DYNAMIC FLOW PROBLEMS WITH BOUNDED NUMBER OF PATHS: MODELS, ALGORITHMS AND APPLICATIONS

Dept. of Computer, Control, and Management Engineering
"Antonio Ruberti"
ABRO PhD - Operations Research Curriculum
XXXI Cycle

DYNAMIC FLOW PROBLEMS WITH BOUNDED NUMBER OF PATHS: MODELS, ALGORITHMS AND APPLICATIONS

Candidate: Anna Melchiori
Advisor: Dr. Antonino Sgalambro

October 2018

Contents

1. Introduction

1.1. Motivation

Since its inception, Network Flow theory has been recognized as a powerful decision support toolbox for all those real-world operations and contexts suitable to be represented as flows traversing a node-arc network. This abstraction can take place for an extremely wide range of fields, starting from those having by reason of their nature a network structure, such as transportation and distribution logistics, where flows of people and freights travel throughout road, rail, sea and air networks, and telecommunication, where data are managed and sent across server-based networks. In addition to them, other different contexts can be translated in these terms, leading to an efficient and well-structured modellization of their operations. This is the case as instance of production planning, where the manufacturing process can be viewed as items flowing over the supply network, going for the raw stage to the finalization step.

Scientific contributions in Network Flow theory allow for an effective representation and resolution of complex network-based operations, where the best solutions, w.r.t. different measures of quality, must be identified among a combinatorial large number of feasible alternatives, while taking into account network structural limitations, such as finite capacities on the arcs, and possibly other specific decisional constraints. This theory has been enriched over years with original and cutting-edge optimization models and algorithms, all of them contributing to improve the adherence of these optimization tools to reality, securing an actionable decisional support in a increasingly amount of practical situations.

In particular, considerable results have been achieved in the Sixties with the introduction in the Network Flow theory of the so-called *dynamic flows*, often referred to as *flows over time*, designed to integrate the time dimension into flow modeling on networks. Indeed, classical contributions, being them based on the concept of *static flows*, are well fit to represent average steady flows behavior, but fail to capture the evolution of dynamic phenomena occurring over a considered time horizon, hence preventing an efficient modellization and optimization of time-driven operations. In the last decades, monitoring the transitional regime of the system over time has become a crucial and common requirement in many real-life situations, as it can lead to dramatically better results in terms of performances of the overall implemented processes, quality of the provided services and customer responsiveness among others. All of these come at a price of a considerably higher level of complexity in the planning and management of the operations to be performed. In this context, dynamic flows represent a refined and advanced optimization tool that allow to rigorously capture the position of flows at each point in time and to retrieve informations related to the network level utilization

over time.
A relevant family of dynamic flow optimization problems is the Quickest Flow class, where the completion time for transshipment operations is to be minimized on a network with limited inflow capacities and transit times on the arcs. The practical interest in this class of problems is supported by a vast scientific literature of real-life applications, sharing the common requirement of accomplishing complicated processes in the shortest possible time. This covers operational and tactical planning activities in the transportation field, particularly in emergency management when aiming at devising fast and secure evacuation plans, see as instance [61, 58, 87, 75], and in telecommunication [27], where quick routing of data packets are often preferred to contain energy consumptions and network management costs. Limitations of Quickest Flows as a planning tool are sometimes associated with the optimal solutions requiring the use of a large and unrestricted number of active paths in the network, which is unrealistic for practical purposes due to the frequent limitations in the availability of resources and, in some cases, to the additional requirement of a tight supervision on the operations to be implemented. This poses the need for encompassing realistic restrictions based on bounding the number of support paths to be activated for flows transshipment.
From a methodological point of view, such additional feature translates into the integration of the well-known concept of k-splittable flows, securing a transshipment process through at most a prefixed number k of paths, within the dynamic setting of network flows.
A large amount of real-life situations are expected to benefit from the resulting framework, as a concurrent control on both the time spent for flow transshipment and on the number of the activated support paths would be enabled. As instance, this might result significantly effective in distribution logistics, in particular in tactical planning of time-efficient goods dispatching, where the medium level of details involved, resembling network flow modeling rather than a more articulated vehicle routing framework, still requires to account for the limited number of available vehicles. Other major examples are represented by emergency transportation management, as enhancing path control in supervised evacuations might avert fatal congestion episodes hence increasing the safety of the process, and by telecommunication, where a more efficient quick data-packets routing is achieved by imposing a limited number of paths, thus preventing the overload of used devices and protocols.
Despite this great potential, the combination of k-splittable and dynamic flows had not been addressed at all in the scientific literature so far, except for a very preliminary work [85], whereas a considerable number of contributions addressing k-Splittable Flow Problems in the static setting can be found. Such research gap leaves thus a large room for original contributions that could substantially improve the impact of the Quickest Flow class, enabling a complete release of its potential as an optimization support tool by accounting for k-splittable path limitations.
Our research activity in developing this thesis is specifically devoted to fill this gap, introducing and studying the first dynamic k-splittable flow problem. In particular, we design a mathematical optimization model based on path-flows that explicitly bounds to k the maximum number of paths that can be used in a quickest multicommodity flow transshipment. The strongly NP-hard time complexity of the proposed problem and the exponential number of path-related variables in the provided formulation pose a number of questions for the design of efficient resolution strategies, in particular when aiming at tackling practical real-life situations often translating into extremely large instances and strict requirements related to

the quality of the solutions to be obtained. These issues have been addressed in our research activity by developing two novel ad-hoc resolution approaches, being the first a matheuristic scheme and the second a Branch and Price exact algorithm. Both strategies turn out to be based on the resolution of restricted versions of the path-flow formulation of the problem, where only a limited number of variables are accounted whose quality is heuristically or exactly proved, respectively. This way, through the development of the matheuristic approach, we secure good quality solutions in reasonable computational time even for very large-sized and highly congested instances, while with the design of an exact resolution tool we provide support to all those variety of real-life situations where a precise quality measure is significant for an actual and more conscious implementation of the identified solutions. Further details on the novel introduced optimization problem and on the developed algorithms are provided in the next section.

1.2. Contribution

In this thesis we investigate the concept of path limitations, expressed as the maximum number of paths that can be activated by each distinct commodity, hence secured by the so-called k-splittable flows, within the dynamic framework of Network Flow theory. In particular, we formally introduce the first flows over time problem explicitly encompassing path number restrictions, namely the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$). The problem asks for routing and scheduling each commodity demand through at most k different paths while minimizing the makespan of the overall process, i.e. the time required to accomplish the transshipment operations. We model the $QMCkSFP$ in a dynamic digraph with time-independent structural features, considering flow traveling over a discretized time horizon, and motivated by practical applications, we force the transshipment operations to be performed only through elementary/loopless paths and without flow storage at intermediate nodes. In this setting, we provide a path-flow Mixed-Integer Linear Programming formulation for the problem that requires as an input the complete set of available paths for each distinct commodity. In detail, fractional variables are employed to track release of flows at each point in time for each commodity and path; a first set of binary variables allows for the identification and linear minimization of the makespan by recording arrival times at destination of each routed commodity demand, while a second set of binaries imposes k-splittable path limitations to each commodity. The formulation presents a number of columns which grows exponentially with the instance size, due to the presence of variables associated with the number of paths for each commodity multiplied by the number of instants in the discretized time horizon.
The computational time complexity of the introduced problem is formally investigated, proving its belongs to the class of strongly NP-hard problems by a reduction from the static Minimum Cost k-Splittable Flow Problem.
When it comes to solution approaches, the first algorithm designed for the ad-hoc resolution of the $QMCkSFP$ is an improvement matheuristic hybridizing a Very Large-scale Neighborhood Search ($VLNS$) with a mathematical programming strategy in its exploration routine. The method employs a state-of-the-art ranking procedure to generate a sufficiently large list of good and promising candidate paths to be used throughout its implementation. In par-

ticular, the initial solution is constructed by selecting for each commodity the k top-ranked paths from the generated list w.r.t. their transmission time, i.e. the time required to tranship the entire commodity demand. The $QMCkSFP$ path-flow formulation is restricted to these initial paths and solved to optimality via a MIP solver. Then, at each improvement iteration of the matheuristic, a different neighborhood is constructed by heuristically identifying for each commodity a collection of candidate paths from the list and it is explored to optimality by solving the related restricted path-flow formulation. The improvement search proceeds through a Variable Neighborhood Descent scheme by increasing the cardinality of the path collections to be identified till a time limit is reached.
A thorough computational experience is conducted on the proposed model and matheuristic approach, starting by the resolution to optimality of a set of very reduced-size instances, making use of the path-based formulation fed with a complete enumeration of all the available paths. A tuning process is then carried out on the matheuristic's parameters for the identification of the optimal setting to be adopted in the evaluation of its performance. An exhaustive proof-of-concept for the correctness and effectiveness of the matheuristic resolution strategy is then conducted: to this aim, a comparison is performed on a set of small to medium-sized instances against the Quickest Multicommodity Flow Problem, i.e. the free-flow relaxation of the problem with no upper bounds on the number of usable paths. Finally, we test performances of our matheuristic against those of a Randomized Rounding-based algorithm on two different benchmark test sets, the first one collecting networks of large size w.r.t. number of nodes and arcs, and the second networks with an extremely high number of commodities.
Our second algorithmic contribution is represented by an exact strategy for solving the $QMCkSFP$ at the optimum. The developed technique falls within the Branch and Price paradigm and is based on original relaxation, pricing and branching procedures. In particular, the so-called Restricted Relaxed Master Problem ($RRMP$) is obtained by linearizing time-arrival binary variables and by substituting those related to k-splittable path restrictions from the path-based formulation of the problem. The same ranking algorithm employed in the matheuristic initialization phase is here adopted for the construction of the path-related columns to be considered at the root node of the Branch and Bound decisional tree. In particular, the best k paths w.r.t. the transmission time are generated for each commodity and all related fractional variables are included as initial columns. The Column Generation procedure is then applied to identify, through the resolution of the so-called pricing problem, new entering variables defined as pair (path,departure time), or to certify the optimality of the nodes' $RRMP$s. This is done on a slightly modified time-expansion of the original dynamic digraph where the time dimension is decoupled from the network structure to give rise to a novel equivalent static digraph.
Two original branching rules are designed for restoring feasibility. A first one, performed whenever k-splittable flow constraints are violated, forces the usage of some paths and simultaneously forbids the activation of others over the discretized time horizon. Working on path-based variables, the generated branching cuts are included in $RRMP$s of the child nodes. A second rule implements a refined version of the standard 0-1 branching on fractional variables by selecting the candidate time-arrival variable at the highest time instant.
In order to account for k-splittable-related branching cuts, the pricing problem is modeled as a Shortest Path Problem with Forbidden Paths where additional node-set resources are

introduced to secure the generation of loopless paths. A tailored labelling algorithm based on dynamic programming is designed and employed to solve the pricing problem to optimality. All variables modeling the utilization of the optimal identified path at each point in time are included as new columns.
The computational experience conducted on the designed Branch and Price algorithm begins by studying its correctness and computational performances when applied on a collection of small to medium-sized instances. As the final experiment, we employ the Branch and Price approach as a tool to assess the quality of the solutions obtained by the developed $VLNS$-based matheuristic. To this aim, we consider the same testbed of small to medium-sized instances used in the matheuristic experiments with our Branch and Price, feeding it with the matheuristic results as initial incumbent solutions, then measuring and comparing the residual optimality GAP achieved after the execution of the exact algorithm.
The overall toolbox developed in this research lays the foundations for a number of refined and additional novel contributions within the dynamic k-splittable flow setting, whose preliminary guidelines are discussed in the closing chapter of this thesis.

1.3. Organization of the thesis

In this section we provide a brief summary of the contents presented in this thesis.
Chapter 2 is thought to introduce the reader to the general scientific context over which we operate. In particular, we provide preliminary notions and notations of Network Flow theory, starting from the definition of *static* flows on digraphs and their traditional mathematical representations. A brief excursus on relevant static flow problems is then followed by a discussion on the limitations of such modeling approaches w.r.t. their capability in capturing the potential evolution over time of the process to optimize.
In **Chapter 3** we focus on the specific class of network flow problems known as *dynamic flow problems* or *flows over time*. In contrast to static flows, they incorporate the temporal dimension in flow modeling, hence enabling a precise tracking of flows over a considered time horizon. A thorough review of the state-of-the-art contributions in the field is provided, highlighting the impact of the time parameter in the computational tractability of the problems. The required network flow setting is presented, specifically introducing the so-called dynamic digraph structure and its time-related features. Significant arc- and path-flow formulations follow, together with a general algorithm based on a time-expansion procedure and an efficient quickest path ranking strategy, both of which will be employed in the original contributions presented in the thesis. Finally, a dedicated section is reserved to the class of *Quickest Flow Problems* addressing the minimization of the time horizon required to accomplish flow transshipments in capacitated dynamic networks. The scientific literature provides evidence of the relevance of this class, presenting a large variety of quickest flow optimization problems and showing promising room for further important contributions.
Chapter 4 is devoted to the concept of *k-splittable flows*, a well-known modeling tool that allows for the representation of path limitations in flow networks. More precisely, it explicitly bounds to k the maximum number of paths that can be used for flow transshipment. The large variety of network flow problems studied in this context are presented and analyzed in-depth in the literature overview section. General models and a significant approximation

result are then presented. We conclude this chapter by emphasizing the scientific interest on k-splittable flows highlighting in particular the lack of related contributions within the field of flows over time.
The previous discussions serve as a basis for introducing the core of our research activity that concerns the design of a modeling toolbox integrating k-splittable flows within the dynamic environment of network flows. **Chapters 5, 6**, and **7** are thus devoted to the presentation of the original developed contributions starting from the novel dynamic flow optimization problem, namely the *Quickest Multicommodity k-splittable Flow Problem (QMCkSFP)*, that explicitly accounts for path limitations while performing a quickest multicommodity flow transshipment over a discretized time horizon, for which we provide a path-based Mixed-Integer Linear Programming mathematical formulation and prove its strongly NP-hardness complexity. Then, we proceed with the discussion of our research production presenting the original ad-hoc resolution techniques: a *Very Large-scale Neighborhood Search-based matheuristic* and a *Branch and Price approach*, respectively. For each of the algorithms we provide a detailed description of all the developed tailored procedures and present a thorough computational experience conducted to test their correctness and to evaluate their performances also against those of competing approaches from the state-of-the-art literature. We refer the reader to the previous section for a richer preliminary introduction to our original contributions.
Finally, in **Chapter 8** we review and summarize the presented research outcome highlighting future research lines that could be pursued from it.

2. Preliminaries of Network Flow theory

2.1. Introduction

Network Flow theory is a branch of Combinatorial Optimization dealing with the mathematical modellization of flows on networks. Besides being a major topic of the utmost scientific relevance in the context of mathematical optimization, network flows represent a powerful toolbox to support decision making as many real-world situations can be easily translated in terms of flows on network structures. For instance, in the recent book of Nagurney et al., some relevant novel applications of Network Flows to blood and medical nuclear supply chains are presented [93].
In this chapter we present some basic notions of this theory by introducing in Section 2.2 the required notation and the concept of static flows on networks and in Section 2.3 their classical mathematical representations. In the last Section 2.4 we briefly discuss significant static network flow optimization problems and conclude the chapter by highlighting some of their relevant limitations. Our aim is to introduce the reader to the general scientific framework of our research providing preliminary fundamentals of network flows that will be required for an understanding of the novel developed contributions. For a wider overview on static network flow problems we refer to the book of Ahuja et al. and the seminal works by Ford and Fulkerson [1, 39, 38].

2.2. Notation

In this section we provide the notation that will be used throughout the thesis and we summarize it in a box. A *flow network* is defined as a digraph $\mathcal{D} = (\mathcal{V}, \mathcal{A})$ composed of a set $\mathcal{V}$ of n nodes and a set $\mathcal{A}$ of m directed arcs, each of them associated to a non-negative *capacity* c_{ij} expressing the maximum number of units of flows which can enter it. Node i of a given arc (i, j) is called the *tail* of the arc while node j its *head*. The tails of arcs entering node i are collected in the set $\delta^{-}(i) = \{j \in \mathcal{V} \,|\, (j, i) \in \mathcal{A}\}$ while the heads of outgoing arcs in $\delta^{+}(i) = \{j \in \mathcal{V} \,|\, (i, j) \in \mathcal{A}\}$. A *commodity* is a tuple (o, d, σ) representing a specified amount of demand σ that is generated at a *source* node o and is required at a *sink* node d. In the case of multiple commodities, we indicate and collect them as (o_h, d_h, σ_h), $h \in \mathcal{H}$. An o-d path is an ordered set of arcs $\{(i_0, i_1), (i_1, i_2), (i_2, i_3), \dots, (i_{k-1}, i_k)\}$ such that $i_0 = o$, $i_k = d$ and $(i_l, i_{l+1}) \in \mathcal{A}$, $\forall l = 0, \dots, k-1$. The arcs of a path p are identified by the binary parameters δ_{ij}^{p}, taking value one if arc (i, j) is traversed by the path, zero otherwise; the capacity of p equals the minimum capacity of its arcs, i.e. $u_p = min_{(i,j)\in p} c_{ij}$. Finally, a path is said to be *elementary/loopless* if any node of the digraph is traversed at most once by the path, while it is called a *cycle* if it starts and ends at the same node. In the case of a single commodity all available paths are collected in $\mathcal{P}$, while in the multicommodity case in the $\mathcal{P}_h$ sets for $h \in \mathcal{H}$.

Static network flow notation:

$\mathcal{D} = (\mathcal{V}, \mathcal{A})$ a flow network,
$\mathcal{V}$ set of n nodes,
$\mathcal{A}$ set of m arcs,
$\delta^-(i) = \{j : (j, i) \in \mathcal{A}\}$ incoming arcs at node i,
$\delta^+(i) = \{j : (i, j) \in \mathcal{A}\}$ outgoing arcs from node i,
c_{ij} capacity of arc (i, j),

$\mathcal{H}$ set of commodities,
(o_h, d_h, σ_h) origin, destination and demand/population of commodity h,
$\mathcal{P}_h$ set of available paths for commodity h,
$u_p = \min_{(i,j)\in p} c_{ij}$ capacity of path p,
δ^p_{ij} binary indicator is 1 if arc (i, j) is traversed by path p, zero otherwise.

2.3. Formulations

We now introduce the *static flows* modeling tools and present their arc-flow and path-flow linear formulations. Decision variables of both representations are collected in the following box.

Static network flow variables:

x_{ij} amount of flow traversing arc (i, j),
x^h_{ij} amount of flow of commodity h traversing arc (i, j),

x_p amount of flow routed through path p,
x^h_p amount of flow of commodity h routed through path p.

Arc-flow formulation A *static flow* on a capacitated flow network $\mathcal{D}$ can be represented as a function that associates to each arc a non-negative amount of flow, formally:

$$\begin{aligned} x : \mathcal{A} &\longrightarrow \mathbb{R}^+ \\ (i, j) &\longrightarrow x_{ij} \end{aligned}$$

It is said that arc (i, j) is traversed by x_{ij} units of flow. A static flow from a node o to a node d of value σ, shortly an o-d flow, is *feasible* if it satisfies the following conditions:

$$\sum_{j \in \delta^+(i)} x_{ij} - \sum_{j \in \delta^-(i)} x_{ji} = \begin{cases} \sigma, & i = o \\ -\sigma, & i = d \\ 0, & i \notin \{o, d\} \end{cases} \qquad \forall i \in \mathcal{V}. \tag{2.1}$$

$$0 \leq x_{ij} \leq c_{ij} \qquad \forall (i, j) \in \mathcal{A}. \tag{2.2}$$

Flow conservation constraints (2.1) impose a continuous flow in the network: at any node i different from the source and the sink the same amount of flow is received and released, i.e. the net flow is zero at all intermediate nodes. Otherwise, an amount of σ units of flow is generated at node o and has to be entirely collected at node d. The second class of constraints force the flow to respect arc capacities of the network. This representation is known as the *arc-flow formulation* of a feasible static flow.

Path-flow formulation Feasible static flows can be alternatively modeled in terms of paths used for flow transshipment. More in detail, the following relevant result due to Ford and Fulkerson [39, 38] suggests how to construct an equivalent path-based representation from a feasible arc-flow assignment.

Theorem 2.3.1. ***Flow decomposition (Ford and Fulkerson** 1962) Any feasible o-d static flow x of value σ can be decomposed in at most m ($= |\mathcal{A}|$) cycles and elementary o-d paths carrying a non-zero amount of flow. Moreover, flow values on the decomposed paths sum up to σ.*

The decomposition procedure identifies an o-d *activated* path by x, i.e. a path p with a strictly positive amount of flow assigned to it, $x_p := \min_{(i,j) \in p} x_{ij} > 0$. Then, it reduces the feasible flow x on the arcs of path p by the x_p value and adds the so obtained pair (path, assigned flow) (p, x_p) to the list $\mathcal{L}$ of decomposed paths. The process is repeated until no more paths of such type can be found in the residual flow of x. Note that at most m of these iterations can be computed, and thus at most m paths are identified, as each step reduces to zero the flow on at least one arc. At this point, if all arcs have zero flow, the decomposition is concluded and the $\mathcal{L}$ list is returned. Otherwise, assume that (i, j) is an arc with a non-zero amount of flow associated to it, i.e. $x_{ij} > 0$. The satisfaction of the flow conservation Constraints (2.1) ensures the existence of at least one cycle passing through (i, j) with value x_{ij}. For each cycle of this type the correspondent amount of flow value is removed from x, till no more non-zero flow arcs are left. Note that the decomposed paths are enough to represent the original feasible flow as their associated flow values sum up exactly to σ. Indeed, none of the identified cycles contributes to increase flow between the given nodes.
The *path-flow* representation of a static o-d flow with value σ results thus in the following function:

$$\begin{aligned} x : \mathcal{P} &\longrightarrow \mathbb{R}^+ \\ p &\longrightarrow x_p \end{aligned}$$

associating a non-negative amount of flow to each path in $\mathcal{P}$; the path-flow assignment is *feasible* if:

$$\sum_{p\in\mathcal{P}} x_p = \sigma \tag{2.3}$$

$$\sum_{p\in\mathcal{P}} \delta_{ij}^p x_p \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}. \tag{2.4}$$

$$x_p \geq 0 \qquad \forall p \in \mathcal{P}. \tag{2.5}$$

Constraints (2.3) impose the transshipment of σ amount of flow, while Constraints (2.4) ensure that the activated paths respect all arc capacities (and implicitly path capacities). Flow conservation constraints at nodes are not required anymore, as once the flow is assigned to a certain path its amount is automatically preserved till the destination node is reached. Note that the number of paths between any pair of nodes exponentially increases in the dimension of the digraph, making thus impracticable a complete path enumeration and consequently the employment of a path-flow formulation when solving large- or real-sized scale problems. However in practice, the path-based representation is still preferred in many resolution algorithms as instance in those based on column generation techniques, including Branch and Price and matheuristics approaches, that efficiently overcome this issue by focusing on a sufficiently large subset of highly qualified paths. These concepts will be better detailed in Chapter 6 and 7. Finally, note that from a path-flow representation the arc-flow assignment can be reconstructed as follows: $x_{ij} = \sum_{p\in\mathcal{P}} \delta_{ij}^p x_p$. Being the two formulations equivalent, we will make use of them interchangeably in the discussion of theoretical results from the literature.

Multicommodity flows Suppose to have multiple commodities in a capacitated flow network, i.e. multiple origin-destination node pairs o_h-d_h each of them with a specified amount of demand σ_h. The arc-flow representation of such static *multicommodity flow* is a function that associates to each arc a non negative amount of flow separately for each commodity, formally:

$$\begin{aligned} x\ :(\mathcal{H},\mathcal{A}) &\longrightarrow \mathbb{R}^+ \\ (h,(i,j)) &\longrightarrow x_{ij}^h \end{aligned}$$

The multicommodity flow is said *feasible* if the following linear constraints are satisfied:

$$\sum_{j\in\delta^+(i)} x_{ij}^h - \sum_{j\in\delta^-(i)} x_{ji}^h = \begin{cases} \sigma_h, & i = o_h \\ -\sigma_h, & i = d_h \\ 0, & i \notin \{o_h, d_h\} \end{cases} \qquad \forall h \in \mathcal{H}, i \in \mathcal{V}. \tag{2.6}$$

$$0 \leq \sum_{h\in\mathcal{H}} x_{ij}^h \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}. \tag{2.7}$$

Constraints (2.6) ensure that flow conservation is satisfied by each single commodity, while

Constraints (2.7) impose the compliance of an arc capacity to the overall multicommodity flow passing through it.
The path-flow formulation of a feasible multicommodity flow can be similarly obtained by introducing an additional index to distinguish each commodity flow from the others, i.e. x_p^h for a given commodity $h \in \mathcal{H}$ and path $p \in \mathcal{P}_h$, and by forcing, this time explicitly, all commodity paths passing thorough an arc to respect its capacity, i.e. $\sum_{h\in\mathcal{H}} \sum_{p\in\mathcal{P}_h} \delta_{ij}^p x_p^h \leq c_{ij}$. As it will be detailed in the next paragraph through a practical example, the allocation of arc capacities to multiple commodities represents one of the main critical aspect of multicommodity flow problems, that frequently impacts their computational time complexity and inspires the design of tailored solutions approaches, such as those based on decomposition, see [66, 29, 5, 28].

2.4. Discussion on static network flow problems and their limitation

Static network flow problems represent the classical optimization problems in the Network Flow theory. They aim at identifying a feasible (single or multicommodity) static flow that, among all the possible candidates, optimizes a certain measure of quality. Depending on the definition of the measure, several static flow problems have been developed among years. As instance, the well-known Maximum o-d Flow Problem (MFP) seeks to maximize the amount of flow to be transshipped from node o to node d without exceeding network arc capacities [39, 38]. It can be solved by linear programming making use of the formulation (2.1)-(2.2) with $\max \sigma$ as objective function, or alternatively, the Ford and Fulkerson method based on the search for augmenting paths in the network can be employed among others. Note that in the single commodity case and when all arc capacities are integers, the same formulation can be employed to identify an optimal integer solution to the MFP, i.e. for $x_{ij} \in \mathbb{N}, \quad \forall (i,j) \in \mathcal{A}$, without the need of additional integrality constraints. This follows from the total unimodularity of the constraints' matrix and thus by the satisfaction of the hypothesis of the fundamental Integrality Theorem of Linear Programming [39, 38]. The integer version of the MFP has been also proved to be solvable in strongly polynomial time [78]. The same result doesn't hold in the multicommodity version of the MFP, where the total flow between each origin-destination pair must be maximized while respecting shared arcs' capacities. Indeed, the request of an optimal integer multicommodity flow shifts the computational complexity of the problem from strongly polynomial, see [55], to NP-hard, see [48]. This simple observation reveals a consistent increase in the level of difficulty of managing networks when limited resources and distinct flows are simultaneously involved.
Another relevant problem is the Minimum Cost o-d Flow Problem ($MinCFP$) that, associating a non-negative unit cost f_{ij} to each arc, asks to find a feasible o-d flow of a given value σ that minimizes the total cost, i.e. $min \sum_{(i,j)\in\mathcal{A}} f_{ij} x_{ij}$ in the arc-flow formulation. Many algorithms ideated for the MFP have been generalized to approach this problem, proved that the MFP is a special case of it in a slightly modified version of the network. Contributions related to different versions of the $MinCFP$ and to other algorithms developed for its resolution can be found in [1, 2]. A specific version of this problem will be used to deduce

the computational complexity of a novel network flow optimization problem in Section 5.3. Besides these problems, the network flow theory presents a large number of contributions requiring the satisfaction of additional specific aspects. Among them, a static flow might be requested to traverse at most a prefixed number of paths during its transshipment: when setting this limitation to one the so-called Unsplittable Flow Problem (UFP) is obtained, while increasing such maximum number of paths to a given integer value k leads to the class of k-Splittable Flow Problems ($kSFP$). The first contributions related to unsplittable flows have been developed by Kleinberg as a generalization of the edge-disjoint path problem: the single-source UFP presents multiple commodities sharing the same source node each of them forced to route its entire demand through a unique path to destination [69, 70]. The problem was proved to be NP-hard as many complex packing, scheduling and partitioning optimization problems can be reformulated as unsplittable flows. Kleinberg investigated several versions of the UFP. Among them, the Minimum Congestion UFP asks for finding the smallest value $\alpha \geq 1$ such that the routed unsplittable flows exceed arc capacities at most of a factor α. Equivalently, the Maximum Concurrent UFP aims at maximizing the common fraction of the commodity demands such that a feasible unsplittable multicommodity transshipment can be performed. Finally, in the minimum rounds version of the UFP, it is required to partition the set of commodities into the minimum number of subsets (rounds) such that a feasible unsplittable flow is secured for each of the generated subsets. Most of the developed contributions for the UFP and its variants are approximation schemes [77, 24], while only a restricted number of heuristics and exact algorithms based on the Branch and Price paradigm have been developed [115, 9, 33].
The Network Flow theory presents a large amount of static network flow problems dealing with the k-splittable limitation. Note that, due to the Flow decomposition Theorem presented in Section 2.3, having $k \geq m$ is equivalent to set flows free in their transshipment. Therefore, when dealing with k-splittable flows we implicitly assume $1 < k < m$. Further details of $kSFP$s will be extensively provided in Chapter 4, being them a central topic in this thesis.
Despite the great potential of static network flow problems in providing support to a broad variety of real-world contexts, they fail to represent, with a high level of detail, those practical situations where the dynamic of flows across a considered time horizon becomes relevant. Indeed, static flows are well fit to represent average flows behavior *to speed*, but their structural features prevent them from capturing the development of flows on networks under the lens of microscopic aspects, such as the time required by each unit of flow to proceed along the network arcs and the variation in the network utilization level over time. Moreover, flow movements are modeled as if occurring instantaneously or "on the average", with no room for realistic concepts as delays or breaks. As a result, static flows don't allow a correct and precise modeling of an increasing number of real-life operations whose efficiency nowadays heavily depends on time, such as processes of advanced logistics, telecommunication networks and synchronized transport systems. The scientific literature on Network Flows presents several contributions overcoming such limitations by efficiently incorporating the time parameter in network flow modeling: the so-called *dynamic flows* or *flows over time* tools, representing the core of our research activity, will be extensively detailed in the next chapter.

3. Dynamic flow problems

3.1. Introduction

In our research we focus on the specific class of *dynamic flow problems* often referred to as *flows over time*. As introduced in the previous chapter, see Section 2.4, dynamic flows have been conceived to overcome structural limitations of traditional static flows, that emerge when the evolution of transshipment operations must be represented and monitored over time, e.g. for application purposes. Indeed in these contexts, static flow-based modeling tools are not able to capture the amount of time spent by flows in proceeding along the network, nor the utilization and congestion levels of the digraph over time. Conversely, dynamic flow problems efficiently integrate the time parameter in a novel network structure, namely *dynamic digraph*, by means of two arc labels, a capacity and a delay/travel time, and by specifying a time horizon over which the process to be optimized is observed. This specific setting allows to identify at each point in time the exact position of flows and thus to closely manage and control their routing over the considered time period.
Scientific contributions in this field began to appear in the Sixties with the seminal works by Ford and Fulkerson [39, 38]. Since then, dynamic flow optimization problems have been attracting an increasing attention and nowadays, they are among the most suitable modeling tools to support decision making in those situations where *time* represents a relevant driver for planning strategies. Flows over time result to be particularly effective as instance in modeling traffic or goods distribution in transportation networks, data routing in telecommunication networks, production scheduling operations, and finally evacuation plans in case of emergency scenarios, see as instance [4, 98, 74, 67]. A recent application to shared-ride platforms built upon *Intelligent Transportation Systems* (ITS) technology can be found in [17].
A general classification of dynamic flow problems is based on the type of time horizon and network features considered. Indeed, the time period of interest can be either finite or infinite, continuous or discretized into a finite set of time intervals. The decision concerning the time parameter represents a relevant aspect in dynamic flow problems, in particular in the case of a finite discrete parameter, as the more discretized the time horizon, the better the representation of the flow transshipment over time. In fact, a non accurate discretization might lead to a lack of precision and to consequent inconsistent solutions. However, a too refined discretization of time might notably compromise the computational complexity of the problem to be solved, as it will be detailed later. Concerning network features, arc labels may vary during the considered time horizon or not. Moreover, the speed of flows may be influenced by the rate of flow concurrently entering the same arc, by the total load already

present in the arc at that time or be completely independent.
A further characterization of flows over time is related to flow storage, an additional modeling feature used to delay the transshipment process allowing flow to be held at nodes for a certain time period. In this situation, a node capacity is frequently introduced to bound the amount of flow that can be stored at each time instant. Note that even though flow storage is theoretically beneficial especially in congested networks, in some real-life applications it results completely inadequate. This is particularly true in emergency management where flow storage translates into forcing part of a population to wait at certain area, stopping a delicate evacuation procedure.
Dynamic flow problems might also forbid the usage of paths traversing nodes more than once over time, hence imposing flows to be routed only through elementary/loopless paths. This requirement is fundamental in many practical situations particularly when the minimization of the completion time for transshipment operations is required, such as in emergency transportation, where loops over time might translate into crossing unsafe areas several times with a consequent decrease in the stability and safety of the overall evacuation process.
Finally, flows over time have been investigated and largely explored both in the single and multicommodity versions. Contributions related to the multiple commodity setting confirm its relevance for modeling real-world contexts but also its impact on the computational complexity as it frequently leads to strongly NP-hard optimization problems.
Throughout this thesis we focus on flows over time with a finite discretized time parameter, time-independent arc attributes and a constant flow speed. Moreover, in the design of an original dynamic flow problem, we allow flow to be delayed at the respective source node but require, once the routing process has started, an elementary transshipment till destination with no breaks at intermediate nodes. These features are to increase the impact of the developed modeling tools in real-life applications where flow storage and loops over time don't represent a standard practice and might lead to high inefficiency in the implemented operations. See as instance the work by Melchiori and Sgalambro [88] applied to emergency management.
This chapter is organized as follows: Section 3.2 provides a thorough literature review of flows over time covering the main relevant optimization problems and the related algorithms; Section 3.3 collects the notation used throughout this chapter and Section 3.4 presents the general arc-flow and path-flow formulation for dynamic flow problems, a popular resolution strategy that, based on the concept of network time-expansion, can be applied in a broad number of cases, and finally a state-of-the-art ranking algorithm for quickest flows. In the last section we focus on Quickest Flow Problems (QFPs), a specific class of flows over time that is subject to the research activity presented in this thesis. We motivate our interest in such problems, providing evidence of the relevant impact of quickest flows on real-world contexts and identify potential room for novel contributions. We believe that these contents can provide a solid basis for addressing the core results of our research activities.

3.2. Literature overview

The first contribution integrating time into the mathematical modellization of networks flows traces back to the Sixties when Ford and Fulkerson developed the concept of *dynamic flows*,

more recently known as *flows over time* [39, 38]. The *dynamic digraph* structure was there introduced to track the movement of flows over a discretized time horizon: to each arc in the digraph a *transit time/delay* and a *capacity* arc attributes were associated, expressing the units of time required for the flow to travel through the arc from the tail to the head and the maximum inflow that can enter the arc at every time instant, respectively. Note that the given definition of capacity doesn't refer to the total amount of flow on the arc at a certain time instant. This alternative attribute, which will not be considered in our thesis, is referred to as aggregated capacity [89, 32].
Ford and Fulkerson have shown that a general dynamic flow problem can be always modeled, and thus tackled, as an equivalent static one on a specific Time-Expanded Network (TEN), obtained by replicating for each time step the original underlying dynamic digraph and by connecting consecutive time layers through holdover arcs to model flow storage. In this novel network, classical efficient algorithms developed for static flow problems can be employed. However, the price for such simplification lies in the size of the so-generated TEN that might convert any polynomial algorithm in a pseudo-polynomial one due to the possible non-linear dependence of the time horizon in the input size.
Relevant static flow problems have been extended to the dynamic case by allowing the transshipment to span a fixed time horizon. This is the case as instance of the Maximum Dynamic Flow Problem, the Minimum Cost Dynamic Flow Problem and some of their variants [39, 38, 57, 71, 51]. In addition to the above mentioned cases, entirely novel dynamic flow problems found their specific root and motivations in the dynamic environment: the Quickest Flow Problem, the Quickest Path Problem, the Quickest Transshipment Problem and the Earliest Arrival Problem among others, see [19, 26, 61, 40]. We refer to Aronson [4], Kotnyek [79] and Skutella [109] for a complete survey and works on flows over time.
Multicommodity flows over time A very broad study on the multicommodity version of dynamic flow problems has been conducted by Hall et al. [57]. They proved the weak NP-hardness of general flows over time even in the case of two commodities and the strong NP-hardness when only elementary paths are allowed and intermediate flow storage is forbidden. Note that in the first case, a multicommodity solution can be still computed in pseudo-polynomial time via the time-expansion procedure of Ford and Fulkerson of the original dynamic digraph. Polynomial-time strategies have been designed for multicommodity flows over time when specific assumption on the arc delays or on the network topology were made. In particular, Hall et al. explored the setting where all available paths for any commodity present the same length, namely networks with uniform path lengths, and where each node presents at most one outgoing arc, resulting this in a single path for each origin-destination commodity.
The Maximum Dynamic Flow Problem An instance of the Maximum Dynamic o-d Flow Problem asks for maximizing the amount of flow to be transshipped from o to d within a given time horizon. A relevant result from Ford and Fulkerson prove that there always exists a solution to this problem that can be represented as a *temporally repeated flow*, i.e. a feasible static flow re-occurring over time as long as there is enough time for the flow to reach the destination. Based on this fact, they designed an efficient algorithm that is strongly polynomial if all arc delays are non-negative [39]: at a fist stage it computes a minimum cost flow from o to d in a slightly modified digraph where costs equal arc delays; an optimal solution to this problem is then decomposed into paths (see the Decomposition flow Theorem in Section

2.3) and a temporally-repeated flow is sent along them. The multicommodity version of this problem becomes weakly NP-hard as proved by [57]; Fully Polynomial-Time Approximation Schemes ($FPTAS$s) designed for its resolution can be found in [37, 54, 52].

The Minimum Cost Dynamic Flow Problem The dynamic version of the Minimum Cost o-d Flow Problem asks for finding a feasible o-d flow at minimum cost whose transshipment can be accomplished within a given time horizon. This problem has been studied by Klinz and Woeginger [71], who proved its NP-hardness even for series parallel graphs. The same authors provided a greedy algorithm together with a characterization of the graph structures on which this strategy results to have polynomial-time. Moreover, they proved that the temporally-repeated scheme cannot be applied in this context and that in general a computation of a temporally-repeated flow of minimum cost is strongly NP-hard. Fleischer and Skutella [36] developed the first $FPTAS$ for the problem in the case of no flow storage and a capacity scaling $FPTAS$ for the specific case with costs proportional to travel times. In particular the first method works on a so-called condensed TEN where a roughly discretization of the time horizon is performed. For the multicommodity version of the problem with no flow storage and loopless paths a Column Generation algorithm has been designed by Grande et al. [51]. In the work of Hall et al. a polynomial-time algorithm was provided for uniform path lengths networks [57]. The resolution approach works on a polynomial-sized TEN with $O(n)$ time-layers and can be applied for both the single and multicommodity case and with or without flow storage.

The Quickest Flow Problems In the Quickest Flow Problem (QFP) one aims at routing a given amount of flow between a pair of nodes in a capacitated network as quickly as possible, i.e. minimizing the *makespan* of the process. A straightforward approach to solve the QFP performs a binary search on the time horizon computing a maximum dynamic flow at each iteration and stops whenever the obtained optimal solution covers at least the amount of demand requested by the problem. Integrating the Megiddo's parametric search into this scheme, a strongly polynomial-time algorithm is obtained [19]. This approach doesn't require flow storage nor loopless paths. A recent formulation for the QFP can be found in Lin and Jaillet [80] together with a cost-scaling algorithm that presents the same time bound as one of the fastest algorithm for solving the static Minimum Cost Flow Problem.

The generalization of the QFP to multiple sinks and sources is called the Quickest Transshipment Problem. The first strongly polynomial-time algorithm for this problem was developed by Hoppe and Tardos [61] and it is based on several computations of a parametric submodular function minimization. Their algorithm identifies optimal solutions that make use of only simple paths without the need of flow storage. Recently, Schlöter and Skutella provided a faster algorithm that requires only one call of the previously mentioned subroutine and provides structurally easier solutions being them convex combinations of simple lexicography-maximal flows over time, see [105].

Several approximation algorithms for the multicommodity extension of the QFP problem, namely the Quickest Multicommodity Flow Problem ($QMCFP$), have been designed by Fleischer and Skutella [37]. First, they extended the Ford and Fulkerson algorithm proving that any feasible multicommodity flow can be computed as a temporally-repeated multicommodity flow within a doubled time horizon. From this, an approximation scheme with performance guaranteed 2 follows by embedding the temporally-repeated flow computation within a parametric search for the minimum time horizon. As temporally-repeated flows do not require flow storage, they showed that forbidding flow storage increases the makespan of a quickest multicommodity flow of at most two times. Moreover, flow storage is proved to be unnecessary in the case of single commodity. The same authors presented a $FPTAS$ with performance guaranteed $(1+\epsilon)$ with $\epsilon > 0$ for the $QMCFP$ that works on a condensed TEN of polynomial size in the input data with only n/ϵ^2 time layers. A simple polynomial-time greedy algorithm has been designed by Hall et al. [57] for the $QMCFP$ in the case of networks with at most one outgoing arc for each node and with flow storage allowed: the strategy schedules commodity flows giving priority on shared arcs to those that are farthest from reaching their destination.

The Quickest Path Problem A rich variety of contributions focused on the specific case of quickest flows restricted to a single path, the so-called Quickest Path Problem (QPP). The problem was firstly posed by Moore in 1976 [91] in the context of convoy-type traffic flow but formalized only later by Chen and Chin in 1990 [26]. Martins and Santos [86] modeled the QPP as a minsum-maxmin bicriteria path problem; indeed, any optimal solution to the QPP is a non-dominated solution to the problem of simultaneously minimizing the total path delay and maximizing its capacity. Different polynomial-time strategies have been developed for its resolution, some of them reducing it to the computation of Shortest Path Problems (SPP) in enlarged networks [26, 20], others performing sequences of SPP in subnetworks of the original one [86, 91, 103] or being label-setting algorithms [94, 107]. All of these techniques present the same time computational complexity of $O(r(m + nlogn))$, where r is the number of distinct arc capacity values, m the number of arcs and n of nodes. However, the recent algorithm of Sedeño-Node and Gonzáles-Barrera [107] is proved to outperform all of the others. We refer to the following surveys and works for models and algorithms of the QPP and some of its variants such as the reliable QPP and the weak and strongly stable QPP [26, 96, 107, 87, 49, 104]. This class of problems perfectly suites to model real-life situations where a quickest transshipment together with a very tight control on the used paths is key for an efficient and well-organized process. See the work of Melchiori and Sgalambro [87] for the application to emergency management where a single path is highly preferred to contain and reduce turbolences and congestion during the operational implementation of evacuation plans. In this contribution the first arc-flow formulation for the multicommodity extension of quickest paths, namely the Multicommodity Quickest Path Problem ($MCQPP$), in the TEN was provided with forbidden flow storage at intermediate nodes and additional time-scheduling constraints. Note that requiring a single path for each commodity represents the transposition in the dynamic environment of the concept of unsplittable flows, developed and largely studied for classical static network flow problems [69, 77, 76].

The related problems of ranking K-quickest paths and K-quickest loopless paths in a non decreasing order of the transmission time, i.e. the time required for the transshipment of the entire commodity demand, have been addressed by Chen [25], Rosen, Sun and Xue [103] and Pascoal et al. [96, 95, 97] and applied to the routing of data packets in Internet networks in Clímaco et al. [27]. These algorithms either resemble strategies developed for ranking K-shortest paths, or they generalize resolution methods for QPP. The best behavior is provided by the Pascoal et al. algorithm [96, 97], a lazy version of the Chen's approach based on ranking simple shortest paths in a sequence of subnetworks and on alternating paths identification and paths output.

The Earliest Arrival Flow Problem Alternatively to the makespan minimization, one can request that at each point in time the number of units of flow arrived at destination is maximized. This is modeled by the so-called Earliest Arrival Problem (EAP), that results particularly suited for representing evacuation processes [67]. However, the existence of earliest flows has been proved only for the setting with one sink and a unrestricted number of sources. In this case, pseudo-polynomial algorithms and $FPTAS$ schemes have been designed [90, 60, 37]. For the setting with multiple sinks, it is not completely clear whether earliest arrival flows exist or not, except for some specific network structures where an answer to this question has been provided [12, 11, 106]. Approximation algorithms for arbitrary networks have been formulated by Gross et al. [53] and [52].

We conclude this section by citing the recent work of Hajjem et al. [56] that consists of a general evolutionary framework to solve flows over time and presents a specific case study applied to an evacuation process from a building.

3.3. Notation

We provide here the complete notation required in the dynamic setting of flows on networks and summarized it in a box.
A *dynamic flow network* is defined as a structure $\mathcal{D} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$, with $\mathcal{V}$ being a set of n nodes, $\mathcal{A}$ a set of m directed arcs and $\mathcal{T} = \{0, 1, \dots, T\}$ a finite set of time steps into which a certain time horizon is discretized. Two time-independent labels are associated to each arc $(i, j) \in \mathcal{A}$: a strictly positive *capacity* c_{ij} representing the maximum amount of flow that can concurrently enter the arc from its tail at each time instant, and a non-negative *travel time/delay* λ_{ij}, specifying the number of time steps needed to traverse the arc from its tail to the head. Given a certain path p, its capacity u_p is defined, similarly to the static environment, as the minimum of its arcs' inflow capacities and its length l_p as the sum of its arcs' delays, $\sum_{(i,j)\in\mathcal{A}} \delta^p_{ij}\lambda_{ij}$, where the binary indicator δ^p_{ij} is set to one for all arcs (i, j) part of p. The number of units of time needed to reach a node $i \in p$ when using path p from its origin node is indicated as t^p_i. Finally, the *transmission time* of an o-d path, related to a certain commodity (o, d, σ), expresses the amount of time required to complete the transshipment of the entire commodity demand through it, formally $T(p) = l_p + \left\lceil \frac{\sigma}{u_p} \right\rceil - 1$. The path is called *feasible* for the commodity if at least a positive amount of demand can arrive at destination within the considered time horizon traveling through it, i.e. $u_p \geq 1$ and $l_p \leq T$. For seek of simplicity, in this subsection we suppose that the commodity origin o and destination d have no incoming and no outgoing arcs, respectively. This specific setting can be easily created by adding a dummy source node connected just to node o and a dummy sink node reached only by node d. Moreover, we suppose that no arcs of the type (i, i), namely *loops*, are included in $\mathcal{A}$. Again, this situation can be resolved by considering the head of such arc as a new node j in the set $\mathcal{V}$ and by generating a new outgoing arc from i to j with same attributes and from j to i with zero travel time and the same capacity.
Figure 3.1 depicts the utilization over time of an arc with delay equal to three time steps and a capacity set to one unit in the first case on the top and increased to two units on the bottom case. In the first case the arc is traversed by one unit of blue flow entering its tail at time zero and exiting from its head at time three. In the second case at time zero two

units of blue flow enter the arc completely fulfilling its capacity and proceed at a constant rate over time toward its head. A single unit of green flow is routed through the same arc at the first available time instant and will clear the arc at time four.

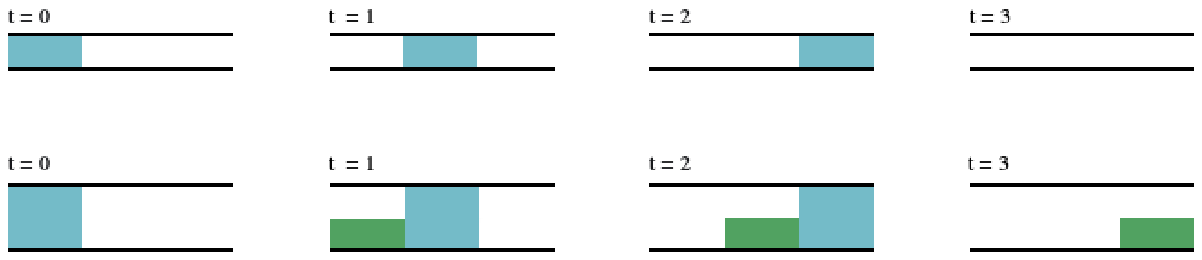


Fig. 3.1: Flow traversing arcs in a dynamic network

Dynamic network flow notation:

$\mathcal{D} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$ a dynamic flow network,
$\mathcal{V}$ set of n nodes,
$\mathcal{A}$ set of m arcs,
$\mathcal{T}$ set of T time steps,
$\delta^-(i) = \{j : (j,i) \in \mathcal{A}\}$ incoming arcs at node i,
$\delta^+(i) = \{j : (i,j) \in \mathcal{A}\}$ outgoing arcs from node i,
c_{ij} inflow capacity of arc (i,j) at each time instant,
λ_{ij} travel time/delay of arc (i,j) at each time instant,

$\mathcal{H}$ set of commodities,
(o_h, d_h, σ_h) origin, destination and demand/population of commodity h,
$\mathcal{P}_h$ set of available paths for commodity h,
$u_p = \min_{(i,j)\in p} c_{ij}$ capacity of path p,
δ^p_{ij} binary indicator is 1 if arc (i,j) is traversed by path p,
$l_p = \sum_{(i,j)\in\mathcal{A}} \delta^p_{ij}\lambda_{ij}$ length of path p,
t^p_i time required to reach node i through path p,
$T(p) = l_p + \frac{\sigma}{u_p}$ transmission time of path p.

3.4. Formulations and significant algorithms

In this section we formally introduce dynamic flow problems and their traditional arc-flow and path-flow representations. The decision variables follow in the box. Two state-of-the-art resolution strategies conclude the section.

Dynamic network flow variables:

x_{ijt} amount of flow traversing arc (i, j) at time t,
x_{ijt}^h amount of flow of commodity h traversing arc (i, j) at time t,

x_{pt} amount of flow leaving the source through path p at time t,
x_{pt}^h amount of flow of commodity h leaving the source through path p at time t.

Arc-flow formulation A *dynamic flow* in a dynamic digraph $\mathcal{D}$ can be defined by the following function:

$$\begin{aligned} x : (\mathcal{A}, \mathcal{T}) &\longrightarrow \mathbb{R}^+ \\ ((i, j), t) &\longrightarrow x_{ijt} \end{aligned}$$

that associates to each arc (i, j) and time t an amount of flow entering its tail at the given time instant and exiting from the arc's head at time $t + \lambda_{ij}$. A dynamic o-d flow of value σ is *feasible* if the following constraints are obeyed:

$$\sum_{j \in \delta^+(o)} \sum_{t \in \mathcal{T}} x_{ojt} = \sigma \tag{3.1}$$

$$\sum_{i \in \delta^-(d)} \sum_{t \in \mathcal{T}} x_{idt} = \sigma \tag{3.2}$$

$$\sum_{j \in \delta^+(i)} x_{ijt} - \sum_{j \in \delta^-(i)} x_{ji(t-\lambda_{ji})} = 0 \qquad \forall i \in \mathcal{V} \neq \{o, d\}, t \in \mathcal{T}. \tag{3.3}$$

$$0 \leq x_{ijt} \leq c_{ij} \qquad \forall (i, j) \in \mathcal{A}, t \in \mathcal{T}. \tag{3.4}$$

where for seek of simplicity we suppose that $x_{ijt} = 0$ whenever $t < 0$ or $t + \lambda_{ij} > T$, modeling the fact that movements of flows are traced only within the considered time horizon. The first constraint imposes the origin node to release σ units of flow over the time horizon; the second constraint ensures that the entire amount of flow is received at the destination node, while Constraints (3.3) represent the transposition to the dynamic environment of the classical flow conservation constraints: they require the net flow at each intermediate node to be zero at each time step. The last family of constraints deal with the satisfaction of arc capacities at each time instant.

Path-flow formulation As in the static environment, an equivalent path-flow formulation can be employed to model dynamic flows. It requires as an input the complete set $\mathcal{P}$ of all the feasible paths that can be used for flow transshipment within the time horizon. The path-based flow function associates to each path p and each time instant t a non-negative value x_{pt} expressing the amount of flow leaving the origin of the path at that time, formally:

$$\begin{aligned} x : (\mathcal{P}, \mathcal{T}) &\longrightarrow \mathbb{R}^+ \\ (p, t) &\longrightarrow x_{pt} \end{aligned}$$

After leaving the origin node, the flow proceeds at a constant rate along the path thus arriving at destination at time $t + l_p$. Again, for any t s.t. $t < 0$ or $t + l_p > T$ the related x_{pt} value is set to zero. A dynamic o-d flow of value σ is *feasible* if the following constraints are obeyed:

$$\sum_{p\in\mathcal{P}}\sum_{t\in\mathcal{T}} x_{pt} = \sigma \tag{3.5}$$

$$\sum_{p\in\mathcal{P}} \delta^p_{ij} x_{p(t-t^p_i)} \leq c_{ij} \qquad \forall (i,j)\in\mathcal{A}, t\in\mathcal{T}. \tag{3.6}$$

$$x_{pt} \geq 0 \qquad \forall p\in\mathcal{P}, t\in\mathcal{T}. \tag{3.7}$$

The first constraint models the transshipment of σ units of flow along all the available paths and over time, while Constraints (3.6) ensure that arc capacities (and so path capacities) are respected at each time point in time. No conservation constraints are required as flow is preserved along each path at each time instant. Note that with the above definition flow storage at intermediate nodes is avoided as the set of arcs contains no loops. Delayed departures are modeled by simply scheduling flows along the time horizon, i.e. by activating variables x_{pt} for $t > 0$. Finally, note that the arc-flow formulation can be reconstructed as follows $x_{ijt} = \sum_{p\in\mathcal{P}} \delta^p_{ij} x_{p(t-t^p_i)}$.

Multicommodity dynamic flows The multiple commodities setting presents distinct dynamic flows that have to be simultaneously transshipped in the same network during the same time horizon. Recall that each commodity is expressed as a tuple origin destination and demand (o_h, d_h, σ_h). A *multicommodity dynamic flow* can be modeled by simply introducing an index $h \in \mathcal{H}$ related to the commodity to which the flow refers in the arc- and path-flow representations, x^h_{ijt} and x^h_{pt}, respectively. A multicommodity dynamic flow is said *feasible* if in the arc-flow formulation satisfies the following constraints:

$$\sum_{j\in\delta^+(o_h)}\sum_{t\in\mathcal{T}} x^h_{o_hjt} = \sigma_h \qquad \forall h\in\mathcal{H}. \tag{3.8}$$

$$\sum_{i\in\delta^-(d_h)}\sum_{t\in\mathcal{T}} x^h_{id_ht} = \sigma_h \qquad \forall h\in\mathcal{H}. \tag{3.9}$$

$$\sum_{j\in\delta^+(i)} x^h_{ijt} - \sum_{j\in\delta^-(i)} x^h_{ji(t-\lambda_{ji})} = 0 \qquad \forall h\in\mathcal{H}, i\in\mathcal{V}\neq\{o_h,d_h\}, t\in\mathcal{T}. \tag{3.10}$$

$$\sum_{h\in\mathcal{H}} x^h_{ijt} \leq c_{ij} \qquad \forall (i,j)\in\mathcal{A}, t\in\mathcal{T}. \tag{3.11}$$

$$x^h_{ijt} \geq 0 \qquad \forall h\in\mathcal{H}, (i,j)\in\mathcal{A}, t\in\mathcal{T}. \tag{3.12}$$

Note that the formulation requires each single commodity to route the given amount of flow within the time horizon while respecting flow conservation constraints, Constraints (3.8)-(3.10); moreover, the overall multicommodity flows must respect shared arcs' capacities, Constraints (3.11).

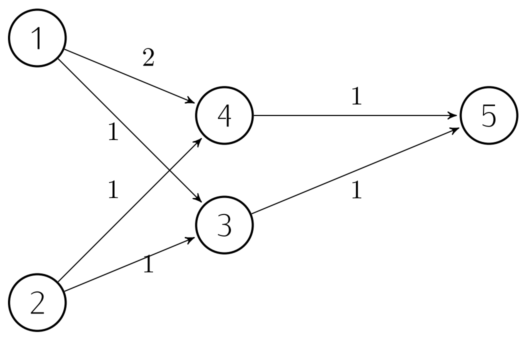
3.4.1. The time-expansion procedure

In this paragraph we describe the first resolution approach developed for general dynamic flow problems that is based on the so-called *time-expansion procedure*. The general idea underlying the strategy is to transform a dynamic flow problem into an equivalent static one where all the toolbox of classical network flow algorithms can be exploited for its resolution. For seek of simplicity we suppose integral arc transit times. The time-expansion procedure generates from the original dynamic network $\mathcal{D} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$ an equivalent static network, namely the Time-Expanded Network (TEN), by copying the underlying set of nodes for each discrete time step in $\mathcal{T}$ and by linking those copies through original arcs accordingly to their travel times. Moreover, consecutive time layers are connected through holdover arcs to model flow storage. Note that through this procedure the time dimension is completely removed from the novel network structure as the time horizon and travel times are now explicitly accounted by the time-replica of nodes and arcs. Formally, the TEN is a directed digraph $\mathcal{D}_{\mathcal{T}} = (\mathcal{V}_{\mathcal{T}}, \mathcal{A}_{\mathcal{T}})$ where:

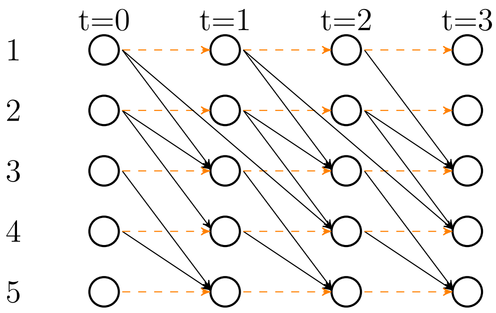
$$\begin{aligned}
&\mathcal{V}_{\mathcal{T}} := \{i_t \mid i \in \mathcal{V}, t \in \mathcal{T}\}\\
&\mathcal{A}_M := \{(i_t, j_{t'}) \mid (i,j) \in \mathcal{A},\ t' = t + \lambda_{ij} \leq T\}\\
&\mathcal{A}_H := \{(i_t, i_{t+1}) \mid i \in \mathcal{V}, t = 0, \ldots, T-1\}\\
&\mathcal{A}_{\mathcal{T}} = \mathcal{A}_M \cup \mathcal{A}_H
\end{aligned}$$

The set $\mathcal{V}_{\mathcal{T}}$ contains one copy of the original node set for each discrete time step and $\mathcal{A}_M$ contains the time replica of the original arcs with the same capacity attributes. For the sake of clarity in the notation, we will sometimes indicate arc $(i_t, j_{t'})$ simply as $(i,j)_t$. The additional set $\mathcal{A}_H$ contains the holdover arcs, in general with a non-bounding infinite capacity, that allow the flow to be held at a node over a one time period. Note that in our setting $\mathcal{A}_H$ is restricted to holdover arcs only at the source node as flow storage at intermediate nodes is forbidden. Any feasible static flow in the TEN can be thus interpreted as an equivalent feasible dynamic flow in the original digraph. Specifically, any flow on arc $(i,j)_t$ corresponds to a flow of the same amount passing through arc (i,j) at time t in the dynamic digraph; conversely, given a dynamic flow in $\mathcal{D}$ with $\mathcal{T}$ as time horizon, the equivalent flow in $\mathcal{D}_{\mathcal{T}}$ can be constructed by sending on arc $(i,j)_t$ the same amount of flow assigned to arc $(i,j) \in \mathcal{A}$ at time t and by repeating this for any arc an any time instant.

Thanks to its simplicity and effectiveness in capturing the process evolution over time, the time-expansion procedure is a commonly employed approach, as demonstrated by the following recent contributions in the context of train timetabling [34], water supply management [100], and fleet management [18]. Moreover, note that this procedure can be applied to a broad variety of flows over time regardless of their objective function and that can present as instance time-dependent or flow-dependent arc attributes and that can account for flow storage at all nodes. However, a major (and in many cases fatal) drawback of the time-expansion procedure lies in the dimension of the generated TEN that might prevent this procedure to be employed for very large instances of dynamic flow problems. Indeed, a possible non-polynomial dependence of the time horizon in the input size would convert efficient static network flows algorithms into pseudo-polynomial ones in the TEN. As introduced in the literature overview of Section 3.2, recent contributions have focused on this issue proposing



(a) Dynamic digraph.



(b) Time-Expanded Network.

Fig. 3.2: Time-expansion procedure over $T = 3$.

to perform a rougher expansion of the time horizon that replaces the unit time step with a carefully chosen larger step and that simultaneously ensures a polynomial-time algorithm and a good level of quality of the solutions that can be achieved. The so-constructed digraphs are called *condensed* or *geometrically condensed* TEN depending on whether a unique factor is used to scale travel times or the size of time intervals increases geometrically, see [37, 53, 52]. The developed procedure results in theoretically efficient $FPTAS$s for many dynamic flow problems with only elementary paths allowed such as the Minimum Cost Flow over Time, Quickest Flow Problem, Earliest Arrival Flow Problem, and multicommodity flows over time in general. Finally, a *sequence condensation* procedure and a sequence-based formulation have been recently developed to account for the use also of non-elementary paths, see [54, 52]. Note that the time-expansion procedure can be extended to rational travel times by simply scaling them by their least common denominator and by accordingly adapting arc capacities.

Figure 3.2 depicts the general time-expansion procedure applied to the dynamic digraph on the left over a time horizon of three time intervals. As arc capacities are preserved through this procedure, we have omitted them in the descriptive labels and only delays are indicated. The original five nodes are placed in the first vertical column in the figure on the right and are reproduced along the time horizontal axis making a separate copy for every discrete time instant; arcs are drawn according to their delay attribute at each time instant whenever their head falls within the considered time horizon. Holdover arcs (orange dashed in the figure) are added to connect each node in the TEN with its replica at the next time layer.

In our thesis we will make use of the time-expansion strategy in the pricing routine of a tailored Branch and Price approach designed for the resolution of a novel dynamic flow problem, see Section 7.2.3.

3.4.2. Ranking K-quickest loopless paths

We discuss here the so-called lazy version of the Chen's algorithm for solving the K-quickest loopless paths problem as presented in the works of Pascoal et al. [96, 97]. The problem requires the identification of best K elementary quickest paths provided in an increasing order of their transmission time. The idea underlying the method is to reduce the problem to several computations of a state-of-the-art K-shortest loopless path algorithm in the static

environment and to alternate path identification steps to path selection steps till a total of K paths are identified. A pseudocode of the algorithm is provided in Algorithm 1. It requires as an input the dynamic digraph and the given commodity to be transshipped. In the initialization phase, Step 1, it lists all the arc capacities in an increasing order excluding potential repetitions, hence obtaining r different values with $r \leq m$. Then, for each of these values it constructs a subgraph by considering only arcs whose capacities exceed or are equal to the given value set as a lower bound and in the so-constructed digraph it identifies the shortest loopless path w.r.t. arc travel times, Step $1b$. The generated shortest paths are stored in the array L. In Step 2, the algorithm iteratively selects the best shortest path in the array w.r.t. the transmission time and replaces it by the next shortest loopless path found in the related subgraph. At this stage any algorithm for solving the K shortest loopless path problem can be employed, such as the Yen's algorithm [116] or Katoh et al. [68]. The procedure stops whenever K different paths have been selected or no more shortest paths are available.
This strategy will be used in Subsection 6.2.1 and Subsection 7.2.2 when presenting the algorithms developed for the ad-hoc resolution of a novel dynamic flow problem. In particular, it will be employed in their initial stages to generate a set of high-quality paths whose related columns will be considered eligible for inclusion in restricted versions of the path-flow formulation of the problem.

Algorithm 1 Lazy version of Chen's algorithm

```
    Input: $\mathcal{D} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$, $(o, d, \sigma)$;
    Output: the $K$ quickest paths in $\mathcal{D}$;

1:  Initialization
    a. Order arc capacities: $c_1, \dots, c_r$;
    b. $\mathcal{A}' \leftarrow \mathcal{A}$;
       for $l = 1, \dots, r$ :
          $\mathcal{A}' \leftarrow \{(i, j) \in \mathcal{A}' \mid c_{ij} \geq c_l\}$ ;
          $L[l] \leftarrow$ the $o$-$d$ shortest loopless path in $\mathcal{D}' = (\mathcal{V}, \mathcal{A}')$, $\emptyset$ otherwise;

2:  Find $K$-Quickest Paths
    $k \leftarrow 0$;
    while $k < K$ and $\exists\, l$ s.t. $L[l] \neq \emptyset$ do:
     a. $p \leftarrow L[l]$ s.t. $L[l] = \text{argmin}_{i=1,\dots,r}\, T(L[i])$;
     b. $\mathcal{A}' \leftarrow \{(i, j) \in \mathcal{A} \mid c_{ij} \geq c_l\}$;
          $L[l] \leftarrow$ the next $o$-$d$ shortest loopless path in $\mathcal{D}' = (\mathcal{V}, \mathcal{A}')$, $\emptyset$ otherwise;
     c. if $k = 0$ or $p \notin \{\bar{p}_1, \dots, \bar{p}_k\}$ then $k \leftarrow k + 1,\ \bar{p}_k \leftarrow p$;

3:  Return $\{\bar{p}_1, \dots, \bar{p}_K\}$;
```

3.5. Relevance of Quickest Flows

An increasing amount of real-world contexts are concerned with the minimization of the time required to complete a set of operations modeled as flows to be transshipped on networks, the latter representing a critical aspect for an overall optimized process. For example this can be observed in evacuation management when one aims at devising fast evacuation plans to efficiently respond to emergency scenario [61, 58, 87], or in the transportation sector [75], as instance when planning at a tactical level vehicle movements in storage areas subject to strict timing requirements imposed by an urgent need for the products to be delivered. Moreover, in telecommunication networks [27], where scheduling of jobs or routing of data packets are often required to be performed within the shortest possible time to contain energy consumptions and management costs.
The scientific literature on flows over time provides support in this context through the Quickest Flow Problem class that allows for the identification of the quickest solution among the feasible ones. This class has been enhanced over years with several novel contributions ranging from different dynamic flow problems to ad-hoc resolution algorithms, confirming a solid potential of such optimization tools in representing real-life situations as well as a constant interest in improving and refining their modeling capabilities. Hence, the basic version of the Quickest Flow Problem has been extended to multiple commodities and to multiple sink and sources to allow for the representation of realistic situations where limited network resources are shared by different flows while pursuing an overall quickest transshipment [61, 35]. At the same time, more specific requests have been integrated, first and foremost among them is the limitation to only one path when routing quickest flows over time that found its way through the design of the Quickest Path Problem [96, 87]. Note that the restriction to loopless/elementary paths and the prohibition of flow storage at intermediate nodes are very common in quickest flow modeling, being them in line with many real-world applications, such as emergency transportation, as previously discussed in the introduction of this chapter.
The general concept of intensifying control on routing operations by bounding the number of active paths is a noteworthy topic in the Network Flow theory, and contributions related to the QPP represent an initial investigation in such direction within the framework of flows over time. As previously mentioned in Section 2.4, path restrictions have already a valid representation in the traditional static framework of network flows, with a large variety of research works known under the name of Unsplittable and k-Splittable Flow Problems. On the contrary, the dynamic setting of flows on networks still presents a large room for improvements, as the current state-of-the-art contributions is able to model only widely different situations, where either an unsupervised flow routing can be performed, as for the standard QFPs, or a very firm limitation to only one path per each distinct flow is imposed, as for the QPPs. No tools are available, except for a very preliminary work [85], to efficiently represent intermediate, and actually more realistic, situations where a limited, but greater than one, number of paths are allowed to be employed for flow transshipment operations over a certain time horizon of interest.
With this thesis we aim at filling the current gap between free-flow quickest flows and quickest paths tools, by developing original contributions that explicitly account for k-splittable

path limitations in the Quickest Flow class. Such integration could considerably increase the adherence of quickest flow-based modeling tools to real-life applications, securing a valid support in a number of fields where feasible fast transshipments can be achieved only by rationalizing the employment of limited resources.
In the next chapter we provide a rich and exhaustive dissertation on $kSFP$s highlighting in the last section the lack of works in the dynamic framework, while from Chapter 5 we start presenting our research activity that concerns a novel flows over time problem and two tailored algorithms for its resolution.

4. Bounding the number of paths in the static environment

4.1. Introduction

Classical network flow problems, both in the static and in the dynamic environment, traditionally allow commodity demands to be split into an arbitrary and unrestricted number of subgroups while being transshipped, and no control is put on the total number of support paths activated by the routing process. This degree of freedom, also referred to as the *splittable or free-flow* feature, secures an efficient usage of the network structure, especially in the presence of finite arc label capacities and concurrent distinct demands to be satisfied. However, a splittable flow setting becomes highly inadequate to represent real-life situations where a tight control on the transshipment operations is key for an efficient planning and an unbounded flow partition is undesirable or even prohibitive. This can be observed as instance in the telecommunication field, where data packets are sent across a capacitated network in the presence of the Multiple Protocol Label Switching Network ($MPLS$), a technology that allows data to be routed on multiple paths between node clients [102]. In this context, communication channels consisting of a prefixed maximum number of paths are preferred among others to support efficient protocol performances and contain costs motivated by routes supervision, path maintenance and information reconstruction at destinations [63, 13]. In transportation and distribution logistic, the number of available vehicles to deliver passengers and freights is in general a limited resource. Therefore, any efficient network design approach for describing transshipment processes must account for route restrictions [73]. In bioinformatics, a recent example is represented by the protein isoforms identification process to explain specific spectra. Indeed, a more reliable representation can be achieved when restricting the search to only a small number of protein isoforms due to the presence of noise and inaccuracy in the input data [117]. The above mentioned real-world situations, despite not being exhaustive, provide large evidence of a common practical-driven need for a dedicated feature in network flow modeling addressing restrictions related to the number of paths used for flow transshipment.
In this chapter we present a state-of-the-art for network flow problems that copes with path number limitations, focusing on those contributions employing the so-called *k-splittable flow* tools to explicitly limit the maximum number of paths that can be activated to an integer value k.
The first optimization problem dealing with k-splittable flows was conceived as a generalization of the well-known Unsplittable Flow Problem (UFP); since then, a relevant amount

of models and algorithms have been designed in this context, giving rise to the class of k-Splittable Flow Problems ($kSFP$s). Scientific contributions have been mainly developed for the static network setting, whereas the integration of such modeling tools within flows over time is nowadays still an uncharted field, presenting only a preliminary work in the literature.
In the first part of this chapter we present a thorough review of the scientific literature concerning k-splittable flows. Then, in Section 4.3 we formally introduce the concept of k-splittable flow and provide general representations that can be employed for different k-Splittable Flow Problems in the static environment. Moreover, a significant approximation scheme frequently employed in the field is provided. We will make use of this technique in Chapter 6 while performing computational experiments on a resolution algorithm tailored to solve a novel dynamic k-splittable flow problem. Finally, in Section 4.4 we will discuss the relevance of k-splittable flows and highlight the need for enriching the related scientific literature, in particular w.r.t. the dynamic environment.

4.2. Literature overview

The first contributions coping with k-splittable flows have been conceived as a generalization of the Unsplittable Flow Problem (UFP) and its variants [69, 70]. Indeed, unsplittable flows can be derived from k-splittable ones when forcing the k parameter to one. A relevant research work in this specific setting is due to Barnhart et al. [9] who presented a path-based formulation for the Minimum Cost Multicommodity UFP (also called the Integer Multicommodity Flow Problem) together with a Branch and Price and Cut algorithm. In this problem one aims at routing each commodity flow through a unique path at minimum cost. The developed branching strategy is based on the notion of divergence node for a fractional, thus non feasible, solution, i.e. the first vertex in the graph from which the integer commodity flow splits into two or more paths; a binary decisional tree is constructed by partitioning the arcs leaving the identified vertex into two sets, balanced w.r.t. the amount of flow, and by forbidding the usage of a subset in each child. Setting to a very high value the costs of the forbidden arcs, the pricing problem is modeled and efficiently solved as a Shortest Path Problem (SPP) for each single commodity.
The employment of k-splittable flows within the static environment of network flow modeling and with a general value $k \geq 2$ started in 2002 with the seminal paper by Baier et al. [7, 6]. In general, any optimization version of the UFP introduced by Kleinberg [69, 70] have found its extension to the k-splittable setting, as detailed in the reminder of this section. We summarize the main contributions related to k-Splittable Flow Problems ($kSFP$) in Table 4.1 w.r.t. complexity results and developed algorithms. For a thorough overview of k-splittable modeling tools we refer to [83].
Maximum and Minimum k-Splittable Flow Problem An instance of the Maximum $kSFP$ requires to maximize the flow between a pair of nodes such that it can be decomposed in at most k paths. Baier et al. [7, 6] proved the strongly NP-hardness of the problem by a reduction from the single-source Unsplittable Flow Problem (UFP) and showed that the results hold on directed graphs even for the single commodity case and for any constant value of $k \geq 2$. A $\frac{1}{2}$-approximation algorithm for the problem was presented too, derived

from solving in polynomial time the Maximum Uniform Exactly-$kSFP$ where k paths must be activated each of them carrying the same amount of flow. Koch et al. [72] modeled the Maximum $kSFP$ as a two-stage problem where a first packing step generates a set of candidate flow values among which at least one is proved to be contained in an optimal solution to the problem if k is a constant value, in a nearly optimal solution if k is part of the input; a second routing step tries to identify a collection of paths through which the generated flows can be routed while respecting arc capacities. In their contribution, a specific focus is put on graphs of bounded treewidth for which they proved the existence of a polynomial-time algorithm for solving the Maximum $kSFP$ if k is constant. In the case of k being part of the input a Polynomial-Time Approximation Scheme ($PTAS$) was presented. The work of Koch and Spenke [73] introduces refined and novel approximation and complexity results for the Maximum $kSFP$: the authors extended the complexity result to undirected graphs and proved that for $k > m - n + 2$, being $n \geq 3$ the number of nodes in the graph, and m the number of arcs, the problem becomes polinomially solvable. Finally, they derived the best bound on the approximability of the problem that equals $\frac{5}{6}$, unless $P = NP$. Complexity in the case of k being a function of the network parameters is investigated too.
Truffot et al. presented Branch and Price algorithms for the single and multicommodity version of the Maximum $kSFP$ [112, 111]. The problems, formulated with path-flow and path-design variables for each of the (at most) k path positions, are firstly linearized and further simplified by focusing on the solution space where coupling constraints are saturated. The designed pricing problem aims at identifying a shortest path with highest capacity and it is solved by means of a tailored polynomial algorithm. A first branching strategy works on the arc-flow space and resembles the one developed by Barnhart et al. based on the concept of divergence node; a second alternate branching imposes a subset of path positions to use or not a certain arc. The work of Gamst and Petersen [46] focused on a 2-index formulation of the problem where the path position indexes used in [112, 111] have been merged in a unique solution. In their novel Branch and Price approach two branching strategies have beed designed: the first one forbids the transshipment on subpaths emanated from the divergence node while the second rule, proved to outperform all others, forbids the usage of subsets of paths while forbidding the activation of others for a commodity. Further Branch and Price algorithms have been discussed and compared by Gamst et al. [45] for the Minimum Cost Multicommodity $kSFP$, where the k-splittable transshipment with minimum total costs must be identified. In particular, a heuristic method was introduced to faster reach feasible solutions by eliminating some symmetries in the 3-index model and a novel branching rule forbidding subpaths was developed for the 2-index formulation. Computational results present better performances of the 2-index formulation. Gamst investigated a different Dantzig-Wolfe decomposition for the Maximum Multicommodity $kSFP$ [42]. The strategy is based on the concept of path sets, i.e. a collection of at most k paths for a given commodity carrying a given amount of flow. In this scheme, the Restricted Master Problem identifies a combination of path sets (one for each commodity) that is feasible w.r.t. the arc capacities. The pricing problem, in charge of generating path sets, results in the NP-hard single-commodity Maximum $kSFP$. A tailored heuristic method and a mathematical formulation are presented for its resolution. The branching rule forces and forbids different subpaths and the so-generated cuts are accounted in both resolution methods with slight modifications. The method results competitive but presents some scaling issues due to the

complexity of the pricing problem. A local search heuristic for the same problem can be found in [43]: it iteratively looks for a shortest path in a reduced capacitated graph and assigns flow to it according to one among three designed strategies that differently account for congestion phenomena. The change of strategy occurs after k paths are identified with their respective flows for each commodity. The solution with the maximum total routed flow obtained by applying the three strategies is then returned.

Minimum Congestion and Maximum Concurrent k-Splittable Flow Problem Martens and Skutella [85] focused on minimizing congestion while adopting k-splittable flows, i.e. on finding the smallest $\alpha \geq 1$ such that routing the flow on at most k paths would violate arc capacities at most of a factor α. In particular, motivated by a real-world application where containers must be loaded and then shipped, they considered specific constraints limiting the amount of flow sent along each path. They showed that any ρ-approximation for the UFP provides a 2ρ-approximation for the considered problem. For the Minimum Congestion $kSFP$ with all commodities sharing the same source node, Caramia and Sgalambro developed a constant factor approximated algorithm [22], generalizing a result by Dinitz et al. [31] for the unsplittable counterpart and proposing a heuristic improvement that provides experimentally better results while preserving the approximation guarantee. Jiao et al. focused on k-splittable flows applied to multiple protocol label-switched ($MPLS$) networks with the aim of minimizing congestion while routing data traffic between clients in a capacitated telecommunication networks, see [65] and [63]. For the multicommodity single-source version of this problem they proposed a 2-index path-flow and a 3-index arc-flow Mixed-Integer Linear Programming formulations together with two different simple heuristics that at the first stage, make use of the free-flow relaxation of the problem to quickly identify transmission paths for each commodity. Then, the first heuristic selects for each commodity k paths among the active ones and when required solves a minimum congestion flow problem to reassign demand flow to them. The second heuristic solves for each commodity a restricted path-flow formulation of the original problem where only the active paths are available. The same authors proposed three heuristics for the bi-objective Minimizing Congestion and Cost in the multicommodity $kSFP$ in [64]. The strategies differ themselves on the type of relaxation employed to obtain an initial solution satisfying the commodity demands. Some recent studies on the complexity of the Minimum Congestion $kSFP$ can be found in the work of Jiao et al. [62], together with results on the relationship in the k-splittable setting between minimizing congestion and minimizing number of rounds, i.e. the number of commodity subsets such that for each of them a feasible k-splittable flow is proved to exist.

The multicommodity Maximum Concurrent $kSFP$, equivalent to the Minimum Congestion $kSFP$ as for in the unsplittable setting, has been investigated by Caramia and Sgalambro in [21] and previously in [108]: they presented a 3-index arc-flow formulation and a Branch and Bound algorithm that performs 0-1 branchings on binary variables which are responsible of bounding the number of active paths for each commodity. Fathoming rules, based on the definition of a minimal assignment for binary variables, are designed to speed up the identification of infeasible nodes and thus to reduce the size of the decisional tree to be explored. In another work the authors designed a fast two stage heuristic algorithm for the same problem [23]. The heuristic routes the flow using an augmenting path algorithm and then performs a local search routine in order to reroute it. The Randomized Rounding (RR) technique has been frequently employed in the context of $kSFP$s, often as a benchmark algorithm, starting

kSFP	Formulations	Complexity	Exact	Heuristics/Approx.
Max $kSFP$	[111, 112, 46, 43]	[7, 72, 73]	[111, 112, 46, 42]	[7, 43, 72, 73]
Min Cost $kSFP$	[45, 9]		[45, 9]	
Min Congestion and Max Concurrent $kSFP$	[21, 108, 65, 63]	[62]	[21, 108, 65, 63]	[23, 22, 64]
Dynamic $kSFP$				[85]

Table 4.1: Discussed state-of-the-art $kSFP$ contributions

from the Minimum Congestion UFP variant [101] with Raghavan and Thompson, and later moving to the Minimum Congestion or Maximal Concurrent $kSFP$, with Białoń, Martens and Skutella and Caramia and Sgalambro [13, 84, 23]. In particular, the RR technique in presence of the balance condition assumption, i.e. if the maximum demand is bounded from above by the minimum arc capacity, is the best known approximated approach for the Maximum Concurrent $kSFP$. A more detailed description of the RR scheme will be provided in Subsection 4.3.1.

Dynamic k-Splittable Flow Problem The concept of path restrictions has been rarely addressed in dynamic networks. The only result can be found in Martens and Skutella [85], where a $(3 + 2\sqrt{2})$-approximation algorithm for a single commodity Quickest k-Splittable Flow Problem with a continuous time parameter is provided.

We conclude the review by citing the recent work by Zhu and Xiaowenby where the concept of k-splittable flows was applied to mass spectra identification in bioinformatics [117]. The resulted novel NP-hard problem, named the Minimum Error kSP, presents node capacities that can be exceeded during the transshipment but whose total amount must be minimized. The proposed resolution algorithm mimics the two-stage strategy of Koch et al. [72] and is polynomial on layered digraphs and with $k = 2$.

4.3. Formulations and approximation results

The network flow setting for the representation of static k-splittable flows coincides with the one presented in Section 2.2. In particular, it is required a capacitated flow network $\mathcal{D} = (\mathcal{V}, \mathcal{A})$ and a set of commodities (o_h, d_h, σ_h). Moreover, an integer parameter k, namely the *splittable* or *flow split* parameter is provided. A *k-splittable flow* in this framework is represented by a static flow in the directed digraph where each commodity population is routed through at most k paths to its destination. Thus, the number of different activated paths, i.e. paths carrying a strictly positive amount of commodity flow, can not exceed the flow split parameter k.

Figure 4.1 and 4.2 provide a graphical representation of the effects of introducing path limitations in static flows for different values of the k-splittable upper bound. The flow network depicted in the top figure presents a single commodity with eight units of flow that must be sent from its origin node o to the destination node d at minimum cost; arc labels are indicated as a pair of unitary cost and capacity. In Case (a) no restrictions are imposed during

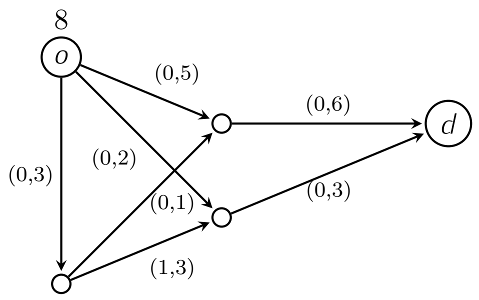


Fig. 4.1: Flow network with (cost,capacity) arc labels.

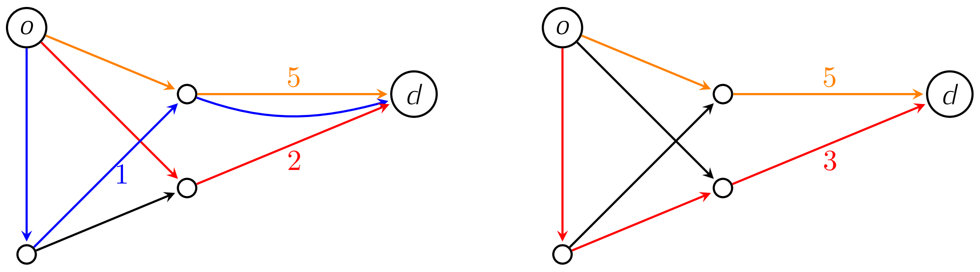


Fig. 4.2: Case (a) : the free-flow setting. Case (b) : the 2-splittable setting.

the flow transshipment: an optimal solution is thus represented by the three colored paths carrying 5, 2 and 1 units of flow each and with a total cost of zero. This solution results infeasible in Case (b), where the flow split parameter k has been set to two. An optimal solution is represented by the two colored paths with a total cost increased to three. Note that no feasible solution would have been identified in the unsplittable case as all available paths in the digraph can transship at most five units of flow.
In line with our research interest throughout this thesis, we now focus on multicommodity flows treating the single commodity as a special case. We provide an arc-flow and a path-flow formulation valid for general static k-Splittable Flow Problems. For the notation we refer to Section 2.2 while decision variables are collected in the following box. At the end of the subsection we present the Randomized Rounding (RR) technique, a heuristic resolution approach frequently used to provide good solutions to $kSFP$s.

k-Splittable flow variables:

x_{ij}^{hl} amount of flow of commodity h traversing arc (i, j) at path position l,
z_{ij}^{hl} binary variable is 1 if arc (i, j) is used by commodity h at path position l,

x_p^h amount of flow of commodity h leaving the source through path p,
z_p^h binary variable is 1 if path p is used by commodity h.

Arc-flow formulation We provide a state-of-the-art 3-index arc-flow representation of a feasible multicommodity k-splittable flow, see [112, 111]. The path number restriction is here accounted by limiting to k the number of subgroups in which each commodity demand can be partitioned and by formulating a network flow subproblem for each of the subgroups where the usage of the same unique path to destination is ensured. Formally, the subgroups are indexed by $l \in \mathcal{L} = \{1, \dots, k\}$ and the following decision variables are defined: a fractional flow variable x_{ij}^{hl} expresses for each commodity h, arc (i, j) and subgroup l the amount of flow of the given commodity associated to the given subgroup traversing the given arc, and a binary decision variable z_{ij}^{hl} takes value one whenever arc (i, j) is used by subgroup l of commodity h. Hence, a multicommodity k-splittable flow is said to be *feasible* if it respects the following constraints.

$$\sum_{j \in \delta^+(i)} x_{ij}^{hl} - \sum_{j \in \delta^-(i)} x_{ji}^{hl} = \begin{cases} \sigma_h^l, & i = o_h \\ -\sigma_h^l, & i = d_h \\ 0, & i \neq \{o_h, d_h\} \end{cases} \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}, i \in \mathcal{V}. \tag{4.1}$$

$$\sum_{j \in \delta^+(i)} z_{ij}^{hl} - \sum_{j \in \delta^-(i)} z_{ji}^{hl} = \begin{cases} 1, & i = o_h \\ -1, & i = d_h \\ 0, & i \neq \{o_h, d_h\} \end{cases} \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}, i \in \mathcal{V}. \tag{4.2}$$

$$\sum_{j \in \delta^+(i)} z_{ij}^{hl} \leq 1 \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}, i \in \mathcal{V}. \tag{4.3}$$

$$x_{ij}^{hl} \leq c_{ij} z_{ij}^{hl} \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}, (i, j) \in \mathcal{A}. \tag{4.4}$$

$$\sum_{h \in \mathcal{H}} \sum_{l \in \mathcal{L}} x_{ij}^{hl} \leq c_{ij} \qquad \forall (i, j) \in \mathcal{A}. \tag{4.5}$$

$$\sum_{l \in \mathcal{L}} \sigma_h^l = \sigma_h \qquad \forall h \in \mathcal{H}. \tag{4.6}$$

$$\sigma_h^l \geq 0 \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}. \tag{4.7}$$

$$z_{ij}^{hl} \in \{0, 1\} \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}, (i, j) \in \mathcal{A}. \tag{4.8}$$

$$x_{ij}^{hl} \geq 0 \qquad \forall h \in \mathcal{H}, l \in \mathcal{L}, (i, j) \in \mathcal{A}. \tag{4.9}$$

Constraints 4.1 and 4.2 impose separately for each commodity subgroup flow and path conservation on fractional x's and binary z's variables, respectively. This way a continuous transshipment from the origin to the destination is secured. Constraints 4.3, namely the *k-splittable constraints*, forbid each commodity subgroup to use more than one path during the transshipment. Note that thanks to these constraints the so-constructed paths present no loops as any node can be traversed at most once. Moreover, only cycles that are not attached to any activated o_h-d_h path can be identified by this formulation. Coupling constraints 4.4 impose the activation of a design variable z_{ij}^{hl} whenever part of the commodity flow h associated to subgroup l is routed through arc (i, j). Finally, constraints 4.5 require that arc capacities are respected and constraints 4.6 and non-negative variables 4.7 ensure that the entire commodity demands are partitioned into the indexed subgroups.

Path-flow formulation In a path-flow representation, a multicommodity k-splittable flow can be expressed as a collection for each commodity of at most k paths with their associated flow, i.e. $\{(p_1^h, f_1^h), (p_2^h, f_2^h), \dots, (p_k^h, f_k^h)\}$ where f_i^h is the amount of demand of commodity h routed through path $p_i^h \in \mathcal{P}_h$. In order to resemble the problem modellization with arc-flow variables, we require paths to be elementary. A 3-index and a 2-index path-flow formulations have been proposed in the scientific literature for multicommodity k-splittable flows, depending on whether an explicit discretization of flow in subgroups is considered or not [111, 46, 45]. In the first case a flow support variable x_p^{hl} is employed to express the amount of flow of subgroup l of commodity h that uses path p, and a binary design variable z_p^{hl} to ensure that the k-splittable restriction is observed by each distinct commodity. In the 2-index formulation, the different subgroup indexes are merged into a single unique variable leading to variables $x_p^h = \sum_{l \in \mathcal{L}} x_p^{hl}$. Similarly, variable $z_p^h = \max_{l \in \mathcal{L}} z_p^{hl}$ is now activated whenever commodity h uses path p. This simple flow aggregation/disaggregation step ensures the equivalence of the two formulations, as a solution of one representation can be transformed in one of the other and viceversa. In this paragraph we focus on the 2-index formulation; for the 3-index formulation see [111, 46, 45].

$$\sum_{p \in \mathcal{P}_h} x_p^h = \sigma_h \qquad \forall h \in \mathcal{H}. \tag{4.10}$$

$$\sum_{p \in \mathcal{P}_h} z_p^h \le k \qquad \forall h \in \mathcal{H}. \tag{4.11}$$

$$\sum_{h \in \mathcal{H}} \sum_{p \in \mathcal{P}_h} \delta_{ij}^p x_p^h \le c_{ij} \qquad \forall (i,j) \in \mathcal{A}. \tag{4.12}$$

$$x_p^h \le u_p z_p^h \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h. \tag{4.13}$$

$$z_p^h \in \{0,1\} \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h. \tag{4.14}$$

$$x_p^h \ge 0 \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h. \tag{4.15}$$

Constraints 4.10 impose the routing of the entire commodity demand through the available paths; Constraints 4.11 ensure that the bound on the number of active paths is respected by each commodity while arc capacities are accounted by constraints 4.12. Bottleneck Constraints 4.13, also called *coupling constraints*, relate flow support variable x's with binary variables z's: if path p is not used for flow transshipment by commodity h, then no unit of flow can be assigned to the related support variable x_p^h. Oppositely, flow on each path must respect the capacity of the path itself.
The path-flow formulation presents a very large number of variables as the number of paths for each origin-destination pair exponentially increases in the dimension of the dynamic digraph. On the other side, flow conservation constraints are not needed anymore as the amount of demand is preserved along each path and the k-splittable constraints can be now imposed in a compact way by simply counting the number of activated design variables for each commodity. Note that the path-flow formulation defines a tighter feasible region w.r.t. the arc-flow formulation as solutions with disconnected cycles are excluded.

4.3.1. An approximation result

In 1987 Raghavan and Thompson introduced the Randomized Rounding (RR) resolution approach for general 0-1 integer linear programs and applied it to solve, among others, the Minimum Congestion UFP with one-unit demand multiple commodities in undirected graphs [101]. At a first stage, their RR technique solves to optimality the free-flow relaxation of the problem, obtaining an edge flow assignment that might present for some commodities an unrestricted number of activated paths; from this, the used paths and their associated amount of flow are retrieved. Note that at this step, any path-decomposition algorithm such as the one of Ford and Fulkerson could be employed (see Decomposition flow Theorem in Section 4.3.1). A randomized procedure is then applied to select at random one of these paths for each commodity, with a probability that is equal to the path flow over the commodity demand. Finally, a feasible solution to the problem is obtained through a rounding step that assigns the entire commodity population to the selected path and a zero flow to the remainings.
Under the assumption that the balance condition holds, i.e. when the maximum commodity demand is bounded from above by the minimum arc capacity, the RR provides the best known approximation result for the problem, both in directed and undirected graphs, with a performance ratio of $\Omega(\log m / \log \log m)$ [84].
The RR has been later used as a basis for probabilistic approximation schemes for several $kSFP$s with $k \geq 2$. In this case, a feasible k-splittable flow solution is obtained by routing each commodity demand through at most k paths randomly chosen from the optimal free-flow assignment. As instance, the RR was employed for comparison purposes in the resolution of the Maximum Concurrent $kSFP$ by Caramia and Sgalambro [23]. Their rounding step solves to optimality an arc-flow linear programming formulation of the splittable relaxation of the problem, ensuring that only the chosen paths are used for routing flows. A recent RR-based method for the Minimum Congestion $kSFP$ has been presented in [13]. With the aim of reducing some episodes of low quality generated solutions, the proposed algorithm relies on a modified free-flow relaxation where large commodities are penalized on the links they could saturate. The path selection probability of [101] is here adjusted by a constant factor and the rounding step mimics the proportion of the splittable relaxation solution. The method provides an approximation factor of $O(\sqrt{logm})$ when $k = 1$. This research was motivated by an application in telecommunication field to support traffic management and routing in virtualized networks with restrictions on the minimum path load. Another real-world application of the RR, is represented by the so-called global routing phase in the wise placement design of VLSI circuits where the RR showed very good performances [92, 3].
In our research we will make use of a Randomized Rounding-based algorithm to compare performances of a novel matheuristic strategy developed for a dynamic k-splittable flow problem. Details will be provided in Section 6.3, where the adaptation of the RR procedure to the dynamic environment will be discussed.

4.4. Discussion and open contributions

It is evident from the rich scientific literature on k-splittable flows that the concept of path restriction plays a central role in securing precious optimization modeling tools for the support of strategic, tactical and operational planning of transshipment operations. Indeed, the incorporation of such path limitations allows for the representation of an increasing multitude of real-world situations, where a limited amount of resources is experienced and efficient solutions can be achieved only with a firm control on the routing processes.
However, most of the state-of-the-art $kSFP$s are still settled in the static environment. This modeling setting, as already observed in Section 2.4, might prevent the release of the complete potential that could be enabled by a mathematical representation and optimization based on k-splittable flows. Indeed, structural limitations of static network flows related to the absence of time-dedicated tools, make them fail to capture the time dimension of the transshipment phenomena and to monitor and manage the network utilization over time.
Hence, in order to increase benefits that can be achieved by adopting k-splittable flows as a modeling tool, a consistent step towards the dynamic setting should be taken, following the trace posed by several static contributions that have been already extended with success to the dynamic environment of Network Flow theory aiming at increasing their adherence to real-world contexts, see Section 3.2. The implementation of this step for k-splittable flows would open the door to novel optimization problems on one side and to a wider range of tractable real-world applications on the other.
Our research line goes in this direction, formally introducing for the first time a novel dynamic flow problem integrating the concepts of quickest flows with k-splittable flows and providing two tailored resolution approaches for the identification of optimal and good quality solutions to the considered problem.

5. Introducing the Quickest k-Splittable Flow Problem

5.1. Introduction

In the previous chapters significant optimization problems and results of the Network Flow theory have been presented, with more emphasis placed on the dynamic setting for flow modeling and on the concept of k-splittable flows, being the first suitable to capture and represent time dimension in flow transshipment, and the second fit to guarantee routing processes through at most k paths.
In this chapter we finally get to the core of our research, where these two aspects are merged together in a novel optimization problem that results in the first modellization of flow transshipments over time where the number of usable paths is bounded by a prefixed parameter. In particular, we integrate here the path number limitation secured by k-splittable flows within the Quickest Flow class, whose scientific relevance and practical potential is widely recognized and has been largely discussed in Section 3.5. Indeed, modeling approaches based on quickest flows result particularly appropriate when one aims at minimizing the completion time for a set of processes, a situation that frequently occurs as instance in telecommunication, evacuation and transportation management [61, 75, 87, 27]. The aim pursued with this research line is to increase the impact of the Quickest Flow optimization class in real-world contexts by encompassing the relevant modeling tool of path restrictions.
As detailed in the Section 1.1 on motivations for this work, the interest in this setting is supported by a large variety of potential applications sharing as a common requirement a thorough control on both the time spent for flow transshipment and on the number and characteristics of the activated support paths. Such contribution might be relevant for instance when modeling tactical planning operations in distribution logistics, particularly for the case of perishable goods or medical supply chains as in these contexts one seeks for a quick transshipment process while taking properly into account the fleet size and planning goods dispatching with a medium level of detail. Similarly, when adopting flows over time models for planning and managing emergency transportation, a control on the number of different paths will enable an increasingly supervised evacuation process, which would likely yield a substantial reduction in the risk of those unforeseen episodes such as interferences, turbulences, and congestions that often affect the transportation process worsening the overall clearance time. When it comes to the context of telecommunication, combining multicommodity quickest flows tools with k-splittable routing would result in quick data-packets transmissions where the overload in the used devices and protocols, often related to

an excessive amount of support paths, is successfully reduced.
All of the above mentioned examples show how imposing a realistic number of usable paths represents an essential and relevant tool when planning movements of objects over time in network-based structures. Nevertheless, the scientific literature of flows over time presents a lack of contributions combining these two modeling aspects, with only one work active in this framework [85]. Note that quickest flow-related contributions such as the basic variant of the Quickest Flow Problem or the Quickest Path Problem, are not able to capture this advanced setting. Indeed, the QFP allows flows to be spread on an unlimited and arbitrarily high number of different paths, resulting thus unsuitable in many real cases; on the other side the QPP imposes a limitation to only one single path for each source-destination pair, turning often to be far too restrictive and equivalently not realistic.
In our research activity we aim at bridging this gap, introducing the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$) that accounts for a limited number (k) of paths to be allowed in dynamic flow routing over a network, combining the requirement of a quickest (dynamic) multicommodity flow with path restrictions on each distinct commodity. In Section 5.2 we formally introduce this problem by providing a Mixed-Integer Linear Programming formulation. Then, in Section 5.3 its computational tractability is investigated. The tailored matheuristic and exact algorithmic contributions designed for its resolution will be covered in the next chapters of this thesis. Part of the outcome and results of this research have been recently published and the reader can refer to [88].

5.2. Problem statement and formulation

The Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$) is modeled in a capacitated dynamic digraph $\mathcal{D} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$, with a discretized planning horizon, time- and flow-independent arc attributes. An integer k, i.e. the flow split parameter, is provided, together with a set $\mathcal{H}$ of commodities each with a given origin, destination and amount of population that has to be transshipped within the considered time horizon.
In this setting, the $QMCkSFP$ asks for routing and scheduling each commodity flow through at most k different paths (namely, paths differing from each other at least in one arc) while minimizing the number of time instants needed to accomplish the process (makespan). Whilst computing the optimal dynamic routing, shared arc capacities have to be obeyed. Finally, in line with the discussions presented in the previous chapters, see Section 3.1 and 3.5, and in order to provide feasible and effective support to many real-world operations, the holdover of each population is allowed only at the respective source node and the usage of elementary paths is strictly required. The following box integrates the notation presented in Section 3.3 and provides the decisional variables used in the original path-flow formulation of the $QMCkSFP$.

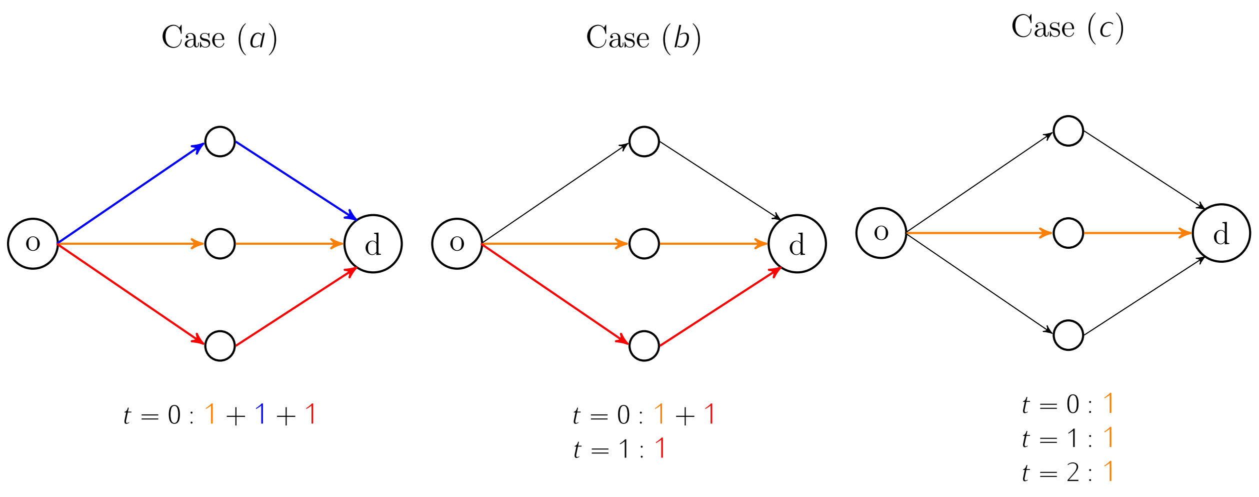


Fig. 5.1: Effects of path restrictions on quickest flows.

Quickest k-Splittable path-flow notation and decision variables:

$C_{ht} = \min\{\sigma_h, \sum_{p\in\mathcal{P}_h:(l_p\leq t)} u_p\}$ maximal demand of h allowed to arrive at time t,

x^h_{pt} amount of flow of commodity h leaving the source through p at time t,
y^h_t binary variable is 1 if some flow of commodity h arrives at destination at time t,
z^h_p binary variable is 1 if path p is chosen by commodity h.

Figure 5.1 depicts the effect of path restrictions on dynamic network flows for different k-splittable values. In particular, a 3-unit commodity from node o to node d has to be transshipped as quickly as possible in the given dynamic digraph where all arcs present capacities and delays equal to one. Case (a) represents an optimal solution to the unrestricted free-flow setting: the entire demand is routed at time zero through all of the available paths, each of them carrying one unit of flow, and the process is completed by time 2. In the next two cases the number of usable paths is progressively reduced, forcing the flow to be scheduled at successive time instants hence leading to a consequent increase in the makespan value. Indeed, in the 2-splittable setting, Case (b), the colored paths are both activated at time zero to transship one unit of flow and then, in order to satisfy the shipment of the entire commodity demand within the quickest possible time horizon, the remaining unit of population is scheduled at the first available time step on one of the already activated paths, e.g. the red one in figure. The optimal makespan associated to this solution equals 3. Finally, Case (c) forces the commodity to employ only one path, therefore imposing to route the population at three consecutive time instants starting from time zero on the same orange path. Such a strict path limitation leads to a further increase in the makespan value, which now turns to 4.

Path-flow formulation We provide a 3-index path-based formulation for the $QMCkSFP$ where transshipment of commodity flows are modeled in terms of activated paths. Fractional flow support variables x^h_{pt} are in charge of tracking the release of flows over time for each commodity and path, being all paths for commodity h collected into the $\mathcal{P}_h$ set. Additional binary variables are introduced in the problem modellization: time-arrival y^h_t variables allow for the linearization of the makespan minimization by recording arrival times at destination of each routed commodity demand, while path design z^h_p variables implement k-splittable path restrictions by bounding the number of activated paths for each commodity. A feasible dynamic flow for the $QMCkSFP$ is therefore completely identified by a set of at most k paths for each commodity h, namely $\{p^h_1, p^h_2, \dots, p^h_k\}$, with corresponding flow values x^h_{pt} at the specific departure time instants, satisfying the network arc capacities and the commodity demands.

$$\min \zeta \tag{5.1}$$
$$t y^h_t \leq \zeta \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{5.2}$$
$$\sum_{p \in \mathcal{P}_h} x^h_{p(t-l_p)} \leq C_{ht} y^h_t \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{5.3}$$
$$\sum_{p \in \mathcal{P}_h} \sum_{t \in \mathcal{T}} x^h_{pt} = \sigma_h \qquad \forall h \in \mathcal{H}. \tag{5.4}$$
$$\sum_{p \in \mathcal{P}_h} z^h_p \leq k \qquad \forall h \in \mathcal{H}. \tag{5.5}$$
$$\sum_{h \in \mathcal{H}} \sum_{p \in \mathcal{P}_h} \delta^p_{ij} x^h_{p(t-t^p_i)} \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}. \tag{5.6}$$
$$x^h_{pt} \leq u_p z^h_p \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h, t \in \mathcal{T}. \tag{5.7}$$
$$y^h_t \in \{0,1\} \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{5.8}$$
$$z^h_p \in \{0,1\} \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h. \tag{5.9}$$
$$x^h_{pt} \geq 0 \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h, t \in \mathcal{T}. \tag{5.10}$$
$$\zeta \geq 0 \tag{5.11}$$

For seek of simplicity whenever $t + l_p > T$ or $t < 0$ for a given path p and departure time instant t, we set the related x^h_{pt} variable to zero.
The objective function seeks to minimize the overall makespan, represented by the ζ variable, i.e. the number of time instants required to accomplish the transshipment of all the commodity demands. Constraints (5.2) identify for each commodity the time arrivals of its population, imposing a lower bound to the total makespan. Constraints (5.3) are used to couple flow variables x with time-related activation variables y: if no units of flow of commodity h arrive at destination at time t, then the flow routed on any path p of commodity h at time $t - l_p$ must be equal to zero. The maximum amount of flow of commodity h that can reach its destination at a given time t, i.e. C_{ht}, is here adopted as a parameter to enhance constraints' tightness. Constraints (5.5), namely k-splittable constraints, force each commodity to use at most k different paths. The amount of flow to be transshipped within the time horizon is stated by Constraints (5.4) while arc capacities at each time step are

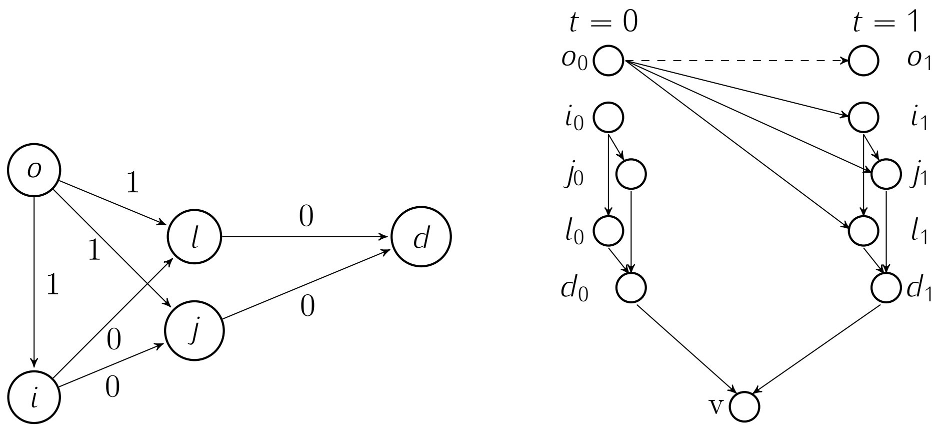


Fig. 5.2: Computational complexity reduction process.

accounted by Constraints (5.6). Bottleneck Constraints (5.7) impose the amount of flow on a given path to be null at every time instant if the path has not been selected, otherwise it must respect the capacity of the path.

5.3. Complexity

In this subsection we prove the strongly NP-hard computational time complexity of the introduced $QMCkSFP$ by reduction from the Minimum Cost $kSFP$. The complexity of the latter can be easily deduced from the works by Baier et al. [7] and Koch and Spenke [73] even for the single commodity version and for any constant $k \geq 2$. Figure 5.2 depicts the construction of the network used in the reduction process, starting from the dynamic network on the left. Consider a source node o and a sink node d in it, a one-unit planning horizon $\mathcal{T} = \{0, 1\}$, non-negative capacities on the arcs and zero travel times except for the outgoing arcs originating from the source for which we set $\lambda_{oj} = 1$, $\forall j \in \delta^+(o)$. Intermediate nodes and arcs in the figure are purely indicative. Expand this digraph over the considered time horizon applying the time-expansion procedure defined by Ford and Fulkerson [39, 38] and described in Subsection 3.4.1. Nodes of the TEN have been renamed with an apex according to the related time layer and holdover arcs have been depicted only at the source node to model delayed departure. The digraph on the right is obtained by adding to the standard TEN an extra node v and by connecting it with incoming arcs from the time-replica of the destination node, i.e. d_0 and d_1. In this static network, we construct an instance of the single commodity Minimum Cost $kSFP$ setting a demand of σ units that has to be transshipped from o_0 to node v; costs are equal to zero except for arc (d_1, v) that presents a cost equal to one. It is trivial to see that a feasible o_0-v k-splittable flow of value σ exists in the static digraph iff a quickest o-d k-splittable flow with makespan 1 exists within the considered planning horizon. We can thus conclude that the single commodity Quickest $kSFP$ is strongly NP-hard, being at least as hard as the single commodity Minimum Cost $kSFP$. The validity of such result can be generalized to the multicommodity version as well, namely the variant considered in the next chapters of this thesis.

6. A Matheuristic approach for the Quickest Multicommodity k-Splittable Flow Problem

6.1. Introduction

In the previous chapter we introduced a novel dynamic flow optimization problem, namely the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$), that explicitly integrates the concept of k-splittable flows, discussed in Chapter 4, within flows over time, presented in Chapter 3. For this problem a $MILP$ path-based formulation was provided, and its computational complexity was analyzed in Section 5.3, where the $QMCkSFP$ was proved to fall in the class of strongly NP-hard problems by reduction from the Minimum Cost $kSFP$.

The complexity result poses relevant questions related to the development of efficient strategies to solve the proposed $QMCkSFP$, in particular when dealing with practical applications, often modeled by instances of extremely large size. Recall indeed, that for strongly NP-hard problems it is known how no psuedo-polynomial time resolution algorithms can be designed for their resolution, unless $\mathcal{P} = \mathcal{NP}$, see [48, 47]. Hence in these situations, heuristic and approximated strategies are frequently employed, focusing the computational effort on a fast identification of high quality solutions rather than on the generation of provably optimal solutions. Among them, matheuristics methods, sometimes called *model-based metaheuristics*, have been intensively investigated in recent years, resulting nowadays an attractive topic in the field of optimization [8, 14, 15, 110]. Such resolution techniques hybridize (meta)heuristics and mathematical programming algorithms in several schemes, thus combining time efficiency and methodological rigor. As instance, heuristic methods can require the resolution to optimality of a sequence of subproblems via mathematical programming while constructing a complete feasible solution or improving the quality of an identified one. In the former case, the resulting method can be viewed as a decomposition approach and applications can arise in fleet and crew planning among others; in the latter, the integration can be implemented by exactly exploring large-scale neighborhoods, as often occurs in vehicle routing problems. On the other side, mathematical programming approaches such as Branch and Bound and Column Generation can be "relaxed" to generate approximated solutions, as instance by introducing heuristic decisions in MIP solvers, or to identify valid bounds to the optimal solution of the original problem that can be later exploited by heuristics. We refer

the reader to the work by Ball [8] for a detailed classification of matheuristics and to the book by Talbi [110] for real-world applications.
As a first step towards the resolution of the newly introduced $QMCkSFP$, our research activity focused on the design of an ad-hoc matheuristic approach, integrating the path-based formulation described in Section 5.2 into a metaheuristic framework with the aim of finding efficiently good quality solutions for large instances of the problem. The developed algorithm is a hybrid Very Large-scale Neighborhood Search that employs a mathematical programming strategy in its exploration routine. At each iteration a neighborhood is constructed by identifying a collection of paths for each commodity and then explored to optimality by solving, via a MIP-solver, the path-based formulation of Section 5.2 restricted to the current selected paths. The improvement search proceeds through a Variable Neighborhood Descent scheme generating multiple large neighborhoods by increasing the cardinality of the identified sets.
In Section 6.2 we provide full details of the original matheuristic algorithm, describing its tailored construction and improvement phases. Moreover, we present the dynamic version of the Randomized Rounding heuristic (RR) that we implemented for comparison purposes. We refer the reader to Section 4.3.1 for the standard RR in the static framework. Finally, in Section 6.3 the design of experiments and the computational results are discussed. In particular, in Subsection 6.3.1 a proof-of-concept of our model is provided by solving a set of reduced-size instances to optimality via a commercial MIP-solver making use of the introduced path-based formulation. Subsection 6.3.2 is devoted to the evaluation of the matheuristic's performance: it starts with a detailed description of the benchmark test sets utilized in the experiments and presents the tuning process carried out on the matheuristic's parameters. The next paragraph is dedicated to prove the correctness and effectiveness of the developed algorithm. To this aim, a comparison is performed on a set of small to medium-sized instances against the free-flow relaxation of the considered problem with no upper bounds on the number of paths, namely the Quickest Multicommodity Flow Problem. In the last part of the subsection we test performances of our matheuristic against those of the RR-based algorithm on two different benchmark test sets, where the first collects networks of increasing size in terms of number of nodes and arcs and the second network structures with an extremely high number of commodities.

6.2. A Matheuristic approach

The matheuristic algorithm developed to solve the $QMCkSFP$ builds on a MIP-based Very Large-scale Neighborhood Search employing a Variable Neighborhood Descent (VND) scheme for constructing and visiting multiple large neighborhood structures. Recall that the standard VND technique introduced by Hansen and Mladenovic [59] performs a local search on multiple neighborhood structures ordered in list to enhance intensification and avoid stopping at local optima. Any time an improved solution is found while exploring a certain neighborhood, the VND restarts the local search with the first neighborhood centered at the new incumbent; otherwise it moves to the next structure in the list. Our hybridization of the standard Very Large-scale Neighborhood Search scheme occurs in the local search routine where a mathematical programming method is embedded to explore the large search space.

Indeed, each neighborhood structure matches a restricted collection of paths for each commodity and its exploration is realized by solving to optimality the related path-flow model of Section 5.2 via calling a commercial MIP-solver, thus providing the local optimum in the current large neighborhood. In the following paragraphs we detail information about the matheuristic and provide its pseudocode description in Algorithm 2.

6.2.1. Generation of candidate paths and initial solution

Each neighborhood structure considered in our matheuristic is meant to identify a finite collection of paths for each commodity. Thus, we restrict the number of candidates focusing on a large set of promising paths for each commodity and let the neighborhood structures extract their choices from it. More in detail, we generate for each commodity a finite array of feasible paths ordered accordingly to their transmission time, i.e. the time required for transshipping the demand along it. Recall that a path is considered as feasible if at least a unit of population can complete the transshipment within the planned time horizon. The construction of the arrays is performed by adopting the algorithm for ranking quickest loopless paths described in Section 3.4.2 and proposed by Pascoal et al. [96, 97]. We apply this algorithm separately to each commodity h and get in output the array L_h of quickest paths ranked in increasing order of transmission time, see Instruction 1 of Algorithm 2. Note that the length L of the arrays must be calibrated in order to balance the tradeoff between quality and speed.
The construction of the initial solution, see Instruction 2 of Algorithm 2, is performed as follows: for each commodity the first k paths in the respective candidates' array L_h are selected, i.e. the k top-ranked quickest paths, then the path-based model presented in Section 5.2 is fed with such paths by populating the P_h sets, and finally the MIP-solver is called and the optimal solution is returned together with its associate makespan value.

6.2.2. Improvement phase

The aim of this phase is to iteratively improve the current best solution by exactly exploring multiple large search neighborhoods. It represents a crucial step that strongly influences the result of the overall approach: as a rule of thumb, the better the generated neighborhoods, the closer the solution to a global optimum. The process stops when a time limit is reached and the best solution found is returned. In the following, the finite list of neighborhood structures is indicated as $N^s,\ s = 1, \ldots, s_{max}$.

Neighborhood structures The construction of a neighborhood is linked to the identification for each commodity of a finite collection of feasible paths extracted from the related L_h array. In this way, elements of a neighborhood are all the feasible solutions to the $QMCkSFP$ where only the identified paths can be used for the multicommodity flow transshipment. The collections are generated by including all paths used by the current incumbent (at most k for each commodity due to the k-splittable constraints (5.5)), and by selecting additional paths at random from the candidates' arrays until the desired cardinality is fulfilled. Formally, we express a given neighborhood centered at the incumbent solution $\bar{x}$ with structure type s as $N^s(\bar{x}) = \bigcup_{h\in\mathcal{H}} N^s_h(\bar{x})$, where each $N^s_h(\bar{x})$ represents the collection of paths for commodity h

identified by the adopted heuristic rule, i.e. $N_h^s(\bar{x}) \leftrightarrow \{p_{s_1}^h(\bar{x}), \dots, p_{s_l}^h(\bar{x}), p_{s_{l+1}}^h, \dots, p_{s_S}^h\}$. Note that each collection presents the same cardinality of paths among all commodities. Moving forward in the list of neighborhood structures in a VND fashion, the number of paths to be identified for each commodity increases by a factor of the flow split parameter k, i.e. $|N_h^s(\bar{x})| = k\,(s\,\Delta+1)$, being $1 \leq s \leq s\text{-}max$ and Δ a matheuristic's parameter that permits to control the growth factor of the neighborhood structures, see Instruction $3a.$ of Algorithm 2. Note that the value of the Δ parameter directly affects the number of feasible paths to be identified when constructing a new neighborhood in the list. Hence, a variation on its value allows to regulate and balance speed and accuracy in the exploration of the search space.

Exact local search Once the collections of paths have been identified by the current neighborhood, the $\mathcal{P}_h$ sets of the path-flow formulation are updated accordingly, i.e. $\mathcal{P}_h = N_h^s(\bar{x}),\ \forall h \in \mathcal{H}.$ Then, the generated model is solved to optimality by means of a MIP-solver, Instruction $3b.$ in the pseudocode. To speed up the execution times and perform a higher number of runs, the large search space is restricted imposing an upper bound on the makespan which must be at most equal to $val(\bar{x}) - 1$. Note that by considering increasingly larger collections of paths, the size of the model to be solved increases but the solver gets wider degrees of freedom to identify an optimal multicommodity combination of routes and flow schedules over time.

Acceptance decision The upper bound introduced in the path-flow formulation guarantees that a feasible solution returned by the exact local search represents a new incumbent for the original $QMCkSFP$. In this case the new solution is accepted, Instruction $3c.$, and the matheuristic algorithm restarts with the first neighborhood structure in the list centered at the updated incumbent, see Instruction $3d.$ in Algorithm 2. If no improvement is obtained in the current neighborhood, meaning that no feasible solution was obtained by the exact local search, see Instruction $3e.$, the *r-max* parameter is checked to choose between re-generating a new neighborhood with the same size or skipping to the next neighborhood structure. Moreover, the *s-max* parameter states the end of the neighborhoods list and once reached the algorithm is forced to restart from the first structure.

Algorithm 2 Matheuristic for the $QMCkSFP$

```
Matheuristic's Parameters:
$L$;              length of the candidate paths' lists
$s\text{-}max$;   length of the neighborhood structures' list
$r\text{-}max$;   maximum number of re-iterations without improvement for each neighborhood
$\Delta$;         growth factor of the neighborhood structures
Other Input Parameters:
$k$;              flow split parameter
$time\text{-}lim$;  matheuristic time limit
Output:           $(\bar{x}, val(\bar{x}))$ a feasible solution and its makespan

Initialization
$s := 1$;
$r := 1$;
$val(\bar{x}) :=$ MAX-VAL;

1: Generation of the candidate paths' lists
   for $h \in \mathcal{H}$:
     $L_h \leftarrow Pascoal(L, h)$;

2: Construction of the initial solution
   for $h \in \mathcal{H}$:
     $\mathcal{P}_h \leftarrow L_h[0 : \text{k-1}]$;
   $(\bar{x}, val(\bar{x})) \leftarrow solveMIP(\mathcal{P}_h, val(\bar{x}))$;

3: Improvement phase
   do{
     for $h \in \mathcal{H}$:                                    ▷ $3a$. construction of the neighborhood
       $N_h^s(\bar{x}) \leftarrow generateNeigh(h, \bar{x}, k, s, \Delta)$;
       $\mathcal{P}_h \leftarrow N_h^s$;

     $(x, val(x)) \leftarrow solveMIP(\mathcal{P}_h, val(\bar{x}))$;                 ▷ $3b$. exact local search

     if $(val(x) < val(\bar{x}))$:                               ▷ $3c$. acceptance decision
       $(\bar{x}, val(\bar{x})) \leftarrow (x, val(x))$;                  ▷ $3d$. re-centering and restart
       $s \leftarrow 1$;
       $r \leftarrow 1$;
     else:                                              ▷ $3e$. regeneration or change neighborhood
       if $(r < r\text{-}max)$:
         $r \leftarrow r + 1$;
       else if $(s < s\text{-}max)$:
         $s \leftarrow s + 1$;
       else:
         $s \leftarrow 1$;
         $r \leftarrow 1$;
   }while$(time \leq time\text{-}lim)$;
   return $(\bar{x}, val(\bar{x}))$;
```

6.2.3. A competing approach

We adopt a dynamic version of the Randomized Rounding (RR) algorithm as a competing approach to our matheuristic on very large-sized instances where optimal values are not available as a benchmark for quality solution. The original RR technique, previously detailed in Section 4.3, was proposed by Raghavan and Thompson [101] to tackle large instances of static k-Splittable Flow Problems [13, 23]. We adapt their RR heuristic to our dynamic case as follows. The free-flow relaxation of the problem, equivalent to the NP-hard $QMCFP$, is solved to optimality through a binary search on the given time horizon that iteratively looks for a feasible multicommodity flow on a reduced TEN where the expansion is performed over a shortest time horizon. The solution that accomplishes the transshipment in the minimum time is returned as the optimal solution to the free-flow relaxation $QMCFP$. The Decomposition flow Theorem, see Section 2.3, is then recalled to design a procedure aimed at decomposing the optimal free-flow and obtaining a list of paths in the original dynamic graph, each with its respective flow over time, i.e. $(p, \sum_{t\in\mathcal{T}} x^h_{pt})$, $p \in \mathcal{P}_h$. The steps described so far can be viewed as the RR initialization procedure to generate a list of candidate paths that will be used throughout its implementation. From our tests this resolution approach results to require longer, but still acceptable, times w.r.t. the Pascoal's strategy adopted in the first stage of our matheuristic.
The RR initial solution is constructed by choosing at random k of the decomposed paths for each commodity, each path with a probability proportional to its associated flow over time, by storing them into the $\mathcal{P}_h$ sets and by solving to optimality the restricted path-flow formulation of the original $QMCkSFP$, see Section 5.2. Note that Constraints (5.5) in this case are redundant.
The improvement step, named here the randomization phase, randomly reprocesses the k path selection and the related MIP resolution. The heuristic stops when a time limit is reached, providing the best solution found w.r.t. the objective function.

6.3. Computational Experiments

In this section we present computational experiments conducted to solve the $QMCkSFP$ with the proposed matheuristic. In all experiments we gave our algorithm one hour of running time and 72 time instants as time horizon to perform the transshipment. In Subsection 6.3.1 we provide a proof-of-concept of our model, evaluating the performance of the matheuristic against a Branch and Cut-based MIP-solver solving to optimality the developed path-based formulation. This is done on a set of grid networks of reduced size. In Subsection 6.3.2 we present the three benchmark testbeds from the literature of static k-Splittable Flow Problems that have been selected and adapted to our dynamic framework: the Grid test set, the Dense test set [23] and the Carbin test set [44]. The matheuristic's parameters tuning conducted using the *irace* package [81] is then discussed. In the next paragraphs we present different experiments performed with the so-calibrated matheuristic: first on the Grid test set we prove the correctness and effectiveness of our matheuristic using the free-flow relaxation of the problem as a benchmark value. Second, on the Dense test set we assess the scalability of the developed algorithm w.r.t. the size of the networks. Finally, we evaluate

Table 6.1: Grid test set: b identifies the commodities' combination and k varies in $\{1,...,6\}$.

Instance	nodes	arcs	commodities	Instance	nodes	arcs	commodities
g-2-b-k	50	185	1,2,3,4,5	g-7-b-k	175	710	6,12,18,24,30
g-3-b-k	75	290	2,4,6,8,10	g-8-b-k	200	815	7,14,21,28,35
g-4-b-k	100	395	3,6,9,12,15	g-9-b-k	225	920	8,16,24,32,40
g-5-b-k	125	500	4,8,12,16,20	g-10-b-k	250	1025	9,18,27,36,45
g-6-b-k	150	605	5,10,15,20,25				

Table 6.2: Dense test set: the number of commodities is fixed to 5 and k varies in $\{1,...,6\}$.

Instance	nodes	arcs	Instance	nodes	arcs
d-10-k	10	45	d-100-k	100	4950
d-20-k	20	190	d-150-k	150	11175
d-30-k	30	435	d-200-k	200	19900
d-40-k	40	780	d-250-k	250	31125
d-50-k	50	1225	d-300-k	300	44850
d-60-k	60	1770	d-350-k	350	61075
d-70-k	70	2415	d-400-k	400	79800
d-80-k	80	3160	d-450-k	450	101025
d-90-k	90	4005	d-500-k	500	124750

the matheuristic's performance when the number of commodities to be routed is extremely high if compared to the size of the network structure: to this aim, the Carbin test set is considered as a final benchmark in our computational framework. In the second and the third set of experiments the matheuristic is compared against the RR algorithm described in Subsection 6.2.3. All techniques are implemented in the C++ language and experiments conducted using the ILOG CPLEX v.12.6.0.0 solver in parallel deterministic mode (up to 20 threads) on a 64bit Intel Xeon CPU at 2.80GHz with 64 GB memory, running Ubuntu 14.04.2.

Table 6.3: Carbin test set: a identifies the level of congestion and k varies in $\{1,...,6\}$.

Instance	nodes	arcs	commodities
Ba01-k	32	96	48
Ba03-k	32	96	48
Ba05-k	32	320	48
Ba07-k	32	320	48

6.3.1. Solving the path-based formulation to optimality on reduced size instances

In this first computational experiment we compare our algorithm against CPLEX solving the $QMCkSFP$ formulation presented in Section 5.2. The testbed is composed of a collection of grid instances of reduced size, such that a complete enumeration of all the available paths for each commodity is possible and the resulting dimension of the MIPs can be handled by the solver. Each instance has been tested with three different combinations of commodities and from the unsplittable to the 6-splittable case. CPLEX was given 3600 seconds of time limit and 72 time instants as time horizon. The matheuristic's parameters have been set as follows: the length L of the lists of candidate paths to 100, the re-iteration parameter *r-max* to 1000, the Δ growth factor parameter to 1 and the length of the neighborhood structures' list *s-max* to $\lfloor (L-k)/(k\,\Delta) \rfloor$. The complete results of the experiment are provided in Appendix A where the name s-a-b-k in the first column identifies a grid instance with a connected layers each with 3×3 nodes and 24 directed arcs from 1 to 10 time periods long; arcs within the same layer present a very large capacity while arcs connecting different layers represent bottlenecks for the demand flows; the number of commodities equals $b(a-1)$ and k stands for the flow split parameter. The next four columns recall features of the instance in terms of the number of nodes and arcs, "nodes" and "arcs" columns, number of commodities, "h" column, and the flow split parameter, "k" column. In the next five columns some key results obtained by our matheuristic are reported: the makespan of the initial solution, "init sol" column, the time "t" in seconds needed for its construction, i.e. Instruction 1. and 2. of Algorithm 2, the makespan of the best solution identified after the improvement phase, "best sol" column, the time "t" in seconds needed for its identification and the number of intermediate incumbents found during the one-hour local search procedure, "moves" column. The last two columns present the optimal solution obtained by CPLEX "opt sol" and the computational time "t" required for the resolution of the problem.
Results show that the best solution provided by our matheuristic matches the optimal solution returned by the exact method in all the considered instances. In 50% of the cases the optimal solution is found by our matheuristic instantly in the initialization phase; in the rest of the cases it is reached in average after two intermediate moves and in less than one second while CPLEX requires at least an order of magnitude higher of time. The significant increase in the computational times in the last instance s-3-3-k is motivated by the large number of variables CPLEX has to deal with in this specific network: one commodity presents around 44000 paths, a value that is almost 75 times larger than the average number of paths occurred in the previous instances. Due to the need of feeding the formulation with an explicit enumeration of the complete set of feasible paths for each commodity, an extended comparison on further higher-sized instances would result impracticable. Nevertheless, with this preliminary experiment we proved that our matheuristic is able to obtain the same optimal solutions of CPLEX in considerably smaller computational times.

6.3.2. Evaluation of the matheuristic's performance

We proceed to validate the proposed algorithm and test its performances in terms of solution quality, scalability and computational times. We first present the features of the considered benchmark test sets, then we describe the thorough parameter tuning performed and finally the experiments conducted on each type of test sets with the so-calibrated matheuristic are analyzed in separate paragraphs.

Benchmark instances Three benchmark testbeds from the literature of static k-Splittable Flow Problems have been selected and adapted to our dynamic framework as follows. The Grid test set includes 9 networks, each of them tested with 5 different combinations of commodities and from the unsplittable setting to the 6-splittable case. Table 6.1 presents features of the set: a grid instance named g-a-b-k has a connected layers each with 5×5 nodes and 80 directed arcs from 1 to 10 time periods long; bottleneck arcs are placed only between different layers. The type of the commodities' combination and the flow split parameter are identified by the b and k values, respectively. Instances have been calibrated in order to get a fixed optimal value of 24 when solving the free-flow relaxation of the problem, i.e. the $QMCFP$. Note that its optimal makespan is a valid lower bound for the $QMCkSFP$ for any value of k, as the multicommodity flow is unrestricted during the transshipment. A valid certificate of optimality for our matheuristic is therefore a makespan value exactly equal to 24 time instants. The quantitative and qualitative analysis carried out on this experiment is presented in "The Grid test set" paragraph.
Table 6.2 reports the features of the Dense test set, identifying each of the considered 18 instances as d-a-k with a representing the number of nodes and k the value of the flow split parameter varying in $\{1, 2, \ldots, 6\}$. As in a dense instance each node i is connected only to node j s.t. $i < j$, the number of arcs results to be equal to $a(a-1)/2$. Their lengths have been randomly chosen between 1 and 10 time units. The number of commodities is fixed to 5 in all the test set. We report our comprehensive analysis on this experiment in the paragraph "The Dense test set".
The Carbin test set is composed of 8 networks divided into two subgroups according to the congestion ratio *mean capacity of arcs/mean demand of commodities*: the B*s* subgroup presents a small congestion ratio while the B*l* a large one. Each instance has 32 nodes, 48 commodities and a number of arcs equal to 96 or 320; arcs lengths range in the $[1, 10]$ interval. The collection is presented in Table 6.3 with each row Bab-k representing the Carbin instance with level a of congestion ratio and k as flow the split parameter, always varying from the unsplittable to the 6-splittable case. Results are analyzed and interpreted in the last paragraph of the current subsection.

Parameter tuning A parameter tuning process has been conducted on our matheuristic, tailored on the benchmark test sets to be solved. For this purpose the automatic algorithm configuration method *irace* [81] has been applied to the following matheuristic's parameters: the length L of the candidates' lists, the *r-max* parameter and the growth factor of the neighborhood structures Δ. Their domains have been set to $L = \{50, 100, 150\}$, *r-max* $=\{500, 1000, 1500\}$ and $\Delta = \{0.5, 1, 1.5\}$, respectively. Note that the remaining matheuristic's parameter *s-max* directly depends in turn on the cardinality L of the lists, being it equal to $\lfloor (L-k)/(k\,\Delta) \rfloor$. Thus, all the parameters affecting the matheuristic are involved in the tuning process.
For the configuration process we divided the Grid, Dense, and Carbin test sets into 9, 18 and

4 classes respectively, according to the instances' number of nodes in the first two cases and on the number of arcs and congestion ratio in the third case. The basic version of *irace* has been called separately for each class with a budget of 500 runs for each tuning process and a CPU time limit of 500 seconds for each run. The obtained optimal configurations of the parameters are presented in Table 6.4 for each Dense class, d-class-x, Grid class, g-class-x and Carbin class, B-class-x. From the tuning results we can evince that the *irace* applications span the configuration combinations without exhibiting evident tendencies apart from the case of the Δ parameter assuming value 0.5 that appears in only one optimal setting out of the 31 total classes. Also, the combination of $(L, \Delta, r\text{-}max) = (100, 1, 500)$ in the Dense classes is preferred among the other combinations in 28% of the cases. Note that the unique class of instances with optimal setting $\Delta = 0.5$ is the one that collects the smallest and simplest networks among all the considered test sets. This slow enlargement of the neighborhoods is combined with a lower randomization parameter likely to rebalance the exploration routine implementing faster changes of the neighborhood structures. In the next experiments the matheuristic has been tuned accordingly to the obtained optimal settings.

Table 6.4: Irace tuning results for the L, Δ, $r\text{-}max$ parameters

Instance class	L	Δ	*r-max*	Instance class	L	Δ	*r-max*
d-class-1	100	0.5	500	g-class-1	100	1	500
d-class-2	100	1	1500	g-class-2	50	1.5	500
d-class-3	100	1	500	g-class-3	50	1	500
d-class-4	100	1	500	g-class-4	50	1.5	500
d-class-5	100	1	1500	g-class-5	50	1.5	1000
d-class-6	100	1	500	g-class-6	100	1	1000
d-class-7	100	1	500	g-class-7	100	1.5	1000
d-class-8	150	1	500	g-class-8	50	1	1500
d-class-9	50	1	1500	g-class-9	50	1.5	1000
d-class-10	50	1.5	1500	B-class-1	150	1.5	1500
d-class-11	150	1	500	B-class-2	100	1.5	1500
d-class-12	150	1.5	1500	B-class-3	50	1.5	1500
d-class-13	50	1.5	500	B-class-4	50	1	500
d-class-14	50	1.5	1500				
d-class-15	100	1	1500				
d-class-16	100	1.5	1000				
d-class-17	100	1	500				
d-class-18	50	1	1500				

The Grid test set Appendix B collects the complete results of this experiment where the free-flow relaxation is employed to provide a benchmark value to the matheuristic's solutions. Each row reports the instance features in terms of number of nodes, arcs, commodities and flow split parameter in the "nodes", "arcs", "h" and "k" columns, respectively. Then it presents the makespan value and the computational time in seconds for the initial and the best solutions provided by our matheuristic, see the "init sol", "t", "best sol" and "t"

column respectively. The number of encountered improvement in the makespan value is reported in the "moves" column. Recall that a makespan of 24 time instants represents a valid certificate of optimality being equal to the lower bound. Hence, we mark these provably optimal solutions with the symbol *. Table 6.5 collects some results aggregated by the k-splittable parameter value. Columns report the averages of the above presented key indicators computed on the whole Grid test set. The last column "GAP (%)" shows the average distance of our best solutions from the lower bound provided by the $QMCFP$.
The matheuristic provides, as expected, initial and best solutions with a makespan always greater than or equal to the free-flow relaxation. The results get closer to this lower bound when increasing the k parameter: in the unsplittable setting the optimal transshipment is performed within 24 time instants only in 13.34% of the instances with an average GAP of 20.65%. In the 2-splittable and 3-splittable cases the percentage of instances closing at the lower bound grows to 42.23% and 82.23% respectively, with a consequent significant reduction in the average GAP that reaches 5.10% and 1.67%, respectively. This simply results from the higher degree of freedom granted by the flow split parameter to route the flows. In 28.89% of the instances the matheuristic has been able to construct an initial solution with exactly the free-flow makespan, thus a provably optimal solution, without the need of any improvement step. This reveals the efficacy in these specific cases of the employed initialization procedure to perform the best path selection for each single commodity. In the remaining instances, either the improvement phase identified better solutions during the one-hour process, 87.50% of the cases, or it stayed sticked to the initial solution found with a zero value in the "moves" column. Note that in the latter case the algorithm might have potentially reached the optimal value but no guarantees can be ensured. Some instances, see for example the g-3-5-k and g-10-4-k, present a considerable total improvement of the initial solution with several intermediate incumbents. This suggests that there exists some cases where the best path choice for each independent commodity performs bad for the overall simultaneous multicommodity transshipment. From a computational time point of view we can deduce that an increase in the dimension of the networks or in the number of commodities within the same grid graph reflects in an increase of time to construct the initial solution. Instead, a change in the flow split value has no influence on it, see the second column in Table 6.5. This behavior can be motivated by the generation step of the candidates' lists performed for each single commodity in subgraphs of the original one and independently of the flow split parameter value. Except for a few cases, the best solution has been found by the matheuristic in the very early part of the search process, with an average time always smaller than 4 minutes in all the k-splittable cases as shown in Table 6.5. The general limited number of intermediate incumbents, in average around 2 moves, and some statistics on the total number of iterations show that the matheuristic is capable of performing a quick and efficient neighborhood search despite the growing dimension of the problems, this also thanks to the introduction of the upper bound to speed up the MIP problems. This preliminary analysis confirms the validity of the considered lower bound and the correctness of our algorithm both in its initialization and final outputs.

Table 6.5: Average results on the Grid test set aggregated by the flow split parameter

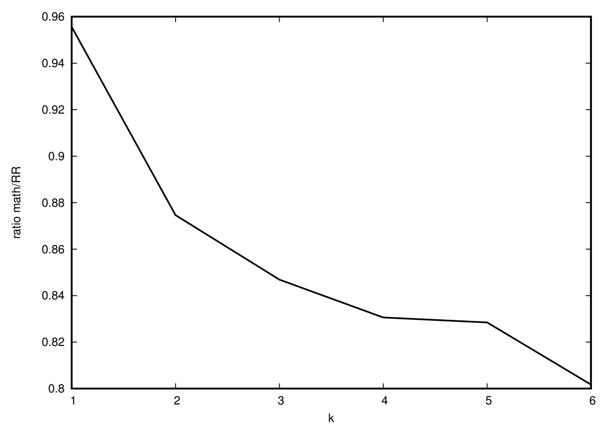
k	init sol	t (s)	best sol	t (s)	moves	GAP(%)
1	35.64	3.22	28.96	222.16	2.20	20.65
2	30.44	3.18	25.22	58.22	2.29	5.10
3	27.33	3.31	24.40	156.09	1.76	1.67
4	26.60	3.42	24.20	126.84	1.47	0.83
5	26.11	3.44	24.09	21.60	1.24	0.37
6	25.76	3.56	24.07	61.27	1.07	0.28

Table 6.6: Average results on the Dense test set aggregated by the flow split parameter

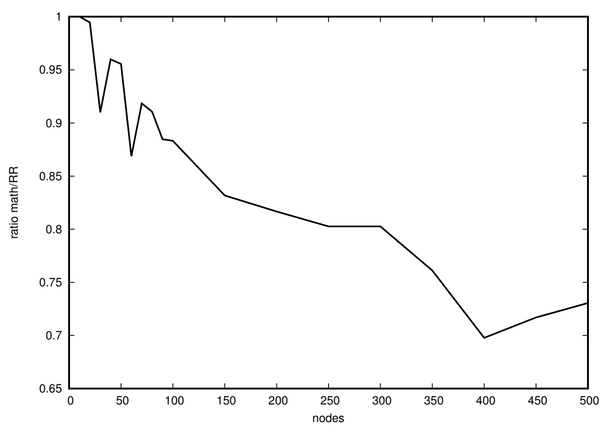
	Matheuristic			RR			Comparison	
k	t (s) init sol	t (s) best sol	moves	t (s) init sol	t (s) best sol	moves	ratio init sol	ratio best sol
1	3.28	61.94	0.33	263.83	1020.50	3.17	0.8208	0.9556
2	3.33	19.06	1.22	96.50	1283.78	4.89	0.7363	0.8746
3	3.72	74.94	2.33	95.00	1651.61	5.83	0.7142	0.8468
4	3.61	232.61	2.72	94.50	1450.00	4.00	0.9038	0.8306
5	3.94	58.94	2.94	94.56	1511.72	4.11	0.9700	0.8285
6	3.94	65.61	3.28	94.28	786.22	2.44	1.0611	0.8017

Table 6.7: Average results on the Carbin test set aggregated by the flow split parameter

	Matheuristic			RR			Comparison	
k	t (s) init sol	t (s) best sol	moves	t (s) init sol	t (s) best sol	moves	ratio init sol	ratio best sol
1	2.13	588.5	3.38	12.13	784.25	4.63	1.0827	0.9255
2	2.13	75.00	2.00	12.13	171.63	2.88	0.9997	0.9717
3	3.13	515.38	1.38	11.88	167.88	2.38	0.9938	0.9725
4	4.75	368.13	1.13	12.13	419.63	2.00	1.0284	0.9950
5	5.13	346.38	0.75	12.13	74.25	2.13	1.0171	0.9950
6	5.63	101.00	0.75	12.25	158.63	0.88	1.0631	1.0033



(a) Average ratio between the best solution found by the matheuristic and by the RR on the Dense test set depending on the k parameter.



(b) Average ratio between the best solution found by the matheuristic and by the RR on the Dense test set depending on the number of nodes.

Fig. 6.1: Comparison of the performances of the matheuristic and the RR algorithm depending on the k parameter and on the number of nodes.

The Dense test set In this experiment we compare our algorithm against the dynamic version of the RR heuristic, see Subsection 6.2.3, on the Dense test set. As for our algorithm, we gave the RR a time limit of one hour for the improvement phase, a time horizon of 72 instants for rerouting the flows through the selected paths and we fed its formulations with the current best upper bound to the makespan. The complete results obtained with the two strategies are collected in tables and reported in Appendix C. The first four columns present features of the instance: nodes, arcs, the number h of commodities and the k parameter. The next five columns refer to our matheuristic, while the remaining to the RR heuristic. For each resolution technique and each instance we report the same values as in the Grid tables: the initial solution makespan and its construction time, the best solution makespan and its computational time and the number of intermediate incumbents. In Table 6.6 we report for both strategies some average results aggregated by the flow split parameter: the average time to construct the initial solution, "t(s) init sol" column, the one to identify the final best solution, "t(s) best sol" column, and the average number of intermediate solutions, "moves" column. Further indicators are presented in the last two columns "ratio init sol" and "ratio best sol": each entry of the former represents the ratio between the initial solution makespan provided by our matheuristic and the one provided by the RR averaged among all the complete Dense test set. Similarly, entries of the latter express the same idea applied to the best final solutions identified by the algorithms.
From the results we can evince that our matheuristic considerably outperforms the RR w.r.t. several aspects. In particular, in 76.85% of the cases the initial solution constructed by our algorithm has a strictly better value than the one provided by the competing approach, it becomes the 85.18% if we include equality. Moreover, this initial advantage appears more significant on instances with a smaller value of the k flow split parameter. This prelimi-

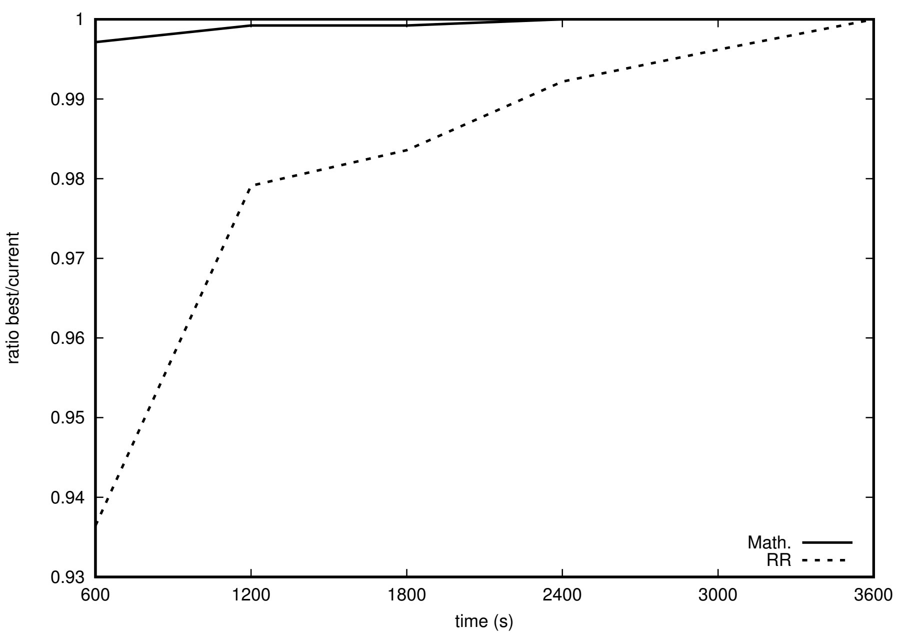


Fig. 6.2: Average ratio between the best solution and the solution found after a given time for the matheuristic and the RR method.

nary analysis shows the effectiveness of the initialization phase of our matheuristic: feeding the MIP problem with the top k ranked quickest paths for each commodity is better than relying on a random selection based on the multicommodity free-flow relaxation's choices. This can be further confirmed by looking at those 24.07% of the instances where no improvements were found by our matheuristic during the one-hour search, see for example almost all the unsplittable cases d-a-1 or instances d-90-2 and d-300-2. Here the constructed initial solutions might actually be optimal for the problems while the RR hasn't been able to provide an initial or even final solution with a strictly better makespan. We now focus on the computational times required by the strategies to construct their initial solutions. Recall that in both situations the initial construction time includes the generation of the lists of candidate paths and the resolution of the first MIP problem with the selected k paths as input for each commodity. Our matheuristic presents very fast initial times that slightly increase when solving bigger instances but never exceed 15 seconds among all the testbed. On the other side, the initial procedure of the RR heuristic requires longer computational times in 89.81% of the instances and is much more sensible to the increase in the dimension of the considered network. In particular, the resolution of the free-flow relaxation reaches just itself a maximum of 6 minutes in the bigger network. This different computational effort is motivated by the additional need of the RR heuristic of associating to each candidate path a flow value in order to deduce probabilities for the random selection procedure. From Table 6.6 we observe how a variation in the k-splittable parameter does not substantially affect

the times for generating an initial solution except for the unsplittable case with the RR strategy, see in particular instances d-70-1, d-250-1, d-450-1 and d-500-1. In these cases, the competing approach presents problems in finding a feasible unsplittable flow within 72 time instants requiring several reiteration of the selection and resolution process. Such repeated infeasibilities suggest that the random selection of the paths is an inadvisable strategy in the unsplittable case. Some experiments were further conducted providing the RR a longer time horizon to reduce/avoid this initial infeasibility. This resulted in a faster initialization process but no improvements w.r.t. the final best solution were achieved during the one-hour process.
The improvement phase of our matheuristic reveals its potential, too: despite the few cases of initial disadvantage, the matheuristic always ends up with an equal or better final solution (strictly in 80.56% of the cases). Thus, we can deduce how the lists of paths collects a set of adequate and high-quality candidates that can be efficiently adopted to improve the solutions. Moreover, the sequential enlargement of the neighborhood, controlled by s and Δ, and its construction rule allow for a better and fast exploration of the large feasible region. In terms of computational time efficiency, our method needed substantially less time to identify the final best value in almost all instances, particularly in the 2-splittable case as shown by the aggregated results in Table 6.6. Within 32 minutes the improvement phase of the matheuristic has already found all the final best solutions (95.37% even within 10 minutes), while in 32.41% of the instances the RR is still conducting a fruitful randomization search.
The graphics in Fig. 6.1 represent the behavior of the algorithms depending on the k parameter and on the size of the digraph in terms of number of nodes, case 6.1a and 6.1b respectively. We report on the y-axis the ratio between the final best solution provided by our matheuristic and the one by the RR, averaged among all considered Dense instances. Note that case 6.1a graphically represents the values of the last column of Table 6.6. We can observe a decreasing trend as the parameters increase, with more regularity in the left case, with an average ratio almost always under value 1.0. This means that the overall box implemented by our matheuristic, both construction technique and improvement rule, is substantially less affected by the increase in the number k of allowed paths and in the dimension of the network w.r.t. the competing approach scheme to provide good final solutions to the problem. This confirms the scalability and robustness of our proposed approach. Figure 6.2 analyses how strategies achieved neighbor improvements over the one-hour time limit. Here the y-axis shows the ratio between the final best solution and the current best solution at a given time, averaged among all considered instances (we report values starting from 10 minutes; note that only after 1600 seconds the RR heuristic founds an initial solution to all the Dense instances). We can observe that the matheuristic presents a faster overall convergence to its best solution, with an initial ratio already higher than 0.99. These results are strictly related to the neighborhood construction rules employed by the methods: relying on a ranked list of quickest paths instead of a complete randomization selection results a more efficient policy that allows to get good solutions in shorter times.

The Carbin test set Instances in this test set present a number of commodities which is high if compared to the network size. Such a final experiment is therefore carried out to stress test the algorithms on instances where a relevant number of commodities must be si-

multaneously routed in the dynamic network. Results of our algorithm and of the competing RR approach are collected in tables with the same layout as for the Dense case, see Table 6.7 for the aggregated results and Appendix D for the complete results.
In 50% of the cases our algorithm finds a strictly better initial solution with an additional 22.92% where both initialization procedures identify a solution with the same makespan value. The remaining cases of initial advantage of the RR, see values greater than 1.0 in the "ratio init sol" column of Table 6.7, mainly happen in the most congested networks, see instances Bs05 and Bs07 in Appendix D. Despite this, our matheuristic ends up with a better or equal final solution in 91.66% of the instances, revealing thus the capabilities of its improvement phase. In only four instances the RR prevails, see network Bl01, identifying a makespan equal to 38 while our matheuristic is stuck at solutions with objective value 39. A further insight is given by the "ratio best sol" column in Table 6.7: in the 6-splittable group of instances the RR heuristic outperforms on the average the matheuristic. In all the other cases our algorithm dominates. In terms of computational times the initialization and exploration procedures adopted in the matheuristic are faster than those of the RR in 93.75% and 83.33% of the cases, respectively. Moreover, according to the aggregated results, our matheuristic is faster on the average in constructing the initial solution for all k-splittable cases. The best average time to compute the final solution occurs when the k-splittable parameter equals two.
Overall, the experiment confirms a better performance of the matheuristic, albeit its advantage on the competing RR approach turns out to be somewhat reduced on this test set when compared with the results on the Dense instances. Such behavior could be motivated by a faster growth of the matheuristic MIP problems size when increasing the number of commodities in combination with s, Δ and k.

7. A Branch and Price approach for the Quickest Multicommodity k-splittable Flow Problem

7.1. Introduction

In the previous chapters, we introduced the Quickest Multicommodity k-Splittable Flow Problem and provided a quick and powerful algorithm for solving medium to large-sized instances of such novel optimization problem. Motivated by the proved strong NP-hard complexity of the problem, a matheuristic approach was designed with the aim of identifying good quality solutions in short computational times. The resulted $VLNS$-based matheuristic was validated and compared with different suitable strategies in a thorough computational experience, but the lack of available benchmarks to assess optimality and quality of the matheuristic results prevented a complete performance analysis on its capabilities. Indeed, a preliminary batch of optimality check experiments was achieved by solving the path-based formulation of the problem by linear programming on a restricted set of very reduced-size instances, where an enumeration of all the available paths was still possible. A further insight into optimality analysis was provided by the free-flow relaxation employed as a valid lower bound to the problem. Indeed, the availability of such bound led to a rough estimation of the optimality GAP and to a proof of optimality if the free-flow makespan was exactly met by the best matheuristic's solution. This experiment was conducted on small to medium-sized instances to ensure time-efficiency in identifying an optimal unrestricted multicommodity flow (recall the free-flow relaxation of the problem is still NP-hard). Finally, in larger networks and with a higher level of congestion, both of the previous benchmark methods could not be applied and a comparison with a state-of-the-art heuristic adapted to fit the dynamic environment was conducted, with a consequent loss of information related to optimality of the solutions in favor of a more general quality assessment.
It is now clear that the design of exact algorithms could fill this gap, permitting a dependable evaluation of the matheuristic performances and in general of any tailored heuristic method for solving the $QMCkSFP$. Limitations of exact approaches in tackling increasing-sized instances are well-known in the scientific literature; however, the potential availability of such exact tools represents an added value and sometimes a strict requirement in many real-life situations where a precise quality measure is significant for an actual and more conscious implementation of the identified solutions. Moreover, in particular contexts such as strategic

or tactical planning of emergency operations, the need for optimal solutions might provide a valid motivation for adopting exact solution methods allowing very large computational times.
To provide a complete answer to such relevant questions, the research activity focused on the design, development, implementation and test of the first exact algorithm for the $QMCkFP$. The proposed resolution approach exploits the path-based formulation of the problem and is based on the Branch and Price paradigm which, integrating Column Generation and Branch and Bound in an unique framework, is employed to cope with the exponential number of path-related mixed integer variables. In particular, the algorithm implicitly and efficiently explores the feasible region by iteratively solving restricted and relaxed versions of the original problem and provides as an output either an optimal solution or a certificate of infeasibility. We refer the reader to the following works for details on the Branch and Price strategy [10, 113, 82, 41].
In this Chapter we present the developed Branch and Price algorithm detailing all of its procedures in dedicated sections and presenting the thorough computational analysis conducted to test its performance. Finally, the strategy has been employed as a tool to validate and assess the quality of the matheuristic solutions presented in the previous chapter.
The Branch and Price steps are introduced starting from the construction procedure of the so-called Restricted Relaxed Master Problem ($RRMP$) in Subsection 7.2.1: the method performs a linearization of the time-arrival variables and excludes by substitution those path design variables in charge of ensuring k-splittability restrictions. In Subsection 7.2.2, we present the state-of-the-art method employed by the Branch and Price algorithm for the initialization of the pool of variables. Based on the ranking procedure described in Subsection 3.4.2, it generates for each commodity the top-ranked k quickest paths w.r.t. their transmission time. The same technique was integrated in the matheuristic approach for the generation of a set of high quality candidate paths, see Section 6.2.1. The dynamic nature of the considered optimization problem translates in the need of employing a time-expansion procedure within the Column Generation phase, when new promising variables are identified through a pricing routine. This aspect is thoroughly discussed in Subsection 7.2.3. The next Subsection 7.2.4 is devoted to present the branching rules designed for restoring feasibility. A first original strategy is introduced whenever an optimal solution to the $RRMP$ violates k-splittable flow constraints: it forces the usage of some paths while forbidding others over the discretized time horizon. Since such k-splittable branching rule works on path-based variables, the generated branching cuts are included in $RRMP$s of the child nodes. The second rule implements a refined version of the traditional binary branching on fractional variables by selecting the candidate time-arrival variable at the highest time instant. The pricing oracle is modeled as a Shortest Path Problem with Forbidden Paths ($SPPFP$) to ensure that no variables related to local forbidden paths are identified as new entering candidates. The pricing problem is solved through a tailored labeling algorithm based on dynamic programming in an extended TEN of the original digraph, see Subsection 7.2.5. Additional limited resources are included in the labels of the algorithm to avoid the generation of non-elementary paths. The last section is dedicated to computational experiments conducted on the developed Branch and Price algorithm: in Subsection 7.3.1 we focus on small to medium-sized instances to prove the correctness of the strategy and provide measures of quality of the solutions identified, whereas in Subsection 7.3.2 we make use of our exact algorithm to

assess quality of the matheuristic results providing optimal solutions or better certified lower bounds on a testbed previously utilized in Section 6.3.2.

7.2. An exact algorithmic approach

The Branch and Price strategy, integrating Column Generation and Branch and Bound in an unique framework, results particularly recommended for tackling $MILP$ problems that present a huge number of variables [10, 113, 82, 41]. Indeed, it allows to work on restricted and relaxed versions of the original problem where only certified good quality variables are considered and to implicitly explore the space of feasible solutions ending up with provably optimal solutions or a proof of infeasibility. More in detail, it applies the Branch and Bound strategy to the original problem restricted to a proper subset of columns. At every node of the decisional tree, a relaxed version of this problem, namely the Restricted Relaxed Master Problem ($RRMP$), is solved to optimality by means of Column Generation. To this aim, the so-called pricing oracle iteratively checks whether the optimal solution to the current $RRMP$ could be improved by introducing new columns. If no such column exists, then all original variables are implicitly treated in the current relaxation. At this point, the feasibility of the solution for the original problem is checked, eventually leading to the application of the branching phase whenever at least one of the original constraints is not satisfied.
Note that Branch and Price perfectly suites the $QMCkSFP$, as fractional x and binary z variables in the provided formulation are related to the number of paths for each commodity, which grows exponentially with the size of the digraph, and to the considered discretized set of time instants, see Section 5.2. Therefore, an explicit enumeration of all these variables, and thus of all paths available for flow transshipment, would result prohibitive for real-sized instances of the problem.
The design of an efficient Branch and Price algorithm requires a high level of customization, based on the specific features of the optimization problem to be solved. The following subsections are devoted to provide full details for each procedure of the original Branch and Price approach tailored to solve the $QMCkSFP$.

7.2.1. The Relaxed Restricted Master Problem

The Branch and Price algorithm works on restricted versions of the path-based formulation of the $QMCkSFP$, obtained by considering for each commodity only a proper subset of the available paths, i.e. $\mathcal{P}_h^{'} \subset \mathcal{P}_h, \ \forall h \in \mathcal{H}$, and iteratively solves a relaxation of these restricted problems at each node of the Branch and Bound tree by a Column Generation procedure.
The employed relaxation method first linearizes all integer variables and then replaces bottleneck constraints (5.7) with the following feasible constraints, implementing an aggregation over time and a relaxation of the right hand side:

$$\sum_{t \in \mathcal{T}} x_{pt}^h \leq \sigma_h z_p^h \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h^{'} \tag{7.1}$$

The resulted problem is known as the Relaxed Restricted Master Problem ($RRMP$).
We now prove that the following Lemma holds.

Lemma 7.2.1. *There exists an optimal solution to the $RRMP$ saturating Constraints (7.1).*

Proof. Let $(\hat{\zeta}, \hat{x}^h_{pt}, \hat{y}^h_t, \hat{z}^h_p) \in (\mathbb{R}_+, \mathbb{R}_+^{|\mathcal{H}||\mathcal{P}'_h|T}, \mathbb{B}^{|\mathcal{H}|T}, \mathbb{B}^{|\mathcal{H}||\mathcal{P}'_h|})$ be an optimal solution to the $RRMP$, and let

$$0 \leq \hat{\hat{z}}^h_p := \frac{\sum_{t \in \mathcal{T}} \hat{x}^h_{pt}}{\sigma_h} \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}'_h. \tag{7.2}$$

We must prove that the solution $(\hat{\zeta}, \hat{x}^h_{pt}, \hat{y}^h_t, \hat{\hat{z}}^h_p)$ is optimal for the $RRMP$, too. Note that its feasibility simply follows from constraints (7.1) i.e.:

$$\sum_{p \in \mathcal{P}'_h} \hat{\hat{z}}^h_p = \sum_{p \in \mathcal{P}'_h} \left(\frac{\sum_{t \in \mathcal{T}} \hat{x}^h_{pt}}{\sigma_h} \right) \leq \sum_{p \in \mathcal{P}'_h} \frac{\sigma_h \hat{z}^h_p}{\sigma_h} = \sum_{p \in \mathcal{P}'_h} \hat{z}^h_p \leq k \quad \forall h \in \mathcal{H}.$$

Moreover, domain constraints $\hat{\hat{z}}^h_p \leq 1 \ \forall h \in \mathcal{H}, p \in \mathcal{P}'_h$ are implied by population constraints (5.4) as $\sum_{t \in \mathcal{T}} \hat{x}^h_{pt} \leq \sigma_h \ \forall p \in \mathcal{P}'_h$. Finally, the optimality follows from the observation that both solutions have the same objective function value. $\square$

Lemma (7.2.1) allows for a substantial reduction of the number of variables and thus a consequently simplification of the overall Branch and Price approach. Indeed, the z's decision variables can be eliminated from the $RRMP$ by substituting each of their occurrence with their definition provided in (7.2). Moreover, in this new variable space, the k-splittable constraints are always satisfied and can be therefore excluded from the formulation. See Truffot et al. [111] for a similar approach adopted in a Branch and Price algorithm for the static Maximum k-splittable Flow Problem.
The final formulation of the $RRMP$ results in:

$$\min \zeta \tag{7.3}$$

$$t y^h_t \leq \zeta \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{7.4}$$

$$\sum_{p \in \mathcal{P}'_h} x^h_{p(t-l_p)} \leq C_{ht} y^h_t \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \quad (\pi^1_{ht} \leq 0) \tag{7.5}$$

$$\sum_{p \in \mathcal{P}'_h} \sum_{t \in \mathcal{T}} x^h_{pt} = \sigma_h \qquad \forall h \in \mathcal{H}. \quad (\pi^2_h) \tag{7.6}$$

$$\sum_{h \in \mathcal{H}} \sum_{p \in \mathcal{P}'_h} \delta^p_{ij} x^h_{p(t-t^p_i)} \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}. \quad (\pi^3_{ijt} \leq 0) \tag{7.7}$$

$$y^h_t \in [0,1] \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{7.8}$$

$$x^h_{pt} \geq 0 \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}'_h, t \in \mathcal{T}. \tag{7.9}$$

$$\zeta \geq 0 \tag{7.10}$$

Note that the $RRMP$ corresponds to the linear relaxation of the free-flow version of the problem restricted to only the $\mathcal{P}_h^{'}$ paths.

7.2.2. Initial columns

At the beginning of our algorithm we generate the initial subsets of paths to be included in the $RRMP$ at the root node. In particular, for each commodity h we identify the collection of o_h-d_h paths that occupy the first k positions when listed in an increasing order of their transmission time. This means that for each commodity the k top-ranked quickest paths are selected. The construction of such paths is performed by adopting the lazy version of the Chen's approach presented in Pascoal et al. for ranking quickest loopless paths [96, 97] together with the Yen's algorithm [116], see Subsection 3.4.2. Recall that the method keeps in memory an ordered list of shortest loopless paths, each of them obtained in a specific subgraph of the original graph. At every iteration the best shortest path in the array w.r.t. the transmission time is selected and replaced with the next shortest loopless path found in the related subgraph. The columns of the initial $RRMP$ are then accordingly populated by including, in addition to all path-independent variables, those fractional variables x_{pt}^h associated with each identified quickest path p and repeated for each discrete time step $t \in \mathcal{T}$.

7.2.3. The Pricing Problem

The pricing problem is in charge of identifying additional fractional x variables that, when added to the current $RRMP$, would lead to an improvement in the objective function value. This is done separately for each commodity by looking for the pair (path, departure time) that presents the minimum reduced costs in the optimal solution of the $RRMP$ at the current node. Formally, the following problem is solved:

$$\hat{c}_{\hat{p}\hat{t}}^h := \bar{\pi}_h^2 + \min_{p,t} \left(\Delta_{(t+l_p \leq T)} \bar{\pi}^1_{h(t+l_p)} + \Delta_{(t+t_i^p \leq T)} \sum_{(i,j)\in\mathcal{A}} \delta_{ij}^p \bar{\pi}^3_{ij(t+t_i^p)}\right) \qquad \forall h \in \mathcal{H},$$

where the minimization occurs over all o_h-d_h paths for the commodity h and departure instants within the considered time horizon, Δ_{exp} is a binary indicator taking value 1 if exp is true, and $\bar{\pi}^i,\ i = 1, 2, 3$ are the opposite values of the current optimal dual variables associated to constraints (7.5), (7.6) and (7.7), respectively. Therefore, if $\hat{c}_{\hat{p}\hat{t}}^h < 0$ the current solution is not optimal for the $RRMP$ and the columns related to all variables $x_{\hat{p}t}^h\ \ \forall t \in \mathcal{T}$ are included to the $RRMP$. On the contrary, if $\hat{c}_{\hat{p}\hat{t}}^h \geq 0$ no new columns need to be added: in this case, feasibility of the current solution for the initial original problem has to be checked and, when this is not met, the branching phase has to be performed.
We model the pricing problem on a modified version of the original TEN (see Subsection 3.4.1) where an additional super source node o' and super sink node d' are introduced and then connected through outgoing arcs directed to each time-replica of the origin, for seek of simplicity indicated as $(o', o_h)_t$, and through incoming arcs originating from each time-replica of the destination of the commodity, i.e. $(d_h, d')_t$, respectively. Capacities of these

artificial arcs are unrestricted and their lengths are set to zero. The set of holdover arcs in the so-generated TEN, already restricted to only those deriving from time-replica of the source node, is not required anymore, as a postponed departure is now modeled through the activation of an arc between the super source and the origin of the commodity at the desired departure time.
The pricing problem in the extended TEN graph, indicated as $\mathcal{D}'_{\mathcal{T}} = (\mathcal{V}'_{\mathcal{T}}, \mathcal{A}'_{\mathcal{T}})$, results thus in a o'-d' Shortest Path Problem where the length associated to arc $(i,j)_t \in \mathcal{A}'_{\mathcal{T}}$ equals:

$$b_{ijt} = (\gamma^j_{d_h} \bar{\pi}^1_{h(t+\lambda_{ij})} + \bar{\pi}^3_{ijt})$$

where binary indicator $\gamma^j_{d_h}$ is one if $j = d_h$, zero otherwise and $b_{ijt} = 0 \ \forall (i,j)_t \in \mathcal{A}'_{\mathcal{T}} \setminus \mathcal{A}_{\mathcal{T}}$. Note that arc lengths have positive values and no directed cycles can be detected due to the specific structure of the TEN graph.
The explicit model of the pricing problem follows, being binary variables $\omega_{ijt} = 1$ if the commodity traverses arc $(i,j)_t$, zero otherwise.

$$\min \sum_{(i,j)_t \in \mathcal{A}'_{\mathcal{T}}} b_{ijt}\omega_{ijt} \tag{7.11}$$

$$\sum_{t\in\mathcal{T}} \omega_{o'o_h t} = 1 \tag{7.12}$$

$$\sum_{t\in\mathcal{T}} \omega_{d_h d' t} = 1 \tag{7.13}$$

$$\sum_{(j,i)_{\bar{t}} \in \mathcal{A}'_{\mathcal{T}} \,:\, t=\bar{t}+\lambda_{ji}} \omega_{ji\bar{t}} - \sum_{(i,j)_t \in \mathcal{A}'_{\mathcal{T}}} \omega_{ijt} = 0 \qquad \forall\, i_t \in \mathcal{V}_T. \tag{7.14}$$

$$\omega_{ijt} \in \{0,1\} \qquad \forall\, (i,j)_t \in \mathcal{A}'_{\mathcal{T}}. \tag{7.15}$$

Constraints (7.12) and (7.13) impose the origin of the path at node o' and the destination at node d', respectively, while Constraints (7.14) ensure the identification of a non-interrupted path. These are applied to all time-replica of all original nodes, including the origin and destination nodes for the commodity. The optimal solution to this problem identifies a pair $(\hat{p}, \hat{t})$ where $\hat{p}$ is defined by the sequence of selected arcs and the departure time $\hat{t}$ by the time instant at which the origin node of the commodity is traversed. In the case of an infeasible $RRMP$, Farkas pricing is applied to identify candidate pairs for path and departure time that can reduce the infeasibility, see [41]. If at least one such pair exists, its related path-flow variables for each feasible time instant are added to the $RRMP$. Otherwise, the current node is discarded. Note that in the case of this happening at the root node, i.e. before any branching step is performed, the algorithm stops, having proved the infeasibility of the original dynamic problem.
Figure 7.1 depicts a possible solution to the pricing problem for a given commodity. More in detail, the red o-d path used at time one in the original dynamic network on the left results in the red o'-d' path in the modified TEN on the right.
As it will be explained in detail in the next subsections, the presented linear programming

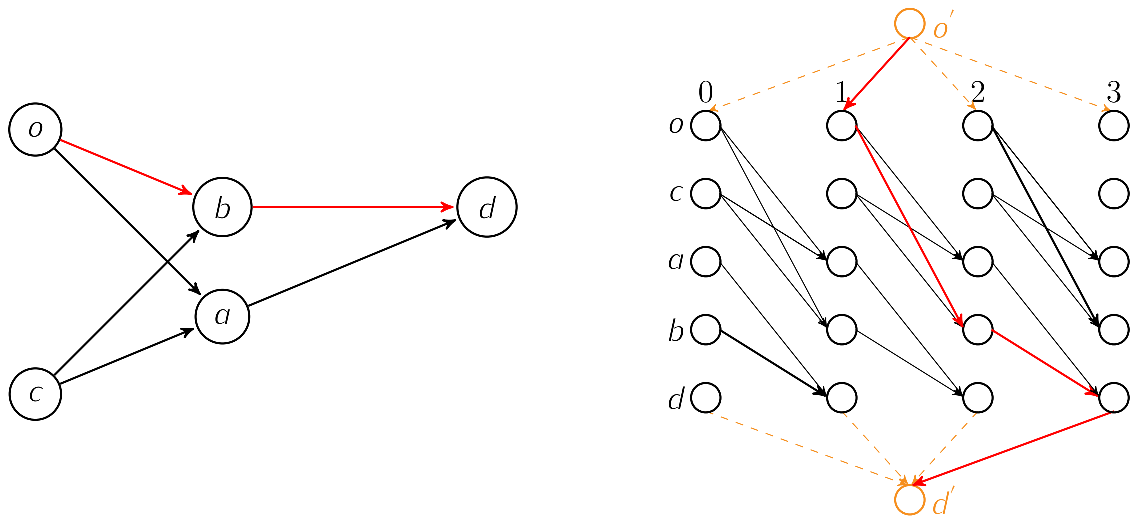


Fig. 7.1: Modellization of the pricing problem in an extended version of the TEN.

formulation itself cannot be straightly adopted as a tool for solving the pricing problem, as additional constraints deriving from the application of an original branching strategy need to be embedded throughout the resolution process. Instead, a dynamic programming approach is designed to solve an ad-hoc variant of the SPP that implicitly accounts for the above mentioned branching decisions.

7.2.4. Branching Strategy

The branching step occurs whenever the optimal solution to the current $RRMP$ violates the original problem's constraints. Recall that the $RRMP$ doesn't ensure y's integrality and the satisfaction of the k-splittable flow constraints as binary z's have been discarded during the relaxation phase.
In the following we first describe the branching rule developed for restoring feasibility w.r.t. limitations on the number of paths and then we present the refined strategy related to the binary branching on time-arrival variables.

The k-splittable branching We discuss here the case of having at least a commodity sending positive amount of flow on $k+\alpha$ paths with $\alpha \geq 1$ in the optimal $RRMP$ solution of a given node. The branching strategy applied to one of these commodity h identifies $k+1$ quickest paths among the used ones, i.e. the $k+1$ activated paths with minimum transmission times, and collects them in the ordered set $\mathcal{P}''_h$. Then, it generates $k+1$ child nodes N_j, $j = 1, \dots, k+1$ in the decisional tree as follows: in a given children N_j it imposes a branching constraint that forbids the usage over time of the j-th path and a second one that forces to route flow through paths $p_i \in \mathcal{P}''_h$, $\forall i < j$ over the time horizon. This results in:

$$\sum_{t \in \mathcal{T}} x^h_{p_j t} = 0 \qquad \sum_{t \in \mathcal{T}} x^h_{p_i t} \geq \epsilon, \;\; \epsilon > 0, \;\; \forall i < j.$$

The integration of such constraints in the $RRMP$ of the child nodes generates additional dual variables that must be accounted for computing reduced costs. However, note that dual variables related to forced paths can be excluded from this computation still ensuring a correct implementation of the column generation procedure. The same doesn't apply to local forbidden paths as the SPP formulation provided in Section 7.2.3 might not be able to avoid their repeated generation. Subsection 7.2.5 is devoted to present the technique designed to deal with this issue. In particular, we decided to reformulate the pricing problem as a Shortest Path Problem with Forbidden Paths and additional node-set resources to implicitly account for the forbidden path-related branching decisions.

An improved binary branching The current $RRMP$ optimal solution might present binary time-arrival variables with fractional values, i.e. $\exists h \in \mathcal{H}, t \in \mathcal{T}$ s.t. $y_t^h \in]0, 1[$. In this case, we apply a 0-1 branching to the y_t^h variable presenting the highest time instant among all candidates. However, note that there exist cases where the integrality of y's can be manually adjusted without requiring a branching step. As instance, if no commodity flow arrives at time instant t in the optimal solution $(\hat{\zeta}, \hat{x}_{pt}^h, \hat{y}_t^h)$ of the $RRMP$, i.e. $\sum_{p \in \mathcal{P}'_h} \hat{x}_{p(t-l_p)}^h = 0$, we simply set $\hat{y}_t^h = 0$. Moreover, if $\hat{y}_t^h t < t \leq \lceil \hat{\zeta} \rceil$, i.e. rounding up the time-arrival variable would generate an equivalent optimal solution, we simply set $\hat{y}_t^h = 1$.

7.2.5. A novel Dynamic Programming algorithm for the Pricing Problem

As detailed in the previous section, the developed k-splittable branching strategy imposes a reformulation of the pricing oracle. Indeed, the SPP structure presented in Section 7.2.3 is no longer valid due to the need of accounting for local forbidden paths during column generation. To this aim, we re-model the pricing problem as an o'-d' Shortest Path Problem with Forbidden Paths ($SPPFP$). The latter is a particular variant of the well-known Resource-Constrained SPP that asks for the identification of a minimum length path that is not part of a given set of forbidden paths, see [114, 99].
We solve this novel pricing problem via a dedicated label-setting algorithm that utilizes for each forbidden path a limited resource whose consumption is checked and updated during the label extension procedure. The finite availability of resources ensures that the optimal solution, if it exists, doesn't encompass flows traversing completely a forbidden path at any time. The optimal (path, departure time) pair can be then retrieved by the label at the super sink minimizing the total cost of the traversed arcs. We refer the reader to the work by Di Puglia Pugliese and Guerriero [30] for the resolution technique designed for the static flows environment.
Note that a computation of the previously described algorithm might generate a path with loops over time, i.e. a path traversing the same node at multiple instants over the time horizon. To avoid this identification, we introduce for each node in the original dynamic digraph a limited resource, namely node-set resource to prevent the flow from traversing the same node multiple times. We now provide formal details of the developed strategy; a pseudocode of the algorithm is provided in Algorithm 3.
Consider the set $\mathcal{F}$ of forbidden o_h-d_h paths for commodity h that are locally imposed at a certain node in the decisional tree. To each path $p_k \in \mathcal{F}$ we associate a finite amount of

path-related resource equal to $P^k = n(p_k) - 2$, where $n(p_k)$ is the number of nodes in the path. All path-related resources are collected in the vector $P = \left[P^1, P^2, \dots, P^{|\mathcal{F}|}\right]$. Given a subpath q from the super source o' to a given node i_t in the TEN graph, we denote by $U_{i_t}(q) = \left[U^1_{i_t}(q), \dots, U^{|\mathcal{F}|}_{i_t}(q)\right]$ the vector of resources consumed along q, being $U^k_{i_t}(q)$ the number of consecutive arcs of the k-th forbidden path p_k that are traversed by subpath q starting from the initial arc at a certain instant in the time horizon. Let the *multiplicity over time* of a node $j \in \mathcal{V}$ in a subpath q be the number of times the node is traversed along the path within the time horizon, i.e. $m_j(q) = \sum_{t \in \mathcal{T}} \delta_{j_t}(q)$, where $\delta_{j_t}(q)$ is one if node j_t is part of q, zero otherwise. For each node in the original digraph, say it j with position index $l \in \{1, \dots, n\}$ in $\mathcal{V}$, we introduce a *node-set resource* N^l with limit equal to one. Whenever the node is traversed at a given time by subpath q from o' to i_t, the resource consumption vector $V_{i_t}(q) = \left[V^1_{i_t}(q), \dots, V^{|\mathcal{V}|}_{i_t}(q)\right]$ is activated in the correspondent node-set resource, i.e. $V^l_{i_t}(q) = 1$. In this way we ensure a multiplicity over time which is at most one for each original node. The complete vector of the considered limited resources results in $R = \left[P, N\right] = \left[n(p_1) - 2, \dots, n(p_{|\mathcal{F}|}) - 2, 1, \dots, 1\right]$. To each subpath q from o' to node i_t we associate a state $s_{i_t}(q) = (b_{i_t}(q), W_{i_t}(q))$ where $b_{i_t}(q)$ is the total cost of the subpath expressed as the sum of its arcs' costs and $W_{i_t}(q) = \left[U_{i_t}(q), V_{i_t}(q)\right]$ its resources consumption. As more than one path might exist from o' to node i_t we collect all the associated states in the set $\mathcal{S}_{i_t}$, where the element $s_{i_t}(q_k)$ refers to the k-th path in the list.
We extend the definition of *dominance* provided in [30] to our case as follows: given two states associated to the same node i_t in the TEN, i.e. $s_{i_t}(q_1)$ and $s_{i_t}(q_2) \in \mathcal{S}_{i_t}$, we say that the first *dominates* the second iff $b_{i_t}(q_1) \leq b_{i_t}(q_2)$, $W_{i_t}(q_1) \leq W_{i_t}(q_2)$ and at least one of the inequalities is strict.
A state is said to be *feasible* if $W_{i_t}(q) \leq R$ while it is named *efficient* or *non dominated* if no other state associated to the same node dominates or equals it. Note that the definition of dominance is correctly preserved by the developed extension rule.
The proposed algorithm collects in the list L the unprocessed states that can be still extended to at least one new state and initialize it with state $s_{o'}(q^1) = (0, \bar{0})$, corresponding to the initial label at the super source node. Then, it iteratively selects the state $s_{i_t}(q) = (b_{i_t}(q), W_{i_t}(q)) \in L$ with lower cost and extends it generating efficient and feasible states for each adjacent node in the TEN of node i_t. Suppose node $j_{t'} \in \delta^+(i_t)$ is selected, with j being the l-th node in $\mathcal{V}$. If $V^l_{i_t}(q) = 1$, meaning that node j has been already visited by subpath q over time, the related extension is not performed and the adjacent node is skipped. Otherwise, the extension function generates a new state $\bar{s} = (\bar{b}, \bar{W})$ where $\bar{b} = b_{i_t}(q) + b_{ijt}$ and:

$$\bar{U}^k = \begin{cases} 1 & \text{if } (i,j) \text{ is the first arc of path } p_k, \\ U^k_{i_t}(q) + 1 & \text{if } (i,j) \text{ directly follows the last arc of subpath } q \text{ in path } p_k, \\ 0 & \text{otherwise} \end{cases}$$

$$\bar{V}^l = \begin{cases} 1 & \text{if } j \text{ is the l-th node in } \mathcal{V}, \\ V^l_{i_t}(q) & \text{otherwise} \end{cases}$$

for each $k = 1, \dots, |\mathcal{F}|$ and $l = 1, \dots, |\mathcal{V}|$. The novel state is added to the list $\mathcal{S}_{j_{t'}}$ iff it satisfies the path-related resource limits $\bar{U}_{j_{t'}}(q') \leq P$, where path $q' = q \cup (i_t, j_{t'})$. The dominance rule is then applied to delete non promising states at node $j_{t'}$, i.e. any $s_{j_{t'}}(q^k) \in \mathcal{S}_{j_{t'}}$ such that $\bar{s} \leq s_{j_{t'}}(q^k)$. Finally, state $s_{i_t}(q)$ is deleted from list L after checking all of its successors. The algorithm stops when the list L is empty and returns the least cost state in the set $\mathcal{S}_{d'}$ of the super sink node and the related reconstructed o'-d' path.

Algorithm 3 Dynamic programming-based approach

```
1: **Initialization**
   $s_{o'}(q_1) = (0, \bar{0}),\ \ L = \{s_{o'}(q_1)\},\ \ \mathcal{S}_{o'} = \{s_{o'}(q_1)\}$;
   $\mathcal{S}_{i_t} = \emptyset \ \ \forall i_t \in \mathcal{V}_{\mathcal{I}}' \setminus o'$;

2: **Selection of the state**
   **if** $L = \emptyset$:
     select the minimum cost path $q_k$ among all $s_{d'}(q_k) \in \mathcal{S}_{d'}$;
     $q^* \leftarrow q_k$;
     **return** $q^*$;
   **else:**
     select a state $s_{i_t}(q_k) \in L$ with minimum cost;

3: **State extension**
   **for all** $j_{t'} \in \delta^+(i_t)$, $j$ the $l$-th node in $\mathcal{V}$:
     **if** $V_{i_t}^l = 1$:
       **continue**;
     $\bar{s} \leftarrow ExtendState(s_{i_t}(q_k), j_{t'})$;
     **if** $\bar{s}$ is not dominated in $\mathcal{S}_{j_{t'}}$;
         $\mathcal{S}_{j_{t'}} \leftarrow append(\bar{s})$;
         $L \leftarrow append(\bar{s})$;
         update efficient states in $L$ and $\mathcal{S}_{j_{t'}}$;
   $L \leftarrow L \setminus s_{i_t}(q_k)$;
   **goto** 2;
```

7.3. Computational Experience

In this section a thorough computational experience performed on the designed Branch and Price algorithm to solve instances of the $QMCkSFP$ is presented and discussed.
In particular, in Subsection 7.3.1 we begin by studying the correctness of our exact algorithm and measuring its computational performances when applied on a collection of small to medium-sized instances extracted from the Carbin testbed (see [44] for the original testbed in the context of static k-splittable flows) and adapted here to fit the dynamic flow setting. Subsection 7.3.2 is devoted to the use of the Branch and Price approach as a tool to assess the quality of the solutions obtained by the designed $VLNS$-based matheuristic, see Chapter 6. In particular, we solve the Grid testbed previously used in Section 6.3.2 with our Branch

and Price algorithm, feeding it with results of the matheuristic as starting solutions, then measuring and comparing the residual optimality GAP.
Our Branch and Price approach has been implemented in the C++ language and the experiments conducted using the $SCIP$ Optimization Suite v5.0.0 [50] in its default setting on a 64bit Intel Xeon CPU at 2.80GHz with 64GB memory, running Ubuntu 14.04.2. In both of the experiments the k top-ranked quickest paths for each commodity have been provided as initial set of columns, implementing the ranking procedure described in Section 3.4. Following the results of a preliminary batch of tuning experiments, the binary branching on the time-arrival variables has been applied as a first branching strategy before the branching strategy restoring the k-splittable feasibility.

7.3.1. Performance analysis of the Branch and Price algorithm

In this experiment we test our exact algorithm on a collection of Carbin instances to analyze its computational performance by allowing three hours of computational time. The test set is composed of 8 instances with 32 nodes and 96 arcs (from 1 to 10 periods long), each of them tested with three different levels of flow split parameter $k = 1, 3, 5$ and three different levels of commodities $4, 8$ and 12 for a total of 72 instances. In this experiment we fed our exact algorithm with a valid lower bound (LB) on the optimal makespan of the instance provided by the free-flow relaxation of the problem, i.e. the Quickest Multicommodity Flow Problem ($QMCFP$), where no bound is imposed to the number of active paths. We refer to Subsection 6.2.3 for a description of the procedure used for the resolution of the $QMCFP$.
For each instance we also consider a tailored time horizon for flow transshipment obtained by computing a feasible solution through the initialization step of the $VLNS$-based matheuristic, see Subsection 6.2.1. Recall, the procedure generates a feasible initial solution to the $QMCkSFP$ by solving a restricted version of the mixed-integer path-flow formulation of the problem where only the best k quickest paths for each commodity are considered. Note that the obtained makespan is a valid upper bound (UB) to the instance optimum and it helps in containing the impact of the time dimension in the Branch and Price, especially in its pricing routine.
The complete results of this experiment are presented in Appendix E. In the first column the name of the instance is provided as Bl0x-y-z where y represents the number of commodities and z the k-splittable parameter. Columns “primal” and “dual” show the primal and dual solution values provided after three hours of running time of our exact algorithm. The optimality GAP is presented in column “GAP” while column “nodes” expresses the number of nodes explored during the computational process and column “t” the required computational time expressed in seconds. The next two columns show information related to bounds provided in input to the instances, i.e. the upper bound “UB” and the lower bound “LB” . The next values present performance indicators w.r.t. the execution of the LP solver, the pricing procedure and the branching rules, where “br. K” refers to the branching rule on the k-splittable constraints and “br. Y” to the modified fractional branching on the time-arrival variables. “Calls” entries show the number of times the procedure has been activated; time “t” entries the time spent (in seconds) in performing the procedure. Column “pr. paths” indicates the number of times a path with strictly negative reduced costs was identified by the pricing routine. Recall that for each of these paths all the time-replica of the related flow

Table 7.1: Comparison between the Default and the Incumbent settings on the Carbin test set

	Default	Incumbent
optimal	49	49
suboptimal	19	19
not identified	4	4
optimal (%)	68.06	68.06
suboptimal (%)	26.39	26.39
not identified (%)	5.55	5.55
GAP (%)	8.12	7.88
t (s)	3744.22	3610.10
nodes	273.82	285.67
UB	51.74	51.74
LB	44.25	44.25
UB opt	6	6
LB opt	40	40
UB opt (%)	12.24	12.24
LB opt (%)	81.63	81.63
LP calls	2186.64	1958.65
LP t (s)	1.17	1.10
pr. calls	317.72	307.58
pr. t (s)	3737.74	3605.43
pr. t amean (s)	327.47	415.09
pr. paths	17.94	17.14
pr. added vars	321.56	316.63
max labels	7747.33	7394.72
mean labels	2660.51	2502.08
br. Y calls	164.88	170.32
br. K calls	92.33	70.33
br. K branchings	48.99	45.66
leaves	72.75	69.92
nodes left	147.19	146.11
max depth	13.92	15.34
backtracks	53.18	50.25
backtracks (%)	9.67	10.20

Table 7.2: Average results on the Carbin test set aggregated by the flow split parameter

	k		
	1	3	5
optimal	6	20	23
suboptimal	14	4	1
not identified	4	0	0
optimal (%)	25.00	83.33	95.83
suboptimal(%)	58.33	16.67	4.17
not identified(%)	16.67	0.00	0.00
GAP (%)	23.86	2.67	0.46
t (s)	8352.23	2352.90	527.55
nodes	511.63	305.96	3.88
UB	60.08	48.67	46.46
LB	44.25	44.25	44.25
UB opt	6	0	0
LB opt	0	17	23
UB opt (%)	25.00	0.00	0.00
LB opt (%)	0.00	70.83	95.83
LP calls	3881.29	2612.92	65.71
LP t (s)	1.98	1.43	0.10
pr. calls	617.13	329.96	6.08
pr. t (s)	8342.47	2349.34	521.41
pr. t amean (s)	888.47	32.60	61.34
pr. paths	33.42	17.83	2.58
pr. added vars	692.46	235.33	36.88
max labels	17365.67	3320.13	2556.21
mean labels	6606.92	708.29	666.33
br. Y calls	334.29	158.29	2.04
br. K calls	170.17	105.71	1.13
br. K branchings	94.33	51.67	0.96

Table 7.3: Average results on the Carbin test set aggregated by the number of commodities

	h		
	4	8	12
optimal	18	17	14
suboptimal	6	5	8
not identified	0	2	2
optimal (%)	75.00	70.83	58.33
suboptimal (%)	25.00	20.83	33.33
not identified (%)	0.00	8.33	8.33
GAP (%)	9.61	4.31	10.31
t (s)	3054.13	3507.58	4670.96
nodes	353.54	290.46	177.46
UB	49.46	49.79	55.96
LB	42.38	42.88	47.50
UB opt	2	3	1
LB opt	13	14	13
UB opt (%)	8.33	12.50	4.17
LB opt (%)	54.17	58.33	54.17
LP calls	2612.08	2711.21	1236.63
LP t (s)	1.16	1.43	0.93
pr. calls	377.75	374.75	200.67
pr. t (s)	3052.02	3495.89	4665.32
pr. t amean (s)	17.34	187.67	777.39
pr. paths	16.63	15.67	21.54
pr. added vars	321.38	230.54	412.75
max labels	2858.00	6765.54	13618.46
mean labels	1118.21	2005.96	4857.38
br. Y calls	186.67	182.75	125.42
br. K calls	117.71	122.00	37.29
br. K branching	49.58	74.71	22.67

Table 7.4: Results on the Grid test set after the application of the Branch and Price algorithm

	After *BP*
improved GAP	44
improved GAP (%)	55%
improved LB	44
improved best sol	0
improved LB (%)	100%
improved best sol (%)	0%
optimal	29
suboptimal	15
optimal (%)	65.91%
suboptimal (%)	34.09%
GAP amean	4.59%
GAP improv amean	10.11%

variables were added to the *RRMP* as new columns. The total number of added variables is collected in the "pr. added vars" column. The next two columns collect information on the labels generated during the dynamic programming algorithm: the maximum number of labels generated by a single pricer call is presented in column "max labels", and the mean value among all the calls in column "mean labels". Due to the specific features of the *br.K* rule, we implemented additional steps to early detect potential infeasibility of the node before processing it. In particular, whenever more than k paths are forced or a certain path is simultaneously forbidden and forced at the current node we stop the branching procedure and cutoff the node having proved its infeasibility for our original problem. Moreover, whenever exactly k paths are forced, we perform a fast look ahead by introducing in the current *RRMP* an additional constraint that forbids the usage of any other non-forced path. If the problem results infeasible, we again cut off the node, otherwise we add the constraints as a valid local cut to the current node. In all these cases the generation of child nodes is avoided. To reflect this diversification of cases, we collect in the "br. K branching" column the number of times the call to the branching rule led to the generation of at least two child nodes. The last bunch of entries are related to branch and bound tree statistics: the number of leaves in column "leaves", the number of nodes still to be processed in column "nodes left", the maximum depth reached during the searching process in "max depth" and in the last column the number of the backtracks steps performed, column "backtracks".
Table 7.1 collects the above described key performance indicators averaged on the entire Carbin testbed. Additionally, row "optimal" shows the number of instances that have been solved to optimality within the time limit, row "suboptimal" the number of times our method stopped with a finite but strictly non-zero GAP, and row "not identified" the number of instances for which at least one among the primal and the dual bound was not identified. Rows "UB opt" and "LB opt" show the number of times the bounds were proved to be the optimal makespan. The same values are provided also as a percentage. Row "pr. t amean" represents

the amount of seconds required by a single pricer call on the average. Note that in each call a dynamic programming round for each commodity is executed. Finally in the last row we provide the number of backtracks in percentage over the number of total nodes.
Column *Default* of Table 7.1 is referred to our Branch and Price in its default setting. Results show overall good performances of the developed algorithm, with a number of instances solved to optimality equal to 68.06%, an average GAP among all the testbed that is less than 10% and an average computational time of around 3700 seconds. Entries related to the optimality of the bounds show that in 81.63% of the cases the optimal makespan equals the free-flow one, while in only 6 instances the UB was proved to be optimal. This shows on one side that the path limitation in the original problem was on the average not a severe restriction and on the other side that the provided set of initial paths was in general not optimal. However, the limited number of identified candidate paths, 17.94 on the average out of a total of 273.82 nodes, reveals that few more paths were needed to obtain the optimal multicommodity transshipment. On four instances our method was not able to identify a valid dual solution and stopped while executing the pricing routine at the root node after a contained number of iterations. As observed from the complete tables in Appendix E, these situations correspond to the longest time horizons. Therefore, the behavior can be attributed to a difficulty of the pricing problem when working on a TEN with large dimension that requires the generation and extension of a huge number of labels, as confirmed by the "max labels" values that in these cases turn out to be one order of magnitude higher than the average number on the total test set. Timing values show that most of the computational time is spent by our strategy in performing the pricer step with an average time of around 300 seconds per call, while the resolution of the LPs took few seconds on the average. Branching-related results show on average a branching on the y variables around every two nodes while only one every five nodes for the k branching. In this second case, note that half of the time a cutoff or a valid constraint were identified. Finally, note that during the three hours of running time around two thirds of the generated decisional tree were processed with a contained number of backtracks in the exploration.
The second column of the table presents average results obtained when complete information on a feasible solution were provided to our algorithm. In particular, the incumbent was identified through the initialization step of the $VLNS$-based matheuristic presented in the previous chapter. Note that its makespan value equals the tailored time horizon of the instance. Results with the *Incumbent* setting shows somewhat a limited improvement in the algorithm performances with the same instances solved to optimality but an average decrease in the computational time of around 100 seconds and a GAP decreasing from 8.12% to 7.88%. In general this setting required a lower number of runs of the pricing routine and k-splittable-related branchings but a higher number of branchings on the binary variables that led to a slightly larger branch and bound tree. Finally, a similar behavior can be observed in both settings when exploring the decisional tree. As no significant achievements have been achieved providing a complete feasible solution, the remainder of this analysis section focuses on the results obtained with the default setting.

Analyzing aggregate results We now analyze key indicators aggregated by the k-splittable parameter as presented in Table 7.2. Results show evident improvements in the algorithm performance when increasing the flow split parameter, confirming how a higher number of

usable paths that can be activated is associated with a higher degree of freedom and thus with an easier identification of the optimal solutions for the transshipment process. More in detail, the number of instances closed to optimality raises from only 25.00% in the unsplittable case to 83.33% in the 3-splittable till reaching a total of 95.83% in 5-splittable setting with only one suboptimal instance. Accordingly, the average GAP decreases by one order of magnitude when more than one path can be used by each commodity, the average computational time decreases from around 8300 seconds when $k = 1$ to just 530 seconds when $k = 5$ and the number of nodes presents the same decreasing trend in favor of the 5-splittable setting. Finally, a similar behavior can be noticed in the average results on the bounds of the instances where the UB presents a higher optimality on the average in the unsplittable case whereas the LB in relaxed path restriction settings. In general the unsplittable case presents the highest average number of generated nodes, pricer calls and both types of branching iterations, revealing that a tighter number of usable paths entailed a consistent need for new paths and a recurrent violation of the integrality and k-splittable constraints. A final consideration concerns the impact of the k-splittable parameter on the pricing timings: note indeed that the average time per pricing call in the unsplittable instances is around 27 times higher than those with a 5-splittable setting. The reasons for this might be twofold: on one side this is related to a larger TEN size on which the pricer operates being this dimension directly determined by the considered time horizon; on the other side it is related to a higher number of performed branchings operations of type k that, introducing forbidden paths at the nodes, reduce the efficiency of the comparison rule to discard dominated labels. These result in a more time-consuming procedure, a limitation that is frequently experienced when dealing with label-based dynamic programming algorithms.
Table 7.3 presents averaged results this time aggregated by the number of commodities. Our Branch and Price algorithm presents better performances on less congested instances, i.e. when a lower number of commodities must be transshipped. However, this trend is not completely clear in the entire experiment as shown by the average GAP on the medium congestion level setting that significantly outperforms the others. Indeed, we can observe that the 4 commodity setting presents the higher percentage of instances solved to optimality in the shorter computational time, 75.00% in around 3000 seconds, against 58.33% in more than 4600 seconds for the 12 commodity case. Moreover, faster pricing steps are performed in this case, with an average pricing time per call that is 45 times less in the 4 commodity case than in the 12 commodity case. This behavior is motivated by the fact that on each call of the pricer the dynamic programming algorithm must be performed separately for each commodity and, as in the previous analysis, on a TEN whose dimension is strongly influenced by the considered time horizon. Therefore, the case with 12 commodities required an equal number of executions of the label-correcting algorithm at each node in the time-expansion of the original digraph over 55 time instants on the average (49 time instants in the 4 commodity case). Finally, the longer times spent on the pricing routine and the larger amount of columns added to the nodes reveal a higher need in the congested instances of additional paths to allow for a better multicommodity transshipment and arc capacities usage. This resulted in fewer branching steps and thus smaller branch and bound trees.

7.3.2. Use of the Branch and Price algorithm to assess the Matheuristic results quality

Possible limitations of exact Branch and Price approaches in coping with increasing-sized network instances are well know in general, and in the case of our algorithm such scalability issues are related to the specific pricing routine that relies on a iterative resolution of a Resource-Constrained SPP via a dynamic programming algorithm in a time-expansion of the original dynamic digraph. Indeed, performances of this procedure strongly depend on the size of the instances in terms of length of the considered time horizon and number of paths available for flow transsshipment. Note that both of these aspects directly affect the amount of labels to be generated and extended and thus of resources required during the resolution process. Therefore, coping with very large-scale instances, such as those considered in Subsection 6.3.2, would result prohibitive for this class of exact approaches. Nevertheless, our Branch and Price algorithm can give rise to a further valuable contribution when adopted as a tool to enrich the computational analysis of heuristic methods and secure enhanced quality assessment for the solutions identified on $QMCkSFP$ instances.
In particular in this section we present the experiments realized by making use of the Branch and Price algorithm to certifying the quality of the solutions obtained by the matheuristic approach presented in Chapter 6. The testbed for this batch of experiments is composed of the Grid instances where the solutions of the matheuristic were not certified as being optimal when compared to the free-flow relaxation makespan, see Subsection 6.3.2. This corresponds to a testbed presenting a total of 80 instances with a number of layers ranging in $\{2, 3, \ldots, 10\}$ (being each of them a 5×5 grid with 80 arcs from 1 to 10 periods long and bottleneck arcs connecting different layers) 5 different levels of commodities and a flow split parameter that varies in $\{1, \ldots, 6\}$. For each experiment we allow six hours of computational time, provide the free-flow relaxation makespan as a valid lower bound (LB), recall by construction it is equal to 24 time instants for all the instances, and tailor the allowed time-horizon based on the makespan of the best solution obtained by the matheuristic approach.
Appendix F presents the complete set of results for this Grid experiment. The first six columns show information related to the instance: its name is reported as g-x-y-z where x is the number of layers, y the level of the commodity and z the k-splittable parameter; column “nodes”, “arcs”,“h” and “k” report the number of nodes, directed arcs, commodities and the flow split parameter, respectively. Column “LB” shows the value of the free-flow makespan, thus representing a lower bound on the instance, while the next two columns present the matheuristic best solutions after three hours of running time, column “best sol”, and the related optimality GAP w.r.t. the known free-flow lower bound, column “GAP”. The remaining columns are devoted to report the results obtained after applying our exact method with six hours of running time: the certified dual solution is presented in “dual sol”, the best final solution in “best sol” and the new optimality GAP calculated by these new information in column “GAP”.
Averaged results are collected in Table 7.4. Results show that in 55% of the instances the Branch and Price algorithm was able to identify better optimality GAPs with respect to the ones previously computed, see rows “improved GAP”. In particular, all improvements were achieved thanks to better certified lower bounds while none of the matheuristic best

solutions were discarded for better incumbents, see rows "improved LB" and "improved best sol". Note that in 65.91% of these cases the resulting GAP is equal to zero, proving thus the previously unknown optimality of the matheuristic best solutions, see rows "optimal". In the remaining 15 cases, collected in rows "suboptimal", the new GAP reached 4.59% on the average, with an improvement w.r.t. the matheuristic's ones of 10.11% on the average. On one hand, these results are relevant as they confirm the very high quality levels of the matheuristic best solutions on the whole Grid test bed, where a large number of certified optimalities was already identified, mainly on small to medium-sized Grid instances. On the other hand, these second batch of experiments was instrumental to validate the suitability of the Branch and Price algorithm as a precious tool to better assess the quality of the heuristic solutions on those medium to large-sized instances where a purely exact approach is not an option due to computational hardness.

8. Conclusions and outlook

In this thesis the challenge of routing flows on a bounded number of active paths within dynamic flow networks was investigated. In particular, we focused on the joint requirements of performing flow transshipment operations as quickly as possible and through a prefixed maximum number of supporting paths for each distinct commodity flow.
In the first chapters we recalled some major methodological results, relevant for the considered network flow modeling setting, ranging from basic notions from Network Flow theory to more sophisticated optimization tools known as Flows over time, dealing with the modellization of flow routing over a certain time horizon, focusing in particular on the quickest flow variants. Moreover, we discussed some results from the literature on k-splittable flows, a class of modeling tools securing the activation of at most k paths for each commodity demand. Despite the apparent advantages arising from the integration of realistic path number restrictions in dynamic flow routing, an almost complete lack of scientific contributions addressing this topic was identified through a wide and detailed literature review process.
Exploring the combination of the above mentioned tools within the same network optimization framework represented the core of our research activity in this thesis. The aim was to match a research question widely motivated by an increased amount of real-world applications that can provably benefit from such enhanced optimization toolbox. This encompasses all those contexts where a thorough control on both the time spent for flow transshipment and on the number and features of the activated support paths is a key enabling factor for the overall success of the processes to be implemented.
Several original contributions were developed in this thesis, organized as follows. In Chapter 5 we formally introduced a novel dynamic flow problem, namely the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$), explicitly integrating a number of concepts and modeling characteristics from the literature on k-splittable flows within the class of Quickest Flow optimization problems. Moreover, the strongly NP-hard complexity of the problem was here proved and a path-based Mixed-Integer Linear Programming formulation provided.
In Chapter 6 the first resolution approach, designed ad-hoc to solve efficiently large instances of the $QMCkSFP$, was detailed. The proposed method builds on a Very Large-scale Neighborhood Search algorithm, hybridized in its exploration routine with an exact mathematical programming technique. Following a Variable Neighborhood Descent scheme, the algorithm iteratively constructs large neighborhoods identifying for each commodity a collection of paths with a given cardinality. The elements of the collections are selected, according to a set of heuristic rules, from a large list of promising quickest paths generated during the initialization phase. The matheuristic explores to optimality each constructed neighborhood by means of the presented path-based formulation, fed with such restricted collections of paths,

and the best solution found during the overall computational process is finally returned.
The conducted computational experience provided an exhaustive proof-of-concept for the correctness and effectiveness of the proposed matheuristic resolution strategy and the associated MIP formulation, gaining provably optimal solutions on some small to medium-sized instances. On very large-sized benchmark instances and even in presence of a large number of commodities, the matheuristic revealed its high level of efficiency, outperforming the solution quality and computational times of the considered competing approach, which is based on the Randomized Rounding heuristic, a state-of-the-art scheme for path-restricted flow optimization problems.
Chapter 7 was devoted to present a second original algorithm, designed for the resolution to optimality of the $\mathsf{QMCkSFP}$ with the aim of providing an efficient modeling support tool to real-world contexts where the need for optimal solutions is highly required. The exact algorithm, based on the Branch and Price scheme, explores the feasible region alternating Column Generation and Branch and Bound steps by employing a relaxed and linearized path-based formulation of the $\mathsf{QMCkSFP}$ where flow split-related constraints are removed. A first original branching rule aims at recovering the path number limitation feasibility of the optimal solutions to the RRMPs. Working on path-flow variables, it forbids the usage of some paths while forcing the activation of some others over the considered time horizon. A second strategy implements a refined version of the classical fractional branching to restore integrality of binary variables. The pricing problem, in charge of selecting new promising entering columns while taking into account local branching cuts, is modeled for each commodity in the TEN of the original dynamic digraph as an extended version of the Shortest Path Problem with Forbidden Paths (SPPFP) with additional limited node-set resources required to avoid the generation of loops over time. Such SPPFP is then solved via a dynamic programming algorithm, where the consumption of limited resources, in terms of consecutive arcs for forbidden path-related resources and of traversed nodes for the node-set resources, is stored into labels that are extended throughout the procedure.
A computational experience was conducted to test quality performances of the proposed Branch and Price algorithm examining in particular the impact of its pricing and branching routines on the overall resolution process. The experiments performed on small to medium-sized instances showed overall good performances in terms of reached optimality, achieved in 68.06% of the cases, with average GAP lower than 10% on the average after three hours of running time. No relevant further improvements were obtained when providing complete information on a feasible solution as an input. The pricing problem showed a significant impact on the algorithm performances, in particular on more congested instances, revealing a difficulty of the dynamic programming routine in solving resource constrained SPPs on large networks. These scalability issues are motivated by the specific dynamic setting of the problem, in particular by the considered time horizon that directly determines the size of the TENs where the pricing problem is modeled, that in turn strongly affects the number of labels to be generated and extended. An additional experiment was performed with the aim of enriching computational analysis of the VLNS-based matheuristic approach on the same collection of small to medium-sized instances previously employed in the matheuristic experiments. Better lower bounds have been achieved thanks to the Branch and Price algorithm, often providing a certificate of optimality for the matheuristic's solutions.
The overall outcome of this thesis results therefore in a complete optimization toolbox dealing

with the introduction of realistic limitations on the number of paths for routing multicommodity flows on capacitated network structures while minimizing the required time horizon. The development of two efficient algorithmic contributions, based on a matheuristic and on an exact approach respectively, secure an effective support to different decision making operations in a wide range of real-life applications, both when good quality solutions in short computational times or optimal solutions must be identified.

8.1. Outlook

Some of the future developments for this research will be related to the refinement of theoretical and algorithmic features of the proposed approaches and to the increase of the adherence of such complete toolbox to real-world situations. As concerns the former, a strong research focus will be devoted to improve the performance of the pricing problem solver, which showed sometimes scalability issues while tackling very large instances of the dynamic problem. Some preliminary efforts in this direction were already conducted while developing this work, by adapting state-of-the-art techniques [30, 16] conceived to reduce the burden of labels to be generated and extended in the context of static flows. In particular, a bounding procedure on the costs, identifying all-pairs shortest paths w.r.t. reduced costs in the TEN, was employed to early discard non-dominated labels by provably worst associated costs. Moreover, an iterative procedure was devised to incrementally include node-set resources in the label structures only when needed, i.e. in the case of identified optimal non-elementary paths. Finally, a heuristic rule was implemented to interrupt the dynamic programming algorithm at the first realization of a label reaching the sink node with an associated negative reduced cost. However, these first attempts produced no relevant achievements w.r.t. pricing computational times in the conducted experiments. Therefore, ad-hoc improvement strategies, specifically tailored to the problem features, will be investigated to further increase the applicability of the developed Branch and Price exact approach securing the identification of certified optimal solutions in a wider range of practical applications.
A second future research line that may be worth to investigate starting from this thesis is represented by a concrete application of the arising contributions, namely the adoption and effective use of the proposed $QMCkSFP$ formulation and the associated resolution algorithms to support relevant and complex situations arising as instance in evacuation management and in the transportation field, pursuing thus the practical goal that motivated the study of this scientific setting. This could require an adaptation of the optimization problem in order to account for specific further operational features, hence posing new challenges from the modeling and algorithmic perspective.
To conclude, we believe that the formal scientific structure underlying the novel Network Flow optimization framework introduced in this thesis, based on flows over time subject to path limitations, will open the door to a number of original contributions in the field and thus to an efficient and valid support for a wider range of real-world applications.

Appendices

Appendix A.

	Instance				Matheuristic					CPLEX	
	nodes	arcs	h	k	init sol	t (s)	best sol	t (s)	moves	opt sol	t (s)
s-2-1-1	18	57	1	1	31	0	31	0	0	31	24
s-2-1-2	18	57	1	2	22	0	21	0	1	21	9
s-2-1-3	18	57	1	3	19	0	19	0	0	19	10
s-2-1-4	18	57	1	4	19	0	18	0	1	18	9
s-2-1-5	18	57	1	5	19	0	18	0	1	18	8
s-2-1-6	18	57	1	6	19	0	18	0	1	18	8
s-2-2-1	18	57	2	1	37	0	37	0	0	37	35
s-2-2-2	18	57	2	2	24	0	24	0	0	24	25
s-2-2-3	18	57	2	3	24	0	22	0	2	22	24
s-2-2-4	18	57	2	4	24	0	21	0	3	21	23
s-2-2-5	18	57	2	5	22	0	21	0	1	21	22
s-2-2-6	18	57	2	6	21	0	21	0	0	21	24
s-2-3-1	18	57	3	1	35	0	35	0	0	35	92
s-2-3-2	18	57	3	2	24	0	24	0	0	24	43
s-2-3-3	18	57	3	3	23	0	20	0	2	20	41
s-2-3-4	18	57	3	4	23	0	19	2	4	19	40
s-2-3-5	18	57	3	5	23	0	19	3	4	19	39
s-2-3-6	18	57	3	6	20	0	19	1	1	19	41
s-3-1-1	27	90	2	1	23	0	23	0	0	23	32
s-3-1-2	27	90	2	2	23	0	19	0	2	19	24
s-3-1-3	27	90	2	3	19	0	19	0	0	19	22
s-3-1-4	27	90	2	4	19	0	19	0	0	19	23
s-3-1-5	27	90	2	5	19	0	19	0	0	19	22
s-3-1-6	27	90	2	6	19	0	19	0	0	19	23
s-3-2-1	27	90	4	1	38	0	38	0	0	38	56
s-3-2-2	27	90	4	2	38	0	33	0	1	33	56
s-3-2-3	27	90	4	3	38	0	31	2	3	31	54
s-3-2-4	27	90	4	4	33	0	30	0	2	30	54
s-3-2-5	27	90	4	5	33	0	30	0	3	30	54
s-3-2-6	27	90	4	6	31	0	30	0	1	30	53
s-3-3-1	27	90	6	1	42	3	35	7	2	35	66662
s-3-3-2	27	90	6	2	29	0	29	0	0	29	82717
s-3-3-3	27	90	6	3	28	0	28	0	0	28	78536
s-3-3-4	27	90	6	4	28	0	28	0	0	28	75298
s-3-3-5	27	90	6	5	28	0	28	0	0	28	75411
s-3-3-6	27	90	6	6	28	1	28	1	0	28	79853

Table 1: Matheuristic: experiments on reduced size instances

Appendix B.

	Instance				Matheuristic				
	nodes	arcs	h	k	init sol	t (s)	best sol	t (s)	moves
g-2-1-1	50	185	1	1	32	0	32	0	0
g-2-1-2	50	185	1	2	32	0	24*	1	4
g-2-1-3	50	185	1	3	28	0	24*	0	3
g-2-1-4	50	185	1	4	24*	0	24*	0	0
g-2-1-5	50	185	1	5	24*	0	24*	0	0
g-2-1-6	50	185	1	6	24*	0	24*	0	0
g-2-2-1	50	185	2	1	36	0	36	0	0
g-2-2-2	50	185	2	2	27	0	26	0	1
g-2-2-3	50	185	2	3	24*	0	24*	0	0
g-2-2-4	50	185	2	4	24*	0	24*	0	0
g-2-2-5	50	185	2	5	24*	0	24*	0	0
g-2-2-6	50	185	2	6	24*	0	24*	0	0
g-2-3-1	50	185	3	1	35	0	35	0	0
g-2-3-2	50	185	3	2	27	0	26	1	1
g-2-3-3	50	185	3	3	26	1	25	1	1
g-2-3-4	50	185	3	4	25	0	24*	21	1
g-2-3-5	50	185	3	5	25	1	24*	2	1
g-2-3-6	50	185	3	6	25	0	24*	2	1
g-2-4-1	50	185	4	1	31	0	31	0	0
g-2-4-2	50	185	4	2	31	1	26	1	4
g-2-4-3	50	185	4	3	29	0	24*	8	5
g-2-4-4	50	185	4	4	27	1	24*	8	3
g-2-4-5	50	185	4	5	27	0	24*	0	2
g-2-4-6	50	185	4	6	27	1	24*	4	3
g-2-5-1	50	185	5	1	44	0	34	1	2
g-2-5-2	50	185	5	2	28	0	28	0	0
g-2-5-3	50	185	5	3	26	0	26	0	0
g-2-5-4	50	185	5	4	26	0	25	1	1
g-2-5-5	50	185	5	5	26	1	24*	1	1
g-2-5-6	50	185	5	6	26	1	24*	1	1
g-3-1-1	75	290	2	1	37	0	37	0	0
g-3-1-2	75	290	2	2	37	0	26	1	6
g-3-1-3	75	290	2	3	26	0	24*	2	2
g-3-1-4	75	290	2	4	26	0	24*	0	2
g-3-1-5	75	290	2	5	24*	0	24*	0	0
g-3-1-6	75	290	2	6	24*	0	24*	0	0
g-3-2-1	75	290	4	1	40	0	34	1	2
g-3-2-2	75	290	4	2	40	0	27	1	4
g-3-2-3	75	290	4	3	30	0	26	0	2
g-3-2-4	75	290	4	4	30	0	25	1	3
g-3-2-5	75	290	4	5	30	0	24*	1	5
g-3-2-6	75	290	4	6	26	0	24*	1	2
g-3-3-1	75	290	6	1	28	0	28	0	0
g-3-3-2	75	290	6	2	28	0	24*	1	3
g-3-3-3	75	290	6	3	28	0	24*	1	2
g-3-3-4	75	290	6	4	28	0	24*	1	2
g-3-3-5	75	290	6	5	24*	0	24*	0	0
g-3-3-6	75	290	6	6	24*	0	24*	0	0
g-3-4-1	75	290	8	1	48	0	32	7	6
g-3-4-2	75	290	8	2	40	0	26	68	4
g-3-4-3	75	290	8	3	40	1	24*	98	5
g-3-4-4	75	290	8	4	34	0	24*	17	4
g-3-4-5	75	290	8	5	34	0	24*	8	3
g-3-4-6	75	290	8	6	34	0	24*	1	3
g-3-5-1	75	290	10	1	67	0	39	10	9
g-3-5-2	75	290	10	2	56	0	29	123	7
g-3-5-3	75	290	10	3	35	1	26	15	5
g-3-5-4	75	290	10	4	33	1	25	2	4
g-3-5-5	75	290	10	5	30	0	24*	72	4
g-3-5-6	75	290	10	6	30	1	24*	5	4
g-4-1-1	100	395	3	1	30	1	30	1	0
g-4-1-2	100	395	3	2	30	0	25	1	4
g-4-1-3	100	395	3	3	30	0	24*	1	3
g-4-1-4	100	395	3	4	30	0	24*	1	4
g-4-1-5	100	395	3	5	25	0	24*	0	1
g-4-1-6	100	395	3	6	25	0	24*	1	1
g-4-2-1	100	395	6	1	42	1	42	1	0
g-4-2-2	100	395	6	2	42	1	33	3	4
g-4-2-3	100	395	6	3	34	1	30	1	1
g-4-2-4	100	395	6	4	34	1	28	7	5
g-4-2-5	100	395	6	5	34	1	27	24	3
g-4-2-6	100	395	6	6	31	2	27	9	3
g-4-3-1	100	395	9	1	40	0	33	2	4
g-4-3-2	100	395	9	2	32	0	27	1	1
g-4-3-3	100	395	9	3	32	1	25	4	4
g-4-3-4	100	395	9	4	30	1	24	3	2
g-4-3-5	100	395	9	5	26	1	24*	3	2
g-4-3-6	100	395	9	6	26	0	24*	1	1
g-4-4-1	100	395	12	1	55	1	31	55	7
g-4-4-2	100	395	12	2	51	1	24*	65	7
g-4-4-3	100	395	12	3	27	1	24*	1	1
g-4-4-4	100	395	12	4	26	1	24*	1	1
g-4-4-5	100	395	12	5	25	2	24*	2	1
g-4-4-6	100	395	12	6	25	1	24*	1	1

Table 2: Matheuristic: experiments on the Grid test set

	Instance				Matheuristic				
	nodes	arcs	h	k	init sol	t (s)	best sol	t (s)	moves
g-4-5-1	100	395	15	1	64	1	43	3	2
g-4-5-2	100	395	15	2	37	1	30	25	2
g-4-5-3	100	395	15	3	31	2	26	68	4
g-4-5-4	100	395	15	4	31	3	25	45	3
g-4-5-5	100	395	15	5	30	1	24*	563	4
g-4-5-6	100	395	15	6	28	2	24*	138	3
g-5-1-1	125	500	4	1	27	0	27	0	0
g-5-1-2	125	500	4	2	27	0	25	1	2
g-5-1-3	125	500	4	3	25	0	24*	1	1
g-5-1-4	125	500	4	4	24*	0	24*	0	0
g-5-1-5	125	500	4	5	24*	0	24*	0	0
g-5-1-6	125	500	4	6	24*	0	24*	0	0
g-5-2-1	125	500	8	1	24*	1	24*	1	0
g-5-2-2	125	500	8	2	24*	1	24*	1	0
g-5-2-3	125	500	8	3	24*	0	24*	0	0
g-5-2-4	125	500	8	4	24*	1	24*	1	0
g-5-2-5	125	500	8	5	24*	1	24*	1	0
g-5-2-6	125	500	8	6	24*	1	24*	1	0
g-5-3-1	125	500	12	1	28	2	28	2	0
g-5-3-2	125	500	12	2	25	1	25	1	0
g-5-3-3	125	500	12	3	25	1	25	1	0
g-5-3-4	125	500	12	4	25	1	25	1	0
g-5-3-5	125	500	12	5	24*	1	24*	1	0
g-5-3-6	125	500	12	6	24*	1	24*	1	0
g-5-4-1	125	500	16	1	27	1	27	1	0
g-5-4-2	125	500	16	2	26	2	24*	7	2
g-5-4-3	125	500	16	3	26	2	24*	3	1
g-5-4-4	125	500	16	4	24*	1	24*	1	0
g-5-4-5	125	500	16	5	24*	1	24*	1	0
g-5-4-6	125	500	16	6	24*	2	24*	2	0
g-5-5-1	125	500	20	1	39	2	27	464	7
g-5-5-2	125	500	20	2	31	2	25	169	5
g-5-5-3	125	500	20	3	26	2	24*	9	2
g-5-5-4	125	500	20	4	25	2	24*	6	1
g-5-5-5	125	500	20	5	25	2	24*	2	1
g-5-5-6	125	500	20	6	24*	2	24*	2	0
g-6-1-1	150	605	5	1	29	0	29	0	0
g-6-1-2	150	605	5	2	27	0	25	1	2
g-6-1-3	150	605	5	3	25	1	24*	1	1
g-6-1-4	150	605	5	4	24*	0	24*	0	0
g-6-1-5	150	605	5	5	24*	1	24*	1	0
g-6-1-6	150	605	5	6	24*	0	24*	0	0
g-6-2-1	150	605	10	1	27	1	27	1	0
g-6-2-2	150	605	10	2	25	2	25	2	0
g-6-2-3	150	605	10	3	25	1	24*	7	1
g-6-2-4	150	605	10	4	25	2	24*	2	1
g-6-2-5	150	605	10	5	25	2	24*	3	1
g-6-2-6	150	605	10	6	24*	2	24*	2	0
g-6-3-1	150	605	15	1	33	2	28	5	3
g-6-3-2	150	605	15	2	29	2	25	30	3
g-6-3-3	150	605	15	3	27	2	24*	192	3
g-6-3-4	150	605	15	4	25	2	24*	5	1
g-6-3-5	150	605	15	5	25	2	24*	2	0
g-6-3-6	150	605	15	6	24*	2	24*	2	0
g-6-4-1	150	605	20	1	30	3	26	4	2
g-6-4-2	150	605	20	2	24*	3	24*	3	0
g-6-4-3	150	605	20	3	24*	3	24*	3	0
g-6-4-4	150	605	20	4	24*	3	24*	3	0
g-6-4-5	150	605	20	5	24*	3	24*	3	0
g-6-4-6	150	605	20	6	24*	3	24*	3	0

	Instance				Matheuristic				
	nodes	arcs	h	k	init sol	t (s)	best sol	t (s)	moves
g-6-5-1	150	605	25	1	40	3	28	132	4
g-6-5-2	150	605	25	2	33	3	25	99	3
g-6-5-3	150	605	25	3	29	4	24*	833	4
g-6-5-4	150	605	25	4	29	4	24*	25	3
g-6-5-5	150	605	25	5	29	3	24*	10	2
g-6-5-6	150	605	25	6	29	4	24*	10	3
g-7-1-1	175	710	6	1	26	3	26	3	0
g-7-1-2	175	710	6	2	24*	2	24*	2	0
g-7-1-3	175	710	6	3	24*	2	24*	2	0
g-7-1-4	175	710	6	4	24*	2	24*	2	0
g-7-1-5	175	710	6	5	24*	2	24*	2	0
g-7-1-6	175	710	6	6	24*	2	24*	2	0
g-7-2-1	175	710	12	1	27	4	27	4	0
g-7-2-2	175	710	12	2	27	3	25	40	2
g-7-2-3	175	710	12	3	27	4	24*	15	3
g-7-2-4	175	710	12	4	27	4	24*	10	2
g-7-2-5	175	710	12	5	27	4	24*	59	3
g-7-2-6	175	710	12	6	27	6	24*	9	3
g-7-3-1	175	710	18	1	32	4	24*	37	4
g-7-3-2	175	710	18	2	25	4	24*	7	1
g-7-3-3	175	710	18	3	24*	4	24*	4	0
g-7-3-4	175	710	18	4	24*	4	24*	4	0
g-7-3-5	175	710	18	5	24*	5	24*	5	0
g-7-3-6	175	710	18	6	24*	4	24*	4	0
g-7-4-1	175	710	24	1	41	7	27	226	6
g-7-4-2	175	710	24	2	35	7	25	262	4
g-7-4-3	175	710	24	3	29	7	25	16	3
g-7-4-4	175	710	24	4	29	7	24*	3321	4
g-7-4-5	175	710	24	5	29	7	25	24	4
g-7-4-6	175	710	24	6	29	7	24*	2333	3
g-7-5-1	175	710	30	1	37	9	24*	2362	7
g-7-5-2	175	710	30	2	32	10	24*	16	3
g-7-5-3	175	710	30	3	24*	9	24*	9	0
g-7-5-4	175	710	30	4	24*	9	24*	9	0
g-7-5-5	175	710	30	5	24*	9	24*	9	0
g-7-5-6	175	710	30	6	24*	9	24*	9	0
g-8-1-1	200	815	7	1	26	3	26	3	0
g-8-1-2	200	815	7	2	26	3	24*	15	2
g-8-1-3	200	815	7	3	26	3	24*	4	2
g-8-1-4	200	815	7	4	26	3	24*	6	2
g-8-1-5	200	815	7	5	26	3	24*	5	2
g-8-1-6	200	815	7	6	26	3	24*	5	2
g-8-2-1	200	815	14	1	28	6	27	31	1
g-8-2-2	200	815	14	2	27	5	24*	10	2
g-8-2-3	200	815	14	3	27	5	24*	6	2
g-8-2-4	200	815	14	4	27	5	24*	6	2
g-8-2-5	200	815	14	5	27	6	24*	7	2
g-8-2-6	200	815	14	6	27	6	24*	7	2
g-8-3-1	200	815	21	1	26	8	25	162	1
g-8-3-2	200	815	21	2	26	8	24*	271	2
g-8-3-3	200	815	21	3	26	8	24*	113	2
g-8-3-4	200	815	21	4	25	8	24*	8	1
g-8-3-5	200	815	21	5	25	8	24*	9	1
g-8-3-6	200	815	21	6	25	8	24*	9	1
g-8-4-1	200	815	28	1	34	10	27	14	5
g-8-4-2	200	815	28	2	25	10	24*	15	1
g-8-4-3	200	815	28	3	25	10	24*	12	1
g-8-4-4	200	815	28	4	25	10	24*	27	1
g-8-4-5	200	815	28	5	25	11	24*	12	1
g-8-4-6	200	815	28	6	25	11	24*	16	1

	Instance				Matheuristic				
	nodes	arcs	h	k	init	t	best	t	moves
					sol	(s)	sol	(s)	
g-8-5-1	200	815	35	1	57	14	27	1148	6
g-8-5-2	200	815	35	2	35	14	25	35	3
g-8-5-3	200	815	35	3	35	14	24*	1267	5
g-8-5-4	200	815	35	4	32	13	24*	1369	3
g-8-5-5	200	815	35	5	32	14	24*	43	4
g-8-6-5	200	815	35	6	32	15	24*	71	3
g-9-1-1	225	920	8	1	24*	2	24*	2	0
g-9-1-2	225	920	8	2	24*	1	24*	1	0
g-9-1-3	225	920	8	3	24*	1	24*	1	0
g-9-1-4	225	920	8	4	24*	2	24*	2	0
g-9-1-5	225	920	8	5	24*	2	24*	2	0
g-9-1-6	225	920	8	6	24*	2	24*	2	0
g-9-2-1	225	920	16	1	25	4	25	4	0
g-9-2-2	225	920	16	2	24*	3	24*	3	0
g-9-2-3	225	920	16	3	24*	3	24*	3	0
g-9-2-4	225	920	16	4	24*	4	24*	4	0
g-9-2-5	225	920	16	5	24*	4	24*	4	0
g-9-2-6	225	920	16	6	24*	4	24*	4	0
g-9-3-1	225	920	24	1	31	5	27	7	2
g-9-3-2	225	920	24	2	26	5	25	14	1
g-9-3-3	225	920	24	3	26	5	24*	1567	2
g-9-3-4	225	920	24	4	26	6	24*	102	2
g-9-3-5	225	920	24	5	25	6	24*	14	1
g-9-3-6	225	920	24	6	25	5	24*	29	1
g-9-4-1	225	920	32	1	36	7	26	38	2
g-9-4-2	225	920	32	2	27	7	24*	9	2
g-9-4-3	225	920	32	3	26	7	24*	10	2
g-9-4-4	225	920	32	4	25	7	24*	7	1
g-9-4-5	225	920	32	5	25	7	24*	8	1
g-9-4-6	225	920	32	6	24*	8	24*	8	0
g-9-5-1	225	920	40	1	26	8	26	8	0
g-9-5-2	225	920	40	2	24*	8	24*	8	0
g-9-5-3	225	920	40	3	24*	8	24*	8	0
g-9-5-4	225	920	40	4	24*	9	24*	9	0
g-9-5-5	225	920	40	5	24*	8	24*	8	0
g-9-5-6	225	920	40	6	24*	8	24*	8	0
g-10-1-1	250	1025	9	1	24*	2	24*	2	0
g-10-1-2	250	1025	9	2	24*	2	24*	2	0
g-10-1-3	250	1025	9	3	24*	2	24*	2	0
g-10-1-4	250	1025	9	4	24*	2	24*	2	0
g-10-1-5	250	1025	9	5	24*	3	24*	3	0
g-10-1-6	250	1025	9	6	24*	3	24*	3	0
g-10-2-1	250	1025	18	1	24*	4	24*	4	0
g-10-2-2	250	1025	18	2	24*	4	24*	4	0
g-10-2-3	250	1025	18	3	24*	4	24*	4	0
g-10-2-4	250	1025	18	4	24*	5	24*	5	0
g-10-2-5	250	1025	18	5	24*	4	24*	4	0
g-10-2-6	250	1025	18	6	24*	4	24*	4	0
g-10-3-1	250	1025	27	1	31	7	26	8	3
g-10-3-2	250	1025	27	2	29	7	25	9	2
g-10-3-3	250	1025	27	3	25	7	24*	45	1
g-10-3-4	250	1025	27	4	25	7	24*	9	1
g-10-3-5	250	1025	27	5	25	7	24*	8	1
g-10-3-6	250	1025	27	6	25	7	24*	9	1
g-10-4-1	250	1025	36	1	61	8	27	3022	9
g-10-4-2	250	1025	36	2	48	8	25	1276	5
g-10-4-3	250	1025	36	3	37	9	24*	2666	4
g-10-4-4	250	1025	36	4	34	10	24*	637	4
g-10-4-5	250	1025	36	5	34	9	24*	30	4
g-10-4-6	250	1025	36	6	31	10	24*	17	3
g-10-5-1	250	1025	45	1	55	11	26	2221	5
g-10-5-2	250	1025	45	2	29	12	25	15	4
g-10-5-3	250	1025	45	3	27	13	24*	20	1
g-10-5-4	250	1025	45	4	27	13	24*	17	2
g-10-5-5	250	1025	45	5	27	13	24*	16	1
g-10-5-6	250	1025	45	6	27	13	24*	21	2

Appendix C.

Instance					Matheuristic					*RR*				
	nodes	arcs	h	k	init sol	t (s)	best sol	t (s)	moves	init sol	t (s)	best sol	t (s)	moves
d-10-1	10	45	5	1	31	0	31	0	0	41	0	31	0	3
d-10-2	10	45	5	2	31	0	31	0	0	31	0	31	0	0
d-10-3	10	45	5	3	31	0	31	0	0	31	0	31	0	0
d-10-4	10	45	5	4	31	0	31	0	0	31	0	31	0	0
d-10-5	10	45	5	5	31	0	31	0	0	31	0	31	0	0
d-10-6	10	45	5	6	31	0	31	0	0	31	0	31	0	0
d-20-1	20	190	5	1	51	0	51	0	0	56	1	51	1	1
d-20-2	20	190	5	2	30	0	30	0	0	41	0	31	1	2
d-20-3	20	190	5	3	30	0	24	1	5	38	0	24	882	5
d-20-4	20	190	5	4	30	0	21	1	2	27	0	21	1260	5
d-20-5	20	190	5	5	26	0	19	54	3	29	0	19	502	9
d-20-6	20	190	5	6	22	0	18	145	4	28	0	18	57	6
d-30-1	30	435	5	1	63	0	63	0	0	67	3	63	13	1
d-30-2	30	435	5	2	55	0	34	3	2	63	1	36	695	8
d-30-3	30	435	5	3	30	0	25	2	3	32	1	29	163	3
d-30-4	30	435	5	4	27	0	21	5	5	37	1	25	482	5
d-30-5	30	435	5	5	26	0	20	46	5	37	1	22	2579	8
d-30-6	30	435	5	6	26	0	19	12	5	27	1	21	39	4
d-40-1	40	780	5	1	67	0	67	0	0	72	3	67	55	3
d-40-2	40	780	5	2	51	0	37	5	2	48	2	39	467	5
d-40-3	40	780	5	3	30	0	28	0	1	39	2	28	595	8
d-40-4	40	780	5	4	23	0	23	0	0	35	2	24	993	7
d-40-5	40	780	5	5	23	0	20	2	2	40	2	21	1583	6
d-40-6	40	780	5	6	23	0	18	8	3	28	2	20	2172	4
d-50-1	50	1225	5	1	56	0	56	0	0	68	17	56	57	4
d-50-2	50	1225	5	2	30	0	30	0	0	68	3	30	3056	6
d-50-3	50	1225	5	3	30	0	22	4	4	31	3	23	3494	5
d-50-4	50	1225	5	4	22	0	19	2	2	25	3	20	356	3
d-50-5	50	1225	5	5	19	0	17	0	2	25	4	18	164	5
d-50-6	50	1225	5	6	19	0	15	8	4	21	4	17	55	3
d-60-1	60	1770	5	1	53	0	53	0	0	61	7	56	73	3
d-60-2	60	1770	5	2	51	0	30	4	3	54	6	33	1135	6
d-60-3	60	1770	5	3	30	0	22	129	3	38	5	25	2851	6
d-60-4	60	1770	5	4	23	1	18	5	3	30	5	21	2179	4
d-60-5	60	1770	5	5	21	0	16	3	3	28	5	19	2045	5
d-60-6	60	1770	5	6	19	0	14	146	4	25	5	18	43	3
d-70-1	70	2415	5	1	68	0	68	0	0	68	113	68	113	0
d-70-2	70	2415	5	2	52	0	38	1	2	54	8	42	1414	7
d-70-3	70	2415	5	3	30	1	28	1	1	61	8	31	2050	10
d-70-4	70	2415	5	4	30	1	23	2	4	38	8	26	339	7
d-70-5	70	2415	5	5	29	0	21	12	4	31	7	23	2950	5
d-70-6	70	2415	5	6	28	1	19	12	5	28	7	21	94	3

Table 3: Matheuristic: experiments on the Dense test set

		Instance			Matheuristic					*RR*				
	nodes	arcs	h	k	init sol	t (s)	best sol	t (s)	moves	init sol	t (s)	best sol	t (s)	moves
d-80-1	80	3160	5	1	49	0	49	0	0	70	10	49	317	5
d-80-2	80	3160	5	2	30	0	27	1	1	67	11	30	3430	8
d-80-3	80	3160	5	3	30	1	21	2	3	47	9	25	68	8
d-80-4	80	3160	5	4	27	1	18	2	3	29	9	20	1772	6
d-80-5	80	3160	5	5	27	1	16	5	5	22	9	17	3326	5
d-80-6	80	3160	5	6	21	1	15	7	3	19	9	17	36	2
d-90-1	90	4005	5	1	55	0	55	0	0	70	25	55	210	5
d-90-2	90	4005	5	2	30	0	30	0	0	49	13	32	1703	6
d-90-3	90	4005	5	3	30	0	21	106	3	58	14	24	1896	8
d-90-4	90	4005	5	4	29	0	17	3	3	33	12	20	96	5
d-90-5	90	4005	5	5	27	1	15	12	3	27	12	18	162	6
d-90-6	90	4005	5	6	27	1	13	40	5	19	13	16	2100	3
d-100-1	100	4950	5	1	52	1	52	1	0	63	52	52	1943	4
d-100-2	100	4950	5	2	31	0	30	17	1	69	15	32	1360	5
d-100-3	100	4950	5	3	30	0	23	5	2	52	16	26	1670	8
d-100-4	100	4950	5	4	23	1	18	4	3	32	15	21	1314	7
d-100-5	100	4950	5	5	23	1	16	4	4	27	15	19	1157	5
d-100-6	100	4950	5	6	19	1	14	227	4	21	16	18	1019	2
d-150-1	150	11175	5	1	37	2	37	2	0	63	43	37	2200	5
d-150-2	150	11175	5	2	37	2	21	55	3	39	39	25	115	5
d-150-3	150	11175	5	3	37	2	16	133	3	40	39	20	141	3
d-150-4	150	11175	5	4	22	2	13	913	3	20	39	17	651	2
d-150-5	150	11175	5	5	17	2	12	9	2	17	39	15	1361	1
d-150-6	150	11175	5	6	15	2	11	5	2	17	40	14	187	2
d-200-1	200	19900	5	1	70	2	52	4	1	69	114	58	1011	3
d-200-2	200	19900	5	2	38	3	30	6	1	69	76	36	1838	5
d-200-3	200	19900	5	3	30	3	22	26	2	56	77	27	1071	6
d-200-4	200	19900	5	4	23	2	18	625	3	34	79	22	2282	5
d-200-5	200	19900	5	5	23	3	16	18	3	27	78	20	3671	5
d-200-6	200	19900	5	6	23	3	14	62	4	20	76	19	135	1
d-250-1	250	31125	5	1	57	3	57	3	0	62	470	61	1997	1
d-250-2	250	31125	5	2	39	3	31	6	1	50	139	38	1643	6
d-250-3	250	31125	5	3	30	4	22	7	1	55	131	27	1873	8
d-250-4	250	31125	5	4	22	4	17	118	2	32	132	23	905	2
d-250-5	250	31125	5	5	22	4	15	7	1	23	132	19	1382	2
d-250-6	250	31125	5	6	18	5	13	8	1	20	134	18	386	1
d-300-1	300	44850	5	1	57	4	57	4	0	66	691	63	2852	2
d-300-2	300	44850	5	2	31	4	31	4	0	49	226	39	620	2
d-300-3	300	44850	5	3	30	5	22	13	1	63	219	28	2536	7
d-300-4	300	44850	5	4	22	6	17	72	2	33	212	23	3778	2
d-300-5	300	44850	5	5	21	5	15	21	3	25	221	21	686	3
d-300-6	300	44850	5	6	21	5	13	38	3	20	211	19	574	1
d-350-1	350	61075	5	1	31	11	31	11	0	68	319	35	1130	7
d-350-2	350	61075	5	2	31	11	18	121	2	35	330	23	996	4
d-350-3	350	61075	5	3	31	12	14	49	3	33	327	19	2873	3
d-350-4	350	61075	5	4	31	11	11	446	5	21	326	15	3170	4
d-350-5	350	61075	5	5	31	13	10	382	6	17	316	14	2623	3
d-350-6	350	61075	5	6	31	10	10	136	4	15	318	14	510	1
d-400-1	400	79800	5	1	43	14	31	1068	5	64	238	37	1174	6
d-400-2	400	79800	5	2	31	14	18	77	2	47	210	24	872	3
d-400-3	400	79800	5	3	31	14	13	704	4	42	210	19	3089	5
d-400-4	400	79800	5	4	31	11	11	1903	5	22	209	17	667	2
d-400-5	400	79800	5	5	31	15	10	88	3	18	210	16	790	2
d-400-6	400	79800	5	6	31	15	9	146	5	18	211	14	2719	3
d-450-1	450	101025	5	1	57	11	57	11	0	71	1062	65	3642	4
d-450-2	450	101025	5	2	36	11	31	20	1	62	280	41	1363	4
d-450-3	450	101025	5	3	36	12	22	143	2	64	281	32	2960	6
d-450-4	450	101025	5	4	22	13	18	24	1	39	281	25	2307	4
d-450-5	450	101025	5	5	21	13	15	358	2	31	280	24	596	2
d-450-6	450	101025	5	6	18	13	14	24	1	23	282	22	371	1
d-500-1	500	124750	5	1	56	11	56	11	0	61	1581	61	1581	0
d-500-2	500	124750	5	2	38	12	31	23	1	55	378	43	2400	6
d-500-3	500	124750	5	3	38	13	23	24	1	58	368	32	1517	6
d-500-4	500	124750	5	4	38	12	18	62	3	31	368	26	3549	2
d-500-5	500	124750	5	5	24	13	16	40	2	27	371	24	1634	2
d-500-6	500	124750	5	6	22	14	14	157	2	25	368	21	3655	4

Appendix D.

	Instance				Matheuristic					RR				
	nodes	arcs	h	k	init	t	best	t	moves	init	t	best	t	moves
					sol	(s)	sol	(s)		sol	(s)	sol	(s)	
Bl01-1	32	96	48	1	39	1	39	1	0	55	9	41	10	10
Bl01-2	32	96	48	2	39	1	39	1	0	39	9	39	9	0
Bl01-3	32	96	48	3	39	2	39	2	0	39	9	38	9	2
Bl01-4	32	96	48	4	39	2	39	2	0	42	9	38	9	3
Bl01-5	32	96	48	5	39	2	39	2	0	39	9	38	9	3
Bl01-6	32	96	48	6	39	2	39	2	0	38	9	38	9	0
Bl03-1	32	96	48	1	59	2	37	72	7	44	11	40	859	3
Bl03-2	32	96	48	2	52	2	37	4	3	41	11	37	29	3
Bl03-3	32	96	48	3	37	4	37	4	0	38	11	37	11	2
Bl03-4	32	96	48	4	37	5	37	5	0	38	12	37	14	2
Bl03-5	32	96	48	5	37	3	37	3	0	38	11	37	12	2
Bl03-6	32	96	48	6	37	2	37	2	0	37	11	37	11	0
Bl05-1	32	320	48	1	16	4	16	4	0	27	11	20	20	4
Bl05-2	32	320	48	2	15	5	15	5	0	21	11	15	121	5
Bl05-3	32	320	48	3	15	5	14	3314	1	22	11	14	66	3
Bl05-4	32	320	48	4	15	4	14	2656	1	22	11	14	14	4
Bl05-5	32	320	48	5	15	4	14	2696	1	17	11	14	21	2
Bl05-6	32	320	48	6	15	6	14	731	1	15	11	14	17	1
Bl07-1	32	320	48	1	17	4	17	4	0	25	11	19	2143	6
Bl07-2	32	320	48	2	17	4	15	306	2	21	11	16	381	4
Bl07-3	32	320	48	3	16	4	15	5	1	18	11	16	12	2
Bl07-4	32	320	48	4	16	4	15	5	1	16	11	15	899	1
Bl07-5	32	320	48	5	16	4	15	7	1	17	11	15	45	2
Bl07-6	32	320	48	6	16	7	15	10	1	17	11	15	19	2
Bs01-1	32	96	48	1	49	1	45	11	1	57	9	45	10	3
Bs01-2	32	96	48	2	45	0	42	1	1	49	9	42	11	2
Bs01-3	32	96	48	3	42	0	41	3	1	46	8	41	11	3
Bs01-4	32	96	48	4	41	1	41	1	0	41	8	41	8	0
Bs01-5	32	96	48	5	41	1	41	1	0	44	8	41	12	3
Bs01-6	32	96	48	6	41	1	41	1	0	41	9	41	9	0
Bs03-1	32	96	48	1	54	1	54	1	0	57	17	54	17	1
Bs03-2	32	96	48	2	52	0	52	0	0	57	17	52	17	2
Bs03-3	32	96	48	3	52	1	52	1	0	52	16	52	16	0
Bs03-4	32	96	48	4	52	1	52	1	0	52	17	52	17	0
Bs03-5	32	96	48	5	52	1	52	1	0	54	17	52	17	1
Bs03-6	32	96	48	6	52	1	52	1	0	52	17	52	17	0
Bs05-1	32	320	48	1	57	2	22	1656	8	29	17	22	2268	5
Bs05-2	32	320	48	2	28	3	16	272	8	24	17	17	415	4
Bs05-3	32	320	48	3	27	6	14	760	5	20	17	16	27	3
Bs05-4	32	320	48	4	27	17	14	264	4	18	17	15	270	3
Bs05-5	32	320	48	5	27	23	14	54	3	18	18	15	20	2
Bs05-6	32	320	48	6	27	23	14	54	3	16	18	14	1159	2
Bs07-1	32	320	48	1	47	2	20	2959	11	30	12	24	947	5
Bs07-2	32	320	48	2	29	2	17	11	2	24	12	19	390	3
Bs07-3	32	320	48	3	24	3	16	34	3	21	12	17	1191	4
Bs07-4	32	320	48	4	24	4	16	11	3	21	12	16	2126	3
Bs07-5	32	320	48	5	17	3	16	7	1	18	12	16	458	2
Bs07-6	32	320	48	6	17	3	16	7	1	20	12	16	28	2

Table 4: Matheuristic: experiments on the Carbin test set

Appendix E.

	primal	dual	GAP	nodes	t (s)	UB	LB	LP calls	LP t (s)	pr. call	pr. t (s)	pr. paths	pr. added vars	max labels	mean labels
Bl01-4-1	57	47	21.28%	1319	10800.00	57	39	7078	3.59	1340	10793.54	19	409	1727	616
Bl01-4-3	39	39	0.00%	14	4.66	41	39	84	0.00	18	4.62	3	24	349	158
Bl01-4-5	39	39	0.00%	1	0.30	40	39	6	0.00	2	0.28	1	13	310	158
Bl01-8-1	50	41	21.93%	715	10800.00	50	41	4610	1.77	776	10800.00	47	793	1752	90
Bl01-8-3	41	41	0.00%	476	2083.80	45	41	2751	1.22	510	2081.58	18	187	901	46
Bl01-8-5	41	41	0.00%	8	38.78	45	41	49	0.02	11	38.70	5	90	901	395
Bl01-12-1	65	47	38.23%	19	10800.00	65	47	112	0.12	40	10800.00	26	778	15700	4480
Bl01-12-3	54	47	14.87%	243	10800.00	54	47	1206	0.92	248	10799.51	7	106	4410	581
Bl01-12-5	47	47	0.00%	4	128.71	50	47	25	0.06	7	128.54	3	44	2583	803
Bs01-4-1	60	42	42.86%	1065	10800.00	60	39	6642	4.87	1137	10793.14	54	1376	2069	408
Bs01-4-3	40	40	0.00%	112	65.71	47	39	591	0.32	124	65.25	9	165	456	168
Bs01-4-5	39	39	0.00%	1	0.18	40	39	6	0.00	2	0.17	1	9	193	136
Bs01-8-1	54	43	25.33%	2096	10800.00	54	43	26305	13.39	3755	10689.17	86	1165	1686	55
Bs01-8-3	43	43	0.00%	1	0.69	44	43	6	0.01	2	0.66	1	10	497	173
Bs01-8-5	43	43	0.00%	1	0.70	44	43	6	0.00	2	0.67	1	10	497	174
Bs01-12-1	57	43	32.54%	819	10800.00	57	43	8973	6.64	1008	10779.99	166	3196	2362	126
Bs01-12-3	43	43	0.00%	1	5.17	47	43	8	0.00	3	5.13	3	50	735	288
Bs01-12-5	43	43	0.00%	1	1.42	44	43	6	0.01	2	1.37	1	10	497	199
Bl02-4-1	36	36	0.00%	37	20.60	36	35	234	0.03	48	20.51	9	35	172	69
Bl02-4-3	35	35	0.00%	4	2.53	36	35	25	0.00	7	2.51	3	12	172	118
Bl02-4-5	35	35	0.00%	2	1.12	36	35	11	0.00	3	1.09	1	3	172	122
Bl02-8-1	36	36	0.00%	19	27.83	36	34	104	0.01	23	27.77	4	12	1560	296
Bl02-8-3	34	34	0.00%	1	2.22	35	34	6	0.01	2	2.20	1	2	1345	291
Bl02-8-5	34	34	0.00%	1	1.61	35	34	6	0.01	2	1.56	1	2	1345	291
Bl02-12-1	36	36	0.00%	93	70.96	36	35	504	0.12	105	70.70	10	39	278	8
Bl02-12-3	35	35	0.00%	4	3.27	36	35	25	0.01	7	3.21	3	12	278	93
Bl02-12-5	35	35	0.00%	1	0.92	36	35	6	0.00	2	0.90	1	3	278	106
Bs02-4-1	57	39	46.15%	2642	10800.00	57	37	15412	6.15	2743	10780.21	44	1095	3008	432
Bs02-4-3	37	37	0.00%	1	1.41	44	37	8	0.01	3	1.37	2	40	681	284
Bs02-4-5	37	37	0.00%	1	0.24	40	37	6	0.01	2	0.21	1	11	390	176
Bs02-8-1	95	-	–	1	10800.00	95	53	6	0.05	4	10800.00	9	488	69119	21767
Bs02-8-3	53	53	0.00%	12	118.27	57	53	85	0.05	16	118.15	5	64	2871	1126
Bs02-8-5	53	53	0.00%	4	49.53	56	53	25	0.02	5	49.43	2	35	2528	1027
Bs02-12-1	57	37	53.84%	1566	10800.00	57	37	8647	4.20	1657	10742.33	42	1098	3008	48
Bs02-12-3	37	37	0.00%	1	1.49	44	37	8	0.00	3	1.45	2	40	659	171
Bs02-12-5	37	37	0.00%	1	0.47	40	37	6	0.00	2	0.43	1	11	390	107

Table 5: Branch and Price: general results and pricing table on the Carbin test set

	primal	dual	GAP	nodes	t (s)	UB	LB	LP calls	LP t (s)	pr. call	pr. t (s)	pr. paths	pr. added vars	max labels	mean labels
Bl03-4-1	60	42	42.84%	253	10800.00	60	42	3200	2.30	385	10800.00	113	2445	4769	2154
Bl03-4-3	43	43	0.00%	218	588.00	48	42	1378	0.56	239	587.04	14	157	1304	396
Bl03-4-5	42	42	0.00%	1	1.33	44	42	6	0.01	2	1.31	1	10	801	409
Bl03-8-1	76	-	–	1	10800.00	76	45	22	0.04	13	10800.00	16	519	32938	12743
Bl03-8-3	51	45	13.33%	2445	10800.00	51	45	21387	12.85	2610	10737.78	104	1178	1494	96
Bl03-8-5	50	45	11.11%	7	10800.00	50	45	1048	2.00	14	10700.00	7	200	1292	559
Bl03-12-1	58	41	41.41%	321	10800.00	58	41	1762	1.16	351	10800.00	35	802	5043	896
Bl03-12-3	55	41	34.15%	676	10800.00	55	41	4593	3.88	735	10797.72	40	828	3625	297
Bl03-12-5	41	41	0.00%	8	78.27	44	41	24	0.06	18	32.48	12	150	886	218
Bs03-4-1	68	44	54.55%	34	10800.00	68	44	306	0.21	62	10800.00	32	1069	16697	7606
Bs03-4-3	46	46	0.00%	2625	7286.69	48	44	26730	9.49	2758	7271.47	55	422	1389	105
Bs03-4-5	44	44	0.00%	41	144.99	47	44	279	0.15	50	144.73	11	108	1201	445
Bs03-8-1	48	48	0.00%	949	5860.61	48	44	7337	2.47	990	5856.51	42	449	7920	1571
Bs03-8-3	44	44	0.00%	11	94.47	47	44	58	0.02	12	94.34	2	10	1384	456
Bs03-8-5	44	44	0.00%	1	11.44	47	44	6	0.00	2	11.40	1	7	1384	483
Bs03-12-1	84	-	–	1	10800.00	84	48	2	0.01	2	10792.44	3	180	75175	34987
Bs03-12-3	48	48	0.00%	67	814.61	49	48	446	0.35	79	814.04	17	167	1605	370
Bs03-12-5	48	48	0.00%	1	17.81	49	48	6	0.03	2	17.72	2	16	1605	598
Bl04-4-1	53	53	0.00%	23	67.24	53	51	202	0.09	37	67.06	13	74	750	419
Bl04-4-3	51	51	0.00%	2	2.77	52	51	11	0.00	3	2.74	1	3	675	389
Bl04-4-5	51	51	0.00%	1	1.83	52	51	6	0.00	2	1.81	1	3	675	401
Bl04-8-1	34	34	0.00%	79	6.21	34	31	508	0.11	89	6.00	10	68	122	15
Bl04-8-3	31	31	0.00%	1	0.16	33	31	8	0.00	3	0.15	2	10	109	40
Bl04-8-5	31	31	0.00%	1	0.09	32	31	6	0.00	2	0.06	1	4	96	36
Bl04-12-1	97	-	–	1	10800.00	97	60	2	0.01	1	10800.00	0	0	96714	54994
Bl04-12-3	61	60	1.67%	422	10800.00	61	60	3253	4.59	523	10800.00	128	1989	2642	352
Bl04-12-5	60	60	0.00%	2	93.23	61	60	11	0.02	3	93.04	2	64	2642	1247
Bs04-4-1	64	52	23.02%	83	10800.00	64	52	436	0.12	89	10800.00	6	131	14157	5390
Bs04-4-3	52	52	0.00%	3	254.40	61	52	20	0.00	6	254.35	3	68	10368	3985
Bs04-4-5	52	52	0.00%	2	55.20	56	52	13	0.01	4	55.15	2	31	6107	2293
Bs04-8-1	64	52	23.02%	138	10800.00	64	52	707	0.13	143	10800.00	6	131	14157	2826
Bs04-8-3	52	52	0.00%	2	230.73	61	52	15	0.03	5	230.64	3	68	10368	2278
Bs04-8-5	52	52	0.00%	1	54.68	56	52	8	0.01	3	54.62	2	31	6107	1309
Bs04-12-1	76	69	10.14%	5	10800.00	76	69	40	0.02	13	10800.00	10	267	45893	6570
Bs04-12-3	69	69	0.00%	1	1708.46	72	69	8	0.03	3	1708.36	2	36	31366	4738
Bs04-12-5	69	69	0.00%	1	1178.33	71	69	6	0.02	2	1178.23	1	20	28469	4300

	br. Y calls	br. K calls	br. K branchings	leaves	nodes left	max depth	backtracks
Bl01-4-1	812	454	62	445	431	37	333
Bl01-4-3	5	4	4	6	0	5	4
Bl01-4-5	0	0	0	1	0	0	0
Bl01-8-1	548	123	53	114	489	49	127
Bl01-8-3	326	111	111	57	0	17	43
Bl01-8-5	6	1	1	1	0	5	1
Bl01-12-1	16	6	6	1	18	11	1
Bl01-12-3	206	20	19	17	232	25	34
Bl01-12-5	1	2	2	1	0	2	0
Bs01-4-1	685	263	129	251	565	81	127
Bs01-4-3	54	4	1	57	0	25	31
Bs01-4-5	0	0	0	1	0	0	0
Bs01-8-1	1318	1687	1022	1327	1014	57	867
Bs01-8-3	0	0	0	1	0	0	0
Bs01-8-5	0	0	0	1	0	0	0
Bs01-12-1	662	92	49	108	604	54	102
Bs01-12-3	0	0	0	1	0	0	0
Bs01-12-5	0	0	0	1	0	0	0
Bl02-4-1	10	10	8	19	0	6	18
Bl02-4-3	0	3	3	1	0	3	0
Bl02-4-5	0	1	1	1	0	1	0
Bl02-8-1	7	3	2	10	0	6	11
Bl02-8-3	0	0	0	1	0	0	0
Bl02-8-5	0	0	0	1	0	0	0
Bl02-12-1	40	8	6	47	0	12	37
Bl02-12-3	0	3	3	1	0	3	0
Bl02-12-5	0	0	0	1	0	0	0
Bs02-4-1	1929	431	308	405	1833	83	308
Bs02-4-3	0	0	0	1	0	0	0
Bs02-4-5	0	0	0	1	0	0	0
Bs02-8-1	0	0	0	1	0	0	0
Bs02-8-3	10	1	1	1	0	5	2
Bs02-8-5	2	1	1	1	0	1	0
Bs02-12-1	1154	301	177	235	1097	80	169
Bs02-12-3	0	0	0	1	0	0	0
Bs02-12-5	0	0	0	1	0	0	0

Table 6: Branch and Price: branching table on the Carbin test set

	br. Y calls	br. K calls	br. K branchings	leaves	nodes left	max depth	backtracks
Bl03-4-1	207	48	24	22	211	18	23
Bl03-4-3	106	17	5	102	0	19	57
Bl03-4-5	0	0	0	1	0	0	0
Bl03-8-1	0	0	0	1	0	0	0
Bl03-8-3	1895	379	208	256	2256	29	129
Bl03-8-5	5	10	10	1	6	6	0
Bl03-12-1	271	26	18	32	259	72	49
Bl03-12-3	565	65	11	69	575	29	62
Bl03-12-5	4	2	2	0	0	0	0
Bs03-4-1	26	6	6	2	32	7	5
Bs03-4-3	532	1560	623	1025	0	38	815
Bs03-4-5	31	8	4	2	0	7	7
Bs03-8-1	113	588	361	475	0	31	364
Bs03-8-3	9	1	1	1	0	5	2
Bs03-8-5	0	0	0	1	0	0	0
Bs03-12-1	0	0	0	1	0	0	0
Bs03-12-3	39	22	22	9	0	14	8
Bs03-12-5	0	0	0	1	0	0	0
Bl04-4-1	5	10	6	12	0	8	11
Bl04-4-3	0	1	1	1	0	1	0
Bl04-4-5	0	0	0	1	0	0	0
Bl04-8-1	20	20	19	40	0	15	30
Bl04-8-3	0	0	0	1	0	0	0
Bl04-8-5	0	0	0	1	0	0	0
Bl04-12-1	0	0	0	1	0	0	0
Bl04-12-3	52	343	224	40	772	61	9
Bl04-12-5	0	1	1	1	0	1	0
Bs04-4-1	78	2	2	3	79	31	15
Bs04-4-3	0	2	2	1	0	2	0
Bs04-4-5	0	1	1	1	0	1	0
Bs04-8-1	127	2	2	9	121	34	28
Bs04-8-3	0	1	1	1	0	1	0
Bs04-8-5	0	0	0	1	0	0	0
Bs04-12-1	0	4	4	1	4	4	0
Bs04-12-3	0	0	0	1	0	0	0
Bs04-12-5	0	0	0	1	0	0	0

Appendix F.

	Instance				Matheuristic			After *BP*		
	nodes	arcs	h	k	LB	best sol	GAP	dual sol	best sol	GAP
g-2-1-1	50	185	1	1	24	32	25.00%	32	32	0.00%
g-2-2-1	50	185	1	1	24	36	33.33%	34	36	5.56%
g-2-2-2	50	185	2	2	24	26	7.69%	26	26	0.00%
g-2-3-1	50	185	1	1	24	35	31.43%	32	35	8.57%
g-2-3-2	50	185	2	2	24	26	7.69%	26	26	0.00%
g-2-3-3	50	185	3	3	24	25	4.00%	25	25	0.00%
g-2-4-1	50	185	1	1	24	31	22.58%	30	31	3.23%
g-2-4-2	50	185	2	2	24	26	7.69%	26	26	0.00%
g-2-5-1	50	185	1	1	24	34	29.41%	27	34	20.59%
g-2-5-2	50	185	2	2	24	28	14.29%	26	28	7.14%
g-2-5-3	50	185	3	3	24	26	7.69%	26	26	0.00%
g-2-5-4	50	185	4	4	24	25	4.00%	25	25	0.00%
g-3-1-1	75	290	2	1	24	37	35.14%	35	37	5.41%
g-3-1-2	75	290	4	2	24	26	7.69%	26	26	0.00%
g-3-2-1	75	290	2	1	24	34	29.41%	24	34	29.14%
g-3-2-2	75	290	4	2	24	27	11.11%	27	27	0.00%
g-3-2-3	75	290	6	3	24	26	7.69%	26	26	0.00%
g-3-2-4	75	290	8	4	24	25	4.00%	25	25	0.00%
g-3-3-1	75	290	2	1	24	28	14.29%	28	28	0.00%
g-3-4-1	75	290	2	1	24	32	25.00%	24	32	25.00%
g-3-4-2	75	290	4	2	24	26	7.69%	24	26	7.69%
g-3-5-1	75	290	2	1	24	39	38.46%	-	39	38.46%
g-3-5-2	75	290	4	2	24	29	17.24%	24	29	17.24%
g-3-5-3	75	290	6	3	24	26	7.69%	24	26	7.69%
g-3-5-4	75	290	8	4	24	25	4.00%	24	25	4.00%
g-4-1-1	100	395	3	1	24	30	20.00%	30	30	0.00%
g-4-1-2	100	395	6	2	24	25	4.00%	25	25	0.00%
g-4-2-1	100	395	3	1	24	42	42.86%	24	42	42.44%
g-4-2-2	100	395	6	2	24	33	27.27%	26	33	21.21%
g-4-2-3	100	395	9	3	24	30	20.00%	26	30	13.33%
g-4-2-4	100	395	12	4	24	28	14.29%	26	28	7.14%
g-4-3-1	100	395	3	1	24	33	27.27%	24	33	27.27%
g-4-3-2	100	395	6	2	24	27	11.11%	24	27	11.11%
g-4-3-3	100	395	9	3	24	25	4.00%	25	25	0.00%
g-4-4-1	100	395	3	1	24	31	22.58%	24	31	22.58%
g-4-5-1	100	395	3	1	24	43	44.19%	-	43	44.19%
g-4-5-2	100	395	6	2	24	30	20.00%	24	30	20.00%
g-4-5-3	100	395	9	3	24	26	7.69%	24	26	7.69%
g-4-5-4	100	395	12	4	24	25	4.00%	24	25	4.00%

Table 7: Branch and Price: experiments on the Grid test set

	Instance				Matheuristic			After BP		
	nodes	arcs	h	k	LB	best sol	GAP	dual sol	best sol	GAP
g-5-1-1	125	500	4	1	24	27	11.11%	27	27	0.00%
g-5-1-2	125	500	8	2	24	25	4.00%	25	25	0.00%
g-5-3-1	125	500	4	1	24	28	14.29%	28	28	0.00%
g-5-3-2	125	500	8	2	24	25	4.00%	25	25	0.00%
g-5-3-3	125	500	12	3	24	25	4.00%	25	25	0.00%
g-5-3-4	125	500	16	4	24	25	4.00%	25	25	0.00%
g-5-4-1	125	500	4	1	24	27	11.11%	24	27	11.11%
g-5-5-1	125	500	4	1	24	27	11.11%	24	27	11.05%
g-5-5-2	125	500	8	2	24	25	4.00%	24	25	4.00%
g-6-1-1	150	605	5	1	24	29	17.24%	28	29	3.45%
g-6-1-2	150	605	10	2	24	25	4.00%	25	25	0.00%
g-6-2-1	150	605	5	1	24	27	11.11%	24	27	9.58%
g-6-2-2	150	605	10	2	24	25	4.00%	25	25	0.00%
g-6-3-1	150	605	5	1	24	28	14.29%	24	28	14.28%
g-6-3-2	150	605	10	2	24	25	4.00%	25	25	0.00%
g-6-4-1	150	605	5	1	24	26	7.69%	24	26	7.69%
g-6-5-1	150	605	5	1	24	28	14.29%	24	28	14.29%
g-6-5-2	150	605	10	2	24	25	4.00%	24	25	4.00%
g-7-1-1	175	710	6	1	24	26	7.69%	26	26	0.00%
g-7-2-1	175	710	6	1	24	27	11.11%	27	27	0.00%
g-7-2-2	175	710	12	2	24	25	4.00%	25	25	0.00%
g-7-4-1	175	710	6	1	24	27	11.11%	-	27	11.11%
g-7-4-2	175	710	12	2	24	25	4.00%	24	25	4.00%
g-7-4-3	175	710	18	3	24	25	4.00%	24	25	4.00%
g-8-1-1	200	815	7	1	24	26	7.69%	26	26	0.00%
g-8-2-1	200	815	7	1	24	27	11.11%	-	27	11.11%
g-8-3-1	200	815	7	1	24	25	4.00%	24	25	4.00%
g-8-4-1	200	815	7	1	24	27	11.11%	-	27	11.11%
g-8-5-1	200	815	7	1	24	27	11.11%	-	27	11.11%
g-8-5-2	200	815	14	2	24	25	4.00%	-	25	4.00%
g-9-2-1	225	920	8	1	24	25	4.00%	25	25	0.00%
g-9-3-1	225	920	8	1	24	27	11.11%	-	27	11.11%
g-9-3-2	225	920	16	2	24	25	4.00%	24	25	4.00%
g-9-4-1	225	920	8	1	24	26	7.69%	-	26	7.69%
g-9-5-1	225	920	8	1	24	26	7.69%	-	26	7.69%
g-10-3-1	250	1025	9	1	24	26	7.69%	-	26	7.69%
g-10-3-2	250	1025	18	2	24	25	4.00%	-	25	4.00%
g-10-4-1	250	1025	9	1	24	27	11.11%	-	27	11.11%
g-10-4-2	250	1025	18	2	24	25	4.00%	-	25	4.00%
g-10-5-1	250	1025	9	1	24	26	7.69%	-	26	7.69%
g-10-5-2	250	1025	18	2	24	25	4.00%	-	25	4.00%

Bibliography

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

[2] Ahuja, R., Magnanti, T., Orlin, J., 2009. Minimum cost flow problem. Springer US, pp. 2095–2108.

[3] Albrecht, C., 2001. Global routing by new approximation algorithms for multicommodity flow. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 20 (5), 622–632.

[4] Aronson, J., 1989. A survey of dynamic network flows. Annals of Operations Research 20 (1), 1–66.

[5] Assad, A. A., 1978. Multicommodity network flows: a survey. Networks 8 (1), 37–91.

[6] Baier, G., 2003. Flows with Path Restriction. Ph.D. thesis, Technische Universität, Berlin.

[7] Baier, G., Köhler, E., Skutella, M., 2005. The k-Splittable Flow Problem. Algorithmica 42 (3), 231–248.

[8] Ball, M., 2011. Heuristics based on mathematical programming. Surveys in Operations Research and Management Science 16 (1), 21–38.

[9] Barnhart, C., Hane, C., Vance, P., 2000. Using Branch-and-Price-and-Cut to Solve Origin-Destination Integer Multicommodity Flow Problems. Operations Research 48 (2), 318–326.

[10] Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations Research 46, 316–329.

[11] Baumann, N., Köhler, E., 2007. Approximating earliest arrival flows with flow-dependent transit times. Discrete applied mathematics 155 (2), 161–171.

[12] Baumann, N., Skutella, M., 2006. Solving evacuation problems efficiently–earliest arrival flows with multiple sources. In: Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on. IEEE, pp. 399–410.

[13] Białoń, P., 2017. A randomized rounding approach to a k-splittable multicommodity flow problem with lower path flow bounds affording solution quality guarantees. Telecommunication Systems 64 (3), 525–542.

[14] Blum, C., Puchinger, J., Raidl, G., Roli, A., 2011. Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing 11 (6), 4135–4151.

[15] Blum, C., Raidl, G., 2016. Hybrid Metaheuristics - Powerful Tools for Optimization. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer.

[16] Boland, N., Dethridge, J., Dumitrescu, I., 2006. Accelerated label setting algorithms for the elementary resource constrained shortest path problem. Operations Research Letters 34 (1), 58–68.

[17] Braun, M., Winter, S., 2009. Ad Hoc Solution of the Multicommodity-Flow-Over-Time Problem. IEEE Transactions on Intelligent Transportation Systems 10 (4), 658–667.

[18] Bsaybes, S., Quilliot, A., Wagler, A. K., 2017. Fleet management for autonomous vehicles using flows in time-expanded networks. Electronic Notes in Discrete Mathematics 62, 255–260.

[19] Burkard, E., Dlaska, K., Klinz, B., 1993. The quickest flow problem. Zeitschrift für Operations Research 37 (1), 31–58.

[20] Calvete, H., del Pozo, L., Iranzo, J., May 2012. Algorithms for the quickest path problem and the reliable quickest path problem. Computational Management Science 9 (2), 255–272.

[21] Caramia, M., Sgalambro, A., 2008. An exact approach for the maximum concurrent k-splittable flow problem. Optimization Letters 2 (2), 251–265.

[22] Caramia, M., Sgalambro, A., 2008. On the approximation of the single source k-splittable flow problem. Journal of Discrete Algorithms 6 (2), 277–289.

[23] Caramia, M., Sgalambro, A., 2010. A fast heuristic algorithm for the maximum concurrent k-splittable flow problem. Optimization Letters 4 (1), 37–55.

[24] Chakrabarti, A., Chekuri, C., Gupta, A., Kumar, A., 2007. Approximation algorithms for the unsplittable flow problem. Algorithmica 47 (1), 53–78.

[25] Chen, Y., 1993. An algorithm for finding the k quickest paths in a network. Computers & Operations Research 20 (1), 59–65.

[26] Chen, Y., Chin, Y., 1990. The Quickest Path Problem. Computers and Operations Research 17 (2), 153–161.

[27] Clímaco, J., Pascoal, M., Craveirinha, J., Captivo, M., 2007. Internet packet routing: Application of a k-quickest path algorithm. European Journal of Operational Research 181 (3), 1045–1054.

[28] Conejo, A. J., Castillo, E., Minguez, R., Garcia-Bertrand, R., 2006. Decomposition techniques in mathematical programming: engineering and science applications. Springer Science & Business Media.

[29] Dantzig, G. B., Wolfe, P., 1960. Decomposition principle for linear programs. Operations Research 8 (1), 101–111.

[30] Di Puglia Pugliese, L., Guerriero, F., 2013. Dynamic Programming Approaches to Solve the Shortest Path Problem with Forbidden Paths. Optimization Methods and Software 28 (2), 221–255.

[31] Dinitz, Y., Garg, N., Goemans, M. X., 1999. On the single-source unsplittable flow problem. Combinatorica 19 (1), 17–41.

[32] Dressler, D., Skutella, M., 2011. An FPTAS for Flows over Time with Aggregate Arc Capacities. In: Jansen, K., Solis-Oba, R. (Eds.), Approximation and Online Algorithms. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 106–117.

[33] Filipe Alvelos, J. M. V. V. d. C., 2003. Comparing branch-and-price algorithms for the unsplittable multicommodity flow problem. Proceedings of the International Network Optimization Conference, 7–12.

[34] Fischer, F., Helmberg, C., 2014. Dynamic graph generation for the shortest path problem in time expanded networks. Mathematical Programming 143 (1), 257–297.

[35] Fleischer, L., Skutella, M., 2002. The Quickest Multicommodity Flow Problem. In: Cook, W. J., Schulz, A. S. (Eds.), Integer Programming and Combinatorial Optimization. Springer Berlin Heidelberg, pp. 36–53.

[36] Fleischer, L., Skutella, M., 2003. Minimum Cost Flows over Time Without Intermediate Storage. Society for Industrial and Applied Mathematics, pp. 66–75.

[37] Fleischer, L., Skutella, M., 2007. Quickest Flows Over Time. SIAM Journal on Computing 36 (6), 1600–1630.

[38] Ford, D., Fulkerson, D., 1962. Flows in Networks. Princeton University Press, Princeton, NJ, USA.

[39] Ford, L., Fulkerson, D., 1958. Constructing Maximal Dynamic Flows from Static Flows. Operations Research 6 (3), 419–433.

[40] Gale, D., 1959. Transient flows in networks. The Michigan Mathematical Journal 6 (1), 59–63.

[41] Gamrath, G., 2010. Generic Branch-Cut-and-Price. Master's thesis, Technische Universität, Berlin.

[42] Gamst, M., 2013. A decomposition based on path sets for the multi-commodity k-splittable maximum flow problem. Dep. of Management Engineering: Technical University of Denmark, Report No. 6.

[43] Gamst, M., 2014. A local search heuristic for the Multi-Commodity k-splittable Maximum Flow Problem. Optimization Letters 8 (3), 919–937.

[44] Gamst, M., Jensen, P., Pisinger, D., Plum, C., 2010. Two- and three-index formulations of the Minimum Cost Multicommodity k-splittable Flow Problem. European Journal of Operational Research 202 (1), 82–89.

[45] Gamst, M., Jensen, P. N., Pisinger, D., Plum, C., 2010. Two- and three-index formulations of the minimum cost multicommodity k-splittable flow problem. European Journal of Operational Research 202 (1), 82–89.

[46] Gamst, M., Petersen, B., 2012. Comparing branch-and-price algorithms for the Multi-Commodity k-splittable Maximum Flow Problem. European Journal of Operational Research 217 (2), 278–286.

[47] Garey, M. R., Johnson, D. S., 1978. Strong NP-Completeness results: Motivation, examples, and implications. J. ACM 25 (3), 499–508.

[48] Garey, M. R., Johnson, D. S., 2002. Computers and intractability. Vol. 29. W.H. Freeman and Company.

[49] Ghiyasvand, M., Ramezanipour, A., 2018. Solving the MCQP, MLT, and MMLT problems and computing weakly and strongly stable quickest paths. Telecommunication Systems 68 (2), 217–230.

[50] Gleixner, A., Eifler, L., Gally, T., Gamrath, G., Gemander, P., Gottwald, R. L., Hendel, G., Hojny, C., Koch, T., Miltenberger, M., Müller, B., Pfetsch, M. E., Puchert, C., Rehfeldt, D., Schlösser, F., Serrano, F., Shinano, Y., Viernickel, J. M., Vigerske, S., Weninger, D., Witt, J. T., Witzig, J., 2017. The scip optimization suite 5.0. Tech. Rep. 17–61, ZIB, Takustr.7, 14195 Berlin.

[51] Grande, E., Nicosia, G., Pacifici, A., V., R., 2018. An exact algorithm for a multicommodity min-cost flow over time problem. Electronic Notes in Discrete Mathematics 64, 125–134.

[52] Groß, M., 2014. Approximation algorithms for complex network flow over time problems. Ph.D. thesis, Technische Universität, Berlin.

[53] Groß, M., Kappmeier, J.-P. W., Schmidt, D. R., Schmidt, M., 2012. Approximating Earliest Arrival Flows in Arbitrary Networks. In: Epstein, L., Ferragina, P. (Eds.), Algorithms – ESA 2012. Springer Berlin Heidelberg, pp. 551–562.

[54] Groß, M., Skutella, M., 2012. Maximum Multicommodity Flows over Time without Intermediate Storage. In: Epstein, L., Ferragina, P. (Eds.), Algorithms – ESA 2012. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 539–550.

[55] Grötschel, M., Lovász, L., Schrijver, A., 2012. Geometric algorithms and combinatorial optimization. Vol. 2. Springer Science & Business Media.

[56] Hajjem, M., Bouziri, H., Talbi, E.-G., 2018. A metaheuristic framework for dynamic network flow problems. In: Recent Developments in Metaheuristics. Springer, pp. 285–304.

[57] Hall, A., Hippler, S., Skutella, M., 2007. Multicommodity Flows over Time: Efficient algorithms and complexity. Theoretical Computer Science 379 (3), 387–404.

[58] Hamacher, H., Tjandra, S., 2001. Mathematical modelling of evacuation problems: A state of art. Tech. Rep. 24, Fraunhofer (ITWM).

[59] Hansen, P., Mladenovic, N., 2001. Variable neighborhood search: Principles and applications. European Journal of Operational Research 130 (3), 449–467.

[60] Hoppe, B., Tardos, É., 1994. Polynomial Time Algorithms for Some Evacuation Problems. In: SODA. Vol. 94. pp. 433–441.

[61] Hoppe, B., Tardos, É., 2000. The Quickest Transshipment Problem. Mathematics of Operations Research 25 (1), 36–62.

[62] Jiao, C., Feng, Q., Bu, W., 2018. Some Complexity Results for the k-Splittable Flow Minimizing Congestion problem. Communications and Network 10.

[63] Jiao, C., Gao, S., Wenguo, Y., Xia, Y., Zhu, M., 2014. A Fast Heuristic Algorithm for Minimizing Congestion in the MPLS networks 07, 294–302.

[64] Jiao, C., Gao, S., Yang, W., 2015. Comparing algorithms for Minimizing Congestion and Cost in the Multi-Commodity k-Splittable Flow. Computer and Information Science 8 (2).

[65] Jiao, C., Yang, W., Gao, S., Xia, Y., Zhu, M., 2014. The k-splittable flow model and a heuristic algorithm for minimizing congestion in the MPLS networks. pp. 1050–1055.

[66] Jones, K. L., Lustig, I. J., Farvolden, J. M., Powell, W. B., 1993. Multicommodity network flows: The impact of formulation on decomposition. Mathematical Programming 62 (1-3), 95–117.

[67] Kappmeier, J.-P., 2015. Generalizations of flows over time with applications in evacuation optimization. Ph.D. thesis, Technische Universität, Berlin.

[68] Katoh, N., Ibaraki, T., Mine, H., 1982. An efficient algorithm for k shortest simple paths. Networks 12 (4), 411–427.

[69] Kleinberg, J. M., 1996. Approximation algorithms for disjoint paths problems. Ph.D. thesis, Massachusetts Institute of Technology.

[70] Kleinberg, J. M., 1996. Single-source unsplittable flow. In: Proceedings of the 37th Annual Symposium on Foundations of Computer Science. pp. 68–77.

[71] Klinz, B., Woeginger, G., 2004. Minimum-cost dynamic flows: The series-parallel case. Networks 43 (3), 153–162.

[72] Koch, R., Skutella, M., Spenke, I., 2005. Approximation and complexity of k-splittable flows. In: International Workshop on Approximation and Online Algorithms. Springer, pp. 244–257.

[73] Koch, R., Spenke, I., 2006. Complexity and approximability of k-splittable flows. Theoretical Computer Science 369 (1), 338–347.

[74] Köhler, E., Möhring, R., Skutella, M., 2009. Traffic Networks and Flows over Time. Springer Berlin Heidelberg, pp. 166–196.

[75] Köhler, E., Möhring, R. H., Spenke, I., 2008. Quickest Flows: A practical model. Technische Universität, Berlin.

[76] Kolliopoulos, S. G., 2018. Disjoint paths and unsplittable flow. Handbook of Approximation Algorithms and Metaheuristics: Contemporary and Emerging Applications 2.

[77] Kolman, P., Scheideler, C., 2006. Improved bounds for the unsplittable flow problem. Journal of Algorithms 61 (1), 20–44.

[78] Korte, B., Vygen, J., 2012. Combinatorial optimization. Vol. 2. Springer.

[79] Kotnyek, B., 2003. An annotated overview of dynamic network flows. Tech. Rep. RR-4936, INRIA.

[80] Lin, M., Jaillet, P., 2015. On the Quickest Flow Problem in Dynamic Networks: A Parametric Min-Cost Flow Approach. In: Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1343–1356.

[81] López−Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stutzle, T., Birattari, M., 2016. The *irace* package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3 (Supplement C), 43–58.

[82] Lübbecke, M. E., Desrosiers, J., 2005. Selected Topics in Column Generation. Operations Research 53 (6), 1007–1023.

[83] Martens, M., 2007. Path-constrained Network Flows. Ph.D. thesis, Technische Universität Berlin.

[84] Martens, M., Skutella, M., 2006. Flows on few paths: Algorithms and lower bounds. Networks 48 (2), 68–76.

[85] Martens, M., Skutella, M., 2006. Length-Bounded and Dynamic k-Splittable Flows. In: Haasis, H.-D., Kopfer, H., Schönberger, J. (Eds.), Operations Research Proceedings 2005. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 297–302.

[86] Martins, E. D. Q. V., Dos Santos, J. L. E., 1997. An Algorithm for the Quickest Path Problem. Operations Research Letters 20 (4), 195–198.

[87] Melchiori, A., Sgalambro, A., 2015. Optimizing Emergency Transportation through Multicommodity Quickest Paths. Transportation Research Procedia 10, 756–765, 18th Euro Working Group on Transportation, EWGT 2015, Delft, The Netherlands.

[88] Melchiori, A., Sgalambro, A., 2018. A matheuristic approach for the Quickest Multicommodity k-Splittable Flow Problem. Computers and Operations Research 92, 111–129.

[89] Melkonian, V., 2007. Flows in Dynamic Networks with Aggregate Arc Capacities. Inf. Process. Lett. 101 (1), 30–35.

[90] Minieka, E., 1973. Maximal, lexicographic, and dynamic network flows. Operations Research 21 (2), 517–527.

[91] Moore, M. H., 1976. On the Fastest Route for Convoy-Type Traffic in Flowrate-Constrained Networks. Transportation Science 10 (2), 113–124.

[92] Müller, D., 2006. Optimizing yield in global routing. In: Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design. ACM, pp. 480–486.

[93] Nagurney, A., Yu, M., Masoumi, A. H., Nagurney, L. S., 2013. Networks against time: Supply chain analytics for perishable products. Springer Science & Business Media.

[94] Park, C.-K., Lee, S., Park, S., 2004. A label-setting algorithm for finding a quickest path. Computers and Operations Research 31 (14), 2405–2418.

[95] Pascoal, M., Captivo, M., Clímaco, J., 2005. An Algorithm for Ranking Quickest Simple Paths. Computers and Opererations Research 32 (3), 509–520.

[96] Pascoal, M., Captivo, M., Clímaco, J., 2006. A comprehensive survey on the quickest path problem. Annals of Operations Research 147 (1), 5–21.

[97] Pascoal, M., Captivo, M., Clímaco, J., 2007. Computational experiments with a lazy version of a k quickest simple path ranking algorithm. TOP 15 (2), 372–382.

[98] Powell, W. B., Jaillet, P., Odoni, A., 1995. Chapter 3 stochastic and dynamic networks and routing. In: Network Routing. Vol. 8 of Handbooks in Operations Research and Management Science. Elsevier, pp. 141 – 295.

[99] Pugliese, L. D. P., Guerriero, F., 2013. A survey of resource constrained shortest path problems: Exact solution approaches. Networks 62 (3), 183–200.

[100] Raayatpanah, M., Ghasvari, H., Ebrahimnejad, A., 2013. Generalized water supply management over time. Middle-East Journal of Scientific Research, 383–388.

[101] Raghavan, P., Tompson, C., 1987. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. Combinatorica 7 (4), 365–374.

[102] Rosen, E., Viswanathan, A., Callon, R., 2001. Multiprotocol Label Switching Architecture. Tech. rep., United States.

[103] Rosen, J., Sun, S.-Z., Xue, G.-L., 1991. Algorithms for the quickest path problem and the enumeration of quickest paths. Computers and Operations Research 18 (6), 579–584.

[104] Ruzika, S., Thiemann, M., 2012. Min−Max quickest path problems. Networks 60 (4), 253–258.

[105] Schlöter, M., Skutella, M., 2017. Fast and Memory-efficient Algorithms for Evacuation Problems. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '17. Society for Industrial and Applied Mathematics, pp. 821–840.

[106] Schmidt, M., Skutella, M., 2014. Earliest arrival flows in networks with multiple sinks. Discrete Applied Mathematics 164, 320–327.

[107] Sedeño-Node, A., Gonzáles-Barrera, J. D., 2014. Fast and fine quickest path algorithm. European Journal of Operational Research 238 (2), 596–606.

[108] Sgalambro, A., 2005. Algorithms for the Maximum Concurrent k-Splittable Flow Problem. Ph.D. thesis, Università di Roma La Sapienza.

[109] Skutella, M., 2009. An Introduction to Network Flows over Time. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 451–482.

[110] Talbi, E.-G., 2016. Combining metaheuristics with mathematical programming, constraint programming and machine learning. Annals of Operations Research 240 (1), 171–215.

[111] Truffot, J., Duhamel, C., 2008. A Branch and Price algorithm for the k-splittable maximum flow problem. Discrete Optimization 5 (3), 629–646.

[112] Truffot, J., Duhamel, C., Mahey, P., 2005. Using Branch-and-Price to solve Multicommodity k-Splittable Flow Problems. In: The Proceedings of 2005 International Network Optimisation Conference. pp. 811–816.

[113] Vanderbeck, F., Wolsey, L. A., 1996. An exact algorithm for IP column generation. Operations Research Letters 19 (4), 151–159.

[114] Villeneuve, D., Desaulniers, G., 2005. The shortest path problem with forbidden paths. European Journal of Operational Research 165 (1), 97–107.

[115] Wang, Y., Wang, Z., 1999. Explicit routing algorithms for Internet traffic engineering. pp. 582–588.

[116] Yen, J., 1971. Finding the k shortest loopless paths in a network. Management Science 17 (11), 712–716.

[117] Zhu, K., Liu, X., 2018. A graph-based approach for proteoform identification and quantification using top-down homogeneous multiplexed tandem mass spectra. BMC Bioinformatics 19 (9), 161.