qquad \forall i \le t,$$

where the optimal values H_t represent the worst-case deviation of the cumulative demand from its nominal value with $\underline{\zeta} \in \mathcal{P}$. The associated robust formulation becomes:

$$\begin{aligned}
&\min \sum_{t \in T} (c_t q_t + K_t z_t + y_t) && \\
&y_t \ge h_t \left(u_0 + \sum_{i=1}^{t} (q_i - d_i) + \sum_{i=0}^{t} (\nu_{it}\Gamma_i + r_{it}) \right) && t \in T \\
&y_t \ge -p_t \left(u_0 + \sum_{i=1}^{t} (q_i - d_i) - \sum_{i=0}^{t} (\nu_{it}\Gamma_i + r_{it}) \right) && t \in T \\
&\sum_{j=i}^{t} \nu_{jt} + r_{it} \ge \widehat{d}_i && \forall k, i \le k \\
&\nu_{it} \ge 0, r_{it} \ge 0 && \forall t, i \le t \\
&0 \le q_t \le Q_t z_t && t \in T \\
&z_t \in \{0,1\} && t \in T
\end{aligned}$$

The problem requires additional continuous variables ($O(n^2)$ instead of $O(n)$ dual variables in BT), but even in this case it is possible to compute *a priori* the optimal values H_t, as n linear programs, and substitute them in the formulation as follows:

$$\min \sum_{t \in T} (c_t q_t + K_t z_t + y_t) \tag{9.34}$$
$$y_t \ge h_t \left(u_0 + \sum_{i=1}^{t} (q_i - d_i) + H_t \right) \qquad t \in T \tag{9.35}$$
$$y_t \ge -p_t \left(u_0 + \sum_{i=1}^{t} (q_i - d_i) - H_t \right) \qquad t \in T. \tag{9.36}$$
$$0 \le q_t \le Q_t z_t \qquad t \in T \tag{9.37}$$
$$z_t \in \{0,1\} \qquad t \in T \tag{9.38}$$

In Example 9.1 we show an example where the formulation of Bertsimas and Thiele leads to an overprotection, and compare it with the revised formulation. Observe that, if the deviations $\hat{d}_t$ are nondecreasing with t, the BT formulation and the revised one are equivalent, since $A_t = H_t \ \forall t \in T$.

Some of the results valid for formulation BT can be extended to the revised formulation. In particular, if the cost coefficients are constant over time, it is still possible to reformulate the robust problem as a nominal one with modified demands computed as in Equation (9.30), with H_t in place of A_t.

Example 9.1. Consider an instance with 3 periods, $T = \{1, 2, 3\}$, and the following nominal demands, deviations and budgets of uncertainty:

t	$\bar{d}_t$	$\widehat{d}_t$	Γ_t
1	6	4	1
2	6	4	1
3	3	1	2

With formulation BT (Eq. (9.21)–(9.27)), the worst-case cumulative deviation for $t = 1$ is correctly computed to be $A_1 = 4$. For $t = 2$, the number of deviations in the interval $[1, 2]$ is bounded by $\Gamma_2 = 1$, so that $A_2 = 4$. For $t = 3$, the formulation protects against all the vectors such that $\zeta_1 + \zeta_2 + \zeta_3 \leq \Gamma_3 = 2$, including the vector $\underline{\zeta} = (1, 1, 0) \notin \mathcal{P}$, which results in a maximum cumulative deviation $A_3 = 8$. Clearly, this is overconservative, since our starting assumption was that, since $\Gamma_2 = 1$, only *one* of the maximum deviations in the first two periods could occur[1]

With the revised formulation (9.34)–(9.38), the worst-case cumulative deviations for $t = 1$ and $t = 2$ are the same, $H_1 = H_2 = 4$. In the third period, the worst-case deviation satisfies at the same time:

$$\begin{aligned}
&\zeta_1 \leq \Gamma_1 = 1\\
&\zeta_1 + \zeta_2 \leq \Gamma_2 = 1\\
&\zeta_1 + \zeta_2 + \zeta_3 \leq \Gamma_3 = 2,
\end{aligned}$$

yielding a maximum cumulative deviation $H_3 = 5$, for $\underline{\zeta} = (1, 0, 1)$ or $\underline{\zeta} = (0, 1, 1)$. We are protecting row-wise against a realization of the demand vector which is consistent with the assumed uncertainty set $\mathcal{P}$.

Uncertainty set generalization

The definition of the uncertainty set $\mathcal{P}$ can be easily generalized by defining a budget of uncertainty for any possible interval in T, and not only those starting from period 1. If the demand parameters are expressed as $d_i = \bar{d}_i + \widehat{d}_i\zeta_i$, we define the uncertainty set by considering the scaled deviation vectors $\underline{\zeta}$ that belong to the set $\mathcal{Q} = \{\underline{\zeta} : |\zeta_i| \leq 1 \ \forall i \geq 0, \sum_{i=j}^{l} |\zeta_i| \leq \Gamma_{jl} \ \forall (j, l) \in \mathcal{I}\}$ where $\mathcal{I} \subseteq \{(j, l) \in T \times T : j \leq l\}$. In other words, we define a budget Γ_{jl} for each possible subset of consecutive time periods. Then, the primal-dual pair for the maximum scaled deviation problem up to a given period t reads:

[1]Note also that it is false that $A_3 - A_2 \leq \hat{d}_3$ and that $A_3 - A_2 \leq \bar{d}_3$, thus invalidating a minor result (Lemma 3.4) in [BT06].

$$\begin{aligned} \max \sum_{i=1}^{t} \widehat{d}_i \zeta_i & \\ \sum_{i=j}^{\min(l,t)} \zeta_i \le \Gamma_{jl} & \qquad \forall (j,l) \in \mathcal{I}^t \\ 0 \le \zeta_i \le 1 & \qquad \forall i \le t \end{aligned} \qquad \begin{aligned} \min \sum_{(j,l)\in\mathcal{I}^t} \nu_{jl}^t \Gamma_{jl} + \sum_{i=1}^{t} r_i^t & \\ r_i^t + \sum_{\substack{(j,l)\in\mathcal{I}^t:\\ i\in[j,l]}} \nu_{jl}^t \ge \widehat{d}_i & \qquad \forall i \le t \\ \nu_{jl}^t \ge 0, r_i^t \ge 0 & \qquad \forall i \le t, \\ & \qquad \forall (j,l) \in \mathcal{I}^t, \end{aligned}$$

where $\mathcal{I}^t = \{(j,l) \in \mathcal{I} : j \le t\}$. In general, the fact that the cumulative deviations H_t can be computed a priori gives great flexibility in the definition of the uncertainty set, and it is possible to exploit knowledge domain to represent the deviations in a more refined way.

9.2.2 Robust (l, S) inequalities

Let us now consider the robust lot-sizing problem where backlogging is not allowed. Then, Constraints (9.36) become the nonnegativity constraints:

$$u_0 + \sum_{i=1}^{t} (q_i - \bar{d}_i) - H_t \ge 0. \tag{9.39}$$

We denote by X^{R-LS} the polyhedron described by Constraints (9.35), (9.38) and (9.39). A class of inequalities, in the spirit of (l, S) inequalities for the deterministic version, can be derived, if we assume that $\Gamma_t \ge \Gamma_{t-1}$ $\forall t \in T$ (thus $H_t \ge H_{t-1}$). Let us start from a basic valid inequality.

Lemma 9.2. *For a given $t \in T$, the inequality of the form:*

$$q_t \le \bar{d}_t z_t + u_0 + \sum_{j=1}^{t} (q_j - \bar{d}_j) - H_{t-1}, \tag{9.40}$$

with $H_0 = 0$, is valid for X^{R-LS}.

Proof. With a simple algebraic manipulation, we have that:

$$u_0 + \bar{d}_t - q_t + \sum_{j=1}^{t} (q_j - \bar{d}_j) - H_{t-1} = u_0 + \sum_{j=1}^{t-1} (q_j - \bar{d}_j) - H_{t-1} \ge 0$$

where the inequality follows from Constraint (9.39). The inequality can be rewritten as

$$q_t \le \bar{d}_t + u_0 + \sum_{j=1}^{t} (q_j - \bar{d}_j) - H_{t-1}. \tag{9.41}$$

Then, the claim follows since $z_t \in \{0, 1\}$. □

Observe that, by summing $\sum_{j=t+1}^{l} q_j$ to both sides of (9.41), we obtain:

$$\sum_{j=t}^{l} q_j \leq \bar{d}_{tl} + u_0 + \sum_{j=1}^{l}(q_j - \bar{d}_j) - H_{t-1}. \tag{9.42}$$

We can then generalize (9.40) in a way similar to what can be done for the deterministic lot-sizing, obtaining a class of robust (l, S) inequalities.

Proposition 9.3. *Let $1 \leq l \leq n$, $L = \{1, \dots, l\}$, $S \subseteq L$ and $a = \min\{j \in S\}$, then the following robust (l, S) inequality:*

$$\sum_{j \in S} q_j \leq \sum_{j \in S} \bar{d}_{jl} z_j + u_0 + \sum_{j=1}^{l}(q_j - \bar{d}_j) - H_{a-1}, \tag{9.43}$$

with $H_0 = 0$, is valid for X^{R-LS}.

Proof. Consider a feasible point $(\underline{q}, \underline{z})$. If $\sum_{j \in S} z_j = 0$, then $q_j = 0$ for $j \in S$, and the inequality is satisfied since $H_l \geq H_{a-1}$ by assumption and $u_0 + \sum_1^l (q_j - \bar{d}_j) - H_l \geq 0$ by Constraint (9.39). Otherwise, let $b = \min\{j \in S : z_j = 1\}$. Then:

$$\sum_{j \in S} q_j \leq \sum_{j=b}^{l} q_j \leq \bar{d}_{bl} + u_0 + \sum_{j=1}^{l}(q_j - \bar{d}_j) - H_{b-1} \leq \sum_{j \in S} \bar{d}_{jl} z_j + u_0 + \sum_{j=1}^{l}(q_j - \bar{d}_j) - H_{a-1},$$

where the first inequality follows from the definition of S and the nonnegativity of q_j, the second from (9.42), and the third from $z_b = 1$ and $H_{b-1} \geq H_{a-1}$, since $b \geq a$. □

Given a $(\underline{q}^*, \underline{z}^*)$ solution to the LP relaxation, the separation problem can be solved by considering the inequality rewritten as:

$$\sum_{j \in S} \bar{d}_{jl} z_j + \sum_{j \in L \setminus S} q_j \geq \bar{d}_{1l} + H_{a-1} - u_0. \tag{9.44}$$

Then, for each $L = \{1, \dots, l\}$ with $l \in \{1, \dots, n\}$, the set S minimizing the left-hand side is $S = \{j \in L : q_j^* > \bar{d}_{jl} z_j^*\}$, and the inequality is violated if the quantity $\sum_{j \in L} \min\{q_j^*, \bar{d}_{jl} z_j^*\}$ is smaller than $\bar{d}_{1l} + H_{a-1} - u_0$.

Numerical example

Let us show with a brief example that the valid (l, S) inequalities (9.43) can have a significant impact solving the robust production planning problem defined by Equations (9.34), (9.35), (9.37), (9.38) and (9.39). We consider an instance with $n = 70$, with the demand profile in Figure 9.1, $Q_t^{max} = 100$, the time-varying costs:

$$c_t = \begin{cases} 6 & 19 \leq t \leq 53 \\ 5 & t \leq 18 \vee t \geq 54, \end{cases} \qquad h_t = \begin{cases} 4 & 19 \leq t \leq 53 \\ 4 & t \leq 18 \vee t \geq 54, \end{cases}$$

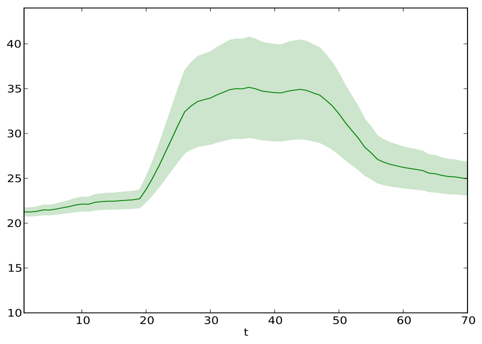


Figure 9.1: *Demand profile and uncertainty interval ($\bar{d}_t \pm \hat{d}_t$) for the considered instance.*

and the budgets of uncertainty computed as $\Gamma_t = 0.5\sqrt{t}\ \forall t \in T$.

Table 9.1 reports the linear programming bound with and without the introduction of the robust (l, S) inequalities. The valid inequalities yield a remarkable

Valid inequalities	bound	gap
None	19815.774	25.95
Robust (l, S)	24945.375	0.05

Table 9.1: *Linear programming bound and relative gap with and without robust (l, S) inequalities (optimal value 24959.167).*

improvement of the linear programming bound with respect to the original formulation, which is more than 25% from the optimal value. Note that this is due to the fact that the activation Constraints (9.37) cause a very weak relaxation, unless strengthened. In Table 9.2 we report the results obtained with a full branch-and-cut algorithm. The solver we use is Gurobi 5.6, with default settings (all the generic MIP cutting planes are enabled). The separation algorithm is implemented in the Julia language via the JuMP library. The table reports the number of explored

Valid inequalities	nodes	time	generated cuts	
			robust (l, S)	default MIP cuts
None	832	2.48	–	105
Robust (l, S)	2	0.96	69	70

Table 9.2: *Solving to optimality with and without robust (l, S) inequalities.*

nodes, the computing time, the number of robust (l, S) inequalities that we separate and the number of generic MIP cuts separated by Gurobi[2]. The robust (l, S) inequalities appear to be very effective in reducing the computing time and the number of explored nodes, even with only 69 generated inequalities.

9.3 Robust operational planning in cogeneration systems

Let us now turn to the operational planning problem with single-item, single-unit discussed in Section 8.1. We consider the linear variant, where we have the production variables q_t with production costs $c_t > 0$ and fixed costs $K_t > 0$, while there are no holding costs on the inventory u_t, which has an upper bound U and must be nonnegative. The nominal formulation is as follows:

$$\min \sum_{t \in T} c_t q_t + K_t z_t \tag{9.45}$$

$$u_{t-1} + q_t = u_t + d_t \qquad t \in T \tag{9.46}$$

$$0 \leq u_t \leq U \qquad t \in T \tag{9.47}$$

$$Q_t^{min} z_t \leq q_t \leq Q_t^{max} z_t \qquad t \in T \tag{9.48}$$

$$z_t \in \{0, 1\} \qquad t \in T. \tag{9.49}$$

In this case, note that, given a production plan $\underline{q}^*$, a variation in the demands d_t does not affect the cost, but may introduce an infeasibility. Similarly to what we have described in the previous section, we can project out the u_t variables and work only in the space of the production variables q_t. Then, we can reformulate the problem in a way that is more suitable for a robust approach, obtaining:

$$\min \sum_{t \in T} c_t q_t + K_t z_t \tag{9.50}$$

$$u_0 + \sum_{i=1}^{t} (q_i - d_i) \geq 0 \qquad t \in T \tag{9.51}$$

$$u_0 + \sum_{i=1}^{t} (q_i - d_i) \leq U \qquad t \in T \tag{9.52}$$

$$Q_t^{min} z_t \leq q_t \leq Q_t^{max} z_t \qquad t \in T \tag{9.53}$$

$$z_t \in \{0, 1\} \qquad t \in T. \tag{9.54}$$

This formulation is close to formulation (9.5)–(9.9), with $h_t = 0$, the shortage cost replaced by the nonnegativity constraint, and an upper bound on u_t. Let us consider again the budget-constrained uncertainty set $\mathcal{P} = \{\underline{\zeta} : |\zeta_i| \leq 1 \ \forall i \geq$

[2]In particular, for this problem Gurobi generates Gomory and flow cover cuts.

$1, \sum_{i=1}^{t} |\zeta_i| \leq \Gamma_t \ \forall t \geq 0\}$. Following the revised approach, the robust version of the problem is:

$$\min \sum_{t \in T} (c_t q_t + K_t z_t) \tag{9.55}$$

$$u_0 + \sum_{i=1}^{t} (q_i - \bar{d}_i) - H_t \geq 0 \qquad t \in T \tag{9.56}$$

$$u_0 + \sum_{i=1}^{t} (q_i - \bar{d}_i) + H_t \leq U \qquad t \in T \tag{9.57}$$

$$Q_t^{min} z_t \leq q_t \leq Q_t^{max} z_t \qquad t \in T \tag{9.58}$$

$$z_t \in \{0, 1\} \qquad t \in T, \tag{9.59}$$

where the values H_t are computed a priori as in (9.32). Note that, since there are no holding costs involved, we can actually compute modified demands and inventory upper bounds to formulate the robust problem with Formulation (9.45)–(9.49).

Proposition 9.4. *The optimal solution for Formulation* (9.55)–(9.59) *is equivalent to the optimal solution of the nominal problem* (9.45)–(9.49) *with the modified demand:*

$$d'_t = \bar{d}_t + H_t - H_{t-1}, \tag{9.60}$$

where H_t is the cumulative deviation of the demands at period t and $H_0 = 0$, and the modified inventory upper bound on the stock variables u_t:

$$U_t = U - 2H_t. \tag{9.61}$$

Proof. The transformed nominal problem follows the balance equation $u_{t-1} + q_t = u_t + d'_t$, with $0 \leq u_t \leq U_t$ and $d'_t = \bar{d}_t + H_t - H_{t-1}$. Then, $u_t = u_{t-1} + q_t - d'_t = u_0 + \sum_{i=1}^{t}(q_i - d'_i) = u_0 + \sum_{i=1}^{t}(q_i - \bar{d}_i) - \sum_{i=1}^{t}(H_i - H_{i-1}) = u_0 + \sum_{i=1}^{t}(q_i - \bar{d}_i) - H_t$. From $0 \leq u_t \leq U_t$ follows that, for any $t \in T$, $u_0 + \sum_{i=1}^{t}(q_i - \bar{d}_i) - H_t \geq 0$ and, according to the definition of U_t, $u_0 + \sum_{i=1}^{t}(q_i - \bar{d}_i) + H_t \leq U$. Since the objective functions are the same, the two problems are equivalent. □

Note that the modified problem is not exactly equivalent to the original one, since the inventory bound is no longer constant. However, this proposition allows us to extend the majority of the results for the nominal problem to the robust case. In particular, the problem is polynomially solvable if the production bounds Q_t^{min} and Q_t^{max} are constant (see Section 8.1.3).

Probability bounds of constraint violations

An interesting feature of the Γ-robustness approach is that it also allows us to establish upper bounds on the probability that the t-th Constraint (9.52) is violated.

Given an optimal solution $\underline{q}^*$, the probability that the inventory upper bound is exceeded can be bounded as follows:

$$\begin{aligned}
Pr\left(u_0+\sum_{i=1}^{t}(q_i^*-d_i)>U\right) = Pr\left(u_0+\sum_{i=1}^{t}(q_i^*-\bar{d}_i+\hat{d}_i\zeta_i)>U\right) \leq \\
Pr\left(\sum_{i=1}^{t}\hat{d}_i\zeta_i>H_t)\right) = Pr\left(\sum_{i=1}^{t}\frac{\hat{d}_i}{\max_j\hat{d}_j}\zeta_i>\frac{H_t}{\max_j\hat{d}_j})\right) \leq \\
Pr\left(\sum_{i=1}^{t}\gamma_i\zeta_i\geq\frac{H_t}{\max_j\hat{d}_j})\right) \qquad (9.62)
\end{aligned}$$

where $\gamma_i=\frac{\hat{d}_i}{\max_j\hat{d}_j}$ and the first inequality follows from the fact that q^* satisfies $u_0+\sum_{i=1}^{t}(q_i^*-\bar{d}_i)+H_t\leq U$ (Eq. (9.57)). Since $\gamma_i\leq 1$, assuming that ζ_i are i.i.d. random variables in $[-1,+1]$, we can obtain a probabilistic bound applying, e.g., the following result[3]:

Theorem 9.5 ([BS04]). *If ζ_i, $i=1,\dots,t$, are independent and symmetrically distributed random variables in $[-1,+1]$, then*

$$Pr\left(\sum_{i=1}^{t}\gamma_i\zeta_i\geq N)\right)\leq\exp\left(\frac{-N^2}{2t}\right).$$

An equivalent bound can be obtained also for the t-th nonnegativity constraint. It is worth pointing out that, since the random variables ζ_j in different constraints are *not* independent, a meaningful bound on the probability that none of the constraint is violated cannot be simply obtained as $\prod_{t\in T}\left(1-Pr\left[u_0+\sum_{i=1}^{t}(q_i-d_i)>U\right]\right)$.

9.3.1 Uncertainty in the objective function

Let us now consider what happens in the case where the cost vector $\underline{c}$ is subject to data uncertainty. This is a relevant issue when dealing with longer time horizons: as an example, electrical energy prices in liberalized markets may be subject to significant fluctuations over time, and the same can happen to fuel/gas price. However, uncertainty in the cost coefficients can be important also in short-term planning: the efficiency of each cogeneration unit is affected by the environmental conditions (especially the temperature), thus affecting the final production cost.

Let us assume that the cost coefficients can be expressed as $c_i=\bar{c}_i+\widehat{c}_i\cdot\zeta_i$ with the uncertainty set defined by the vectors $\underline{\zeta}$ that belong to the set:

$$\mathcal{R}=\{\underline{\zeta}:|\zeta_i|\leq 1\ \forall i\geq 0,\sum_{i=1}^{t}|\zeta_i|\leq\Gamma\}.$$

[3]A stronger bound, which we do not report for sake of brevity, can be obtained by applying Theorem 3 in [BS04]

Observe that, in this case, a single budget of uncertainty Γ is sufficient. The worst-case occurs where the cost coefficients are maximized, thus the robust formulation is the following:

$$\min \max_{\zeta \in \mathcal{R}} \sum_{t \in T} ((\bar{c}_t + \hat{c}_t \zeta_t) q_t + K_t z_t) \tag{9.63}$$

$$u_{t-1} + q_t = u_t + d_t \qquad t \in T \tag{9.64}$$

$$0 \leq u_t \leq U \qquad t \in T \tag{9.65}$$

$$Q_t^{min} z_t \leq q_t \leq Q_t^{max} z_t \qquad t \in T \tag{9.66}$$

$$z_t \in \{0, 1\} \qquad t \in T \tag{9.67}$$

which, by strong duality, can be rewritten as:

$$\min \sum_{t \in T} (\bar{c}_t q_t + K_t z_t + \sigma\Gamma + \sum_{i \in T} \rho_i) \tag{9.68}$$

$$u_{t-1} + q_t = u_t + d_t \qquad t \in T \tag{9.69}$$

$$0 \leq u_t \leq U \qquad t \in T \tag{9.70}$$

$$\sigma + \rho_i \geq \widehat{c}_i q_i \qquad \forall i \in T \tag{9.71}$$

$$Q_t^{min} z_t \leq q_t \leq Q_t^{max} z_t \qquad t \in T \tag{9.72}$$

$$\sigma \geq 0, \rho_i \geq 0 \qquad \forall i \in T \tag{9.73}$$

$$z_t \in \{0, 1\} \qquad t \in T \tag{9.74}$$

where σ and ρ_i are the dual variables associated with, respectively, the budget constraint and the upper bounds on ζ_i. Notice that, in this case, it is not possible to compute a priori the worst-case deviation. A similar reformulation can also be applied for uncertainty in the fixed costs K_t, and they can be combined with the formulations for uncertainty in the right-hand side.

9.4 Computational experiments

In this section, we summarize some computational results that show, for the single-item single-unit case, how the choice of the Γ-robustness parameters for the right-hand side affects the cost and the feasibility of the optimal plans in the realizations. We first consider the case of a general production planning problem with backlogging, where the uncertainty only affects the cost of the plan. Then, we consider the operational planning problem, where the demand uncertainty threatens the feasibility of the chosen plan. In both cases, we show the difference between the BT approach and our revised formulation.

9.4.1 Production planning

Let us consider the case of a production planning problem with backlogging (Section 9.2). Since no bounds are imposed on the inventory variables u_t (that can also be negative), any plan $\underline{q}^*$ will always be feasible in any realization of the demands. However, the holding and shortage costs, that contribute to the cost of the chosen plan, will be greatly affected by the actual demand.

The results summarized in this section have been obtained with a realistic instance of $n = 50$ time periods, with the demand profile in Figure 9.2: the dark line corresponds to the nominal demands $\bar{d}_t$, while the filled area represents the deviations $\widehat{d}_t$. In particular, we assume a high demand and high variance ($\widehat{d}_t = 5$) during the central periods, while in the initial and final part of the horizon we assume a low demand with low variance ($\widehat{d}_t = 1$). The other parameters in the instance are:

$$c_t = \begin{cases} 6 & 14 \leq t \leq 38 \\ 5 & t \leq 13 \vee t \geq 39, \end{cases} \qquad h_t = \begin{cases} 4 & 14 \leq t \leq 38 \\ 3 & t \leq 13 \vee t \geq 39, \end{cases} \qquad p_t = h_t + 20,$$

$K_t = 120$, and $Q_t = 100$ $\forall t \in T$. Note that backlogging is possible, but is greatly penalized with the cost p_t. To choose the values of Γ_t, [BT06] shows that, under

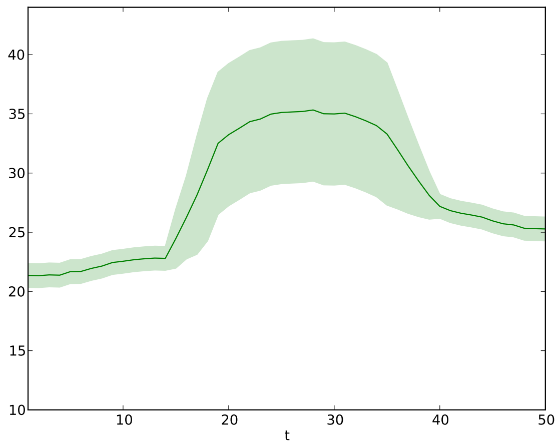


Figure 9.2: *Demand profile and uncertainty interval ($\bar{d}_t \pm \hat{d}_t$) for the considered instance*

some assumptions, the optimal value of the parameters Γ_t is proportional to $\sqrt{t}$; then, we select their value, for each time period t, according to the equation:

$$\Gamma_t = \alpha\sqrt{t} \qquad \forall t \in T,$$

where the parameter α represents the level of protection that we impose.

The experiments are carried out as follows. First, in the optimization phase, the robust formulation is solved with $\alpha \in [0, 2]$. Note that, for $\alpha = 0$, all Γ_t take value 0 and the model is the nominal one, with $H_t = 0$ $\forall t \in T$. Then, in order to evaluate the obtained plans, we generate 1000 realizations, assuming that each d_t is distributed as a normal variable with mean $\mu = \bar{d}_t$ and standard deviation $\sigma = \frac{\widehat{d_t}}{2}$.

Figure 9.3 reports the optimal costs obtained in the optimization phase (red line) and the distribution of the actual costs in the realizations (average and standard deviation in blue). We also include results obtained with Formulation BT, corresponding to the dashed lines.

If the values of Γ_t are sufficiently large, the optimal cost of the robust formulation provides an upper bound on the actual cost in any realization. However, the robust plan gives, in principle, no indication of the expected value of the realizations.

The average realization cost is larger than the optimization result for $\alpha < 0.3$, suggesting that the protection level is not sufficient. For $\alpha = 0$, that is, the nominal case, the overall cost in the realization is, on average, almost 50% greater than the value obtained from the optimization. Observe also how the deviation is large and that almost all the realizations fall above the expected cost. Raising α, the value obtained in the optimization phase provides an upper bound which is valid with increasingly high probability. The plot shows that the robust approach allows us to obtain the best results in terms of average cost if α is around 0.5. Further raising α gives no advantage: the upper bound gets considerably weaker, the actual cost of the realizations is worse on average, and the variance of the overall cost in the realization does not decrease. Comparing the two formulations, the revised approach provides a tighter upper bound, and optimal plans with a slightly superior performance in practice, although the difference seems to be small for the considered instance.

9.4.2 Operational planning

In the case of operational planning, the cost of the plan is determined by solving the robust version of the problem, and is not modified by the realization of the parameters, since there are no holding costs, and shortages are not allowed. Then the focus is exclusively on the constraint violations.

The instance we use in the experiments has the same production costs and demand profile of the one described in the previous section. In addition, the storage (inventory) has an upper bound $U = 200$. We evaluate the plans over 1000 realizations obtained as previously described, and we experiment again with values of Γ_t growing as the square root of t:

$$\Gamma_t = \alpha\sqrt{t} \qquad \forall t \in T.$$

Figure 9.4 summarizes how the feasibility of the robust plans changes as α is increased. In particular, we report, in red, the number of realizations where the inventory constraints were not satisfied, and, in blue, the percentage increase in

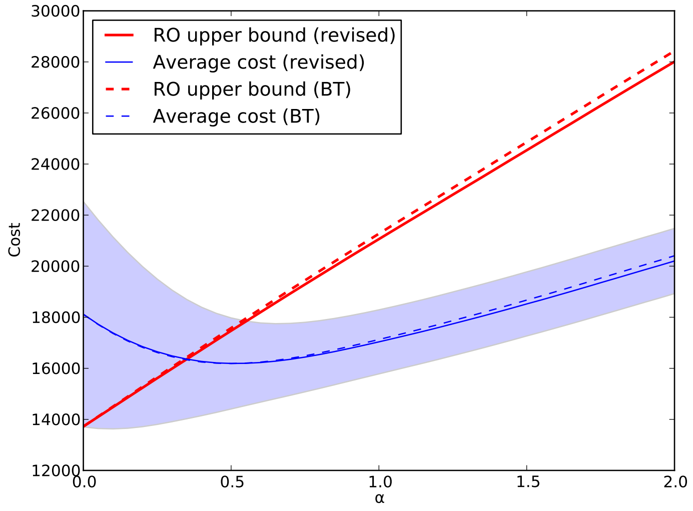


Figure 9.3: *Upper bound obtained with the RO formulations (red line) and average $\pm$ standard deviations of the actual costs in 500 realizations with budget of uncertainty $\Gamma_t = \alpha\sqrt{t}$.*

the cost of the robust plan with respect to the cost of the nominal version ($\alpha = 0$). Note that, in this case, the cost of a plan is exactly determined by solving the robust problem, since, given a production plan $\underline{q}$, the objective function does not depend on the actual realization of the demands.

The optimal plan obtained with nominal parameters ($\alpha = 0$) is quite fragile, in the sense that it fails to satisfy the constraints in more than 50% of the realization. If we want all the constraints to be satisfied in *any* realization (or with high probability), it is necessary to be slightly more aggressive than in the case with backlogging. Indeed, avoiding *failures* (violations of the bounds) with probability close to 1 requires a rather conservative approach, with α larger than 0.7. Nevertheless, the extra cost of the robust plan is acceptable, as it is well below 2%.

Compared to the BT approach, the revised formulation gives comparable results with respect to the fraction of failures in the realizations. Observe, however, that the cost of the BT plan, which is more conservative, can be significantly larger, even more than 1% for large protection levels.

It is worthwhile observing that the value of the storage capacity is crucial in determining even the existence of a plan which is robust against a given uncertainty set: indeed, we can view the inventory as a buffer that protects against uncertainties. Let us apply again the robust approach to the same instance, but decreasing the storage capacity to $U = 40$, instead of $U = 200$. Figure 9.5 shows that, with $\Gamma_t = \alpha\sqrt{t}$, no robust plans exist with $\alpha > 0.5$ for BT and with $\alpha > 0.6$ for

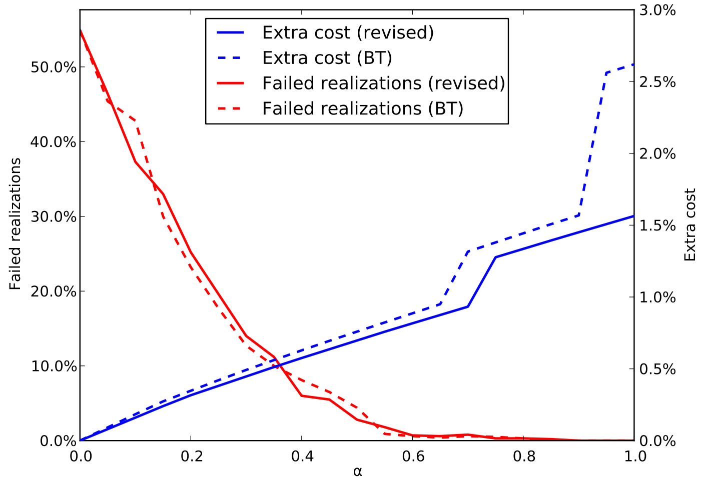


Figure 9.4: *Percentage of realizations where the robust plan is infeasible (red) and additional cost of the robust plan with respect to the nominal one (blue), with $\Gamma_t = \alpha\sqrt{t}$ and $U = 200$. The steep cost increase for $\alpha \approx 0.7$ is due to an increase in the number of production periods, hence incurring extra fixed cost ($K_t \gg c_t$).*

the revised formulation[4]. For both formulations, the most *robust* plan that can be achieved still has a significant failure rate (almost 10% for the revised formulation).

In a way, this tells us something about the robustness of the system itself. It would actually be useful, then, to formulate a problem where the operational planning and the design decisions are combined, so as to determine jointly the schedule and the optimal design of the system which minimizes the overall cost, while ensuring protection against the uncertainty set. This could be done in a quite straightforward way, adding a term in the objective function which is proportional to the size of the installed storage tank, or integer variables if, e.g., one can choose only among a finite set of storage tank models/sizes.

9.4.3 Concluding remarks

In this chapter, we have discussed a revised version of the Γ-robust formulation for production planning problems proposed in [BT06]. We have also described a family of robust (l, S) inequalities that are valid for the robust formulation of general production planning problems, when backlogging is not allowed. The approach is of interest to us because it can be easily extended to the operational planning problem. An attractive feature is that, for the single-unit, single-item operational planning

[4]In the sense that the robust formulation is infeasible.

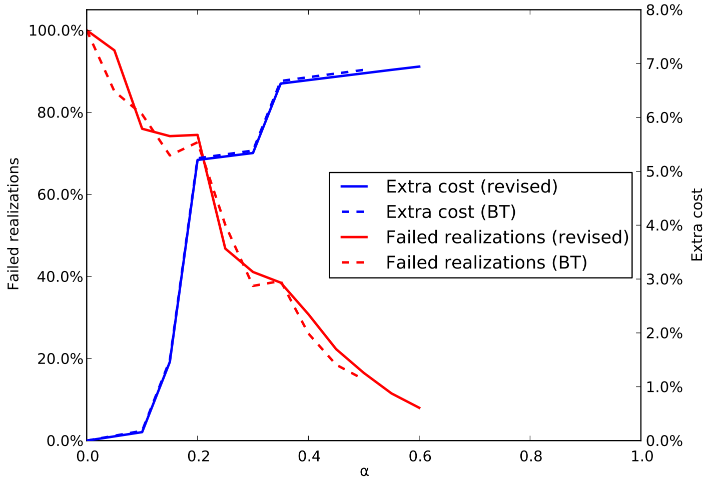


Figure 9.5: *When the inventory capacity is reduced to $U = 40$, robust plans that guarantee a high protection level do not exist.*

problem with uncertainty in the right-hand side, the approach leads to a problem which has almost exactly the same structure of the nominal one. This property is not preserved when uncertainty affects the objective function coefficients.

We have also summarized some computational experiments that show the behavior of the robust formulations with various degrees of protection. Note that we have not discussed the optimal choice of the robust parameters Γ_t, which is a fundamental issue in itself. Future work may also include the extension of the approach to multiple-item, multiple-units variants; in particular, it seems worthwhile investigating in which cases it remains possible to obtain a robust solution just by modifying some of the parameters of the nominal problem.

CHAPTER 10

MIP-based approaches for a real-world application

In this chapter we describe a real-world application where we apply mixed-integer programming methods to a short-term operational planning problem. First, we give some details on the components of the systems that we consider. Then, we describe a complete mixed-integer nonlinear programming formulation, including detailed technical constraints, and describe how we can deal with nonlinearities in practice, by using a piecewise linear approximation. Moreover, since it is often useful to extend the operational planning problem to cases with a larger number of time periods, we propose a rolling-horizon MILP-based heuristic to tackle large-size instances from an Italian energy company, where yearly incentives have to be considered. Some of the results in this chapter can be found in [TABM15, TAMB15, BTM^{+}14].

10.1 Real-world energy cogeneration systems

Real-world cogeneration systems can vary substantially in terms of number and types of units as well as in terms of scale, ranging from small scale plants (domestic applications with $<$ 50 kW fuel input) to large scale ones (industrial applications with >100 MW fuel input). Cogeneration energy systems may involve the following types of cogeneration units:

- One-degree-of-freedom cogeneration units that simultaneously generate electric and termal power, e.g., gas turbines, internal combustion engines, back-pressure steam cycles, fuel cells.

- Two-degree-of-freedom cogeneration units that simultaneously generate electric and thermal power (depending on two operating variables). This class includes, for instance, gas turbines with supplementary firing in the heat recovery section, steam cycles with extraction-condensing turbine, combined cycles with supplementary firing in the Heat Recovery Steam Generator (HRSG) and back-pressure bottoming cycle.

Cogeneration systems may also include single-item generation units such as:

- boilers (i.e., one-degree-of-freedom units generating only heat from fuel),

- compression heat pumps (i.e., one-degree-of-freedom units generating only heat from electricity),

- compression chillers (i.e., one-degree-of-freedom units generating only refrigeration power from electricity),

- absorption chillers (i.e., one-degree-of-freedom units generating only refrigeration power from heat).

With these types of units, we account for a wide variety of cogeneration systems, involving units with multiple degrees of freedom (two or more operating variables) and different size.

Figure 10.1 gives a schematic representation of a cogeneration system comprising multiple cogeneration and generation units as well as networks for the distribution of electric power, refrigeration power, high and low temperature thermal power. For instance, the HT heat network allows to model a steam network for an industrial heat user, while the LT heat network accounts for a district heating network. Storage tanks can be connected to the heat networks as well as to the refrigeration power network. The electric power generated by the units can be used to fulfill the customers' demands and, at the same time, drive the compression heat pumps and compression chillers, and satisfy the electricity needs of the absorption chillers. Electric power can be sold/purchased to/from the electric grid. The HT and LT heat networks are interconnected in order to have the possibility to downgrade high-temperature heat down to the low temperature heat network. Finally, thermal power in excess can, if needed, be dissipated through a dedicated heat exchanger.

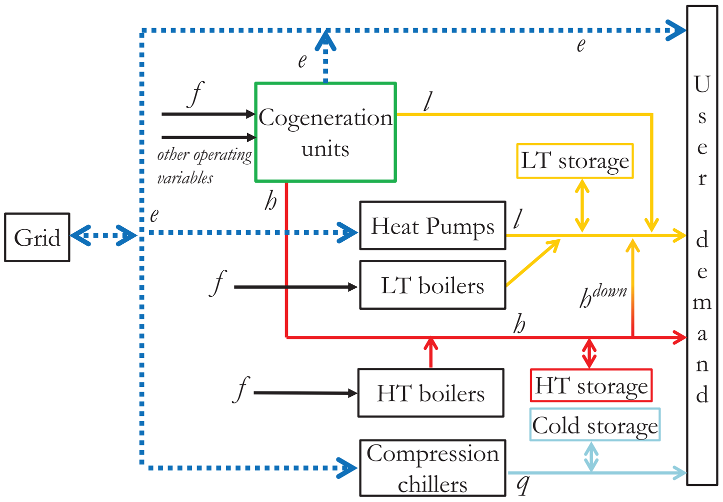


Figure 10.1: *Schematic representation of a CCHP network connecting the (co)generation units with the storage tanks, the electric grid and the users. Red and yellow arrows represent, respectively, the high (h) and low-temperature (l) thermal power flows, light blue arrows represent the refrigeration power flows (q), blue dotted arrows represent the electric power (e), and the black ones the fuel (f) consumed by each unit.*

10.2 MINLP formulation

As seen in Chapter 8, the short-term operational planning of cogeneration systems can be modelled as a mixed-integer nonlinear problem. Let us now describe in detail the model for a real-world application, starting from the parameters and variables. Then, we will give a MINLP formulation, and describe a piecewise-linear approximation that we can use in practice to approximate the nonlinear functions, when solving directly the MINLP is not possible.

Sets and parameters

$\mathcal{T}$: set of time periods (hours)
$\mathcal{U}$: set of all generation units
$\mathcal{F}$: set of units consuming fuel
$\mathcal{E}$: set of units consuming electricity
$\mathcal{C}$: set of units that generate refrigeration
$\mathcal{H}$: set of units that generate high-temperature heat
$\mathcal{L}$: set of units that generate low-temperature heat
$\mathcal{G}$: set of units that generate electricity
c_i^{OM}: hourly operation and maintenance cost for unit i [€]

c_i^δ: start-up cost for unit i [€]
c_i^f: unit cost of fuel consumed by unit i [€/kWh]
b_t: unit price of electricity bought from the grid at time t [€/kWh]
p_t: unit price of electricity sold to the grid at time t [€/kWh]
$F_{it}^{min}, F_{it}^{max}$: minimum and maximum fuel input for unit $i \in \mathcal{F}$ at time t [kWh]
$E_{it}^{min}, E_{it}^{max}$: minimum and maximum electricity input for unit $i \in \mathcal{E}$ [kWh]
N_i: maximum number of start-ups for unit i
U, V, W: capacity of low/high-temperature heat and refrigeration storage [kWh]
α, β, γ: constant deterioration rate for thermal and refrigeration storage
$D_t^{low}, D_t^{high}, D_t^{cold}, D_t^e$: demand for low and high-temperature heat, refrigeration power, electricity at time t [kWh]

Decision variables

f_{it}: fuel consumed by unit $i \in \mathcal{F}$ in period t [kWh]
y_{it}: secondary fuel consumed by unit $i \in \mathcal{F}$ with post-firing injection [kWh]
x_{it}: extraction valve opening percentage for combined cycle units [%]
e_{it}^{cons}: electricity consumed by unit $i \in \mathcal{E}$ in period t [kWh]
e_{it}^{gen}: electricity generated by unit $i \in \mathcal{G}$ in period t [kWh]
l_{it}: low-temperature heat generated by unit $i \in \mathcal{L}$ in period t [kWh]
h_{it}: high-temperature heat generated by unit $i \in \mathcal{H}$ in period t [kWh]
h_t^{down}: high-temperature heat downgraded to low-temperature in period t [kWh]
q_{it}: refrigeration energy generated by unit $i \in \mathcal{C}$ in period t [kWh]
e_t^-: electricity sold to the grid in period t [kWh]
e_t^+: electricity bought from the grid in period t [kWh]
u_t: high-temperature thermal energy stored at the beginning of period t [kWh]
v_t: low-temperature thermal energy stored at the beginning of period t [kWh]
w_t: refrigeration energy stored at the beginning of period t [kWh]
z_{it}: binary variable, on/off status of unit i in period t
δ_{it}: binary start-up variable ($\delta_{it} = 1$ if unit i is switched on at beginning of period t)

Using these sets, parameters and decision variables the short-term cogeneration systems planning problem can be formulated as the following MINLP:

$$\min \quad \sum_{t\in\mathcal{T}} \left(\sum_{i\in\mathcal{U}} c_i^{\mathsf{OM}} z_{it} + \sum_{i\in\mathcal{U}} c_i^\delta \delta_{it} + \sum_{i\in\mathcal{F}} c_i^f f_{it} + b_t e_t^+ - p_t e_t^- \right) \tag{10.1}$$

$$
\begin{aligned}
s.t.\quad & \sum_{i\in\mathcal{G}} e^{gen}_{it} - \sum_{i\in\mathcal{E}} e^{cons}_{it} + e^{+}_{t} - e^{-}_{t} = D^{e}_{t} && \forall t\in\mathcal{T} && (10.2)\\
& \sum_{i\in\mathcal{H}} h_{it} - h^{down}_{t} + (u_t - \frac{u_{t+1}}{1-\alpha}) \geq D^{high}_{t} && \forall t\in\mathcal{T}, i\in\mathcal{U} && (10.3)\\
& \sum_{i\in\mathcal{L}} l_{it} + h^{down}_{t} + (v_t - \frac{v_{t+1}}{1-\beta}) \geq D^{low}_{t} && \forall t\in\mathcal{T}, i\in\mathcal{U} && (10.4)\\
& \sum_{i\in\mathcal{C}} q_{it} + (w_t - \frac{w_{t+1}}{1-\gamma}) \geq D^{cold}_{t} && \forall t\in\mathcal{T}, i\in\mathcal{U} && (10.5)\\
& z_{it}F^{min}_{it} \leq f_{it} \leq z_{it}F^{max}_{it} && \forall t\in\mathcal{T}, i\in\mathcal{F} && (10.6)\\
& z_{it}E^{min}_{it} \leq e^{cons}_{it} \leq z_{it}E^{max}_{it} && \forall t\in\mathcal{T}, i\in\mathcal{E} && (10.7)\\
& \textit{Performance constraints} && \forall t\in\mathcal{T}, i\in\mathcal{U} && (10.8)\\
& \sum_{t\in\mathcal{T}} \delta_{it} \leq N_i && \forall t\in\mathcal{T}, i\in\mathcal{U} && (10.9)\\
& \delta_{it} \geq z_{it} - z_{it-1} && \forall t\in\mathcal{T}, i\in\mathcal{U} && (10.10)\\
& e^{gen}_{it}, h_{it}, l_{it}, q_{it}, h^{down}_{t}, e^{+}_{t}, e^{-}_{t} \geq 0 && \forall t\in\mathcal{T}, i\in\mathcal{U} && (10.11)\\
& 0\leq u_t \leq U,\ 0\leq v_t\leq V,\ 0\leq w_t\leq W && \forall t\in\mathcal{T} && (10.12)\\
& 0\leq \delta_{it}\leq 1,\ z_{it}\in\{0,1\} && \forall t\in\mathcal{T}, i\in\mathcal{U} && (10.13)
\end{aligned}
$$

The aim is to minimize the operational costs minus the revenue obtained by selling extra electricity to the grid. In the objective function (10.1), we consider unit-dependent fuel costs c^{f}_{i}. Start-up penalties c^{δ}_{i} account for the extra cost due to the the warm-up phase. The fixed cost c^{OM}_{i} accounts for Operation and Maintenance costs proportional to the number of working hours. It can include the cost of staff needed to operate and maintain the unit, or machine deterioration costs.

Constraints (10.2) are balance equations for electricity. The net amount of electric power, either generated or bought, must satisfy the demand D^{e}_{t} for period t. Note that some units generate electric power, while others consume electricity. It is necessary to separate energy that is purchased from the power grid, e^{+}_{t}, from the one that is sold, e^{-}_{t}, since their price is different. Constraints (10.3) are balance constraints for high-temperature heat. The requirement D^{high}_{t} for period t must be covered by the generated high-temperature heat and/or by that which is available in the storage (u_t). High-temperature heat can be downgraded to low-temperature. Thermal energy can be stored in the tank for the next period, as long as the capacity U is not saturated. Accordingly, the stored energy at the beginning of the following period will be:

$$
u_{t+1} = \min\left\{U,\ (1-\alpha)\left(\sum_{i\in\mathcal{H}} h_{it} - h^{down}_{t} + u_t - D^{high}_{t}\right)\right\},
$$

where $\alpha\in[0,1)$ is the constant deterioration rate for high-temperature heat. Thermal energy in excess can always be dissipated with no additional costs. Similarly, Constraints (10.4) are balance constraints for low-temperature heat, where we also include the high-temperature heat that has been downgraded to low-temperature one. Constraints (10.5) are balance constraints for the refrigeration units. Constraints (10.6) and (10.7) ensure that the operating variables for a unit i (fuel, consumed electricity) are within the technical minimum and maximum. Constraints (10.8), that model the nonlinear behaviour of the generation units, are described in detail in the next paragraph. Constraints (10.9) and (10.10) limit the number of startups in a day. Finally, Constraints (10.11)-(10.13) impose lower and upper bounds, and integrality for variables z_{it}. The integrality of variables δ_{it} is implied by Constraints (10.10) and the direction of the optimization.

Nonlinear performance constraints We have seen in Chapter 8 that each unit can described in terms of nonlinear functions $g_{it}(\cdot)$, usually continuous and non-decreasing, that map one or more operating variables (fuel, consumed electricity, supplementary fuel) to an output variable (low or high-temperature heat, refrigeration power, electric power). The performance curves are, in general, non-convex and time-varying due to the non-negligible temperature effect in each period t. In addition, if unit $i \in \mathcal{U}$ is off, its output has to be 0. Thus, the corresponding output variables are semi-continuous. The performance constraints for the generation units can then be expressed as equations of the form:

$$\zeta = z_{it} g_{it}(\underline{\theta})$$

where $\underline{\theta}$ is the vector of input variables, and ζ an output variable. The equation can be relaxed, obtaining an inequality, if it is not necessary to consider explicitly the amount of energy that is dissipated:

$$\zeta \leq z_{it} g_{it}(\underline{\theta}). \tag{10.14}$$

In the case of generation or cogeneration units with one degree of freedom, each performance curve g_{it} will be a function of one variable (θ is scalar). For instance, given a high-temperature auxiliary boiler, the output variable is high-temperature thermal power h_{it}, while the only operating variable is fuel f_{it}. The feasible region for h_{it} will be $\{0\} \cup [g_{it}(F_{it}^{min}), g_{it}(F_{it}^{max})]$.

In the case of cogeneration units with more degrees of freedom, the performance curves are functions of two or more operating variables ($\underline{\theta}$ is a vector). Two examples are combined cycles with extraction valve regulation (left) and gas turbines with post-firing (right):

$$\begin{cases} l_{it} \leq z_{it} g_{it}^{l}(f_{it}, x_{it}) \\ h_{it} \leq z_{it} g_{it}^{h}(f_{it}, x_{it}) \\ e_{it}^{gen} \leq z_{it} g_{it}^{e}(f_{it}, x_{it}) \end{cases} \quad (10.15) \qquad \begin{cases} l_{it} \leq z_{it} g_{it}^{l}(f_{it}, y_{it}) \\ h_{it} \leq z_{it} g_{it}^{h}(f_{it}, y_{it}) \\ e_{it}^{gen} \leq z_{it} g_{it}^{e}(f_{it}, y_{it}) \end{cases} \quad (10.16)$$

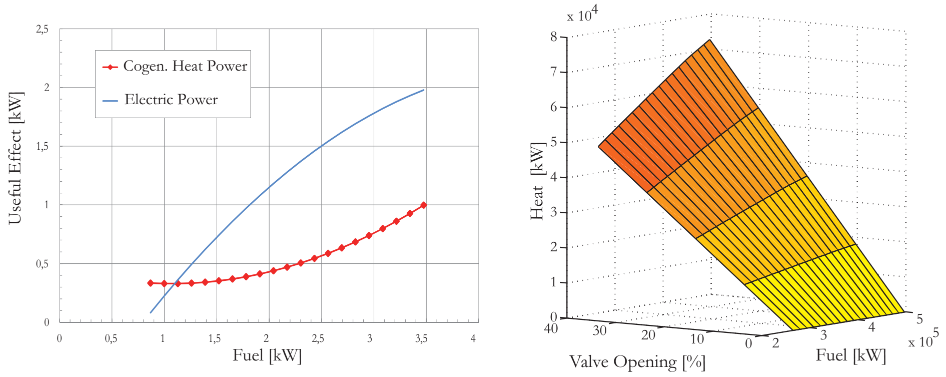


Figure 10.2: *(a) Useful effect (electric and thermal power) of a fuel cell unit as a function of consumed fuel. Note that the concavity is different: at larger loads the thermal efficiency increases, while the electrical efficiency decreases. The performance curves are derived from the data of a commercially available machine. (b) Heat as a function of fuel and extraction valve opening percentage for a natural gas combined cycle. Heat production is* 0 *when the valve is closed, and it increases with fuel when the valve is opened. Data obtained via simulation with the dedicated software* THERMOFLEX *[The14].*

where the operating variables are the fuel quantity f_{it}, the valve opening percentage $x_{it} \in [0, 0.4]$ for the combined cycle (10.15) and the supplementary fuel y_{it} for the gas turbine (10.16). The variable y_{it} has a positive cost that must be added to the objective function, and must satisfy an additional technical constraint $y_{it} \leq a + df_{it}$ with a, $d \geq 0$.

Piecewise linear approximation Nowadays, several global solvers able to deal with nonconvex MINLPs are available. However, computational results for relatively simple scenarios of the short-term operation planning problem indicate that even small-size instances can be very challenging for a MINLP solver, as we show in [TABM15]. The additional constraints and (binary) variables needed to capture additional typical features, would likely make the MINLP models even harder to solve. An alternative approach consists in approximating the nonlinear performance functions with piecewise linear functions, see e.g. [BTM$^+$14] or [ZLL$^+$13], obtaining a more tractable Mixed-Integer Linear Program (MILP). The piecewise linear approximation of 1-d.o.f. performance functions is rather straightforward, as it is sufficient to select a set of discretization points on a line, and connect them via line segments. For 2-d.o.f. units the approximation involves functions of two variables. Several approaches are available for approximating 2-D functions, differing considerably in terms of accuracy of the approximation and computational cost of the resulting MILP. In our model, we consider the so-called *lambda (or* *triangle) method* described, e.g., in [LW01] and [DLM10], that is implemented by triangulating the domain of the nonlinear function. Then, the value in a point $\underline{x}$ is computed as the convex combination of the function values in the vertices of the triangle containing $\underline{x}$. This method requires the introduction of $O(n_1 \times n_2)$ binary variables, where n_1 and n_2 are the number of discretization points per dimension.

10.3 Computational experiments with MINLP and MILP formulations

Let us now summarize computational experiments so as to compare the MINLP formulation with the approach based on the piecewise-linear approximation of the nonlinear functions. Given the wide variety of cogeneration systems, ranging from small to large scale, in our experiments we consider two scenarios: a domestic application (first scenario) with a few small-size cogeneration units, and an industrial application (second scenario) with a considerable number of larger-size cogeneration units.

10.3.1 Scenario 1

The first scenario is a micro-cogeneration system designed to provide thermal power, refrigeration power and electricity to a $2,000$ m^2 building. More in detail, the building has the following power requirements: high-temperature thermal power (hot water above 60 °C) for domestic hot water; low temperature thermal power (hot water $35 - 45$ °C) for heating; refrigeration power for air conditioning during summer period; electric power. The cogeneration system is made of the following units:

- a Solid Oxide Fuel Cell (SOFC) using natural gas to cogenerate up to 30kW and 15kW of, respectively, electric and thermal power;
- a Heat Pump (HP) using electric power to generate low temperature heat by "pumping" heat from ambient temperature up to 35-45 °C. It generates about 130kW at nominal conditions, but it is very sensitive to ambient temperature.
- an Auxiliary Boiler (AB) burning natural gas to generate up to 100kW of high-temperature heat;
- a thermal storage system to store up to 100kWh of high-temperature heat energy.

Figure 10.2 shows the performance curves of the SOFC units, i.e., the useful effects, heat and electric power as a function of the fuel input. Due to the fact that the thermal and electric request may have independent time profiles, the heat storage tank is essential in order to allow the cogeneration system to generate extra electric power (to be sold to the grid) when the selling price is higher without wasting the cogenerated heat (which will be stored and used when needed). The auxiliary boiler is included in the system mainly as a backup and it is capable to fulfill the requirement peaks of both high and low temperature heat.

10.3.2 Scenario 2

The second scenario is a large scale cogeneration system providing heat to a district heating network. The requirement is thermal power at one level of temperature, about 90 °C, while the whole electricity production is sold to the electric grid. The cogeneration system includes one or more of the following units[1]:

- Gas Turbines (GT) with heat recovery, burning natural gas to generate up to about 10MW of heat and 5.5MW of electricity;
- Gas Turbines (GT-2) with supplementary firing and heat recovery, burning natural gas to generate up to about 40MW of heat and 11MW of electricity;
- Natural Gas Combined Cycles (NGCC) with a bottoming back-pressure steam turbine, burning natural gas to generate up to 30MW of heat and 45MW of electricity;
- Natural Gas Combined Cycles (NGCC-2) with a bottoming extraction-type steam turbine, burning natural gas to generate up to about 70MW of heat and 30MW of electricity;
- Auxiliary Boilers (AB) burning natural gas to generate up to about 40MW of heat;
- a thermal storage system to store up to 50MWh of high-temperature heat energy.

The thermal power requirements are fulfilled by well established CHP units, like gas turbines and combined cycles, with the help of auxiliary boilers. This scenario includes cogeneration units with two degrees of freedom, namely, gas turbines (GT-2) with post-firing injection, and combined cycles with extraction condensing steam turbine (NGCC-2). In GT-2, it is possible to burn supplementary fuel to increase the amount of heat that can be recovered from the exhaust gases (10.16). In NGCC-2, the amount of cogenerated heat and electric power is a function of the consumed fuel and the opening of a steam extraction valve, as described in (10.15). Opening the valve reduces the electric power efficiency and increases the amount of recovered heat, while closing the valve drives heat production to 0 (see Figure 10.2), but provides larger electric output.

[1]We report nominal values at an ambient temperature of 15 °C.

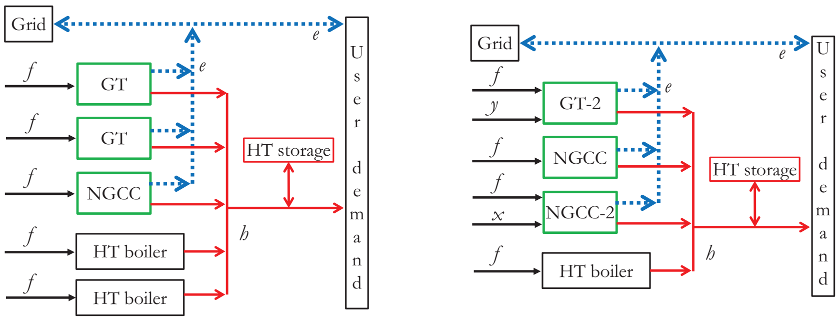


Figure 10.3: *Representation of instances `2-a` and `2-b`.*

10.3.3 Results

In Table 10.1, the type and number of (co)generation units contained in each instance are specified. The performance curves are obtained by fitting experimental or simulated data with quadratic functions, since the curves are generally quite smooth.

For scenario 1, we consider a single instance. For scenario 2, we consider four different unit configurations. In instances `2-a` and `2-b` (see Figure 10.3 for a schematic representation), heat demand is relatively low. In instances `2-c` and `2-d`, heat requirements are higher, and more units are necessary to fulfill them. Instances `2-b` and `2-d` include also cogeneration units with two degrees of freedom. All the instances have $n = 24$ time periods.

Table 10.1: *Type of units included in each instance. Input and output variables for each unit are indicated on the second row. For instance, unit NGCC produces heat power h and electric power e from fuel f.*

	HP	AB	SOFC	NGCC	NGCC-2	GT	GT-2	number
	$(f \rightarrow l)$	$(f \rightarrow h)$	$(f \rightarrow h, l, e)$	$(f \rightarrow h, e)$	$(f, x \rightarrow h, e)$	$(f \rightarrow h, e)$	$(f, y \rightarrow h, e)$	of units
`1-a`	1	1	1	-	-	-	-	3
`2-a`	-	2	-	1	-	2	-	5
`2-b`	-	1	-	1	1	-	1	4
`2-c`	-	4	-	4	-	4	-	12
`2-d`	-	4	-	2	1	2	2	11

Computational experiments were performed, for the MINLP formulations, with the open-source solver SCIP 3.1.0 [Ach09]), while for the MILP formulations IBM Ilog CPLEX 12.6 was used, both with default settings. For the MINLP, we have also experimented with BARON and Couenne, whose results are not included for sake of brevity, since their efficiency on the considered instances was inferior. The tests were carried out on an Intel Xeon with E3125@3.30GHz CPUs and 16GB of RAM, with a time limit of 2 hours.

Note that, while MILP solvers are quite mature and stable, dealing with MINLP problems requires some additional precautions. Compared to Formulation (10.1), the MINLP model used in the computational experiments is strengthened by adding valid bounds to all the decision variables, in order to help the spatial branch-and-bound. We also add valid inequalities that provides a rough approximation of the convex hull of the (non-convex) region defined by the performance constraints (10.14) if g_{it} is convex. The simple cutting plane is obtained in the input-output space for a unit i (see e.g., Figure 10.2), by connecting the extreme point $(F_{it}^{max}, g_{it}(F_{it}^{max}))$ either with the origin $(0,0)$ or with the point $\big(F_{it}^{min}, g_{it}(F_{it}^{min})\big)$. Although these valid inequalities are very simple, they appear to be of great help, in particular for instance `1-a`. Scaling is also essential, since the original data often contain values that are several orders of magnitude apart (e.g., generated energy with respect to cost coefficients), leading to numerical difficulties or, sometimes, even incorrect results.

Table 10.2: *Optimal values, computing time (seconds) and lower/upper bounds for the MINLP and the approximate MILP with an increasing number of discretization points (d.p.) per dimension.*

	2 d.p.		3 d.p.		5 d.p.		9 d.p.		MINLP			
	time	opt	time	opt	time	opt	time	opt	time	LB	UB	gap
`1-a`	0.01	94.33	0.03	91.73	0.06	91.30	0.10	91.17	42.58	91.07	**91.07**	0.0
`2-a`	0.04	104.17	0.07	102.49	0.10	101.95	0.12	101.86	812.8	101.76	**101.76**	0.0
`2-b`	0.06	-80.12	0.14	-80.24	1.19	-80.33	4.53	-80.36	7200	-94.36	-71.97	31.1
`2-c`	0.15	307.01	0.04	302.73	0.09	300.75	0.09	300.51	7200	284.83	300.19	5.4
`2-d`	0.22	121.27	0.29	119.43	0.31	118.20	0.50	118.09	7200	-83.01	140.93	∞

Table 10.2 summarizes the computational results. The MINLP instances turn out to be challenging. SCIP is able to certify optimality for 2 out of 5, and on `2-c` is close to the optimum, while the instances with 2-d.o.f. units are harder. In comparison, the MILP formulations can be solved to optimality by CPLEX in a few seconds. The MILP solutions are not necessarily feasible for the original formulation, since the approximate model might overestimate the amount that is generated. However, it is always possible to recover infeasibility *a posteriori* by increasing the production level. Interestingly, on our instances all the MILP optimal solutions are feasible (barring minor numerical errors), as they tend to be on the discretization points.

The results show that the approximate optimal values approach the optimal value of the original formulation as the number of discretization points is increased.

The optimal values are significantly different when the approximation is less accurate – except for instance `2-b`, where the variation is small, since the solution is dominated by a large NGCC-2 unit always at full load.

The structure of the solutions can differ significantly. As an example, we report in Figure 10.4 two optimal schedules for a low-temperature heat pump in instance `1-a` with a 2-point MILP approximation (left) and with the MINLP model (right). Although the instance is simple, the structure of the optimal approximate solution differs from that of the optimal MINLP solution even when the number of discretization points is increased to 5. To obtain optimal solutions to these two problems that are equivalent, one needs to use at least 9 discretization points.

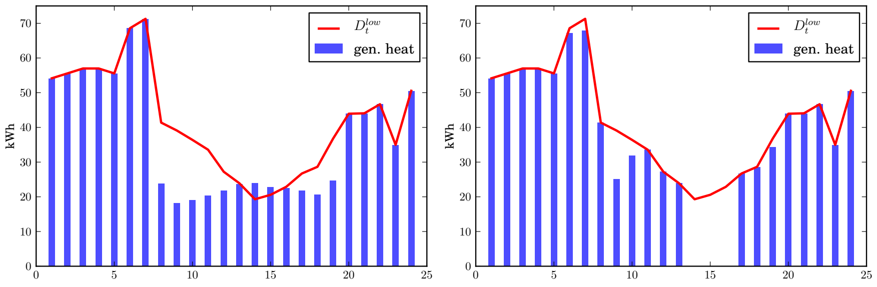


Figure 10.4: *Optimal plan for the heat pump of instance `1-a` obtained with a 2-point piecewise approximation (left) and with the MINLP (right).*

The results for two relatively simple scenarios of the short-term operational planning problem indicate that even small-size instances of the MINLP can be computationally very challenging. Tightening the formulation, along the lines of the methods proposed by Frangioni and Gentile in [FG05], might be of great help in improving the convergence of the MINLP solvers. Approximating the nonlinear performance functions with piecewise linear functions is an alternative that seems to work quite well in practice. For the considered instances, the resulting approximate MILP models can generally be solved more efficiently than their MINLP counterparts, and they appear to be already fairly accurate with a few linear pieces.

Attention must be paid to the feasibility of the solutions obtained with the approximations. Indeed, if the optimal operating point of a unit is far from the approximation discretization points, the piecewise linear function value might be quite different from the actual value. Underestimating the actual performance value may lead to suboptimal solutions with more than 3% error, while overestimating it may lead to infeasible solutions.

10.4 Operational planning with incentives

When dealing with real-world problems, it is often important to keep into account also environmental considerations (e.g., pollution management). Indeed, in recent years many governments have introduced laws that offer incentives on electricity and fuel prices if a plant satisfies certain requirements. As an example, recent Italian laws offer defiscalized prices if, over a time horizon of one year, the cogeneration units reach a specific efficiency. These constraints and incentives are often nonlinear, and expressed in a form that is very difficult to translate into a mathematical programming formulation[2]. We can usually approximate the constraints in a reasonable way with simpler inequalities that impose linear bounds on the overall efficiency of the system. An example of such constraint is the following:

$$\frac{1}{\eta_{th}} \sum_{t \in T} q_t + \frac{1}{\eta_{el}} \sum_{t \in T} e_t \geq \sum_{t \in T} f_t, \tag{10.17}$$

where η_{th} and η_{el} are reference efficiency values for heat and electricity generation. In essence, the constraint is imposing that the overall efficiency of the cogeneration system is sufficiently high. If the constraint is satisfied, then some incentives, that significantly reduce the overall cost, are offered. Specifically, the incentives consist of fiscal reductions that translate into reduced objective coefficience costs. In practice, the incentives can only be computed *a posteriori*, since they include several complex rules that cannot be incorporated in the model.

The incentives, and the associated constraints, are typically imposed over longer time horizons, e.g., a whole year. If one has to plan the operations over such a long timespan, it is necessary to adopt some heuristic to tackle the resulting large-scale problems. In the following section, we describe a heuristic algorithm that we have developed, and that has been adopted by a large Italian energy company to schedule the operations of residential buildings and hospitals.

10.4.1 A MILP-based rolling-horizon heuristic

The idea of the approach is to decompose the time horizon into smaller time frames that are computationally manageable. In our case, a week, with 1-hour time periods, can be solved in a fairly efficient way. Let us denote by $W = \{1, \dots, 52\}$ the set of weeks in a year and with T_i the 24 time periods in week i. Then, we solve sequentially the weeks in W. To ensure that the global efficiency constraints are satisfied when optimizing over a single week i, it is necessary to use an estimate of the contribution of the remaining weeks.

[2]As an example, we refer the (extremely) willing reader to the decree [DM11] (in Italian), and the associated guidelines, to be found on `www.gse.it`: in order to access the incentives reserved to high-efficiency cogeneration systems, a hierarchy of nonlinear criteria have to be met. The amount of the incentive is also highly nonlinear with respect to the total energy production during the year.

The algorithm is composed of two phases. We give a high-level description in Algorithm 10.1. In the first phase, we assume that we have a subset $S \subset W$ of *reference weeks*, with $|S| \ll |W|$, that are used to approximate the whole year[3]. We associate each week of the year $w \in W$ with one of the reference weeks, and denote by $\pi(i)$ the function that maps each week $i \in W$ to its reference week $j \in S$. For a given week i, where T_i are its time periods, let us denote the total amount of a variable over T_i by the aggregated variable $Q_i = \sum_{t \in T_i} q_t$. Then, we solve the *weighted problem*, which is an approximation of the complete yearly problem, where we optimize simultaneously all the reference weeks in S, each with a weight proportional to the number of weeks $i \in W$ that are associated with it. As an example, the Constraint (10.17) on cogeneration efficiency becomes:

$$\frac{1}{\eta_{th}} \sum_{j \in S} Q_j m_j + \frac{1}{\eta_{el}} \sum_{j \in S} E_j m_j \geq \sum_{j \in S} F_j m_j, \tag{10.18}$$

where $m_j = card\{i \in W : \pi(i) = j\}$ is the number of weeks in W that are associated with the sample week $j \in S$. At the end of the first phase, then, we have a plan for each of the reference week which satisfies approximately the yearly efficiency constraints.

Once we have solved the weighted problem, which is a first rough approximation of the complete yearly problem, the second phase begins, and we start iterating over the weeks $i \in W$. At each iteration, we solve sequentially each *rolling problem* i, defined as the problem where only the variables for week $i \in W$ are free and can be optimized, while the values in the previous and following weeks are considered to be fixed. The efficiency constraints (10.17) link together all the weeks in a year. Then, in such constraints, for each variable that does not belong to i, we act as follows:

1. for weeks $i' < i$, which have been optimized in the current iteration l, we consider the result of the last optimization phase for i' in the current iteration;
2. for weeks $i' > i$, which still have not been optimized in the current iteration, we have to consider an estimate of the optimal values. If $l > 1$, we consider the result of the optimization phase for i' that occured at the previous iteration $(l-1)$, while during the first iteration, we consider as an estimate of the future results the solution of the *weighted problem*.

[3]The set S can be obtained in several ways. One way to combine the construction of the sample weeks and the assignment of the weeks of the year is to consider it as a k-means clustering in the space of the parameters (hourly demands, efficiencies and costs), where each dimension corresponds to a parameter for a specific period t, and a week corresponds to a point. The result of a k-means approach will be k groups of weeks around k centroids (the reference weeks).

For a given week i, the efficiency constraint, that links all weeks in a year, becomes:

$$\frac{1}{\eta_{th}}(\sum_{i'<i} \bar{Q}_{i'}^{(l)} + \sum_{t\in T_i} q_t + \sum_{i''>i} \bar{Q}_{i''}^{(l-1)}) + \frac{1}{\eta_{el}}(\sum_{i'<i} \bar{E}_{i'}^{(l)} + \sum_{t\in T_i} e_t + \sum_{i''>i} \bar{E}_{i''}^{(l-1)})$$
$$\geq \sum_{i'<i} \bar{F}_{i'}^{(l)} + \sum_{t\in T_i} f_t + \sum_{i''>i} \bar{F}_{i''}^{(l-1)}, \quad (10.19)$$

where the terms involving variables that are not in week i are replaced by values previously obtained as described. In particular, we denote by $\bar{Q}_i^{(l)}, \bar{E}_i^{(l)}, \bar{F}_i^{(l)}$ the optimal aggregated values of the variables for week i at iteration l.

The values of the incentives are recomputed at the end of each iteration. The algorithm stops when the value of the global objective function (i.e., the sum of the optimal values of all the weeks) for two consecutive iterations is within a certain error $\epsilon > 0$.

In practice, the heuristic algorithm works well: convergence is fast, usually requiring no more than 10 iterations. The first iteration is the slowest, typically requiring between 1 and 5 minutes per week, whereas following iterations are fast, typically between 5 seconds and 1 minute per week, since our implementation allows the optimization for week i at iteration l to start from the optimal solution of the same week at iteration $l-1$ (warm start), which, in most cases, is already close to optimal. Note that, to obtain valid lower bounds, we can relax the problem simply disregarding the efficiency constraints, or dualizing them in a Lagrangian fashion.

Algorithm 10.1: MILP-based rolling horizon heuristic

```
Data: Reference weeks $S$, year weeks $W$
Assign each $i \in W$ to the closest reference week $j \in S$: $\pi(i) \leftarrow j$;
Solve the *weighted problem* obtaining the optimal solution $(q^*, e^*, f^*)$;
for $j \in S$ do
    Store the optimal values of reference week $j$:
        $\bar{Q}^*_j \leftarrow \sum_{t \in T_j} q^*_t$;
        $\bar{E}^*_j \leftarrow \sum_{t \in T_j} e^*_t$;
        $\bar{F}^*_j \leftarrow \sum_{t \in T_j} f^*_t$;
end
for $i \in W$ do
    Initialize estimates for week $i$ based on reference week $\pi(i)$:
        $\bar{Q}^{(0)}_i \leftarrow Q^*_{\pi(i)}$;
        $\bar{E}^{(0)}_i \leftarrow E^*_{\pi(i)}$;
        $\bar{F}^{(0)}_i \leftarrow F^*_{\pi(i)}$;
end
$l \leftarrow 1$;
while $error > \epsilon$ do
    for $i \in W$ do
        Solve the single-week problem for week $i$ with the efficiency
        Constraint (10.19);
        Store the optimal production levels for current week at iteration $l$:
            $\bar{Q}^{(l)}_i \leftarrow \sum_{t \in T_i} q^*_t$;
            $\bar{E}^{(l)}_i \leftarrow \sum_{t \in T_i} e^*_t$;
            $\bar{F}^{(l)}_i \leftarrow \sum_{t \in T_i} f^*_t$;
    end
    Compute incentive values;
    Compute error w.r.t. iteration $l - 1$;
    $l \leftarrow l + 1$;
end
```

10.5 Concluding remarks

In this chapter, we have focused on real-world cogeneration systems, for which the operational planning problem can be formulated as a mixed-integer nonlinear program with a number of additional technical constraints.

In real cases, the problem is very challenging for MINLP global solvers. A reasonable way to deal with instances of nontrivial size, is to use a piecewise linear approximation of the nonlinear functions in the model, which leads to MILP which can be solved quite efficiently with modern solvers. A possible route to investigate is whether the convergence of the exact global optimization methods, such as spatial branch-and-bound, can be improved with tighter formulation, e.g., adding valid inequalities that approximate the convex hull of the feasible region, at the point of being competitive with the piecewise linear approximation approach.

When the short-term operational problems have to be extended, e.g., to account for yearly incentives, we have devised a rolling-horizon MILP-based heuristic which handles large-scale problems quite efficiently, and the method has been adopted by an Italian energy company to schedule the weekly operations of large buildings and hospitals.

A natural extension of this work would involve taking into account uncertainty in the demands, and integrating the robust optimization approach considered in Chapter 9 into the complete formulation for operational planning of cogeneration systems.

Notation

LP	Linear Programming
MILP	Mixed-integer Linear Programming
MINLP	Mixed-integer Nonlinear Programming
$\mathbb{R}, \mathbb{R}_+$	set of (non-negative) real numbers
$\mathbb{Q}, \mathbb{Q}_+$	set of (non-negative) rational numbers
$\mathbb{Z}, \mathbb{Z}_+$	set of (non-negative) rational numbers
Part I	
MMF	Max-Min Fairness or Max-Min Fair, if used as an adjective
UFP	Maximum Unsplittable Flow Problem
UFP-MMF	Maximum Unsplittable Flow Problem subject to MMF
MB	Max-Bottleneck Fairness
r-MMF	Relaxed Max-Min Fairness with factor r
UFP-MB	Maximum Unsplittable Flow Problem subject to MB Fairness
OD	origin-destination pair
$G = (V, A)$	directed graph with nodes V and arcs A
(i, j)	directed arc connecting node i to node j
K	set of origin-destination pairs
$k = \|K\|$	number of origin-destination pairs (cardinality of the set K)
(s, t)	origin-destination pair with origin s and destination t
ϕ^{st}	total flow allocated to the (s, t) pair
c_{ij}	capacity of the arc (i, j)
f_{ij}^{st}	flow allocated to the pair (s, t) passing through the arc (i, j)
u_{ij}	upper bound on the flows allocated on the arc (i, j)
y_{ij}^{st}	binary bottleneck arc variable, if 1, arc (i, j) is bottleneck for (s, t)
x_{ij}^{st}	binary arc variable, if arc (i, j) is used by (s, t)
λ_{st}^{p}	binary path variable, if path p is used by (s, t)

Bibliography

[ABF11] E. Amaldi, M. Bruglieri, and B. Fortz, *On the hazmat transport network design problem*, Network Optimization, Springer, 2011, pp. 327–338. 11

[ACCG13] E. Amaldi, A. Capone, S. Coniglio, and L. G. Gianoli, *Network optimization problems subject to max-min fair flow allocation*, IEEE Communications Letters **17** (2013), no. 7, 1463–1466. 15, 31, 84, 93

[ACG$^+$10] M. Andrews, J. Chuzhoy, V. Guruswami, S. Khanna, K. Talwar, and L. Zhang, *Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs*, Combinatorica **30** (2010), no. 5, 485–520. 23

[Ach09] T. Achterberg, *SCIP: solving constraint integer programs*, Mathematical Programming Computation **1** (2009), no. 1, 1–41. 85, 176

[ACT14] E. Amaldi, S. Coniglio, and L. Taccari, *Maximum throughput network routing subject to fair flow allocation*, ISCO, Lecture Notes in Computer Science, vol. 8596, Springer, 2014, pp. 1–12. 15, 31, 84

[AdC03] F. Alvelos and J. V. de Carvalho, *Comparing branch-and-price algorithms for the unsplittable multicommodity flow problem*, International Network Optimization Conference, 2003, pp. 7–12. 46

[AK05] A. Atamtürk and S. Küçükyavuz, *Lot sizing with inventory bounds and fixed costs: Polyhedral study and computation*, Operations Research **53** (2005), no. 4, 711–730. 125

[AK08] ______, *An $O(n^2)$ algorithm for lot sizing with inventory bounds and fixed costs*, Operations Research Letters **36** (2008), no. 3, 297–299. 118, 127

[AKM05] T. Achterberg, T. Koch, and A. Martin, *Branching rules revisited*, Operations Research Letters **33** (2005), no. 1, 42–54. 49

[Alv05] F. P. Alvelos, *Branch-and-price and multicommodity flows*, Ph.D. thesis, Universidade do Minho, 2005. 41, 46

[AM03] R. Aringhieri and F. Malucelli, *Optimal operations management and network planning of a district heating system with a combined heat and power plant*, Annals of Operations Research **120** (2003), no. 1-4, 173–199. 115

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*, Prentice hall, 1993. 79

[AS04] M. Allalouf and Y. Shavitt, *Maximum flow routing with weighted max-min fairness*, Quality of Service in the Emerging Networking Panorama, Springer, 2004, pp. 278–287. 15

[AYZ95] N. Alon, R. Yuster, and U. Zwick, *Color-coding*, Journal of the ACM (JACM) **42** (1995), no. 4, 844–856. 68

[BA93] O. Ben-Ayed, *Bilevel linear programming*, Computers & operations research **20** (1993), no. 5, 485–501. 11

[BBC11] D. Bertsimas, D. B. Brown, and C. Caramanis, *Theory and applications of robust optimization*, SIAM review **53** (2011), no. 3, 464–501. 147

[BC89] J. Beasley and N. Christofides, *An algorithm for the resource constrained shortest path problem*, Networks **19** (1989), no. 4, 379–394. 68

[BDD06] N. Boland, J. Dethridge, and I. Dumitrescu, *Accelerated label setting algorithms for the elementary resource constrained shortest path problem*, Operations Research Letters **34** (2006), no. 1, 58–68. 68

[Ben79] J. Bentham, *An introduction to the principles of morals and legislation*, Clarendon Press, 1879. 4

[Ber95] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1, Athena Scientific Belmont, MA, 1995. 147

[BFT11] D. Bertsimas, V. F. Farias, and N. Trichakis, *The price of fairness*, Operations research **59** (2011), no. 1, 17–31. 15, 24, 25, 26

[BG92] D. P. Bertsekas and R. G. Gallager, *Data networks, 1992*, Prentice-Hall, 1992. 5, 7, 30

[BGG^{+}71] M. Benichou, J. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent, *Experiments in mixed-integer linear programming*, Mathematical Programming **1** (1971), no. 1, 76–94. 49

[BHK03] A. Björklund, T. Husfeldt, and S. Khanna, *Approximating longest directed path*, Electronic Colloq. on Comp. Complexity, Rept, no. 32, 2003. 68

[BJN^{+}98] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, *Branch-and-price: Column generation for solving huge integer programs*, Operations research **46** (1998), no. 3, 316–329. 41, 42, 46, 48

[BKK11a] C. Büsing, A. M. Koster, and M. Kutschka, *Recoverable robust knapsacks: γ-scenarios*, Network Optimization, Springer, 2011, pp. 583–588. 147

[BKK11b] ———, *Recoverable robust knapsacks: the discrete scenario case*, Optimization Letters **5** (2011), no. 3, 379–392. 147

[BL11] J. R. Birge and F. Louveaux, *Introduction to stochastic programming*, Springer, 2011. 147

[BLL^{+}09] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter, *Branching and bounds tighteningtechniques for non-convex minlp*, Optimization Methods & Software **24** (2009), no. 4-5, 597–634. 64

[BM90] J. F. Bard and J. T. Moore, *A branch and bound algorithm for the bilevel programming problem*, SIAM Journal on Scientific and Statistical Computing **11** (1990), no. 2, 281–292. 10

[BM92] J. F. Bard and J. T. Moore, *An algorithm for the discrete bilevel programming problem*, Naval Research Logistics (NRL) **39** (1992), no. 3, 419–435. 11

[BÖ08] D. Bienstock and N. Özbay, *Computing robust basestock levels*, Discrete Optimization **5** (2008), no. 2, 389–414. 147

[Boy10] M. P. Boyce, *Handbook for cogeneration and combined cycle power plants*, ASME International, 2010. 111

[BS04] D. Bertsimas and M. Sim, *The price of robustness*, Operations research **52** (2004), no. 1, 35–53. 146, 160

[BT04] D. Bertsimas and A. Thiele, *A robust optimization approach to supply chain management*, Integer programming and combinatorial optimization, Springer, 2004, pp. 86–100. 147, 148

[BT06] ______, *A robust optimization approach to inventory theory*, Operations Research **54** (2006), no. 1, 150–168. ii, 145, 147, 148, 150, 154, 162, 165

[BTEGN09] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust optimization*, Princeton University Press, 2009. 147

[BTGGN04] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, *Adjustable robust solutions of uncertain linear programs*, Mathematical Programming **99** (2004), no. 2, 351–376. 147

[BTGNV05] A. Ben-Tal, B. Golany, A. Nemirovski, and J.-P. Vial, *Retailer-supplier flexible commitments contracts: a robust optimization approach*, Manufacturing & Service Operations Management **7** (2005), no. 3, 248–271. 147

[BTGS09] A. Ben-Tal, B. Golany, and S. Shtern, *Robust multi-echelon multi-period inventory control*, European Journal of Operational Research **199** (2009), no. 3, 922–935. 147

[BTM$^+$14] A. Bischi, L. Taccari, E. Martelli, E. Amaldi, G. Manzolini, P. Silva, S. Campanari, and E. Macchi, *A detailed MILP optimization model for combined cooling, heat and power system operation planning*, Energy **74** (2014), 12 – 26. 115, 116, 167, 173

[BTN98] A. Ben-Tal and A. Nemirovski, *Robust convex optimization*, Mathematics of Operations Research **23** (1998), no. 4, 769–805. 146

[BTN00] ______, *Robust solutions of linear programming problems contaminated with uncertain data*, Mathematical programming **88** (2000), no. 3, 411–424. 146

[Büs11] C. Büsing, *Recoverable robustness in combinatorial optimization*, Cuvillier, 2011. 147

[BVRW84a] I. Barany, T. J. Van Roy, and L. A. Wolsey, *Strong formulations for multi-item capacitated lot sizing*, Management Science **30** (1984), no. 10, 1255–1261. 118

[BVRW84b] ______, *Uncapacitated lot-sizing: The convex hull of solutions*, Springer, 1984. 118, 125

[BW01] G. Belvaux and L. A. Wolsey, *Modelling practical lot-sizing problems as mixed-integer programs*, Management Science **47** (2001), no. 7, 993–1007. 118

[BY82] G. R. Bitran and H. H. Yanasse, *Computational complexity of the capacitated lot size problem*, Management Science **28** (1982), no. 10, 1174–1186. 118

[CA06] M. Carrión and J. M. Arroyo, *A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem*, Power Systems, IEEE Transactions on **21** (2006), no. 3, 1371–1378. 117

[CCG09] M. Chouman, T. G. Crainic, and B. Gendron, *A cutting-plane algorithm for multicommodity capacitated fixed-charge network design*, Tech. report, CIRRELT, 2009. 37, 42

[CCLW14] A. Caprara, M. Carvalho, A. Lodi, and G. J. Woeginger, *A study on the computational complexity of the bilevel knapsack problem*, SIAM Journal on Optimization **24** (2014), no. 2, 823–838. 11

[CJ89] D. Chiu and R. Jain, *Analysis of the increase and decrease algorithms for congestion avoidance in computer networks*, Computer Networks and ISDN systems **17** (1989), no. 1, 1–14. 14

[CKPT12] A. Christidis, C. Koch, L. Pottel, and G. Tsatsaronis, *The contribution of heat storage to the profitable operation of combined heat and power plants in liberalized electricity markets*, Energy **41** (2012), no. 1, 75–82. 115

[Cla84] A. Claus, *A new formulation for the travelling salesman problem*, SIAM Journal on Algebraic Discrete Methods **5** (1984), no. 1, 21–25. 36, 73

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, MIT press, 2001. 9

[CMS07] B. Colson, P. Marcotte, and G. Savard, *An overview of bilevel optimization*, Annals of operations research **153** (2007), no. 1, 235–256. 11

[Dan98] G. B. Dantzig, *Linear programming and extensions*, Princeton university press, 1998. 47

[Dem03] S. Dempe, *Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints*, Optimization **52** (2003), no. 3, 333–359. 11

[DeN11] S. DeNegre, *Interdiction and discrete bilevel linear programming*, Ph.D. thesis, Lehigh University, 2011. 11

[DFJ54] G. Dantzig, R. Fulkerson, and S. Johnson, *Solution of a large-scale traveling-salesman problem*, Operations Research **2** (1954), no. 4, 393–410. 69

[DH12] M. Dvořák and P. Havel, *Combined heat and power production planning under liberalized market conditions*, Applied Thermal Engineering **43** (2012), 163–173. 115

[DHK$^+$12] E. Danna, A. Hassidim, H. Kaplan, A. Kumar, Y. Mansour, D. Raz, and M. Segalov, *Upward max min fairness*, Proceedings IEEE INFOCOM, 2012, march 2012, pp. 837 –845. 15

[DHR99] E. Dotzauer, K. Holmström, and H. F. Ravn, *Optimal unit commitment and economic dispatch of cogeneration systems with a storage*, 13th PSCC Proceedings, 1999, pp. 738–744. 116

[DI14] M. Drexl and S. Irnich, *Solving elementary shortest-path problems as mixed-integer programs*, OR spectrum **36** (2014), no. 2, 281–296. 36, 68, 70

[DJK11] B. Dezső, A. Jüttner, and P. Kovács, *LEMON – an open source C++ graph template library*, Electronic Notes in Theoretical Computer Science **264** (2011), no. 5, 23–45. 79, 85

[DKKRA13] P. Damcı-Kurt, S. Küçükyavuz, D. Rajan, and A. Atamtürk, *A polyhedral study of ramping in unit commitment*, Tech. report, University of California, Berkeley, 2013. 125

[DKS89] A. Demers, S. Keshav, and S. Shenker, *Analysis and simulation of a fair queueing algorithm*, ACM SIGCOMM Computer Communication Review, vol. 19, ACM, 1989, pp. 1–12. 14

[DL05] J. Desrosiers and M. E. Lübbecke, *A primer in column generation*, Column Generation, Springer US, 2005, pp. 1–32. 45

[DLM10] C. D'Ambrosio, A. Lodi, and S. Martello, *Piecewise linear approximation of functions of two variables in milp models*, Operations Research Letters **38** (2010), no. 1, 39–46. 174

[DM11] *DM 5 settembre 2011 - Definizione del nuovo regime di sostegno per la cogenerazione ad alto rendimento*, `www.gazzettaufficiale.it/eli/id/2011/09/19/11A12047/sg`, September 2011, in Italian. 179

[DMS12] E. Danna, S. Mandal, and A. Singh, *A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering*, Proceedings IEEE INFOCOM, 2012, march 2012, pp. 846 –854. 15

[Dot97] Erik Dotzauer, *Algorithms for short-term production-planning of cogeneration plants*, 1997, Lic. Thesis, Linköping University. 116

[Dot01] ———, *Energy system operation by lagrangian relaxation*, Ph.D. thesis, Linköping University, 2001. 116

[Dre13] M. Drexl, *A note on the separation of subtour elimination constraints in elementary shortest path problems*, European Journal of Operational Research **229** (2013), no. 3, 595–598. 77, 79

[DW60] G. B. Dantzig and P. Wolfe, *Decomposition principle for linear programs*, Operations research **8** (1960), no. 1, 101–111. 45

[EO12] *Executive order 13626 – accelerating investment in industrial energy efficiency*, August 2012. 112

[FDGG04] D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen, *An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems*, Networks **44** (2004), no. 3, 216–229. 68

[FG92] J. J. Forrest and D. Goldfarb, *Steepest-edge simplex algorithms for linear programming*, Mathematical programming **57** (1992), no. 1-3, 341–374. 47

[FG05] A. Frangioni and C. Gentile, *Perspective cuts for a class of convex 0–1 mixed integer programs*, Mathematical Programming **106** (2005), no. 2, 225–236. 178

[FG06] ———, *Solving nonlinear single-unit commitment problems with ramping constraints*, Operations Research **54** (2006), no. 4, 767–775. 117, 135

[FGL09] A. Frangioni, C. Gentile, and F. Lacalandra, *Tighter Approximated MILP Formulations for Unit Commitment Problems*, IEEE Transactions on Power Systems **24** (2009), no. 1, 105–113. 117, 122

[FGL11] ———, *Sequential Lagrangian-MILP approaches for Unit Commitment problems*, International Journal of Electrical Power & Energy Systems **33** (2011), no. 3, 585–593. 117

[FGT98] M. Fischetti, J. J. S. Gonzalez, and P. Toth, *Solving the orienteering problem through branch-and-cut*, INFORMS Journal on Computing **10** (1998), no. 2, 133–148. 70

[FHW80] S. Fortune, J. Hopcroft, and J. Wyllie, *The directed subgraph homeomorphism problem*, Theoretical Computer Science **10** (1980), no. 2, 111–121. 20

[FK71] M. Florian and M. Klein, *Deterministic production planning with concave costs and capacity constraints*, Management Science **18** (1971), no. 1, 12–20. 128

[Flo91] S. Floyd, *Connections with multiple congested gateways in packet-switched networks part 1: One-way traffic*, ACM SIGCOMM Computer Communication Review **21** (1991), no. 5, 30–47. 14

[GG78] B. Gavish and S. C. Graves, *The travelling salesman problem and related problems*, Working Paper GR-078-78, Massachusetts Institute of Technology (1978). 71

[GKR$^+$03] V. Guruswami, S. Khanna, R. Rajaraman, B. Shepherd, and M. Yannakakis, *Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems*, Journal of Computer and System Sciences **67** (2003), no. 3, 473 – 496. 20

[GL10] O. Günlük and J. Linderoth, *Perspective reformulations of mixed integer nonlinear programs with indicator variables*, Mathematical Programming **124** (2010), no. 1-2, 183–205. 122

[GS97] D. T. Gardner and J. Scott Rogers, *Joint planning of combined heat and power and electric power systems: An efficient model formulation*, European Journal of Operational Research **102** (1997), no. 1, 58–72. 116

[HHSS13] T. Harks, M. Hoefer, K. Schewior, and A. Skopalik, *Routing games with progressive filling*, pre-print (2013), http://arxiv.org/abs/1308.3161. 27

[HHSS14] T. Harks, M. Hoefer, K. Schewior, and A. Skopalik, *Routing games with progressive filling*, INFOCOM, 2014 Proceedings IEEE, April 2014, pp. 352–360. 27

[HJS92] P. Hansen, B. Jaumard, and G. Savard, *New branch-and-bound rules for linear bilevel programming*, SIAM Journal on Scientific and Statistical Computing **13** (1992), no. 5, 1194–1217. 10

[HMM13] M. Haouari, N. Maculan, and M. Mrad, *Enhanced compact models for the connected subgraph problem and for the shortest path problem in digraphs with negative cycles*, Computers & Operations Research **40** (2013), no. 10, 2485–2492. 71

[HOO09] K. W. Hedman, R. P. O'Neill, and S. S. Oren, *Analyzing valid inequalities of the generation unit commitment problem*, Power Systems Conference and Exposition, 2009. PSCE'09. IEEE/PES, IEEE, 2009, pp. 1–6. 125

[HSS08] A. A. Hagberg, D. A. Schult, and P. J. Swart, *Exploring network structure, dynamics, and function using NetworkX*, Proceedings of the 7th Python in Science Conference (SciPy2008) (Pasadena, CA USA), August 2008, pp. 11–15. 94

[IMM09] M. Ibrahim, N. Maculan, and M. Minoux, *A strong flow-based formulation for the shortest path problem in digraphs with negative cycles*, International Transactions in Operational Research **16** (2009), no. 3, 361–369. 36, 72

[Jac88] V. Jacobson, *Congestion avoidance and control*, ACM SIGCOMM Computer Communication Review, vol. 18, ACM, 1988, pp. 314–329. 14

[JAM12] D. José Alem and R. Morabito, *Production planning in furniture settings via robust optimization*, Computers & Operations Research **39** (2012), no. 2, 139–150. 148

[Jer85] R. G. Jeroslow, *The polynomial hierarchy and a simple model for competitive analysis*, Mathematical programming **32** (1985), no. 2, 146–164. 10, 11

[JPS08] M. K. Jepsen, B. Petersen, and S. Spoorendonk, *A branch-and-cut algorithm for the elementary shortest path problem with a capacity constraint*, Tech. report, Department of Computer Science, University of Copenhagen, 2008. 68, 70

[JPSP14] M. K. Jepsen, B. Petersen, S. Spoorendonk, and D. Pisinger, *A branch-and-cut algorithm for the capacitated profitable tour problem*, Discrete Optimization **14** (2014), 78–96. 70

[JVT09] M. Jüdes, S. Vigerske, and G. Tsatsaronis, *Optimization of the design and partial-load operation of power plants using mixed-integer nonlinear programming*, Optimization in the energy industry, Springer, 2009, pp. 193–220. 116

[KL08] K. Kaparis and A. N. Letchford, *Local and global lifted cover inequalities for the 0–1 multidimensional knapsack problem*, European journal of operational research **186** (2008), no. 1, 91–103. 140

[Kle96] J. M. Kleinberg, *Approximation algorithms for disjoint paths problems*, Ph.D. thesis, Massachusetts Institute of Technology, 1996. 21

[KMT98] F. Kelly, A. Maulloo, and D. Tan, *Rate control for communication networks: shadow prices, proportional fairness and stability*, Journal of the Operational Research society **49** (1998), no. 3, 237–252. 14, 63

[Kol11] B. F. Kolanowski, *Small-scale cogeneration handbook, 4th edition*, Fairmont Press, 2011. 111

[KP99] E. Koutsoupias and C. Papadimitriou, *Worst-case equilibria*, STACS 99, Springer, 1999, pp. 404–413. 27

[KP06] V. Kaibel and M. A. Peinhardt, *On the bottleneck shortest path problem*, Tech. Report 06-22, ZIB, May 2006. 37

[KRT99] J. Kleinberg, Y. Rabani, and É. Tardos, *Fairness in routing and load balancing*, 40th Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 1999, pp. 568–578. 62

[KRT01] J. Kleinberg, Y. Rabani, and É. Tardos, *Fairness in routing and load balancing*, Journal of Computer and System Sciences **63** (2001), no. 1, 2–20. 9, 14

[LH03] R. Lahdelma and H. Hakonen, *An efficient linear programming algorithm for combined heat and power production*, European Journal of Operational Research **148** (2003), no. 1, 141–151. 115, 116

[LLM04] J. Lee, J. Leung, and F. Margot, *Min-up/min-down polytopes*, Discrete Optimization **1** (2004), no. 1, 77–85. 125

[Lov73] S. F. Love, *Bounded production and inventory models with piecewise concave costs*, Management Science **20** (1973), no. 3, 313–318. 118

[LPR96] Z.-Q. Luo, J.-S. Pang, and D. Ralph, *Mathematical programs with equilibrium constraints*, Cambridge University Press, 1996. 10

[LSO12] C. Losada, M. P. Scaparra, and J. R. O'Hanley, *Optimizing system resilience: a facility protection model with recovery time*, European Journal of Operational Research **217** (2012), no. 3, 519–530. 11

[LW01] J. Lee and D. Wilson, *Polyhedral methods for piecewise-linear functions I: the lambda method*, Discrete Applied Mathematics **108** (2001), no. 3, 269–285. 174

[Mar10] F. Margot, *Symmetry in integer linear programming*, 50 Years of Integer Programming 1958-2008, Springer, 2010, pp. 647–686. 41

[MB90] J. T. Moore and J. F. Bard, *The mixed integer linear bilevel programming problem*, Operations research **38** (1990), no. 5, 911–921. 11

[McC76] G. P. McCormick, *Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems*, Mathematical programming **10** (1976), no. 1, 147–175. 38

[Meg74] N. Megiddo, *Optimal flows in networks with multiple sources and sinks*, Mathematical Programming **7** (1974), no. 1, 97–107. 9, 14

[MELR13] G. Morales-España, J. M. Latorre, and A. Ramos, *Tight and compact milp formulation of start-up and shut-down ramping in unit commitment*, Power Systems, IEEE Transactions on **28** (2013), no. 2, 1288–1296. 125

[MGPA12] S. Mitra, I. E. Grossmann, J. M. Pinto, and N. Arora, *Optimal production planning under time-sensitive electricity prices for continuous power-intensive processes*, Computers & Chemical Engineering **38** (2012), 171–184. 115, 116

[MNS00] A. J. Miller, G. L. Nemhauser, and M. W. Savelsbergh, *On the capacitated lot-sizing and continuous 0–1 knapsack polyhedra*, European Journal of Operational Research **125** (2000), no. 2, 298–315. 125

[MR02] L. Massoulié and J. Roberts, *Bandwidth sharing: objectives and algorithms*, Networking, IEEE/ACM Transactions on **10** (2002), no. 3, 320–328. 14, 63, 64

[MTZ60] C. E. Miller, A. W. Tucker, and R. A. Zemlin, *Integer programming formulation of traveling salesman problems*, Journal of the ACM (JACM) **7** (1960), no. 4, 326–329. 70

[MW99] H. Marchand and L. A. Wolsey, *The 0-1 knapsack problem with a single continuous variable*, Mathematical Programming **85** (1999), no. 1, 15–33. 125, 140

[Nea00] P. J. Neame, *Nonsmooth dual methods in integer programming*, Ph.D. thesis, University of Melbourne, Department of Mathematics and Statistics, 2000. 87

[Nil06] P. Nilsson, *Fairness in communication and computer network design*, Ph.D. thesis, Lund University, Sweden, 2006. 5, 6, 9, 21

[NP08] D. Nace and M. Pióro, *Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial*, Communications Surveys & Tutorials **10** (2008), no. 4, 5–17. 4, 6

[NR02] D. Naddef and G. Rinaldi, *Branch-and-cut algorithms for the capacitated VRP*, ch. 3, pp. 53–84, SIAM Monographs on Discrete Mathematics and Applications Philadelphia, PA, 2002. 70

[OAV12] J. Ostrowski, M. F. Anjos, and A. Vannelli, *Tight mixed integer linear programming formulations for the unit commitment problem*, IEEE Transactions on Power Systems **27** (2012), no. 1, 39. 125

[OPT05] W. Ogryczak, M. Pióro, and A. Tomaszewski, *Telecommunications network design and max-min optimization problem*, Journal of Telecommunications and Information Technology **3** (2005), 1–14. 6, 9

[OŚ06] W. Ogryczak and T. Śliwiśki, *On direct methods for lexicographic min-max optimization*, Computational Science and Its Applications - ICCSA 2006, Lecture Notes in Computer Science, vol. 3982, Springer Berlin Heidelberg, 2006, pp. 802–811. 9

[OW04] W. Ogryczak and A. Wierzbicki, *On multi-criteria approaches to bandwidth allocation*, Control and Cybernetics **33** (2004), 427–448. 6

[OWPT10] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, *SNDlib 1.0 - survivable network design library*, Networks **55** (2010), no. 3, 276–286. ii, 77, 84

[Pad04] N. P. Padhy, *Unit Commitment – A Bibliographical Survey*, IEEE Transactions on Power Systems **19** (2004), no. 2, 1196–1205. 116

[Pap03] C. H. Papadimitriou, *Computational complexity*, John Wiley and Sons Ltd., 2003. 11

[Por02] E. L. Porteus, *Foundations of stochastic inventory theory*, Stanford University Press, 2002. 147

[PR93] M. Parker and J. Ryan, *A column generation algorithm for bandwidth packing*, Telecommunication Systems **2** (1993), no. 1, 185–195. 49

[PS91] M. Padberg and T.-Y. Sung, *An analytical comparison of different formulations of the travelling salesman problem*, Mathematical Programming **52** (1991), no. 1-3, 315–357. 74, 75

[PW06] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*, Springer, 2006. ii, 117, 124

[Raw71] J. Rawls, *A theory of justice*, Harvard university press, 1971, Revised in 1999. 4

[RB07] B. Radunovic and J.-Y. L. Boudec, *A unified framework for max-min and min-max fairness with applications*, IEEE/ACM Transactions on Networking **15** (2007), no. 5, 1073 –1083. 6

[RHL08] A. Rong, H. Hakonen, and R. Lahdelma, *A variant of the dynamic programming algorithm for unit commitment of combined heat and power systems*, European Journal of Operational Research **190** (2008), no. 3, 741–755. 116

[RL07a] A. Rong and R. Lahdelma, *An effective heuristic for combined heat-and-power production planning with power ramp constraints*, Applied Energy **84** (2007), no. 3, 307–325. 116

[RL07b] ———, *An efficient envelope-based Branch and Bound algorithm for non-convex combined heat and power production planning*, European Journal of Operational Research **183** (2007), no. 1, 412–431. 116

[RLG09] A. Rong, R. Lahdelma, and M. Grunow, *An improved unit decommitment algorithm for combined heat and power systems*, European Journal of Operational Research **195** (2009), no. 2, 552–562. 116

[RLL08] A. Rong, R. Lahdelma, and P. B. Luh, *Lagrangian relaxation based algorithm for trigeneration planning with storages*, European Journal of Operational Research **188** (2008), no. 1, 240–257. 117

[RS06] G. Righini and M. Salani, *Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints*, Discrete Optimization **3** (2006), no. 3, 255–273. 68

[RT05] D. Rajan and S. Takriti, *Minimum up/down polytopes of the unit commitment problem with start-up costs*, Tech. report, IBM, 2005. 125

[SA94] H. D. Sherali and W. P. Adams, *A hierarchy of relaxations and convex hull characterizations for mixed-integer zero—one programming problems*, Discrete Applied Mathematics **52** (1994), no. 1, 83–106. 42

[SB08] R. M. Salles and J. A. Barria, *Lexicographic maximin optimisation for fair bandwidth allocation in computer networks*, European Journal of Operational Research **185** (2008), no. 2, 778–794. 15

[Sch03] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*, Springer, 2003. 37

[Scu03] M. G. Scutellà, *An approximation algorithm for computing longest paths*, European Journal of Operational Research **148** (2003), no. 3, 584–590. 68

[SFT$^+$05] G. Sandou, S. Font, S. Tebbani, A. Hiret, and C. Mondon, *Short term optimization of cogeneration systems considering heat and electricity demands*, Proc. 15th Power Systems Computation Conference (PSCC), 2005. 116

[Soy73] A. L. Soyster, *Convex programming with set-inclusive constraints and applications to inexact linear programming*, Operations research **21** (1973), no. 5, 1154–1157. 146

[SP08] F. Salgado and P. Pedrero, *Short-term operation planning on cogeneration systems: A survey*, Electric Power Systems Research **78** (2008), no. 5, 835–848. 115

[SS10] C.-T. See and M. Sim, *Robust approximation to multiperiod inventory management*, Operations research **58** (2010), no. 3, 583–594. 147

[SV91] T. Seeger and J. Verstege, *Short term scheduling in cogeneration systems*, Power Industry Computer Application Conference, 1991. Conference Proceedings, IEEE, 1991, pp. 106–112. 115

[TABM15] L. Taccari, E. Amaldi, A. Bischi, and E. Martelli, *Short-term planning of cogeneration energy systems*, Advances and Trends in Optimization with Engineering Applications, SIAM, Philadelphia, PA, 2015, to appear. 167, 173

[Tac14] L. Taccari, *Integer programming formulations for the elementary shortest path problem*, Optimization Online. 67

[TAMB15] L. Taccari, E. Amaldi, E. Martelli, and A. Bischi, *Short-term planning of cogeneration power plants: a comparison between MINLP and piecewise-linear MILP formulations*, Computer Aided Chemical Engineering, vol. 37, 2015, pp. 2429–2435. 167

[Tar74] R. E. Tarjan, *A note on finding the bridges of a graph*, Information Processing Letters **2** (1974), no. 6, 160–161. 23

[The14] Thermoflow, *Thermoflex 24 – design and simulation of power plants*, 2014. 173

[Tom05] A. Tomaszewski, *A polynomial algorithm for solving a general max-min fairness problem*, European Transactions on Telecommunications **16** (2005), no. 3, 233–240. 6, 31

[VC94] L. N. Vicente and P. H. Calamai, *Bilevel and multilevel programming: A bibliography review*, Journal of Global optimization **5** (1994), no. 3, 291–306. 10, 11

[VHW96] C. Van Hoesel and A. P. M. Wagelmans, *An $O(T^3)$ algorithm for the economic lot-sizing problem with constant capacities*, Management Science **42** (1996), no. 1, 142–150. 118

[VS52] H. Von Stackelberg, *The theory of the market economy*, Oxford University Press, 1952. 10, 16

[Wol06] L. A. Wolsey, *Lot-sizing with production and delivery time windows*, Mathematical Programming **107** (2006), no. 3, 471–489. 127

[Won80] R. Wong, *Integer programming formulations of the travelling salesman problem*, Proceedings IEEE Conference on Circuits and Computers, 1980, pp. 149–152. 36, 73

[WVHK92] A. Wagelmans, S. Van Hoesel, and A. Kolen, *Economic lot sizing: an $O(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case*, Operations Research **40** (1992), no. 1-supplement-1, S145–S156. 118

[WW58] H. M. Wagner and T. M. Whitin, *Dynamic version of the economic lot size model*, Management science **5** (1958), no. 1, 89–96. 117

[WY90] U.-P. Wen and Y. Yang, *Algorithms for solving the mixed integer two-level linear programming problem*, Computers & Operations Research **17** (1990), no. 2, 133–142. 11

[YXF$^+$13] D. Yang, G. Xue, X. Fang, S. Misra, and J. Zhang, *A game-theoretic approach to stable routing in max-min fair networks*, IEEE/ACM Transactions on Networking (TON) **21** (2013), no. 6, 1947–1959. 27

[ZLL$^+$13] Z. Zhou, P. Liu, Z. Li, E. N. Pistikopoulos, and M. C. Georgiadis, *Impacts of equipment off-design characteristics on the optimal design and operation of combined cooling, heating and power systems*, Computers & Chemical Engineering **48** (2013), 40–47. 115, 173

[ZZ13] L. Zhao and B. Zeng, *An exact algorithm for two-stage robust optimization with mixed integer recourse problems*, preprint. 147