

A randomized rounding approach to a k -splittable multicommodity flow problem with lower path flow bounds affording solution quality guarantees

Paweł M. Białoń¹

Published online: 23 June 2016

© The Author(s) 2016. This article is published with open access at Springerlink.com

Abstract In 2005, Baier et al. introduced a “ k -splittable” variant of the multicommodity flow (MCF) problem in which a given commodity was allowed to flow through a number of paths limited by a small integer. This problem enables a better use of the link capacities than the classical Kleinberg’s unsplittable MCF problem while not overloading the used devices and protocols with a large number of paths. We solve a minimum-congestion k -splittable MCF problem coming from a practice of managing an software-defined, circuit switching network. We introduce a lower bound for a path flow in order to model a QoS demand for a single connection running the path. Instead of reducing the problem to the unsplittable flow problem, as suggested by Baier et al., we propose a potentially more exact method. We directly enhance the Raghavan and Thompson’s randomized rounding for ordinary MCF problems to account for k -splittability and the lower flow bounds. A mechanism is constructed that prevents rounding up low flows in the subproblem solution to big values. It is based on modifying the continuous subproblem by additionally penalizing flows of certain commodities in certain links. When $k = 1$, this allows us to prove a property similar to the $\mathcal{O}(\sqrt{\log m})$ approximation factor, where m denotes the number of network links. We give probabilistic guarantees for the solution quality and examine the behavior of the method experimentally.

Keywords k -Splittable flow · Randomized rounding · Multicommodity flow

✉ Paweł M. Białoń
P.Bialon@itl.waw.pl

¹ National Institute of Telecommunications, Szachowa Str. 1,
04-894 Warsaw, Poland

1 Introduction

1.1 Problem

Modern communication networks are experiencing a growth of techniques enabling complex traffic engineering. One important step in this process was the introduction of Multiple Protocol Label Switching Network-MPLS [10]. With the introduction of MPLS, it became possible to project the traffic so that several paths conduct the traffic between two nodes. This allows a better use of the available link capacities without congesting the network. The Software Defined Networking (SDN) paradigm [22] has decoupled the control of network traffic from the actual packet forwarding. Control is centralized in a selected network node, it is programmable in an elastic programming language and is based on the global information of the network topology, collected with dedicated SDN mechanisms. Thus SDN opens the way to elastically design the traffic, and manage the traffic using even complex optimization (mathematical programming) algorithms.

We present an algorithm that uses the above possibilities. It solves a κ -splittable multicommodity flow problem with lower path capacity bounds. Each of the defined commodities is allowed to be transmitted through at most κ paths. The goal is to minimize the network congestion (the highest link saturation). The problem parameter κ allows steering the compromise between a good usage of the available network capacity and reducing the number of paths that must be maintained in the network devices and protocols. A minimum path capacity can be set for each commodity, in order to keep the path capacities for a commodity of a similar range or to satisfy some QoS constraints. The algorithm is based on the randomized rounding technique [21].

The described pattern of traffic engineering is applicable to a range of classical networks, on different levels of management: from LANs, single domains to WANs, whenever they are controlled with some elastic technique like SDN. A particular circumstance stimulating the diversity of models of traffic engineering is a division of the available network resources into smaller pieces leased to other operators. A refined variant of such an approach is virtualization, i.e., defining virtual networks upon an existing physical infrastructure. The virtual or leased resources are quite abstract constructs, of a high position in the protocol stack, and the flexibility of traffic engineering for such constructs is natural. Actually, the algorithm described in the paper comes from a management module in a virtualized network, as described later. The considered traffic pattern might be as well present in post-IP data-centric, or content-aware networks [2], data-intensive grid computing, cloud computing and multi-host data centers [23]. However, in most applications, an augmentation of the presented approach would be necessary to account for the respective structural specificity, for example for multicast, limited radio reach and energy constraints, scheduling of computations. We suggest one augmentation serving for the link disjointness of the paths for a commodity. It is directed to the network resilience and can make the approach more practical for all the areas mentioned above, also, opening it also to life-critical applications.

The multicommodity flow problem presented here comes from a network management module [11] developed for a circuit switching network developed in continuation of the 7FP Future Internet Engineering project [4] (aimed at obtaining a coexistence of various network techniques, e.g., circuit switching, IP, post-IP in a common physical network infrastructure by means of virtualization). The problem consists in calculating intra-domain traffic paths for the network to meet specific requirements. The network is a “software-defined network” as it uses the OpenFlow protocol [18]. The constructed management plane can define several paths in a given node-to-node intra-domain relation upon the network setup. Calculating these paths forms our optimization problem. In an operating network, the set of these paths is used by the control plane. When establishing a connection, the control plane chooses between these paths, taking into account the current load of the paths. When multiple connections are simultaneously open, and when the load balancing works efficiently, we effectively transmit the traffic in the relation through several “large” paths, making a better use of the available link capacities.

1.2 Related work

The notion of a κ -splittable problem (customarily called “ k -splittable”) has been introduced in [1], as a natural gener-

alization of the unsplittable flow problem, in which we allow only one path per commodity [14].

Researches have investigated several formulations of the κ -splittable problem, analogous to that listed in [14] for the unsplittable problem. A frequently considered one is the minimum-congestion formulation, in which we minimize the maximum saturation of a link, while satisfying the flow demands given for each commodity. In the maximum flow formulation, the summaric volume of commodities sent is maximized subject to link capacity constraints. If we fix proportions of flows for the commodities, this version becomes equivalent to the minimum-congestion formulation by a simple flow scaling. As another example, the “maximum routable demands” choose the subset of the commodities that will be transmitted so as to maximize its cardinality, subject to arc capacity constraints. Often various cost models are considered, minimizing the cost defined as some function of flow subject to arc capacity bounds and the flow demands for the commodities. Additional restrictions on the problem structure are frequently imposed, the most popular is the “balance condition”: the maximum demand for a commodity is not greater than the minimum link capacity.

We shall by default consider the minimum-congestion formulation and problems with directed graphs. Already with $\kappa = 1$, the κ -splittable problem is NP-hard [16].

Let n denote the number of nodes, m —number of arcs, K —number of commodities. Similarly to unsplittable flow problems, κ -splittable problems are usually solved with a few groups of methods. The first group gathers typical optimization approaches aimed basically at obtaining an exact solution. Here belong the branch-and-bound approaches and their accelerations called branch and-price that generate variables with solving shortest path subproblems—see, e.g. the approach [24] for a maximum flow formulation. Another approach [8] produces different subproblems that describe a set of paths for a commodity instead of a single path. However, the branch-and-bound technique is of an exponential complexity. The solution time can be very large. Moreover, it is unpredictable, it can highly vary between instances of the problem, even of a similar size. This has discouraged the use of such approaches in a practical, operating network management module.

Another group of methods are heuristic methods. They are faster, they often deliver guarantees for the solving time but give only approximate solutions without a formal assessment of the quality (proximity). Caramia and Sgalambro [5] consider a maximum-flow model. They compute each path for a commodity and the related flows according to the polynomial scheme from [1]. This is repeated for all the commodities, ordered in random. Certainly, this leads to conflicts of computed path flows regarding link capacities, since the paths have been computed to a large extent independently. The conflicts are resolved with another polynomial proce-

ture, a local improvement step, based on the ideas of flow deviation technique of [7], invented originally for splittable multicommodity flows. In the local improvement step, bottlenecks are detected and by-passed. The authors achieve run time $\mathcal{O}(K\kappa^2 \cdot m \log(m) \cdot \log(\rho))$ under the balance condition, with ρ being the ratio of the sum of demands on the minimum link capacity. Another polynomial-time algorithm for the maximum-flow problem is given in [13], although the authors do not calculate the rank of this time explicitly. They build an initial feasible-solution in terms of link flows and reinterpret it as a superposition of path flows, with a possibility of having several different paths for a commodity. They neglect all but the κ widest paths for a commodity (actually, the splittability level κ may be different for each commodity in this work). Once the paths have been chosen, the values of flows along them that maximize the goal function are found as a solution of an auxiliary linear programming problem. An interesting heuristic is proposed in [12]. The authors cope with arbitrary in choosing a particular flow model by defining a bi-criteria model with the twofold goal of minimizing the congestion and minimizing the cost of the transfer. However, they use the mere weighting in the goal function in order to combine the criteria, with all the possible consequences having been shown in the theory of decision making, like impossibility of obtaining all efficient (Pareto-optimal) solutions by choosing the weight [25].

Approximation algorithms afford some guarantees of the quality of the obtained solution. For the minimum-congestion formulation, and assumed $P \neq NP$, the approximation factor cannot be better than $\frac{6}{5}$, as shown in [16]. It cannot be either better than $\kappa/(\kappa + 1)$ when $2 \leq \kappa \leq m$ [1]. On the constructive side, the authors of [1] proposed a reduction of the κ -splittable problem to the so called uniform k -splittable problem, where the path flows for a commodity are required to be equal. In this way, when the uniform problem is solved within the approximation factor of ρ , then the initial κ -splittable problem can be solved within the 2ρ ratio. The uniform problem, in turn, may be in an obvious manner drawn to an unsplittable multicommodity flow problem. Probabilistic approximation based on randomized rounding, started by Raghavan and Thompson [21] is the main tool in approximating unsplittable flow problems. There, a continuous Multicommodity Flow (MCF) relaxation of the problem (without restrictions on splittability) is solved and then the continuous solution, interpreted as a superposition of path flows, is “rounded” to retain the unsplittability of flows. In the solution of the continuous problem, for a given commodity number k , we can see flow demand $D(k)$ as realized by a flows on several paths, where the flows sum up to $D(k)$. Raghavan and Thompson choose randomly only one of these paths for the final solution (and attribute the flow of $D(k)$ to this path in this solution). The probability of choosing a particular path is proportional to the flow on this path in the

continuous solution. In other words, the flows in the continuous solution are rounded either to zero or to $D(k)$. In this method, the expected value of the flow of a commodity in a link after the rounding is the same as the value of this flow before the rounding, i.e., in the solution of the continuous subproblem. However, in the later solution, the conflicts about the link capacities are already well resolved. In turn, for problems where many commodities pass a link, the real value of the link saturation is close to its expected value.

The run time of randomized rounding methods is low and well predictable. The solutions are inexact but with precise probabilistic guarantees for the level of inexactness. This perfectly fits the needs of our application. The inexactness of solution can be easily neutralized by the oversizing (overdimensioning) of resources commonly present in modern network; many different network algorithms, not necessarily mathematical programming algorithms, are rough heuristics taking advantage of such an oversizing. Raghavan and Thompson obtain the approximation ratio of $\mathcal{O}((\log m)^{1/2})$ for a specific version of the minimum-congestion formulation with unit link capacities and undirected graphs. For more general settings, in directed graphs, it was shown in [19] that randomized rounding methods cannot obtain a ratio better than $\mathcal{O}(\log m / (\log \log m))$.

1.3 Contribution of this work

We propose a method of solving a minimum-congestion κ -splittable multicommodity flow problem in a directed graph, with lower bounds on a path flow, without simplifying structural assumptions like balance condition or unit edge capacities.

Our method is based on randomized rounding. However, unlike in [16], we do not rely on the simple reduction of the κ -splittable problem to unsplittable problem. The deterioration of solution quality by a factor up to 2 still seems wasteful from the practical point of view. Instead, we try to directly intervene in the randomized rounding scheme. Our rounding may yield up to κ paths for a commodity and the flows assigned to these paths try to mimic the corresponding proportion in the fractional flow. We prove that if the optimal solution of the 1-splittable version of our κ -splittable problem does not use more than α part capacity of each link, where $\alpha = 1/\mathcal{O}((\ln m)^{1/2} + \ln \kappa)$, then with probability arbitrarily close to 1 our method yields a solution of the κ -splittable that does not oversaturate links. Increasing κ usually lowers the network congestion. The latter feature is not visible in the estimates but is left as a heuristics coming from the proportions setting in the algorithm mentioned above.

Our method admits lower bound on flows in the computed paths. In our problem, the flows in the graphs represent a traffic composed of many connections. The bounds on the path flows might model the QoS flow requirements that must

hold even for a single connection. Without the bounds, an algorithm could compute a narrow path, insufficient even to transfer a single connection. Narrow paths can be uncomfortable also for other reasons: they might occupy the routing tables without giving a considerable contribution into the transfer. A lower bound on a path is intuitively a combinatorially difficult element, since it introduces a discontinuity in the allowed values: either zero or at least the lower bound value. The author is not aware of considering an identical bound in the literature on k -splittable multicommodity problem. Some authors consider flows of commodities sent in quants (containers) in transportation networks [19], which is a slightly more restrictive setting.

For $\kappa = 1$, our method reduces to an unsplittable flow algorithm that (in essence) yields a goal value at most $\mathcal{O}((\log m)^{1/2})$ worse than the optimal value (disregarding the arithmetic precision and on condition that the true optimal solution does not oversaturate links). Paradoxically, this is better than the $\mathcal{O}(\log m / (\log \log m))$ lower bound shown in [19] for randomized rounding methods solving the minimum-congestion formulation in directed graphs. The paradox is resolved by the fact that we propose a different, modified continuous multicommodity subproblem. We modify the coefficients in the continuous MCF subproblem formulation to decrease the flows of commodities of large demands by low capacity links in its solution. In this manner, we cope with the known effect of possible rounding low flows in the solution of the continuous MCF to large values, that can result in a large inaccuracy of the obtained solution.

A previous stage of the method development has been reported in conference materials [3]. That approach did not modify the continuous MCF subproblem and did not establish theoretical guarantees of the solution quality (proximity), which could occasionally become very poor due to the effect of rounding small flows to big values, despite that a heuristics preventing this was present.

Throughout the paper, we assume that summing over an empty set of indexes yields zero.

In Sect. 2 we formulate our problem. Its solution is described and analyzed in Sect. 3. Section 4 analyzes the approximation quality and Sect. 5 summarizes the experiments with the method. Conclusions and indications for the further work are given in Sect. 6.

2 Problem formulation

Let \mathbb{N} be the set of natural numbers (including zero). We have a directed graph $G = (V, E)$, where $V \subset \mathbb{N}$ is the set of nodes (vertices), $E \subset V \times V$ is the set of links (arcs); arcs connecting a node to itself are not allowed. Thus $n = |V|$ is the number of nodes, $m = |E|$ is the number of links. Furthermore:

- $K \in \mathbb{N}$, $K \geq 1$ is the number of commodities;
- $s(k), t(k)$ are the source node and the sink node for commodity k ; $k = 1, \dots, K$;
- $D(k) \in \mathbb{R}$, with $D(k) > 0$ for $k = 1, \dots, K$ is the traffic (flow) demand for commodity k (problem parameter);
- $\phi^{lo}(k) \in \mathbb{R}$, where $0 \leq \phi^{lo}(k) \leq D(k)$, for $k = 1, \dots, K$ is the minimal flow on a path for commodity k (problem parameter);
- $\kappa \in \mathbb{N}$, where $\kappa > 0$, is the maximum number of paths for a flow (our problem is a problem of κ -splittable flow).

A path p is a sequence $(p[1], p[2], \dots, p[j])$ of different nodes in V , where $j \geq 1$, such that $(p[i], p[i + 1]) \in E$ for $i = 1, \dots, j - 1$. Let P^k be the set of all paths from $s(k)$ to $t(k)$ in graph G .

For each commodity k , the integer number η_k (where $1 \leq \eta_k \leq \kappa$), flow paths $p_1^k, p_{\eta_k}^k$ (where $p_i^k \in P^k$) together with corresponding positive flow values (shortly: flows) $\phi_1^k, \dots, \phi_{\eta_k}^k$ (where $\phi_i^k \in \mathbb{R}$, $\phi_i^k > 0$) must be found such that $\sum_{j=1}^{\eta_k} \phi_j^k = D(k)$.

We now formulate our problem:

$$(P)$$

$$\text{minimize } \xi^{\max} \quad (1)$$

over

$$\eta_k \text{ (where } 1 \leq \eta_k \leq \kappa) \quad (2)$$

$$p_i^k \in P^k, \phi_i^k \geq 0 \text{ for } k = 1, \dots, K, i = 1, \dots, \eta_k \quad (3)$$

subject to

$$\xi^{\max} = \max_{e \in E} \left(\left(\sum_{k=1}^K \sum_{i=1, \dots, \eta_k: e \text{ on } p_i^k} \phi_i^k \right) / c_e \right) \quad (4)$$

$$\sum_{i=1}^{\eta_k} \phi_i^k = D(k) \text{ for } k = 1, \dots, K \quad (5)$$

$$\phi_i^k \geq \phi^{lo}(k) \text{ for } k = 1, \dots, K, i = 1, \dots, \eta_k \quad (6)$$

where e on p means that arc e lies on path p .

Constraints (5) enforce that the network transfers exactly the amount of traffic $D(k)$ for a commodity k and (6) expresses the lower bounds for path flows. The goal function ξ^{\max} is simply the highest saturation level of a link. The saturation level of a link is the ratio of total flow in this link and the link capacity—cf. (4). The saturation level of a link means that the link capacity is actually exceeded (the link is overloaded). The value of goal function, ξ^{\max} is less than 1 iff none of the links are overloaded, i.e., the given variable values yield a practically viable set of transportation paths. A greater goal value indicates that the network needs enhancements, and the capacity excess by particular link is listed by the solver to the administrator in such a case. Alternatively (and more likely) the administrator might want to adjust his

demands $D(k)$, that often express some prognosis of the traffic to be carried by the network and thus are imprecise.

3 Solution method

3.1 General scheme

The solution process is depicted as Algorithm 1, with given parameter $r \geq 1$ —number of rounds. The algorithm uses

Algorithm 1 Main algorithm.

- 1: Solve the continuous MCF problem (C) as in Sect. 3.2. If (C) is infeasible, STOP with error.
 - 2: (Path reconstruction) Find a representation of the solution in terms of a superposition of path flows. I.e, find integers η'_k , where $\eta'_k \geq 1$, paths $p_i^k \in P^k$
 - 3: **for** r times **do** {Do r rounds}
 - 4: Based on the result of the previous step, find η_k (where $1 \leq \eta_k \leq \kappa$) for $k = 1, \dots, K$ and $p_i^k \in P^k$, $\phi_i^k \geq 0$ for $k = 1, \dots, K$, $i = 1, \dots, \eta_k$ in the randomized rounding procedure.
 - 5: **end for**
 - 6: Take the lowest-goal-value result of rounding among the rounds done as the final solution.
-

a folklore observation (see e.g. Theorem 1.7.3 in [9]). Assume we have some network flow of some commodity given by flows per the links. Then it can be equivalently (though in general nonuniquely) expressed as a superposition of flows on several non-looping paths leading from the source node to the destination node and some circulations (flows on some loops).

The paths p_i^k (2) and flows ϕ_i^k on them are reconstructed from the solution of (C), expressed in the flow-per-link manner. The reconstruction consists in taking the path flow part in the mentioned superpositional representation of the network flow represented by the solution of problem (C) and neglecting the circulation part.

At the end of Step 3.1, the number of paths for a commodity is not yet limited by κ and the lower bounds on path flows are not yet applied.

The paths in the randomized rounding are chosen, semi-randomly, among the paths calculated in Step 3.1. The choice is steered, in a complex way, by the values ϕ_i^k . Generally, a high value of ϕ_i^k increases the probability of choosing path p_i^k . The details of rounding are given in Sect. 3.4. We shall now describe the steps of our algorithm.

3.2 Solving the continuous MCF subproblem

For integer i and j , let $\delta_{i,j} = 0$ for $i \neq j$ and $\delta_{i,j} = 1$ for $i = j$. The continuous MCF problem is defined as follows:

$$\text{minimize } \xi^{\max} \tag{C}$$

$$\text{over} \tag{7}$$

$$\phi_e^k \geq 0 \text{ for } e \in E'(k), k = 1, \dots, K$$

$$\varphi_e \geq 0 \text{ for } e \in E,$$

$$\xi^{\max} \geq 0 \tag{8}$$

subject to

$$\xi^{\max} \geq \varphi_e/c_e \text{ for } e \in E \tag{9}$$

$$\varphi_e = \sum_{k \in \{1, \dots, K\}: c_e \leq \phi^{lo}(k)} \phi_e^k \text{ for } e \in E \tag{10}$$

$$\delta_{i,s(k)} \cdot D(k) + \sum_{(j,i) \in E'(k)} \phi_{(i,j)}^k = \delta_{i,t(k)} \cdot D(k) + \sum_{(i,j') \in E'(k)} \phi_{(i,j')}^k \tag{11}$$

$$\text{for } i \in V, k = 1, \dots, K : E'(k) \neq \emptyset$$

$$\phi_{(i,j)}^k = 0 \text{ for } i, j \in V, k \in 1, \dots, K : (i, j) \in E'(k) \wedge i = s(k) \tag{12}$$

$$\phi_e^k \geq 0 \text{ for } k \in 1, \dots, K, e \in E'(k), \tag{13}$$

where $E'(k) = \{e \in E : c_e \geq \phi^{lo}(k)\}$ (we shall use this notation also later).

Problem (C) can be explained as follows: φ_e^k is the flow of commodity k in link e , φ_e —the total flow in link e . The Kirchoff law is expressed by (11). Equation (12) is an anti-circulation constraint needed for the proper convergence analysis in Remark 2 (it prevents a nonzero circulation of a commodity on a loop that would cross the commodity source node). Note that several problem parts use set $E'(k)$ of arches capable of carrying traffic $\phi^{lo}(k)$, instead of set E . The pruning of links in E serves for a numerical simplification of problem (C). The pruning is possible due to constraints (6) in problem (P). Namely, the rounding procedure is constructed so as to round any possible flow of commodity k through a link to a value not less than $\phi^{lo}(k)$, to satisfy (6). For each pruned link, such a rounding would cause its oversaturation, which should not happen, at least under the assumptions of our algorithm analysis in Sect. 4). There we consider only solutions not oversaturating the links, the most practically important case. (To allow solutions oversturating the links, we could simply resign from the acceleration: use E instead of $E'(k)$; the algorithm analysis would adjust quite trivially).

3.3 Reconstructing paths

Then, for each commodity k , objects η'_k, p_i^k, ϕ_i^k are reconstructed from the solutions φ_e^k (flows in links for commodities) by the rather folklore Algorithm 2.

Algorithm 2 Reconstruction of flow paths from flows for commodity k .

```

1:  $\eta'_k \leftarrow 0$ 
   Set  $l = 0$ ,
    $E'' = \{e \in E'(k) : \phi_e^k > 0\}$ ,  $V'' = \{v \in V : \exists(i, j) \in E : (v = i \vee v = j)\}$ 
    $G'' = (V'', E'')$  ( $G''$  is  $G$  after a rejection of links of capacity lower than the minimum demanded path flow  $\phi^{l_0}(k)$  for commodity  $k$ , and a consecutive rejection of non-linked nodes).
   Set  $\varphi''_e = \phi_e^k$ ,  $e \in E'$ 
2: while there exist a path in  $G''$  from  $s(k)$  to  $t(k)$  do
3:   Find a path  $p$  in  $G''$  from  $s(k)$  to  $t(k)$ , where  $p = (p_1, \dots, p_l)$ ,
      $p_1 = s(k)$ ,  $p_l = t(k)$  as a shortest path in under unit link weights.
     Reconstruct a new path:
      $l \leftarrow l + 1$ 
      $f \leftarrow \min_{e \text{ on } p} \varphi''_e$ 
      $p_1^k \leftarrow p$ 
      $\phi_1^k \leftarrow f$ 
4:   Strip the path flow from the graph:
      $\varphi''_e \leftarrow \varphi''_e - f$  for  $e \in E'$ 
      $E'' \leftarrow \{e \in E'' : \varphi''_e > 0\}$ ,  $V'' \leftarrow \{v \in V'' : \exists(i, j) \in E'' : (v = i \vee v = j)\}$ ,
      $G'' \leftarrow (V'', E'')$ 
5:    $\eta'_k \leftarrow l$ 
6: end while

```

Remark 1 Certainly, checking for the existence of path p in Step 3.3 and finding this p in Step 3.3 are really combined in one run of the Dijkstra method; these activities were separated in the algorithm depiction for clarity.

Remark 2 The following holds for Algorithm 2:

- (i) Algorithm 2 finishes its work after at most $|E|$ iterations of the while loop (with η'^k equal at most $|E|$).
- (ii) Then, after the algorithm is finished, there holds $\sum_{i=1}^{\eta_k} \phi_i^k = D(k)$.

The above remark comes from the common knowledge on decomposing network flows into paths and loops (see e.g. [9]). In each iteration of the “while” loop, we strip at least one link from E'' ; from this, (i) follows.

Property (ii) can be shown in the following reasoning. Consider three network flows in graph (V, E') : flow A is represented by the values of variables φ''_e immediately before the “while” loop is entered, flow B is defined by the values of φ''_e immediately after the “while” loop is exited and the flow C is defined by the superposition of flows ϕ_i^k on paths p_i^k for $i = 1, \dots, \eta'_k$ (after the “while” loop). The stripping is so defined that A equals the superposition of B and C, symbolically: $A = B + C$. Denote by $\mathcal{L}(A)$, $\mathcal{L}(B)$ and $\mathcal{L}(C)$ the sum of link flows leaving node $s(k)$ in network flow A, B or C, respectively. Constraints (11) enforce that $\mathcal{L}(A) = D(k)$. In turn, $\mathcal{L}(C) = \sum_{i=1}^{\eta_k} \phi_i^k$, obviously. By superposition, $\mathcal{L}(A) = \mathcal{L}(B) + \mathcal{L}(C)$. Now it suffices to show that $\mathcal{L}(B) = 0$, which we do in following way. Network flow B by the construction of stripping s a network flow conforming to the Kirchoff law, with some commodity

emission \mathcal{E} in node $s(k)$ and the same absorption \mathcal{E} in node $t(k)$ (a negative emission means an actual absorption and vice versa). The value \mathcal{E} cannot be negative, since nothing flows into $s(k)$: in flow A the constraint (12) held and the link flows in network flow B are not greater than the respective link flows in network flow A by the construction of stripping. If, in turn, \mathcal{E} were positive, network flow B could be decomposed into flows on paths leading from $s(k)$ to $t(k)$, at least one of them being positive, and some flows on loops (circles)—see e.g. Theorem 1.7.3 in [9]. But there was no path in G'' from s to t in the last check in the “while” loop. Therefore, \mathcal{E} must be zero, which completes the proof.

3.4 Randomized rounding

A round of the randomized rounding takes place in step 3.1 of Algorithm 1. The goal of the randomized rounding is to calculate the solution of our initial problem (P) represented as a collection of path flows based on the objects calculated in step 3.1. Namely, for $k = 1, \dots, K$ we calculate η_k (where $1 \leq \eta_k \leq \kappa$), paths $p_i^k \in P^k$ and reals $\phi_i^k \geq 0$ for $i = 1, \dots, \eta_k$ based on η'_k , $p_i^k \in P^k$, $\phi_i^k \geq 0$ for $k = 1, \dots, K$, $i = 1, \dots, \eta'_k$.

The randomized rounding takes place for each k -th commodity, in isolation from other commodities. The results of calculations for one commodity will not influence the calculations for another commodity. Thus, the order of considering commodities may be arbitrary, and our algorithm considers the commodities simply in the order of indexing. We shall now describe the randomized rounding for a given k .

3.4.1 The original Raghavan and Thompson’s randomized rounding

We shall first describe the original Raghavan and Thompson’s basic procedure in translation to our settings, for given k .

They have always $\eta_k = 1$ (since their problem is unsplitable). Thus only the single path p_1^k and the flow ϕ_1 must be evaluated.

Their randomized rounding goes in two steps:

1. Path choice. They choose p_1 as one of the paths p_i^k for $i = 1, 2, \dots, \eta'_k$ with the probability Γ_i of choosing path p_i^k given by $\Gamma_i = \phi_i^k / \sum_{j=1}^{\eta'_k} \phi_j^k = \phi_i^k / D(k)$.
2. Setting the flow. The flow ϕ_1 in path p_1 is simply chosen as equal to the $D(k)$, the demand for commodity k .

We may interpret the above process as follows. First, we round the flows ϕ_i^k for $i = 1, \dots, \eta'^k$:—one of them to $D(k)$ and the remaining ones—to zero. Then, we renumber the paths and flows: variables representing zero flows and their corresponding paths are removed from the considerations,

and the variables for the only nonzero flow and its path receive the index of 1. This view explains the name of “randomized rounding” and will be occasionally used in the further text.

3.4.2 Our method

For given k , our modified Raghavan and Thompson method runs in two steps:

1. Choice of candidate paths. We obtain a set $W \subseteq \{1, \dots, \eta'_k\}$ of indexes of paths such that $|W| \leq \kappa$. Paths p_j^k for $j \in W$ will candidate to become the paths p_i^k in the final solution (a further selection of the candidates will take place in the Step 2). Set W is obtained as the set of numbers chosen in either of κ independent random choices of a number in set $\{1, 2, \dots, \eta'_k\}$, where in each choice the probability Γ_i of choosing i is given by $\Gamma_i = \phi_i^k / \sum_{j=1}^{\eta'_k} \phi_j^k = \phi_i^k / D(k)$ (for $i = 1, \dots, \eta'_k$), as before.
2. Final choice of the paths; choice of the flows. For $i \in W$ we define verified flows: $\bar{\phi}_i^k = \max(\phi_i^k, \phi^{lo}(k))$. Now we take as many as possible paths among the candidating p_i^k with largest $\bar{\phi}_i^k$ so that the sum of $\bar{\phi}_i^k$ s for the chosen paths is not greater than $D(k)$. Let j^* be the number of the chosen paths and S be the sum of their verified flows $\bar{\phi}_i^k$.

We finally choose η_k as j^* , p_i^k for $i = 1, \dots, \eta_k$ as the chosen paths, and ϕ_i^k for $i = 1, \dots, \eta_k$ as the respective values $\bar{\phi}_i^k$ multiplied by

$$\frac{D(k)}{S}. \tag{14}$$

Remark 3 The $\frac{D(k)}{S}$ coefficient in (14) ensures that the path flows for commodity k in the final solution sum up to $D(k)$.

Remark 4 Since, clearly, $S \leq D(k)$, the computed ϕ_i^k satisfy our initial demand $\phi_i^k \geq \phi^{lo}(k)$.

Note that when $\kappa = 1$, our randomized rounding becomes essentially the same as that of Raghavan and Thompson.

Moreover, we can interpret our procedure as rounding η_k flows ϕ_i^k up and the remaining of them—down to zero and a subsequent neglecting/renumbering of variables.

Our method is a simple heuristics: the mutual ratios of the calculated flows ϕ_i^k try to follow the corresponding ratios of the ϕ_i^k . Asymptotically, for large κ and for a low lower bound $\phi^{lo}(k)$, the collection of flows ϕ_i^k after rounding becomes close to the collection of flows ϕ_i^k before rounding. Constructing set W so that $|W| \leq \kappa$ is important for the solution

quality analysis. It allows assessing the probability for flow ϕ_i^k being rounded up by the value $\kappa \phi_i^k / D(K)$, only by a constant factor κ greater than in the Raghavan and Thompson’s case.

3.5 A modification

The Raghavan and Thompson’s randomized rounding has a clear sense when there are many flow paths coming through a link immediately before the rounding and for these paths, the values of $D(k)$ for the respective k s remain small compared to the link capacity. The randomized rounding is constructed in such a way that, in such circumstances, the sum of the flows in the link before the rounding is a good approximation of the sum of these flows after the rounding. When $D(k)$ for some flow path of commodity k is comparable with or greater than the link capacity, the quantization effects of the rounding become notable, the approximation gets inaccurate and there is a big chance that the link capacity will be seriously exceeded after the rounding, especially when $\kappa = 1$ (when rounding up means rounding to as much as $D(k)$). In order to decrease the probability of rounding such continuous flows to $D(k)$ s, we make these flows smaller by penalizing them already in problem (C) in the following modification of our method:

Modification 1 (to problem (C)) We replace (10) with

$$\varphi_e = \sum_{k \in \{1, \dots, K\}: c_e \leq \phi^{lo}(k)} \sigma_e^k \cdot \phi_e^k \text{ for } e \in E \tag{15}$$

where

$$\begin{aligned} \sigma_e^k &= 1 \text{ for } k \in \mathcal{K}_e^b \equiv \{k \in K : D(k) < \alpha c_e\}, \\ \sigma_e^k &= \sigma \equiv 2\kappa|E|/\alpha \text{ for } k \in \mathcal{K}_e^\# \equiv \{k \in K : D(k) \geq \alpha c_e\} \end{aligned} \tag{16}$$

and

$$\alpha = \frac{\sqrt{2}}{\sqrt{\ln(8|E|)} + \sqrt{2}\kappa}. \tag{17}$$

Here σ_e^k are the penalty coefficients, sets \mathcal{K}_e^b and $\mathcal{K}_e^\#$ are the sets of commodities k with “small” and “large” $D(k)$, respectively, compared to the link capacity c_e .

3.6 Run time

We shall discuss the run time of Algorithm 1. Let, reasonably $|E| \geq |V|$. Choosing a random real number will be assumed to be an elementary operation.

Setting up problem (C) in Step 3.1 can be realized so as to take no more than $\mathcal{O}(K \cdot (|E| + |V|))$ time. For each commodity, at most $|E|$ paths are reconstructed from link flows in

the solution of (C) by Remark 2 (i). In an effective implementation, the reconstruction time is dominated by at most $|E|$ Dijkstra method runs, each costing at most $\mathcal{O}(|E| \cdot \log |V|)$. The reconstruction costs thus at most $\mathcal{O}(|E|^2 \log |V|)$. Consider the time of Step 3.1, i.e., of the randomized rounding. Computing the numbers Γ_i costs at most $\mathcal{O}(|E|)$ (in a given round, for a given commodity) since at most $|E|$ paths have been reconstructed from the continuous flow. The computation of an element of W in a single random choice can be done within an $\mathcal{O}(|E|)$ time; the computation of $|W|$ is thus possible within the time of $\mathcal{O}(\kappa \cdot |E|)$. A single final choice of paths is in a reasonable implementation dominated by sorting $|W| \leq \kappa$ numbers, which can be done in $\mathcal{O}(\kappa \log \kappa)$ time. The time of Step 3.1 of Algorithm 1 is clearly dominated by the setup time for problem (C). Putting all these observations together, we obtain the bound on the run time of our method of

$$\mathcal{O}(Kr \cdot (|E|^2 \log |V| + \kappa \log \kappa)) + \text{external solver run time.} \quad (18)$$

However, the assumption that at most $|E|$ paths are reconstructed for a given commodity is very conservative; in practice, rather a few paths (say, a constant number) per commodity are reconstructed, which makes the practical time closer to $\mathcal{O}(Kr \cdot (|E| \log |V| + \kappa \log \kappa)) + \text{external solver run time.}$

4 Solution quality

We shall consider the most practically important question of how we can guarantee that the algorithm finds a realizable solution, in which no link is overloaded (i.e., with the goal function not greater than 1).

The algorithm has been designed to only give approximate solutions. Thus it is possible that the algorithm yields a non-realizable solution while a realizable one exists for the problem. Fortunately, networks are nowadays often oversized, the link capacities are fairly greater than required for the existence of a realizable solution. This existing network oversizing can compensate for the proximity of the algorithm and make the algorithm again obtain a realizable solution. We are going to formally analyze such a compensation.

We shall show the following. Assume that for some problem (P) with $\kappa = 1$, a solution with the goal value less than α exists, where $\alpha \leq 1$ is given by (17). This assumption means that our network is already oversized so that realizing an unsplitable transfer of our demands with the highest link saturation of α is possible. Then, our algorithm finds a realizable solution (i.e., with the goal value not more than 1) for the same problem with any setting of splittability level κ , with a probability that can be drawn arbitrarily close to 1 by increasing the number of rounds r .

To show the above, we shall first analyze a solution yielded by a single round of randomized rounding, i.e., a single pass of step 3.1 of Algorithm 1. We shall prove that, under the sufficient network oversizing, the goal value for its solution is not more than 1 with at least a constant, positive probability. This is sufficient, since then we can apply the well known Las Vegas randomized scheme: by increasing the number r of independent rounds we can quickly draw the probability that any of the solutions of the rounds has the goal value not exceeding 1 arbitrarily close to 1. Then it remains to take the best (lowest-goal-value) of the solutions of the rounds as the final solution, as is actually done in Algorithm 2.

From now until further notice, we shall consider a single round.

In the solution of a round, a link is occupied by flows of the paths (obtained by the reconstruction by Algorithm 2 for the commodities) rounded randomly either to zero or to some positive value. Thus, the final link occupancy may be viewed as a sum of realizations of some random variables. In the further analysis, we shall need a probabilistic description of how well the sum of realizations of random variables can be assessed by the sum of the expectations of the variables. Our main probabilistic tool will be thus the following theorem [15]:

Theorem 1 For $j = 1, \dots, N$ let $X_j \in \{0, 1\}$ denote independent random variables with means \mathfrak{p}_j and let $|\theta_j| < \infty$. Then for all $\epsilon \geq 0$

$$\Pr \left[\left| \frac{1}{N} \sum_{j=1}^N \theta_j \cdot (X_j - \mathfrak{p}_j) \right| > \epsilon \right] \leq 2 \exp \left(-N \epsilon^2 / \chi^2 \right), \quad (19)$$

where $\chi^2 = \frac{1}{N} \sum_{j=1}^N \theta_j^2 \Phi(\mathfrak{p}_j)$ and $\Phi(x) = (1-2x) / \ln((1-x)/x)$ for $x \in (0, 1)$, $\Phi(0) = \Phi(1) = 0$.

To prove that a round of our methods likely yields a solution of (P) with a goal value not greater than 1 under the considered network oversizing, we shall first need to show that the optimal value of the relaxation (C) of problem (P) with Modification 1 is less than α under this oversizing.

Lemma 1 For some problem (P), let α be defined by (17). If the optimal value of the alteration of problem (P) consisting in setting κ to 1 is less than α then the continuous problem (C) with Modification 1 resulting from the unaltered problem (P) has the optimal value less than α .

Proof Take any link $e \in E$. Let η_k, p_i^k, ϕ_i^k for $i = 1, \dots, \eta_k, k = 1, \dots, K$ be taken from an optimal solution of the alteration of problem (P). Each path p_i^k of them that goes through link e must have $k \in \mathcal{K}_e^b$. Otherwise, since the corresponding ϕ_i^k would be equal to $D(k)$ (because $\kappa = 1$), and

since $D(k)$ is from the definition of \mathcal{K}_e^\sharp not less than αc_e , the total flow in link e would be at least αc_e which would contradict the assumption about the optimal value of the alteration of problem (P). A similar reasoning can be made for all links $e \in E$.

The optimal solution of the alteration of problem (P) can be translated into a feasible solution of the problem (C) without Modification 1 resulting from problem (P), I.e., it can be expressed in terms of link flows rather than path flows (the mapping is trivial when we realize that no path for commodity k cannot flow through a link in $E \setminus E'(k)$ in the solution of the alteration of (P), since, by (6), yielding the goal value above 1). In the so translated solution the total flow in each link e is less than αc_e . Thus, switching on Modification 1 does not change the value of goal function of the problem (C) for this solution (because all the sets \mathcal{K}_e^\sharp are empty). Thus the translated solution is a feasible solution of the resulting problem (C) with Modification 1 yielding the goal function value below α . \square

Now that we can assess the optimal value of (C), it remains to take into account the rounding effects during the randomized rounding step. Our main result is contained in the following theorem, stating that under the sufficient oversizing of the network a round of our method with a guaranteed constant positive probability finds a solution not exceeding the capacities of the links.

Theorem 2 *We have problem (P) with $|E| > 1$. Let $\alpha = \frac{\sqrt{2}}{\sqrt{\ln(8\|E\|)} + \sqrt{2}\kappa}$, as in (17). Then with probability at least $1/4$ a round of randomized rounding in our method with Modification 1 finds a feasible solution of the problem yielding the value of goal function not greater than 1 on condition that the alteration of problem (P) consisting in setting κ to 1 has the optimal value less than α .*

Proof That our method finishes returning a solution in the described situation is a consequence of Remark 2(i) and the feasibility of generated problem (C) with Modification 1, implied by Lemma 1. The feasibility of the returned solution in problem (P) follows from Remarks 3 and 4 and from the construction of the Algorithm 2, that yields positive ϕ_i and $\eta_k \leq \kappa$. The claim about the goal value in (P) for this solution will be proved in steps.

1. Consider link $e \in E$ and these paths among p_i^k calculated in Step 3.1 of Algorithm 2 with $k \in \mathcal{K}_e^\sharp$ that go through link e , together with the corresponding flows ϕ_i^k (\mathcal{K}_e^\sharp is defined as in Modification 1). In the course of randomized rounding, these flows are rounded either down—to zero or up—to a positive value not greater than $D(k)$. Let us assess the probability of rounding flow ϕ_i^k (the flow on path p_i^k) up. It is not greater than the probability for i to become the member of set W (a candidate

for rounding up) in our randomized rounding, which is, in turn, not greater than $\kappa \phi_i^k / D(k)$.

Consequently, the probability Π^\sharp of rounding any of the considered flows (i.e., flows ϕ_i^k with $k \in \mathcal{K}_e^\sharp$ and p_i^k coming through e) is not greater than

$$\kappa \sum_{\substack{k \in \mathcal{K}_e^\sharp \\ i=1, \dots, \eta_k \\ e \text{ on } p_i^k}} \phi_i^k / D(k).$$

Algorithm 2 reconstructs paths from the solution of the subproblem (C) in such a way that for each $k \in 1, \dots, K$ $\sum_{k \in \mathcal{K}_e^\sharp, i=1, \dots, \eta_k, e \text{ on } p_i^k} \phi_i^k \leq \varphi_e^k$ where φ_e^k is a part of the solution of the subproblem (C) by the construction of graph flow stripping. Thus,

$$\Pi^\sharp \leq \kappa \sum_{k \in \mathcal{K}_e^\sharp} \varphi_e^k / D(k)$$

and, using the definition of set \mathcal{K}_e^\sharp

$$\Pi^\sharp \leq \kappa \sum_{k \in \mathcal{K}_e^\sharp} \varphi_e^k / (\alpha c_e).$$

Now we will need an assessment of $\sum_{k \in \mathcal{K}_e^\sharp} \varphi_e^k$. By (15), and because the optimal value of subproblem (C) is from Lemma 1 less than α , this is not greater than c_e / σ , where σ is defined as in (16). Consequently,

$$\Pi^\sharp \leq \frac{\kappa}{\alpha \sigma} = \frac{\kappa}{\alpha \cdot 2\kappa |E| / \alpha} = 1 / (2|E|).$$

and, finally

$$\Pi^\sharp \leq 1 / (2|E|).$$

2. Now we shall consider an arbitrary link e and the paths p_i^k calculated in Step 3.1 of Algorithm 1 with $k \in \mathcal{K}_e^\flat$ that lead through link e , together with the corresponding flows ϕ_i^k (\mathcal{K}_e^\flat is defined as in Modification 1). The sum of the above flows after the rounding is done on them will be equal to

$$\sum_{k \in \mathcal{K}_e^\flat} \sum_{i=1, \dots, \eta_k: e \text{ on } p_i^k} \phi_i^k,$$

where η^k and ϕ_i^k are taken from the solution of (P) yielded by the round. This sum can be expressed as:

$$\sum_{k \in \mathcal{K}_e^\flat} Y_k,$$

where $Y_k = \sum_{i=1, \dots, \eta_k: e \text{ on } p_i^k} \phi_i^k$.

The random variables Y_k are independent, since the randomized rounding is performed independently for each commodity k . It is clear that the values Y_k are all within $[0, D(k)]$. The probability of rounding a flow ϕ_i^k (of the flows being considered) up is not less than $\kappa \phi_i^k / D(k)$, due to a similar reasoning as given earlier. Therefore

$$\Pr(Y_k > 0) \leq \kappa \sum_{i=1, \dots, \eta_k: e \text{ on } p_i^k} \phi_i^k / D(k) \leq \kappa \varphi_e^k / D(k),$$

where φ_e are from the solution of problem (C). Thus, the random variable Y_k is majorized by random variable $\theta_k \cdot X_k$, where $\theta_k = D(k)$ and X_k is a binary variable equal to 1 if $Y_k > 0$ and to zero when $Y_k = 0$. We have $\mathbf{p}_k \equiv \Pr(Y_k > 0) = \Pr(X = 1) \leq \kappa \varphi_e^k / D(k)$ and, certainly, variables X_k are independent since they are functions of independent variables Y_k .

We are now interested in probability Π^b that $\sum_{k \in \mathcal{K}_e^b} \sum_{i=1, \dots, \eta_k: e \text{ on } p_i^k} \phi_i^k$ in the solution of (P) yielded by the round exceeds c_e .

$$\begin{aligned} \Pi^b &\leq \Pr \left(\sum_{k \in \mathcal{K}_e^b} Y_k > c_e \right) \leq \Pr \left(\sum_{k \in \mathcal{K}_e^b} \theta_k X_k > c_e \right) \\ &= \Pr \left(\sum_{k \in \mathcal{K}_e^b} \theta_k \mathbf{p}_k + \sum_{k \in \mathcal{K}_e^b} \theta_k (X_k - \mathbf{p}_k) > c_e \right) \\ &\leq \Pr \left(\sum_{k \in \mathcal{K}_e^b} \kappa \theta_k \varphi_e^k / D(k) + \sum_{k \in \mathcal{K}_e^b} \theta_k (X_k - \mathbf{p}_k) > c_e \right). \end{aligned}$$

But from the assumption and from Lemma 1, $\sum_{k \in \mathcal{K}_e^b} \varphi_e^k \leq \alpha c_e$ and $\theta_k = D(k)$. Thus

$$\begin{aligned} \Pi^b &\leq \Pr \left(\kappa \alpha c_e + \sum_{k \in \mathcal{K}_e^b} \theta_k \cdot (X_k - \mathbf{p}_k) > c_e \right) \\ &= \Pr \left(\sum_{k \in \mathcal{K}_e^b} \theta_k \cdot (X_k - \mathbf{p}_k) > c_e \cdot (1 - \alpha \kappa) \right) \\ &= \Pr \left(\frac{1}{|\mathcal{K}_e^b|} \sum_{k \in \mathcal{K}_e^b} \theta_k \cdot (X_k - \mathbf{p}_k) > \frac{c_e \cdot (1 - \alpha \kappa)}{|\mathcal{K}_e^b|} \right). \end{aligned}$$

Now, using Theorem 1

$$\Pi^b \leq 2 \exp \left(-|\mathcal{K}_e^b| \frac{c_e^2 (1 - \alpha \kappa)^2}{|\mathcal{K}_e^b|^2 / \chi^2} \right),$$

where $\chi^2 = \frac{1}{|\mathcal{K}_e^b|} \sum_{k \in \mathcal{K}_e^b} \theta_k^2 \Phi(\mathbf{p}_k)$ and $\Phi(x) = (1 - 2x / \ln((1 - x)/x))$ for $x \in (0, 1)$ and $\Phi(0) = \Phi(1) = 0$. But $\Phi(x) \leq \frac{1}{2}$. Thus

$$\begin{aligned} \Pi^b &\leq 2 \exp \left(-|\mathcal{K}_e^b| \frac{c_e^2 (1 - \alpha \kappa)^2 |\mathcal{K}_e^b| \cdot 2}{|\mathcal{K}_e^b|^2 \cdot \left(\sum_{k \in \mathcal{K}_e^b} D^2(k) \right)} \right) \\ &\leq 2 \exp \left(-\frac{2c_e^2 (1 - \alpha \kappa)^2}{|\mathcal{K}_e^b| \cdot \alpha^2 c_e^2} \right) \leq 2 \exp \left(-\frac{2(1 - \alpha \kappa)^2}{\alpha^2} \right). \end{aligned}$$

By substituting $\alpha = \frac{\sqrt{2}}{\sqrt{\ln(8|E|)} + \sqrt{2\kappa}}$ from the definition and performing a few further calculations, we obtain

$$\Pi^b \leq 1/(4|E|).$$

3. Let us assess the probability Π that the total flow in link e is above c_e in the solution yielded by the round. The sufficient condition for this to hold is that any flow in e for a commodity from \mathcal{K}_e^\sharp is rounded up or $\sum_{k \in \mathcal{K}_e^b} \sum_{i=1}^{\eta_k} \phi_i^k > c_e$. The the two parts of this alternative are precisely that analyzed in the two previous steps of the proof. Consequently, $\Pi \leq \Pi^\sharp + \Pi^b$.

The goal value is less than κ with probability at least $1/4$ iff the total flow in any link $e \in E$ does not exceed c_e in the solution given by the round. The probability of the later is clearly not greater than $|E| \cdot \Pi$.

So, it now suffices to prove that $|E| \cdot \Pi \leq 3/4$. We have $|E| \cdot \Pi \leq |E| \cdot (\Pi^\sharp + \Pi^b) \leq |E| \cdot (1/(4E) + 1/(2E)) = 3/4$. \square

Now we switch to considering the whole Algorithm 1, with many rounds. The solution of Algorithm 1 is the best round solution. Thus, not founding a solution with the goal value not exceeding 1 by Algorithm 1 means that no round has found it. The probability of the latter, assuming the same as in Theorem 2, is clearly $(3/4)^r$, since the rounds are independent.

Corollary 1 *Under the assumptions of Theorem 2, the whole method returns a feasible solution of (P) of the goal value not exceeding 1 with probability $1 - (3/4)^r$.*

5 Experiments

The performed experiments aimed at observing the practical solution quality of the method as well as the reaction of the computation time to changes in the problem sizes.

The method, with Modification 1, has been implemented in C++. The implementation, called Dim, uses Gnu GLPK v. 4.52 (see [17]) as an external solver for the continuous MCF

subproblems. Finding shortest paths in the path reconstruction phase is done using the Bellman-Ford algorithm with heaps (see [6]) being emulated by C++ Sets, of a similar efficiency.

We have resigned acquiring a specific benchmark with a minimum-congestion problem and lower flow bounds and a specific competitor solver for such problems. We have instead decided to use a series of artificial network topologies generated by the well recognized BRITE generator [20].

We assess our method in a threefold way. Firstly, we compare it to the GNU GLPK solver applied to a Mixed Integer Programming Problem (MILP) formulation of the problem instance. The GLPK solver was stopped when the absolute value of the relative error in the calculated optimal value could be guaranteed not to exceed ± 0.03 . This comparison clearly works only for small problems, since the MILP formulation is very difficult. Thus we always assess the optimal value of problem (P) from below by the optimal value of its relaxation (C). It may be seen as a solution of (P) with $\kappa = \infty$. Using (C) as a relaxation is correct as long as the optimal value of a problem (P) is not more than 1. This is because using set E' instead of E in (C) is not a larger restriction on the network flow than constraints (6) in (P): no path in the solution of (P) goes through a link not satisfying the appropriate (6). Also, we solve all the instances of (P) with our method without Modification 1 (we call this method “Dim base”). In this way, we can at least assess the effect of introducing our “anti-quantization improvement” (as well as compare the solution times). Actually, we may expect “Dim base” to sometimes give better solutions than the more conservative “Dim”, which, however, has some quality guarantees. It is valuable to see their practical comparison.

The generation of problem instances can be outlined as follows. We use a topology generated by BRITE due to the Waxman model with the node average degree parameter RT_m equal to 5. The number of nodes n is a problem instance parameter, varied in the experiments. The commodities are defined in the relations of consecutive pairs of different nodes, until the presumed number K of commodities is reached. If all the pairs have been exhausted before reaching K , the consecutive relations reuse the pairs in the Round-Robin fashion. The link capacities are chosen randomly and independently from interval $[1E3, 1E5]$ with a uniform distribution. The demand $D(k)$ for each k th commodity is chosen randomly and independently on other choices from $[0.01 \cdot \tilde{D}, \tilde{D}]$, where \tilde{D} —a positive parameter. Their logarithms have a uniform distribution. The default number of rounds $r = 100$ has been always used.

Two experiments have been performed:

1. Experiment A. A series of problem instances with different n has been generated. In an instance, the number m of links was chosen by BRITE as approximately $RT_m \cdot n$,

the number of relations was chosen as $K = n \cdot (n - 1)$. Note that here the complexity of the problem grew very quickly with n . The maximum demand parameter \tilde{D} has been chosen so as to keep the optimal value of the continuous relaxation of the problem equal to 0.5.

2. Experiment B. There have been constant settings $n = 20$, $m = 80$, $\tilde{D} = 1428.6$ and K was varied. This series has allowed observing both the reaction of solution quality for various network saturation levels and the reaction of the computing time on K .

The experiments were run on a Dell E640 Latitude with the Intel i7-3740QM CPU and 8GB RAM, running the Fedora 20 Linux.

The results are shown in Tables 1, 2 and Figures 1, 2, 3, 4, 5, and 6. Results are marked with ‘-’ whenever the corresponding solver run was not performed due to an excessive solution time expected. The goal values for the ξ^{\max} the found solutions are denoted simply by ξ^{\max} .

The measured approximation ratio of our method, understood as the goal value of the obtained solution divided by the goal value of the exact solution, is not worse than 1.5. For example, in Experiment A, even when we limit our attention to $\kappa = 1$, the obtained goal value for our method (Dim) exceeds the optimal goal value for the relaxation most for $n = 7$ (compare Table 1; Fig. 2). The ratio of the values is then $0.623/0.500 = 1.246$. The optimal value for the relaxation is only the underestimation of the true optimal problem value, so the real solution approximation factor can be lower than 1.246. In the near problem instance with $k = 7, \kappa = 1$, we know the true optimal value of the problem, computed by GLPK for the MILP formulation; the approximation factor is $0.517/0.503 = 1.028$ there. Interestingly, the (upper assessment of) approximation ratio seems have a peak in $n = 8$ and approaches 1 for lower or greater values of n . For example, already for $n = 6$ or for $n = 30$, our method seems to find an exact solution, with respect to the 4-meaningful-digit accuracy of the goal value registration taken in our experiments. This behavior might be explained as follows. Small problems are simple, there are a few possible paths for a commodity and randomization in 100 performed rounds may have a similar effect as a “brute-force” searching of the space of possible solutions. In turn, for large problems, there are many commodities per link on average. Thus many path flows come through a single link in the solution of the continuous relaxation; thus the effects of rounding the particular flows cancel out statistically (which is a fundamental of the randomized rounding method). Despite that we have performed our experiments on a particular class of problems, this behavioral pattern may be expected to reproduce for many other problem classes. The highest upper assessment of approximation ratio in Experiment B for $\kappa = 1$ seems to appear for $K = 50$ and equal

Table 1 Results of experiment A

n	m	K	\bar{D}	ϕ^{lo}	κ	Time (s) Dim	Time (s) Dim base	Time (s) MILP/GLPK	ξ^{max} Dim	ξ^{max} Dim base	ξ^{max} MILP/GLPK	ξ^{max} Relaxation ($\kappa = \infty$)
5	4	20	1000	1	1	0.02	0.02	0.01	0.500	0.500	0.500	0.500
5	4	20	1135.5	1.135	2	0.06	0.06	0.04	0.500	0.500	0.500	0.500
5	4	20	1135.5	1.135	3	0.04	0.06	0.05	0.500	0.500	0.500	0.500
5	4	20	1135.5	1.135	4	0.04	0.06	0.06	0.500	0.500	0.500	0.500
5	4	20	1135.5	1.135	5	0.06	0.06	0.08	0.500	0.500	0.500	0.500
6	8	30	1333.3	1.333	1	0.05	0.08	0.05	0.500	0.500	0.500	0.500
6	8	30	2903.5	2.904	2	0.05	0.05	0.05	0.500	0.500	0.500	0.500
6	8	30	2903.5	2.904	3	0.12	0.1	0.07	0.500	0.500	0.500	0.500
6	8	30	2903.5	2.904	4	0.07	0.1	0.16	0.500	0.500	0.500	0.500
6	8	30	2903.5	2.904	5	0.07	0.15	0.19	0.500	0.500	0.500	0.500
7	14	42	1666.7	1.667	1	0.16	0.16	0.29	0.517	0.508	0.503	0.500
7	14	42	1811.1	1.811	2	0.12	0.11	1.32	0.500	0.500	0.500	0.500
7	14	42	1811.1	1.811	3	0.12	0.09	1.99	0.500	0.500	0.500	0.500
7	14	42	1811.1	1.811	4	0.17	0.2	–	0.500	0.500	–	0.500
7	14	42	1811.1	1.811	5	0.21	0.2	–	0.500	0.500	–	0.500
8	20	56	1785.7	1.786	1	0.17	0.17	4.76	0.623	0.799	0.560	0.500
8	20	56	10,986.3	10.986	2	0.2	0.22	44.94	0.587	0.609	0.502	0.500
8	20	56	10,986.3	10.986	3	0.17	0.17	–	0.561	0.514	–	0.500
8	20	56	10,986.3	10.986	4	0.24	0.25	–	0.500	0.511	–	0.500
8	20	56	10,986.3	10.986	5	0.21	0.27	–	0.500	0.500	–	0.500
9	27	72	2500.0	2.500	1	0.29	0.29	–	0.677	0.697	–	0.500
9	27	72	9492.1	9.492	2	0.3	0.32	–	0.546	0.500	–	0.500
9	27	72	9492.1	9.492	3	0.33	0.39	–	0.500	0.505	–	0.500
9	27	72	9492.1	9.492	4	0.38	0.37	–	0.504	0.500	–	0.500
9	27	72	9492.1	9.492	5	0.34	0.36	–	0.500	0.500	–	0.500

Table 1 continued

n	m	K	\bar{D}	ϕ^{lo}	κ	Time (s) Dim	Time (s) Dim base	Time (s) MILP/GLPK	ξ^{max} Dim	ξ^{max} Dim base	ξ^{max} MILP/GLPK	ξ^{max} Relaxation ($\kappa = \infty$)
10	32	90	2222.2	2.222	1	0.43	0.48	-	0.615	0.536	-	0.500
10	32	90	7051.8	7.052	2	0.45	0.42	-	0.504	0.504	-	0.500
10	32	90	7051.8	7.052	3	0.49	0.42	-	0.500	0.504	-	0.500
10	32	90	7051.8	7.052	4	0.51	0.53	-	0.500	0.501	-	0.500
10	32	90	7051.8	7.052	5	0.46	0.42	-	0.500	0.501	-	0.500
20	80	380	1052.6	1.053	1	21.6	9.02	-	0.548	0.551	-	0.500
20	80	380	4245.7	4.246	2	25.94	9.07	-	0.523	0.519	-	0.500
20	80	380	4245.7	4.246	3	31.4	8.99	-	0.505	0.507	-	0.500
20	80	380	4245.7	4.246	4	30.55	8.75	-	0.508	0.505	-	0.500
20	80	380	4245.7	4.246	5	31.99	8.68	-	0.500	0.505	-	0.500
30	120	870	689.7	0.690	1	221.13	60.6	-	0.500	0.500	-	0.500
30	120	870	119.9	0.120	2	247.2	59	-	0.500	0.500	-	0.500
30	120	870	119.9	0.120	3	272.51	61.93	-	0.500	0.500	-	0.500
30	120	870	119.9	0.120	4	280.73	59.5	-	0.500	0.500	-	0.500
30	120	870	119.9	0.120	5	227.18	60.42	-	0.500	0.500	-	0.500
40	160	1560	512.8	0.513	1	1349.31	265.87	-	0.500	0.513	-	0.500
40	160	1560	510.7	0.511	2	1605.47	271.94	-	0.500	0.500	-	0.500
40	160	1560	510.7	0.511	3	2132.59	276.51	-	0.500	0.500	-	0.500
40	160	1560	510.7	0.511	4	1401.28	275	-	0.500	0.500	-	0.500
40	160	1560	510.7	0.511	5	1230.88	266.56	-	0.500	0.500	-	0.500
50	200	2450	408.2	0.408	1	7900.29	973.85	-	0.500	0.500	-	0.500
50	200	2450	65.7	0.066	2	7724.09	978.75	-	0.500	0.500	-	0.500
50	200	2450	65.7	0.066	3	8447.86	966.97	-	0.500	0.500	-	0.500
50	200	2450	65.7	0.066	4	6328.29	967.48	-	0.500	0.500	-	0.500
50	200	2450	65.7	0.066	5	5070.51	973.56	-	0.500	0.500	-	0.500

Table 2 Results of experiment B

n	m	K	\tilde{D}	ϕ^{lo}	κ	Time (s) Dim	time [s] Dim base	Time (s) MILP/GLPK	ξ^{max} Dim	ξ^{max} Dim base	ξ^{max} MILP/GLPK	ξ^{max} Relaxation ($\kappa = \infty$)
20	80	50	1428.6	1.43	1	0.9	0.81	–	0.0553	0.0517	–	0.0380
20	80	50	1428.6	1.43	2	0.94	0.74	–	0.0440	0.0445	–	0.0380
20	80	50	1428.6	1.43	3	1.13	0.8	–	0.0434	0.0440	–	0.0380
20	80	50	1428.6	1.43	4	1.09	0.7	–	0.0388	0.0399	–	0.0380
20	80	50	1428.6	1.43	5	1.04	0.82	–	0.0394	0.0392	–	0.0380
20	80	100	1428.6	1.43	1	2.33	1.52	–	0.0675	0.0677	–	0.0560
20	80	100	1428.6	1.43	2	2.73	1.54	–	0.0589	0.0647	–	0.0560
20	80	100	1428.6	1.43	3	3.09	1.52	–	0.0621	0.0600	–	0.0560
20	80	100	1428.6	1.43	4	3.03	1.54	–	0.0584	0.0586	–	0.0560
20	80	100	1428.6	1.43	5	2.86	1.49	–	0.0580	0.0580	–	0.0560
20	80	200	1428.6	1.43	1	8.62	4.82	–	0.1084	0.1158	–	0.0893
20	80	200	1428.6	1.43	2	10.74	4.5	–	0.1033	0.1037	–	0.0893
20	80	200	1428.6	1.43	3	11.5	4.58	–	0.0939	0.1012	–	0.0893
20	80	200	1428.6	1.43	4	10.26	4.63	–	0.0954	0.0942	–	0.0893
20	80	200	1428.6	1.43	5	12.21	4.48	–	0.0931	0.0931	–	0.0893
20	80	400	1428.6	1.43	1	27.12	9.58	–	0.1858	0.2127	–	0.1689
20	80	400	1428.6	1.43	2	34.62	9.54	–	0.1713	0.1793	–	0.1689
20	80	400	1428.6	1.43	3	32.48	9.53	–	0.1701	0.1775	–	0.1689
20	80	400	1428.6	1.43	4	33.7	10.24	–	0.1697	0.1753	–	0.1689
20	80	400	1428.6	1.43	5	37	9.71	–	0.1706	0.1733	–	0.1689
20	80	800	1428.6	1.43	1	97.1	33.8	–	0.3223	0.3295	–	0.3078
20	80	800	1428.6	1.43	2	143.66	34.8	–	0.3165	0.3173	–	0.3078
20	80	800	1428.6	1.43	3	139.27	34.22	–	0.3114	0.3142	–	0.3078
20	80	800	1428.6	1.43	4	187.41	34.92	–	0.3104	0.3095	–	0.3078
20	80	800	1428.6	1.43	5	202.78	33.88	–	0.3100	0.3100	–	0.3078
20	80	1600	1428.6	1.43	1	427.52	125.72	–	0.6388	0.6584	–	0.6349
20	80	1600	1428.6	1.43	2	624.02	113.99	–	0.6411	0.6457	–	0.6349
20	80	1600	1428.6	1.43	3	691.63	115.86	–	0.6363	0.6409	–	0.6349
20	80	1600	1428.6	1.43	4	901.79	114.27	–	0.6352	0.6391	–	0.6349
20	80	1600	1428.6	1.43	5	930.09	112.84	–	0.6395	0.6350	–	0.6349

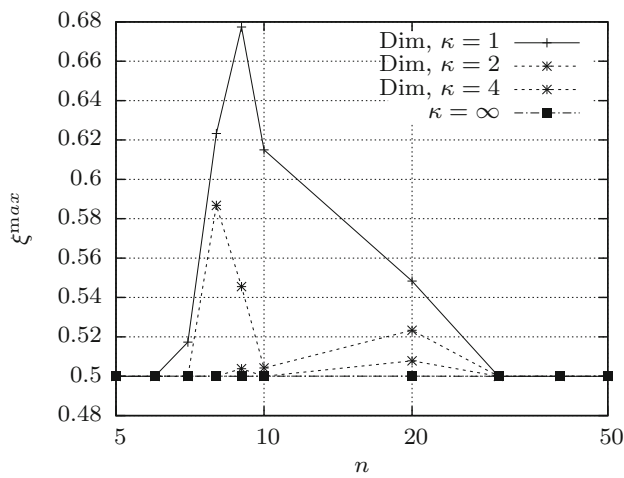


Fig. 1 Solution quality for different κ and network sizes in experiment A

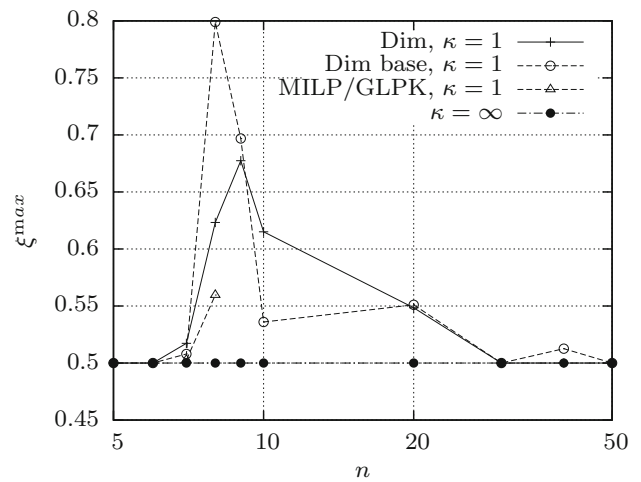


Fig. 2 Solution quality for different methods and network sizes in experiment A

$0.0553/0.0380 = 1.46$ and in general with the increase of the number of commodities K , seem to fall, as expected, becoming 1.04 for $K = 1600$. The approximation may be considered as practically satisfactory. Note that the practical approximation ratios are considerably better than our theoretical estimates. For example, for experiment B, $K = 50$, $\kappa = 1$, we have $\alpha = \sqrt{2}/(\sqrt{\ln(8|E|)} + \sqrt{2}\kappa) = 0.55$ (compare (17)), therefore, by Theorem 2, we would oversize the network by $1/\kappa = 1.79$ and Corollary 1 in order not to overload links with probability at least $(1 - (3/4)^{100})$. The factor 1.79 is greater than the observed upper assessment of the approximation ratio equal to 1.46 for the problem instance.

Increasing the maximum path splittability level κ usually considerably lowers the goal value of the solution calculated with our method (see, for example, Fig. 1, Table 1; Table 2). This behavior stops at κ for which the goal value of the solution is near the optimal value of the relaxation, say, higher not more than by 4 percent, and thus any substantial improvement is clearly not possible. In most cases, such a proximity has been achieved for κ between 1 and 4.

Our method seems most frequently yield lower ξ^{\max} than its “Dim base” version, though this behavior inverses for small problems. The inversion could be explained with the influence Modification 1 that with a considerable probability excludes some narrow links from conducting large-volume commodities. This exclusion has a conservative character and, for small problems, an exhaustive, almost “brute-force” searching the solution space may bring better results.

For big problems, our method, Dim, is considerably slower than Dim base. Modification 1 turns out complicate the optimization subproblems. Although the practical complexity of continuous optimization problems is hard to

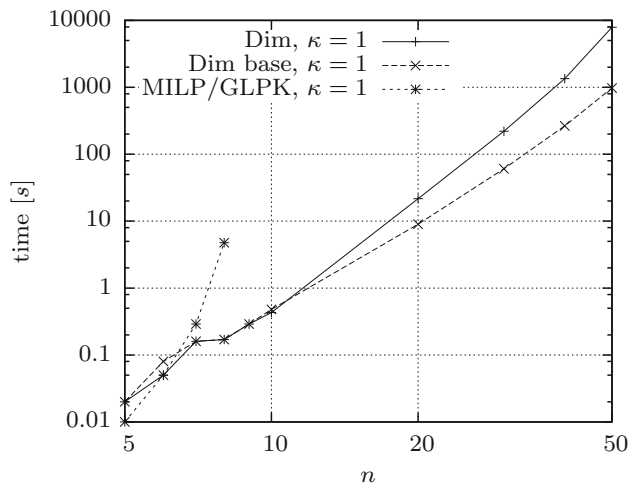


Fig. 3 Influence of network size on the computation time, Experiment A. Remember that $m \approx RT_m \cdot n$, $K = n \cdot (n - 1)$

investigate theoretically, we may suppose that the introduction of high penalties in Modification 1 introduces a large difference in the order of magnitude of the subproblem coefficients and this is the reason of slowing down the computations.

The solution time seems quite predictable, for example, in Fig. 3 with logarithmic scales of n and the time, an almost straight line is present. This is important for practical applications. This time is essentially lower than for Branch-and-Bound in GLPK solving the MILPs. In Experiment A, the time for Dim grows as about 4–5th power of n , but other problem sizes also grow with n : there holds $m \propto n$ and $K \propto n^2$. In Experiment 2, it grows with K as about $K^{1.5}$. An exemplary solution time is of the rank of 30s for 20 nodes, 80 links and 400 commodities.

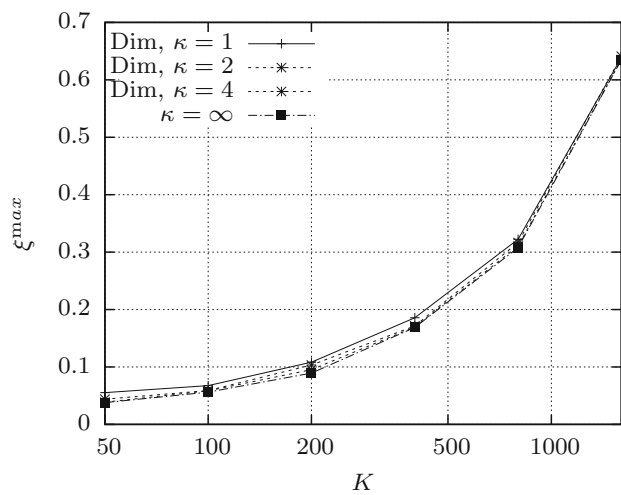


Fig. 4 Solution quality for different κ and number of commodities, Experiment B

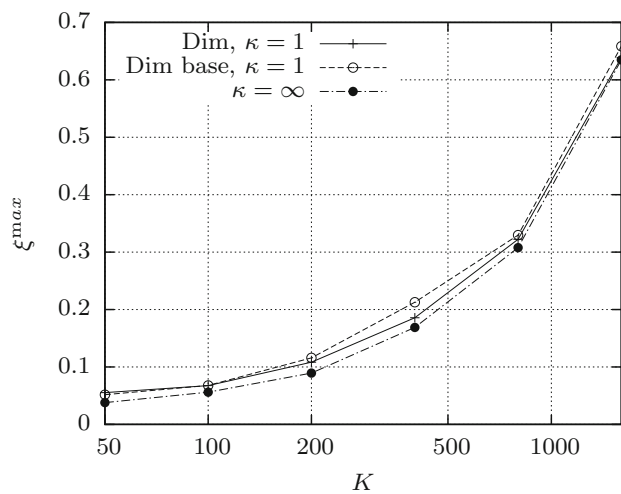


Fig. 5 Solution quality for different methods and number of commodities, Experiment B

6 Conclusions and further work

We have presented a practically viable method of calculating a minimum-congestion multicommodity network flow. It accommodates the flow κ -splittability and lower bounds on path flows. The quality of the solution obtained in the variation of randomized rounding is guaranteed to be worse from optimal 1-splittable solution by a factor logarithmic in the number of links. The practical inaccuracy of the solution seems even less and the solution time is quite low and predictable.

It would be practical to extend the method with reliability issues. For example, one might require that there are at least two paths generated for the commodity and one of the paths for the commodity be edge-disjoint with the others. There is at least one clear way to achieve this. However,

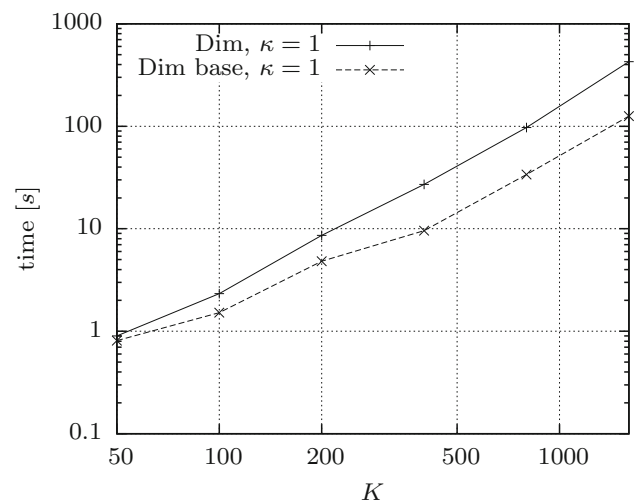


Fig. 6 Influence of the number of commodities on computation time, Experiment B

the approximation factor would grow at least by the factor of 2. This way leads through the following steps: (1) Add constraints: $\phi_e^k \leq D(k)/2$ for $e \in E, k \in K$ to problem (C) in order to enforce finding a flow that for a commodity k is decomposable to at least two edge-disjoint paths. (2) Modify the path reconstruction algorithm (Algorithm 2) so that at least two disjoint paths are obtained, e.g., by immediately pruning the links on one of the reconstructed paths from the graph. (3) Modify the rounding procedure so that at least two disjoint paths are rounded up. Lower bounds on flows redundancy paths should be defined by the user and carefully taken into account throughout the process described above. Node-disjointness of the paths could be obviously added to our problem by artificially splitting each node into an “input node” and “output node”, connected by one link.

Another extension is highly desired. When calculating likelihood Γ_i of rounding a “fractional” path flow up, we assume rounding up means rounding to $D(k)$. This is a rather conservative assessment, the flows are usually rounded to smaller values. Theoretically, this could deteriorate the quality of the solution given by our method for larger κ . Though such a phenomenon has not been observed in experiments, another assessment should be considered. Values σ_e^k in Modification 1 are calculated using the same conservative assessment, with similar consequences. In an improved assessment, the “round-up value” could be for example calculated to fall somewhere between $D(k)$ and the value of the fractional flow. Alternatively, an iterative procedure might be applied. The values from the solution could be used to obtain new likelihoods Γ_i and σ_e^k . Then, the solution could be recalculated.

Acknowledgements The research was sponsored by the National Institute of Telecommunications Grants No. 06.30.002.4, which was a partial continuation of the Future Internet Engineering (Inżynieria Internetu Przyszłości) FP7 project (2010–2013), and 06.30.001.6, and also by European Regional Development Fund under the Operational Programme “Innovative Economy” for the years 2007–2013, project POIG.02.03.01-00-104/13-00 “PLLAB 2020”. The author would like to thank the cooperators of the projects, especially, prof. Burakowski, Mariusz Gajewski, Dr. Janusz, Granat, Konrad Sienkiewicz, Wojciech Szymbak for discussing the multicommodity flow model and providing the application for the method.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix: List of the most important symbols

Problem

Main problem (P)

V	Set of nodes
E	Set of links
G	Problem graph, $G = (V, E)$
c_e	Capacity of link e
$s(k), t(k)$	Source, sink node for commodity k
P^k	Set of all nonlooping paths from $s(k)$ to $t(k)$
n	Number of nodes, $n = V $
m	Number of links, $m = E $
K	Number of commodities
$D(k)$	Demand for commodity k
κ	Flow splittability level
$\phi^{lo}(k)$	Minimum path capacity for commodity k
ξ^{max}	Goal function, network congestion
η_k	Number of paths for commodity k in the solution
p_i^k	i -th path for commodity k in the solution
ϕ_i^k	Capacity of the i -th path for commodity k in the solution

Continuous relaxation (C)

$E'(k)$	Set of links of capacity at least $\phi^{lo}(k)$
φ_e	Total flow in link e
φ_e^k	Flow for commodity k in link e
δ_{ij}	Kronecker delta for i and j

Algorithm and solution quality

Main algorithm

r	Number of rounds (user-given)
η'_k	Number of paths reconstructed from the solution of (C) for commodity k

p_i^k	i -th reconstructed path for commodity k
ϕ_i^k	Capacity of the i -th reconstructed path for commodity k in the solution

Randomized rounding

Γ_i	Likelihood of choosing i th reconstructed path to candidate for the final solution
W	Set of indexes of paths candidating to final solution
$\overline{\phi}_i^k$	Verified (cast on $\phi^{lo}(k)$) flow for i th reconstructed path for commodity k

Modification 1 and solution quality analysis

$\mathcal{K}_e^b, \mathcal{K}_e^\#$	Set of commodities with small (large, respectively) demands compared to the capacity of link e
σ_e^k	Penalty for flow of commodity k in link e —in Modification 1
α	Sufficient network oversizing coefficient, a value dependent on the Problem (P), defined by (17)

References

- Baier, G., Köhler, E., & Skutella, M. (2005). The k -splittable flow problem. *Algorithmica*, 42, 231–248.
- Bęben, A., Wiśniewski, P., Krawiec, P., Nowak, M., Pecka, P., Mongay Batalla, J., Białoń, P., Olender, P., Gutkowski, J., Belter, B., & Łopatowski, Ł. (2012). *Proceedins of the 13th ACIS international conference software engineering, networking and parallel & distributed computing (SNPD)*, Kyoto, Japan, pp. 643–648, August 2012. Content Aware Network Based on Virtual Infrastructure. SNPD 2012: 643–648. Content Aware Network Based on Virtual Infrastructure. *SNPD, 2012*, 643–648.
- Białoń, P. (September 2015). Metoda rozwiązywania k -podzielonego zadania przepływu wielotowarowego z ograniczeniami dolnymi na przepływ w ścieżce oparta o randomizowane zaokrąglanie. *Conference: Krajowe Sympozjum Telekomunikacji i Teleinformatyki*, 16–18 September 2015, Kraków, Poland. Przegląd Telekomunikacyjny i Wiadomości Telekomunikacyjne. SIGMA NOT, Warsaw, 8-9 (2015), 1030-1039 (in Polish).
- Burakowski, W., Tarasiuk, H., Bęben, A., & Danilewicz, G. (2012). Virtualized network infrastructure supporting co-existence of parallel internets. In *Proceedings of the 13th ACIS International conference software engineering, networking and parallel & distributed computing (SNPD)*, Kyoto, Japan, August 2012, pp. 679–684.
- Caramia, M., & Sgalambro, A. (2010). A fast heuristic algorithm for the maximum concurrent k -splittable flow problem. *Optimization Letters*, 4, 37–55.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to Algorithms*. McGraw-Hill: MIT Press.
- Fratta, L., Grela, M., & Kleinrock, L. (1973). The flow deviation method: An approach to store and forward computer communication network design. *Networks*, 3, 97–133.
- Gamst, M. (2013). A decomposition based on path sets for the multi-commodity k -splittable maximum flow problem. Department

- ment of Management Engineering: Technical University of Denmark, DTU Management Engineering Report No. 6.
9. Goldberg, A. V., Tardos, É., & Tarjan, R. E. (1990). Network flow algorithms. In B. Korte (Ed.), *Algorithms and combinatorics* (Vol. 9). Berlin: Springer.
 10. Gheini, L. D. (2006). *MPLS fundamentals*. Cisco Press.
 11. Granat, J., Sienkiewicz, K., & Szymak, W. (2015). Sieci sterowane programowo w Systemie IIP. Conference: Krajowe Sympozjum Telekomunikacji i Teleinformatyki, 16–18 September 2015, Kraków, Poland. *Przegląd Telekomunikacyjny i Wiadomości telekomunikacyjne. SIGMA NOT*, Warsaw, 8–9 (2015), pp. 1020–1029 (in Polish).
 12. Jiao, Ch., Gao, S., & Yang, W. (2015). Comparing algorithms for minimizing congestion and cost in the multi-commodity k-splittable flow. *Computer and Information Science*, 8(2), 1–8.
 13. Jiao, C., & Xia, Y (2014). The k-splittable flow model and heuristic algorithm for minimizing congestion in the MPLS networks. In *International conference on natural computation, ICNC 2014*, Xiamen, China, August 19–21. IEEE, 2014.
 14. Kleinberg, J. M. (1996). *Approximation algorithms for disjoint path problems*. PhD. thesis, Massachusetts Institute of Technology, USA.
 15. Kearns, M., & Saul, L. (2010). Large deviation methods for approximate probabilistic inference. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence (UAI1998)*, Madison, Wisconsin, USA, Morgam Kaufmanni, San Mateo, California, USA.
 16. Koch, R., Skutella, M., & Spenke, I. (2008). Maximum k-splittable s-t flows. *Theory of Computing Systems*, 43(1), 56–66.
 17. Makhorin, A. (2013). GNU linear programming kit. Reference manual for GLPK Version 4.52, Draft 2013, Free Software Foundation Inc., Boston, USA.
 18. McKeown, N., et al. (2008). OpenFlow: Enabling innovation in campus networks, *ACM SIGCOMM Computer Communication Review*, 38(2), 1 April 2008.
 19. Martens, M., & Skutella, M. (2006). Flows on a few paths: Algorithms and lower bounds. *Networks*, 48(2), 68–76.
 20. Medina, A., Laghina, A., Matta, I., & Byers, J. (2001). BRITE: Universal topology generation from a user's perspective, Boston University, Boston, User Manual, <http://www.cs.bu.edu/brite/publications/usermanual>.
 21. Raghavan, P., & Thompson, C. (1987). Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4), 365–374.
 22. Software-Defined Networking: The New Norm for Networks (2012). Open Networking Foundation White Paper, April 13, 2012.
 23. Synergistic Challenges in Data-Intensive Science and Exascale Computing. (2014). Summary report of the advanced scientific computing advisory committee (ASCAC). The U.S. Department of Energy, Department of Science, March 2014, <http://science.energy.gov>.
 24. Truffot, J., & Duhamel, C. (2008). A branch and price algorithm for the k-splittable maximum flow problem. *Discrete Optimization*, 5, 629–646.
 25. Wierzbicki, A. P., Makowski, M., & Wessels, J. (Eds). (2000). *Model-based decision support methodology with environmental applications*. Kluwer, Dodrecht.



Paweł M. Białoń received his Ph.D. degree in automatic control and robotics from the Warsaw University of Technology in 2013. Dr. Białoń has been with the Advanced Information Technologies Department of the National Institute of Telecommunications in Warsaw. His research interests include nonlinear and nondifferentiable optimization, decision support and scientific and commercial applications of these sciences in telecommunications, automatic computer network monitoring,

network management, techno-economical analyses of operator networks.