
Article:
Melchiori, A. and Sgalambro, A. orcid.org/0000-0002-0052-4950 (2020) A branch and price algorithm to solve the quickest multicommodity k-splittable flow problem. European Journal of Operational Research, 282 (3). pp. 846-857. ISSN 0377-2217

https://doi.org/10.1016/j.ejor.2019.10.016

A Branch and Price Algorithm to solve the Quickest Multicommodity k-Splittable Flow Problem

Anna Melchiori[a,c], Antonino Sgalambro[b,c,*]

[a] *Sapienza Università di Roma, Pzz.le Aldo Moro 5, Rome, Italy*
[b] *Sheffield University Management School, Conduit Road, Sheffield, United Kingdom*
[c] *Istituto per le Applicazioni del Calcolo "Mauro Picone", Consiglio Nazionale delle Ricerche, Via dei Taurini 19, Rome, Italy*

Abstract

In the literature on Network Optimization, k-splittable flows were introduced to enhance modeling accuracy in cases where an upper bound on the number of supporting paths for each commodity needs to be imposed, thus extending the suitability of network flow tools for an increased number of practical applications. Such modeling feature has recently been extended to dynamic flows with the introduction of the novel strongly NP-hard Quickest Multicommodity k-splittable Flow Problem ($QMCkFP$). Such a flows over time problem asks for routing and scheduling of each commodity demand through at most k different paths in a dynamic network with arc capacities per time step, while minimizing the time required by the overall process. In this work, we propose the first exact algorithm for solving the $QMCkSFP$. The developed technique falls within the Branch and Price class and is based on original relaxation, pricing and branching procedures. Linearization and variable substitution are used to obtain the relaxation problem from the path-based formulation of the $QMCkSFP$. The pricing problem is modeled as a Shortest Path Problem with Forbidden Paths with additional node-set resources on a time expansion of the original digraph and is solved via a tailored dynamic programming algorithm. Two branching rules are designed for restoring feasibility whenever k-splittable or binary variable domain constraints are violated. The results of an extensive batch of computational experiments conducted on small to medium-size reference instances are presented, showing a highly satisfactory performance of the proposed algorithm. The paper concludes with a discussion on further lines of research.

Keywords: Networks, Flows over time, Quickest flow, k-splittable flow, Branch and Price

*Corresponding author. Sheffield University Management School, Conduit Road, Sheffield (UK), S10 1FL , Phone: + 44 (0) 114 222 3393

Email addresses: a.melchiori@iac.cnr.it (Anna Melchiori), a.sgalambro@sheffield.ac.uk (Antonino Sgalambro)

1. Motivation and background

Despite the renowned impact of static network flow modeling tools in an extremely broad variety of applications, some structural limitations prevent them from providing actionable decisional support in those contexts where monitoring of the system's transitional regime represents a crucial requirement. Indeed, static flows are well fitted to represent average steady flow behavior, whereas they fail to represent the evolution of dynamic phenomena occurring over a considered time horizon. In contrast, dynamic flows, often referred to as flows over time, allow to capture the time dimension in network flow modeling, hence enabling the representation and optimization of an increasing amount of real-world processes whose efficiency nowadays heavily depends on time. This is the case for application fields such as telecommunication networks, emergency management, advanced logistics and production scheduling, see [2, 34].

A relevant family of dynamic optimization problems is represented by the Quickest Flow class, where the completion time for flow transshipment operations is minimized on a capacitated network with transit time labels on arcs and a discretized time horizon. The practical interest in this class of problems is widely recognized in the literature, see for example [14, 39]. Limitations associated with the Quickest Flows as a planning tool include the possible use of a large and unrestricted number of active paths in optimal solutions. This is unrealistic in all those contexts where availability of resources is limited and a tight supervision of the various operations is needed. Therefore, effective practical applications require limitation of the number of support paths to be activated for flows transshipment.

From a methodological point of view, introducing such an additional feature translates into the integration of the well-known concept of k-splittable flows, enabling a transshipment process through at most the prefixed number of paths, within dynamic network flow models. The interest in this setting is motivated by the large variety of applications that share the common requirement of thorough control of both the time spent for flow transshipment and the number and characteristics of the activated support paths. This can be observed, for instance, in distribution planning, where time-efficient goods dispatch has to be achieved by a limited number of fleet vehicles [32], in emergency transportation management, where an unsupervised evacuation can lead to fatal congestion episodes and thus drastically preclude its overall success [39], and in the telecommunication field, where cost-efficient use of the network for quick data packets transmission is crucial [6, 29]. This novel combination of k-splittable and dynamic flows has rarely been addressed in the literature and only recently Melchiori and Sgalambro [40] provided a formal introduction of the Quickest Multicommodity k-splittable Flow Problem ($QMCkSFP$). This strongly NP-hard problem explicitly accounts for a limited number (k) of paths to be allowed in flow routing in a dynamic network,

combining the requirement of a quickest (dynamic) multicommodity flow with restrictions on the number of active paths on each distinct commodity. The authors designed an original $VLNS$-based matheuristic approach that was provably effective in identifying high quality solutions in short computational times for medium to large-size instances of the $QMCkSFP$.

The design of an exact algorithm to identify optimal solutions to the $QMCkSFP$ was so far an open research problem, as the limitations of such resolution approaches to tackling complex and increasing-size instances are well known in the scientific literature. However, the potential availability of exact tools represents a strict requirement in many real-life situations where a precise quality measure is a significant prerequisite for actual and more conscious implementation of the identified solutions. Moreover, for those particular situations where obtaining even a small decrease in the objective function for the $QMCkSFP$ might represent a goal of the utmost relevance, such as in strategic or tactical planning of emergency operations, the need to obtain optimal solutions provides valid and strong motivation for adopting exact methods whilst allowing very large computational times.

With this work we provide a complete answer to such relevant questions by developing the first exact resolution approach to solve the $QMCkSFP$ to optimality. Hence, our research activity is focused on the design, development, implementation and testing of a tailored Branch and Price algorithm which exploits the path-based formulation of the problem and copes with the exponential number of path-related mixed integer variables to identify provably optimal solutions to the considered problem.

The remainder of this section presents the main literature contributions on Quickest Flow optimization problems and static variants of the k-Splittable Flow Problem and sets out the structure of our paper.

1.1. Related results from the literature

The first contribution integrating time into mathematical modelization of network flows can be traced back to the Sixties when Ford and Fulkerson developed the concept of *dynamic flows*, more recently known as *flows over time* [19, 20]. The introduced dynamic digraph structure associates to each arc a *transit time/delay* and a *capacity* arc attributes, expressing the units of time required for the flow to travel through the arc and the maximum inflow that can enter the arc at every time instant, respectively. In the case of a finite discrete time horizon, Ford and Fulkerson showed that it is always possible to model and tackle a general dynamic flow problem 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. Although this novel network allows employment of classical and efficient

algorithms from static flow problems, the resulting algorithms might be pseudo-polynomial due to the possible non-linear dependence of the time horizon in the input size.

Relevant static problems have been extended to the dynamic case by allowing the transshipment to span a fixed time horizon, such as the Maximum Dynamic Flow Problem, the Minimum Cost Dynamic Flow Problem and some of their variants [19, 20, 27, 31]. Moreover, entirely novel network flow problems have emerged in the dynamic environment: the Quickest Flow Problem, the Quickest Path Problem, the Quickest Transshipment Problem and the Earliest Arrival Problem among others, [8, 13, 18, 28, 45]. We refer to Aronson [2], Kotnyek [35] and Skutella [47] for a complete survey of flows over time. Hall et al. [27] conducted a broad study on the multicommodity version of dynamic flow problems, proving their weak NP-hardness even in the case of two commodities and the strong NP-hardness when flow storage is forbidden and only elementary paths are allowed.

In the Quickest Flow Problem (QFP) the aim is to route a given amount of flow between a pair of nodes in a capacitated network as quickly as possible, i.e. minimizing the number of time steps required to complete the demand transshipment process, equivalently referred to as *makespan*. The problem can be solved in strongly polynomial time by integrating the repeated flows technique of Ford and Fulkerson within Megiddo's parametric search [8]. See the work of Lin and Jaillet [36] for a QFP formulation and a cost-scaling algorithm.

A rich variety of contributions focused on the specific case of restricting quickest flow to a single path, the so-called *Quickest Path Problem* (QPP). We refer to the following contributions for models and algorithms for the QPP and some of its variants [9, 13, 25, 39, 42, 44, 46]. The multicommodity version of the problem has been introduced in the work by Melchiori and Sgalambro [39] motivated by a specific application to emergency evacuation management. We refer to Sedeño-Noda and González-Barrera [46] for a recent labeling algorithm for the QPP that has been proved to outperform all of the previously developed approaches.

The related problem of ranking K-quickest (loopless) paths w.r.t. their transmission time has been widely investigated, among others by Chen [12] and Pascoal et al. [41, 42, 43], and applied to the routing of data packets in Internet networks by Clímaco et al. [14].

The concept of bounding the number of support paths in flow routing has been widely addressed in the static environment with the so-called *k-Splittable Flow Problems* ($kSFP$), starting from the work of Baier et al. [3]. Problems in this class explicitly limit to the parameter k the number of paths that each distinct commodity can use for flow routing. In general, their strong NP-hardness has been proved even for the single commodity case, both in directed and undirected graphs and for different constant values of k, see [3]. A comprehensive overview of approximation and complexity results for $kSFPs$ can be found in [32].

The specific case of $k = 1$, namely the Unsplittable Flow Problem (UFP), has been introduced by Kleinberg [30] and its multicommodity version investigated by Barnhart et al. in [4] through development of a Branch and Price and Cut algorithm. The designed branching strategy is based on the notion of *divergence node* for a fractional solution, i.e. the first vertex at which a commodity flow splits into two or more paths, and involves the construction of a binary decisional tree by forbidding the usage of subsets of arcs leaving the divergence node. The pricing problem is modeled and efficiently solved as a Shortest Path Problem (SPP) for each single commodity by setting a very high value on the costs of the forbidden arcs.

Truffot et al. presented Branch and Price algorithms for the Maximum $kSFP$ [48, 49], where the amount of transshipped flow is to be maximized. The problem, formulated with path-flow and path-design variables for each of the (at most) k path positions, is linearized and simplified by focusing on the solution space where coupling constraints are saturated. The designed pricing problem looks for the shortest path with highest capacity and is solved by means of a polynomial algorithm. A first branching strategy works on the arc-flow space and resembles that developed by Barnhart et al. [4], whereas a second imposes a subset of path positions for the possible use of a certain arc. Further Branch and Price approaches for the Multicommodity Maximum $kSFP$ and the minimum cost version, where the total cost of used arcs is to be minimized, have been discussed and compared in [23] and [24]. In particular, novel branching strategies have been introduced for a 2-index formulation of the problem: a first rule forbids the transshipment on subpaths emanating from the divergence node, while a second, proved to outperform all others, forbids the usage of certain paths while forcing the activation of others. Their computational results present better performances of the complete toolbox developed for the 2-index formulation. Finally, a Branch and Price approach based on a different Dantzig-Wolfe decomposition can be found in [22]. Specifically, each column in the master problem represents a collection of at most k paths carrying a given amount of flow and the pricing problem, resulting in the NP-hard single-commodity Maximum $kSFP$, is solved through heuristic method and an exact algorithm. The branching rule forces and forbids different subpaths and the cuts so generated are accounted for in both resolution methods with slight modifications. The method is competitive but presents some scaling issues due to the complexity of the pricing problem.

Caramia and Sgalambro dealt with the multicommodity Maximum Concurrent $kSFP$, where the aim is to maximize the routable demand fraction. They presented a 3-index arc-flow formulation and a Branch and Bound algorithm. It performs binary branchings on k-splittable-related variables and fathoming rules are designed to speed up the process and to reduce the size of the decisional tree [10]. To address the same problem, a fast two stage heuristic algorithm has been designed in

[11], whereby flow routing is initialized through a tailored augmenting path algorithm and then refined by a local search routine.

The first contribution integrating k-splittable flow tools in dynamic networks flows involved a $(3 + 2\sqrt{2})$-approximation algorithm for the single commodity Dynamic k-Splittable Flow Problem with a continuous time parameter [38]. Recently, Melchiori and Sgalambro [40] formally introduced the Quickest Multicommodity k-splittable Flow Problem ($QMCkSFP$), presenting a Mixed-Integer Linear Programming path-based formulation and proving its strong NP-hardness. This entailed developing an original resolution matheuristic algorithm, by hybridization of a Very Large-scale Neighborhood Search metaheuristic strategy using a Mathematical Programming-based technique.

1.2. Contribution of this paper

In this paper we focus on the Quickest Multicommodity k-splittable Flow Problem ($QMCkFP$) as presented in the work by Melchiori and Sgalambro [40]. In more detail, we consider a discretized time horizon, time- and flow-independent arc attributes, and require transshipment operations without flow storage at intermediate nodes and only through elementary paths, i.e. paths such that each node is visited at most once. The main contribution of our research lies in the design of the first exact algorithm for the resolution to optimality of the $QMCkSFP$. The proposed strategy exploits the path-based formulation of the problem provided in [40] and is based on the Branch and Price paradigm to cope with the exponential number of path-related mixed integer variables.

The work is organized as follows: Section 2 presents the path-based formulation of the $QMCkSFP$; Section 3 concentrates on description of the algorithm, starting from the procedure for constructing the so-called Restricted Relaxed Master Problem ($RRMP$), which is obtained by linearizing some binary variables and substituting those related to k-splittable path restrictions, see Subsection 3.1. Subsection 3.2 presents the algorithm selected from the literature for initializing the pool of variables employed by the Branch and Price algorithm. Subsection 3.3 discusses the Column Generation procedure for identification of promising new variables, i.e. pairs (path,departure time), on a time expansion of the original dynamic digraph. The next section presents the branching rules designed for restoring feasibility. An original branching rule, 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. In Subsection 3.5 the pricing oracle is presented, modeled for each commodity as a Shortest Path Problem with Forbidden Paths ($SPPFP$) and with an additional resource for each node introduced to secure the generation of elementary paths. It is solved in a time extension of the original digraph via a tailored dynamic programming algorithm

that tracks the consumption of the entire set of limited resources. In Section 3.6, we propose an enhanced version of this strategy that dynamically includes node-set and forbidden path-related resources only when needed to restore feasibility. Finally, Section 4 presents discussion of the computational experience. Subsection 4.1 focuses on small to medium-size instances to verify the correctness of the strategy and provide measures of quality of the solutions identified by both of the versions of the algorithm, while Subsection 4.2 explains the use of our exact algorithm on a set of medium to large-size instances to assess quality of the solutions provided by the matheuristic approach introduced in [40].

2. Problem definition and formulation

Consider a *dynamic flow network* $\mathcal{D} = (\mathcal{V}, \mathcal{A}, \mathcal{T})$, with $\mathcal{V}$ as the set of nodes, $\mathcal{A}$ the set of directed arcs, and $\mathcal{T} = \{0, 1, \ldots, T\}$ the 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 inflow that can concurrently enter the arc from its tail at a given time instant, and a non-negative *travel time/delay* λ_{ij}, specifying the number of time steps required to traverse the arc from its tail to the head. A set $\mathcal{H}$ of commodities is given, each identified by an amount of demand/population σ_h to be routed from a designated source node o_h to a destination node d_h. All elementary paths are collected in the $\mathcal{P}_h$ set for each commodity h. In this setting, the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$) asks for a multicommodity transshipment using at most k different elementary paths for each commodity while respecting shared arc capacities and minimizing the makespan of the process, i.e. the number of time steps required to accomplish the overall multicommodity routing, assuming that flow storage is not allowed. The three-index path-based $MILP$ formulation for the $QMCkSFP$ as presented in the work of [40] follows, with the complete notation collected in the box.

Notation:

$\mathcal{V}$ set of nodes,

$\mathcal{A}$ set of arcs,

$\mathcal{T}$ set of considered time intervals,

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) source, destination, demand/population of commodity h,

$\mathcal{P}_h$ set of available paths for commodity h,

$u_p = \min_{(ij)\in p} c_{ij}$ bottleneck of path p,

δ_{ij}^p binary coefficient is 1 iff arc (i,j) is traversed by path p,

$l_p = \sum_{(ij)\in\mathcal{A}} \delta_{ij}^p \lambda_{ij}$ travel time of path p,

t_i^p time required to reach node i following path p,

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

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

y_t^h binary variable is 1 iff some flow of commodity h arrives at destination at time t,

z_p^h binary variable is 1 iff path p is chosen by commodity h.

Fractional variables x_{pt}^h are employed to track the release of flows at each point in time for each commodity and path. The identification of the makespan is achieved through dedicated binary variables y_t^h, recording arrival times at destination of each routed commodity demand. Path limitations are implemented by z_p^h variables that identify all activated paths for each commodity. For the sake of simplicity, whenever $t + l_p > T$ for a given path p and time instant t, we set the related x_{pt}^h variable to zero as no unit of flow leaving the origin at time t through path p would arrive at destination within the observed time horizon.

$$\min \zeta \tag{1}$$

$$ty_t^h \leq \zeta \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{2}$$

$$\sum_{p \in \mathcal{P}_h} x^h_{p(t-l_p)} \leq C_{ht} y_t^h \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{3}$$

$$\sum_{p \in \mathcal{P}_h} z_p^h \leq k \qquad \forall h \in \mathcal{H}. \tag{4}$$

$$\sum_{p \in \mathcal{P}_h} \sum_{t \in \mathcal{T}} x^h_{pt} = \sigma_h \qquad \forall h \in \mathcal{H}. \tag{5}$$

$$\sum_{h \in \mathcal{H}} \sum_{p \in \mathcal{P}_h} \delta^p_{ij} x^h_{p(t-t_i^p)} \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}. \tag{6}$$

$$x^h_{pt} \leq u_p z_p^h \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h, t \in \mathcal{T}. \tag{7}$$

$$y_t^h \in \{0,1\} \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{8}$$

$$z_p^h \in \{0,1\} \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h. \tag{9}$$

$$x^h_{pt} \geq 0 \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h, t \in \mathcal{T}. \tag{10}$$

$$\zeta \geq 0 \tag{11}$$

The objective function minimizes the makespan of the process, represented by the ζ variable, by imposing that transshipment of the commodity demands is to be completed as quickly as possible. Each arrival time at destination, corresponding to an activated y variable, is imposed as a lower bound to the makespan by Constraints (2). For each commodity, Constraints (3) relate flow support variables x with time-arrival variables y: whenever no units of flow are detected to reach the destination at time t, a zero flow leaving any path p at time $t - l_p$ is imposed. Parameter C_{ht} is adopted here to enhance the constraints' tightness, representing the maximum amount of flow of commodity h that can arrive at destination at a given time t. Constraints (4), namely k-splittable constraints, bound to k the maximum number of usable paths for each commodity, while the amounts of commodity demands to be transshipped within the time horizon are accounted for by Constraints (5). Arc capacities at each point in time are stated by Constraints (6) and finally, bottleneck Constraints (7) force the amount of flow on a given path to be zero at every time instant if the path is not activated for flow transshipment. Proof of the strong NP-hardness of the $QMCkSFP$ can be found in [40], where a reduction from the Minimum Cost kSFP is proposed.

3. An exact algorithmic approach

Based on integrating Column Generation and Branch and Bound in a unique framework, Branch and Price is particularly recommended for tackling $MILP$ problems that present a huge number of variables, see [1, 5, 37, 50]. Indeed, working on restricted and relaxed versions of the original problem where only certified good quality variables are considered, it allows for an implicit and efficient exploration of the space of feasible solutions, to produce either a provably optimal solution or proof of infeasibility. Note that this is essential in the case of solving the $QMCkSFP$, since the fractional x and binary z variables 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. Explicit enumeration of all these variables, and thus of all paths available for flow transshipment, would be prohibitive for real-size 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 provide full details of each procedure designed in the tailored Branch and Price approach to solve the $QMCkSFP$.

3.1. 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 designed relaxation method first linearizes all integer variables and then replaces bottleneck constraints (7) with the following feasible constraints:

$$\sum_{t \in \mathcal{T}} x_{pt}^h \leq \sigma_h z_p^h \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h^{'} \tag{12}$$

obtained from (7) through an aggregation over time and by employing the commodity demand in the right-hand side as a valid estimation of the maximum amount of flow that could be sent through any path during the overall time horizon. The resulting problem is known as the Relaxed Restricted Master Problem ($RRMP$).

We now prove that the following Lemma holds.

Lemma 3.1. *There exists an optimal solution to the RRMP saturating Constraints (12).*

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

$$0 \leq \bar{z}_p^h := \frac{\sum_{t \in \mathcal{T}} \hat{x}_{pt}^h}{\sigma_h} \quad \forall h \in \mathcal{H}, p \in \mathcal{P}_h^{'}. \tag{13}$$

We must prove that the solution $(\hat{\zeta}, \hat{x}_{pt}^h, \hat{y}_t^h, \bar{z}_p^h)$ is also optimal for the $RRMP$. Note that its feasibility simply follows from constraints (12) i.e.:

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

Moreover, domain constraints $\bar{z}_p^h \leq 1 \; \forall h \in \mathcal{H}, p \in \mathcal{P}_h^{'}$ are implied by population constraints (5). Finally, the optimality follows from the observation that both solutions have the same objective function value. □

Lemma (3.1) allows for a substantial reduction of the number of variables and thus a consequent 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 occurrences with their definition provided in (13). Moreover, in this new variable space, the k-splittable constraints are always satisfied and can therefore be excluded from the formulation. A similar reduction based on the saturation of coupling constraints has been designed by Truffot et al. [48] for the static k-splittable Maximum Flow Problem. The final formulation of the $RRMP$ results in:

$$\min \zeta \tag{14}$$

$$t y_t^h \leq \zeta \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{15}$$

$$\sum_{p \in \mathcal{P}_h^{'}} x_{p(t-l_p)}^h \leq C_{ht} y_t^h \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \quad (\pi_{ht}^1 \leq 0) \tag{16}$$

$$\sum_{p \in \mathcal{P}_h^{'}} \sum_{t \in \mathcal{T}} x_{pt}^h = \sigma_h \qquad \forall h \in \mathcal{H}. \quad (\pi_h^2) \tag{17}$$

$$\sum_{h \in \mathcal{H}} \sum_{p \in \mathcal{P}_h^{'}} \delta_{ij}^p x_{p(t-t_i^p)}^h \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}, t \in \mathcal{T}. \quad (\pi_{ijt}^3 \leq 0) \tag{18}$$

$$y_t^h \in [0,1] \qquad \forall h \in \mathcal{H}, t \in \mathcal{T}. \tag{19}$$

$$x_{pt}^h \geq 0 \qquad \forall h \in \mathcal{H}, p \in \mathcal{P}_h^{'}, t \in \mathcal{T}. \tag{20}$$

$$\zeta \geq 0 \tag{21}$$

Note that the $RRMP$ corresponds to the linear relaxation of the free-flow version of the problem where each commodity flow is unrestricted during the transshipment. Dual variables, π_{ht}^1 and

π^3_{ijt} non-positive and π^2_h unrestricted in sign, are associated to Constraints (16), (18) and (17), respectively.

3.2. 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 select the best k o_h-d_h paths w.r.t. the transmission time as they would likely lead to an overall quickest initial solution. This is done by adopting the lazy version of Chen's approach as presented in Pascoal et al. for ranking the best k quickest loopless paths [42, 43], together with Yen's algorithm [52]. The method keeps in memory an ordered list of shortest loopless paths, each of which is 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 same approach has been successfully used in [40] for generating a pool of good and promising paths to be used throughout the matheuristic approach to obtain a good quality starting feasible solution. The columns of the initial $RRMP$ are then accordingly populated by including all path-independent y variables and those fractional variables x^h_{pt} associated with each identified path p and repeated for each discrete time step $t \in \mathcal{T}$.

3.3. The Pricing Problem

The pricing problem is in charge of identifying new fractional x^h_{pt} variables to be added to the current $RRMP$ that present non-negative reduced costs computed as follows:

$$\hat{c}(x^h_{pt}) := -\Delta_{(t+l_p \leq T)}\pi^1_{h(t+l_p)} - \pi^2_h - \Delta_{(t+t^p_i \leq T)} \sum_{(i,j)\in\mathcal{A}} \delta^p_{ij}\pi^3_{ij(t+t^p_i)}.$$

The identification of the best candidate variable is done separately for each commodity by minimizing the reduced costs over all o_h-d_h elementary paths for the commodity h and all departure instants within the considered time horizon, i.e. $\hat{c}^h_{\bar{p}\bar{t}} := \min_{p,t} \hat{c}(x^h_{pt})\ \forall h \in \mathcal{H}$. Therefore, if the best identified pair $(\bar{p}, \bar{t})$ has $\hat{c}^h_{\bar{p}\bar{t}} < 0$ the current solution is not optimal for the $RRMP$ and the columns related to variables $x^h_{\bar{p}t}\ \forall t \in \mathcal{T}$ are included in the $RRMP$. In contrast, if $\hat{c}^h_{\bar{p}\bar{t}} \geq 0$ 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 not met, the branching phase has to be performed.

In order to model the pricing problem we apply a modified version of the time expansion procedure designed by Ford and Fulkerson [19, 20]. Their method enables the translation of flows over time into static problems, formulated in a novel Time-Expanded Network (TEN) obtained by decoupling the time parameter from the original dynamic digraph. In our case, the considered TEN, indicated

as $\mathcal{D}_\mathcal{T} = (\mathcal{V}_\mathcal{T}, \mathcal{A}_\mathcal{T})$, is composed of the set of original nodes replicated at each discrete time step, $\mathcal{V}_\mathcal{T} = \{i_t \,|\, i \in \mathcal{V}, t \in \mathcal{T}\}$, and the set of the original arcs linking copies of nodes according to their travel time, $\mathcal{A}_\mathcal{T} = \{(i,j)_t \mid (i,j) \in \mathcal{A},\ t + \lambda_{ij} \leq T\}$. We furthermore introduce an additional super source node o' and a super sink node d' and we connect them through outgoing arcs directed to each time-replica of the origin, for sake 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. Note that a cycle in the original dynamic digraph reaching node i at times t' and t'', $t'' \geq t'$, corresponds in the TEN to a subpath starting at node $i_{t'}$ and finishing at node $i_{t''}$. The pricing problem in this extended TEN graph, indicated as $\mathcal{D}'_\mathcal{T} = (\mathcal{V}'_\mathcal{T}, \mathcal{A}'_\mathcal{T})$, results thus in an extended version of the o'-d' Shortest Path Problem (SPP) where the length associated to arc $(i,j)_t \in \mathcal{A}'_\mathcal{T}$ equals:

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

and binary indicator $\gamma^j_{d_h}$ is one if $j = d_h$, zero otherwise, $b_{ijt} = 0\ \forall (i,j)_t \in \mathcal{A}'_\mathcal{T} \setminus \mathcal{A}_\mathcal{T}$.
The explicit model of the pricing problem follows, with binary variables being $\omega_{ijt} = 1$ if the commodity traverses arc $(i,j)_t$, zero otherwise.

$$\min \sum_{(i,j)_t \in \mathcal{A}'_\mathcal{T}} b_{ijt}\omega_{ijt} \tag{22}$$

$$\sum_{t \in \mathcal{T}} \omega_{o' o_h t} = 1 \tag{23}$$

$$\sum_{t \in \mathcal{T}} \omega_{d_h d' t} = 1 \tag{24}$$

$$\sum_{(j,i)_{\bar{t}} \in \mathcal{A}'_\mathcal{T} \,:\, t = \bar{t} + \lambda_{ji}} \omega_{ji\bar{t}} - \sum_{(i,j)_t \in \mathcal{A}'_\mathcal{T}} \omega_{ijt} = 0 \qquad \forall i_t \in \mathcal{V}_T. \tag{25}$$

$$\sum_{t \in \mathcal{T}} \sum_{(i,j)_t \in \mathcal{A}'_\mathcal{T}} \omega_{ijt} \leq 1 \qquad \forall i \in \mathcal{V}. \tag{26}$$

$$\omega_{ijt} \in \{0,1\} \qquad \forall (i,j)_t \in \mathcal{A}'_\mathcal{T}. \tag{27}$$

Constraints (23) and (24) impose the origin of the path at node o' and the destination at node d', respectively, while Constraints (25) 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. Finally Constraints (26) are required to avoid to traverse multiple time replica of the same node and thus to ensure the identification of an elementary path for the original dynamic problem. The optimal solution to this problem identifies a pair $(\bar{p}, \bar{t})$ where $\bar{p}$ is defined by the sequence of selected arcs and the departure time $\bar{t}$ 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 restore the feasibility, see [21]. 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. As it will be explained in detail in the next subsections, the presented linear programming formulation cannot itself be adopted straightly 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. A dynamic programming approach is instead designed and developed to solve an ad-hoc variant of the SPP that implicitly accounts for the above mentioned branching decisions.

3.4. Branching Strategy

The branching step occurs whenever the optimal solution to the current $RRMP$ violates the original problem's constraints. Recall that the $RRMP$ does not 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 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 at least one commodity sending a 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 commodities, say it is h, identifies $k+1$ quickest paths among the used ones, i.e. those $k+1$ activated paths with minimum transmission times, and collects them in the ordered set $\mathcal{P}_h''$. Then, it generates child nodes $N_j,\ j = 1, \ldots, k+1$ as follows: in a given child node N_j it imposes a branching constraint that forbids the usage over time of the j-th path and a second one that forces routing of the 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 when computing reduced costs. However, note that dual variables related to forced paths can be excluded from this computation while still ensuring correct implementation of the column generation procedure. The same does not apply to local forbidden paths as the SPP formulation provided in Subsection 3.3 might not be able to avoid their repeated generation. Subsection 3.5 presents the technique designed to deal with this issue.

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. Moreover, we manually adjust the integrality of y's in the following cases: if $\hat{y}_h^t t < t \leq \lceil \hat{\zeta} \rceil$, i.e. rounding up the time-arrival variable would generate an equivalent optimal solution, we simply set this as $\hat{y}_t^h = 1$.

3.5. A tailored 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 to account for local forbidden paths during Column Generation. To this aim, we re-model the pricing problem in the TEN digraph as an extended version of the o'-d' Shortest Path Problem with Forbidden Paths ($SPPFP$), a particular variant of the well-known Resource-Constrained SPP, that asks for the identification of the minimum length path that is not part of a given set of forbidden paths, see [16, 17, 51]. 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. Moreover, to avoid the identification of subpaths corresponding to cycles in the dynamic digraph, we introduce an additional limited resource for each group of time replica of the same node, namely a node-set resource. The finite availability of such resources ensures that the optimal solution, if one exists, encompasses an elementary path which is not in the forbidden list at any time. We refer the reader to the work by Pugliese and Guerriero [15] for the original resolution technique designed for the $SPPFP$ and Kohl [33] for similar implementations of node-related resources applied to the Elementary Resource-Constrained SPP. Differences with these approaches lie on the specific dynamic structure of our problem that requires to track flow, and thus consumption of resources, at each time instant. We now provide formal details of the developed strategy; a pseudocode is provided in Algorithm 1.

Algorithm 1 Dynamic programming-based approach

1: **Initialization**
$s_{o'}(q_1) = (0,0),\ L = \{s_{o'}(q_1)\},\ \mathcal{S}_{o'} = \{s_{o'}(q_1)\},\ \mathcal{S}_{i_t} = \emptyset\ \ \forall i_t \in \mathcal{V}'_{\mathcal{T}} \setminus \{o'\}$;

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 $V^l_{i_t} = 1$: **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;

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 $P = \left[P^1, P^2, \ldots, P^{|\mathcal{F}|}\right]$. Given a subpath q from the super source o' to a node i_t in the TEN, we denote by $U_{i_t}(q) = \left[U^1_{i_t}(q), \ldots, U^{|\mathcal{F}|}_{i_t}(q)\right]$ the vector of path-related 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 is j with position index $l \in \{1, \ldots, n\}$ in $\mathcal{V}$, we introduce a *node-set resource* with limit N^l equal to one. Whenever the node is traversed at a given time by subpath q from o' to i_t, the resource consumption vector $V_{i_t}(q) = \left[V^1_{i_t}(q), \ldots, V^{|\mathcal{V}|}_{i_t}(q)\right]$ 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 considered resources is thus $R = \left[P, N\right] = \left[n(p_1) - 2, \ldots, n(p_{|\mathcal{F}|}) - 2, 1, \ldots, 1\right]$. 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. We collect all the states associated to paths from o' to node i_t w 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*

introduced in [15] to store only promising states to our case as follows: given $s_{i_t}(q_1), s_{i_t}(q_2) \in \mathcal{S}_{i_t}$, we say that the first *dominates* the second iff $b_{i_t}(q_1) \leq b_{i_t}(q_2)$ and $W_{i_t}(q_1) \leq W_{i_t}(q_2)$, 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 *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 candidate states initialized with state $s_{o'}(q_1) = (0, 0)$, 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 $k = 1, \ldots, |\mathcal{F}|$ and $l = 1, \ldots, |\mathcal{V}|$. The novel state is added to $\mathcal{S}_{j_{t'}}$ iff it satisfies the path-related resource limits and is non dominated. 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, returning the least cost state in the set $\mathcal{S}_{d'}$ related to the super sink node and the reconstructed o'-d' path.

3.6. An incremental approach to enhance the Pricing routine

In this section, we describe an enhanced pricing routine based on the consideration that the inclusion of the entire set of limited resources in the dynamic programming procedure, besides provoking in general a substantial increase in the computational effort, is redundant in all those cases where the potential solutions do not include any forbidden path and cycle over time. Building on this intuition, we introduce an incremental approach based on a state-space relaxation procedure that iteratively executes the proposed dynamic programming algorithm while taking into account only a subset of limited resources. In particular, as a first stage, none of the forbidden path and node-set resources are considered, meaning that a simple SPP is solved in the TEN digraph for each commodity. Whenever the identified path doesn't fulfill the desired original requirements, i.e. it matches a

forbidden path at a certain time step or it embeds a cycle over time, the dynamic programming algorithm is re-executed tracking the consumption of the violated resource during the generation and extension of the subpath labels. In particular, in the case of several nodes traversed multiple times over the time horizon, we select the one with highest multiplicity. In the extension procedure the dominance rule is accordingly restricted to the considered subset of resources. We refer to this approach as the *Enhanced Pricing routine* while to the original one described in the previous section as the *Default Pricing routine*. See the work of Boland [7] for a state-space augmenting algorithm designed to incrementally include limited resources for the Elementary Resource-Constrained SPP.

4. Computational Experience

This section presents thorough discussion of the computational experience performed on the designed Branch and Price algorithm to solve instances of the $QMCkSFP$. In Subsection 4.1 we perform a comparison between the Default and the Enhanced pricing routines on a collection of small to medium-size instances extracted from the Carbin testbed (see [23] for the original testbed in the context of static k-splittable flows) and adapted here to fit the dynamic flow setting. Subsection 4.2 focuses on the use of the Branch and Price approach as a tool to assess the quality of the solutions obtained by a $VLNS$-based matheuristic on a set of medium to large-size Grid instances as presented in [40]. To this aim, we use the same testbed with our Branch and Price algorithm, feeding it with results of the matheuristic as starting solutions, then measuring and comparing the residual optimality GAP, computed as the difference between the best incumbent and the lower bound, divided by the lower bound itself.

Our Branch and Price approach has been implemented in the C++ language and the experiments conducted using the $SCIP$ Optimization Suite v5.0.0 [26] in its default setting on a 64bit Intel Xeon CPU at 2.80GHz with 64GB memory, running Ubuntu 14.04.2. In both the experiments the k paths with lowest transmission time for each commodity have been provided as an initial set of columns, as described in Subsection 3.2. Following the results of a preliminary batch of tuning experiments, binary branching on the time-arrival variables has been applied as a first branching strategy before that to restore the k-splittable feasibility.

4.1. Performance Analysis of the Branch and Price Algorithm

This experiment tests performances of our exact algorithm on a collection of Carbin instances during three hours of computational time. The test set comprised of 8 instances with 32 nodes and 96 arcs, each of which was tested with three different levels of splittable 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

Table 1: Average results obtained on the Carbin test set with Default and Enhanced pricing routines.

	Default Pricing	Enhanced Pricing
Optimal (%)	68.06	70.83
Suboptimal (%)	26.39	29.27
Not identified (%)	5.55	0.00
GAP (%)	8.12	9.27 (*5.63)
CPU time (s)	3744.22	3206.94
Nodes	273.82	2183.67
LP calls	2186.64	1163.5
LP time (s)	1.17	4.32
Pr. calls	317.72	2224.08
Pr. time (s)	3737.74	3053.68
Pr. paths	17.94	18.86
Pr. added vars	321.56	354.88
Max labels	7747.33	1652.19
Mean labels	2660.51	43.43
Br. Y branchings	164.88	918.74
Br. K branchings	48.99	316.83
Node-set it	-	243.18
Forbidden path it	-	1555.44

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, namely the Quickest Multicommodity Flow Problem ($QMCFP$), where no bound is imposed on the number of active paths. We refer to [40] for a detailed description of the technique used for its resolution. Moreover, for each instance we consider a tailored time horizon for flow transshipment to contain the impact of the time dimension on the Branch and Price, especially in its pricing routine. The time horizon of a certain instance is defined by computing a feasible solution through the matheuristic approach described in [40], whose initialization phase provides a valid upper bound (UB) on the instance optimum.

While analyzing the computational results at different aggregation levels, we adopt the following set of key performance indicators, which are utilized in Tables 1, 2 and 3 as row labels. In particular, the *Optimal* entry refers to the percentage of instances that have been solved to optimality within the time limit, *Suboptimal* to the percentage of times our method stopped with a finite but strictly non-zero optimality GAP, and *Not identified* to the percentage of instances for which at least one among the primal and the dual bounds was not identified. The average optimality GAP is presented in row GAP while the average computational time expressed in seconds is reported in *CPU Time (s)* and the average number of nodes explored during the process in row *Nodes*. The next

values present aggregated performance indicators w.r.t. the execution of the LP solver, the pricing procedure, the branching rules, where $Br.K$ refers to the branching strategy on the k-splittable constraints and $Br.Y$ to the one on the time-arrival variables. Finally, the last bunch of rows report some statistics specifically focused on the performances of the Enhanced Pricer, showing the average number of times it identified a path with loops over time (row *Node-set it*), or a forbidden path (row *Forbidden path it*). For a detailed description of the table entries, we refer to Appendix A where disaggregated results on the entire testbed are presented.

Analyzing aggregated results. Table 1 presents the above described key performance indicators averaged on the entire Carbin testbed and compared on the basis of the Default and Enhanced versions of the pricing routine. Column *Default* shows overall good performances of the developed algorithm, with a number of instances solved to optimality equal to 68.06%, an average GAP across the whole testbed of 8% and an average computational time of around 3700 seconds. On 5.55% of the cases, corresponding to four instances, the Default version 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 A, these situations correspond to the longest time horizons. Therefore, the behavior can be attributed to a pricing problem difficulty 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 turned out to be one order of magnitude higher than the average number for the total test set. Timing values show that the strategy expended most of the computational time on performing the pricer step, while the resolution of the LPs took a few seconds on 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.

The second column relates to the use of the Enhanced Pricer, and provides wide evidence of a faster and more efficient exploration of the feasible region with respect to the Default Pricer. This is confirmed by a general improvement in the algorithm's performance, with an increased number of instances solved to optimality, a reduced average computational time and a number of explored nodes 10 times higher. The moderate rise in the average GAP is due to a finite optimality GAP available for the entire testbed, which is not the case for the Default setting, and denotes the ability of the Enhanced pricer to cope with hard instances. When the mean GAP is computed only on the reduced subset of instances tackled by the Default setting, the Enhanced setting prevails by almost 3%. On the average, the time spent for the pricing routine is similar for the two settings, albeit the Enhanced setting permits more iterations of Column Generation, thanks to a considerable decrease

Table 2: Average results on the Carbin test set aggregated by the flow split parameter

	Default Pricer			Enhanced Pricer		
	1	3	5	1	3	5
Optimal(%)	25.00	83.33	95.83	25.00	87.50	100.00
Suboptimal(%)	58.33	16.67	4.17	75.00	12.50	0.00
Not identified (%)	16.67	0.00	0.00	0.00	0.00	0.00
GAP (%)	23.86	2.67	0.46	25.87(*19.14)	1.95	0.00
CPU time (s)	8352.23	2352.90	527.55	7925.88	1691.29	3.66
Nodes	511.63	305.96	3.88	6012.38	534.88	3.75
LP calls	3881.29	2612.92	65.71	30989.54	3891.25	25.71
LP time (s)	1.98	1.43	0.10	10.41	2.53	0.02
Pr. calls	617.13	329.96	6.08	6092.50	573.75	6.00
Pr. time (s)	8342.47	2349.34	521.41	7470.30	1687.13	3.61
Pr. paths	33.42	17.83	2.58	33.42	20.54	2.63
Pr. added vars	692.46	235.33	36.88	773.75	256.50	34.38
Max labels	17365.67	3320.13	2556.21	4497.38	379.21	80.00
Mean labels	6606.92	708.29	666.33	127.25	1.04	2.00
Br. Y branchings	334.29	158.29	2.04	2491.79	262.42	2.00
Br. K branchings	94.33	51.67	0.96	633.92	312.08	2.25
Node-set it	-	-	-	606.25	123.08	0.21
Forbidden path it	-	-	-	3133.75	1529.50	3.08

Table 3: Average results on the Carbin test set aggregated by number of commodities

	Default Pricer			Enhanced Pricer		
	4	8	12	4	8	12
Optimal (%)	75.00	70.83	58.33	75.00	75.00	62.50
Suboptimal (%)	25.00	20.83	33.33	25.00	25.00	37.50
Not identified (%)	0.00	8.33	8.33	0.00	0.00	0.00
GAP (%)	9.61	4.31	10.31	6.26	8.88(*2.95)	12.68(*6.18)
Time (s)	3054.13	3507.58	4670.96	2913.81	2599.39	4107.63
Nodes	353.54	290.46	177.46	1443.67	3859.17	1248.17
LP calls	2612.08	2711.21	1236.63	8482.46	19666.71	6757.33
LP time (s)	1.16	1.43	0.93	4.52	4.37	4.08
Pr. calls	377.75	374.75	200.67	1509.21	3880.17	1282.88
Pr. time (s)	3052.02	3495.89	4665.32	2464.99	2590.12	4105.93
Pr. paths	16.63	15.67	21.54	15.46	15.63	25.50
Pr. added vars	321.38	230.54	412.75	284.92	259.17	520.54
Max labels	2858.00	6765.54	13618.46	1450.88	1499.88	2005.83
Mean labels	1118.21	2005.96	4857.38	71.21	40.08	19.00
Br. Y branching	186.67	182.75	125.42	260.21	1932.33	563.67
Br. K branching	49.58	74.71	22.67	514.71	77.33	356.21
Node-set it	-	-	-	440.42	198.50	90.63
Forbidden path it	-	-	-	1807.21	1410.63	1448.50

in the number of labels to be treated, and a number of added paths and variables that is somewhat slightly increased. Finally it is interesting to observe how, every two pricer calls on the average, the Enhanced Pricer identified a path falling in the forbidden list due to the restricted subset of limited resources considered, while only in few cases cycles over time were detected.

Mid-aggregation level results. We now analyze Table 2 and 3 collecting the same key indicators this time with a mid-aggregation level, based on number of commodities and allowed paths, respectively. Results show evident performance improvements in both of the pricer versions when shifting from the unsplittable to the 5-splittable setting, in terms of solved instances, average GAPs and computational times. This confirms that a higher number of usable paths is associated with higher degrees of freedom and thus with easier identification of optimal solutions for the transshipment process. 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. Higher computational times in both of the algorithm versions might be related on one side to a larger TEN size on which the pricer operates, this dimension being directly determined by the considered time horizon; on the other side, to the higher number of performed branching operations of type k which, by introducing forbidden paths at the nodes, reduce the efficiency of the comparison rule in discarding dominated labels in the Default Pricer and requires higher number of re-iterations in the Enhanced Pricer, as confirmed by statistics indicators at the bottom of the table. When comparing the two strategies, it can be observed that the Enhanced Pricer reveals its potential already in the unsplittable case, where it was able to identify a dual and a primal bound for all instances in the testbed and improve the average GAP of around 4 point in percentage. Moreover, in the 5-splittable case it succeeded in closing all instances to optimality in a considerable shorter computational time, around 140 times faster than the Default version, while constructing a similarly sized branch and bound tree.

Table 3 presents averaged results, this time aggregated by the number of commodities. The Default Pricer algorithm presents better performances on less congested instances, i.e. when a lower number of commodities must be transshipped. Indeed, we can observe that the 4 commodities setting presents the higher percentage of instances solved to optimality in the shortest computational time, 75.00% in around 3000 seconds, against 58.33% in more than 4600 seconds for the 12 commodities case. Note that the apparent decrease in the GAP on the medium congestion level setting is biased by the restricted subset of instances on which the GAP is measured. Moreover, pricing steps are performed more quickly in the 4 commodities setting, being this behavior 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. Finally, the larger number of columns added to the nodes reveal, in the congested instances, that more paths are required for better multicommodity transshipment and arc capacities usage. This resulted in fewer branching steps and thus smaller branch and bound trees. The performance trend in the Enhanced experiment is not fully apparent, as shown by the results on the intermediate 8 commodities case. Indeed, despite the increase in the average GAP from 6.26% to 8.88%, it presents the same amount of instances solved to optimality with a faster overall computational time and a larger number of explored nodes on the average, somehow denoting a higher variability on the difficulty level of the instances within the subgroup. Finally, improvements achieved through this pricer version are tangible in all of the three settings, thanks to a faster identification of the entering columns and thus a more efficient exploration procedure.

4.2. Use of the Branch and Price Algorithm to assess the quality of Matheuristic results

Our Branch and Price algorithm can make a further valuable contribution through its adoption as a tool to enrich computational analysis of heuristic methods and enhance quality assessment of their identified solutions on $QMCkSFP$ instances. In this section we particularly describe the experiments using our exact algorithm in its Default setting to certify the quality of the solutions obtained by the matheuristic approach presented in [40]. The testbed for this batch of experiments is composed of Grid instances where the solutions of the matheuristic were not certified as being optimal due to a lack of optimal benchmarks and the ineffectiveness of the valid lower bound provided by the free-flow relaxation makespan. This corresponds to a testbed presenting a total of 80 instances with a number of layers ranging in $\{2, 3, \ldots, 10\}$, each being a 5×5 grid with 80 arcs, 5 different levels of commodities and a k-splittable parameter that varies in $\{1, \ldots, 6\}$. For each experiment we allowed six hours of computational time, setting the free-flow relaxation makespan as a valid lower bound (LB) and tailoring the allowed time-horizon based on the makespan of the best solution obtained by the matheuristic approach.

Appendix B presents the complete set of results for this Grid experiment together with detailed description of the key indicators, while averaged results are collected in Table 4. Results show that in 55% of the instances the Branch and Price algorithm was able to identify better optimality GAPs with respect to those previously computed, see rows *Improved GAP*. In particular, all improvements were achieved due to better certification of the lower bounds, while none of the matheuristic best solutions were discarded for better incumbents, rows *Improved LB* and *Improved best sol.* Note that in 65.91% of these cases the resulting GAP is equal to zero, thus proving the previously unknown optimality of the matheuristic best solutions, see rows *Optimal*. In the

Table 4: Results on the Grid test set after the application of the Branch and Price algorithm

	After BP
Improved GAP	44
Improved LB	44
Improved best sol	0
Improved GAP (%)	55%
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%

remaining 15 cases, the new GAP reached 13.47% on average, with an improvement w.r.t. the matheuristic results of 10.11% on average. On one hand, these results are relevant as they confirm the very high quality of the matheuristic best solutions across the whole Grid test bed, where a large number of certified optimal solutions were already identified, mainly on the subset of small to medium-size Grid instances. On the other hand, this batch of experiments was instrumental in validating the suitability of the Branch and Price algorithm as a precious tool for better assessment of the quality of heuristic solutions in those medium to large-size instances where a purely exact approach would not have been an option due to computational hardness.

5. Conclusions

The presented work concerns the development of the first exact algorithm, based on the Branch and Price paradigm, for the strongly NP-hard Quickest Multicommodity k-splittable Flow Problem ($QMCkSFP$) [40]. This dynamic flow problem has been recently introduced in the scientific literature, due to its relevance as a tool in an increasing number of real-life situations where there is a need to minimize the completion time for a set of processes while taking account of path number restrictions in routing flows. Its path-based formulation presents fractional flow support variables and two sets of binary variables charged with identifying time-arrivals of demand flows at destinations and implementing path restrictions by limiting the number of paths for each commodity to at most k.

A restricted relaxed version of this formulation ($RRMP$) is employed in the original Branch and Price approach to explore the feasible region through alternating Column Generation and Branch and Bound steps. In particular, a dedicated relaxation procedure is utilized to linearize binary

variables and those related to path restrictions are discarded by substitution.

A tailored branching rule is then applied to recover feasibility when constraints related to path limitations are violated by the optimal solution to the $RRMP$. 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 involves implementing a refined version of the classical fractional binary branching on the time-arrival variables.

The pricing problem, aimed at selecting new promising entering columns, each of them identified by a (path,departure time) pair, 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 cycles over time. The pricing problem is 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. An enhanced version of the pricing routine is designed with the aim of containing the required amount of resources and hence the overall computational times. The strategy iteratively solves a state-space relaxation of the problem by taking into account only a subset of node-set and forbidden path-related resources in the dynamic programming algorithm. The set of tracked resources is iteratively enlarged whenever infeasible solutions are found.

A computational experiment was conducted to test quality performances of the proposed Branch and Price algorithm in its Default and Enhanced version, examining in particular the impact of its pricing and branching routines on the overall resolution process. The experiments performed on small to medium-size Carbin instances show overall good performances in terms of reaching optimality, which was achieved in 68.06% of the cases, with the average GAP reduced to less than 10% after three hours of running time. Further improvements were obtained with the Enhanced Pricing, resulting in more instances solved to optimality or with a finite optimality GAP in shorter computational times, and a general decrease of the mean GAP. The pricing problem showed a significant impact on the algorithm performances, in particular on more congested instances, revealing the difficulty of using the dynamic programming routine to solve 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 matheuristic approach developed in [40], using a collection of medium to large-size Grid instances previously employed in the matheuristic computational experience.

Our Branch and Price algorithm has improved achievement of the lower bounds, often providing a certificate of optimality of the solutions previously provided by the matheuristic algorithm.
Future research activities will mainly concentrate on improvement of the pricing step performances. To this aim, some preliminary experiments had already been conducted, before the development of the Enhanced Pricing routine, by adapting state-of-the-art techniques conceived to reduce the burden of generating and extending labels in the dynamic programming routing. In particular, a bounding procedure encompassing costs, identifying all-pairs shortest paths w.r.t. reduced costs in the TEN, was employed to enable early discarding of non-dominated labels based on provably worst associated costs. We refer to the following work for the application of a similar procedure to different static network flow problems [15]. 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, further ad-hoc improvement strategies, specifically tailored to the problem features, will be investigated to increase the applicability of the developed Branch and Price exact approach on increasingly larger instances and thus on a wider range of practical applications.

[1] Alvelos, F., Carvalho, J.d., 2005. Branch-and-price and multicommodity flows. Ph.D. thesis. Universidade do Minho, Escola de Engenharia.

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

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

[4] Barnhart, C., Hane, C.A., Vance, P.H., 2000. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Operations Research 48, 318–326.

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

[6] 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, 525–542.

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

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

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

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

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

[12] Chen, Y., 1993. An algorithm for finding the k quickest paths in a network. Computers and Operations Research 20, 59 – 65.

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

[14] 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, 1045–1054.

[15] Di Puglia Pugliese, L., Guerriero, F., 2013a. Dynamic programming approaches to solve the shortest path problem with forbidden paths. Optimization Methods and Software 28, 221–255.

[16] Di Puglia Pugliese, L., Guerriero, F., 2013b. Shortest path problem with forbidden paths: The elementary version. European Journal of Operational Research 227, 254 – 267.

[17] Di Puglia Pugliese, L., Guerriero, F., 2013c. A survey of resource constrained shortest path problems: Exact solution approaches. Networks 62, 183–200.

[18] Fleischer, L., Skutella, M., 2007. Quickest flows over time. SIAM Journal on Computing 36, 1600–1630.

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

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

[21] Gamrath, G., 2010. Generic Branch-Cut-and-Price. Master's thesis. TU Berlin.

[22] 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 .

[23] 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, 82–89.

[24] 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, 278–286.

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

[26] 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. Technical Report 17-61. ZIB. Takustr.7, 14195 Berlin.

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

[28] Hoppe, B., Tardos, É., 2000. The quickest transshipment problem. Mathematics of Operations Research 25, 36–62.

[29] 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, in: 10th International Conference on Natural Computation (ICNC), pp. 1050–1055.

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

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

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

[33] Kohl, N., 1995. Exact methods for time constrained routing and related scheduling problems. Ph.D. thesis.

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

[35] Kotnyek, B., 2003. An annotated overview of dynamic network flows. Technical Report RR-4936. INRIA.

[36] 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.

[37] Lübbecke, M.E., Desrosiers, J., 2005. Selected topics in column generation. Operations Research 53, 1007–1023.

[38] Martens, M., Skutella, M., 2006. Length-Bounded and Dynamic k-Splittable Flows. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 297–302.

[39] 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.

[40] Melchiori, A., Sgalambro, A., 2018. A matheuristic approach for the quickest multicommodity k-splittable flow problem. Computers and Operations Research 92, 111–129.

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

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

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

[44] Ruzika, S., Thiemann, M., 2012. Min−max quickest path problems. Networks 60, 253–258.

[45] 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, Society for Industrial and Applied Mathematics. pp. 821–840.

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

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

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

[49] 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.

[50] Vanderbeck, F., Wolsey, L.A., 1996. An exact algorithm for ip column generation. Operations Research Letters 19, 151 – 159.

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

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