Unicast QoS Routing Algorithms for SDN: A Comprehensive Survey and Performance Evaluation

Jochen W. Guck, Amaury Van Bemten, Martin Reisslein, *Fellow, IEEE*, and Wolfgang Kellerer, *Senior Member, IEEE*

***Abstract*—A variety of communication networks, such as industrial communication systems, have to provide strict delay guarantees to the carried flows. Fast and close to optimal quality of service (QoS) routing algorithms, e.g., delay-constrained least-cost (DCLC) routing algorithms, are required for routing flows in such networks with strict delay requirements. The emerging software-defined networking (SDN) paradigm centralizes the network control in SDN controllers that can centrally execute QoS routing algorithms. A wide range of QoS routing algorithms have been proposed in the literature and examined in individual studies. However, a comprehensive evaluation framework and quantitative comparison of QoS routing algorithms that can serve as a basis for selecting and further advancing QoS routing in SDN networks is missing in the literature. This makes it difficult to select the most appropriate QoS routing algorithm for a particular use case, e.g., for SDN controlled industrial communications. We close this gap in the literature by conducting a comprehensive up-to-date survey of centralized QoS routing algorithms. We introduce a novel four-dimensional (4D) evaluation framework for QoS routing algorithms, whereby the 4D correspond to the type of topology, two forms of scalability of a topology, and the tightness of the delay constraint. We implemented 26 selected DCLC algorithms and compared their runtime and cost inefficiency within the 4D evaluation framework. While the main conclusion of this evaluation is that the best algorithm depends on the specific sub-space of the 4D space that is targeted, we identify two algorithms, namely *Lagrange relaxation-based aggregated cost* (LARAC) and *search space reduction delay-cost-constrained routing* (SSR+DCCR), that perform very well in most of the 4D evaluation space.**

***Index Terms*—Delay-constrained least-cost (DCLC) routing, performance evaluation framework, quality of service (QoS), scalability, software-defined networking (SDN).**

Manuscript received March 7, 2017; revised July 24, 2017; accepted August 29, 2017. Date of publication September 7, 2017; date of current version February 26, 2018. This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 671648 (VirtuWind), in part by ERC Grant 647158 (FlexNets), in part by the Alexander von Humboldt Foundation through a Friedrich Wilhelm Bessel Research Award, and in part by the U.S. National Science Foundation under Grant #1716121. *(Corresponding author: Wolfgang Kellerer.)*

J. W. Guck, A. Van Bemten, and W. Kellerer are with the Lehrstuhl für Kommunikationsnetze, Technical University of Munich, 80290 Munich, Germany (e-mail: guck@tum.de; amaury.van-bemten@tum.de; wolfgang.kellerer@tum.de).

M. Reisslein is with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: reisslein@asu.edu).

Digital Object Identifier 10.1109/COMST.2017.2749760

I. Introduction

A. Topic Area: Routing Algorithms for QoS Networking

ROUTING, i.e., determining a route (path) from a source node to a destination node through a sequence of intermediate switching nodes, is an elementary function of the network layer in communication networks. Given the importance of routing for communication networks, a diverse array of routing algorithms have been designed. Many routing algorithms have been specifically designed for specific network settings or applications, see Section I-C.

Providing quality of service (QoS) is an important requirement for a wide range of communication network settings and applications. For instance, multimedia network applications require QoS from the network service, as do many network applications in industrial networks [1] and the smart grid [2] as well as networked control systems [3]. The required QoS is often in the form of delay bounds (constraints) for the data packets traversing the network. Accordingly, extensive research has developed routing algorithms that satisfy given delay constraints while minimizing some cost metric, i.e., so-called delay-constrained least-cost (DCLC) routing algorithms. DCLC routing algorithms and similar routing algorithms that support QoS networking are often referred to as *QoS routing algorithms*.

Generally, the route determination (computation) is either carried out in distributed nodes, e.g., the control modules in individual distributed Internet Protocol (IP) routers, or by a centralized controller, e.g., a Software-Defined Networking (SDN) controller [4]–[8]. Distributed routing algorithms had been intensely researched for traditional IP routing, e.g., [9]–[11], and more recently for ad hoc networks, see [12]–[16]. In the mid 1990s, the development of QoS paradigms for the Internet, see [17]–[22], led to a renewed interest in examining routing and spurred the development of a plethora of QoS routing algorithms, which mainly targeted distributed computation. In sharp contrast, the emergence of the Software-Defined Networking (SDN) paradigm [23], [24] has shifted the research focus to centralized network control, including centralized routing computations [25]–[30]. The purpose of the present survey is to provide a baseline for the use of existing QoS routing algorithms in centralized SDN based network control as well as for the further development of QoS routing algorithms that focus on centralized computation.

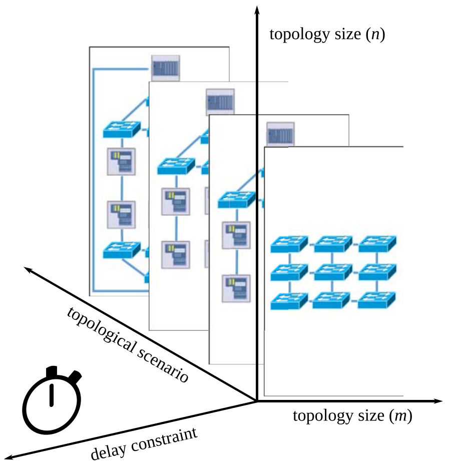


Fig. 1. Illustration of four dimensions of performance evaluation framework for delay-constrained least-cost (DCLC) routing algorithms: type of topology, scaling of the network (topology) in two dimensions, and delay constraint.

B. Contributions of This Survey

This article presents a comprehensive up-to-date survey of unicast QoS routing algorithms accompanied by a large-scale evaluation based on a consistent re-implementation of all the studied algorithms. We classify the unicast QoS routing algorithms according to the underlying routing strategy into several main categories, including priority queue based algorithms, Bellman-Ford based algorithms, Lagrange relaxation based algorithms, as well as algorithms that follow the least-cost and least-delay paths. In order to facilitate a comprehensive evaluation of QoS routing algorithms, we introduce a four dimensional (4D) evaluation framework, as illustrated in Fig. 1.

The first dimension corresponds to the type of topology. The second and third dimensions correspond to the scaling of a given type of topology into two dimensions that characterize the "size" of the network. The fourth dimension corresponds to the tightness of the delay constraint. The comprehensive evaluation of the existing QoS routing algorithms with this novel 4D evaluation framework provides valuable insights into the behaviors of the algorithms. Our evaluation has yielded a very large data set; we only present the most significant and insightful evaluation data in this article. We have made the entire evaluation data publicly available at [31], an interactive Web interface that allows for convenient navigation through the 4D evaluation space.

We have observed from our evaluations that it is not possible to elect an algorithm as "the best QoS routing algorithm". Indeed, we show that the performance of the algorithms strongly depends on the considered specific sub-space of the 4D evaluation space. Nevertheless, we identify two algorithms (out of a total of 26 compared algorithms) that achieve the best cost-runtime trade-off for most cases. Furthermore, we observe the general trend that algorithms based on a shortest path (SP) algorithm have shorter runtimes than algorithms based on a shortest path tree (SP tree) algorithm, which in turn have shorter runtimes than algorithms relying on a k shortest path (kSP) algorithm to reach a given optimality level.

C. Relationships to Prior Surveys

Given the key importance of routing for communication networks, routing algorithms have been extensively studied for a wide range of network settings and applications. Several prior survey articles have covered routing algorithms for several different special network settings and applications. Our focus in this survey is on routing algorithms that are suitable for supporting quality of service (QoS) networking in SDN networks. We proceed to contrast our present survey on QoS routing algorithms from prior related surveys on routing algorithms. Multicast routing algorithms for finding routes from a source node to multiple destination nodes have been surveyed in [32]–[34]; the closely related geocast routing to a prescribed geographic area has been considered in [35]. In contrast, we focus on unicast routing from a single source node to a single destination node. Multipath routing has been surveyed in [36]; we focus on routing algorithms for finding a single path in this survey. Routing in wireless networks has been covered by several prior surveys, e.g., [37]–[39]. Several prior surveys have focused on specialized forms of wireless networks. Specifically, routing in wireless mesh networks has been surveyed in [40]–[42], while routing for wireless sensor networks has been surveyed in [43]–[59]. Ad hoc network routing has been surveyed in [60]–[63], while routing for mobile ad hoc networks (MANETs) has been surveyed in [64]–[74] and vehicular ad hoc network (VANET) routing has been surveyed in [75]–[78]. Routing metrics for cognitive radio networks have been covered in [79], while routing in delay and disruption tolerant networks has been surveyed in [80]–[82]. Routing and route optimization for mobile nodes has been surveyed in [83] and [84]. Routing algorithms that consider the specific physical layer characteristics of optical (photonic) networks have been surveyed in [85]–[90]. Our survey focuses on routing for wired static networks without disruptions and does not specifically consider physical layer photonics. A few surveys have considered routing strategies for specific networking contexts, such as locator/identifier split Internet routing [91], traffic engineering and load balancing in the Internet [92], [93], content-based publish/subscribe systems [94], and green routing protocols with sleep scheduling [95]. In contrast, we consider general QoS routing algorithms.

General QoS routing algorithms have been covered in a few prior surveys that are less comprehensive than this survey. General overviews of the area of QoS routing and its research challenges has been provided in [96] and [97]. A handful of surveys have covered the QoS routing algorithms that have been developed up to around the years 2002-2003 [98]–[103]. Our survey is more up-to-date by covering QoS routing algorithms that have been developed up to the present time. A recent survey focused on multi-constrained QoS routing algorithms has been provided in [104]. Our survey is complementary to the survey [104] in that we broadly cover QoS routing algorithms satisfying a single or multiple constraints. Moreover, existing surveys have been limited to qualitative comparisons of the different QoS routing algorithms. In contrast, we introduce a novel 4D evaluation framework for comparing the quantitative performance levels of QoS routing algorithms and provide extensive quantitative performance comparison results. For completeness, we note that topology aggregation mechanisms for QoS routing have been surveyed in [105], while routing topology inference mechanisms have been surveyed in [106].

D. Survey Structure

Section II provides tutorial background on QoS routing algorithms, including brief reviews of elementary shortest-path algorithms that are utilized as underlying mechanisms in QoS routing algorithms. Section III provides a comprehensive up-to-date survey of the existing QoS routing algorithms. This survey section is organized into subsections dedicated to elementary algorithms, as well as algorithms that are based on a priority queue, on Bellman-Ford, on Lagrange relaxation, or on least-cost and/or least-delay paths in the network. Section IV introduces the four dimensional (4D) evaluation framework. Section V presents the results of the evaluation, while Section VI gives interesting anecdotal insights gained from our evaluations. Finally, Section VII summarizes the main conclusions.

II. BACKGROUND

This section provides tutorial background on QoS routing algorithms. First, Section II-A explains how QoS routing algorithms operate as a component within the broader context of a QoS networking framework. Then, Section II-B gives a brief tutorial on the basic definitions and the terminologies related to QoS routing algorithms and Section II-C outlines the goals that good QoS routing algorithms should strive for. Finally, Sections II-D and II-E give brief overviews of algorithms for computing one shortest path (SP) and multiple (k) shortest paths (kSP), which are elementary mechanisms for designing QoS routing algorithms.

A. QoS Routing as a Component of QoS Networking Framework

1) QoS Networking Framework Overview: Generally, a comprehensive network QoS management framework, e.g., [107]–[117], is required for providing QoS to network applications. A network QoS management framework consists of and coordinates among several components, including admission control [118]–[121], real-time scheduling [122], and QoS routing. That is, the QoS routing algorithm is one of several components required for achieving QoS in a communication network. A comprehensive survey of QoS networking frameworks is beyond the scope of this article. We only briefly note that, broadly speaking, there are two types of approaches for network QoS management, namely *global offline* QoS networking and *greedy online* QoS networking.

Global offline QoS networking jointly considers the complete set of network traffic flows and the various QoS networking components, such as routing and scheduling, to holistically determine the routes and schedules for all admissible flows. Global offline QoS networking typically involves complex optimization problems, such as integer or mixed integer linear programs [123]–[127], or assumes that routing paths are given, e.g., from a standard spanning tree protocol or unconstrained shortest path routing, to then optimize the scheduling, see for instance [128] and [129].

On the other hand, greedy online QoS networking considers a network with a given set of already ongoing network traffic flows and attempts to add (embed) a new flow to the network while maintaining the QoS requirements of the already ongoing flows as well as the new flow. We consider QoS routing algorithms for greedy online QoS networking in this survey. The surveyed routing algorithms require that the cost and QoS metric values (e.g., delay) are known for each edge (link) in the network graph prior to the execution of the routing algorithm. The usage of the surveyed routing algorithms in a practical network requires that the data plane forwarding behavior is abstracted to the granularity of costs and QoS metric values per edge of the network graph. Such abstraction can be achieved through measurement based systems, e.g., [107], [108], and [111], that probe the link behavior to estimate the edge costs and QoS metric values. The abstraction can also be achieved through model based systems, e.g., [130]–[133], that calculate the edge costs and QoS metric values based on a mathematical model of the underlying link scheduling and traffic shaping. While such mathematical models exist for some link scheduling and traffic shaping approaches, such as priority scheduling [134], [135], other emerging approaches, such as IEEE 802.1 Time Sensitive Networking [136]–[141] and IETF Deterministic Networking [142], require the development of new mathematical abstraction models. We consider one example greedy online QoS networking framework in the next section to explain the concrete functioning of QoS routing in the context of QoS networking in more detail.

2) Example Framework: Industrial QoS: In this section we briefly explain the QoS networking framework [133] as an example instance of a QoS networking framework. The framework [133] is designed to provide strict delay QoS, which is typically required for industrial communication systems [143]–[147], automotive networks [148], as well as some types of smart grid communication [149], [150] and Internet of Things communication [151]–[154]. Industrial communication systems carry critical messages, e.g., control signals for large automated production facilities, which have to be delivered with tight deterministic real-time quality of service (QoS) [109], [155]–[157]. A wide gamut of proprietary industrial communication technologies have emerged to provide this strict QoS [143]. Nevertheless, these proprietary technologies are typically costly and lack a uniformly accepted standardized communication framework. To overcome these challenges, the framework [133] has been based on Software-Defined Networking (SDN) [158]–[165]. SDN enables packet switches with a centralized control interface, e.g., OpenFlow [166], to provide deterministic real-time QoS guarantees. In order to satisfy the end-to-end deadline of each connection, the framework [133] encompasses four modules running on a centralized controller (see Fig. 2).

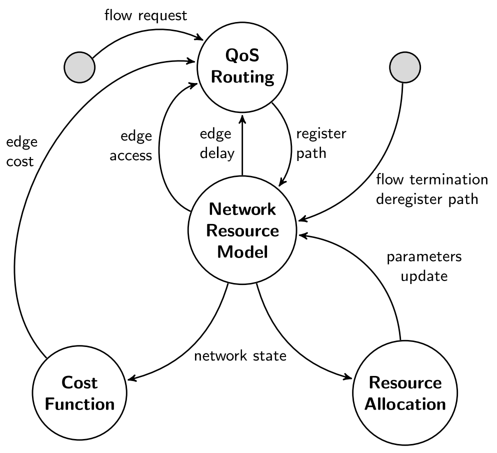


Fig. 2. Overview of the different function blocks of QoS networking framework [133]. The *network resource model* provides, through detailed queue modeling, deterministic delay bounds and implements access control that guarantees isolation of flows. The *cost function* indicates which edges should preferentially be used to maximize the probability of future flows to be accepted. The *resource allocation* adapts the allocation of resources to the different queues in order to maximize the probability of future flows to be accepted. *QoS routing* is then responsible for routing incoming requests on a path satisfying the end-to-end delay requirement of the request.

- The *cost function* transforms the network state, characterized for instance by data rates, buffer consumption, and already embedded flows, into a cost metric for each edge. The cost metric should maximize the number of flows that the network can serve.
- The *resource allocation* module adapts the allocation of resources (e.g., data rates) to the different queues in order to maximize the total number of flows that can be accepted in the network.
- The *network resource model* implements access control and worst-case delay computation based on the resources allocated to the different queues in the network. The network resource model tracks the consumption of resources by the embedded (already accepted) flows. To achieve a deterministic system behavior, the resource model is based on a mathematical traffic model, thus avoiding measurements of the actual network utilization. Since the scheduling delay experienced by a flow at a node depends on the queue at which it is buffered, worst-case delays are computed per queue. More specifically, the maximum (worst-case) delay per queue is bounded through deterministic network calculus modeling [134], [135]. The resource allocation pre-allocates worst-case delay budgets to the priority queues, which are scheduled according to the non-preemptive static priority policy. This ensures that the models of the different priority queues at a given link (hop) are independent [133]. Thus, admission decisions of low-priority queues do not have to be recalculated every time a flow is added to a high-priority queue.
- The *QoS routing*, e.g., delay-constrained least-cost (DCLC) routing, finds the least-cost path satisfying the end-to-end delay requirement of a connection request. The network resource model in [133] provides delays per queue, thus, routing has to be performed on a so-called *queue-link topology* illustrated in Fig. 3, where a given edge (link) of the physical topology is modeled by as many queue links as it has distinct QoS queues. In such a way, routing chooses both the links followed by a flow and the queues at which the flow will be buffered.

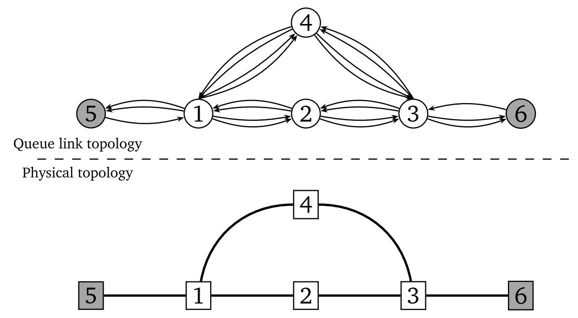


Fig. 3. Illustration of the queue-link topology concept [133]: A queue link models the outgoing queue (buffer) for an actual physical link. For instance, the bi-directional physical link between nodes 1 and 4 has two distinct QoS queues (e.g., with different delay bounds) in each direction; correspondingly there are two queue links from node 1 to node 4 and two queue links from node 4 to node 1.

TABLE I
CONCEPTUAL COMPARISON OF QoS ROUTING PROBLEM TYPES

Problem Type (Acronym)	Number of Optimized Metrics	Number of Constrained Metrics
Shortest Path (SP)	1	0
Constrained Shortest Path (CSP)	1	1
Multi-Constr. Shortest Path (MCSP)	1	M
Multi-Constrained Path (MCP)	0	M

B. Basic Definitions for QoS Routing Algorithms

Unicast QoS routing refers to the problem of routing a flow from a single source to a single destination so as to fulfill the QoS requirements of the flow. Depending on the QoS requirements, different problems can be defined. The most commonly encountered problems are defined as follows and contrasted in Table I.

- *Shortest Path* (SP): The route has to minimize a unique end-to-end QoS metric.
- *Constrained Shortest Path* (CSP): The route has to minimize an end-to-end QoS metric while keeping another metric below a prescribed bound.
- *Multi-Constrained Shortest Path* (MCSP): CSP problem with multiple end-to-end metrics that are constrained by individual bounds.
- *Multi-Constrained Path* (MCP): MCSP problem without optimization metric, i.e., the route only has to keep end-to-end QoS metrics below prescribed bounds.

These problems can be extended to k path versions that find k distinct paths. We refer to these extended problems as kSP, kCSP, kMCSP, and kMCP, respectively. It is also possible to define multi-objective problems that optimize more than one metric [104], [167]–[170]. These multi-objective problems are beyond the scope of this article.

We refer to the metrics that have to be optimized (or minimized) by the routing algorithm as the *costs*. On the other hand, we refer to the metrics that have to be kept below prescribed bounds as the *constraints*. The maximum end-to-end values below which these constraints have to be kept are then referred to as the *constraint bounds*. Note that the term *metric* can refer to either a cost or a constraint. Any metric is always associated to all the individual edges of the network. Depending on how an end-to-end QoS metric is computed from the metric values for individual links, three different categories of end-to-end QoS metrics can be defined: additive, multiplicative, and concave metrics. The end-to-end values of these three metric categories are, respectively, the sum, the product, and the minimum (or the maximum) of the metric values for the individual links. Delay, packet loss probability, and bandwidth are examples of additive, multiplicative and concave metrics, respectively.

Consider routing to be performed on a network graph $G = \{V, E\}$, whereby V is the set of vertices (network nodes) and E is the set of directed edges (with $|E|$ denoting the number of edges in the network). The vector of costs of the edges is denoted by c, $\mathsf{c} \in \mathbb{R}_+^{|E|}$. Let d, $\mathsf{d} \in \mathbb{R}_+^{M}$, denote a vector with M elements that represent the bounds for the constrained metrics. Let D, $\mathsf{D} \in \mathbb{R}_+^{M\times|E|}$, denote a matrix of the constraint values for the individual edges. Let P_{sd}, $P_{sd} \subseteq \{0, 1\}^{|E|}$, denote the set of paths from source node s to destination node d (whereby a value of 1 for an edge means that the edge belongs to the path). For additive metrics, the SP, CSP, and MCSP problems can be mathematically formulated as:

$$z_{\text{opt}} = \min_{\mathsf{x}\in P_{sd}} \mathsf{c}^{\mathsf{T}}\mathsf{x} \tag{1}$$

$$\text{s.t. } \mathsf{D}\mathsf{x} \leq \mathsf{d}. \tag{2}$$

The SP, CSP, and MCSP problems correspond to the cases $M = 0$, $M = 1$, and $M > 1$, respectively.

An *optimal* algorithm is an algorithm that always finds the optimal path with cost z_{opt}. A *heuristic* is an algorithm that finds a possibly sub-optimal path, i.e., a path with cost $z' \geq z_{\text{opt}}$. The *cost inefficiency* (CI) of an algorithm, measured in %, is defined as

$$\text{CI} = \frac{z' - z_{\text{opt}}}{z_{\text{opt}}} \times 100. \tag{3}$$

An optimal algorithm therefore always has a CI of 0%. An algorithm is said to be *complete* if it always finds a feasible solution if one exists. Completeness does not imply optimality.

QoS networking contexts (including the QoS networking framework [133], see Section II-A2) typically require the QoS routing algorithm to find a least-cost path satisfying an end-to-end delay constraint. This corresponds to a CSP problem with two additive metrics. This subset of CSP problems is also commonly referred to as *delay-constrained least-cost* (DCLC) routing problem. For this reason, we will often refer to the optimized QoS metric as *cost* and to the constrained metric as *delay*. The routing algorithm for QoS networking has to be complete. Indeed, if a connection request can actually be accommodated in the network, then the request should not be rejected.

In this article, we survey existing unicast CSP routing algorithms for additive metrics. Moreover, since MCSP algorithms can be used for solving CSP problems, we also present MCSP algorithms. While MCP algorithms can find feasible solutions for CSP problems, MCP algorithms do not optimize the cost metric and we therefore do not consider MCP algorithms in this survey.

C. Goals of QoS Routing

We proceed to summarize the key goals of a good QoS routing algorithm for centralized execution within the SDN paradigm:

- The algorithm should be complete. Indeed, we do not want to reject a connection request if it can actually be accepted.
- Generally, the DCLC problem is NP-complete [171]. Therefore, there is a fundamental trade-off between cost inefficiency and low runtime. Thus, a QoS routing algorithm should achieve a short runtime as well as a low cost inefficiency. Indeed, since routing is triggered upon receipt of a connection request and cost minimization leads to a network that can accept more flows, both short runtime and low cost inefficiency are important for good and fast request handling.
- The algorithm should be able to accommodate real values for the metrics and should not be based on value space reductions. Algorithms that only accommodate integer values for the cost metric may incur quantization errors and the evaluation of the impact of this quantization on the cost inefficiency is outside the scope of this study.
- The algorithm should not be restricted to the hop count as the only possible cost function. Furthermore, the hop count should not be considered in addition to the cost function for optimization. Indeed, we are primarily interested in low resource usage, which is completely represented by the cost function. Further, in dense networks there are typically many paths with the same hop count; hence the hop count is typically not a useful additional optimization criterion.
- The algorithm has exact up-to-date knowledge of the state of the network, which can be readily acquired with the SDN paradigm.
- The algorithm does not exploit any relationship between the cost and delay metrics. This assumption ensures that the algorithm can run with any arbitrary cost function.
- Cost and delay values may change during the runtime of the real system. Thus, results of computations for prior QoS routing runs, e.g., SP trees, cannot be stored and re-used for future QoS routing runs.
- The constraint must be guaranteed by the algorithm. We strive for strict requirements. Soft constraints that may be violated with a small probability are an interesting direction for future work.
- The connections are unicast connections. Multicast is outside the scope of this survey.

D. Overview of Shortest Path (SP) Algorithms

DCLC algorithms often make use of underlying SP and kSP algorithm mechanisms; therefore, we briefly review SP and kSP algorithms in this section and in Section II-E. Shortest path algorithms have been studied for a long time and the best algorithms are now well-known [172]. The ***Dijkstra algorithm*** [173] is a centralized algorithm that computes the SP from a single source node to all other nodes (i.e., an SP tree) in a graph with non-negative edge costs.

The Dijkstra algorithm is a priority queue based algorithm. That is, it maintains a queue containing a set of partial paths, i.e., paths starting from the source node and reaching an intermediate destination node which is not the ultimate destination. At each iteration, it takes the least-cost path among the paths in the queue and generates n new paths by extending this partial path with the n outgoing edges of the node at which the given path terminates. Among those paths, only paths with lower cost than the current least-cost path in the queue towards the same destination are added back to the queue. That is, the Dijkstra algorithm *relaxes* based on the cost values. In other words, the Dijkstra algorithm performs a breadth-first search and maintains the current best path found to each destination node. Nodes with least-cost distance from the source node are expanded first, thereby ensuring that any node has to be visited only once.

The ***Bellman-Ford algorithm*** **(BF)** [174]–[178] is a distributed algorithm that computes an SP tree in a graph, including graphs with negative edge costs. The algorithm maintains the current best path found to each node and runs $|V| - 1$ (where $|V|$ is the number of nodes in the network) iterations updating, for each node, the current best path to all neighbor nodes based on the current best path to the presently considered node. Since the path to any node is at most $|V| - 1$ hops long, all SPs will eventually be found. Note that, in the case of a centralized implementation, if an iteration yields no update, the algorithm can be immediately terminated, as subsequent iterations will not lead to any change. Also, as proposed by Yen [179] for centralized implementations, if the cost of the current best path to a node has not changed since the last iteration, then the outgoing edges of this node can be skipped since they will not lead to any new changes.

Both the Dijkstra and the Bellman-Ford algorithms can be used for finding the SP to a single destination. In such a case, the Dijkstra algorithm can be stopped as soon as the destination node is reached. In contrast, the Bellman-Ford algorithm cannot be stopped earlier than in the SP tree case (when SPs to all network nodes are found). Both algorithms can be adapted to compute the SP from any node to a single destination. These versions are called the *Reverse Dijkstra* and *Reverse Bellman-Ford* algorithms, respectively, and are simply obtained by considering incoming edges rather than outgoing edges when going from one node to the next node(s).

Hart *et al.* [180] proposed an improvement to the Dijkstra algorithm, the ***A* algorithm***, for finding a single-destination SP by introducing a so-called *guess function*. At each node, this guess function provides a guess for the cost of the SP from this node to the destination node. Paths out of the priority queue with least *projected cost* (i.e., sum of the current cost to the last node of the path and of the guess value at this node) are expanded first. To ensure the correctness and optimality of the A* algorithm, the guess values have to be lower than the real values. The closer the guess values are to the real values, the faster the A* algorithm will reach the destination. At one extreme, the A* algorithm with an exact guess function will directly traverse the SP to the destination. At the other extreme, the A* algorithm with a guess function of zero corresponds to the original Dijkstra algorithm. The overhead introduced by computing the guess function and the benefit of this guess function constitute the trade-off introduced by the A* algorithm. A straightforward guess function corresponds to the least-hop count multiplied by the cost of the least-cost edge in the graph. Such a guess function has to be recomputed upon any topology change. In our evaluations, we do not consider topology changes. Thus, the guess function can be computed offline, ensuring that the A* algorithm is, in any case, at least as fast as the Dijkstra algorithm.

The centralized Dijkstra algorithm performs generally better for finding an SP tree than other algorithms [182]. Similarly, the A* algorithm performs generally better than distributed algorithms for finding an SP. Therefore, we only consider the Dijkstra algorithm for finding an SP tree and the A* algorithm for finding an SP as underlying algorithms for the QoS routing algorithms in Table II. A detailed quantitative comparison that includes the Bellman-Ford algorithm, its centralized improvements [179] and [181], SP heuristics [183], and other A* guess functions [184] are left for future research.

E. Overview of k Shortest Paths (kSP) Algorithms

A very well known kSP algorithm, which is also one of the initial proposals for the kSP problem, is ***Yen's algorithm*** [185]. Yen's algorithm consists of two main parts. First, the SP is found using a traditional SP algorithm. Then, subsequent SPs are found based on the knowledge of this initial path. The $(k + 1)$th SP is found by starting at intermediate nodes of previously found paths, blocking the next edge in the path to force the algorithm to find another path, and running an SP algorithm from there. The LC path out of all these new paths is the $(k + 1)$th SP.

Yen's algorithm does not need to know the value of k when starting. We refer to this type of kSP algorithms as *iterative* kSP (ikSP) algorithms. In contrast, ***Chong's algorithm*** [186] requires k to be known in advance. The algorithm is then identical to the Dijkstra (or A* in our case) algorithm, but keeps, at each node, the current k best paths found. Once the destination(s) has (have) been visited k times, the algorithm can stop. We refer to kSP algorithms which have to know the value of k in advance as *static* kSP (skSP) algorithms. Note that any ikSP algorithm can also be used as a skSP algorithm.

We will see that for dense topologies with many edges (e.g., queue-link topologies with an edge for each outgoing QoS queue, see Section II-A2), algorithms using an underlying ikSP algorithm have poor performance. Indeed, while they could possibly perform well for sparse topologies, the high number of edges in dense topologies increases the number of paths that have to be traversed to reach the desired optimality. Consequently, we only consider Yen's algorithm as ikSP algorithm. The study of the possible performance increase introduced by the usage of other ikSP algorithms, e.g., [187]–[190], is left for future work. We are not aware of skSP algorithms other than Chong's, and will therefore only consider Chong's algorithm as an skSP algorithm.

III. Survey of Unicast (Multi-)Constrained Shortest Path (CSP and MCSP) Algorithms

This section provides a comprehensive up-to-date survey of unicast constrained shortest path (CSP) and multi-constrained shortest path (MCSP) algorithms which can be employed for QoS routing. We categorize these unicast QoS routing algorithms according to the underlying algorithm strategy into five main categories: 1) elementary algorithms, 2) algorithms based on a priority queue, 3) algorithms based on Bellman-Ford, 4) algorithms making use of the Lagrange relaxation optimization technique, as well as 5) algorithms making use of the knowledge of the least-cost (LC) and least-delay (LD) paths in the network. These five main categories of QoS routing algorithms are summarized in Table II.

A. Elementary Algorithms

Joksch [191] provided the initial *integer linear programming* (ILP) formulation of the CSP problem along with a proposal to solve it optimally using dynamic programming based on the delay constraint value. Unfortunately, this algorithm can therefore only be used with integer delay metric values and is hence not suitable for real-valued delays. On the other hand, note that dynamic programming approaches based on the hop count, e.g., the Bellman-Ford algorithm, which is by definition integer valued, can be readily used with real-valued metrics.

Aneja *et al.* [192] presented an optimal solution for the MCSP problem. Their algorithm performs pre-processing and is based on an implicit enumeration of all possible paths and is therefore computationally complex.

An elementary algorithm to find a feasible solution to the DCLC problem is to return the least delay (LD) path, which can be found with a single SP algorithm run. We will refer to this algorithm as the ***least-delay path*** **(LDP)** algorithm. The LDP algorithm does not consider the cost parameter; thus, the cost inefficiency may be high.

A way to take the cost into account while still keeping the algorithm simple has been proposed by Lee *et al.* [193] as the ***Fallback*** **(FB)** algorithm. The FB algorithm first computes the least-cost (LC) path using an SP algorithm and checks if it is feasible. If yes, then it can be returned. If not, then the LD path is computed and returned. The algorithm can be extended for solving the MCSP problem by running the SP algorithm successively with the different metrics (first with the cost and then with the different constraints) as cost until a feasible path is found. While the algorithm is complete for the CSP problem, it is not anymore for the MCSP problem. Indeed, for the CSP problem, running an SP algorithm with the constraint as cost will ensure finding a path satisfying the bound of this constraint (if one exists). On the other hand, for the MCSP problem, minimizing one of the constraints does not ensure that the other constraint bounds will be met.

Another simple idea utilizes LC paths as follows. Rather than switching to the LD path if the LC path is not feasible, search for the subsequent LC paths (using an ikSP algorithm) until a feasible path is found. Such an algorithm can also be applied for the MCSP problem and, since it discovers paths in order of increasing cost, is optimal in both cases. We will refer to this algorithm as the ***kSP Multiple Constraints*** **(kSPMC)** algorithm. Obviously, by continuing its search after finding the first feasible path, this algorithm is also able to solve the kCSP and kMCSP problems.

B. Algorithms Based on a Priority Queue

A widely considered algorithm for optimally solving the CSP problem is due to Widyono [194], who proposed the ***Constrained Bellman-Ford*** **(CBF)** algorithm. Despite its name, the algorithm is not similar to the original Bellman-Ford algorithm. CBF performs a breadth-first search. While keeping track of the LC path to each visited node, CBF discovers paths in increasing order of delay, stopping once the constraint is violated. As the algorithm is actually an extension of the Dijkstra algorithm, it is also sometimes referred to as the ***Constrained Dijkstra*** **(CD)** algorithm. Indeed, similar to the Dijkstra algorithm, the CD algorithm is based on a priority queue and relaxes based on the cost values. However, paths are retrieved from the priority queue in increasing value of delay, instead of cost. The discovery process can stop when the delay of the paths to further discover is higher than the deadline, since then no additional feasible paths can be found. Since the relaxation is done based on the cost, the LC path with delay lower than the deadline was found, i.e., the algorithm is optimal.

Liu and Ramakrishnan [195] proposed the ***A*Prune*** algorithm for solving the MCSP problem. As its name suggests, the A*Prune algorithm is in principle similar to the A* algorithm (see Section II-D). The A*Prune algorithm assumes that a guess function is available for each metric (i.e., for the cost and all the constraints), discovers paths (i.e., takes paths out of its priority queue) by increasing value of projected cost (see Section II-D), and prunes (i.e., removes from the set of paths to further extend) those paths for which a projected constraint value exceeds the corresponding end-to-end bound. Once the destination node is reached, the MCSP has been found. Note that, unlike the Dijkstra, A*, and CBF algorithms, the A*Prune algorithm does not keep a single path per node. In other words, its way of reducing the number of paths to further extend is not based on the destination node of these paths but on their projected constraint values. The A*Prune algorithm has the additional feature of being able to solve the kMCSP problem. Indeed, the extension of paths can be continued after the destination has been reached. Once the destination node is reached for the *k*th time, the optimal *k*th MCSP has been found. The A*Prune algorithm also solves the CSP and MCSP problems optimally.

C. Algorithms Based on Bellman-Ford

Jia and Varaiya [196] combined the search strategy of the Bellman-Ford algorithm with the delay-constrained unicast routing (DCUR) algorithm (which is based on LC and LD paths and will be reviewed in Section III-E) to define the ***delay-constrained Bellman-Ford*** **(DCBF)** algorithm. DCBF first computes a reverse LD tree. Then, it runs the Bellman-Ford algorithm, but updates the best path at a node only if it has a lower cost and if the sum of (*i*) the delay of the path built so far, (*ii*) the delay of the next edge, and (*iii*) the delay of the LD path from the terminal node of the next edge (i.e., the node reached via the next edge) to the destination is lower than the delay bound. We will refer to this test as the *projected delay test*. Jia and Varaiya also propose an extension to this algorithm, ***kDCBF***, keeping track of the k_d best paths for the reverse LD tree run and keeping the best k_c paths at each node for the forward Bellman-Ford run.

DCBF and kDCBF are complete, but not optimal. The reason for the sub-optimality of DCBF and kDCBF is as follows. DCBF may be sub-optimal if it "relaxes too much". DCBF relaxes a node when (*i*) the cost of the new path is lower, and (*ii*) the new path satisfies the projected delay test. Two cases are possible: Case 1: The cost of the new path is lower and its delay is also lower than the current path at the node. The relax operation is hence justified. Case 2: The cost of the new path is lower and its delay is higher, but still satisfies the projected delay test. Since the new path has a higher delay than the older path, it can happen that DCBF then has to take a higher cost path to satisfy the delay constraint until the destination. Therefore, although at the current node, the path was cheaper, because of its higher delay, DCBF then has to follow a higher cost path to reach the destination in time. This is the DCBF sub-optimality scenario, where DCBF is not optimal anymore because the final path would have been cheaper by keeping the original path (which was more expensive at one node).

Cheng and Ansari [197] proposed the ***dual extended Bellman-Ford*** **(DEB)** algorithm, an algorithm for the CSP problem similar to FB (see Section III-A). Instead of running an SP algorithm for the LC and LD searches, DEB runs a so-called *extended Bellman-Ford* (EB) algorithm which is able to find, for every possible hop count h, the optimal h-hop constrained path. For both runs, the path considered is then the best path among all those found. As FB, DEB is complete, but not optimal.

D. Algorithms Based on the Lagrange Relaxation

1) Background on Lagrange Relaxation: In mathematical optimization, the Lagrange relaxation technique allows to remove some constraints of the original problem and to introduce them in the optimization objective [171], [198], [199]. For example, the Lagrange relaxation of problem (1)–(2) is

$$L(\mathsf{u}) = \min_{\mathsf{x}\in P_{sd}} \quad \mathsf{c}^{\mathsf{T}}\mathsf{x} + \mathsf{u}^{\mathsf{T}}(\mathsf{D}\mathsf{x} - \mathsf{d}), \qquad (4)$$

where $\mathsf{u} \in \mathbb{R}_+^M$ is called the *Langrangian multiplier*. The minimized function is called the *Lagrange function* of path x and is also denoted as $L(\mathsf{u}, \mathsf{x})$. It can be shown that, if the original

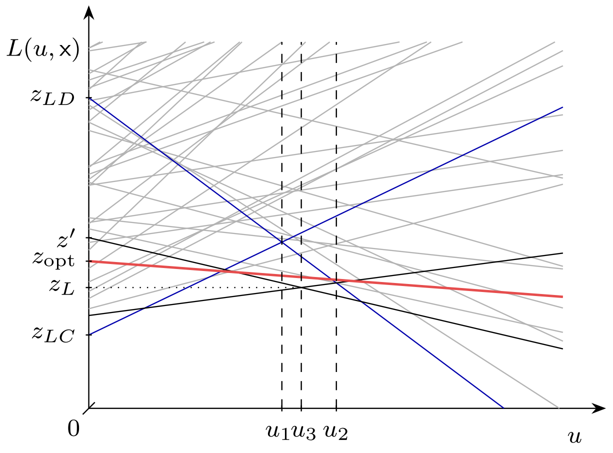


Fig. 4. Illustration of Lagrange functions $L(u, \mathsf{x})$ of paths in a network as a function of Lagrange multiplier u. The LARAC algorithm [200]–[203] finds the maximum of the lower boundary of this set of curves through a binary search, always keeping track of a best feasible (negative slope) path and a best infeasible (positive slope) path. The search starts with the LD and LC paths, found by simple SP searches, and continues with further SP searches with a modified cost function $c_u = c + ud$, where u is obtained as the intersection of the current best feasible and infeasible paths. From mathematical optimization theory, this is an approximation of the optimal solution of the original DCLC problem.

problem is feasible, then there is an optimal solution to

$$z_L = \max_{\mathsf{u}\in\mathbb{R}_+^M} L(\mathsf{u}), \qquad (5)$$

which is a feasible solution of the original problem. Problem (5) is referred to as the Lagrangian *dual* of the original problem (1)–(2), which is then referred to as the *primal* problem. Because solving the dual problem does not necessarily optimally solve the primal problem, we say that there is a *duality gap*.

2) Lagrange Relaxation Based Aggregate Cost (LARAC) Without and With Gap Closing (GC): Solving the dual problem requires to solve the relaxed problem (4) several times. The interesting aspect of this procedure is that, for the CSP problem, the relaxed problem corresponds to an SP problem with a modified cost function $c_u = c + ud$. This concept is illustrated in Fig. 4. Each line in Fig. 4 corresponds to the Lagrange function of a path in the network. Lines with null or negative slopes correspond to feasible paths while lines with positive slopes correspond to infeasible paths. The intercept of a line corresponds to the cost of the path. In our example, the optimal path (with cost z_{opt}) is highlighted in red. Since $L(\mathsf{u})$ is a piecewise-linear concave function [204], the u value maximizing $L(\mathsf{u})$ can be found using a binary search and always keeping track of a best feasible path and a best infeasible path, starting with the LC and LD paths (shown in blue in Fig. 4). As these two paths have slopes of different signs,[1] they intersect at a point u_1. This point is then used as the Lagrange multiplier for the next SP run. This run will find a new path. If the path is primal feasible (resp. infeasible), it replaces the current best feasible (resp. infeasible) path. The new pair of best feasible

[1]If this is not the case, either the problem is infeasible (both paths have positive slopes) or the LC path is optimal and can be returned (both paths have negative slopes).

and infeasible paths defines a new point u_2. The procedure then continues until the Lagrange multiplier does not change. The stored best feasible path at this point is then returned. In the example of Fig. 4, we can see that the found path (which corresponds to z' and u_3) is sub-optimal ($z' > z_{\text{opt}}$).

Aneja and Nair [200] initially proposed this algorithm as an optimal algorithm. They did not notice the duality gap. Later, Handler and Zang [201] proposed to close the gap as follows. At the end of the execution of the algorithm [200], the Lagrange value z_L of the found path is a lower bound on the optimal cost z_{opt}. Similarly, the cost z' of this path is an upper bound of the optimal cost z_{opt}. The gap can then be closed by running an ikSP algorithm with the last Lagrange multiplier, i.e., u_3 in our example. Figuratively speaking, the intersections are not relevant for the gap closing; rather, the gap closing traverses the vertical line at u_3 from bottom to top. For each path found by the ikSP algorithm, the upper and lower bounds on the optimal cost are updated with, respectively, the Lagrange value of the new path found and the cost of the best path found so far. When the lower bound gets greater than the upper bound, i.e., when the Lagrange value of the new path is greater than the cost of the best path found so far, it is ensured that no better path can be found. The best feasible path found so far is then the optimal solution. In the example of Fig. 4, the ikSP algorithm finds the optimal path as the third SP for u_3 and has to find the fifth SP (which is actually the LD path) to notice that no better path exists.

Handler and Zang [201] also introduced a parameter δ to stop the gap closing when the relative distance between the lower and upper bounds is less than δ. As this relative distance is an upper bound on the cost inefficiency (CI), the δ parameter allows to ensure that the cost inefficiency of the algorithm is always lower than δ. Blokh and Gutin [202] also proposed the algorithm without gap closing. Finally, Jüttner *et al.* [203] proposed again the same algorithm (without gap closing) and gave it a name: ***Lagrange Relaxation based Aggregate Cost*** **(LARAC)**. They also introduced a *maximal difference* (MD) parameter. The binary search is then stopped when the relative distance between the cost of the current best feasible path and the cost of the current best infeasible path is less than MD. We will refer to the LARAC algorithm with gap closing as ***LARACGC***. As elaborated, LARACGC with $\delta = 0$ is optimal and LARAC is not. Since they find at least the LD path, both are complete.

3) LARAC Variations and Extensions: Santos *et al.* [205] proposed an algorithm similar to LARACGC. The difference is that, after having computed the LC and LD paths, Santos *et al.* directly close the gap without performing the binary search. In particular, Santos *et al.* use a specific Lagrange multiplier computed based on the knowledge of the delay bound as well as the costs and delays of the LC and LD paths. From the name of its authors, we will refer to this algorithm as ***SCRC***. The algorithm has the same stopping condition as LARACGC and is therefore also optimal.

Jia and Varaiya [196] then proposed ***kLARAC***, an extension of LARAC that uses a kSP algorithm at each iteration, instead of an SP algorithm. The set of k paths found for a given u either contains only feasible paths, only infeasible paths, or a mix of both. As long as only feasible and infeasible sets are found, the new Lagrange multiplier is computed as for LARAC using the LC paths of the two sets. Once a mixed set is found, the LC feasible path of the set is returned. Jia and Varaiya show that the algorithm is always at least as good as LARAC in terms of cost inefficiency. Since k is a parameter of the algorithm, an skSP algorithm can be used. kLARAC is not optimal.

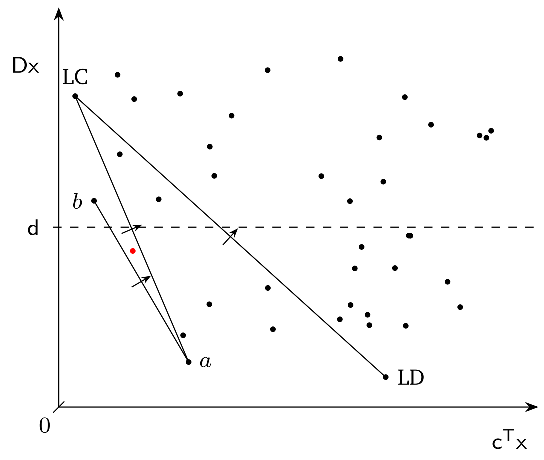
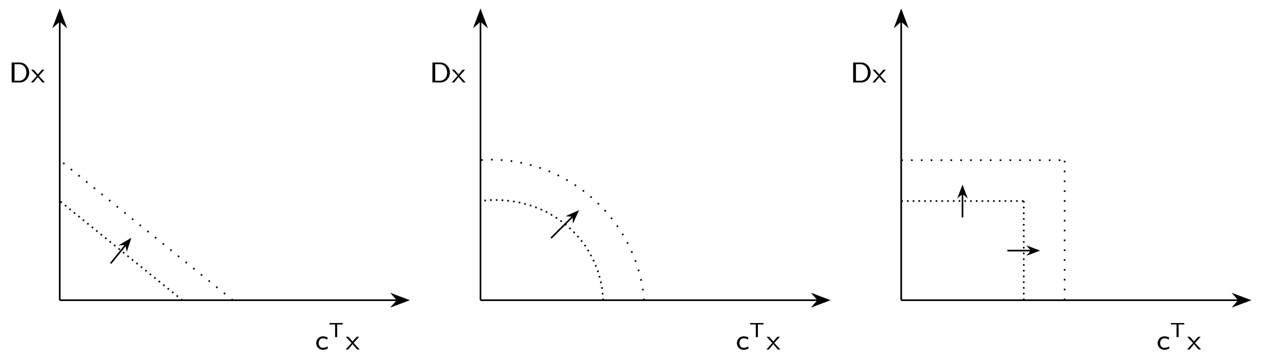


Fig. 5. Illustration of operation of LARAC algorithm in the delay (Dx)-cost ($\mathsf{c^T x}$) space: At each iteration, LARAC runs an SP search in the direction defined by the Lagrange multiplier. Here, the algorithm will first find the LC and LD paths. Then, based on the Lagrange multiplier computed with these two paths (which corresponds to the normal to the line connecting these two paths), a will be found. Similarly, b will then be found. Then, a or b will again be found, meaning that the Lagrange multiplier will not change. Hence, the algorithm will stop and the best feasible path, i.e., a, will be returned.

The LARAC algorithm can also be visualized in the delay-cost space, see Fig. 5, where a point corresponds to a given path in the network. At each iteration, the Lagrange multiplier defines the search direction of the SP run to be perpendicular to the line connecting the current best feasible and infeasible paths. This is shown by the small arrows perpendicular to the solid lines in Fig. 5. An SP run in a given direction finds the first point that the corresponding solid line would hit if pushed in this direction starting from point $(0, 0)$.

In the example of Fig. 5, LARAC will find the LD and LC paths, then a, then b, and the algorithm will then stop and return a. We see that the optimal path, shown in red, is missed. Korkmaz and Krunz [206] argued that this is due to the fact that the search direction is linear in the delay-cost space (Fig. 6a). They hence proposed an algorithm called ***Heuristic for Multi-Constrained Optimal Path*** **(H_MCOP)** which tries to search simultaneously in the delay and cost directions (Fig. 6c). To do so, the algorithm first finds the LD paths from any node to the destination using a reverse SP tree algorithm. Then, it runs an LC forward SP search but updates the best path at a node only when the new path is feasible or has a lower delay than the previously stored best path. The feasibility of the new path is checked using the LD paths stored from the reverse SP tree run. A new path is then considered feasible if it passes the projected delay test. The best path is hence sometimes updated based on the delay and sometimes based on the cost, which is how the algorithm



(a) Linear search direction ($\lambda = 1$). (b) Quadratic search direction ($\lambda = 2$). (c) Non-linear search direction ($\lambda \to \infty$).

Fig. 6. When running an SP (or kSP) algorithm with an aggregated cost, the type of aggregation of the initial metrics influences the search direction of the SP algorithm. In the case of two metrics, linearly combining the metrics leads to the linear search direction shown in Fig. 6a. In some cases, including DCLC routing, one may want to explore the delay-cost space simultaneously in both directions. To do so, the metrics can be combined in a non-linear fashion. For example, if the aggregated cost is computed by combining each metric to the power of two, a search direction similar to Fig. 6b is obtained. By increasing the power used to combine the metrics, one can reach the desired search direction shown in Fig. 6c. Unfortunately, once the metrics are not linearly aggregated, the optimal sub-structure property does not hold anymore; thus, classical SP and kSP algorithms (e.g., Dijkstra, A* [180], and Yen [179]) are not optimal anymore.

tries to simultaneously follow the search directions shown in Fig. 6c. The algorithm is nevertheless not optimal because this depends on how fast the delay and cost directions are respectively explored.

As its name suggests, the H_MCOP algorithm is also valid for the MCSP problem. While the explanation of the algorithm for the MCSP problem is more complicated (and is not included because we focus on the CSP problem in this article), the algorithm still tries to scan the multi-dimensional cost-constraints space simultaneously in all directions. However, to do so, an additional parameter λ has to be introduced and is defined as the power value used to combine the original constraints into an aggregated cost. $\lambda = 1$ corresponds to a linear search direction (Fig. 6a) and increasing λ towards infinity leads to the search direction shown in Fig. 6c. Besides, in the MCSP case, the algorithm is not complete anymore [206], [207]. Korkmaz and Krunz [206] then also proposed to use Chong's skSP algorithm for the forward run in order to continue searching in both directions until k paths have been found, thereby possible finding better paths. We will refer to this algorithm as ***kH_MCOP***. Clearly, this does not solve the incompleteness problem of the algorithm in the MCSP case [206] and the algorithm is still not optimal.

H_MCOP can be used to solve the MCP problem by observing the directions of all the constraints simultaneously and returning the first path found [206], [207]. It is then referred to as *H_MCP* and is incomplete [206]. Feng *et al.* [208] proposed ***NR_DCLC***, a CSP algorithm using H_MCP as underlying MCP algorithm, although NR_DCLC works with any other MCP algorithm. The LC and LD paths are first computed to check for infeasibility or for LC as elementary solution. Then, the cost of the LD path is set as first cost bound and the delay bound is the one of the original DCLC problem. Then, H_MCP finds an MCP path within these constraints. The cost of the path found is then used as new cost bound. This process is repeated until H_MCP does not find any path. To prevent H_MCP from returning the path found at the previous iteration, the cost bound for the next iteration is always set to a value *a little bit* smaller than the actual cost of the found path. Because NR_DCLC finds at least the LD path, it is complete. Since the underlying H_MCP algorithm is incomplete, it can be that NR_DCLC stops before having explored the entire cost-delay space and therefore NR_DCLC is not optimal.

Feng *et al.* [207] then proposed a variation of NR_DCLC. Instead of running H_MCP with the cost of the LD path as bound, H_MCP is run with the cost of the path found by H_MCOP as bound. As this algorithm improves on the solution found by H_MCOP, the authors refer to it as ***Modified_H_MCOP*** (**MH_MCOP**). The authors also introduce a parameter to limit the number of MCP iterations. We will refer to this parameter as H. For the same reasons as for NR_DCLC, MH_MCOP is complete, but not optimal. MH_MCOP can additionally solve the MCSP problem but, as it is based on H_MCOP, is not complete for it. On the other hand, NR_DCLC cannot solve the MCSP problem. Indeed, its initial LD search can only accommodate one constraint.

Feng *et al.* [207] additionally proposed an optimal algorithm, **E_MCOP**, similar to SCRC. E_MCOP first runs *E_MCP*, a complete MCP algorithm. E_MCP first runs an SP search for each constraint. If one of them cannot be met, it terminates. Otherwise, it runs an ikSP algorithm with the sum of the individual constraints, individually divided by the difference between their bound and their least value, as an aggregated cost. E_MCP returns the first feasible path found or stops once a path with an aggregated cost higher than the cost obtained by considering that each constraint reaches its bound is found. If E_MCP found no path, E_MCOP concludes that there is no solution. Otherwise, E_MCOP considers the found path as the current solution, uses its cost to define a cost border and runs an SP search for the cost to find its least value. From these two values, E_MCOP restarts an E_MCP search with the cost added to the aggregated cost (also divided by difference between its bound and its least value). The algorithm returns the current solution once the ikSP algorithm finds no path or when it finds a path with an aggregated cost higher than the cost obtained by considering that each metric reaches its bound. When a path is found by the ikSP algorithm, it replaces the current solution if it is feasible and has a lower cost. This algorithm is optimal for both the CSP and MCSP cases [207].

Guo and Matta [209] elaborated on the idea of using an underlying MCP algorithm in their ***delay-cost-constrained routing*** **(DCCR)** algorithm. DCCR behaves similarly to NR_DCLC, but runs the MCP algorithm only once. For this to be effective, they use an MCP algorithm that takes the costs of paths into account. This MCP algorithm can hence also be viewed as a DCLC algorithm which needs a cost bound. The algorithm runs an SP search where the cost of a path is defined as

$$\frac{\text{delay of the path}}{1 - \frac{\text{original cost of the path}}{\text{cost bound}}}, \tag{6}$$

which also tries to emulate the simultaneous scan of the delay-cost space in both directions. Nevertheless, with such a function, the cost of a path is not anymore the sum of the cost of its constituting edges and classical SP algorithms cannot solve the problem optimally. Therefore, Chong's algorithm is used to increase the probability of keeping track of good solutions. DCCR is complete, but not optimal. Instead of using the cost of the LD path as cost bound, Guo and Matta propose to use the cost of the path found by LARAC. This algorithm is then referred to as ***search space reduction DCCR*** **(SSR+DCCR)**. Since the algorithm improves the solution returned by LARAC, it is closing the duality gap, similarly to LARACGC, but only partially. In order to provide a cost bound, LARAC does not have to run until the end. Therefore, Guo and Matta define a parameter, which we will refer to as L, that limits the number of iterations (i.e., the number of SP runs, excluding the LC and LD searches) of the LARAC run. SSR+DCCR is also complete, but not optimal.

Agrawal *et al.* [210] proposed ***E-LARAC***, an extension of LARAC that additionally considers a constraint on the maximum number of hops by using a modified Bellman-Ford subroutine. Because the number of hops is not a constraint in many QoS networking scenarios, we do not consider E-LARAC in our evaluation.

The Lagrange relaxation has also been used by Ribeiro and Minoux [211] for solving the double-sided constrained SP problem, i.e., the problem of finding an SP whose delay (or any other metric) is lower than an upper bound but also greater than a lower bound. We do not consider this double-sided problem in our evaluation.

E. Algorithms Following the LC and LD Paths

Instead of computing a delay-constrained path from SP searches with modified costs, Salama *et al.* [212] and Reeves and Salama [213] proposed to solve the DCLC problem from the knowledge of the LC and LD trees towards the destination. The algorithm builds the path node by node. At each node, the algorithm chooses between the edge belonging to the LC path towards the destination and the edge belonging to the LD path towards the destination. The LC edge is chosen if it satisfies the projected delay test; otherwise, the LD edge is

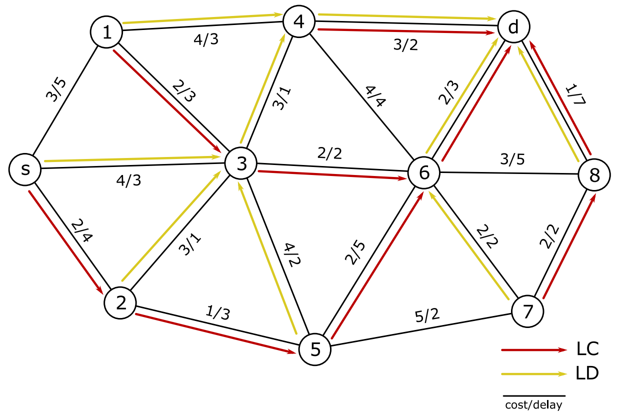


Fig. 7. Least-cost (LC) and least-delay (LD) paths from any node to a given destination in an example graph with links denoted by cost/delay. DCUR, DCR, and IAK are algorithms combining these paths in order to find a delay-constrained least-cost path (DCLC). In this example, with a deadline of 10, DCUR, which alternates between using the LC and LD edges, finds the path *s-2-3-6-d* with cost 9 and delay 10. DCR, which follows LD edges and switches once to the LC edges, finds the path *s-3-6-d* with cost 8 and delay 8. Finally, IAK, which follows LC edges and switches once to the LD edges, finds the path *s-2-3-4-d* with cost 11 and delay 8.

chosen. It can happen that a loop is created. In such a situation, the algorithm backtracks to a node that chose the LC edge and then chooses the LD edge instead. Salama *et al.* show that this backtracking ensures the removal of the loop. This algorithm is called ***delay-constrained unicast routing*** **(DCUR)**. Since DCUR can always find the LD path (by backtracking), it is complete. However, DCUR is not optimal.

Fig. 7 shows an example graph with the corresponding LC (red) and LD (yellow) trees towards the destination node *d*. Starting from the source node *s*, DCUR chooses, at each node, between the red and yellow outgoing edges of the node depending on the result of the projected delay test. In this example, with the delay constraint set to 10, DCUR would choose LC-LD-LC-LC (no loop occurs), thereby finding the path *s-2-3-6-d* with cost 9 and delay 10. Indeed, when at node *2*, DCUR cannot follow the LC edge. If it did, it would reach node *5* with a delay of 7. Since the LD path from node *5* to the destination has a delay of 5, it would not be possible anymore to reach the destination with a delay lower or equal to 10. Note that this is sub-optimal as the optimal path (that, e.g., CBF would find) in this example is *s-3-6-d* with cost 8 and delay 8.

Sun and Langendörfer [214] then proposed a solution, called ***distributed delay constrained routing*** **(DCR)**, to avoid the creation of loops and hence to prevent the algorithm from having to backtrack, thereby reducing runtime. DCR follows the LD path until the sum of (*i*) the delay of the path so far and (*ii*) the delay of the LC path from the current node to the destination is lower than or equal to the delay bound. Starting from this point, the algorithm then follows the LC path until the end, since it is ensured that it will satisfy the delay constraint. Since the LD path is only followed from the source node, it can be computed by a simple SP run. DCR is also complete, but not optimal.

In the example of Fig. 7, still with the delay constraint of 10, DCR follows the LD edge until node *3* as the sum of the delay of the path so far (no path and hence delay of 0) and the delay of the LC path from the current node (*s*) to the destination (path *s-2-5-6-d* with delay 15) is greater than the delay bound ($0 + 15 > 10$). Indeed, following the LC edges already from node *s* would lead to an infeasible path. Then, starting from node *3*, DCR switches to the LC path as the sum of the delay of the path so far (path *s-3* with delay 3) and the delay of the LC path from the current node (*3*) to the destination (path *3-6-d* with delay 5) is lower than the delay bound ($3 + 5 < 10$). Indeed, DCR is now sure that following the LC edges until the destination will lead to a feasible solution. Hence, DCR finds the path *s-3-6-d*, which is actually the optimal path. This example shows that, while DCR is simpler than DCUR, it can still provide, in some circumstances, a path closer to optimality.

Ishida *et al.* [215] then proposed the opposite strategy, i.e., to first follow the LC path and to switch to the LD path as soon as following the LC path would lead to a node from which the delay constraint cannot be satisfied anymore. For the same reason as for DCR, the LC path can be computed by a simple SP run. Based on Ishida *et al.* [215], we refer to this algorithm as ***IAK***. IAK is also complete, but not optimal.

In the example of Fig. 7, still with the delay constraint of 10, IAK follows the LC edge until node *2* as this edge has a delay of 4 and the LD path from node *2* to the destination has a delay of 4, thereby ensuring that the delay constraint can still be met. From node *2*, for the same reason as for DCUR, IAK cannot follow the LC edge anymore. Indeed, it would not be possible anymore to reach the destination with a delay lower than or equal to 10. Hence, IAK switches to the LD edges and finds the path *s-2-3-4-d* with cost 11 and delay 8.

DCUR, DCR, and IAK have been proposed with a distributed implementation in mind and allow only a limited range of choices at each node. Two algorithms have been proposed with the objective of enlarging the set of paths that can be found. First, Sriram *et al.* [216] proposed that each node maintains a list of ordered preferred output links. When path construction reaches a node, it selects its preferred output link for which the delay of the new path satisfies the delay constraint and that does not introduce a loop. A node may not have a preferred output link that satisfies these constraints; then path construction is backtracked to the previous node, which then selects its next preferred output link. If all preferred output links have been exhausted, then path construction is also backtracked to the previous node. Once the destination is reached, the algorithm terminates. Based on Sriram *et al.* [216] we will refer to this algorithm as ***SMS***. The list of preferred links is computed according to a *heuristic function*. In order to reduce runtime, the algorithm allows to limit the size of the list of preferred links at each node to a given parameter *p*. Nevertheless, depending on the heuristic function, this makes the algorithm incomplete. The algorithm is complete only if *p* is, at each node, greater or equal to the degree[2] of the node, thereby ensuring that all links are considered. If $\Delta(G)$ denotes the maximum degree of the nodes in graph *G*, the algorithm is complete if

$$p \geq \Delta(G). \tag{7}$$

Sriram *et al.* [216] define three heuristic functions: (1) *residual delay maximizing* (RDM), ordering links by their cost divided by the delay constraint minus the projected delay of the link (and ensuring that the edges belonging to the LC and LD paths towards the destination are included in the list), (2) *cost delay product* (CDP), ordering links by their cost times their projected delay, and (3) *partition-based ordering* (PBO), ordering links by cost value. RDM requires one SP tree run (LD), CDP two, and PBO none.

If the heuristic function is not efficient, the algorithm could explore an excessive number of paths before reaching the destination. This is especially true for dense topologies with many possible paths. To avoid this, Liu *et al.* [217] proposed ***SF-DCLC***, an algorithm similar to SMS. At each node, instead of computing a list of links and trying them one after the other, the algorithm chooses one output link based on a *selection function* (SF) which is proven to avoid loops and to lead to a solution if one exists. The links are assigned a weight equal to their cost plus (*i*) the cost of the LC path to the destination if the LC path to the destination passes the projected delay test, or, if not, (*ii*) the cost of the LD path. Links for which the LD path is infeasible are not considered. The least-weight link is then chosen. SF-DCLC is complete, but not optimal.

F. Other Approaches

For completeness, we briefly review in this section other QoS routing approaches that we do not include in our evaluation for the various reasons noted for the following algorithms. In order to the reduce the runtime of optimal algorithms, several fully polynomial ϵ-approximation algorithms have been proposed, e.g., [218]–[224]. The ϵ-approximation algorithms ensure to find a path whose cost is at most $(1 + \epsilon)$ times higher than the cost of the optimal path. Unfortunately, ϵ-approximation algorithms consider only integral costs and/or delays and are therefore not suitable for QoS routing with real-valued costs and/or delays.

Several algorithms have been proposed to accommodate imprecise state information, e.g., [12] and [225]–[229]. In centralized network architectures, such as SDN, is it reasonable to assume that the state is well-known and we hence do not consider the class of algorithms for imprecise state information. Also, note that algorithms considering imprecise information cannot provide strict (hard) QoS guarantees, rather these algorithms can only provide soft QoS guarantees. Similarly, algorithms based on probing techniques, e.g., [12], [226] and [229]–[231], or relaxing the constraint, e.g., [232], can also only provide soft QoS guarantees. Our focus is on QoS routing algorithms that can provide strict QoS guarantees and we do therefore not consider the algorithms for imprecise state information, probing, or relaxed constraints in detail in this survey.

Algorithms based on genetic algorithm (GA) [233], [234] and on artificial bee colony optimization techniques [235] have

[2]In a graph, the degree of a node corresponds to the number of edges connected to this node. In our scenario, we define the degree of a node as the number of *outgoing* edges the node has.

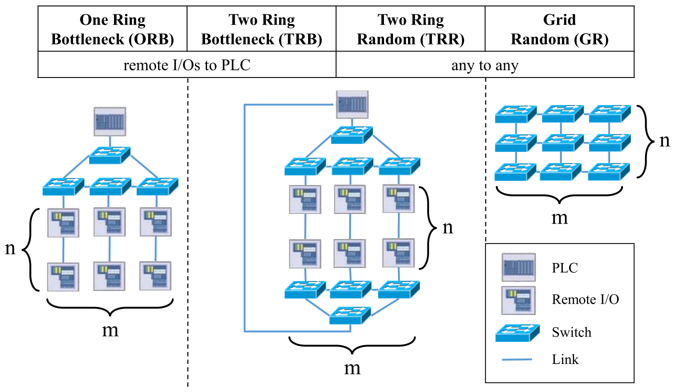


Fig. 8. The four topologies considered in the evaluation are based on three different base topologies which can be scaled in two different directions.

also been proposed. Such randomized algorithms have typically a fairly high runtime and are therefore not well suited for online routing decisions. Our focus is on QoS routing algorithms that are suitable for online routing decisions and we do therefore not cover these randomized algorithms in detail.

Pornavalai *et al.* [236], [237] simplify the bandwidth-jitter-delay constrained problem into an SP problem with maximum number of hops (i.e., a problem that can be solved in polynomial time) by using relationships between bandwidth, delay, jitter, and buffer capacity in weighted fair queuing (WFQ) set-ups. Our focus is on QoS routing algorithms that accommodate independent optimization and constraint metrics and we do therefore not consider [236], [237] in detail.

IV. Four-Dimensional (4D) Evaluation Framework

Generally, the performance of an algorithm depends on the specific scenario in which it is executed. In order to evaluate the behaviors of the different algorithms across a wide set of scenarios, we introduce an evaluation framework that evaluates QoS routing algorithms along four critical dimensions. First, we define four *topologies* which we describe in Section IV-A. A topology describes both the underlying structure of the network and the nodes that communicate with each other in the network. Second and third, we scale these topologies in two directions. Fourth, we distinguish requests based on the level of strictness of the delay constraint, see Section IV-B. Section IV-C presents the evaluation procedure and the metrics used, while Section IV-D identifies the evaluated algorithms.

A. Topology and Scaling

As first dimension of our evaluation framework, we define four topologies (shown in Fig. 8) based on three different base topologies. Although our survey is generic and all the algorithms can be applied to any CSP problem, we focus on industrial topologies where we expect centralized QoS routing to be extensively employed [133]. Nevertheless, the topologies we define are also common in and representative of data center, metro, grid, and enterprise networks. On the contrary, wide-area (star topology) networks are not covered, as strict centralized QoS routing in such environments is unlikely. All topologies can be scaled according to two scale parameters m and n that represent the size of the topology layout, as illustrated in Fig. 8 and defined in detail in the following for the four different topologies. The second and third dimensions of our evaluation framework correspond to varying the two scale parameters m and n from 4 to 13, thereby defining 100 different scalability levels. The four topologies are referred to as *One Ring Bottleneck* (ORB), *Two Ring Bottleneck* (TRB), *Two Ring Random* (TRR) and *Grid Random* (GR).

- *ORB:* The ORB topology consists of a base ring of $m+1$ switches. A so-called programmable logic controller (PLC) is connected to one switch of this ring. A branch composed of a series of n remote input/output nodes (I/Os), e.g., sensors, is connected to each of the other m switches of the ring. Thus, there are a total of mn I/Os. Remote I/Os have an internal switch allowing traffic to flow along the branches. Thus, remote I/Os act as traffic sources as well as traffic forwarders, which is common in sensor networks and industrial networks [51], [144], [145], [147]. Traffic is only considered from the remote I/Os to the PLC.
- *TRB:* The TRB topology extends the ORB topology with an additional ring consisting of $m+1$ switches. The $m+1$ switches connect the loose (bottom) ends of the m branches of remote I/Os (of the ORB topology) to the PLC. Traffic is still considered only from the remote I/Os to the PLC.
- *TRR:* The TRR topology is the same as the TRB topology, but traffic is now considered between any pair of remote I/Os. As the remote I/Os, the PLC is able to forward traffic not destined for it.
- *GR:* The GR topology is a grid of width m and height n. In the GR topology, traffic is considered between any pair of nodes.

We do not consider random topologies generated based on models, such as the Waxman model [238]. Instead, striving for a fair and reproducible evaluation, we only use deterministic topologies.

Each directed link is considered to have four output priority queues and routing is then performed on the corresponding queue-link topologies. For each physical link, the costs of the four queue-link edges with priority levels p, $p = 1$ (high priority), $2, 3, 4$ (low priority), are set to the values $1 + 1/p$ so as to favor the usage of low priority queues. The delay values are obtained with Schmitt's formula [135]. Thus, the costs and delays of the four queue-link edges are respectively set to 2 and 0.48 ms, 1.5 and 1.26 ms, 1.33 and 2.83 ms, as well as 1.25 and 7.55 ms.

Clearly, the number of queues as well as the cost and delay settings influence the performance of the algorithms and could be defined as additional comparison dimensions. However, in order to keep the evaluation tractable, we keep them static.

B. Delay Constraint Tightness

The delay constraint of routing requests can range from loose values for which the LC path is feasible to tight values for which no feasible path exist. Within this range, we define seven subranges of equal size, which we refer to as *delay levels*. The fourth dimension of our evaluation framework corresponds to varying the delay constraint of routing requests between these different delay levels.

C. Evaluation Procedure and Metrics

Each algorithm is evaluated along the four dimensions of our evaluation framework. For each particular topology and combination of the scale parameters m and n, we sequentially simulate 20,000 routing requests. The first 1000 requests are used as warm-up for the Java HotSpot optimizer and their results are not considered. For each request, the source and destination are generated uniformly randomly from the possible set of combinations defined by the topology and scale parameters. The delay constraint is distributed uniformly randomly among the seven delay levels [and then uniformly randomly within the selected delay level (delay constraint subrange)] so as to prevent the Java HotSpot optimizer from optimizing for a specific delay level. (If all test runs for a specific delay level are run successively, the Java HotSpot optimizer could exploit the consideration of a particular delay level in successive runs.)

For a given algorithm under test (AUT) and request, we run three algorithms. First, we run CBF in order to obtain the cost z_{opt} of the optimal solution. Second, we run the AUT to determine the AUT cost z'. The cost inefficiency (CI) of the AUT is then evaluated in % compared to the cost of the optimal path according to Eqn. (3). Third, we run an LD search using A* (which is then equivalent to an LDP search). We define the runtime of the AUT divided by the runtime of the LD search as the runtime ratio of the AUT. This normalization allows to filter out runtime variations due to the varying runtime behaviors of the testing machines (caused by operating system tasks or Java garbage collector execution). Indeed, both algorithms are run one after the other, i.e., within a short time window during which the runtime behavior of the testing machine can be assumed to be constant.

D. Algorithm Selection

Table II summarizes the algorithms that we have identified as suitable for the considered unicast QoS routing in Section III. We implemented all these 26 algorithms in Java 8[3] and, for each of them, ran our evaluation procedure. The specific parameter settings for parameterized algorithms will be given in Section V. We will identify parameterized algorithms by the name of the original algorithm to which we append the dash-separated parameter values in the same order as in Table II. For example, LARACGC with $\delta = 25\%$ will be referred to as *LARACGC-25*. We omit the λ parameter of H_MCOP and kH_MCOP since it has no influence in the CSP case.

[3]We acknowledge that the results may be subject to our specific implementations; however, we tried to be fair and optimize all implementations as much as we could.

V. Evaluation Results

Section V-A presents the evaluation results for the fourth dimension, i.e., the behavior of the algorithms for the different delay levels. Section V-B then focuses on the first three dimensions. Due to the high number of algorithms and the highly detailed results on how they behave and perform, it is not possible to present and discuss all results for all algorithms in detail in this article. Therefore, we only present the most interesting algorithms and discuss the most important conclusions. We have made the entire set of raw results and graphs for all the algorithms publicly available at http://www.lkn.ei.tum.de/lora [31].

We found that kSPMC, A*Prune, LARACGC, SCRC, E_MCOP, and the three SMS variations were not able to complete the evaluation in a reasonable amount of time compared to CBF. This leads to our first observation that algorithms using an ikSP algorithm to reach optimality have a very long runtime. Indeed, the considered queue-link topologies are dense with high numbers of possible paths. Thus, the number of paths to discover until reaching optimality is also high, yielding intractable runtimes for kSPMC, LARACGC, SCRC, and E_MCOP. A*Prune and SMS are not based on an ikSP algorithm but their structure is such that, if their initial search direction is not the correct one, they have to explore a high number of paths to reach the destination. The negative impact of this approach is accentuated by the high density of the considered queue-link topologies.

A. Fingerprints: Influence of the Delay Constraint Tightness

We analyze the fourth dimension using so-called *fingerprint* graphs (Fig. 9). The fingerprint graph for a given combination of topological and scale parameters m and n, shows the distribution of the runtime ratio (left, in red) and CI (right, in yellow) of an algorithm for the seven different delay levels (loose levels on the left and tight levels on the right). Since we have four different topologies with 100 different scalability levels (combinations of m and n values), each algorithm has 400 fingerprint graphs. Nevertheless, we observed that the shapes of all fingerprint graphs for a given algorithm are similar; Fig. 9 shows fingerprints for the grid (GR) topology with scale parameters $m = n = 10$. Since the shapes of these graphs characterize the different algorithms we refer to these graphs as *fingerprints*: they nearly uniquely identify an algorithm based on its behavior and are (nearly) always the same for a given algorithm. Only the absolute values vary depending on the topology and its scaling. These variations will be discussed in Section V-B.

1) Elementary Algorithms: Since the elementary LDP algorithm (see Section III-A) does not take cost into account, its CI is the benchmark for the worst acceptable CI (Fig. 9a). As expected, the CI of LDP gets better for tighter constraints since the LD path becomes closer to the optimal solution. In terms of runtime, as LDP is compared with itself, the LDP fingerprint shows that the accuracy of our runtime metric is reasonable (the 0.5 and 99.5 percentiles are close to one and the median is approximately one). In additional evaluations, which we cannot include due to space constraints, we

TABLE II
COMPREHENSIVE LIST OF CONSTRAINED SHORTEST PATH (CSP) AND MULTI-CONSTRAINED SHORTEST PATH (MCSP) ALGORITHMS, WHICH CAN BE EMPLOYED FOR DELAY-CONSTRAINED LEAST-COST (DCLC) QOS ROUTING. THE ALGORITHMS ARE CATEGORIZED ACCORDING TO THE UNDERLYING ALGORITHMIC STRATEGY INTO ALGORITHMS BASED ON PRIORITY QUEUES, BELLMAN-FORD, LAGRANGE RELAXATION, AS WELL AS LEAST-COST (LC) AND LEAST-DELAY (LD) PATHS. FOR EACH ALGORITHM, WE INDICATE THE TYPE(S), I.E., CSP OR MCSP OR *k* PATH VERSIONS THEREOF, AS WELL AS OTHER KEY CHARACTERISTICS, INCLUDING OPTIMALITY PROPERTY AND THE ACCEPTED PARAMETERS. WE INDICATE THE NUMBER OF UNDERLYING ALGORITHM RUNS, E.G., ITERATIVE KSP (IKSP) AND STATIC KSP (SKSP) ALGORITHMS (SEE SECTION II-B FOR DEFINITIONS). WHEN THE EXACT NUMBER OF RUNS DEPENDS ON THE SPECIFIC SCENARIO, THE POSSIBLE NUMBERS OF RUNS ARE INDICATED THROUGH A COMMA-SEPARATED LIST OR A RANGE (WITH THE ARROW (→) SYMBOL) WITHIN PARENTHESES. UNBOUNDED NUMBERS OF RUNS ARE INDICATED WITH THE GREATER OR EQUAL (≥) SIGN. WE NOTE THAT AN ALGORITHM USING A SKSP ALGORITHM CAN BE IMPLEMENTED WITH AN IKSP ALGORITHM

Algorithm	Type	Number of runs of underlying algorithms					Optimal	Complete	Distr.	Param.
		ikSP	skSP tree	skSP	SP tree	SP				
Elementary Algorithms (Sec. III-A)										
LDP	CSP					1		✓	if SP is	
FB [193]	CSP, MCSP					$(1 \rightarrow M + 1)$		if CSP		
kSPMC	(k)CSP, (k)MCSP	1					✓	✓		
Priority Queue Based Algorithms (Sec. III-B)										
CBF [194]	CSP						✓	✓		
A*Prune [195]	(k)CSP, (k)MCSP						✓	✓		
Algorithms Based on Bellman-Ford (Sec. III-C)										
DCBF [196]	CSP				1			✓	✓	
kDCBF [196]	CSP		1					✓	✓	k_d, k_c
DEB [197]	CSP							✓		
Algorithms Based on the Lagrange Relaxation (Sec. III-D)										
LARAC [200]–[203]	CSP					≥ 1		✓		MD
LARACGC [201]	CSP	(0, 1)				≥ 1	if $\delta = 0$	✓		δ
SCRC [205]	CSP	(0, 1)				(1, 2)	✓	✓		
kLARAC [196]	CSP			≥ 1				✓		k
H_MCOP [206]	CSP, MCSP				1	(0, 1)		if CSP		λ
kH_MCOP [206]	CSP, MCSP			(0, 1)	1			if CSP		λ, k
NR_DCLC [208]	CSP				≥ 0	≥ 1		✓		
MH_MCOP [207]	CSP, MCSP				$(1 \rightarrow H + 1)$	$(0 \rightarrow H + 1)$		if CSP		H
E_MCOP [207]	CSP, MCSP	$(0 \rightarrow 2)$				$(1 \rightarrow M + 1)$	✓	✓		
DCCR [209]	CSP			(0, 1)		(1, 2)		✓		k
SSR+DCCR [209]	CSP			(0, 1)		$(1 \rightarrow L + 2)$		✓		L, k
Algorithms Based on LC and LD Paths (Sec. III-E)										
DCUR [212], [213]	CSP				(1, 2)			✓	✓	
DCR [214]	CSP				(0, 1)	1		✓	✓	
IAK [215]	CSP				1	(0, 1)		✓	✓	
SMS-RDM [216]	CSP				1			if $p \geq \Delta(G)$	✓	p
SMS-CDP [216]	CSP				2			if $p \geq \Delta(G)$	✓	p
SMS-PBO [216]	CSP							if $p \geq \Delta(G)$	✓	p
SF-DCLC [217]	CSP				(1, 2)			✓	✓	

observed that FB exhibited, as expected, exactly the same CI behavior as LDP; except when the LC path is feasible, then the LC path returned by FB (after one SP run) is optimal. When the LC path is infeasible, then FB returns the LD path (which is rarely optimal) at a total cost of two SP runs.

2) Priority Queue Based Algorithms: The benchmark for the highest acceptable runtime ratio is given by CBF, an optimal algorithm based on a priority queue (Section III-B) (Fig. 9b). CBF was the fastest optimal algorithm. Since CBF terminates when the paths it expands have delays higher than the constraint, it terminates earlier for tighter constraints and its runtime therefore improves as the delay constraint gets tighter. The CI of CBF is always zero since CBF is optimal.

3) Algorithms Based on Bellman-Ford: The Bellman-Ford based DCBF algorithm (Fig. 9c) has a CI fingerprint with slightly decreasing runtimes and increasing CI for increasingly tight delay constraints. For tight delay constraints, the delay test during the Bellman-Ford run fails more often and hence allows Bellman-Ford to terminate earlier (as it stops when no relaxation occurs in an iteration) and DCBF therefore gets faster as the delay constraint gets tighter. In additional evaluations we have observed that kDCBF (not included in Fig. 9) has similar shapes, however which much lower CIs and longer runtimes. For example, kDCBF-2, divides the CI by a factor of approximately two, but increases runtime by a similar factor.

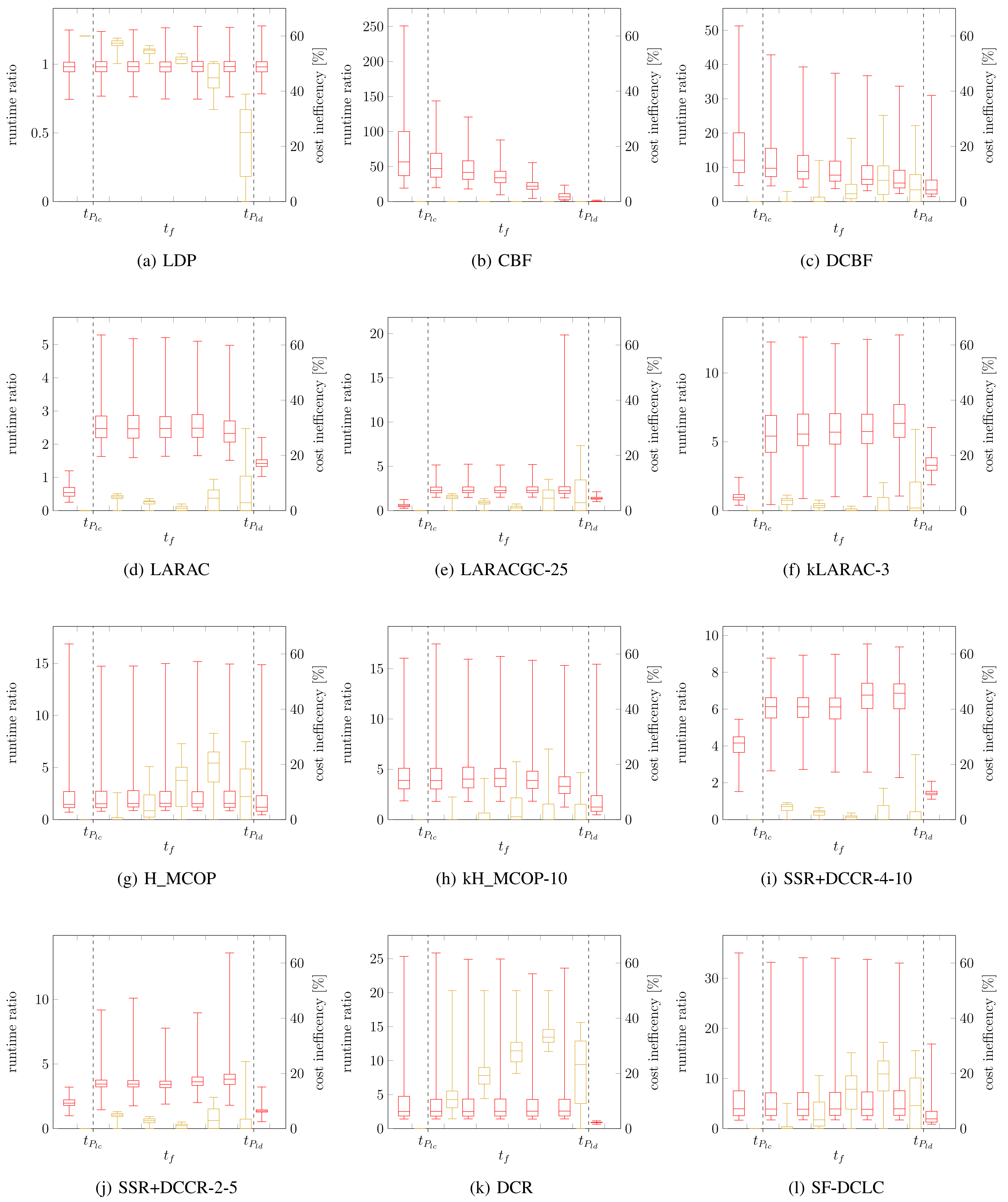


Fig. 9. Fingerprints for selected QoS routing algorithms. These graphs show, for the grid (GR) topology with $m = n = 10$, i.e., 10×10 switching nodes, the runtime ratio (runtime of algorithm normalized by runtime of LD search, plotted in red on left) and cost inefficiency (in yellow on right) of the algorithms for the seven different delay levels t_f (delay constraint subranges, whereby loose delay constraints are on the left and tight delay constraints are on the right). $t_{P_{ld}}$ and $t_{P_{lc}}$ denote the delays of the LD and LC paths, respectively. Since the rightmost delay level corresponds to an infeasible problem (delay constraint t_f lower than the delay $t_{P_{ld}}$ of the LD path), no cost inefficiency value is shown and the runtime then corresponds to the time required to detect that the problem is infeasible. While the cost inefficiency scale is the same for all the algorithms, the runtime scales have to differ because of the high variability between the algorithms. The lower and upper whiskers of the boxplots, respectively, correspond to the 0.5% and 99.5% percentiles.

4) Algorithms Based on Lagrange Relaxation: Similar to FB, LARAC (Fig. 9d) can find the optimal solution with one LC search when the LC path is feasible. Fig. 9d (for the left-most, i.e., loosest delay constraint level t_f) shows that this run is roughly two times faster than an LD search. This is due to the fact that the delay and cost values have ranges of different absolute sizes and the guess function of A* is better for the costs because the costs have a smaller range size than the delay values, i.e., have a range size closer to the one of the least-hop count used for the guess (which is zero). When the problem is infeasible, LARAC notices the infeasibility with an additional LD search. For intermediate delay levels, LARAC requires a few additional SP runs, hence leading to slightly higher runtimes. Nevertheless, these additional runs are worth it as we can observe that the CI of LARAC stays then much lower than 10% in most cases.

While LARACGC did not complete the evaluation within a reasonable amount of time, LARACGC with $\delta = 25\%$ (Fig. 9e) did. As LARAC has a CI higher than 25% only for the tightest feasible delay level, LARACGC-25 only behaves differently than LARAC for this tightest feasible delay constraint. As expected, LARACGC-25 then brings the CI to less than 25% but at a high runtime cost even for such a small gap closing (as the CI of LARAC is at most 30%). This indicates that the gap closing is expensive in terms of runtime and probably not worth it. The high runtime is likely due to dense queue-link topology structures of our evaluation networks and confirms that algorithms based on an ikSP algorithm are not efficient for dense network topologies.

We observe from Fig. 9f that kLARAC with $k = 3$ has the same shape as LARAC, however with longer runtime and lower CI. This is expected, since kLARAC runs an skSP at each iteration, allowing to find lower cost paths but with longer runtime. We nevertheless observe that this cost reduction comes with a much less pronounced runtime increase than for LARACGC.

The fingerprint of H_MCOP (Fig. 9g) shows the difference in runtimes between SP searches and SP tree searches. Indeed, for detecting an infeasible problem, H_MCOP first computes a reverse SP tree. As can be seen, this has a much longer runtime than the single LD search of LDP. More precisely, the H_MCOP median runtime is only slightly longer, but the 99.5% percentile of the runtime is much higher than for LDP. This shows that comparing the runtime of algorithms in terms of "Dijkstra runs" independently of whether these are SP or SP tree runs, as done in some papers, is not a valid metric. For all other cases, H_MCOP requires an additional forward SP search. The H_MCOP runtime for these delay levels is hence always similar and slightly higher (by 0.5 since it is an LC search) than for the infeasible delay level. In terms of CI, H_MCOP interestingly presents a fingerprint of different shape than LARAC, LARAGC-25, and kLARAC-3. While the different LARAC versions have a U-shaped CI fingerprint, H_MCOP reaches higher CIs for problems with tighter constraints but improves again for the tightest feasible delay level. In terms of absolute values, the CIs of H_MCOP are usually slightly worse than for the different LARAC versions, except when the delay constraint is loose, where H_MCOP and LARAC perform similarly. When using Chong's algorithm with $k = 10$ (Fig. 9h), we see that the runtime is only slightly increased while the CI is substantially improved. Indeed, kH_MCOP-10 reaches optimality in nearly 50% of the cases.

In additional evaluations we found that NR_DCLC (which is not shown in Fig. 9) has a similar, but slightly better, CI fingerprint compared to H_MCOP; which is expected since NR_DCLC uses H_MCP (i.e., H_MCOP) as underlying algorithm. On the other hand, the NR_DCLC runtime is much longer, except in the cases where the LC path is feasible or where the problem is infeasible, in which cases NR_DCLC uses SP runs to detect these situations. Within the feasible delay levels, the runtime of NR_DCLC gets shorter as the delay constraint gets tighter. Indeed, NR_DCLC starts with an LD search and then improves on this path. When the delay constraint gets tighter, the LD path is closer to the optimal solution and NR_DCLC hence has less work to do. Additional evaluations have shown that MH_MCOP (which is not shown in Fig. 9) improves the CI of H_MCOP by a factor of around two at the expense of a twofold runtime increase. The H parameter can then be used to tweak the CI/runtime trade-off. While the CI fingerprint of MH_MCOP is similar to the one of H_MCOP, the MH_MCOP runtime fingerprint exhibits a Gaussian bell curve shape. This is due to the fact that MH_MCOP improves on the solution of H_MCOP. Hence, the amount of work it has to perform depends on the CI of H_MCOP, which is similar to a Gaussian bell curve.

In additional evaluations we also found that DCCR-3 (which is not shown in Fig. 9) has a high CI (between 20% and 45% in most cases). On the other hand, SSR+DCCR (Fig. 9i and Fig. 9j) is interesting. Since SSR+DCCR improves the LARAC solution or one of the intermediate LARAC results, SSR+DCCR has a similar CI fingerprint as LARAC. Interestingly, SSR+DCCR especially improves the solution of LARAC when the delay constraint is tight but still feasible. We observe that SSR+DCCR is, similar to LARACGC, closing the gap of LARAC. Nevertheless, SSR+DCCR appears more powerful than LARACGC for our dense network scenarios since SSR+DCCR runtimes stay relatively low compared to those of LARACGC-25. As expected, SSR+DCCR-2-5 reduces the runtime compared to SSR+DCCR-4-10; whereas the CI is not strongly affected. Hence, the tuning of the SSR+DCCR parameters requires additional evaluations and is left for future research.

5) Algorithms Based on the LC and LD Paths: The studies on this type of algorithms usually assume that the LD and LC trees can be computed once and then reused for each request, thereby leading to a low request provisioning time. However, in our scenario, we assume that the delay and cost of the edges can change inbetween requests and we therefore have to recompute the tree for each request. DCUR, DCR, and IAK always follow edges belonging to the LC and LD paths towards other nodes. For our specific queue-link cost and delay settings, which are identical for each physical edge, we have observed that all these algorithms either follow the LC path

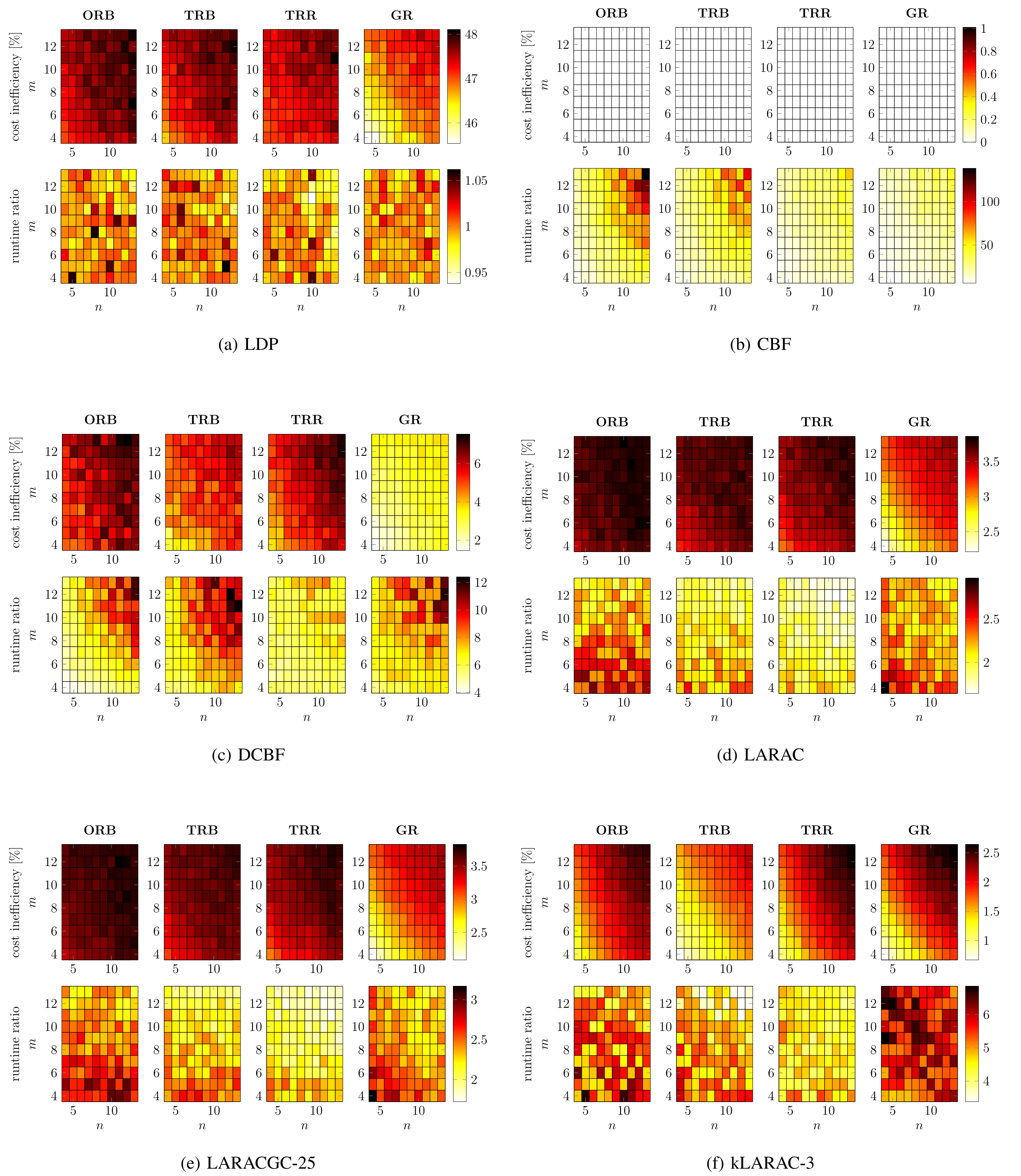
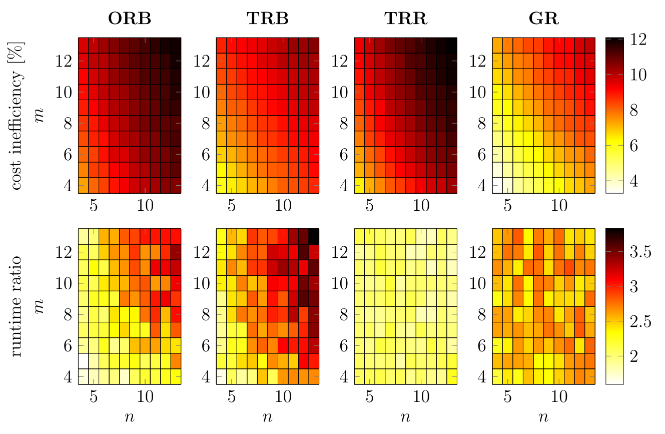
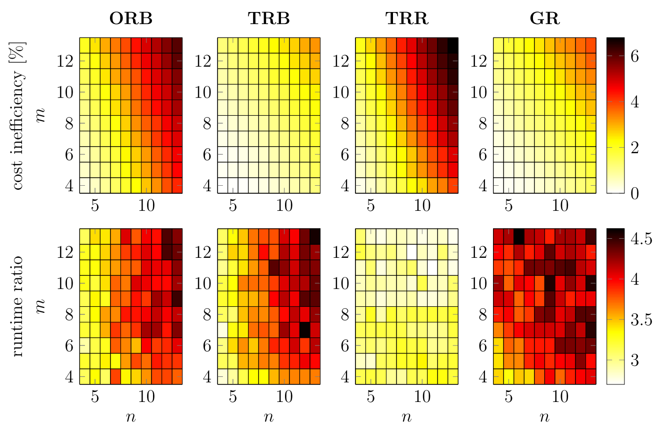


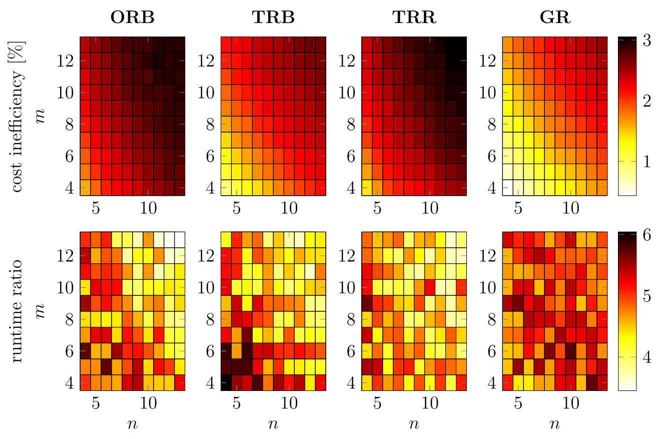
Fig. 10. Heatmaps showing the behaviors of selected QoS routing algorithms for different topologies and scalability levels. For a given algorithm, the four upper heatmaps show the cost inefficiency (CI) for the four different topologies, and the four lower heatmaps show the runtime ratio. A given heatmap shows the CI or runtime ratio as a function of the scale parameters $n = 4, 5, \ldots, 13$, and $m = 4, 5, \ldots, 13$, i.e., for a total of 100 different scalability levels. Each cell corresponds to the average results of 20,000 requests (with randomly drawn delay constraints from across the seven considered delay constraint levels and corresponding subranges) simulated for this specific n and m combination. To prevent outliers from biasing the results, only values between the 1% and 99% percentiles are considered for computing the average. Unfortunately, because of the high variability between the algorithms, the scales are different for each algorithm.



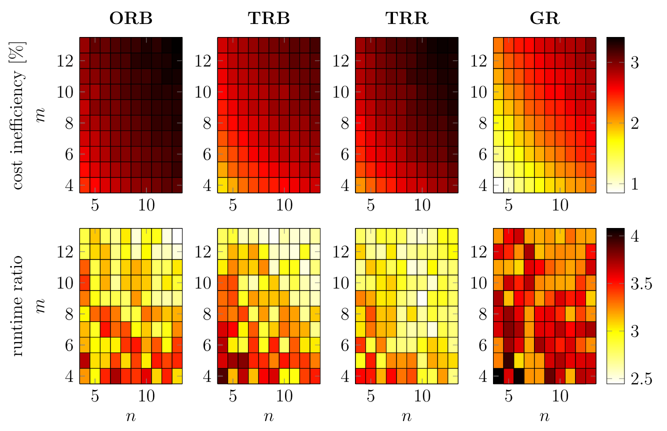
(g) H_MCOP



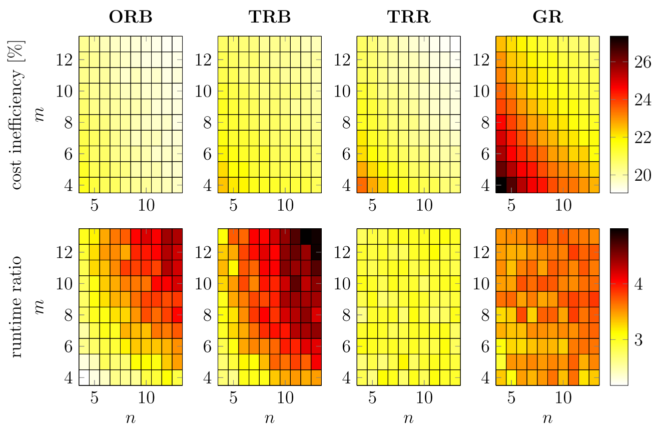
(h) kH_MCOP-10



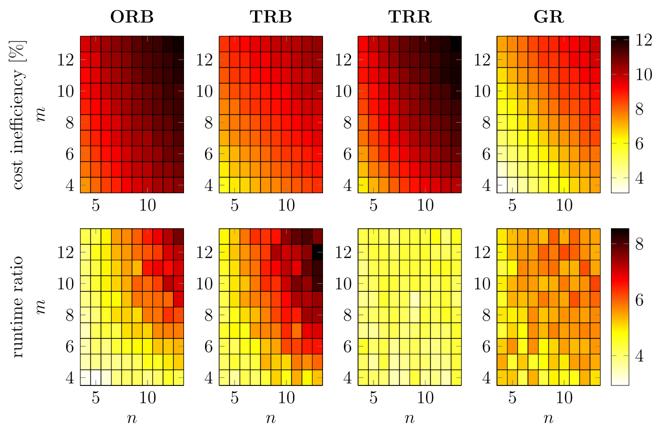
(i) SSR+DCCR-4-10



(j) SSR+DCCR-2-5



(k) DCR



(l) SF-DCLC

Fig. 10. Continued.

and then switch to the LD path until the end, or vice versa. This results in the same cost and these four algorithms hence present exactly the same CI behavior. Although this shows that further study is required with different queue-link cost and delay settings to specifically research the subtle differences in the DCUR, DCR, and IAK dynamics, it also highlights that features of algorithms do not always bring some benefit. Indeed, while DCR and IAK only switch once between following the LC and LD paths towards the destination, DCUR can switch any number of times. Since DCUR executes several SP tree runs, it is slower than IAK and DCR which only run one SP search and one SP tree search. However, these extra computations appear to be useless in our scenario. Because the SP tree search is the most expensive search and because DCR runs an LC SP tree search which enjoys a better guess function than an LD SP tree search, DCR is on average slightly faster than IAK. We show therefore only DCR in Fig. 9k. The fingerprint of DCR again shows the difference between SP and SP tree runs. While DCR detects an infeasible problem fast with an LD SP search, the following LC SP search for the other cases is much more time consuming, at least in the worst-case.

The runtime ratio of SF-DCLC (Fig. 9l) corresponds to two SP tree searches, except for the infeasible problem where one is enough. As SF-DCLC has more options at each hop, it achieves a better CI fingerprint than DCUR (which is the same as DCR, as noted above), but still with a similar shape. More precisely, on average, the DCUR CI is approximately twice the SF-DCLC CI. Interestingly, we observe that SF-DCLC and H_MCOP have very similar CI fingerprints, although they are very different in terms of implementation. Examining the output of the algorithms closely, we noticed that, for identical requests, they always returned paths with identical costs. In particular, we observed that SF-DCLC and H_MCOP both prefer one path over the other either based on the cost or on the delay metric depending on whether these paths are feasible or not. Even though SF-DCLC proceeds node by node and H_MCOP within an SP search, both algorithms find typically the same paths (i.e., they nearly always find identical paths and in the rare cases where the paths are different, the paths have identical costs), at least for our specific delay and cost distributions.

6) Summary: In general, it is interesting to note that most algorithms exhibit a higher variability of the CIs when the delay constraint is tight (but still feasible). This can be explained by the fact that there are relatively few possible paths. Hence, if the best one is not chosen, the cost can quickly increase. Interestingly, Chong's algorithm [186] appears to be a good tool to resolve this issue, as has been demonstrated by kH_MCOP, and SSR+DCCR. In additional evaluations (not included in Fig. 9), we found that DCCR and kDCBF also show this behavior.

B. Heatmaps: Impact of Network Topology and Scale

In order to observe the behaviors of the algorithms for the different topologies and scalability levels, we collapse the fourth dimension (delay constraint tightness) of our evaluation framework by retaining only the average runtime ratio and CI over all delay constraint levels.[4] This yields the heatmaps shown in Fig. 10. Specifically, each cell of each sub-figure in Fig. 10 corresponds to a fingerprint plot, whereby the fingerprint plots for the grid (GR) topology with $n = m = 10$

[4]For both the runtime ratio and the CI, only values between the 1% and 99% percentiles were considered for the computation of the average.

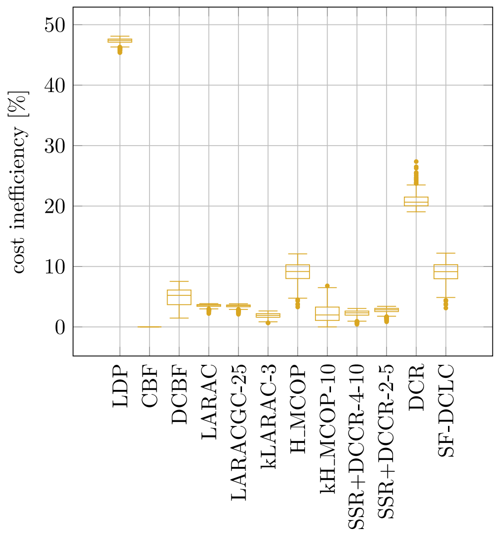


Fig. 11. Box plots of the average cost inefficiencies (as computed from the heatmaps of Fig. 10) of the different algorithms shown in Fig. 10 over the different topologies and scalability levels.

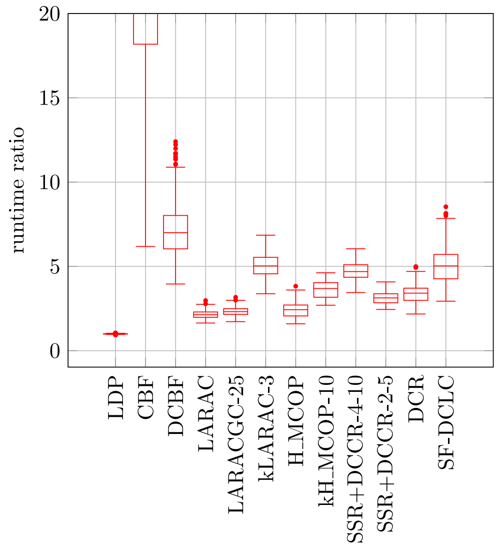


Fig. 12. Box plots of the average runtime ratios (as computed from the heatmaps of Fig. 10) of the different algorithms shown in Fig. 10 over the different topologies and scalability levels.

have been shown in Fig. 9, the other fingerprints are available at [31]. While observing the scalability of the different algorithms with these heatmaps, the reader should pay attention that the scalability of the algorithms is compared to an LD search. That is, if an algorithm presents the same runtime ratio for all the scalability levels of a topology, that does not mean that its runtime is always the same but rather that the considered algorithm has *similar scaling behavior* as an LD search.

Due to the vastly different cost inefficiencies (CIs) and runtime ratios of the different algorithms, Fig. 10 has different scales for the different algorithms. In order to facilitate comparisons, Figs. 11 and 12 show boxplots of the values of the heatmaps of the different algorithms from Fig. 10 on a common scale.

1) Elementary Algorithms (LDP as Runtime Benchmark): The lower four plots for the LDP runtime, see Fig. 10a, confirm that our runtime metric has, on average, an inaccuracy of less than 6%. These inaccuracies are mainly due to measurements errors and side-effects from the operating system and Java. This indicates that our LDP based runtime metric provides a valid runtime reference benchmark across the three evaluation dimensions of topologies and scale parameters *m* and *n*.

2) Priority Queue Based Algorithms: The heatmaps of CBF (Fig. 10b) illustrate the limitation of CBR: the CBF runtime grows exponentially with the size of the network, which is consistent with the observations in [194].

3) Algorithms Based on Bellman-Ford: The DCBF CI (Fig. 10c) is only slightly affected by the size of the topology. Interestingly, the DCBF CI is much better, i.e., lower by a factor of two, in the GR topology compared to the other three considered topologies. These DCBF results can be intuitively explained with the DCBF sub-optimality behavior of "relaxing too much", see Section III-C. The DCBF sub-optimality scenario occurs more frequently when paths have to share nodes, because only one path is kept at each node. The grid topology has a lot of diversity (dense graph), thus DCBF sub-optimality arises only rarely. In the other topologies, paths frequently share nodes (because of the branches in the topologies), which leads to frequent DCBF sub-optimality scenarios and therefore to a higher CI.

In additional evaluations that are not included in Fig. 10, we observed that DEB has a high CI (between 12% and 35% on average). The high DEB CI is due to the fact that DEB tries to reduce the CI by checking paths of different hop counts. Nevertheless, in our queue-link topologies, we have many paths of identical length and the CI can be dramatically changed simply by choosing different queues at each hop and hence without changing the path length.

4) Algorithms Based on Lagrange Relaxation: As already observed for the fingerprints, we observe that the runtime increase of LARACGC-25 (Fig. 10e) compared to LARAC (Fig. 10d) is not worth the slight CI reduction achieved with LARACGC-25. The runtime increase we observe here is not substantial since it only happens for one of the seven delay levels (as observed in the fingerprints, see Section V-A). Interestingly, while LARAC and LARACGC-25 behave better, in terms of CI, for the GR topology, kLARAC-3 (Fig. 10f) behaves better for the TRB topology. Fig. 10d, Fig. 10e, and Fig. 10f indicate an interesting property of LARAC algorithms: in terms of runtime, they scale better than an LD search, but only in the *m* scale direction; the *n* scale dimension affects them as it affects an LD search. In additional evaluations we observed that different *MD* parameters for LARAC do not affect the LARAC scalability behavior, but only change the absolute values [31].

The H_MCOP runtime (Fig. 10g) scales worse than an LD search in both directions (*m* and *n*) for the ORB and TRB topologies. On the other hand, the H_MCOP runtime exhibits much better scaling behavior for the two other topologies (TRR and GR). While H_MCOP reaches low CI for small topologies, we observe that the H_MCOP CI grows quickly for

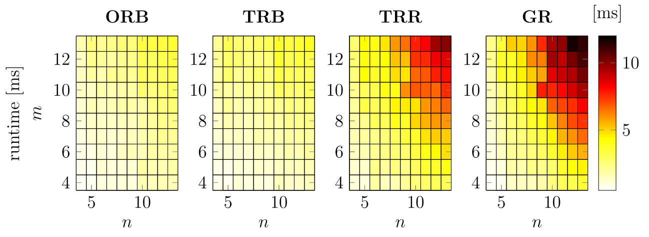


Fig. 13. Absolute runtime [ms] of an LD search (each cell is an average over the different delay levels, analogous to Fig. 10) for the different topologies and scalability levels. The runtime was measured on an Intel Core i7-3770 @ 3.40GHz.

larger topologies. Fig. 10h shows that using Chong *et al.* [186] with H_MCOP does not change its scalability. Indeed, while kH_MCOP-10 is then able to reach optimality for small topologies, its CI grows quickly as the topology sizes increase. Nevertheless, this dramatic CI reduction only leads to a slight increase in runtime (by roughly 1 unit). In additional evaluations, MH_MCOP exhibited an identical scaling behavior, though still improving the CI by a factor of around two at the expense of approximately doubling the runtime.

SSR+DCCR (Fig. 10i and 10j), scales similarly to the underlying LARAC algorithm.

5) Algorithms Based on the LC and LD Paths: DCR (Fig. 10k) and SF-DCLC (Fig. 10l) present a similar scaling behavior, in terms of runtime, as H_MCOP. Thus, the similar H_MCOP, DCR, and SF-DCLC runtime scaling behaviors appear to indicate the scaling behavior of an SP tree search compared to an SP search. DCR, and hence DCUR, and IAK, exhibit the interesting behavior that their CI improves as the topology scales up. This is unfortunately not true for the runtimes of DCR, DCUR, and IAK. These are the only algorithms showing this behavior. SF-DCLC and H_MCOP show the exact same cost (CI) behavior, hence confirming our observation that they actually always return equal cost paths (see Section V-A5).

6) General Impact of Topology: We conclude the discussion of the heatmaps in Fig. 10 by briefly summarizing the general impact of the type of topology. We observe from Fig. 10 that most algorithms have a better CI for the GR topology than the other three topologies, except DCR (and DCUR and IAK) whose CI behavior is opposite to all the others. This observation appears to indicate that switching between LC and LD paths brings improvements for large topologies.

Another common observation is that all algorithms have generally shorter runtime for the TRR topology than the TRB topology. The only difference between the TRB and TRR topologies is the set of communicating nodes. Nevertheless, we observe that this small difference has a major impact on the runtime of most algorithms (though similar in that they get faster in TRR).

C. Absolute Runtime in Practice

While the runtime ratio facilitates the relative comparison of the algorithms in different setups, the absolute runtime is also a relevant metric for networking practice. Fig. 13 shows heatmaps of the absolute runtimes (averaged over the delay constraints considered for Fig. 10) of an LD search for the different topologies and scalability levels. The runtime measurements were performed on an Intel Core i7-3770 @ 3.40 GHz. Fig. 13 indicates that the runtimes for LD search scale worse in the TRR and GR topologies than in the ORB and TRB topologies. As the LD search runtimes are mainly on the order of milliseconds, most algorithms of Figs. 9 and 10 have average absolute runtimes on the order of tens of milliseconds. On the other hand, CBF can reach average runtimes of up to 500 ms, which justifies the need for faster algorithms.

D. Which Algorithm Is Best?

After analyzing the behaviors of all the algorithms, we are in a position to address the question: *which algorithm is the best?* From our observations, the answer is: *it depends*. Indeed, none of the algorithms is better than all others in terms of both runtime and CI for all topologies, scalability levels, and delay levels.

The first "it depends" consideration is in regard to the relative importance of cost and runtime. While kLARAC, kH_MCOP, and all optimal algorithms are good solutions if the cost is the most important criterion, algorithms, such as LDP, FB, or H_MCOP, should be preferred if a very short runtime is critical. LARAC and SSR+DCCR are algorithms that achieve relatively good performance for both the cost and runtime performance metrics.

Secondly, the selection of the best QoS routing algorithm depends on the specific region in the 4D evaluation space where the algorithm is supposed to operate. Indeed, we have seen that for small topologies and/or tight delay constraints, CBF remains a very good candidate. As the topology grows, algorithms with better scalability are needed. In terms of cost, DCR, DCUR, and IAK are the only algorithms with decreasing CI for large topologies and these algorithms are therefore good choices for very large topologies. In terms of runtime, only the different LARAC and SSR+DCCR variations scale better than an LD search and are therefore also good candidates for large topologies. Therefore, the only way of selecting the best QoS routing algorithm for a given scenario is to consider the evaluation for the specific planned usage scenario.

Nevertheless, we can identify LARAC and SSR+DCCR as being among the best QoS routing algorithms at any point of the 4D evaluation space. That is, for any topology, and topology scale, and delay level, LARAC and SSR+DCCR are among the best performing algorithms. Indeed, on average, for the simulated topologies and network scales, both LARAC and SSR+DCCR keep their runtime ratio lower than four and their CI lower than 4%. While LARAC and SSR+DCCR scale well in terms of runtime, their CI grows only slightly for large topologies. Moreover, their behavior on the fourth dimension, i.e., for the different delay levels, is quite stable. Last but not least, both LARAC and SSR+DCCR accept several parameters that allow to tailor them to specific usage scenarios.

VI. Lessons Learned

Besides the results that directly followed from the evaluations, this work allowed us to discover and learn anecdotal facts that are interesting to mention.

We sometimes observed particularly good runtime behaviors when not expected. It turned out that the optimizations done by Java can have a strong impact on the runtime. For example, when running an algorithm twice in a row, the second run can be up to two times faster than the first run because of the Java HotSpot optimizer. We also noticed that introducing a sequence number for ordering equal elements in priority queues (instead of having their order undefined) reduced the runtime of the algorithms by a factor of up to two. We note that this sequence number does not influence our comparison since all algorithms benefited from this enhancement. While we could not find the exact cause of this sequence number effect, we suspect that the sequence numbering facilitates Java optimizations.

We noticed that the pseudo-code of CBF [194], which is the optimal algorithm referenced by many studies in the field, includes an unnecessary iteration. Indeed, the commonly referenced CBF pseudo-code stores a list of paths at the destination and terminates when the last expanded path has a delay higher than the deadline. If this last path (which is, by definition, infeasible) actually reaches the destination and has a lower cost than the current best path at the destination (which happens rarely, typically once in 1000 runs), it will be stored in the list of paths at the destination. The algorithm then solves this issue of having an infeasible path in the list by saying "*to find a constrained path, take the first path in the list that meets the constraint*" [194]. It is actually possible to avoid to check if the paths in the list are feasible by terminating the algorithm *before* expanding the first path with a delay higher than the deadline. In such a way, one iteration is avoided and it is unnecessary to do a feasibility check on the final list. We considered CBF without the unnecessary iteration in our evaluation study.

Interestingly, one of the best algorithms, LARAC, is actually the oldest (initially proposed in 1978) of all the compared algorithms. This shows that, even in modern research, old approaches should not be ignored during literature research. Curiously, the LARAC algorithm has been proposed four times in (nearly) identical forms. The authors of the original LARAC proposal [200] later proposed another algorithm which is based on an implicit enumeration of all possible paths [192]. This later proposed approach [192] is computationally much more demanding than their original proposal [200], especially for dense topologies, such as queue-link structures. They actually made this second proposal when they noticed that their initial proposal was not optimal (because of the duality gap). We have seen that their initial proposal is quite close to optimality. This illustrates that hunting for optimality does not always result in better algorithms and that consistent extensive benchmark evaluations are required to assess the performance of QoS routing algorithms.

VII. Conclusion

Many communication networks and applications have stringent Quality of Service (QoS) requirements, including industrial communication networks and multimedia applications.

For instance, industrial communication networks carry critical messages which have strict end-to-end delay requirements. Similarly, many multimedia applications require timely packet arrivals in order to continue media playback. QoS routing algorithms that support the finding of paths that meet delay constraints while minimizing a cost metric, i.e., so-called delay-constrained least-cost (DCLC) routing algorithms, can greatly help networks in meeting QoS requirements. Due to the distributed control in Internet Protocol (IP) networking, routing research for communication networks has generally focused on distributed computation in the past. The centralized control in Software-Defined Networking (SDN) presents a fundamental paradigm shift in the control of communication networks—including the control of routing—towards centralized control mechanisms. In order to facilitate the selection and further development of QoS routing algorithms for SDN based networks, this article provided a comprehensive up-to-date survey of QoS routing algorithms, including their quantitative performance characteristics, from the perspective of centralized computation.

More specifically, this survey article first presented a comprehensive review of the state-of-the-art in unicast QoS DCLC routing algorithms and identified algorithms suitable for a wide range of centralized network scenarios, including demanding dense and large network topologies. We introduced a novel four-dimensional (4D) evaluation framework for QoS routing algorithms, which includes the type of topology (including the sets of communicating nodes), the scale (size) of the network, and the tightness of the delay constraint. We evaluated all identified unicast QoS routing algorithms with centralized computations within this 4D evaluation framework.

We observed that the performance of the different algorithms is highly dependent on the specific scenario and that there is no one universal best QoS routing algorithm for all scenarios. Indeed, all algorithms have different behaviors depending on the considered region of the 4D evaluation space. Therefore, the selection of the best algorithm depends on the considered evaluation space region. Hence, in order to select the most appropriate algorithm for a given scenario, a specific evaluation is required. Nevertheless, we observed some general trends. First, algorithms using an iterative k shortest path (ikSP) algorithm to reach optimality or a given optimality level, have a very long runtime. Algorithms making use of shortest path tree (SP tree) computations have much shorter runtimes, but are outperformed by algorithms that use only the results of single-source single-destination shortest path (SP) runs. That is, evaluating the runtime of an algorithm in terms of number of Dijkstra SP tree runs is not a valid runtime metric, as the runtime of an SP tree Dijkstra run can be much longer than the runtime of an SP Dijkstra run. Second, we identify two algorithms, LARAC and SSR+DCCR, that achieved relatively good performance in most of the 4D evaluation space. Moreover, these two algorithms accept parameters that allow to tweak their behavior to a specific usage scenario.

Although we included the most critical dimensions in our evaluation framework, other dimensions can be considered in future work. For instance, a varying number of multiple constraints could be an additional evaluation dimension. Another future work direction could be to survey and evaluate multicast and multipath QoS routing algorithms. Furthermore, future work could examine the surveyed QoS routing algorithms in the context of frameworks that seek to reduce the complexity of SDN routing, e.g., [239]–[242].

Acknowledgment

The authors would like to thank Péter Babarczi, Andreas Blenk, Mu He, Qian Liu, Petra Stojsavljevic, Lionel Van Bemten, and Samuele Zoppi for their useful feedback and comments.

References

[1] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.

[2] V. C. Gungor *et al.*, "A survey on smart grid potential applications and communication requirements," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 28–42, Feb. 2013.

[3] B. W. Carabelli, R. Blind, F. Dürr, and K. Rothermel, "State-dependent priority scheduling for networked control systems," in *Proc. Amer. Control Conf. (ACC)*, Seattle, WA, USA, May 2017, pp. 1003–1010.

[4] A. Blenk, A. Basta, J. Zerwas, M. Reisslein, and W. Kellerer, "Control plane latency with SDN network hypervisors: The cost of virtualization," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 366–380, Sep. 2016.

[5] M. Karakus and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017.

[6] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*, 7th ed. Boston, MA, USA: Pearson, 2017.

[7] D. B. Rawat and S. R. Reddy, "Software defined networking architecture, security and energy efficiency: A survey," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 325–346, 1st Quart., 2017.

[8] S. Sezer *et al.*, "Are we ready for SDN? Implementation challenges for software-defined networks," *IEEE Commun. Mag.*, vol. 51, no. 7, pp. 36–43, Jul. 2013.

[9] J. Jaffe and F. Moss, "A responsive distributed routing algorithm for computer networks," *IEEE Trans. Commun.*, vol. 30, no. 7, pp. 1758–1762, Jul. 1982.

[10] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.*, vol. 25, no. 1, pp. 73–85, Jan. 1977.

[11] J. McQuillan, I. Richer, and E. Rosen, "The new routing algorithm for the ARPANET," *IEEE Trans. Commun.*, vol. 28, no. 5, pp. 711–719, May 1980.

[12] S. Chen and K. Nahrstedt, "Distributed quality-of-service routing in ad hoc networks," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 8, pp. 1488–1505, Aug. 1999.

[13] M. S. Corson and A. Ephremides, "A distributed routing algorithm for mobile wireless networks," *Wireless Netw.*, vol. 1, no. 1, pp. 61–81, Mar. 1995.

[14] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Boston, MA, USA: Kluwer Acad., 1996, pp. 153–181.

[15] V. D. Park and M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proc. IEEE INFOCOM*, vol. 3. Kobe, Japan, 1997, pp. 1405–1413.

[16] R. Sivakumar, P. Sinha, and V. Bharghavan, "CEDAR: A core-extraction distributed ad hoc routing algorithm," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 8, pp. 1454–1465, Aug. 1999.

[17] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," Internet Eng. Task Force, Fremont, CA, USA, RFC 1633, Jun. 1994. [Online]. Available: http://www.rfc-editor.org/rfc/rfc1633.txt

[18] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 4, pp. 14–26, Oct. 1992.

[19] M. A. El-Gendy, A. Bose, and K. G. Shin, "Evolution of the Internet QoS and support for soft real-time applications," *Proc. IEEE*, vol. 91, no. 7, pp. 1086–1104, Jul. 2003.

[20] J. Wroclawski, "The use of RSVP with IETF integrated services," Internet Eng. Task Force, Fremont, CA, USA, RFC 2210, Sep. 1997. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2210.txt

[21] X. Xiao and L. M. Ni, "Internet QoS: A big picture," *IEEE Netw.*, vol. 13, no. 2, pp. 8–18, Mar./Apr. 1999.

[22] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 7, pp. 1228–1234, Sep. 1996.

[23] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: An intellectual history of programmable networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[24] M. Jarschel, T. Zinner, T. Hoßfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for SDN," *IEEE Commun. Mag.*, vol. 52, no. 6, pp. 210–217, Jun. 2014.

[25] S. Agarwal, M. Kodialam, and T. V. Lakshman, "Traffic engineering in software defined networks," in *Proc. IEEE INFOCOM*, Turin, Italy, 2013, pp. 2211–2219.

[26] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "A roadmap for traffic engineering in SDN-OpenFlow networks," *Comput. Netw.*, vol. 71, pp. 1–30, Oct. 2014.

[27] I. Bueno, J. I. Aznar, E. Escalona, J. Ferrer, and J. A. Garcia-Espin, "An OpenNaaS based SDN framework for dynamic QoS control," in *Proc. IEEE SDN Future Netw. Services (SDN4FNS)*, Trento, Italy, 2013, pp. 1–7.

[28] V. Kotronis, X. Dimitropoulos, and B. Ager, "Outsourcing the routing control logic: Better Internet routing based on SDN principles," in *Proc. ACM Workshop Hot Topics Netw.*, Redmond, WA, USA, 2012, pp. 55–60.

[29] C. E. Rothenberg *et al.*, "Revisiting routing control platforms with the eyes and muscles of software-defined networking," in *Proc. ACM Workshop Hot Topics Softw. Defined Netw.*, Helsinki, Finland, 2012, pp. 13–18.

[30] H. Zhang and J. Yan, "Performance of SDN routing in comparison with legacy routing protocols," in *Proc. IEEE Int. Conf. Cyber Enabled Distrib. Comput. Knowl. Disc. (CyberC)*, Xi'an, China, 2015, pp. 491–494.

[31] *The League of Routing Algorithms*. Accessed: Sep. 15, 2017. [Online]. Available: http://www.lkn.ei.tum.de/lora

[32] M. Ramalho, "Intra-and inter-domain multicast routing protocols: A survey and taxonomy," *IEEE Commun. Surveys Tuts.*, vol. 3, no. 1, pp. 2–25, 1st Quart., 2000.

[33] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas, "A survey of application-layer multicast protocols," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 3, pp. 58–74, 3rd Quart., 2007.

[34] B. Wang and J. C. Hou, "Multicast routing and its QoS extension: Problems, algorithms, and protocols," *IEEE Netw.*, vol. 14, no. 1, pp. 22–36, Jan./Feb. 2000.

[35] C. Maihofer, "A survey of geocast routing protocols," *IEEE Commun. Surveys Tuts.*, vol. 6, no. 2, pp. 32–42, 2nd Quart., 2004.

[36] S. K. Singh, T. Das, and A. Jukan, "A survey on Internet multipath routing and provisioning," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2157–2175, 4th Quart., 2015.

[37] N. Chakchouk, "A survey on opportunistic routing in wireless communication networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2214–2241, 4th Quart., 2015.

[38] D. Chen and P. K. Varshney, "A survey of void handling techniques for geographic routing in wireless networks," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 1, pp. 50–67, 1st Quart., 2007.

[39] F. Mansourkiaie and M. H. Ahmed, "Cooperative routing in wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 604–626, 2nd Quart., 2015.

[40] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Comput. Netw.*, vol. 47, no. 4, pp. 445–487, Mar. 2005.

[41] J. Tang, G. Xue, and W. Zhang, "Interference-aware topology control and QoS routing in multi-channel wireless mesh networks," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Urbana, IL, USA, 2005, pp. 68–77.

[42] Y. Tsado *et al.*, "Improving the reliability of optimised link state routing in a smart grid neighbour area network based wireless mesh network using multiple metrics," *Energies*, vol. 10, no. 3, pp. 1–23, 2017.

[43] J. Agarkhed, P. Y. Dattatraya, and S. R. Patil, "Performance evaluation of QoS-aware routing protocols in wireless sensor networks," in *Proc. Int. Conf. Comput. Intell. Informat.*, 2017, pp. 559–569.

[44] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Netw.*, vol. 3, no. 3, pp. 325–349, May 2005.

[45] M. Z. Hasan, F. Al-Turjman, and H. Al-Rizzo, "Optimized multi-constrained quality-of-service multipath routing approach for multimedia sensor networks," *IEEE Sensors J.*, vol. 17, no. 7, pp. 2298–2309, Apr. 2017.

[46] J. Ben-Othman and B. Yahya, "Energy efficient and QoS based routing protocol for wireless sensor networks," *J. Parallel Distrib. Comput.*, vol. 70, no. 8, pp. 849–857, Aug. 2010.

[47] S. Ehsan and B. Hamdaoui, "A survey on energy-efficient routing techniques with QoS assurances for wireless multimedia sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 2, pp. 265–278, 2nd Quart., 2012.

[48] S. M. Ghoreyshi, A. Shahrabi, and T. Boutaleb, "Void-handling techniques for routing protocols in underwater sensor networks: Survey and challenges," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 800–827, 2nd Quart., 2017.

[49] D. Goyal and M. R. Tripathy, "Routing protocols in wireless sensor networks: A survey," in *Proc. IEEE Int. Conf. Adv. Comput. Commun. Technol.*, Rohtak, India, 2012, pp. 474–480.

[50] M. Z. Hasan, H. Al-Rizzo, and F. Al-Turjman, "A survey on multipath routing protocols for QoS assurances in real-time wireless multimedia sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1424–1456, 3rd Quart., 2017.

[51] A. A. Kumar S., K. Ovsthus, and L. M. Kristensen, "An industrial perspective on wireless sensor networks—A survey of requirements, protocols, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1391–1412, 3rd Quart., 2014.

[52] N. A. Pantazis, S. A. Nikolidakis, and D. D. Vergados, "Energy-efficient routing protocols in wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 551–591, 2nd Quart., 2013.

[53] M. Radi, B. Dezfouli, K. A. Bakar, and M. Lee, "Multipath routing in wireless sensor networks: Survey and research challenges," *Sensors*, vol. 12, no. 1, pp. 650–685, 2012.

[54] M. Saleem, G. A. Di Caro, and M. Farooq, "Swarm intelligence based routing protocol for wireless sensor networks: Survey and future directions," *Inf. Sci.*, vol. 181, no. 20, pp. 4597–4624, Oct. 2011.

[55] R. Sumathi and M. G. Srinivas, "A survey of QoS based routing protocols for wireless sensor networks," *J. Inf. Process. Syst.*, vol. 8, no. 4, pp. 589–602, 2012.

[56] C. Tunca, S. Isik, M. Y. Donmez, and C. Ersoy, "Distributed mobile sink routing for wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 877–897, 2nd Quart., 2014.

[57] R. A. Uthra and S. V. K. Raja, "QoS routing in wireless sensor networks—A survey," *ACM Comput. Surveys*, vol. 45, no. 1, pp. 1–12, Nov. 2012.

[58] Q. Wang and J. Jiang, "Comparative examination on architecture and protocol of industrial wireless sensor network standards," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2197–2219, 3rd Quart., 2016.

[59] T. Watteyne, A. Molinaro, M. G. Richichi, and M. Dohler, "From MANET to IETF ROLL standardization: A paradigm shift in WSN routing protocols," *IEEE Commun. Surveys Tuts.*, vol. 13, no. 4, pp. 688–707, 4th Quart., 2011.

[60] A. Boukerche *et al.*, "Routing protocols in ad hoc networks: A survey," *Comput. Netw.*, vol. 55, no. 13, pp. 3032–3080, Sep. 2011.

[61] F. Cadger, K. Curran, J. Santos, and S. Moffett, "A survey of geographical routing in wireless ad-hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 621–653, 2nd Quart., 2013.

[62] H. Cheng and J. Cao, "A design framework and taxonomy for hybrid routing protocols in mobile ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 3, pp. 62–73, 3rd Quart., 2008.

[63] J. Zuo, C. Dong, S. X. Ng, L.-L. Yang, and L. Hanzo, "Cross-layer aided energy-efficient routing design for ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1214–1238, 3rd Quart., 2015.

[64] L. Abusalah, A. Khokhar, and M. Guizani, "A survey of secure mobile ad hoc routing protocols," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 78–93, 4th Quart., 2008.

[65] P. G. Argyroudis and D. O'Mahony, "Secure routing for mobile ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 7, no. 3, pp. 2–21, 3rd Quart., 2005.

[66] T. R. Andel and A. Yasinsac, "Surveying security analysis techniques in MANET routing protocols," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 4, pp. 70–84, 4th Quart., 2007.

[67] L. Chen and W. B. Heinzelman, "A survey of routing protocols that support QoS in mobile ad hoc networks," *IEEE Netw.*, vol. 21, no. 6, pp. 30–38, Nov./Dec. 2007.

[68] L. Hanzo and R. Tafazolli, "A survey of QoS routing solutions for mobile ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 9, no. 2, pp. 50–70, 2nd Quart., 2007.

[69] L. Junhai, Y. Danxia, X. Liu, and F. Mingyu, "A survey of multicast routing protocols for mobile ad-hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 1, pp. 78–91, 1st Quart., 2009.

[70] D. N. Kanellopoulos, "QoS routing for multimedia communication over wireless mobile ad hoc networks: A survey," *Int. J. Multimedia Data Eng. Manag.*, vol. 8, no. 1, pp. 42–71, 2017.

[71] G. V. Kumar, Y. V. Reddyr, and M. Nagendra, "Current research work on routing protocols for MANET: A literature survey," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 3, pp. 706–713, 2010.

[72] C. Liu and J. Kaiser, "A survey of mobile ad hoc network routing protocols," Universität Ulm, Ulm, Germany, Tech. Rep., 2005. [Online]. Available: http://dx.doi.org/10.18725/OPARU-331

[73] G. A. Walikar and R. C. Biradar, "A survey on hybrid routing mechanisms in mobile ad hoc networks," *J. Netw. Comput. Appl.*, vol. 77, pp. 48–63, Jan. 2017.

[74] Z. Zhang, "Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges," *IEEE Commun. Surveys Tuts.*, vol. 8, no. 1, pp. 24–37, 1st Quart., 2006.

[75] S. Bitam, A. Mellouk, and S. Zeadally, "Bio-inspired routing algorithms survey for vehicular ad hoc networks," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 843–867, 2nd Quart., 2015.

[76] C. Cooper, D. Franklin, M. Ros, F. Safaei, and M. Abolhasan, "A comparative survey of VANET clustering techniques," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 1, pp. 657–681, 1st Quart., 2017.

[77] Y.-W. Lin, Y.-S. Chen, and S.-L. Lee, "Routing protocols in vehicular ad hoc networks: A survey and future perspectives," *J. Inf. Sci. Eng.*, vol. 26, no. 3, pp. 913–932, 2010.

[78] A. Mchergui, T. Moulahi, B. Alaya, and S. Nasri, "A survey and comparative study of QoS aware broadcasting techniques in VANET," *Telecommun. Syst.*, vol. 66, no. 2, pp. 253–281, Oct. 2017.

[79] M. Youssef, M. Ibrahim, M. Abdelatif, L. Chen, and A. V. Vasilakos, "Routing metrics of cognitive radio networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 92–109, 1st Quart., 2014.

[80] Y. Cao and Z. Sun, "Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 654–677, 2nd Quart., 2013.

[81] K. Wei, X. Liang, and K. Xu, "A survey of social-aware routing protocols in delay tolerant networks: Applications, taxonomy and design-related issues," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 556–578, 1st Quart., 2014.

[82] Y. Zhu, B. Xu, X. Shi, and Y. Wang, "A survey of social-based routing in delay tolerant networks: Positive and negative social effects," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 387–401, 1st Quart., 2013.

[83] S. Batabyal and P. Bhaumik, "Mobility models, traces and impact of mobility on opportunistic routing algorithms: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1679–1707, 3rd Quart., 2015.

[84] A. Z. M. Shahriar, M. Atiquzzaman, and W. Ivancic, "Route optimization in network mobility: Solutions, classification, comparison, and future research directions," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 1, pp. 24–38, 1st Quart., 2010.

[85] S. Azodolmolky *et al.*, "A survey on physical layer impairments aware routing and wavelength assignment algorithms in optical networks," *Comput. Netw.*, vol. 53, no. 7, pp. 926–944, May 2009.

[86] N. Charbonneau and V. M. Vokkarane, "A survey of advance reservation routing and wavelength assignment in wavelength-routed WDM networks," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1037–1064, 4th Quart., 2012.

[87] B. C. Chatterjee, N. Sarma, and E. Oki, "Routing and spectrum allocation in elastic optical networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1776–1800, 3rd Quart., 2015.

[88] P.-H. Ho, "State-of-the-art progress in developing survivable routing schemes in mesh WDM networks," *IEEE Commun. Surveys Tuts.*, vol. 6, no. 4, pp. 2–16, 4th Quart., 2004.

[89] A. G. Rahbar, "Review of dynamic impairment-aware routing and wavelength assignment techniques in all-optical wavelength-routed networks," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1065–1089, 4th Quart., 2012.

[90] C. V. Saradhi and S. Subramaniam, "Physical layer impairment aware routing (PLIAR) in WDM optical networks: Issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 4, pp. 109–130, 4th Quart., 2009.

[91] M. Hoefling, M. Menth, and M. Hartmann, "A survey of mapping systems for locator/identifier split Internet routing," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 1842–1858, 4th Quart., 2013.

[92] D. Nace and M. Pioro, "Max-min fairness and its applications to routing and load-balancing in communication networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 5–17, 4th Quart., 2008.

[93] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for Internet traffic engineering," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 36–56, 1st Quart., 2008.

[94] J. L. Martins and S. Duarte, "Routing algorithms for content-based publish/subscribe systems," *IEEE Commun. Surveys Tuts.*, vol. 12, no. 1, pp. 39–58, 1st Quart., 2010.

[95] F. Dabaghi, Z. Movahedi, and R. Langar, "A survey on green routing protocols using sleep-scheduling in wired networks," *J. Netw. Comput. Appl.*, vol. 77, pp. 106–122, Jan. 2017.

[96] X. Masip-Bruin *et al.*, "Research challenges in QoS routing," *Comput. Commun.*, vol. 29, no. 5, pp. 563–581, Mar. 2006.

[97] S. Upadhyaya and G. Dhingra, "Exploring issues for QoS based routing algorithms," *Int. J. Comput. Sci. Eng.*, vol. 2, no. 5, pp. 1792–1795, 2010.

[98] S. Chen and K. Nahrstedt, "An overview of quality of service routing for next-generation high-speed networks: Problems and solutions," *IEEE Netw.*, vol. 12, no. 6, pp. 64–79, Nov./Dec. 1998.

[99] M. Curado and E. Monteiro, "A survey of QoS routing algorithms," in *Proc. Int. Conf. Inf. Technol. (ICIT)*, 2004, pp. 1–4.

[100] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Commun. Mag.*, vol. 40, no. 12, pp. 50–55, Dec. 2002.

[101] J. L. Marzo, E. Calle, C. Scoglio, and T. Anjah, "QoS online routing and MPLS multilevel protection: A survey," *IEEE Commun. Mag.*, vol. 41, no. 10, pp. 126–132, Oct. 2003.

[102] P. Paul and S. V. Raghavan, "Survey of QoS routing," in *Proc. Int. Conf. Comput. Commun. (ICCC)*, Mumbai, India, Aug. 2002, pp. 50–75.

[103] O. Younis and S. Fahmy, "Constraint-based routing in the Internet: Basic principles and recent research," *IEEE Commun. Surveys Tuts.*, vol. 5, no. 1, pp. 2–13, 3rd Quart., 2003.

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

[105] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster, "Analysis of topology aggregation techniques for QoS routing," *ACM Comput. Surveys*, vol. 39, no. 3, pp. 1–31, 2007.

[106] X. Zhang and C. Phillips, "A survey on selective routing topology inference through active probing," *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1129–1141, 4th Quart., 2012.

[107] D. Adami, L. Donatini, S. Giordano, and M. Pagano, "A network control application enabling software-defined quality of service," in *Proc. IEEE Int. Conf. Commun. (ICC)*, London, U.K., 2015, pp. 6074–6079.

[108] M. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "PolicyCop: An autonomic QoS policy enforcement framework for software defined networks," in *Proc. IEEE SDN Future Netw. Services*, Trento, Italy, 2013, pp. 1–7.

[109] B. Briscoe *et al.*, "Reducing Internet latency: A survey of techniques and their merits," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 2149–2196, 3rd Quart., 2016.

[110] C. Caba, A. Mimidis, and J. Soler, "Model-driven policy framework for data centers (short paper)," in *Proc. IEEE Int. Conf. Cloud Netw. (Cloudnet)*, Pisa, Italy, 2016, pp. 126–129.

[111] Á. L. V. Caraguay, J. A. P. Fernández, and L. J. G. Villalba, "Framework for optimized multimedia routing over software defined networks," *Comput. Netw.*, vol. 92, pp. 369–379, Dec. 2015.

[112] A. Ishimori, F. Farias, E. Cerqueira, and A. Abelém, "Control of multiple packet schedulers for improving QoS on OpenFlow/SDN networking," in *Proc. IEEE Eur. Workshop Softw. Defined Netw.*, Berlin, Germany, 2013, pp. 81–86.

[113] A. Mendiola *et al.*, "An architecture for dynamic QoS management at layer 2 for DOCSIS access networks using OpenFlow," *Comput. Netw.*, vol. 94, pp. 112–128, Jan. 2016.

[114] K. Nahrstedt, D. Xu, D. Wichadakul, and B. Li, "QoS-aware middleware for ubiquitous and heterogeneous environments," *IEEE Commun. Mag.*, vol. 39, no. 11, pp. 140–148, Nov. 2001.

[115] M. Reisslein, K. W. Ross, and S. Rajagopal, "A framework for guaranteeing statistical QoS," *IEEE/ACM Trans. Netw.*, vol. 10, no. 1, pp. 27–42, Feb. 2002.

[116] P. Sharma *et al.*, "Enhancing network management frameworks with SDN-like control," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manag. (IM)*, Ghent, Belgium, 2013, pp. 688–691.

[117] S. Sharma *et al.*, "Implementing quality of service for the software defined networking enabled future Internet," in *Proc. IEEE Eur. Workshop Softw. Defined Netw.*, London, U.K., Sep. 2014, pp. 49–54.

[118] E. W. Knightly and N. B. Shroff, "Admission control for statistical QoS: Theory and practice," *IEEE Netw.*, vol. 13, no. 2, pp. 20–29, Mar./Apr. 1999.

[119] J. Liebeherr, D. E. Wrege, and D. Ferrari, "Exact admission control for networks with a bounded delay service," *IEEE/ACM Trans. Netw.*, vol. 4, no. 6, pp. 885–901, Dec. 1996.

[120] M. Reisslein and K. W. Ross, "Call admission for prerecorded sources with packet loss," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 6, pp. 1167–1180, Aug. 1997.

[121] Z.-L. Zhang, Z. Liu, J. Kurose, and D. Towsley, "Call admission control schemes under generalized processor sharing scheduling," *Telecommun. Syst.*, vol. 7, nos. 1–3, pp. 125–152, Jun. 1997.

[122] L. Sha *et al.*, "Real time scheduling theory: A historical perspective," *Real Time Syst.*, vol. 28, nos. 2–3, pp. 101–155, Nov. 2004.

[123] J. W. Guck and W. Kellerer, "Achieving end-to-end real-time quality of service with software defined networking," in *Proc. IEEE Int. Conf. Cloud Netw. (CloudNet)*, Luxembourg, Luxembourg, 2014, pp. 70–76.

[124] R. Kumar *et al.*, "Dependable end-to-end delay constraints for real-time systems using SDNs," *arXiv preprint arXiv:1703.01641*, 2017.

[125] S. M. Laursen, P. Pop, and W. Steiner, "Routing optimization of AVB streams in TSN networks," *ACM SIGBED Rev.*, vol. 13, no. 4, pp. 43–48, Sep. 2016.

[126] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive software-defined network (TSSDN) for real-time applications," in *Proc. ACM Int. Conf. Real Time Netw. Syst.*, Brest, France, 2016, pp. 193–202.

[127] P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks," *IET Cyber Phys. Syst. Theory Appl.*, vol. 1, no. 1, pp. 86–94, Dec. 2016.

[128] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. ACM Int. Conf. Real Time Netw. Syst.*, Brest, France, 2016, pp. 183–192.

[129] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proc. ACM Int. Conf. Real Time Netw. Syst.*, Brest, France, 2016, pp. 203–212.

[130] Q. Duan, "Network-as-a-service in software-defined networks for end-to-end QoS provisioning," in *Proc. IEEE Wireless Opt. Commun. Conf. (WOCC)*, Newark, NJ, USA, 2014, pp. 1–5.

[131] S. Tomovic, N. Prasad, and I. Radusinovic, "SDN control framework for QoS provisioning," in *Proc. IEEE Telecommun. Forum Telfor (TELFOR)*, Belgrade, Serbia, 2014, pp. 111–114.

[132] A. L. King, S. Chen, and I. Lee, "The middleware assurance substrate: Enabling strong real-time guarantees in open systems with OpenFlow," in *Proc. IEEE Int. Symp. Object/Compon./Service Orient. Real Time Distrib. Comput. (ISORC)*, Reno, NV, USA, 2014, pp. 133–140.

[133] J. W. Guck, M. Reisslein, and W. Kellerer, "Function split between delay-constrained routing and resource allocation for centrally managed QoS in industrial networks," *IEEE Trans. Ind. Informat.*, vol. 12, no. 6, pp. 2050–2061, Dec. 2016.

[134] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet* (LNCS 2050). Berlin, Germany: Springer, 2001.

[135] J. Schmitt, P. Hurley, M. Hollick, and R. Steinmetz, "Per-flow guarantees under class-based priority queueing," in *Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM)*, vol. 7. San Francisco, CA, USA, 2003, pp. 4169–4174.

[136] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic*, IEEE Standard 802.1Qbv-2015, pp. 1–57, Mar. 2016.

[137] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 26: Frame Preemption*, IEEE Standard 802.1Qbu-2016, pp. 1–52, Aug. 2016.

[138] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 24: Path Control and Reservation*, IEEE Standard 802.1Qca-2015, pp. 1–120, Mar. 2016.

[139] C. Gunther, "Communications standards news: What's new in the world of IEEE 802.1 TSN," *IEEE Commun. Mag.*, vol. 54, no. 9, pp. 12–15, Sep. 2016.

[140] P. Heise, F. Geyer, and R. Obermaisser, "TSimNet: An industrial time sensitive networking simulation framework based on OMNeT++," in *Proc. IEEE/IFIP Int. Conf. New Tech. Mobility Security (NTMS)*, Larnaca, Cyprus, 2016, pp. 1–5.

[141] Z. Zhou, Y. Yan, S. Ruepp, and M. Berger, "Analysis and implementation of packet preemption for time sensitive networks," in *Proc. IEEE Int. Conf. High Perform. Switch. Routing (HPSR)*, Campinas, Brazil, 2017, pp. 1–6.

[142] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic networking architecture, draft-ietf-detnet-architecture-02," IETF Internet-Draft, pp. 1–43, Jun. 2017.

[143] J.-D. Decotignie, "Ethernet-based real-time and industrial communications," *Proc. IEEE*, vol. 93, no. 6, pp. 1102–1117, Jun. 2005.

[144] B. Galloway and G. P. Hancke, "Introduction to industrial control networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 860–880, 2nd Quart., 2013.

[145] P. Gaj, J. Jasperneite, and M. Felser, "Computer communication within industrial distributed environment—A survey," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 182–189, Feb. 2013.

[146] J.-Q. Li *et al.*, "Industrial Internet: A survey on the enabling technologies, applications, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1504–1526, 3rd Quart., 2017.

[147] L. Seno, F. Tramarin, and S. Vitturi, "Performance of industrial communication systems: Real application contexts," *IEEE Ind. Electron. Mag.*, vol. 6, no. 2, pp. 27–37, Jun. 2012.

[148] D. Thiele and R. Ernst, "Formal analysis based evaluation of software defined networking for time-sensitive Ethernet," in *Proc. IEEE Design Autom. Test Europe Conf. Exhibit. (DATE)*, Dresden, Germany, 2016, pp. 31–36.

[149] A. A. Khan, M. H. Rehmani, and M. Reisslein, "Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 860–898, 1st Quart., 2016.

[150] Y. Yan, Y. Qian, H. Sharif, and D. Tipper, "A survey on smart grid communication infrastructures: Motivations, requirements and challenges," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 5–20, 1st Quart., 2013.

[151] P. G. Benardos and G. C. Vosniakos, "Internet of Things and industrial applications for precision machining," *Solid State Phenomena*, vol. 261, pp. 440–447, Aug. 2017.

[152] J. Lin *et al.*, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, to be published.

[153] I. Yaqoob *et al.*, "Internet of Things architecture: Recent advances, taxonomy, requirements, and open challenges," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 10–16, Jun. 2017.

[154] M. R. Palattella *et al.*, "Standardized protocol stack for the Internet of (important) Things," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1389–1406, 3rd Quart., 2013.

[155] C. Bachhuber, E. Steinbach, M. Freundl, and M. Reisslein, "On the minimization of glass-to-glass and glass-to-algorithm delay in video communication," *IEEE Trans. Multimedia*, to be published.

[156] J. Luo, J. Jin, and F. Shan, "Standardization of low-latency TCP with explicit congestion notification: A survey," *IEEE Internet Comput.*, vol. 21, no. 1, pp. 48–55, Jan./Feb. 2017.

[157] T. Skeie, S. Johannessen, and O. Holmeide, "Timeliness of real-time IP communication in switched industrial Ethernet networks," *IEEE Trans. Ind. Informat.*, vol. 2, no. 1, pp. 25–39, Feb. 2006.

[158] R. Alvizu *et al.*, "Comprehensive survey on T-SDN: Software-defined networking for transport networks," *IEEE Commun. Surveys Tuts.*, to be published.

[159] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 655–685, 1st Quart., 2016.

[160] F. Hu, Q. Hao, and K. Bao, "A survey on software-defined network and OpenFlow: From concept to implementation," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2181–2206, 4th Quart., 2014.

[161] T. Kohler, F. Dürr, and K. Rothermel, "ZeroSDN: A highly flexible and modular architecture for full-range network control distribution," in *Proc. ACM/IEEE Symp. Architect. Netw. Commun. Syst. (ANCS)*, Beijing, China, May 2017, pp. 25–37.

[162] D. Kreutz *et al.*, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[163] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.

[164] A. S. Thyagaturu, A. Mercian, M. P. McGarry, M. Reisslein, and W. Kellerer, "Software defined optical networks (SDONs): A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2738–2786, 4th Quart., 2016.

[165] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 27–51, 1st Quart., 2014.

[166] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 493–512, 1st Quart., 2014.

[167] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, vol. 16. Chichester, U.K.: Wiley, 2001.

[168] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Rel. Eng. Syst. Safety*, vol. 91, no. 9, pp. 992–1007, Sep. 2006.

[169] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Struct. Multidiscipl. Optim.*, vol. 26, no. 6, pp. 369–395, Apr. 2004.

[170] H.-S. Yang, M. Maier, M. Reisslein, and W. M. Carlyle, "A genetic algorithm-based methodology for optimizing multiservice convergence in a metro WDM network," *IEEE/OSA J. Lightw. Technol.*, vol. 21, no. 5, pp. 1114–1133, May 2003.

[171] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.

[172] A. Schrijver, "On the history of the shortest path problem," *Documenta Math.*, vol. 2012, pp. 155–167, 2012.

[173] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[174] R. Bellman, "On a routing problem," *Quart. Appl. Math.*, vol. 16, no. 1, pp. 87–90, Apr. 1958.

[175] D. Cavendish and M. Gerla, "Internet QoS routing using the Bellman–Ford algorithm," in *High Performance Networking*. New York, NY, USA: Springer, 1998, pp. 627–646.

[176] L. R. Ford, Jr., *Network Flow Theory*, document 422842, DTIC, Fort Belvoir, VA, USA, 1956.

[177] E. F. Moore, *The Shortest Path Through a Maze*. New York, NY, USA: Bell Telephone Syst., 1959.

[178] A. Shimbel, "Structure in communication nets," in *Proc. Symp. Inf. Netw.*, New York, NY, USA, 1954, pp. 199–203.

[179] J. Y. Yen, "An algorithm for finding shortest routes from all source nodes to a given destination in general networks," *Quart. Appl. Math.*, vol. 27, no. 4, pp. 526–530, Jan. 1970.

[180] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[181] M. J. Bannister and D. Eppstein, "Randomized speedup of the Bellman–Ford algorithm," in *Proc. SIAM Meeting Anal. Algorithmics Combinatorics*, Kyoto, Japan, 2012, pp. 41–47.

[182] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, "Shortest paths algorithms: Theory and experimental evaluation," *Math. Program.*, vol. 73, no. 2, pp. 129–174, May 1996.

[183] L. Fu, D. Sun, and L. R. Rilett, "Heuristic shortest path algorithms for transportation applications: State of the art," *Comput. Oper. Res.*, vol. 33, no. 11, pp. 3324–3343, Nov. 2006.

[184] E. Chow, "A graph search heuristic for shortest distance paths," Lawrence Livermore Nat. Lab., Livermore, CA, USA, Tech. Rep. UCRL-JRNL-202894, 2005.

[185] J. Y. Yen, "Finding the *k* shortest loopless paths in a network," *Manag. Sci.*, vol. 17, no. 11, pp. 712–716, Jul. 1971.

[186] E. I. Chong, S. Maddila, and S. Morley, "On finding single-source single-destination *k* shortest paths," *J. Comput. Inf. Special Issue ICCI*, vol. 95, pp. 40–47, Jul. 1995.

[187] C. C. Skiscim and B. L. Golden, "Solving *k*-shortest and constrained shortest path problems efficiently," *Ann. Oper. Res.*, vol. 20, nos. 1–4, pp. 249–282, Aug. 1989.

[188] D. Eppstein, "Finding the *k* shortest paths," *SIAM J. Comput.*, vol. 28, no. 2, pp. 652–673, 1998.

[189] V. M. Jiménez and A. Marzal, "A lazy version of Eppstein's *k* shortest paths algorithm," in *Proc. Int. Workshop Exp. Efficient Algorithms*, Ascona, Switzerland, 2003, pp. 179–191.

[190] H. Aljazzar and S. Leue, "K*: A directed on-the-fly algorithm for finding the *k* shortest paths," Dept. Comput. Inf. Sci., Univ. at Konstanz, Konstanz, Germany, Tech. Rep. soft-08-03, 2008.

[191] H. C. Joksch, "The shortest route problem with constraints," *J. Math. Anal. Appl.*, vol. 14, no. 2, pp. 191–197, May 1966.

[192] Y. P. Aneja, V. Aggarwal, and K. P. K. Nair, "Shortest chain subject to side constraints," *Networks*, vol. 13, no. 2, pp. 295–302, 1983.

[193] W. C. Lee, M. G. Hluchyi, and P. A. Humblet, "Routing subject to quality of service constraints in integrated communication networks," *IEEE Netw.*, vol. 9, no. 4, pp. 46–55, Jul./Aug. 1995.

[194] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," Int. Comput. Sci. Inst. Berkeley, Berkeley, CA, USA, Tech. Rep. TR-94-024, 1994.

[195] G. Liu and K. G. Ramakrishnan, "A*Prune: An algorithm for finding *k* shortest paths subject to multiple constraints," in *Proc. IEEE INFOCOM*, vol. 2. Anchorage, AK, USA, 2001, pp. 743–749.

[196] Z. Jia and P. Varaiya, "Heuristic methods for delay-constrained least-cost routing problem using *k*-shortest-path algorithms," in *Proc. IEEE INFOCOM*, 2001, pp. 1–9.

[197] G. Cheng and N. Ansari, "A new heuristics for finding the delay constrained least cost path," in *Proc. IEEE Glob. Telecommun. Conf. (GLOBECOM)*, vol. 7. San Francisco, CA, USA, 2003, pp. 3711–3715.

[198] D. P. Bertsekas, *Nonlinear Programming*. Belmont, MA, USA: Athena Sci., 1999.

[199] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[200] Y. P. Aneja and K. P. K. Nair, "The constrained shortest path problem," *Naval Res. Logistics Quart.*, vol. 25, no. 3, pp. 549–555, Sep. 1978.

[201] G. Y. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, no. 4, pp. 293–309, 1980.

[202] D. Blokh and G. Gutin, "An approximate algorithm for combinatorial optimization problems with two parameters," *Aust. J. Combinatorics*, vol. 14, pp. 157–164, Sep. 1996.

[203] A. Jüttner, B. Szviatovski, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," in *Proc. IEEE INFOCOM*, vol. 2. Anchorage, AK, USA, 2001, pp. 859–868.

[204] L. Wolsey and G. Nemhauser, *Integer and Combinatorial Optimization* (Wiley Series in Discrete Mathematics and Optimization). Hoboken, NJ, USA: Wiley, 1999.

[205] L. Santos, J. Coutinho-Rodrigues, and J. R. Current, "An improved solution algorithm for the constrained shortest path problem," *Transp. Res. B Methodol.*, vol. 41, no. 7, pp. 756–771, Aug. 2007.

[206] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," in *Proc. IEEE INFOCOM*, vol. 2. Anchorage, AK, USA, 2001, pp. 834–843.

[207] G. Feng, K. Makki, N. Pissinou, and C. Douligeris, "Heuristic and exact algorithms for QoS routing with multiple constraints," *IEICE Trans. Commun.*, vol. E85-B, no. 12, pp. 2838–2850, Dec. 2002.

[208] G. Feng, C. Douligeris, K. Makki, and N. Pissinou, "Performance evaluation of delay-constrained least-cost QoS routing algorithms based on linear and nonlinear Lagrange relaxation," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 4. New York, NY, USA, 2002, pp. 2273–2278.

[209] L. Guo and I. Matta, "Search space reduction in QoS routing," *Comput. Netw.*, vol. 41, no. 1, pp. 73–88, Jan. 2003.

[210] H. Agrawal, M. Grah, and M. Gregory, "Optimization of QoS routing," in *Proc. IEEE/ACIS Int. Conf. Comput. Inf. Sci. (ICIS)*, Melbourne, VIC, Australia, 2007, pp. 598–603.

[211] C. C. Ribeiro and M. Minoux, "A heuristic approach to hard constrained shortest path problems," *Discr. Appl. Math.*, vol. 10, no. 2, pp. 125–137, Feb. 1985.

[212] H. F. Salama, D. S. Reeves, and Y. Viniotis, "A distributed algorithm for delay-constrained unicast routing," in *Proc. IEEE INFOCOM*, vol. 1. Kobe, Japan, 1997, pp. 84–91.

[213] D. S. Reeves and H. F. Salama, "A distributed algorithm for delay-constrained unicast routing," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 239–250, Apr. 2000.

[214] Q. Sun and H. Langendörfer, "A new distributed routing algorithm for supporting delay-sensitive applications," *Comput. Commun.*, vol. 21, no. 6, pp. 572–578, May 1998.

[215] K. Ishida, K. Amano, and N. Kannari, "A delay-constrained least-cost path routing protocol and the synthesis method," in *Proc. IEEE Int. Conf. Real Time Comput. Syst. Appl.*, Hiroshima, Japan, 1998, pp. 58–65.

[216] R. Sriram, G. Manimaran, and C. S. R. Murthy, "Preferred link based delay-constrained least-cost routing in wide area networks," *Comput. Commun.*, vol. 21, no. 18, pp. 1655–1669, Dec. 1998.

[217] W. Liu, W. Lou, and Y. Fang, “An efficient quality of service routing algorithm for delay-sensitive applications,” *Comput. Netw.*, vol. 47, no. 1, pp. 87–104, Jan. 2005.

[218] A. Warburton, “Approximation of Pareto optima in multiple-objective, shortest-path problems,” *Oper. Res.*, vol. 35, no. 1, pp. 70–79, Jan./Feb. 1987.

[219] R. Hassin, “Approximation schemes for the restricted shortest path problem,” *Math. Oper. Res.*, vol. 17, no. 1, pp. 36–42, Feb. 1992.

[220] D. Raz and Y. Shavitt, “Optimal partition of QoS requirements with discrete cost functions,” in *Proc. IEEE INFOCOM*, vol. 2. Tel Aviv, Israel, 2000, pp. 613–622.

[221] D. H. Lorenz and D. Raz, “A simple efficient approximation scheme for the restricted shortest path problem,” *Oper. Res. Lett.*, vol. 28, no. 5, pp. 213–219, Jun. 2001.

[222] F. Ergun, R. Sinha, and L. Zhang, “An improved FPTAS for restricted shortest path,” *Inf. Process. Lett.*, vol. 83, no. 5, pp. 287–291, Sep. 2002.

[223] D. H. Lorenz, A. Orda, D. Raz, and Y. Shavitt, “Efficient QoS partition and routing of unicast and multicast,” *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1336–1347, Dec. 2006.

[224] G. Xue, W. Zhang, J. Tang, and K. Thulasiraman, “Polynomial time approximation algorithms for multi-constrained QoS routing,” *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 656–669, Jun. 2008.

[225] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, “Improving QoS routing performance under inaccurate link state information,” in *Proc. Int. Teletraffic Congr.*, 1999, pp. 7–11.

[226] S. Chen and K. Nahrstedt, “Distributed QoS routing with imprecise state information,” in *Proc. IEEE Int. Conf. Comput. Commun. Netw.*, Lafayette, LA, USA, 1998, pp. 614–621.

[227] R. A. Guérin and A. Orda, “QoS routing in networks with inaccurate information: Theory and algorithms,” *IEEE/ACM Trans. Netw.*, vol. 7, no. 3, pp. 350–364, Jun. 1999.

[228] D. H. Lorenz and A. Orda, “QoS routing in networks with uncertain parameters,” *IEEE/ACM Trans. Netw.*, vol. 6, no. 6, pp. 768–778, Dec. 1998.

[229] L. Xiao, J. Wang, and M. Nahrstedt, “The enhanced ticket-based routing algorithm,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 4. New York, NY, USA, 2002, pp. 2222–2226.

[230] K. G. Shin and C.-C. Chou, “A distributed route-selection scheme for establishing real-time channels,” in *High Performance Networking*. Dordecht. The Netherlands: Springer, 1995, pp. 319–330.

[231] S. Chen and K. Nahrstedt, “Distributed quality-of-service routing in high-speed networks based on selective probing,” in *Proc. IEEE Conf. Local Comput. Netw. (LCN)*, Lowell, MA, USA, 1998, pp. 80–89.

[232] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis, “Efficient computation of delay-sensitive routes from one source to all destinations,” in *Proc. IEEE INFOCOM*, vol. 2. Anchorage, AK, USA, 2001, pp. 854–858.

[233] F. Xiang, L. Junzhou, W. Jieyi, and G. Guanqun, “QoS routing based on genetic algorithm,” *Comput. Commun.*, vol. 22, nos. 15–16, pp. 1392–1399, Sep. 1999.

[234] W. Zhengying, S. Bingxin, and Z. Erdun, “Bandwidth-delay-constrained least-cost multicast routing based on heuristic genetic algorithm,” *Comput. Commun.*, vol. 24, nos. 7–8, pp. 685–692, Apr. 2001.

[235] D. Karaboga and B. Basturk, “Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems,” in *Proc. Int. Fuzzy Syst. Assoc. World Congr.*, 2007, pp. 789–798.

[236] C. Pornavalai, G. Chakraborty, and N. Shiratori, “QoS based routing algorithm in integrated services packet networks,” *J. High Speed Netw.*, vol. 7, no. 2, pp. 99–112, 1998.

[237] C. Pornavalai, G. Chakraborty, and N. Shiratori, “Routing with multiple QoS requirements for supporting multimedia applications,” *Telecommun. Syst.*, vol. 9, nos. 3–4, pp. 357–373, Sep. 1998.

[238] B. M. Waxman, “Routing of multipoint connections,” *IEEE J. Sel. Areas Commun.*, vol. SAC-6, no. 9, pp. 1617–1622, Dec. 1988.

[239] M. Caria, A. Jukan, and M. Hoffmann, “SDN partitioning: A centralized control plane for distributed routing protocols,” *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 381–393, Sep. 2016.

[240] S. Krile, M. Rakús, and F. Schindler, “Centralized routing algorithm based on flow permutations,” in *Proc. IEEE Int. Conf. Telecommun. Signal Process. (TSP)*, Vienna, Austria, 2016, pp. 68–73.

[241] S. H. Park *et al.*, “RAON: Recursive abstraction of OpenFlow networks,” in *Proc. IEEE Eur. Workshop Softw. Defined Netw. (EWSDN)*, London, U.K., 2014, pp. 115–116.

[242] S. Vrijders *et al.*, “Reducing the complexity of virtual machine networking,” *IEEE Commun. Mag.*, vol. 54, no. 4, pp. 152–158, Apr. 2016.

Jochen W. Guck received the Dipl.-Ing. degree in Ingenieurinformatik from the University of Applied Sciences Wuerzburg-Schweinfurt, Schweinfurt, Germany, in 2009 and the M.Sc. degree in electrical engineering from the Technical University of Munich, Munich, Germany, in 2011. In 2012, he joined the Chair of Communication Networks with the Technical University of Munich as a member of the research and teaching staff. His research interests include real-time communication, industrial communication, software-defined networking, and routing algorithms.

Amaury Van Bemten was born in Liège, Belgium, in 1993. He received the B.Sc. degree in engineering and the M.Sc. degree in computer science and engineering from the University of Liège, Belgium, in 2013 and 2015, respectively. He is currently pursuing the Ph.D. degree with the Technical University of Munich, where he joined the Chair of Communication Networks in 2015 and as a member of the research and teaching staff. His current research focuses on routing algorithms and the application of software-defined networking for resilient real-time communications in industrial environments.

Martin Reisslein (S'96–A'97–M'98–SM'03–F'14) received the Ph.D. degree in systems engineering from the University of Pennsylvania in 1998. He is a Professor with the School of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON EDUCATION, and IEEE ACCESS as well as *Computer Networks* and *Optical Switching and Networking*. He is an Associate Editor-in-Chief of the IEEE COMMUNICATIONS SURVEYS & TUTORIALS and chairs the steering committee of the IEEE TRANSACTIONS ON MULTIMEDIA.

Wolfgang Kellerer (M'96–SM'11) received the Dr.-Ing. (Ph.D.) degree and the Dipl.-Ing. degree from the Technical University of Munich, Munich, Germany, in 1995 and 2002, respectively, where he is a Full Professor, heading the Chair of Communication Networks with the Department of Electrical and Computer Engineering. He was with NTT DOCOMO's European Research Laboratories for over ten years. His research resulted in over 200 publications and 29 granted patents in the areas of mobile networking and service platforms. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and on the Editorial Board of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS. He is a member of ACM and the VDE ITG.