
Article:
Melchiori, A. and Sgalambro, A. (2018) A matheuristic approach for the quickest multicommodity k-splittable flow problem. Computers and Operations Research, 92. pp. 111-129. ISSN: 0305-0548

https://doi.org/10.1016/j.cor.2017.12.012

A Matheuristic approach for 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] *University of Sheffield, 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

The literature on k-splittable flows, see Baier et al. (2002) [1], provides evidence on how controlling the number of used paths enables practical applications of flows optimization in many real-world contexts. Such a modeling feature has never been integrated so far in Quickest Flows, a class of optimization problems suitable to cope with situations such as emergency evacuations, transportation planning and telecommunication systems, where one aims to minimize the makespan, i.e. the overall time needed to complete all the operations, see Pascoal et al. (2006) [2]. In order to bridge this gap, a novel optimization problem, the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$) is introduced in this paper. The problem seeks to minimize the makespan of transshipment operations for given demands of multiple commodities, while imposing restrictions on the maximum number of paths for each single commodity. The computational complexity of this problem is analyzed, showing its NP-hardness in the strong sense, and an original Mixed-Integer Programming formulation is detailed. We propose a matheuristic algorithm based on a hybridized Very Large-Scale Neighborhood Search that, utilizing the presented mathematical formulation, explores multiple search spaces to solve efficiently large instances of the $QMCkSFP$. High quality computational results obtained on benchmark test sets are presented and discussed, showing how the proposed matheuristic largely outperforms a state-of-the-art heuristic scheme frequently adopted in path-restricted flow problems.

Keywords: Quickest flow, k-splittable flow, Matheuristics, Flows over time, Multicommodity

1. Introduction

Network flows over time, also referred to as dynamic flows, are known to be among the most suitable modeling tools to support decision making in those contexts where *time* becomes a relevant driver for planning strategies, such as emergency management, transporta-

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

Email addresses: anna.melchiori@uniroma1.it (Anna Melchiori), a.sgalambro@sheffield.ac.uk (Antonino Sgalambro)

tion distribution, production scheduling, economic planning and many other more [3, 4]. The scientific literature addresses several variants of dynamic flow optimization problems, based on different objectives and features of the considered operational settings to be modeled. A major class of optimization problems in this context is based on the notion of Quickest Flow and seeks for the minimization of the overall amount of time required to transship a given demand of flow between pairs of nodes on a capacitated network with transit times. Such modeling approach turns out to be particularly appropriate when one is interested in minimizing the completion time for a set of operations, for instance in Internet networks, evacuation and transportation management [5, 6]. In this paper we are concerned with enhancing the modeling accuracy to increase the impact of the Quickest Flow optimization problems in real-world contexts. More in detail, we focus on the need to account for realistic restrictions on the number of different paths utilized throughout the dynamic routing, which is currently not addressed in the literature on flows over time. This aspect results significant, for instance, in distribution planning problems where paths represent routes for goods dispatching. In this situation, a number of planned routes exceeding the number of available vehicles in the fleet would represent an impracticable solution [7]. Similarly, in the context of emergency transport, it is desirable to obtain evacuation plans where the number of subgroups for each population is limited, in order to prevent interferences, turbulences, and congestions that may drastically affect the transportation process [6]. In telecommunication, the use of a very large number of paths for a quick transmission of data packets can decrease the overall performance of the protocol, requiring a high cost for path maintenance in the network devices and to reconstruct the original information at destination [8, 9]. All the above mentioned examples show how imposing a realistic number of paths represents an essential modeling need, which is not yet captured neither by the basic variant of the Quickest Flow Problem [10], nor by the Quickest Path Problem [11]. Indeed, spreading the flow on an unlimited and arbitrarily high number of different paths, as for the case of the Quickest Flow Problem, does not represent a valid option in many real cases. On the other hand, a limitation to only one single path for each source-destination pair, as for the Quickest Path Problem, is often far too restrictive and equivalently not realistic.
In this work we aim at bridging this gap, introducing a novel dynamic flow problem, namely the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$). It explicitly 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 the next paragraph we provide an overview of the main contributions on Quickest Flow optimization problems and static variants of the k-Splittable Flow Problem.

Related results from the literature. The concept of *flows over time*, originally named as *dynamic flows*, has been introduced to integrate the temporal dimension into the mathematical modellization on networks. Firstly posed by Ford and Fulkerson [12, 13] in 1958 and 1962, a dynamic digraph presents for each arc a transit time and a capacity attribute. This setting captures the amount of time required for the flow to travel through the arcs and the maximum flow units that can enter each arc at every time instant. The time can be considered both as continuous or discrete and the arc labels can vary over time or not. Relevant static problems have been extended to the dynamic context asking for a completion of an optimal routing

within a given fixed time horizon, see for example the Maximum Dynamic Flow Problem and the Minimum Cost Dynamic Flow Problem [12, 13, 14]. Ford and Fulkerson [12, 13] presented a general procedure to tackle dynamic problems as static ones in a specific network named the Time Expanded Network (TEN), whose size linearly depends on the observed time horizon. Any polynomial algorithm developed for the static environment can be applied to solve the problem of interest in the TEN graph but might result in a pseudo-polynomial algorithm due to possible non-linear dependence of the time horizon in the input size. The same authors have shown how to efficiently construct a Maximum Dynamic s-t flow: a path decomposition is first applied to the solution of a static minimum cost flow obtained in the original network, then temporally repeated flows are sent among the decomposed paths as long as there is enough time left to reach the destination t within the time horizon.
Novel problems find their roots right in the dynamic environment, for instance the Quickest Transshipment Problem and the Earliest Arrival Flow Problem where the exact time horizon of the process is not known a priori [15, 16, 17]. For a complete overview of dynamic problems we refer to Aronson [3], Kotnyek [18], Köhler et al. [4], Skutella [19]. In many general cases the NP-hardness of the previously mentioned problems has been proved: see Klinz and Woeginger [14] for the Minimum Cost Dynamic Flow Problem, Hall et al. [20] for the extension of dynamic problems to the multicommodity case. A relevant minsum-maxmin bicriteria dynamic problem is the Quickest Flow Problem (QFP) that asks for sending an s-t flow taking into account the capacity limitations of inflow on the arcs and such that the last unit of flow arrives at destination as quickly as possible, thus minimizing the makespan of the process. The QFP was shown to be solvable in strongly polynomial time by Burkard et al. [21] in 1993 by performing the Ford and Fulkerson's repeated flows technique within the Megiddo's parametric search. A recent formulation of the QFP can be found in Lin [22]. Its multicommodity extension, namely the $QMCFP$, presents heterogeneous flows that share the same arc capacities while being transshipped. This additional request makes the problem weakly NP-hard [20, 23]. A $FPTAS$ scheme for the $QMCFP$ has been developed by Fleischer [24]: it approaches the problem in condensed-time networks and provides an approximated solution with performance guaranteed $(1 + \epsilon)$ with $\epsilon > 0$. A related problem is the Quickest Path Problem (QPP). Extensively treated in the literature it forces the s-t flow to be routed on a single path [2, 11]. The problem of k-quickest paths ranking has been addressed by Pascoal et al. [2, 25, 26] and applied to the routing of data packets in Internet networks in Clímaco et al. [5]. Melchiori and Sgalambro [6] considered its NP-hard multicommodity variant to solve instances of emergency transportation problems in an evacuation scenario. The produced evacuation plans are easy to be implemented and able to drastically reduce congestion phenomena.
The *k-Splittable Flow Problem* ($kSFP$) has been widely studied on static networks starting from the seminal paper of Baier et al. [1]. An instance of the Maximum s-t $kSFP$ requires to maximize the flow between a pair of nodes such that it can be decomposed in at most k paths. The authors proved the strongly NP-hardness in directed graphs even for the single commodity case and for different constant values of k. They further presented a $1/2-$approximation algorithm for the problem. In Koch and Spenke [7] a comprehensive overview of approximation and complexity results for the Maximum $kSFP$ is presented. Through a reduction process from the $3SAT$ problem they extended the strongly NP-hardness to undirected graphs and derived that the best bound on the approximability

of the problem for all constant values $k \geq 2$ equals 5/6, unless $P = NP$. Complexity in the case of the parameter k being a function of the network parameters is investigated too. Several arc and path-flow formulations for the single and multicommodity Maximum and Minimum Cost $kSFP$ can be found in the works of Truffot et al. [27], Truffot and Duhamel [28] and Gamst and Petersen [29], where ad-hoc pricing and branching strategies within a Branch&Price framework have been proposed for their resolution. A local search heuristic for the Multicommodity Maximum $kSFP$ has been designed by Gamst [30] in 2014. It iteratively looks for a shortest path in a reduced capacitated graph and assigns flow to it accordingly 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. Jiao et al. [31] proposed three heuristics for the bi-objective Multicommodity $kSFP$ minimizing congestion and costs. The strategies differ themselves on the type of relaxation applied to the original problem to obtain an initial solution satisfying the commodity demands. Caramia and Sgalambro [32, 33] dealt with the Maximum Concurrent $kSFP$, where the aim is to maximize the routable demand fraction. They presented an exact algorithm based on Branch&Bound rules and a fast two stage heuristic algorithm. 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 heuristic (RR) has been frequently employed to design approximation algorithms for k-Splittable Flow Problems [34]. This is the case of the Unsplittable Flow Problem ($k = 1$), the Minimum Congestion and the Maximum Concurrent $kSFP$ [9, 35, 36]. 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 edge capacity, is the best known approximated approach for the Maximum Concurrent $kSFP$. In our work we will make use of the RR for comparison purposes. A more detailed description will be provided in Subsection 3.3.
The concept of path restrictions has been rarely addressed in dynamic networks. The only result can be found in Martens and Skutella [37], where a $(3 + 2\sqrt{2})$-approximation algorithm for a single commodity Dynamic k-Splittable Flow Problem with a continuous time parameter is provided.

Contribution of this paper. The contribution of this work is organized as follows in the remainder of the paper. In Section 2 we introduce the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$), providing an original path-based Mixed-Integer Linear Programming formulation for the problem. A computational complexity analysis is addressed in the last part of the section, showing how the $QMCkSFP$ falls into the class of strongly NP-hard problems. In Section 3 a matheuristic algorithm designed ad-hoc to find efficiently good quality solutions for large instances of the $QMCkSFP$ is detailed. The 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 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 Subsection 3.3 we present the Randomized Rounding heuristic (RR) utilized for comparison purposes providing implementation details of its adaptation from the static to the dynamic environment. Section 4 presents the design of experiments and the computational results. In Subsection 4.1 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 4.2 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-size instances against the Quickest Multicommodity Flow Problem, i.e. the relaxation of the considered problem with no upper bounds on the number of paths. In the last part of the subsection we test performances of our matheuristic against those of the RR 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. Conclusions and future research aims are described in the last Section 5.

2. Problem definition, formulation and complexity

In the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$), we are given a dynamic digraph $\mathcal{D} = (\mathcal{V}, \mathcal{A})$, with $\mathcal{V}$ being the set of nodes and $\mathcal{A}$ the set of directed arcs; each arc (i, j) is associated with labels defining a strictly positive capacity c_{ij}, i.e. the maximum number of flow units that can concurrently enter the arc during a time interval, and a non-negative travel time λ_{ij}, specifying the number of time intervals needed to traverse the arc from the tail to the head. The planning horizon over which we observe the phenomena is discretized into a finite set $\mathcal{T}$ of time intervals. During this period arc attributes are assumed time-independent. A set $\mathcal{H}$ of commodities is given, each associated with an amount of demand/population σ_h, a source node o_h and a destination node d_h.
The aim of the $QMCkSFP$ is to find a routing where commodities are shipped through at most k different paths (namely, paths differing from each other at least in one arc) and the number of time instants needed to accomplish the process (makespan) is minimized. Paths are not required to be different, but if they are, the number of different paths used by each commodity cannot exceed the parameter k. Whilst computing the optimal dynamic routing for the commodities, shared arc capacities have to be obeyed and the holdover of each population is allowed only at the respective source node. In the following we provide a complete mathematical path-based Mixed-Integer Linear Programming formulation to describe the introduced optimization problem, whose notation is extensively presented in the box.
The formulation presents fractional variables x_{pt}^h representing flows over times for each commodity and path, being all paths for commodity h collected into the $\mathcal{P}_h$ set. Two sets of binary variables, y_t^h and z_p^h are introduced to allow the linearization of the makespan minimization and to impose the k-splittable flow constraints, respectively. Recall each commodity population might be divided into many subgroups to be routed on multiple paths and scheduled at diverse starting times, hence arriving at destination at different time instants. Each arrival time t for commodity h is recorded by activating the related binary variable y_t^h while each path used by commodity h for demand transshipment enables the variable z_p^h.

Notation:

$\mathcal{V}$ set of nodes,
$\mathcal{A}$ set of arcs,
$\mathcal{T}$ set of considered time intervals,
$\mathcal{H}$ set of commodities,
c_{ij} capacity of arc (i,j),
λ_{ij} travel time/delay of arc (i,j),
(o_h, d_h, σ_h) source, destination, demand/population of commodity h,
$\mathcal{P}_h$ set of paths for the commodity h,
$u_p = \min_{(ij)\in p} c_{ij}$ bottleneck of path p,
δ_{ij}^p binary indicator is 1 if arc (i,j) is traversed by path p,
$l_p = \sum_{(ij)\in\mathcal{A}} \delta_{ij}^p \lambda_{ij}$ length 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 if some flow of commodity h arrives at destination at time t,
z_p^h binary variable is 1 if path p is chosen by commodity h.

The objective function seeks to minimize the overall makespan, represented by the ζ variable, i.e. the number of time instants necessary to complete the transshipment of all the commodity demands. Constraints (2) identify for each commodity the time arrivals for each subgroup of the population, imposing a lower bound to the total makespan. Constraints (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 (4) force each commodity to use at most k different paths. The amount of flow to be transshipped over the time horizon is stated by Constraints (5) while arc capacities at each time step are accounted by Constraints (6). Coupling Constraints (7) impose the amount of flow on a given path to be null on every time instant if the path has not been selected.
A feasible dynamic flow for the $QMCkSFP$ is therefore completely identified by a set of at most k paths for each commodity h, namely $\{p_1^h, p_2^h, \ldots, p_k^h\}$ with corresponding flow values at each time step x_{pt}^h satisfying the network arc capacities and the commodity demands.

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

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

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

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

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

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

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

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

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

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

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

Complexity. We prove the strongly NP-hardness of the above introduced $QMCkSFP$ by reduction from the Minimum Cost $kSFP$. The complexity of the latter can be easily deduced from the works of Baier et al. [1] and Koch and Spenke [7] even for the single commodity version and for any constant $k \geq 2$. Figure 1 depicts the construction of the network used in the reduction process, starting from the dynamic network on the left. Consider two nodes s, t in it (intermediate nodes and arcs are purely indicative); a one-unit planning horizon $\mathcal{T} = \{0, 1\}$, rational capacities on the arcs and zero travel times except for the outgoing arcs of s for which we set $\lambda_{sj} = 1$, $\forall j \in \delta^+(s)$. Expand this digraph over the considered time horizon applying the time-expansion procedure defined by Ford and Fulkerson [12, 13]. In detail, construct two time layers by replicating the nodes of the original graph and rename them with an apex according to the layer; depict each original arc as an arc with tail and head in the same time layer iff its travel time is zero, for example $(i, j) \to (i_0, j_0)$ and $(i, j) \to (i_1, j_1)$, otherwise as an arc connecting nodes at different time layers, for example $(s, j) \to (s_0, j_1)$; link time replica of each node through an holdover dashed arc expressing the possibility of keeping part of a population at the node for one unit of time. The figure on the right is obtained by adding to the generated TEN graph an extra node v and by connecting it with incoming arcs from nodes t_0 and t_1. We construct an instance of the single commodity Minimum Cost $kSFP$ in this static network setting a demand of σ units that has to be transshipped from s_0 to node v; costs are equal to zero except of arc (t_1, v) with cost equal to one. It is trivial to see that a feasible s_0-v k-splittable flow of cost equal to σ exists in the static digraph iff a quickest s-t k-splittable flow with makespan 1 exists within the 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 result is valid for the multicommodity version as well.

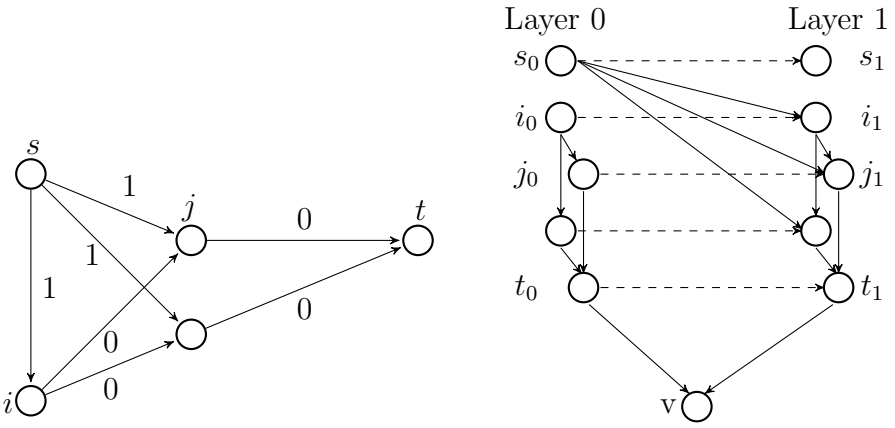


Fig. 1: Reduction process: the initial dynamic digraph with travel times on the arcs on the left and the final static digraph on the right.

3. Matheuristic approach description

The strongly NP-hard complexity proved in the previous section motivates the design of efficient metaheuristic methods to cope with realistic size instances of the $QMCkSFP$ and to provide high quality solutions in reasonable computational times. The proposed algorithm falls in the class of matheuristic techniques, integrating the path-based formulation described in Section 2 into a metaheuristic framework. Matheuristics, sometimes called *model-based metaheuristics*, represent nowadays an attractive topic in the field of optimization. Such resolution techniques hybridize (meta)heuristics and mathematical programming algorithms in several schemes [38, 39, 40, 41], thus combining time efficiency and methodological rigor. In this section we provide a full description of our algorithm detailing its single features. We reserve Subsection 3.3 for introducing the competing approach used in the computational experiments.

The developed matheuristic 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 [42] performs a local search on multiple neighborhood structures ordered in list $N^s,\ s = 1, \ldots, s_{max}$ 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-based model of Section 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, whilst its pseudocode description is provided in Algorithm 1.

Algorithm 1 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
Δ;	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 $h \in \mathcal{H}$:
  $\mathcal{P}_h \leftarrow L_h[0 : k-1]$;
$(x_0, val(x_0)) \leftarrow solveMIP(\mathcal{P}_h, val(\bar{x}))$;
$(\bar{x}, val(\bar{x})) \leftarrow (x_0, val(x_0))$;

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

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

  if $(val(x) < val(\bar{x}))$: ▷ acceptance decision
    $(\bar{x}, val(\bar{x})) \leftarrow (x, val(x))$; ▷ re-centering
    $s \leftarrow 1$; ▷ restart from initial neighborhood
    $r \leftarrow 1$;
  else:
    if (r < $r\text{-}max$):
      r ← r + 1;
    else if (s < $s\text{-}max$):
      s ← s + 1; ▷ generate next neighborhood
    else:
      s ← 1;
      r ← 1;
}**while**$(time \leq time\text{-}lim)$;
return $(\bar{x}, val(\bar{x}))$;

3.1. 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 list of feasible paths ordered accordingly to their transmission time, i.e. the time required for transshipping the demand along it. 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 lists is performed by adopting the lazy version of the Chen's approach presented in Pascoal et al. [2, 26] for ranking quickest loopless paths, together with the Yen's algorithm [43]. The utilized method keeps in memory an array 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 output is thus a list of quickest paths ranked in increasing order of transmission time. We apply this algorithm to each commodity and get the L_h lists (see Instruction 1 of Algorithm 1). The length L of each list must be calibrated in order to balance the tradeoff between quality and speed.
The initial solution, see Instruction 2 of Algorithm 1, is build as follows: for each commodity the first k paths in the respective candidates' list L_h are selected, the path-based model presented in Section 2 is fed with such paths by populating the P_h sets, the MIP-solver is called, and finally the optimal solution is returned together with its associate makespan value.

3.2. 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 then returned.

Neighborhood structures. We construct a neighborhood by linking its structure to a finite collection of feasible paths from the L_h lists for each commodity. In particular, all paths used by the current incumbent (at most k for each commodity due to Constraints (4)) are included; additional paths are selected at random from the candidates' lists until the desired cardinality is fulfilled. Elements of a neighborhood are thus all the feasible solutions to the $QMCkSFP$ where only the identified paths can be used. We express a given neighborhood centered at the incumbent solution $\bar{x}$ as $N^s(\bar{x}) = \bigcup_{h\in\mathcal{H}} N_h^s(\bar{x})$, with each $N_h^s(\bar{x}) \leftrightarrow \{p_{s_1}(\bar{x}), p_{s_2}(\bar{x}), \ldots, p_{s_{S-1}}, p_{s_S}\}$, being S the common cardinality. Moving forward in the list of neighborhood structures the number of paths to be identified for each commodity increases by a factor of the parameter k i.e. $|N_h^s(\bar{x})| = S = k\,(s\,\Delta+1)$ for all $h \in \mathcal{H}$ 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 3 of Algorithm 1). Note that the value of the Δ parameter 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-based formulation are updated accordingly, i.e. $\mathcal{P}_h = N_h^s(\bar{x}),\ \forall h \in \mathcal{H}$. The generated model is then solved to optimality by means of a MIP-solver. To speed up the execution times and perform a higher number of run, 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-based model 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 and the matheuristic algorithm restarts with the first neighborhood structure in the list centered at the updated incumbent. If no improvement is obtained in the current neighborhood, 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. The *s-max* parameter states the end of the neighborhoods list and once reached the algorithm is forced to restart from the first structure.

3.3. A competing approach

We adopt the Randomized Rounding algorithm (RR) as a competing approach to our matheuristic on very large-size instances where optimal values are not available as a benchmark for quality solution. The RR algorithm was proposed by Raghavan and Tompson [34] and frequently employed to tackle large instances of static k-Splittable Flow Problems, see for example Białoń [9] and Caramia and Sgalambro [33]. In the static framework the RR initially solves the relaxation of the path-restricted original problem where no limitation on the maximum number of paths is imposed, namely the free-flow relaxation. Then, it decomposes the so obtained optimal flow into paths making use of the $|\mathcal{A}| \times |\mathcal{H}| \times |\mathcal{T}|$ Ford and Fulkerson's algorithm [44]. For each commodity, k of the decomposed paths are selected at random with a probability equal to their assigned flow and the original k-splittable formulation is employed to optimally reroute the commodities' demands on them. The path selection process is repeated to improve the quality of the current best solution till a maximum number of rounds or a time limit is reached.
We adapt the RR heuristic to our dynamic case as follows. The free-flow relaxation of the problem, equivalent to the $QMCFP$, is solved to optimality through a binary search that explores the given time horizon by iteratively looking for a feasible multicommodity flow on a restricted TEN. The solution that accomplishes the transshipment in the minimum time is then returned as the optimal solution to the free-flow relaxation $QMCFP$. The obtained flow is decomposed into paths in the original dynamic graph each with its respective flow over time, i.e. $(p, \sum_{t \in \mathcal{T}} x_{pt}^h),\ p \in \mathcal{P}_h$. The steps described so far can be viewed as the RR initialization procedure to generate a list of candidate paths. From our tests this 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

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

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				

2. Note that Constraints (4) 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.

4. 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 1 hour of running time and 72 time instants as time horizon to perform the transshipment. In Subsection 4.1 we provide a proof-of-concept of our model, evaluating the performance of the matheuristic against a Branch&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 4.2 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 [33] and the Carbin test set [45]. The matheuristic's parameters tuning conducted using the *irace* package 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 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 3.3. 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.

4.1. 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 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

Table 2: Dense test set: the number of commodities is fixed to 5 and k varies in $\{1, \ldots, 6\}$.

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

Table 3: Carbin test set: a identifies the level of congestion and k varies in $\{1, \ldots, 6\}$.

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

by the solver. Each instance has been tested with 3 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. phase 1. and 2. of Algorithm 1, 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-size 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.

4.2. 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 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 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 ratio *mean capacity of arcs/mean demand of commodities*: the Bs subgroup presents a small congestion ratio while the Bl 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 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* [46] 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. The remaining matheuristic's parameter *s-max* directly depends in turn on the cardinality L of the lists. 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 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 instance class with optimal setting $\Delta = 0.5$ collects the smallest and simplest networks among all the three considered test sets and this parameter value is combined with a lower randomization parameter to speed up the exploration routine. In the next experiments the matheuristic has been tuned accordingly to the obtained optimal settings.

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 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 5 collects some results aggregated by the flow split 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

Table 4: Irace tuning results for the L, Δ, $r\text{-}max$ parameters

Instance class	L	Δ	$r\text{-}max$	Instance class	L	Δ	$r\text{-}max$
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 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 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

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 lower 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 5: Aggregated results on the Grid test set

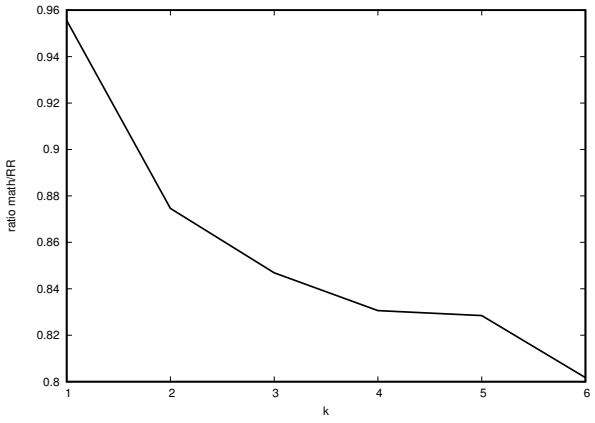
k	init sol	t (s)	best sol	t (s)	moves	gap (%)
1	35.64	3.22	28.96	222.16	2.20	20.65
2	30.44	3.18	25.22	58.22	2.29	5.10
3	27.33	3.31	24.40	156.09	1.76	1.67
4	26.60	3.42	24.20	126.84	1.47	0.83
5	26.11	3.44	24.09	21.60	1.24	0.37
6	25.76	3.56	24.07	61.27	1.07	0.28

Table 6: Aggregated results on the Dense test set

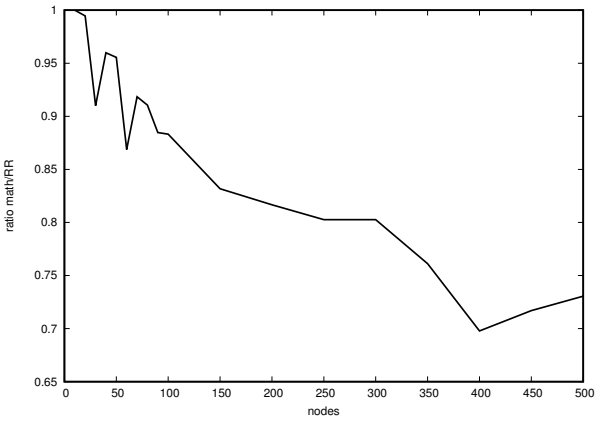
	Matheuristic			RR			Comparison	
k	t (s) init sol	t (s) best sol	moves	t (s) init sol	t (s) best sol	moves	ratio init sol	ratio best sol
1	3.28	61.94	0.33	263.83	1020.50	3.17	0.8208	0.9556
2	3.33	19.06	1.22	96.50	1283.78	4.89	0.7363	0.8746
3	3.72	74.94	2.33	95.00	1651.61	5.83	0.7142	0.8468
4	3.61	232.61	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 7: Aggregated results on the Carbin test set

	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. 2: Comparison of the two algorithms' performance depending on the k parameter and on the number of nodes.

The Dense test set. In this experiment we compare our algorithm against the RR heuristic 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 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 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 preliminary analysis shows the effectiveness of the initialization phase of our matheuristic: feeding the

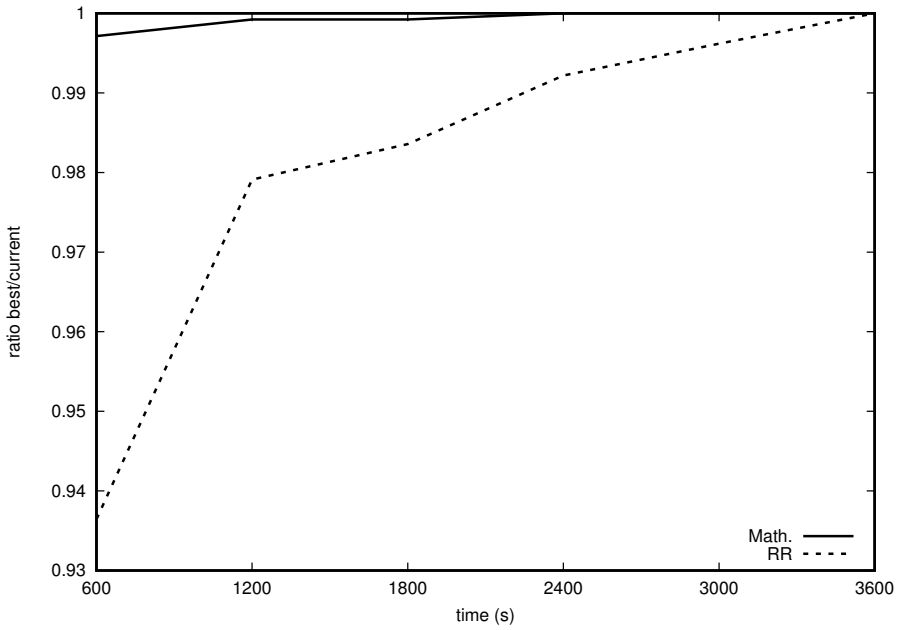


Fig. 3: Average ratio between the best solution and the solution found after a given time for both methods.

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 and 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 we observe how a variation in the flow split parameter does not substantially affect the initial times 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. 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. 2 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 2a and 2b 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 2a graphically represents the values of the last column of Table 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 3 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 presents 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 simultaneously 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 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 mainly happen in the most congested networks Bs05 and Bs07, see values greater than 1.0 in the "ratio init sol" column of Table 7. 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 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 flow split 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.

5. Conclusions and outlook

In this work a novel dynamic flow problem is introduced, the Quickest Multicommodity k-Splittable Flow Problem ($QMCkSFP$), integrating a number of concepts and modeling features from the literature on k-splittable flows within the class of Quickest Flow optimization problems. An increased amount of real-world applications enabled by k-splittable flows motivate the interest in the proposed problem, which accounts for realistic restrictions on the number of different paths utilized throughout the dynamic routing, a feature that had not been previously addressed in the literature on flows over time. The strongly NP-hard complexity of the problem is proved and a path-based Mixed-Integer Linear Programming formulation is provided and embedded within a matheuristic approach, designed ad-hoc to solve efficiently large instances of the $QMCkSFP$. The proposed matheuristic is 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 computational experience presented in the paper provides 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-size instances. In some cases, the optimality of the solutions could not be checked due the lack of benchmarks: suitable exact approaches such as those based on column generation

could be considered in the future as a tool to certify optimality for such solutions. On very large-size benchmark instances and even in presence of a large number of commodities, the matheuristic reveals 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. A relevant future development for this research is represented by the adoption of the proposed $QMCkSFP$ formulation and the associated matheuristic algorithm to cope with real-world problems in evacuation management and transportation planning. This could require an adaptation of the problem to account for specific further operational features, hence posing new challenges from the modeling and algorithmic perspective.

Acknowledgments

The authors would like to thank the area editor and the anonymous referee for their useful suggestions that permitted to improve the quality of the paper.

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 A.8: 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 B.9: 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 sol	t (s)	best sol	t (s)	moves
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

	Instance				Matheuristic				
	nodes	arcs	h	k	init sol	t (s)	best sol	t (s)	moves
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 C.10: 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 sol	t (s)	best sol	t (s)	moves	init sol	t (s)	best sol	t (s)	moves
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 D.11: Experiments on the Carbin test set

References

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

[2] M. Pascoal, M. Captivo, J. Clímaco, A comprehensive survey on the quickest path problem, Annals of Operations Research 147 (1) (2006) 5–21. doi:10.1007/s10479-006-0068-x.

[3] J. Aronson, A survey of dynamic network flows, Annals of Operations Research 20 (1) (1989) 1–66. doi:10.1007/BF02216922.

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

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

[6] A. Melchiori, A. Sgalambro, Optimizing Emergency Transportation through Multicommodity Quickest Paths, Transportation Research Procedia 10 (2015) 756 – 765, 18th Euro Working Group on Transportation, EWGT 2015,Delft, The Netherlands. doi:https://doi.org/10.1016/j.trpro.2015.09.029.

[7] R. Koch, I. Spenke, Complexity and approximability of k-splittable flows, Theoretical Computer Science 369 (1) (2006) 338–347. doi:https://doi.org/10.1016/j.tcs.2006.09.015.

[8] C. Jiao, W. Yang, S. Gao, Y. Xia, M. Zhu, The k-splittable flow model and a heuristic algorithm for minimizing congestion in the MPLS networks, in: 10th International Conference on Natural Computation (ICNC), 2014, pp. 1050–1055.

[9] P. Białoń, A randomized rounding approach to a k-splittable multicommodity flow problem with lower path flow bounds affording solution quality guarantees, Telecommunication Systems 64 (3) (2017) 525–542. doi:10.1007/s11235-016-0190-2.

[10] L. Fleischer, M. Skutella, Quickest flows over time, SIAM Journal on Computing 36 (6) (2007) 1600–1630. doi:10.1137/S0097539703427215.

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

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

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

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

[15] B. Hoppe, E. Tardos, The Quickest Transshipment Problem, in: Mathematics of Operations Research. Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 1995, pp. 512–521.

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

[17] M. Schlöter, M. Skutella, 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, 2017, pp. 821–840.

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

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

[20] A. Hall, K. Langkau, M. Skutella, An FPTAS for Quickest Multicommodity Flows with Inflow-Dependent Transit Times, Algorithmica 47 (3) (2007) 299–321.

[21] E. Burkard, K. Dlaska, B. Klinz, The quickest flow problem, Zeitschrift für Operations Research 37 (1) (1993) 31–58. doi:10.1007/BF01415527.

[22] M. Lin, P. Jaillet, 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, 2015, pp. 1343–1356. doi:10.1137/1.9781611973730.89.

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

[24] L. Fleischer, M. Skutella, The Quickest Multicommodity Flow Problem, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

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

[26] M. Pascoal, M. Captivo, J. Clímaco, Computational experiments with a lazy version of a k quickest simple path ranking algorithm, TOP 15 (2) (2007) 372–382. doi:10.1007/s11750-007-0033-0.

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

[28] J. Truffot, C. Duhamel, A Branch and Price algorithm for the k-splittable maximum flow problem, Discrete Optimization 5 (3) (2008) 629–646. doi:https://doi.org/10.1016/j.disopt.2008.01.002.

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

[30] M. Gamst, A local search heuristic for the Multi-Commodity k-splittable Maximum Flow Problem, Optimization Letters 8 (3) (2014) 919–937. doi:10.1007/s11590-013-0622-9.

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

[32] M. Caramia, A. Sgalambro, An exact approach for the maximum concurrent k-splittable flow problem, Optimization Letters 2 (2) (2008) 251–265. doi:10.1007/s11590-007-0055-4.

[33] M. Caramia, A. Sgalambro, A fast heuristic algorithm for the maximum concurrent k-splittable flow problem, Optimization Letters 4 (1) (2010) 37–55. doi:10.1007/s11590-009-0147-4.

[34] P. Raghavan, C. Tompson, Randomized rounding: A technique for provably good algorithms and algorithmic proofs, Combinatorica 7 (4) (1987) 365–374. doi:10.1007/BF02579324.

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

[36] M. Caramia, A. Sgalambro, On the approximation of the single source k-splittable flow problem, Journal of Discrete Algorithms 6 (2) (2008) 277–289. doi:https://doi.org/10.1016/j.jda.2007.03.001.

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

[38] M. Ball, Heuristics based on mathematical programming, Surveys in Operations Research and Management Science 16 (1) (2011) 21–38. doi:https://doi.org/10.1016/j.sorms.2010.07.001.

[39] C. Blum, J. Puchinger, G. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: A survey, Applied Soft Computing 11 (6) (2011) 4135–4151. doi:https://doi.org/10.1016/j.asoc.2011.02.032.

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

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

[42] P. Hansen, N. Mladenovic, Variable neighborhood search: Principles and applications, European Journal of Operational Research 130 (3) (2001) 449–467. doi:https://doi.org/10.1016/S0377-2217(00)00100-4.

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

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

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

[46] M. López−Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stutzle, M. Birattari, The *irace* package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (Supplement C) (2016) 43–58. doi:https://doi.org/10.1016/j.orp.2016.09.002.