Paper

A Practical Approach to Traffic Engineering using an Unsplittable Multicommodity Flow Problem with QoS Constraints

Paweł Białoń

Department of Advanced Information Systems, National Institute of Telecommunications, Warsaw, Poland

Abstract—The paper presents a practical approach to calculating intra-domain paths within a domain of a content-aware network (CAN) that uses source routing. This approach was used in the prototype CAN constructed as a part of the Future Internet Engineering project outcome. The calculated paths must satisfy demands for capacity (capacity for a single connection and for aggregate connections using the given path are considered distinctly) and for a number of path-additive measures like delay, loss ratio. We state a suitable variant of QoS-aware unsplittable multicommodity flow problem and present the solving algorithm. The algorithm answers to the needs of its immediate application in the constructed system: a quick return within a short and fairly predictable time, simplicity and modifiability, good behavior in the absence of a feasible solution (returning approximately-feasible solutions, showing how to modify demands to retain feasibility). On the other hand, a certain level of overdimensioning of the network is explored, unlike in a typical optimization algorithm. The algorithm is a mixture of: (i) shortest path techniques, (ii) simplified reference-level multicriteria techniques and parametric analysis applied to aggregate the QoS criteria (iii) penalty and mutation techniques to handle the common constraints. Numerical experiments assessing various aspects of the algorithm behavior are given.

Keywords—multicriteria analysis, QoS-aware unsplittable multicommodity flow, traffic engineering.

1. Introduction

A practical approach to traffic engineering in a domain of the Content-Aware Network (CAN) with source routing is presented. The CAN network was a prototype built within the framework of the Future Internet Engineering project [1]. The project aimed at the construction of an architecture allowing a coexistence of various network techniques IP, circuit switching and post-IP, like CAN) on top of a common, virtualized equipment. A traffic engineering module computing content delivery paths satisfying Quality of Service (QoS) requirements within a CAN domain was necessary. Its construction was an interesting challenge, since the module had to be a part of an operational management system, thus it had specific demands, not usually satisfied by the existing relevant optimization algorithms. First, the module had to give any, perhaps by far non-optimal result within the time acceptable by the CAN administrator. On the other hand, a fair level of the network overdimensioning could be assumed, which is the usual case. Also, the constructed module and algorithm should have been easily expandable to encompass changes in the traffic engineering problem statement caused by a rapid development of the prototype.

Represent our network as directed graph (V,E) where $V \in \mathbb{N}$ is the set of nodes (identified with natural numbers) and $E \in \mathbb{N} \times \mathbb{N}$ is the set of arcs. Let $n = |V|$, $m = |E|$. The considered problem is then a variation of the unsplittable multicommodity flow problem with QoS constraints where commodities are defined by pairs: (relation, QoS class) with relation being a pair of different nodes: source and destination. We shall also discuss the possible extension of the problem with the constraints on maximum Protocol Data Units (PDUs) processed in a node within a second. Precisely, the problem is stripped a goal function, and is a feasibility problem rather than an optimization problem.

A commodity must be sent through a single path, due to the construction of the control plane. We avoid excess of the capacities of links. Also, the vector of $L \geq 1$ segment-additive measures of paths, like delay, error rate (when small), loss ratio, should not exceed the vector of demands connected with the given QoS class. The delay, loss ratio, or other additive measure for a given link may differ for different classes of services (CoS), which is determined by the queuing disciplines applied in the system.

1.1. Related Work

The problem of finding a path in a graph subject to multiple additive constraints, the multi-constrained path problem, is already NP-hard [2]. Thus, the same should be expected from our problem, containing that one. Actually, similar problems to ours cause a trouble to researches and their hardness (and their reluctance to distributed solving) ceases the proliferation of QoS technologies. Solving techniques for such problems traditionally use various polynomial-time approximations of them. Such approximations seem, however, aimed at obtaining a solution too precise for our needs at the expense of a too large solving time.

For example, in [3], a QoS-aware transportation planning problem (with a goal function representing the operator's

profit and with client demands elastic to the obtained QoS measures) is reduced to a fractional packing problem and treated using the Lagrangian relaxation technique of [4]. The authors obtain a $(1-\varepsilon)^{-2}(1+\varepsilon)^2$ – approximate solution within a $\mathcal{O}((\frac{1}{\varepsilon})^3(m+k)\log(m+k)mn^2\cdot(\frac{1}{\varepsilon})\log(m+k)+\log(nU))$ time, with U being the maximal ratio of link capacities. A more heuristic approach is presented in [5], for a delay-constrained routing problem, thus with $L=1$ but where a link delay can be a quite arbitrary function of the link traffic (we, in turn, assume this function constant). The main trick there was relaxing the ugly, non-convex delay constraints with the augmented lagrangian, the resulting subproblems were still non-convex but locally convex and could be approximately solved with local optimization techniques. This approach remains quite expensive in requiring optimization (mathematical programming) techniques while yielding only an approximate, local solution of a non-convex problem.

The complex problem of QoS multicommodity routing has been also treated with genetic algorithms. An interesting example is contained in [6], where the chromosomes encode some internal flows. The author tries to preserve the feasibility of chromosomes over the iterations. Genetic algorithms are not very sensitive to the choice of starting point but suffer from a too quick convergence that yields a solution far from the optimum. Therefore, the authors of [7] try to augment the genetic algorithm for a QoS multicast routing problem with a tabu-search technique, that has the opposite character: converges longer but depends of the starting point choice. In neither of the papers, however, very much can be precisely said about the solution quality and time, moreover, the experiments in [6] show an unsmooth, jumpy dependence of the solution time of problem sizes, while a predictable dependence is needed for our application.

An alternative to solving a QoS-constrained multicommodity problems is to allocate routing paths separately, i.e. solve a series of multi-constrained path (or: shortest path) problems that are, however, augmented to take care about leaving sufficient free link capacities for other paths. In the simple Widest-Shortest Path (WSP) approach [8], one picks the widest path between the shortest (simply in terms of hops number) paths from the origin to the destination in the subgraph built of links with capacities not less than the commodity demand. The symmetric Shortest-Widest Path approach is described in [9].

More sophisticated approaches involve more complex measures of bottlenecking potential of a path and rerouting, i.e. a recalculation of some paths if several paths coming through a link form a bottleneck – see [10]. Regarding routing a single QoS-constraints path (finding a multiconstraint path), which is an NP-hard problem, as said, and the solution methods are usually extensions of the Dijkstra shortest path method, which turn out to be variants of the Branch and Bound methods. They contain various accelerations, like fast closing trees (resulting in an approximate solution but of a controlled accuracy), or quite arbitrary reduction of the search space without a full control of the solution quality (TAMCRA, SAMCRA) – see [11], [12].

1.2. Proposed Approach

We propose a quick, rather rough yet effective heuristics, based on a consecutive allocating of paths for particular commodities (cf. Fig. 1). In this, we mimic a hypothetical manual traffic engineering. This is followed by a path rerouting phase, in case of the excess of the link capacities. In practice, however, this excess turns out to be marginal and the rerouting phase is short or even absent. This is important, because, in general, rerouting is a greedy operation, and it is difficult to design a rerouting-based algorithm of a provable low complexity. The path for the given commodity is generated as a shortest path in the appropriate source-sink relation with the link costs in the network graph defined as linear combinations of the link characteristics (delay, jitter, etc.) and also of some penalties for the current excess of the capacity of the link. The weights in the linear combination are, however, varied in a parametric experiment. Consequently, for a relation, many candidate paths are obtained.

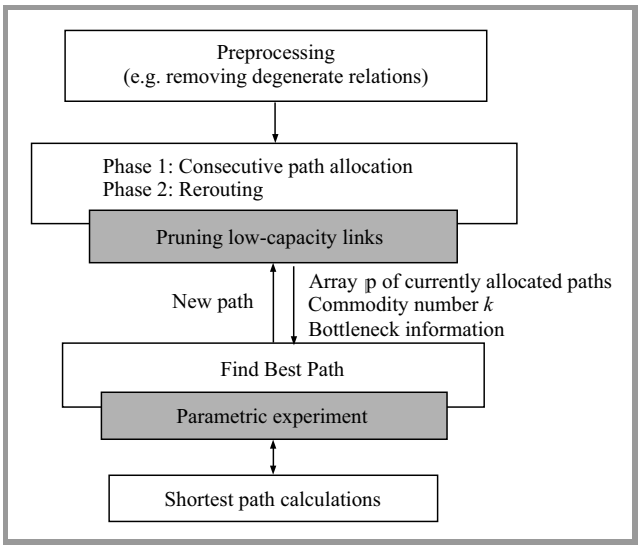

Fig. 1. Structure of the solving method.

The choice of the final path for the relation can be formed very elastically and may reflect various user preferences about the compromise between the particular QoS characteristics of the best path and also may take into account various potential heuristics to avoid link capacity excess in the further iterations of the algorithm. We have decided to use multicriteria technique (reference-level based [13]) to choose the best path. The overall simple and minimalist construction of the method allows its easy augmentation to follow even severe modifications to the solved problem. In developing new network techniques, the requirements from the traffic engineering module change very often. The price paid for the elasticity of our method is performing parametric experiments, which exhaustively search the space of weights. However, since the number of QoS characteristics

is usually low, the running time of the parametric experiments may be still moderate.

Our approach shares a relative computational simplicity with heuristics like SWP that allocate a path for the consecutive commodities and leaves room for other paths. Unlike these heuristics, it can be, however, considered a fully fledged method for solving a QoS-aware MCF problem, which looks at the whole set of commodities. Firstly, this is because of the existence of rerouting phase and secondly, because of the resolving of capacity conflicts of paths, which happens during the parametric experiments. In particular, the weights with which the bottlenecks contribute to the link weights when calculating the shortest path varies in the parametric experiment, unlike in [10], where it is fixed. Our approach is certainly a heuristics, but is computationally lighter than approaches like [4], [5] and has a highly predictable computation time. In addition, in Section 5 and in Appendix A we are able to show some partial possibilities of accessing the quality of obtained solution – in terms of the obtained additive path characteristics.

1.3. Mathematical Notation

We shall use the set membership operator $\in$ also to denote the presence of en element in a sequence. We shall have $\mathbb{R}_+ = [0,\infty)$, $\mathbb{R}_- = (-\infty,0]$, $\mathbb{N} = \{0,1,\ldots\}$, for $p \in \mathbb{R}^n$, $Y \subseteq \mathbb{R}^n$ $p+Y = \{p+y\colon y \in Y\}$, denote the convex hull of a set $A \in \mathbb{R}^n$ – by $\mathrm{conv}(A)$. A $C \in \mathbb{R}^n$ is a *cone* if $\forall c \in C$, $a > 0\, ac \in C$. We shall identify tuples of numbers (elements of $\mathbb{R}^n$) with column vectors.

2. Problem Origin

One of the Parallel Internets running simultaneously in the IIP system is the CAN network [14]. It uses its own transmission protocols in its interior, however, the users and the content storing servers connect to the network via TCP/IP access networks.

In the CAN, a user requests for a particular content (e.g. a video file), the network finds a server on which the content is stored and the content is transferred to the user via a constant (during the connection time) path between the access nodes controlling two access networks: that of the user and that of the content server. The transmission uses source routing, where the definition of the transmission path is stored in the frame header by the emitting node. By routing we mean the calculation of the paths.

The CAN network is divided into domains (Autonomous Systems, ASes). The connection path is set-up upon the connection request, by assembling intra-domain fragments of such paths (shortly: intra-domain paths), that are precalculated in each domain separately, for various possible external relations of the domain. The precalculation of intra-domain paths in the domain (called intra-domain routing) is the main subject of this paper and involves solving a mathematical programming problem.

Hence, in a domain (cf. Fig. 2), we need to compute paths for various CoS and various relations.

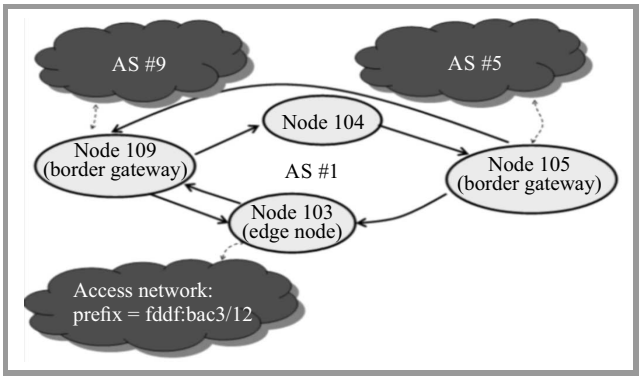

Fig. 2. An exemplary Parallel Internet CAN domain (AS) with $V = \{103,104,105,109\}$, $E = \{(103,109), (109,103), (104,105), (105,103), (109,104), (105,109)\}$, three external elements: two adjacent domains and one access network.

A relation is a pair of elements external to the domain (an external element is either an access network or an adjacent domain/AS, each external element has a unique controlling node in the domain – either a border gateway or an edge node, respectively). QoS demands for the intra-domain paths are given by the administrator for each relation.

3. Problem Statement

We shall sometimes identify network objects with their mathematical descriptions, e.g. identify nodes with their numbers. Our network is a directed graph $G = (V,E)$, where $V \subset \mathbb{N}$ is the nonempty set of nodes, $E \subseteq V \times V$ is the nonempty set of links. We use $n = |V|$, $m = |E|$.

We have the nonempty set $C \subseteq \mathbb{N}$ of classes of services and the number $L \geq 1$ of additive link/path characteristics like delay, loss ratio, etc.

Each link $e \in E$ has the associated capacity $\omega_e > 0$ and characteristics $\chi_{c,e,l} > 0$ for $c \in C$, $i \in \{1,\ldots,L\}$. These characteristics depend on the traffic but this dependence can be suppressed by taking values for some assumed maximum traffic. In traffic engineering we cannot control the actual future traffic intensities (we only calculate them using traffic demands estimates), thus cannot guarantee the values of the characteristics being functions of the traffic.

Note that the characteristics are indexed with a class of service, since different queue disciplines are set for different CoS, while the link capacities are common for all the CoS and it is the algorithm role to split the capacities between particular paths (thus between particular CoS).

We have the nonempty set $\{1,\ldots,K\}$ of commodities, actually describing relation-CoS pairs. For $k \in 1,\ldots,K$ let $\mathrm{start}(k)$ and $\mathrm{end}(k)$ be the be the source node and sink node for the commodity transfer, respectively, $c(k)$ be the class of service of the commodity. We assume $\mathrm{start}(k) \neq \mathrm{end}(k)$ holds[1]. There may be several commodities with the same

[1] Actually, relations between external elements controlled by the same node are possible. However, we can remove them from the considerations, since such relations yield degenerate, 0-link intra-domain paths.

start$(\cdot)$, end$(\cdot)$ and $c(\cdot)$, since they may refer to various relations with the same source and sink nodes (several external elements, i.e. adjacent domains or access networks can be controlled by the same node). Various relations can serve various subsets of classes of services, so the number of commodities may be lower than $|C|$ times the number of relations.
The following demands are defined by the administrator:

1. $\underline{\phi}_k > 0$ for $k = 1,\ldots,K$ – the QoS-required capacity of a path necessary for realizing a single connection for demand k;
2. ϕ_k (satisfying $\phi_k \geq \underline{\phi}_k$) for $k = 1,\ldots,K$ – the aggregated traffic (often referred as "traffic") from many simultaneous connections for commodity k. We use a static, deterministic model of traffic aggregation;
3. $\check{\chi}_{k,l} > 0$ for $k = 1,\ldots,K$, $l = 1,\ldots,L$ – maximum allowed values of the additive characteristics (delay, etc.) for the calculated intra-domain path for commodity k. The administrator sets these values bearing in mind the QoS requirements for the same characteristics for all the client-server paths, crossing several domains.

By a path p we shall mean a nonzero-element sequence $(p_1, p_2, \ldots p_q)$ of different numbers from V such that for all $i \in \{1,\ldots,q-1\}$ $(p_i, p_{i+1}) \in E$. For such a p for $e \in E$, we shall write $e \,\mathrm{on}\, p$ if $\exists k\ 1 \leq k < q$; $e = (p_j, pj+1)$, and $\mathrm{start}(p) = p_1$, $\mathrm{end}(p) = p_q$.
For commodity $k \in \{1,\ldots,K\}$, P^k will denote the set of possible paths for transferring the commodity k, i.e. such acyclic paths p that $\mathrm{start}(p) = \mathrm{start}(k)$, $\mathrm{end}(p) = \mathrm{end}(k)$.
The problem variables are $\mathbb{p}_i \in P^i$ for $i = 1,\ldots K$; $\mathbb{p}$ represents the intra-domain path for commodity i. We also use the vector notations: $\mathbb{p} = (\mathbb{p}_1,\ldots \mathbb{p}_K)$, $\phi = (\phi_1,\ldots,\phi_K)$. Variable $\mathbb{p}$ will be also the solving algorithm iterate. Let us define some functions:

1. $\phi_e(\mathbb{p},\phi) = \sum_{\substack{k=1,\ldots,K:\\ e \,\mathrm{on}\, \mathbb{p}^k}} \phi_k$
   for $e \in E$ – the total flow in link e;
2. $\chi_k(p) : P^k \mapsto \mathbb{R}^L$; $(\chi_k(p))_l = \sum_{e \in q} \chi_{c(k),e,l}$
   for $l = 1,\ldots L$ – vector of the additive characteristics of a potential path p for commodity k.

Our problem is defined as follows:

$$\text{Find } \mathbb{p} \in \{P^1,\ldots,P^K\} \tag{1}$$
satisfying
$$\underline{\phi}_k \leq \omega_e \text{ for } e \text{ on } \mathbb{p}_k,\ k = 1,\ldots,K \text{ (satisfaction of the QoS single connection capacity demand)} \tag{2}$$
$$\phi_e(\mathbb{p},\phi) \leq \omega_e \text{ for } e \in E \text{ (flow constraints of links)} \tag{3}$$
$$\big(\chi_k(\mathbb{p})\big)_l \leq \check{\chi}_{k,l} \text{ for } p \in P^k,\ k = 1,\ldots,K,\ l = 1,\ldots,L \text{ (additive QoS characteristics demands for paths).} \tag{4}$$

Additionally, we understand constraints (2) as hard (inviolable) since their violation would immediately violate some QoS demands of the assembled connection paths. Conversely, the remaining constraints are soft and could be possibly slightly violated if the algorithm cannot find a feasible solution. This setting is substantiated with the fact that these constrains anyway work with imprecise data: (3) – with the estimated traffic demands ϕ and (4) – with demands $\check{\chi}_{k,l}$, which are a kind of a quotas of some true QoS demands for a connection path assigned to the domain by an arbitrary decision of the administrator. They represent the maximum allowed contribution of the intra-domain path of our domain into the additive characteristics of the whole connection path.
Two modifications of the problem are considered:

Modification 1. We add the following constraints:

$$\mathbb{p}^i = \mathbb{p}^j \text{ for } (i,j) \in \{1,\ldots,K\} : \mathrm{start}(i) = \mathrm{start}(j) \wedge \mathrm{end}(i) = \mathrm{end}(j) \wedge c(i) = c(j). \tag{5}$$

In this way we require that paths for commodities with identical class of service, source and sink node should be transferred with the same path. This modification arose due to the limitations of the control plane in CAN, which distinguishes relations by pairs of source-sink nodes. Commodities coming from (or to) different domain external elements (access networks, neighboring domains) controlled by the same node and representing the same CoS cannot be properly distinguished.

Modification 2. We add the following constraints:

$$\varkappa_{c(k)} \sum_{\substack{K=1,\ldots,K:\\ e=(i,v),\\ e \,\mathrm{on}\, \mathbb{p}_k}} \phi_k + \varkappa_{c(k)} \sum_{\substack{k \in 1,\ldots,K:\\ \mathrm{start}(k) = v}} \phi_k \leq \xi_v \text{ for } v \in V, \tag{6}$$

where a parameter ξ_v denotes the Protocol Data Unit per second (PDU/s) throughput of node v, and a parameter $\varkappa_c$ denotes the size of a class c PDU, expressed in the used units of link capacity multiplied by one second. This modification expresses the limitations of the number of PDUs incoming in a second that the nodes can process.

4. Multicriteria Assessment Technique

Inside the algorithm, we shall assess some potential paths by several criteria that express the satisfaction of particular QoS demands, thus need a multicriteria assessment technique. Suppose we assess elements of set X with a vector quality function $Q : X \mapsto R^k$, where the higher $Q_i(\cdot)$, the better the satisfaction of the i-th criterion ($i = 1,\ldots, k$). Further assessments and comparisons of elements of X can be reduced to assessing and comparing the values of Q for them – points in the *space of attainable criteria* $Y = Q(X) \subseteq \mathbb{R}^k$.

Definition 1. For $y^1, y^2 \in \mathbb{R}^k$, y^1 *dominates* y^2 (in the Pareto sense), i.e. $(y^1 \succ y^2)$ if $\exists i \in \{1,\ldots,k\}\ y^1_i > y^2_i \wedge \forall j \in \{1,\ldots, k\} y^1_i > y^2)$.

Pareto dominance $\succ$ seems the only apparent comparison of points in Y but is only a partial order. A linear order can be obtained by scalarizing, i.e. comparing values of scalarizing function $S(q): Y \mapsto \mathbb{R}$ of point in Y. We will simplify the reference-level technique [13] and set $S = S^{\text{ref}} = S^{\text{ref}}_{\check{y},\bar{y}}$ where

$$S^{\text{ref}}_{\check{y},\bar{y}}(y) = \min_{i=1,\ldots m}(y_i - \check{y}_i)/(\bar{y}_i - \check{y}_i), \tag{7}$$

where we have the following *reference levels*: *reservation levels* $\check{y}_i$ for particular criteria (the values below that the relevant criteria should not deteriorate) and *aspiration levels* $\bar{y}_i > \check{y}_i$, here used for scaling purposes[2] ($i = 1,\ldots k$). S^{ref} is consistent with the Pareto dominance in the sense that if $a \succ b$ then $S(a) \geq S(b)$). This is also the property of *weighted summings*, more frequently used as scalarizing functions. Unlike with weighted summings, however, always all Pareto-nondominated (Pareto-optimal) elements of Y can be obtained as maximizers of $S^{\text{ref}}_{\check{y},\bar{y}}$ by a suitable choice of numbers $\check{y}_i$, $\bar{y}_i$ (assumed that the maximizers are unique for all choices of $\bar{y}_i$, $\check{y}_i$, which holds under the following regularity condition: $\forall q^1, q^2 \in Y \forall i \in \{1,\ldots,k\}\ q^1_i \neq q^2_i$, easily achievable for finite sets X by a suitable small random perturbation of Q). With $S^{\text{ref}}_{\check{y},\bar{y}}$ we have also a clear indication that all the criteria values y_i for some point in X (e.g. of a path) do not deteriorate below their reference levels ($\approx$ a path satisfies all the QoS constraints): $S^{\text{ref}}_{\check{y},\bar{y}}(y) \geq 0$.

Example 1. Let a path x be assessed by its bandwidth $\texttt{bandwidth}(x)$ and delay $\texttt{delay}(x)$. In our formalism, reasonably $y = (y_1, y_2) = \mathcal{Q}(x) = (\texttt{bandwidth(x)}, -\texttt{delay}(x))$, so both the outcomes y_1 and y_2 are maximized (the greater – the better). Let us have paths x_A, x_B, x_C with respective vectors of outcomes $y^A = (100, -3), y^B = (200, -2), y^C = (200, -1)$ – in some common units. Then

(i) $y^B \succ y^A$, since y^A is higher than y^B on all its coordinates,

(ii) y^A and y^C are not comparable in with Pareto order $\succ$: neither $y^A \succ y^B$ nor $y^B \succ y^A$.

Let us set reservation levels $\check{y} = (150, -4)$ and aspiration levels $\bar{y} = (400, -1)$. Then

(i) $S^{\text{ref}}_{\check{y},\bar{y}}(y^B) = \min((200-150)/(400-150), (-2--4)/(-1--4)) = 1/5$,

(ii) $S^{\text{ref}}_{\check{y},\bar{y}}(y^A) = \min((100-150)/(400-150), (-3--4)/(-1--4)) = -1/5$ and is negative, since the first coordinate of y^A is worse than its reservation level 150.

Note that since $y^B \succ y^A$, we have $S^{\text{ref}}_{\check{y},\bar{y}}(y^B) \geq S^{\text{ref}}_{\check{y},\bar{y}}(y^A)$.

[2]Particular criteria can have various typical values, be expressed in different units, thus should be made comparable. Note in (7), the i-th term in min (representing the satisfaction from fulfilling criterion i by y) becomes 0 when y_i is at its reservation level $\check{y}_i$ and 1 when y_i is at its aspiration level. Thus, in a sense, criterion i is normalized with $(\bar{y}_i - \check{y}_i)$ – cf. [13].

5. Solving Algorithm

5.1. Algorithm Statement

The main algorithm iterate will be $\mathbb{p} = (\mathbb{p}_1 1, \ldots, \mathbb{p}_K)$, where $p^i \in P^k$ for $i = 1,\ldots,K$; $\mathbb{p}$ is initially set to (NIL, NIL,…,NIL) and changed during the algorithm run.
Let us define some additional functions:

1. $\rho_e(\mathbb{p}, \phi)$ for $e \in E$ is the excess of the capacity of link e by the joint flows in the paths in $\mathbb{p}$ (in the few following definitions we shall assume path flows as given by ϕ) ; $\rho_e(\mathbb{p}) = \max\left(\left(\sum_{e \text{ on } \mathbb{p}^k, k=1,\ldots,K} \phi_k\right) - \omega_e, 0\right)$; we assume that no link belongs to a path being a NIL.

2. $\rho(\mathbb{p}) = \sum_{e \in E'} \rho_e(\mathbb{p})$ – the summary excess of the links capacity.

3. $\rho^-(k, \mathbb{p})$ where $k \in \{1,\ldots,K\}$, while we assume $p_k \neq \text{NIL}$ – is the measure of the contribution of the path $\mathbb{p}_k$ in the $\rho(\mathbb{p})$. We have $\rho^-(k,\mathbb{p}) = \rho(\mathbb{p}) - \rho((\mathbb{p}_1,\ldots,\mathbb{p}_{k-1},\text{NIL},\mathbb{p}_{k+1},\ldots,\mathbb{p}_K))$. Thus $\rho^-(k,\mathbb{p})$ is the decrease in $\rho(\mathbb{p})$ we would get by making $\mathbb{p}_k$ NIL by removing path $\mathbb{p}_k$ from the vector $\mathbb{p}$ of currently constructed paths.

4. $\rho^+(p,k,\mathbb{p})$ where $p \in P^k$, $k \in \{1,\ldots,K\}$, while we assume $p_k = \text{NIL}$ – is the measure of the potential of path p to increase the summary link capacity excess by storing p at the k-th position in $\mathbb{p}$. We have $\rho^+(p, k, \mathbb{p}) = \rho((\mathbb{p}_1,\ldots,\mathbb{p}_{k-1}p,\mathbb{p}_{k+1},\ldots,\mathbb{p}_K)) - \rho(\mathbb{p})$.

5. $\rho^+_e(e,k,\mathbb{p})$ where $e \in E$, $p \in P^k$, $k \in \{1,\ldots,K\}$, while we assume $p_k = \text{NIL}$ – is the measure of the contribution of path $\mathbb{p}_k$ to the excess of the capacity of link e. We have $\rho^+_e(p,k,\mathbb{p}) = \rho_e((\mathbb{p}_1,\ldots,\mathbb{p}_{k-1}p,\mathbb{p}_{k+1},\ldots,\mathbb{p}_K)) - \rho_e(\mathbb{p})$.

6. $\mathcal{Q}(p) = \mathcal{Q}_{k,E'}(p)$ (where $k \in \{1,\ldots K\}$ is the commodity number, $p \in P^k$, $E' \subseteq E$) is a scalar quality assessment function for path p for commodity k relative to subset E' of E; $\mathcal{Q}_{k,E'}(p) = -S^{\text{ref}}_{\check{y},\bar{y}}(-\chi_k(p))$, where $\check{y}_l = -\check{\chi}_{k,l}$, $\bar{y}_l = \check{y}_l + \sum_{e \in E'} \chi_{c(k),e,l}$ $(i = 1,\ldots L)$. The two minus signs serve to adjust the multicriteria apparatus from Section 4, which uses maximized criteria, to our minimized criteria (like path delay, path loss). Aspiration levels $\bar{y}_l$ are (quite arbitrarily) chosen to "scale" criterion i with the sum of the corresponding characteristics (delay, loss) over links in E' (cf. Section 4).

7. $\texttt{random(a,b)}$, where $a, b \in \mathbb{R}$, $a < b$ – returns a random value chosen under the uniform distribution on interval $[a,b]$,

8. $\texttt{randexpweightvector}()$ – returns $w/|w|$, where $w \in \mathbb{R}^{L+1}$, and each w_i is independently calculated as $\exp(10 \cdot \texttt{random}(0,1))$.

All the random choices in the algorithm are independent.

Algorithm 1 Parallel Internet CAN Traffic engineering algorithm

▷ Initialization:
$\mathbb{p}^k \leftarrow$ NIL for $k = 1,\ldots,K$.
▷ Phase 1
for all $k \in \{1,\ldots,K\}$ (in random order) **do**
  $E' \leftarrow \{e \in E:\ \omega_e \geq \underline{\phi}_k\}$ $V' \leftarrow \{v \in V:\ \exists w \in V:(v,w) \in E'\}$ ▷ Choose a subgraph of (V,E) with links of capacity not less than $\underline{\phi}_k$
  $\mathbb{p}_k \leftarrow$ FindBestPath(V',E',k)
end for
▷ Phase 2
$\texttt{it2} \leftarrow 0$
repeat
  if $\Big(\big(\forall k \in \{1,\ldots,K\}, l \in \{1,\ldots,L\}\ \chi(\mathbb{p}_k)_l - \breve{\chi}_{k,l} \geq 0\big) \wedge \rho(\mathbb{p}) = 0\Big) \vee \texttt{it2} \geq \Theta^{\mathrm{I}}(n,K)$ **then**
    return $\mathbb{p}$, $\breve{\phi}$
  end if
  $\texttt{it2} \leftarrow \texttt{it2}+1$
  $\texttt{rhobefore} \leftarrow \rho(\mathbb{p})$ ▷ Try a mutation
  $\widetilde{K}$=SelectToChange(ntochange)
  $p_k \leftarrow$ NIL for $k \in \widetilde{K}$
  for $k \in \widetilde{K}$ (in random order) **do**
    $E' \leftarrow \{e \in E:\ \omega_e \geq \underline{\phi}_k\}$ $V' \leftarrow \{v \in V:\ \exists w \in V:(v,w) \in E'\}$ ▷ Choose a subgraph of (V,E) with links of capacity not less than $\underline{\phi}_k$
    $\mathbb{p}_k \leftarrow \texttt{sparep}_k \leftarrow$ FindBestPath(V',E',k)
  end for
  if $\rho(\mathbb{p}) > \texttt{rhobefore}$ **then**
    $\mathbb{p}_k \leftarrow \texttt{sparep}_k$ for $k \in \widetilde{K}$ ▷ Withdraw mutation
  end if
until false
▷ Selects paths to be changed in the incoming mutation, randomly but with preferring those ps with high potential $\rho^-(\cdot,p)$ of decreasing $\rho(\mathbb{p})$ (heuristics)
function SelectToChange(numofpaths $\in \mathbb{N}$)
  $S \leftarrow \emptyset$
  $\texttt{totalpathrho} \leftarrow \sum_{i=1}^{K} \rho^-(i,\mathbb{p})$
  repeat
    Choose randomly i from $\{1,\ldots,K\} \setminus S$
    if **random**$(0,1) < \Theta^{III}/K + \rho^-(i,p)/\texttt{totalpathrho}$ **then** $S \leftarrow S \cup \{i\}$
    end if
  until $|S| = \texttt{numofpaths}$
  return S
end function

The algorithm, for the problem without Modifications 1 and 2, is depicted as Algorithm 1. It consists of two phases. In phase 1, the paths for the commodities are computed by function FindBestPath that searches a path best in terms of both the additive characteristics (delay, jitter, etc.) and ρ^+, the potential of increasing the current total link capacity excess. The term "best" related to several criteria is understood in terms of a complex formula, involving the reference level technique. Function FindBestPath will be described in Subsection 5.2. If phase 1 does not find a feasible solution and it does not exhaust the iteration limit, phase 2 of the local solution improvement is executed. It iteratively tries to decrease the total link capacity excess by removing a few paths from $\mathbb{p}$ and then recompute these paths in random order (the order of computing paths for a set of commodities is essential, since one computed path influences the current function ρ^+, and the current ρ^+ influences the next computed path, and so on).

Modification 1 is easily taken into account in the realized management module. Problem with this Modification is reduced to the problem without the Modification with artificial commodities, each being an aggregate of the commodities with a particular source node-sink node pair. An aggregation of demands for commodities is also necessary; it is done by summing for the traffic demands, by taking maximum for the single-connection capacity demands, and by taking minimum for the additive characteristics demands. Tackling Modification 2 is discussed later but not implemented.

Function Θ^I and constant Θ^{III} are the algorithm parameters, they heuristically determine the construction of a mutation or the number of iterations in particular algorithm loops. The suggested defaults for these parameters are $\Theta^I(K) = 3\log(K+3)$, $\Theta^{III} = 0.1$.

Remark 1: Whenever for commodity k the algorithm constructs an empty E' the algorithm stops with the message for the administrator that there is no capacity-feasible (in terms of a single connection) path from node start(k) to node end(k) for the given CoS c. This message shows the direction of reengineering of the network.

5.2. The FindBestPath Function

Function FindBestPath(V',E',k) with $V' \subseteq V$, $E' \subseteq E$, $k \in \{1,\ldots,K\}$ returns a path:

$$\operatorname{arg\,lexmax}_{\substack{p \in P^k:\\ \forall v \in p\ v \in V',\\ \forall e \text{ on } p\ e \in E'}} \Big(\min(0, \mathscr{Q}_{k,E'}(p)), -\rho^+(p,k,\mathbb{p}), \mathscr{Q}_{k,E'}(p)\Big). \tag{8}$$

Here $\operatorname{arg\,lexmax}_{x \in X}(a_1(x), a_2(x), \ldots a_t(x))$ is an $x \in X$ that yields the lexicographically lowest sequence $(a_1(x), a_2(x), \ldots, a_t(x))$, where the lexicographical order of sequences is defined by $(c_1,c_2,\ldots,c_t) > (d_1,d_2,\ldots,d_t)$ if and only if $\big((c_1 < d_1) \vee (c_1 = d_1 \wedge c_2 > d_2) \vee \ldots (c_1 = d_1 \wedge \ldots c_{t-1} = d_{t-1} \wedge c_t > d_t)\big)$.

Function FindBestPath, a path for commodity k in the subgraph (V',E') of (V,E) is returned that, if possible:

- first of all, is feasible in terms of the additive QoS characteristics (satisfies constraints 4),
- secondly, contributes low to the total capacity infeasibility,

- lastly, when the two above condition can be satisfied, yields values of the additive characteristics as low as possible.

We try to satisfy the demands for additive characteristics before satisfying the capacity demands for the given traffic for commodities. This is natural, since the additive characteristic demands can be set quite precisely (they can follow from the QoS demands divided by the expected number of domains on the connection paths) while the traffic on commodities can be only a result of rough prognoses.
This presents an "ideal algorithm" with the "ideal" function FindBestPath that is hard to compute. Later we will show the "real" variant of the Algorithm 1 where this ideal function is approximated.
Note that the FindBestPath function implicitly depends on the current algorithm state, namely, on the array p of currently constructed intra-domain paths. Such an implicit dependence of functions on global variables will happen in our depiction of the algorithm.

5.3. Comments on the Algorithm

The ideal algorithm is a heuristics, greedy in that optimizes paths for single commodities. Thus, it cannot have a strict convergence proof. Nonetheless, it at least exhibits properties suggesting a reasonable approaching of some solution. The algorithm yields in each iteration a solution feasible in terms of path additive constraints provided that one exists and, additionally $\rho(p)$, connected with the excess of link capacities, decreases monotonically between the main iterations. The satisfaction of the single connection capacity demands is basically enforced by the construction of E'.

Remark 2: The algorithm yields a solution satisfying constraints (4) on the additive characteristics whenever such a solution exists. This is because all the paths in the algorithm are constructed due to (8), where whenever for commodity k a path $p \in P^k$ with a nonnegative $\mathscr{Q}_{k,E'}(p)$ exists, the argument maximum will be a path with a nonnegative $\mathscr{Q}_{k,E'}(p)$.

Remark 3: By construction, the algorithm produces in its Phase 2 a sequence of $\mathbb{p}$ monotonic in $\rho(\mathbb{p})$ (since a mutation that increases ρ is withdrawn). Note that $\rho(\mathbb{p})$ is zero if and only if all the link capacity constraints (3) are satisfied by $\mathbb{p}$.

A simple extension to take into account node throughput constraints is possible.

Remark 4: Problem Modification 2 could be tackled by the algorithm by adding the term

$$\sum_{k\in\{1,\ldots,K\}} \sum_{e \text{ on } \mathbb{p}_k} \xi_v^{\text{exc}} \tag{9}$$

in the definition of $\rho_e(\mathbb{p})$; here

$$\xi_v^{\text{exc}} = \max\left(0, \Big(\sum_{\substack{K=1,\ldots,K:\\ e=(i,v),\, e \text{ on } \mathbb{p}_k}} \phi_k^{\blacktriangledown} + \sum_{\substack{k\in 1,\ldots,K:\\ \text{start}(k)=v}} \bar{\phi}_k \Big) \varkappa_{c(k)} - \xi_v \right)$$

is the throughput excess at node $v \in V$. This modification shall also update the descent definitions of ρ^+, ρ^- and ρ. Term (9) expresses the summary excess of a node throughput of both the end-nodes of the link. Also, with Modification 2, the commodities with degenerate relations (with the same source and sink node) are not negligible, since they load the nodes, but can be neglected with a simultaneous surrogate decrease of appropriate ξ_v.

5.4. Approximation of the FindBestPath Function

A strict realization of FindBestPath would be clearly a difficult numerical problem itself (it contains the well-known NP-complete multicriteria shortest path problem). In our real algorithm the function is approximated as depicted in Algorithm 2. Instead of searching the whole set of possible paths for a commodity P^k, we search the set of all possible shortest paths under a link cost being obtained by a different (but common for all the links) linear combinations of the additive link characteristics as well as of the link penalties ρ_e.

Algorithm 2 Approximation of function FindBestPath

function FindBestPath($V' \subseteq \mathbb{N}, E' \subseteq E, k \in \mathbb{N}$)
  $\bar{\omega} \leftarrow \sum_{e\in E'} \omega_e$
  $\bar{\chi}_l \leftarrow \sum_{e\in E'} \chi_{c(k),e,l}$ for $l = 1,\ldots,L$
  for $\texttt{itno} = 1,\ldots,\Theta^{II}(K)$ **do**
    $w \leftarrow \texttt{randexpweightvector}(1+\texttt{L})$
    $\texttt{Potentialpaths} \leftarrow \emptyset$
    for $e \in E'$ **do**
      $\mathfrak{c}[e] \leftarrow \sum_{l=1}^{L} w_l \bar{\chi}_{c(k),e,l} / \bar{\chi}_l + w_{L+1} \rho_e(\mathbb{p}) / \bar{\omega}$
▷ $\mathfrak{c}$ is an array of paths indexed with pairs of integers
      $\texttt{Potentialpaths} \leftarrow \texttt{Potentialpaths} \cup \{\texttt{Dijkstra}(V', E', c, \texttt{start}(k), \texttt{end}(k))\}$
    end for
  end for
  return $\arg\text{lexmax}_{p\in\texttt{Potentialpaths}} \Big(\min\big(0, \mathscr{Q}_{k,E'}(p)\big), -\rho^+(p,k,\mathbb{p}), \mathscr{Q}_{k,E'}(p) \Big)$
end function

Function $\texttt{Dijkstra}(V', E', c, \texttt{source}, \texttt{sink})$ returns a shortest path from $\texttt{source} \in V'$ to $\texttt{sink} \in V'$ in graph (V', E') (where $V' \in V$, $E' \in V' \times V'$) with link-weight mapping defined by array c (i.e., $c[e]$ is the weight of the link $e \in V'$) by calling the Dijkstra algorithm (see [15]). Function Θ^{II} is the algorithm parameter with the default value of the constant function of the value of 30.
For given commodity k and class of service c, the approximate variant of FindBestPath can access only a subset, call it $\underline{P}^k$, of the set P^k of paths available to the ideal variant. Namely, $\underline{P}^k$ is the set of paths available as shortest paths under some link weights that actually linearly combine the values of some characteristics of the link. The difference in the sets of available paths is essential. We can easily see it in terms of the supremum of the set of

values of $\mathscr{Q}_{k,E'}(p)$ with p running either of P^k and $\underline{P}^k$. The suprema may differ essentially, which might be easily shown to be equivalent to the well-known fact in the multicriteria analysis. In a set of variants we cannot in general find a Pareto-optimal variant by maximizing the weighted sum of the criteria values for the variants. In this equivalence, $\mathscr{Q}$ plays the role of the scalarizing function Q from Section 4. Fortunately, approximating Pareto-optimal variants with the mere weighting can be shown to introduce an error, measured in terms of Q, not greater than by the factor of $(\text{number of criteria}+1)$. This is the content of Theorem 1 in Appendix A. In translation to our example, it means that the maximal ratio of quality $\mathscr{Q}_{k,E'}$ of paths accessible to the real and ideal variants is not more than $(L+1)$. Of course, it is difficult to precisely calculate how this influence the quality of the solution yielded by our heuristics. However, the above observation gives some imagination about it. Consequently, we could use the number $(L+1)$ as an assessment of the rank of oversizing of the network (in terms of delay, jitter, etc.) necessary to compensate for the inaccuracy of the algorithm.
The practically important advantage of the real algorithm is that its time cost can be strictly assessed.

Remark 5: It is possible to have an implementation in which the references to the matrices elements cost $\mathscr{O}(1)$ and the `Dijkstra` implementation costs $\mathscr{O}(|E'|\log V')$ elementary operations (see [15]). We consider a call to `random` elementary. Even when there were always $V'=V$, $E'=E$, the cost of an iteration of phase 1 or phase 2 would be then dominated by the cost of $\Theta^{II}(K)$ calls to `Dijkstra` plus the cost $\mathscr{O}(\Theta^{II}(K)\cdot mK)$ of evaluating the arg lexmax in line 12 (we assume a reasonable implementation). Other sections of the algorithm would be clearly dominated in time by the above iterations.
Thus, the cost of the real algorithm is not greater than

$$\mathscr{O}\big(\Theta^{I}(n,K)\cdot\Theta^{II}(K)\cdot Km(K+\log n)\big). \tag{10}$$

Remark 6: It has been noticed in the literature that shortest paths (under some graph weights) in QoS-constrained Multicommodity Flow Problems already tend to be effective ducts for commodities also in the husbandry of available capacities. The algorithm constructs its solution as some shortest paths. Thus it may be often expected that the real algorithm finishes in phase 1 with $\mathbb{p}$ already feasible and phase 2 (in which link capacity violations are considered and decreased) makes 0 full iterations. In such a case, the run cost assessment reduces to

$$\mathscr{O}\big(\Theta^{II}(K)\cdot Km(K+\log n)\big). \tag{11}$$

Another formal issue must be noted.

Remark 7: Both the shortest path subproblems and argument maximum in (8), or in the corresponding condition in real can be ambiguous, and the algorithm chooses then any of the maximizing solutions. In scalarizing by function $\mathscr{Q}_{k,E'}$, it may lead to choosing a non-Pareto-optimal solution and in scalarizing by weighted summing, it may cause an impossibility to use Theorem 1. However, small random perturbation to the used link characteristics could rescind the ambiguities and it is the subject of further work.

6. Experiments

The goal of experiments was to verify the postulated properties of the algorithm: (i) the ability to quickly find a feasible solution under some existing overdimensioning of the network (also to examine the dependence of this time on the network size) and (ii) the ability to also quickly find a near-optimal solution when we shrink the resources a little.
The experiments have been run on topologies generated by the well recognized BRITE generator with a two-level hierarchy. Both the level of domains and the intra-domain level were generated due to the Waxman model, a probabilistic model of network growths. The problems have been created to reflect the the prototype CAN network in the IIP project, in particular, in its hierarchical structure and classes of service. Parameters, e.g. link capacities, have been given by hand reasonable values (when expressed in appropriate units), similar to that present in the project. Because of a high speed of modern network devices, delays were modeled as induced only by propagation, i.e. proportionally to the physical link length. Many parameters were set to the defaults taken by the authors of BRITE.
The domain number zero of the generated domains was always taken to construct the problem. The commodities were constructed as follows. The relations were established in all pairs of different domains adjacent to domain 0 (access networks were absent for the generation simplicity). Two classes of service, 1 and 2 (interpretable as 1 – best effort, 2 – premium) were served in each relation.
The algorithm with its defaults settings of Θ^{I}, Θ^{II} and Θ^{III} and with the extension for Modification 1 was implemented in C++ as a part of the CAN management module. The implementation of `Dijkstra` used C++ `Sets` to emulate heaps, and we may expect it to cost $\mathscr{O}(|E'|\log|V'|)$ elementary operations. The algorithm implementation included several small optimizations, e.g. avoiding unnecessary repetitions of invocations of some code fragments for unchanged data.

6.1. Key Experiment Parameters

The key experiment parameters were following:

- number of nodes n in the problem,
- the demand on the capacity of a single connection for all relations: for CoS 1 – `capacity1`, for CoS 2 – `capacity2`,
- the demand on the aggregated traffic for all relations – for CoS 1 – `traffic1`, for CoS 2 – `traffic2`,

- the demand on the path delay for all relations: for CoS 1: `delay1`, for CoS 2: `delay2`,
- the demand on the path loss ratios for all relations: for CoS 1: `loss1`, for CoS 2: `loss2`.

Several series of experiments were performed. Each series of experiments consisted in changing the value of one the above parameters while the remaining of them remained constant.
In each experiment, according to the values of these key parameters, a topology was generated by BRITE and a problem was constructed based on this topology. The random number generator of BRITE was always initialized with the same (default) values so two generations of topologies with the same parameters yielded identical topologies and problems).

6.2. Other Experiment Parameters

Some other BRITE parameters (`RT_N`, `AS_N`, `RT_m`, `RT_m` and `RT_HS` `BWIntraMin` – compare [16]) and problem parameters were following in a particular experiment:

1. Number of nodes in each domain `RT_N` $= n = 30$;
2. Number of domains `AS_N` $= 10$;
3. Approximate domain neighbors count `AS_m` $= 5$. In all the experiments, domain 0 turned out to have exactly 5 neighbors, this yields the number of relations equal to $5 \cdot (5-1) = 20$ and the number of commodities $K = 2 \cdot 20 = 40$;
4. The length of the square the nodes locations to be generated within `RT_HS` $= 300$;
5. The link capacities $\omega_e = 30$ for $e \in E$, thus `BWIntraMin` was set to 30;
6. The number of additive characteristics $L = 2$ (1 refers to delay, 2 to loss ratio)
7. The delay $\chi_{c,e,1}$ of link $e \in E$ for $c \in C$ was taken as $1/300$ of the distance of the endpoint nodes of link e; the distribution of this distance depends on `RT_HS`.
8. The loss ratio $\chi_{c,e,2}$ of link $e \in E$ for $c \in C$ was always set to 0.005.
9. Demands: $\underline{\phi}_k =$ `capacity1`, $\phi_k =$ `traffic1` $\check{\chi}_{k,1} =$ `delay1`, $\check{\chi}_{k,1} =$ `loss1` for $k \in i\{1,\ldots,K\}$, $c(k) = 1$; $\underline{\phi}_k =$ `capacity2`, $\phi_{k,2} =$ `traffic2` $\check{\chi}_{k,1} =$ `delay1`, $\check{\chi}_{k,2} =$ `loss2` for $k \in \{1,\ldots,K\}$ for $k \in \{1,\ldots,K\}$, $c(k) = 2$;

Other BRITE parameter settings were the the distribution defaults taken from the exemplary `TD_ASWaxman_RTWaxman.conf` file, other problem parameter settings were done according to the described construction of the problem from the topology.

6.3. Feasibility Limits

We shall establish the approximate "problem feasibility limit" values of some key parameters that describe demands – `traffic1`, `traffic2`, `delay1`, `delay2`, `loss1`, `loss2`, i.e. for a particular key parameter, we shall assess its best value when we obtain a feasibility problem, assumed the other demands are set loosely and essentially do not intervene. We shall do it in a heuristic reasoning and in performed experiments (described later) in which we gradually increase particular demands observing when the solver falls in troubles with obtaining a feasible solution. The established approximate limits are:

1. 1.25 for `delay1` or `delay2`,
2. $4 \cdot 10^{-5}$ for `loss1` or `loss2`,
3. 15 for `traffic2` (assumed `traffic1` is small).

The heuristic reasoning starts with delays. The node locations are randomly selected from the square of the side of 300 in BRITE (since `RT_HS`=300). Thus, roughly, we can expect that a maximum distance of the source-sink pair of nodes for some commodity is about 300 (remember that there are only 5 adjacent domains to our domain, thus at most 5 sink or source nodes. We cannot expect the extreme case that some two of them are situated at the both sides of a diagonal, i.e. at the distance of $300\sqrt{2}$). If some intra-domain path of this commodity were straight-line, the delay would be, according to the settings explained above, 1/300 of the length of this path, i.e. about 1. As any such a path is rather a segment line, we can expect the minimum possible delay on an intra-domain for this commodity be some more than 1, perhaps between 1 and 2.
Now consider traffic demands. Let us account only for the traffic demand for one CoS, say, CoS 2, and assume `traffic1` is negligibly small. The capacity demands will be always set lower in the experiments, than the respective traffic demands, thus can be neglected as well. In a typical case, each of the external domains is connected to our domain by a separate border gateway node in domain (because there are sufficiently many nodes in our domain). Thus there are 4 intra-domain paths starting at one border gateway node (they lead to the four remaining external domains). According to the setting `RT_m`=3, this node has 3 adjacent nodes in its domain, and it has to dispatch this incoming traffic firstly into three links, each of capacity of 30. This seems to be the bottleneck, the distribution of the traffic within the domain should be easier. The paths demands are equal and the paths cannot be split, so some link has to conduct two paths. Thus the traffic demand `traffic2` should not exceed a number about 15. We have described the bottleneck for ingress traffic. A symmetric bottleneck will clearly appear on egress traffic on some border gateway node but it yields a similar traffic limit.
The limit demands on loss ratios are more difficult to reason about, thus we left their derivation entirely to experiments. We only mention that these limits are connected with the

minimum hop number of a path in a relation (since all the links have the same loss ratio). Since the nodes are interconnected quite randomly in BRITE, the minimal hop length of a path should not highly grow with the growth of the number n of nodes. Hence, the feasibility limit of `loss1` or `loss2` should not depend essentially on n. We derive them for two different ns, for certainty.

6.4. Construction of Experiments

The default values for the key parameters taken in the series of experiments were following: $n = 30$, `capacity1` = 0.005, `capacity2` = 0.5, `traffic1`=1, `traffic2`=4 `delay1` = = 3, `delay2`=3, `loss1`=10^{-4}, `loss2`=10^{-4}. That were set loosely, i.e. far from the limits of problem feasibility.
The series of experiments consisted in changing selected key parameters from the defaults while the remaining key parameters were kept equal to their defaults. The following series were present:

1. Changing `delay2` – from 0.75 to 2 with step 0.25. One goal of this series was to observe the heaviness of the reaction of the solver computation time and the solution infeasibility on breaking feasibility limits while the other demands are set loosely. A second goal was to examine the values of these feasibility limits experimentally.

2. Changing `traffic2` – from 10 to 2 with step 2. The goals were analogous to that of series 1.

3. Changing `loss2` – from 10^{-4} to $6 \cdot 10^{-4}$ with step 10^{-4}, with analogous goals.

4. Changing `loss2` also from 10^{-4} to $6 \cdot 10^{-4}$ with step 10^{-4} but for $n = 100$ nodes (and the remaining key parameters set to the defaults, as before).

5. Changing the number n of nodes: experiments for $n = 10$, 30, 100, 300, 1000, to examine the influence of the number of nodes on the computation time under a large network overdimensioning. The feasibility limits should not depend much on n. The heuristic reasoning about them does not depend essentially on n. Some extra experiments, not presented here, also indicate this.

6. Changing n; experiments for n=10, 20, 100, 300, 1000 as well, however, with `delay1=delay2`=1.75, `loss1=loss2`=$6 \cdot 10^{-4}$, `traffic1`=1, `traffic2`=9. This series brings more demands closer to the feasibility limits, keeping them far from the limits respectively by the factor about 1.5 (or 1/1.5 for traffic demands). Note that the constraints induced by particular demands are not independent. Thus setting all of them to their feasibility limits would probably give an infeasible problem. Thus, with the settings of this series, we are even closer to the feasibility limits than "by the factor of 1.5" and we present a harder problem to the solver.

   The setting of `delay1` `delay2`, `loss1`, `loss2` are quite clear. Setting `traffic1`=3 and `traffic2`=7 gives the sum of `traffic1` and `traffic2` about 10 but the `traffic1` is not quite negligible (one must, however, remember that that now the traffic in a relation can be essentially split into two paths, realizing two CoS and thus the heuristic reasoning about the feasibility limits alters and we get slightly more distant from these limits).

7. Changing n; experiments for $n = 10$, 20, 100, 300, 1000 as well, however, with `delay1=delay2` = = 1.5, `loss1=loss2` = $4.5 \cdot 10^{-4}$, `traffic1` = 1, `traffic2` = 11.5. This series brings even tighter demands, distant from their limits by the factor 1.1–1.2.

6.5. Experiment Results

The experiments were done on a Dell PC with the Pentium 4 2.8 GHz CPU and with 1 GB RAM, under the Fedora Linux.
For a final solution $(\mathbb{p}^\star, \phi^\star)$, (where $p^\star \in \{P^1, \ldots, P^K\}$, $\phi^\star \in \mathbb{R}^n$ we define its traffic infeasibility – as the excess of the link capacity by the found intra-domain paths and its delay (loss) infeasibility – as the summary violation of delay (loss) over all the commodities: "traffic inf."=$\rho(\mathbb{p})$,

$$\text{"delay inf."} = \sum_{k=1}^{K} \max\left(\left(\sum_{e \text{ on } \mathbb{p}_k^\star} \chi_{c(k),e,1} \right) - \check{\chi}_{k,1}, 0 \right),$$

$$\text{"loss inf."} = \sum_{k=1}^{K} \max\left(\left(\sum_{e \text{ on } \mathbb{p}_k^\star} \chi_{c(k),e,2} \right) - \check{\chi}_{k,2}, 0 \right).$$

The results of experiments are shown in Tables 1 through 4. The dependence the computation time on the number n

Table 1
Influence of `delay2` (series 1)

`delay2`	Time [s]	Traffic inf.	Delay inf.	Loss inf.
0.75	3.67	0	3.48	0
1	3.62	0	1.46	0
1.25	1.23	0	0	0
1.5	1.21	0	0	0
1.75	1.25	0	0	0
2	1.21	0	0	0

Table 2
Influence of `traffic2` (series 2)

`traffic2`	Time [s]	Traffic inf.	Delay inf.	Loss inf.
10	1.22	0	0	0
12	1.26	0	0	0
14	1.23	0	0	0
16	3.72	4	0	0
18	3.64	18.2	0	0
20	3.62	30.2	0	0
22	3.62	42.2	0	0

Table 3
Influence of `loss2`

`loss2`	Time [s]	Traffic inf.	Delay inf.	Loss inf.
$n = 30$ (series 3)				
0.00001	3.63	0	0	0.00034
0.00002	3.66	0	0	0.00008
0.00003	3.65	0	0	0.00002
0.00004	1.2	0	0	0
0.00005	1.25	0	0	0
0.00006	1.22	0	0	0
$n = 100$ (series 4)				
0.00001	13.21	0	0	0.00038
0.00002	13.27	0	0	0.00014
0.00003	13.19	0	0	0.00004
0.00004	4.41	0	0	0
0.00005	4.43	0	0	0
0.00006	4.44	0	0	0

Table 4
Influence of n for a varying tightness of the demands

n	Time [s]	Traffic inf.	Delay inf.	Loss inf.
Series 5				
10	0.4	0	0	0
30	1.26	0	0	0
100	4.42	0	0	0
300	15.28	0	0	0
1000	60.6	0	0	0
Series 6*)				
10	0.4	0	0	0
30	1.22	0	0	0
100	4.42	0	0	0
300	15.38	0	0	0
1000	62.09	0	0	0
*) `delay1=delay2` = 1.75, `loss1=loss2` = $5 \cdot 10^{-4}$, `traffic1` = 1, `traffic2` = 9)				
Series 7**)				
10	0.4	0	0	0
30	1.42	0	0	0
100	4.42	0	0	0
300	45.78	8.5	0	0.000365
1000	180.63	6.5	0	0.000305
**) `delay1=delay2` = 1.5, `loss1=loss2` = $4.5 \cdot 10^{-4}$, `traffic1` = 2, `traffic2` = 11.5)				

of nodes for various demand settings is also illustrated in Fig. 3 (unless mentioned the key parameters have their default values).

The solver proved able to solve the problem even for a domain with a thousand of nodes within a time fully acceptable for an off-line management. Feasible solutions were returned in a similar time when the network overdimensioning was large and when many of the demands were situated close the problem feasibility limits, i.e. the network overdimensioning was small.

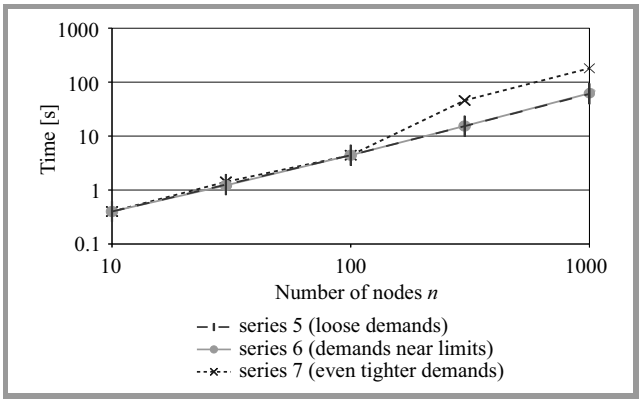

Fig. 3. Influence of n for a varying tightness of the demands.

The practical dependence of computation time on n was slightly more than linear. This is consistent with the theoretical value assumed the solution is already found in phase 1. Note (11) with the default Θ^I, Θ^{II}, with constant K and $m \sim n$ yields the solution time that depends on n like $O(n \log n)$. Another reason why the time grew slightly quicker than linearly was that the realization of some arrays (C++ Maps) raised a nonconstant array element access time.

When the demands were set so tightly that the solver could not obtain a feasible solution (so phase 2 was present), the computation time did not grow much, which could be expected by the construction of the stopping criterion for phase 2. The growth was about 3–4 times through the experiments and the time remained pretty acceptable in terms of its absolute values.

In the experiments where particular demands were gradually tightened, the solution infeasibilities responded with a gradual growth. Anyway, traffic, loss and delay infeasibilities (whose definitions use summing over commodities) were not big values compared with the applied shrinks in `delay2`, `loss2` or `traffic2` multiplied by the number of commodities (40). In series 7, the final infeasibilities that appeared for big n values were neither large in such a view. The experimentally obtained feasibility limits were 14–16 for traffic a demand (for one CoS, assumed the traffic demand for the second class is small), 1–1.25 for a delay demand, $3 \cdot 10^{-4} \ldots 4 \cdot 10^{-4}$ for a loss demand (the latest – for two different ns). This was consistent with the outcome of our heuristic reasoning about the limits and substantiated the previously described settings for the limits taken in the experiments.

7. Conclusions

The presented practical approach to the QoS-aware traffic engineering based on a small level of network overdimen-

sioning proved efficient in the experiments. Unlike sophisticated optimization algorithms, our heuristics most often exhibits a computation time a little-over-linear in the number of nodes and is able to tackle a thousand node network within a pretty-acceptable time. It behaves well when a feasible solution cannot be found.

Appendix A

Quality of the Weight-based Path Scalarization

Lemma 1. We have a nonempty finite set $Y \subset \mathbb{R}^k_-$. Let P be the set of nondominated (Pareto-optimal) points of Y: $P = \{y \in Y : \neg\exists z \in Y\, z \succ y\}$. Let W be the set of maximizers of weighted sum scalarizing functions: $W = \{w \in Y : \exists v \in \mathbb{R}^k_+, v \neq 0\, w \in \mathrm{Arg\,xmax}_{y\in Y} v^\top y\}$. Then $\forall p \in P\ \big(p \in W \vee \exists w^\star \in W\, w^\star \succ p\big)$.

Proof. By contradiction, we assume the negation of the claim : $\exists p \in P\ \big((p+\mathbb{R}^k_+) \cap \mathrm{conv}\,(W) = \emptyset\big)$. So there exists a hyperplane separating the convex sets $p+\mathbb{R}^k_+$ and $\mathrm{conv}\,(W)$, and since $p+\mathbb{R}^k_+$ is a (shifted) cone, the hyperplane can be chosen so as to contain the cone origin, p. I.e, $\exists s \in \mathbb{R}^k, s \neq 0, C \in \mathbb{R}\ \big((\forall x \in p+\mathbb{R}^k_+, s^\top x + C \geq 0) \wedge s^\top p + C = 0 \wedge \forall y \in \mathrm{conv}\,(W)\, s\top y + C < 0\big)$. Vector s cannot have a negative coordinate (if s_i were negative, then $p+\mathfrak{e}_i$, which is in $p+\mathbb{R}^k_+$, would give the negative value of our separating function, i.e. $s^\top(p+\mathfrak{e}_i)+C<0$ would hold; $\mathfrak{e}_i$ means the i-th versor). Thus the true sentence $\forall_{w\in W}\, s^\top w \leq s^\top p$ contradicts, by the definition of W, to $p \notin W$. ■

Theorem 1. We have a finite set $Y = \{y^1, \ldots, y^r\} \subset \mathbb{R}^k_-$ (with $r \geq 1$). Let $W = \{w^1, \ldots, w^m\}$ (with $r \geq 1$), $W = \{y \in Y : \exists v \in \mathbb{R}^k_+, \neq 0\, y \in \mathrm{Arg\,xmax}_{y\in Y} v^\top y\}$ be the set of weighted-sum maximizers of Y. Let $P = \{P^1, \ldots, P^n\}$ (with $n \geq 1$), $P = \{y \in Y : \neg\exists z \in Y\, z \succ y\}$ be the set of Pareto-optimal points of Y (note that both W and P must be nonempty by definition). Then for each $p \in P$ there exists $w \in W$ such that

$$\forall i = 1, \ldots, k\ |\omega_i| \leq (k+1)|p_i|. \tag{12}$$

Proof. Take any $p \in P$. If $p \in W$, the claim is obvious, so further assume $p \notin W$. By Lemma 1 there exists $\hat{w} \in \mathrm{conv}\,(W)$ such that $\hat{w} \succ p$. By Carathéodory's Theorem $\hat{w}$ is a convex combination of at most $k+1$ extremal points of $\mathrm{conv}\,(W)$. But each extremal point of $\mathrm{conv}\,(W)$ is in W, so $\hat{w}$ is a convex combination of at most $k+1$ points of W: $\hat{w} = \alpha_1 v^1 + \ldots + \alpha^{k'} v_{k'}$ with $1 \leq k' \leq k+1$, $\alpha_j \geq 0$, $\alpha \neq 0$, $\sum_j \alpha_j = 1$, $v^i \in W$. For some $j^\star$ there must be $\alpha_{j^\star} \geq 1/(k')$. Thus, bearing in mind that all v^j_i are nonpositive, for each $i \in \{1, \ldots, k\}$ $\hat{w}_i \leq (1/k') \cdot v^{j\star}_i$, thus also $\hat{w}_i \leq 1/(k+1) v^{j\star}_i$, $v^{j^\star}_i \geq (k+1) \cdot \hat{w}$ and, since $w \succ p$, $v^{j\star}_i \geq (k+1) p_i$. ■

Acknowledgments

The author would like to thank the cooperators from the Future Engineering Project, especially Prof. Wojciech Burakowski, Dr. Andrzej Bęben from Warsaw University of Technology, Dr. Janusz Granat and Dr. Paweł Olender form the National Institute of Telecommunications in Warsaw for the extensive discussions on the requirements and problem formulation.

This work was funded by the European Union, European Funds 2007–2013, under contract number POIG.01.01.02-00-045/09-00 "Future Internet Engineering" and by the National Institute of Telecommunications (Poland) Grants 06300013 through 06300016.

References

[1] W. Burakowski, H. Tarasiuk, A. Bęben, and G. Danilewicz, "Virtualized network infrastructure supporting co-existence of Parallel Internets", in *Proc. 13th ACIS Int. Conf. Softw. Engin., Netw. and Parallel & Distrib. Comput. SNPD 2012*, Kioto, Japan, 2012, pp. 679–684 (doi: 10.1109/SNPD.2012.67).

[2] A. Iwata, R. Izmailov, D.-S. Lee, B. Sengupta, G. Ramamurthy, and H. Suzuki, "ATM routing algorithms with multiple QOS requirements for multimedia internetworking", *IEICE Trans. Commun.*, vol. E79-B, no. 8, pp. 999–1006, 1996.

[3] G. Tsaggouris and C. Zaroliagis, "QoS-aware Multicommodity Flows and Transportation Planning", in *6th Workshop Algorithmic Methods and Models for Optimization of Railways (ATMOS'06),* R. Jacob and M. Müller-Hannemann, Eds. *OASIcs – OpenAccess Series in Informatics*, Technical Report ARRIVAL-TR-030. Schloss Dagstuhl–Leibniz-Center for Informatics, Dagstuhl Publishing, Germany, 2006.

[4] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems", in *Proc. 39th Ann. Symp. Foundat. of Comp. Sci. FOCS 1998*, Palo Alto, CA, USA, 1998, pp. 300–309.

[5] C. Duhamel and A. Mahul, "An augmented lagrangean approach for the QoS constrained routing problem", Research Report LIMOS/RR07-15, Université Blaise-Pascal, Aubiére, France, 2007 [Online]. Available: http://limos.isima.fr/IMG/pdf/rr-07-15.pdf

[6] M. Ghatee, "QoS-based cooperative algorithm for integral multicommodity flow problem", *Comp. Commun.*, vol. 34, no. 7, pp. 835–846, 2011.

[7] A. M. Allakany, T. M. Mahomud, K. Okurama, and M. R. Girgis, "Multiple constraints QoS multicast routing optimization algorithm based on Genetic Tabu Search algorithm", *ACSIJ Adv. in Comp. Science: an Int. J.*, vol. 4, issue 3, pp. 118–125, 2015.

[8] G. Apostopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda, "QoS routing mechanisms and OSPF extensions", RFC2676, 1999 [Online]. Available: https://tools.ietf.org/html/rfc2676

[9] J. Wang and J. Crowcroft, "QoS routing for supporting multimedia applications", *IEEE J. Selec. Areas in Communi.*, vol. 14, no. 7, pp. 1228–1234, 1996.

[10] Y. Wang and Z. Wang, "Explicit routing algorithm for internet traffic engineering", in *Proc. 8th Comp. Commun. and Netw.*, Boston, MA, USA, pp. 582–588, 1999.

[11] R. G. Garroppo, S. Giordano, and L. Tavanti, "A survey on multi-constrained optimal path computation: Exact and approximate algorithms", *Computer Networks*, vol. 54, no. 17, pp. 3081–3107, 2010.

[12] A. Kuipers, "Quality of Service routing in the Internet. Theory, complexity and algorithms", Ph.D. thesis, Delft University, Delft University Press, Delft, 2014.

[13] A. P. Wierzbicki, M. Makowski, and J. Weesels, Eds., *Model-Based Decision Support Methodology with Environmental Applications*, Dordrecht, Netherlands: Kluwer Academic Publishers, 2000.

[14] A. Bęben, Ed., "Specification of Parallel Internet Content Aware Network (version 2)", The IIP project, Warsaw, Poland, Report 31.01.2012, Task Z2.6.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. MIT Press and McGraw-Hill, 2001 (1990).

[16] A. Medina, A. Laghina, I. Matta, and J. Byers, "BRITE: Universal Topology Generation from a User's Perspective", Boston University, Boston, 2001, user manual" [Online]. Available: http://www.cs.bu.edu/brite/publications/usermanual.pdf

Paweł M. Białoń has been with the National Institute of Telecommunications. He received his Ph.D. in Automatic Control and Robotics from Warsaw University of Technology in 2013. His scientific interests include decision support and optimization with applications in telecommunications and data mining.

E-mail: P.Bialon@itl.waw.pl
Department of Advanced Information Systems
National Institute of Telecommunications
Szachowa st 1
04-894 Warsaw, Poland