Bandwidth-Delay Constrained Path Selection Under Inaccurate State Information

Turgay Korkmaz and Marwan Krunz

Abstract—**One of the key issues in any quality-of-service (QoS) routing framework is how to compute a path that satisfies given QoS constraints. In this paper, we focus on the path computation problem subject to the bandwidth and delay constraints. This problem can be easily solved if the *exact* state information is available to the node performing the path computation function. In practice, however, nodes have only imprecise knowledge of the network state. The reliance on outdated information (and treating this information as exact) can significantly degrade the effectiveness of the path selection algorithm. To address this problem, we adopt a *probabilistic approach* in which the state parameters (available bandwidth and delay) are characterized by random variables. The goal is then to find the *most-probable bandwidth-delay-constrained path* (MP-BDCP). We provide efficient solutions for the MP-BDCP problem by decomposing it into two subproblems: the most-probable delay-constrained path (MP-DCP) problem and the most-probable bandwidth-constrained path (MP-BCP) problem. MP-DCP by itself is known to be NP-hard, necessitating the use of approximate solutions. By employing the central limit theorem and Lagrange relaxation techniques, we provide two complementary solutions for MP-DCP. These solutions are found to be highly efficient, requiring on average a few iterations of Dijkstra's shortest path algorithm. As for MP-BCP, it can be easily transformed into a variant of the shortest path problem. Our solutions for MP-DCP and MP-BCP are then combined to address the MP-BDCP problem by obtaining a set of *near-nondominated* paths. Decision makers can then select one or more of these paths based on a specific utility function. Extensive simulations are used to demonstrate the efficiency of the proposed algorithmic solutions and, more generally, to contrast the probabilistic path selection approach with the standard threshold-based triggered approach.**

Keywords—**QoS routing, Stochastic shortest path, Lagrange relaxation, Multi-objective optimization.**

I. Introduction

Networked multimedia applications are becoming increasingly popular and are making a good case for the deployment of QoS-based network architectures (e.g., DiffServ, MPLS). One of the key issues in such architectures is how to determine "appropriate" paths that fulfill the QoS requirements of the transported traffic. In general, path selection problems subject to QoS constraints are computationally hard, and can only be dealt with using heuristics and approximate solutions. Examples of such solutions are reviewed in [1], [2]. With few exceptions, previous QoS routing solutions have been developed under the assumption that the *exact* state of the network is known to nodes performing the route computation. In practice, however, network state is not known for certain due to the followings reasons [3], [4]. First, current link-state routing protocols such as OSPF [5] flood link values periodically. To limit the overhead of flooding, long update intervals are used. For example, in OSPF a link update is sent every 30 minutes. Periodic flooding is sufficient for "static" link parameters (e.g., link connectivity), but cannot provide the desired accuracy in the case of highly dynamic link parameters, such as the available link bandwidth, where several changes in the parameter value are likely to occur within two periodic OSPF updates. The reliance on outdated information and treating this information as exact can significantly degrade the effectiveness of the path selection algorithm. Of course, threshold-based (triggered) flooding can be used to partially address this problem [6], but the associated overhead can be excessive[1]. A second source of inaccuracy is attributed to state aggregation. Most link-state routing protocols (e.g., OSPF [5], PNNI [7]) are hierarchical, whereby the state of a group of nodes (an OSPF area or a PNNI peer group) is summarized (aggregated) before being disseminated to other nodes [8]. While state aggregation is essential to ensuring the scalability of any QoS-aware routing protocol, it comes at the expense of *perturbing* the true state of the network. Finally, state inaccuracy can also be attributed to the sampling errors in computing the values of link parameters and to the latency associated with disseminating these values throughout the network (the so-called protocol "convergence time").

In this paper, we consider the path selection problem under both delay and bandwidth constraints. To account for uncertainties in the link-state parameters, we follow a *probabilistic* approach in which these parameters are modelled as random variables (rvs). Very mild assumptions are made on the probability density functions (pdfs) of these rvs. In fact, our solutions do not require the computation and dissemination of pdfs. Instead, a node (e.g., router) is only required to compute and disseminate the *mean and variance* (or two related parameters) of the bandwidth and delay values for the outgoing links. These moments can be computed simply as follows. Each node maintains a moving average and corresponding variance for both the available bandwidth and delay of each outgoing link. The parameters for the bandwidth are updated whenever there is a change in the available bandwidth (e.g., flow is added or terminated), while the ones for the delay are updated whenever a packet leaves the router. Once the local mean and variance are computed for each outgoing link, they can be disseminated using QoS-enhanced versions of OSPF, such as the one described in [9].

One important question here is when and how to advertise the mean and variance values. A triggered-based approach similar to the one in [6] can be used for this purpose, but applied to the mean and variance values rather than the actual instantaneous values of the bandwidth and delay. So, for example, if the *variance* of the packet delay changes by x% from its most recent advertised value, where x is a control threshold, this could trigger an update of the variance. The advertisement overhead depends on the variability of the mean and variance parameters, which in turn is a function of the fluctuations of the instanta-

T. Korkmaz is with the Dept. of Computer Science, University of Texas at San Antonio (e-mail: korkmaz@cs.utsa.edu).

M. Krunz is with Dept. of Electrical and Computer Engineering, University of Arizona (e-mail: krunz@ece.arizona.edu).

[1]Comparisons between threshold-based updating and the proposed probabilistic path selection approach are given in Section V.

neous delay and bandwidth values, the time window over which these moments are computed, and the triggering thresholds. In any case, by virtue of being moments of a time series, the (windowed) mean and variance values will exhibit less fluctuations than the actual instantaneous values. So a considerable reduction in the advertisement overhead is to be expected when the advertisement is done only after significant changes in the mean and variance values take place. Note that as the length of the averaging window increases, a single measurement point will have less impact on the overall mean and variance. The same can be said about the values of the triggering thresholds.

Having motivated the feasibility of capturing and disseminating probabilistic state information, we now address how such information can be used to compute paths subject to bandwidth and delay constraints. The problem at hand is algorithmically stated as follows:

Definition 1 *Most-Probable Bandwidth-Delay Constrained Path (MP-BDCP) Problem*: Consider a network $G = (V, E)$, where V is the set of nodes and E is the set of links. Each link $(i, j) \in E$ is associated with an available bandwidth parameter $b(i, j)$ and a delay parameter $d(i, j)$. We assume that the $b(i, j)$'s and $d(i, j)$'s are independent rvs. For any path p from the source node s to the destination node t, let $b(p) \overset{\text{def}}{=} \min\{b(i, j) \mid (i, j) \in p\}$ and $d(p) \overset{\text{def}}{=} \sum_{(i,j)\in p} d(i, j)$. Given a bandwidth constraint B and a delay constraint D, the problem is to find a path that is most likely to satisfy both constraints. Specifically, the problem is to find a path $r*$ such that for any other path p from s to t,

$$\pi_B(r*) \geq \pi_B(p), \quad \text{and} \tag{1}$$

$$\pi_D(r*) \geq \pi_D(p), \tag{2}$$

where $\pi_B(p) \overset{\text{def}}{=} \Pr[b(p) \geq B]$ and $\pi_D(p) \overset{\text{def}}{=} \Pr[d(p) \leq D]$.

If the $b(i, j)$'s and $d(i, j)$'s are constants, the MP-BDCP problem reduces to the familiar bandwidth-delay constrained path problem, which can be easily solved in two steps [10]: (i) prune every link (i, j) for which $b(i, j) < B$, and (ii) find the shortest path with respect to (w.r.t.) the delay parameter in the pruned graph. However, MP-BDCP is, in general, a hard problem; in fact, even if one only considers the delay constraint, the problem is still NP-hard [11], necessitating the reliance on approximate but computationally feasible solutions. Using the central limit theorem and Lagrange relaxation techniques, we first provide two complementary solutions for the delay case. These solutions are found to be highly efficient, requiring, on average, a few iterations of Dijkstra's shortest path algorithm. The bandwidth case is rather simple, and is dealt with by transforming it into a variant of the shortest path problem. The solutions for the MP-BCP and MP-DCP problems are then combined to address the MP-BDCP problem.

MP-BDCP belongs to the class of multi-objective optimization problems, for which a solution may not even exist (i.e., the optimal path w.r.t. π_B is not optimal w.r.t. π_D, or vice versa). To eliminate the potential conflict between the two optimization objectives, one can specify a *utility function* that relates π_B and π_D, and use this function as a basis for optimization. For example, one could maximize $\min\{\pi_B(p), \pi_D(p)\}$ or the product $\pi_B(p)\pi_D(p)$. Rather than optimizing a specific utility function, we pursue an approach by which a subset of *nearly nondominated paths* is computed for the given bandwidth and delay constraints. A path p is said to be nondominated if and only if it is not possible to find another path p' for which $\pi_B(p') \geq \pi_B(p)$ *and* $\pi_D(p') \geq \pi_D(p)$. Given a set of nondominated paths, a decision maker can select one of these paths according to his/her specific utility function. Unfortunately, finding all nondominated paths is a hard problem that requires an exponential-time algorithm [12]. Accordingly, we provide a heuristic solution that quantizes π_B using a predetermined step size ϵ, $0 < \epsilon < 1$. For each quantization step, the algorithm attempts to return a path p that maximizes π_D while satisfying $\pi_B(p) > \pi_B(r*) + \epsilon$, where $r*$ is the returned path from the previous step. The set of returned paths constitutes a staircase in the (π_D, π_B) space with step heights of at least ϵ. Note that due to its heuristic nature, our solution does not guarantee finding the optimal π_D at every quantization level. Nonetheless, the returned π_D's are reasonably close to their optimal values, and hence the returned paths are nearly nondominated.

The rest of the paper is organized as follows. In the next section, we formalize the algorithmic definition of the problem and discuss some related work. In Section III, we introduce our solution to one component of the MP-BDCP problem, namely, the one dealing with the delay constraint. The complete solution is provided in Section IV. Extensive simulations and performance evaluation are provided in Section V, followed by conclusions in Section VI.

II. Preliminaries and Related Work

Objectives (1) and (2) of the MP-BDCP problem are often considered separately, giving rise to two problems: the *most-probable bandwidth constrained path* (MP-BCP) problem and the *most-probable delay constrained path* (MP-DCP) problem.

A. MP-BCP Problem

In [13] it was shown that the MP-BCP problem by itself is easy to solve. Accordingly, the authors in [14] provided an exact, polynomial-time algorithm for MP-BCP that uses logarithms. The algorithm associates a probability measure $\rho(i, j) \overset{\text{def}}{=} \Pr[b(i, j) \geq B]$ with every link (i, j), so that $\pi_B(p) = \prod_{(i,j)\in p} \rho(i, j)$. To find a path that maximizes π_B, the weight $(-\log \rho(i, j))$ is assigned to each link (i, j). The conventional shortest path algorithm is then executed. We now provide (and later use) another exact, polynomial-time algorithm for this problem that does not require computing logarithms. Specifically, we modify Dijkstra's algorithm as follows. For every node u, we associate a label $\rho[u]$ that represents the maximum probability that a path from s to u will satisfy the constraint B. Initially, $\rho[s]$ is set to 1 while $\rho[v]$ is set to 0 for all nodes $v \neq s$. The graph is then explored by extracting from the heap the next node u that has the maximum ρ and relaxing every link (u, v) for which $\rho[v] < \rho[u] * \rho(u, v)$. The latter step directly finds the optimal value of π_B without computing logarithms.

In both solutions of MP-BCP, the node that performs the path computation needs to know the values of $\Pr[b(i, j) \geq B]$ for all links $(i, j) \in E$. These probabilities can be obtained from the advertised means and variances, assuming some functional form for the pdf of the $b(i,j)$'s. Since QoS routing has not yet been implemented in real networks, we do not know yet what would be an appropriate pdf for the available bandwidth of a link (the statistical characteristics of $b(i,j)$ will depend on how much, how often, and for how long bandwidth is being requested by traffic flows). Fortunately, the above MP-BCP solutions still apply, independent of the form of this pdf. In related studies (e.g., [15], [16], [17]), researchers assumed that $b(i,j)$ is uniformly distributed over some range $[lb, ub]$. The same uniform distribution will be used in our simulations. In this case, the lb and ub values can be advertised in place of the mean and variance.

B. MP-DCP Problem

The MP-DCP problem is known to be NP-hard [11], and can only be dealt with using heuristics and approximate solutions. Essentially, MP-DCP is an instance of the stochastic shortest path problem, which was investigated in the literature (e.g., [18], [19]). One key issue in stochastic shortest path problems, in general, is how to define the optimality of a path. Some formulations (e.g., [20], [21], [22], [23]) aim at finding the most likely shortest path. Others consider the least-expected-delay paths under interdependent or time-varying probabilistic link delays (e.g., [24], [25], [26]). In [27] the author investigated dynamic stochastic shortest path problems in which the probabilistic link weight is "realized" (i.e., becomes exactly known) once the node is visited. Several studies define path optimality in terms of maximizing a user-specified objective function (e.g., [18], [28], [29], [30], [31]). Our formulation of the MP-DCP problem belongs to this category, where the objective is to find a path that is most likely to satisfy the given delay constraint.

In [14] the authors started with the same MP-DCP formulation used in this paper. They acknowledged the problem and provided solutions for special cases of it. Then they modified the problem into one in which the goal is to partition the given end-to-end delay constraint into local link constraints. The optimal path for the new problem is, in general, different from the one for the original MP-DCP problem, and its approximate solutions are computationally more expensive than the MP-DCP solutions presented in this paper. Note that the work in [14] does not address the combined MP-BDCP problem. Due to these differences, our algorithmic developments take a completely different path from the one in [14].

To provide an efficient, general solution for the MP-DCP problem, which can then be integrated into a solution for the MP-BDCP problem, we assume that for each link (i,j), $d(i,j)$ is a nonnegative rv with mean $\mu(i,j)$ and variance $\sigma^2(i,j)$, and that link delays are mutually independent. We make no assumptions on the pdf of $d(i,j)$ except that it is "smooth," which means that the pdf is continuous and differentiable over some domain, say (a,b) [32], [33]. The smoothness assumption, satisfied by many distributions, enables us to apply the central limit theorem (CLT) approximation (see [34, 376-378]), resulting in a *path delay* that is approximately normally distributed. Note that cutoffs or truncations do not preclude a pdf from being smooth. A good example is the uniform distribution, which has two cutoff points. In [35, 214-215] it is shown that the sum of as small as three uniform rvs already tends to a normal distribution. The CLT approximation also applies even when the pdfs of the local delays are "heavy tailed" (e.g., lognormal, Weibull), provided that the variances are finite. A recent experimental study [36] indicated that the round-trip time can be well approximated by a truncated normal distribution, which further supports our use of the CLT approximation for the delay case.

Let $h(p)$ be the number of hops of a path p. According to the CLT, as $h(p)$ increases $d(p)$ tends to a normally distributed rv with mean

$$\mu(p) = \sum_{(i,j)\in p} \mu(i,j)$$

and variance

$$\sigma^2(p) = \sum_{(i,j)\in p} \sigma^2(i,j).$$

With $d(p)$ approximately normal, the objective function π_D becomes

$$\pi_D(p) \approx \Phi(\frac{D-\mu(p)}{\sigma(p)}),$$

where $\Phi(x) \overset{\text{def}}{=} \frac{1}{2\pi}\int_{-\infty}^{x} e^{-y^2/2}dy$ is the cumulative distribution function of a standard normal rv. Since Φ is monotone, maximizing it with respect to p is equivalent to maximizing the argument $(D-\mu(p))/\sigma(p)$. Hence, the MP-DCP problem reduces to finding the path p that maximizes

$$\mathcal{X}_D(p) \overset{\text{def}}{=} \frac{D-\mu(p)}{\sigma(p)} \tag{3}$$

Let $\mathcal{P}$ be the set of paths between the source node s and the destination node t. To maximize (3), we consider two cases, denoted by CASE-I and CASE-II. CASE-I deals with the situation in which there exists a path $p \in \mathcal{P}$ for which $\mu(p) \le D$ (hence, $\mathcal{X}_D(p) \ge 0$). In this case, to maximize (3) we seek a path $r* \in \mathcal{P}$ that minimizes both μ and σ^2. In contrast, CASE-II has that $\forall p \in \mathcal{P},\ \mu(p) > D$, implying $\mathcal{X}_D(p) < 0$. In this case, we seek a path $r* \in \mathcal{P}$ that maximizes σ while simultaneously minimizing μ.

A number of approximate solutions were previously suggested for CASE-I (see [19] for an extensive survey). The main idea in these solutions is to find a path p that minimizes a linear combination of μ and σ^2, i.e., $\alpha\mu + (1-\alpha)\sigma^2$, where $\alpha \in [0,1]$. For a given α, the linear combination can be easily minimized by associating a weight $w(i,j) = \alpha\mu(i,j) + (1-\alpha)\sigma^2(i,j)$ with every link (i,j) and executing the shortest path algorithm w.r.t. $w(i,j)$. This technique, often referred to as the *line search method*, is used in the next section to devise efficient solutions for both cases of the MP-DCP problem.

III. Proposed Algorithms for the MP-DCP Problem

In this section, we provide efficient solutions to both cases of the MP-DCP problem. First, we need to decide which case to consider for a given input (i.e., $G = (V,E), s, t, D$). This can be done using Heuristic-MP-DCP in Figure 1.

Heuristic-MP-DCP starts by computing the shortest path $p*$ w.r.t. μ from s to t. It then checks whether $\mu(p*) \le D$ or not. If so (CASE-I), Heuristic-MP-DCP computes the shortest path

[2]We will sometimes write $\mu(p)$ and $\sigma(p)$ without the argument to simplify the notation.

```
Heuristic-MP-DCP($G = (V,E), s, t, D$)
1.  Using Dijkstra's algorithm find $p*$
    such that $\forall p \in \mathcal{P},\ \mu(p*) \le \mu(p)$
2.  if $\mu(p*) \le D$ then // CASE-I
3.      Using Dijkstra's algorithm find $q*$
        such that $\forall p \in \mathcal{P},\ \sigma^2(q*) \le \sigma^2(p)$
4.      if $\mathcal{X}_D(p*) \ge \mathcal{X}_D(q*)$ then
5.          Set $r* = p*$
6.      else
7.          Set $r* = q*$
8.      end if
9.      Execute min$\mu$min$\sigma$($G = (V,E), s, t, D, p*, q*, r*$)
10. else // CASE-II
11.     Set $r* = p*$
12.     Execute min$\mu$max$\sigma$($G = (V,E), s, t, D, p*, r*$)
13. end if
```

Fig. 1. Main algorithm for the MP-DCP problem.

$q*$ w.r.t. the variance (i.e., $\sigma^2(q*) \le \sigma^2(p)\ \forall p \in \mathcal{P}$), selects the better of $p*$ or $q*$ as its initial best known path (denoted by $r*$), and calls minμminσ. Note that $p*$ and $q*$ can be computed at once for all destinations by executing Dijkstra's algorithm at a given source node. If there are more than one $p*$ ($q*$), the algorithm selects the one with the minimum variance (mean). This can be easily done by using a hierarchical version of Dijkstra's algorithm [37].

If $\mu(p*) > D$ (CASE-II), Heuristic-MP-DCP selects $p*$ as its initial best known path, again denoted by $r*$, and calls algorithm minμmaxσ. Note that in CASE-II, the optimal path (if found) will satisfy the delay constraint with a probability less than 0.5. If this probability is too small, then the decision maker can completely skip the execution of minμmaxσ, avoiding unnecessary computations.

Both algorithms minμminσ and minμmaxσ consider the geometry of the feasibility region along with the contours of the objective function (3) in the (μ, σ^2) parameter space. Therefore, to facilitate the presentation of both algorithms, we will extensively rely on a two-dimensional (μ, σ^2) representation of the paths between s and t. An example of this representation is shown in Figure 2. The four paths from s to t are represented by the black circles in Figure 2(b). The process of minimizing $\mu+\beta\sigma^2$ (which constitutes the basis for the solution of CASE-I) is also illustrated in the figure by sliding a line of slope of $-1/\beta$, starting at the origin and moving outward in the direction of the arrow. The first path (i.e., black circle) to be encountered is the shortest w.r.t. $\mu + \beta\sigma^2$. In CASE-II, a line search method is used but with a positive line slopes (the line search is also combined with a modified version of the k-shortest paths algorithm to yield good results).

A. Algorithm minμminσ for CASE-I

Recall that for CASE-I, it is desired to find a path that minimizes both the mean and the variance so that (3) can be maximized. Algorithm minμminσ, depicted in Figure 3, tries to

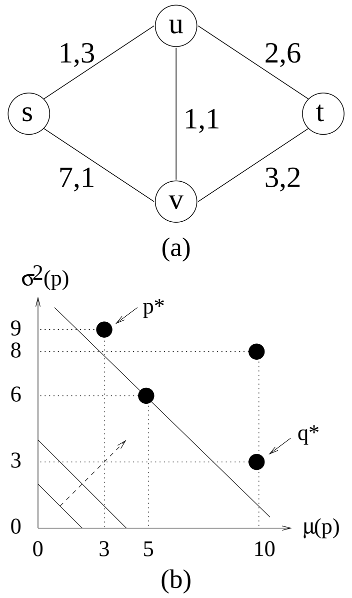


Fig. 2. Representing the paths from s to t in the (μ, σ^2) space: (a) original network with each link assigned a μ and a σ^2 delay parameters; (b) the corresponding (μ, σ^2) representation.

achieve this goal. It starts by checking if $p*$ and $q*$ have the same μ and σ^2. If so, then the currently known $r*$ is the optimal path, since it has the minimum mean and variance. Otherwise, minμminσ proceeds to scan the (μ, σ^2) space for a better $r*$ using the linear combination $\mu + \beta\sigma^2$ with the objective of finding a path that maximizes $\mathcal{X}_D$. Note that $r*$ represents the *so-far* determined optimal path. In each call/step, the algorithm tries to improve the selection and returns a better $r*$.

In the (μ, σ^2) space, the contour of equally probable paths (ones with the same $\mathcal{X}_D$ value) constitutes a parabolic curve of the form $\sigma^2(p) = \frac{D^2-2D\mu(p)+\mu^2(p)}{\mathcal{X}_D^2}$. Two such parabolas are depicted in Figure 4 when $D = 9$. Note that $\mathcal{X}_D(p)$ (and thus $\pi_D(p)$) increases as the contour curve gets closer to the origin. So, algorithm minμminσ needs to determine a path that lies in the shaded region of Figure 4 and whose contour curve is closest to the origin. To do this, minμminσ first computes an appropriate value for β. This is done by computing the coordinates of two points (fictitious paths) p' and q' in the (μ, σ^2) space at which the contour curve of the current $r*$ intersects the vertical and horizontal lines of $p*$ and $q*$, respectively (see Figure 5). Note that in Figure 5 $p' = p*$, but that is not always the case. The points p' and q' satisfy the following equalities:

$$\begin{aligned}\mathcal{X}_D(r*) &= \frac{D-\mu(p')}{\sigma(p')} = \frac{D-\mu(q')}{\sigma(q')}\\ \mu(p') &= \mu(p*)\\ \sigma(q') &= \sigma(q*)\end{aligned}$$

From these equations, we can compute $\sigma(p')$ and $\mu(q')$:

$$\begin{aligned}\sigma(p') &= \frac{D-\mu(p*)}{\mathcal{X}_D(r*)}\\ \mu(q') &= D-\sigma(q*)\mathcal{X}_D(r*)\end{aligned}$$

```
min$\mu$min$\sigma$($G=(V,E), s, t, D, p*, q*, r*$)
1.  if $\mu(p*)=\mu(q*)$ and $\sigma^2(p*)=\sigma^2(q*)$ then
2.      return $r*$ // $r*$ is the optimal path
3.  end if
4.  Compute the mean and variance of the two fictitious
        paths $p'$ and $q'$ for which $\mathcal{X}_D(p')=\mathcal{X}_D(q')=\mathcal{X}_D(r*)$:
4.1     Set $\mu(p')=\mu(p*)$ and $\sigma(q')=\sigma(q*)$
4.2     Set $\sigma(p')=\frac{D-\mu(p*)}{\mathcal{X}_D(r*)}$ and $\mu(q')=D-\sigma(q*)\mathcal{X}_D(r*)$
5.  Set $\beta=\frac{\mu(q')-\mu(p')}{\sigma^2(p')-\sigma^2(q')}$
6.  Set $w(i,j)=\mu(i,j)+\beta\sigma^2(i,j)\ \ \forall(i,j)\in E$
7.  Using Dijkstra's algorithm find the path $r$ such that
        $\forall p\in\mathcal{P},\ \mu(r)+\beta\sigma^2(r)\le\mu(p)+\beta\sigma^2(p)$
8.  if $\mu(r)+\beta\sigma^2(r)\ge\mu(p')+\beta\sigma^2(p')$ then
9.      return $r*$ // $r*$ is the optimal path
10. end if
11. if $\mathcal{X}_D(r)>\mathcal{X}_D(r*)$ then
12.     Set $r*=r$
13.     Recompute the points $p'$ and $q'$ as in step 4
14. end if
15. Compute the intersection points $p^+$ and $q^+$ of the line
        $\mu(r)+\beta\sigma^2(r)$ and the contour curve of $r*$:
15.1    Set $a=\beta$, $b=\mathcal{X}_D(r*)$, and $c=D-(\mu(r)+\beta\sigma^2(r))$
15.2    Set $\sigma(p^+)=\frac{b+\sqrt{b^2-4ac}}{2a}$ and $\mu(p^+)=D-b\sigma(p^+)$
15.3    Set $\sigma(q^+)=\frac{b-\sqrt{b^2-4ac}}{2a}$ and $\mu(q^+)=D-b\sigma(q^+)$
16. Compute the upper bound $opt$ on the optimal path:
16.1    Set $\mu(opt_p)=\mu(p*)$ and $\sigma(opt_p)=\sqrt{\frac{\mu(r)+\beta\sigma^2(r)-\mu(opt_p)}{\beta}}$
16.2    Set $\sigma(opt_q)=\sigma(q*)$ and
        Set $\mu(opt_q)=\mu(r)+\beta\sigma^2(r)-\beta\sigma^2(opt_q)$
16.3    Set $\mathcal{X}_D(opt)=\max\{\mathcal{X}_D(opt_p),\mathcal{X}_D(opt_q)\}$
16.4    if $\mathcal{X}_D(opt)$ is close enough to $\mathcal{X}_D(r*)$, return $r*$
17. if $\sigma(p^+)<\sigma(p')$ then
18.     Execute min$\mu$min$\sigma$($G=(V,E), s, t, D, p', p^+, r*$)
19. end if
20. if $\mu(q^+)<\mu(q')$ then
21.     Execute min$\mu$min$\sigma$($G=(V,E), s, t, D, q^+, q', r*$)
22. end if
23. return $r*$
```

Fig. 3. Algorithm minμminσ for CASE-I of the MP-DCP problem.

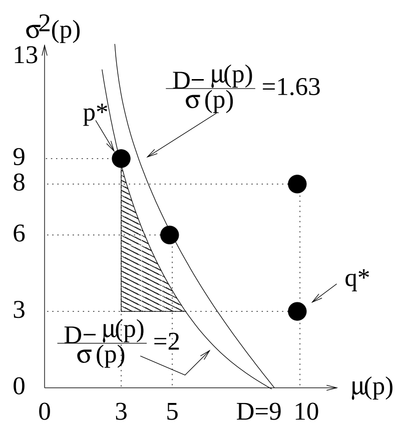


Fig. 4. Equally probable paths in the (μ, σ^2) space (the shaded area represents the region that needs to be searched).

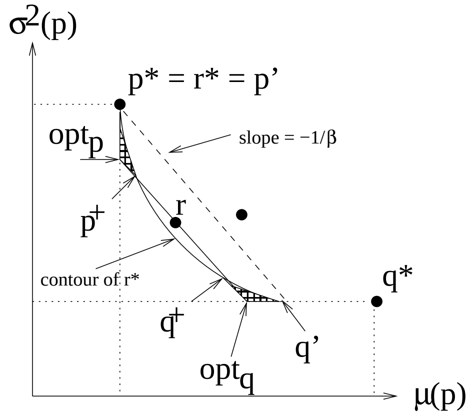


Fig. 5. Points p', q', p^+, q^+, opt_p, and opt_q used in minμminσ. The plot represents one scenario in which $\mathcal{X}_D(r)<\mathcal{X}_D(r*)$ and $L_\beta(r)<L_\beta(p')$ (the two shaded regions are the areas that still need to be searched; actual paths are represented by black circles).

For any point p in the (μ,σ^2) plane, let $L_\beta(p)\overset{\text{def}}{=}\mu(p)+\beta\sigma^2(p)$. The initial value of β is obtained by solving $L_\beta(p')=L_\beta(q')$ for β, i.e., β is the inverse of the slope of the line that passes through both p' and q'. Accordingly,

$$\beta=\frac{\mu(q')-\mu(p')}{\sigma^2(p')-\sigma^2(q')}.$$

By associating the weight $w(i,j)\overset{\text{def}}{=}\mu(i,j)+\beta\sigma^2(i,j)$ with every link $(i,j)\in E$ and using Dijkstra's algorithm w.r.t. $w(i,j)$, minμminσ scans the search space and returns a path r. If $L_\beta(r)\ge L_\beta(p')$, then it is easy to see that the current $r*$ must be optimal since the whole space between the origin and the contour curve of $r*$ has already been scanned, so the algorithm stops (note that p' is not necessarily a real path, so we could have a scenario in which the strict inequality occurs). On the other hand, if $L_\beta(r)<L_\beta(p')$, the algorithm needs to scan the space between the line $\mu(p)+\beta\sigma^2(p)=L_\beta(r)$ and the contour curve of the current $r*$ (i.e., the two shaded regions in Figure 5). To do this, minμminσ first updates $r*$: if $\mathcal{X}_D(r)>\mathcal{X}_D(r*)$, then $r*$ is set to r. The two points p' and q' are recomputed as before w.r.t. the new $r*$. The algorithm then finds the two points, indicated by p^+ and q^+, where the line $L_\beta(p)=L_\beta(r)$ intersects the contour of $r*$ (whose equation is given by $\mathcal{X}_D(p)=\mathcal{X}_D(r*)$). From the line equation, we have

$$\mu(p)=\mu(r)+\beta\sigma^2(r)-\beta\sigma^2(p) \tag{4}$$

From the contour equation, we have

$$\mu(p)=D-\mathcal{X}_D(r*)\sigma(p) \tag{5}$$

Equating the right hand sides of the last two equations, we end up with a quadratic polynomial in $\sigma(p)$:

$$\underbrace{\beta}_{a}\sigma^2(p)-\underbrace{\mathcal{X}_D(r*)}_{b}\sigma(p)+\underbrace{D-(\mu(r)+\beta\sigma^2(r))}_{c}=0. \tag{6}$$

Its roots, denoted by $\sigma(p^+)$ and $\sigma(q^+)$, are:

$$\begin{aligned}\sigma(p^+)&=\frac{b+\sqrt{b^2-4ac}}{2a}\\ \sigma(q^+)&=\frac{b-\sqrt{b^2-4ac}}{2a}\end{aligned}$$

By substituting $\sigma(p^+)$ and $\sigma(q^+)$ for $\sigma(p)$ in $\mu(p) = D - \mathcal{X}_D(r*)\sigma(p)$, we can compute $\mu(p^+)$ and $\mu(q^+)$:

$$\begin{aligned} \mu(p^+) &= D - b\sigma(p^+) \\ \mu(q^+) &= D - b\sigma(q^+). \end{aligned}$$

After determining p^+ and q^+, minμminσ proceeds to scan the regions between the line $L_\beta(p) = L_\beta(r)$ and the contour curve of the current $r*$ (the shaded regions in Figure 5). If a better path than $r*$ exists, then it must fall in one of these regions. To scan these regions, minμminσ determines two lines based on the above fictitious paths. The first line passes through p' and p^+ while the second one passes through q^+ and q'. If $\sigma(p^+) \leq \sigma(p')$, min$\mu$min$\sigma$ calls itself using p' and p^+ as input arguments in place of $p*$ and $q*$, respectively (i.e., it scans using the slope of the first line). Furthermore, if $\mu(q^+) \leq \mu(q')$, then minμminσ calls itself using q^+ and q' in place of $p*$ and $q*$, respectively (i.e., it scans using the slope of the second line).

The algorithm can be run recursively until the optimal path is found. The area in the (μ, σ^2) that is yet to be searched shrinks with each iteration, indicating that $r*$ is getting closer to (if it is not already) the optimal path. A stopping criterion can be devised by computing, in each iteration, an upper bound on the optimal path. If the current $r*$ is close enough to this bound, the algorithm stops and returns the current $r*$ as the solution. This upper bound is computed as follows. In each iteration, minμminσ computes the coordinates of two fictitious paths, opt_p and opt_q, which can be thought of as candidates for the optimal path. The manner in which these paths are determined will be described shortly. Once opt_p and opt_q are computed, the "better" of them, called opt, is determined according to:

$$\mathcal{X}_D(opt) = \max\{\mathcal{X}_D(opt_p), \mathcal{X}_D(opt_q)\}$$

From the definition of opt_p and opt_q, $\mathcal{X}_D(opt)$ represents an upper bound on $\mathcal{X}_D$ of the optimal path. So if in a given iteration $\mathcal{X}_D(r*) \approx \mathcal{X}_D(opt)$, min$\mu$min$\sigma$ returns the current $r*$ and stops. Otherwise, it keeps searching.

The fictitious paths opt_p and opt_q are defined by the intersections of the current search line with the vertical and horizontal lines at $p*$ and $q*$, respectively. By definition, $\mu(opt_p) = \mu(p*)$ and $\sigma(opt_q) = \sigma(q*)$. To compute $\mu(opt_p)$ and $\mu(opt_q)$, we have

$$\begin{aligned} \mu(opt_p) + \beta\sigma^2(opt_p) &= \mu(r) + \beta\sigma^2(r) \\ \mu(opt_q) + \beta\sigma^2(opt_q) &= \mu(r) + \beta\sigma^2(r) \end{aligned}$$

From these equations, we compute:

$$\begin{aligned} \sigma(opt_p) &= \sqrt{\frac{\mu(r) + \beta\sigma^2(r) - \mu(opt_p)}{\beta}} \\ \mu(opt_q) &= \mu(r) + \beta\sigma^2(r) - \beta\sigma^2(opt_q) \end{aligned}$$

which completes the determination of $\mathcal{X}_D(opt)$.

The complexity of minμminσ is equal to the number of calls to Dijkstra's algorithm, i.e., the number of lines that are systematically used to estimate the contour curve of the optimal path that maximizes $\mathcal{X}_D$. Even though this number is not theoretically bounded, it is intuitively clear that one side of a parabolic curve can be adequately estimated with a few lines. In fact, simulation results indicate that minμminσ often finds the optimal path within three iterations of Dijkstra's algorithm (two of which are used to compute $p*$ and $q*$).

It is worth emphasizing that the line search concept has been used in previous studies [19], but based on a line adjustment strategy that is different from the one presented here. More specifically, in previous studies the line search is performed by considering only the *extreme* nondominated paths (e.g., $p*$ and $q*$), without exploiting the convexity and structure of $\mathcal{X}_D$. As a result, the line may unnecessarily scan a vast region in the parameter space that contains dominated paths. Ultimately, it returns a dominated path, wasting the computational resources. In minμminσ, we avoid such a trap by careful selection of β that guarantees that in each iteration *the newly computed $r*$ cannot be worse (in terms of $\mathcal{X}_D$) than the most recent $r*$.*

B. Algorithm minμmaxσ for CASE-II

For CASE-II (i.e., $\forall p \in \mathcal{P},\ \mu(p) > D$), one has to minimize the mean and maximize the variance so that $\mathcal{X}_D$ can be maximized. To achieve this goal, we devise algorithm minμmaxσ, depicted in Figure 6. As in CASE-I, we use a line search method but with a positive slope. The contours of $\mathcal{X}_D$ are now characterized by a parabola of the form $\sigma^2(p) = \frac{D^2 - 2D\mu(p) + \mu^2(p)}{\mathcal{X}_D^2}$. In contrast to CASE-I, the value of $\mathcal{X}_D$ here is always negative, and it increases as the parabola gets closer to the upper-left corner of the (μ, σ^2) space (see the example in Figure 7). This suggests that the line search should proceed from the upper-left corner to the bottom-right corner of the (μ, σ^2) space, as shown in Figure 8. This search can be done by minimizing over $p \in \mathcal{P}$ the objective function $L_\beta(p) \overset{\text{def}}{=} \mu(p) - \beta\sigma^2(p)$, where $\beta > 0$. This is done by associating a weight $w(i,j) \overset{\text{def}}{=} \mu(i,j) - \beta\sigma^2(i,j)$ with every link (i,j) and executing the shortest path algorithim w.r.t. w. Note that β must be chosen so that all link weights are nonnegative (otherwise, the graph may contain negative cycles). Algorithm minμmaxσ starts by selecting $\beta = \min\{\frac{\mu(i,j)}{\sigma^2(i,j)} : (i,j) \in E\}$. It then executes Dijkstra's algorithm, returning a path r that minimizes L_β. If $\mathcal{X}_D(r) > \mathcal{X}_D(r*)$, then $r*$ is set to r (as before, $r*$ represents the so-far best path w.r.t. $\mathcal{X}_D$; it is initially set to $p*$).

In contrast to CASE-I in which the search line faces the outer (convex) surface of the parabola, in CASE-II the line search faces the inner (concave) surface of the parabola (see Figure 8). As a result, the region between the line $L_\beta(p) = L_\beta(r)$ and the contour curve of the current $r*$ (i.e., Region A in Figure 8) will not be scanned during the computation of r. To scan this region, the current search line or an adjusted one (see step 7 in Figure 6) needs to be kept sliding along the arrow. This can be done by using the k-shortest paths algorithm. However, if the standard k-shortest paths algorithm (for example, the one provided in [38]) were to be used (i.e., if the line is kept sliding along the arrow), it will scan both Region A and the region on the right side of the contour curve of $r*$ (Region B in Figure 8). All paths in Region B have smaller $\mathcal{X}_D$ values than $\mathcal{X}_D(r*)$, and hence scanning Region B is not needed. Unless k is very large, Region A cannot be completely scanned since many of the returned k-shortest paths will come from Region B. Obviously, a large k renders the algo-

```
min$\mu$max$\sigma(G = (V, E), s, t, D, p*, r*)$
1.  Set $\beta = \min\{\frac{\mu(i,j)}{\sigma^2(i,j)} \mid (i,j) \in E\}$
2.  Set $w(i,j) = \mu(i,j) - \beta\sigma^2(i,j) \;\; (i,j) \in E$
3.  Let $L_\beta(p) \overset{\text{def}}{=} \mu(p) - \beta\sigma^2(p)$ for any path $p$. Using Dijkstra's
    algorithm find a path $r$ such that $\forall p \in \mathcal{P},\ L_\beta(r) \leq L_\beta(p)$
4.  if $\mathcal{X}_D(r) > \mathcal{X}_D(r*)$ then set $r* = r$
5.  Compute the fictitious paths $opt$ and $tg$
5.1     if $D \geq L_\beta(r)$ then
5.2         Set $\mu(opt) = \mu(p*)$ and $\sigma^2(opt) = \frac{\mu(opt) - L_\beta(r)}{\beta}$
5.3     else
5.4         Set $\sigma^2(opt) = \frac{L_\beta(r) - D}{\beta}$ and $\mu(opt) = L_\beta(r) + \beta\sigma^2(opt)$
5.5         if $\mu(opt) < \mu(p*)$ then
5.6             Set $\mu(opt) = \mu(p*)$ and $\sigma^2(opt) = \frac{\mu(p*) - L_\beta(r)}{\beta}$
5.7     end
5.8     Set $\sigma(tg) = \frac{\mathcal{X}_D(r*)}{-2\beta}$ and $\mu(tg) = D - \mathcal{X}_D(r*)\sigma(tg)$
6.  if $\mathcal{X}_D(opt) \approx \mathcal{X}_D(r*)$ or $L_\beta(r) \approx L_\beta(tg)$, then
6.1     return $r*$
7.  Adjust $\beta$ and recompute $w(i,j)$ if needed
7.1     if $\mathcal{X}_D(r) > \mathcal{X}_D(p*)$ then Set $\beta = \min\{\beta, \frac{\mu(r) - \mu(p*)}{\sigma^2(r) - \sigma^2(p*)}\}$
7.2     else if $\mathcal{X}_D(r) < \mathcal{X}_D(p*)$ then
7.3         Compute the point $r^+$ where $\mu(r) - \beta\sigma^2(r)$ and
            the contour curve of $r*$ intersect:
7.3.1           Set $a = \beta$, $b = \mathcal{X}_D(r*)$, and $c = -D + \mu(r) - \beta\sigma^2(r))$
7.3.2           Set $\sigma(r^+) = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$ and $\mu(r^+) = D - b\sigma(r^+)$
7.4         Set $\beta = \min\{\beta, \frac{\mu(r^+) - \mu(p*)}{\sigma^2(r^+) - \sigma^2(p*)}\}$
7.5     end
7.6     $w(i,j) = \mu(i,j) - \beta\sigma^2(i,j) \;\; \forall(i,j) \in E$
8.  Using Reverse-Dijkstra's algorithm find $\widetilde{r_v}$ from every node $v$ to
    node $t$ such that $\forall p_v \in \widetilde{\mathcal{P}},\ \mu(\widetilde{r_v}) - \beta\sigma^2(\widetilde{r_v}) \leq \mu(p_v) - \beta\sigma^2(p_v)$
9.  Execute the $k$-shortest paths algorithm with the following modifications:
9.1     Extract node $u$ and index $i$, $1 \leq i \leq k$, from the heap
            (i.e., $r_i$, the $i$th shortest path from $s$ to $u$)
9.2     if $u = d$ and $\mathcal{X}_D(r_i) > \mathcal{X}_D(r*)$ then set $r* = r_i$
9.3     Compute the upper bounds $opt$ and $tg$ as in step 5 with $r = r_i$
9.4     if $\mathcal{X}_D(opt) \leq \mathcal{X}_D(r*)$ or $\mu(r_i) - \beta\sigma^2(r_i) \geq \mu(tg) - \beta\sigma^2(tg)$
            then return $r*$ // $r*$ is the optimal path
9.5     if the condition in step 6 holds, then return $r*$
9.6     Otherwise, consider each link $(u,v)$ for relaxation
9.6.1       Set $\mu(tmp) = \mu(r_i) + \mu(u,v)$
            Set $\sigma^2(tmp) = \sigma^2(r_i) + \sigma^2(u,v)$
9.6.2       if $\mu(tmp + \widetilde{r_v}) - \beta\sigma^2(tmp + \widetilde{r_v}) > \mu(tg) - \beta\sigma^2(tg)$ or
              $(\mathcal{X}_D(tmp + \widetilde{r_v}) < \mathcal{X}_D(r*)$ and $v = t)$ or
              $(\mathcal{X}_D(tmp) < \mathcal{X}_D(r*)$ and $\mu(tmp) > \mu(tg))$ then
              do not relax $(u,v)$
9.6.3       else relax $(u,v)$ as described in [38]
10. return $r*$
```

Fig. 6. Algorithm minμmaxσ for CASE-II of the MP-DCP problem.

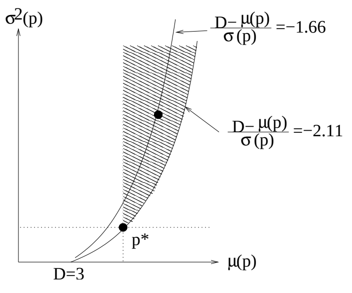


Fig. 7. Equally probable paths in CASE-II (the shaded area is the region that needs to be searched following the computation of $p*$).

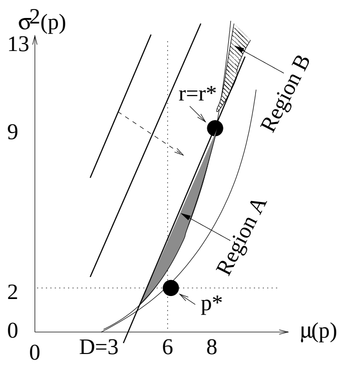


Fig. 8. Scanning the (μ, σ^2) space in CASE-II.

rithm impractical, given the $\mathcal{O}(km\log(kn) + k^2m)$ complexity of the k-shortest paths algorithm [38], where n is the number of nodes and m is the number of links.

To prevent the algorithm from scanning Region B, we modify the k-shortest paths algorithm so that *it does not explore or store dominated paths or sub-paths that lead to dominated paths*. By doing so, the modified algorithm considers more paths from Region A, which improves the performance using a reasonably small k. Our modifications require determining two points (fictitious paths) in the (μ, σ^2) space, which are denoted by opt and tg (see Figure 9). Point opt is used to compute an upper bound on the optimal path w.r.t. $\mathcal{X}_D$, while point tg is used to compute an upper bound on the linear search w.r.t. L_β[3]. These two points are computed as follows. Suppose that paths $p*$, r, and $r*$ have already been determined. If $L_\beta(r) \leq D$, then opt is the point at which the line $L_\beta(p) = L_\beta(r)$ intersects the vertical line at $p*$, as shown in Figure 9(a). Thus, the coordinates of opt are given by $\mu(opt) = \mu(p*)$ and $\sigma^2(opt) = \frac{\mu(opt) - L(r)}{\beta}$. On the other hand, if $L_\beta(r) > D$ then opt is the point on the line $L_\beta(p) = L_\beta(r)$ that maximizes $\mathcal{X}_D$ (see Figure 9(b)). In other words,

$$\frac{d}{d\sigma(p)}\left(\frac{D - \mu(p)}{\sigma(p)}\right)|_{p=opt} = 0.$$

Since $\mu(opt) - \beta\sigma^2(opt) = L_\beta(r)$, we have $\mu(opt) = L(r) + \beta\sigma^2(opt)$ and

$$\frac{d}{d\sigma(p)}\left(\frac{D - L(r) - \beta\sigma^2(p)}{\sigma(p)}\right)|_{p=opt}$$

$$= \frac{-2\beta\sigma^2(opt) - D + L(r) + \beta\sigma^2(opt)}{\sigma^2(opt)} = 0$$

from which

$$\sigma^2(opt) = \frac{L(r) - D}{\beta} \quad \text{and thus} \quad \mu(opt) = L(r) + \beta\sigma^2(opt).$$

If $\mu(opt) < \mu(p*)$, then opt will again be at the point where the vertical line at $p*$ intersects with the line $L_\beta(p) = L_\beta(r)$, as described before.

As for tg, it is the point on the contour curve of $r*$ that maximizes L_β. In other words,

$$\frac{d}{d\sigma(p)}\left(\mu(p) - \beta\sigma^2(p)\right)|_{p=tg} = 0.$$

[3] tg stands for tangent, since the line $L_\beta(p) = L_\beta(tg)$ is tangent to the parabola of $r*$.

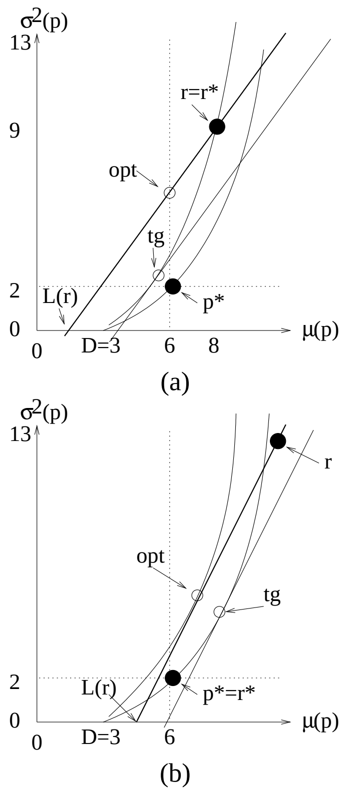


Fig. 9. Computing the fictitious paths opt and tg in CASE-II: (a) $L_\beta(r) \leq D$, (b) $L_\beta(r) > D$.

Since

$$\mathcal{X}_D(tg) = \frac{D - \mu(tg)}{\sigma(tg)} = \mathcal{X}_D(r*)$$

we have $\mu(tg) = D - \mathcal{X}_D(r*)\sigma(tg)$, and thus

$$\frac{d}{d\sigma(p)}\left(D - \mathcal{X}_D(r*)\sigma(p) - \beta\sigma^2(p)\right)|_{p=tg}$$

$$= -\mathcal{X}_D(r*) - 2\beta\sigma(tg) = 0$$

from which

$$\sigma(tg) = \frac{-\mathcal{X}_D(r*)}{2\beta} \text{ and thus } \mu(tg) = D - \mathcal{X}_D(r*)\sigma(tg).$$

After determining opt and tg, we can use them as stopping criteria. More specifically, if $\mathcal{X}_D(opt) \approx \mathcal{X}_D(r*)$ or $L_\beta(r*) \approx L_\beta(tg)$, then $r*$ is sufficiently close to the optimal path. So minμmaxσ stops and returns the current $r*$ as the solution. Otherwise, it proceeds to scan Region A for a better (possibly optimal) path. As mentioned before, to achieve this we use a modified k-shortest paths algorithm (discussed next). Using each shortest path r_i, $1 \leq i \leq k$, we update $r*$ and compute the upper bounds opt and tg as described above, but using r_i in place of r. If $L_\beta(r_i) \geq L_\beta(tg)$, then the current $r*$ is optimal since the whole shaded region is scanned, so the algorithm stops. Otherwise, the algorithm proceeds to the next r_i, and the process is repeated.

Modified k-shortest Paths Algorithm

The k-shortest paths algorithm in [38] is similar to Dijkstra's algorithm except that it associates k labels with every node. At each iteration, the algorithm extracts a node u with index i, $1 \leq i \leq k$, from the heap (i.e., the ith shortest path r_i from s to u) and relaxes every link (u, v). We mainly modify this relaxation step to improve the performance of this algorithm when used in CASE-II of the MP-DCP problem. Our modifications are shown in Figure 6 (starting from step 9). Using Reverse-Dijkstra's algorithm, we first compute the best path $\widetilde{r}_v$ from every node v to the destination node t w.r.t $w(i,j) \overset{\text{def}}{=} \mu(i,j) - \beta\sigma^2(i,j)$. These paths will be used in the modified k-shortest paths algorithm to eliminate sub-paths that lead to paths dominated by $r*$. Assume that node u with index i is extracted from the heap as in the original k-shortest paths algorithm. If $u = t$ then r_i is a complete path from s to t. We can now update $r*$ and compute the points opt and tg as described before using $r = r_i$. If $L_\beta(r_i) \geq L_\beta(tg)$, then the current $r*$ is optimal since all of Region A in Figure 8 is scanned, so the algorithm stops. If $u \neq t$, then r_i is a sub-path from s to u and needs to be extended towards t by considering (relaxing) every link (u, v). We first compute a temporary path tmp from s to v by adding the mean and variance of the link (u, v) to the mean and variance of r_i, i.e., $\mu(tmp) = \mu(r_i) + \mu(u, v)$ and $\sigma^2(tmp) = \sigma^2(r_i) + \sigma^2(u, v)$. If $v = t$ then tmp is a complete path. In this case, if tmp is dominated by $r*$ (i.e., if $\mathcal{X}_D(tmp) \leq \mathcal{X}_D(r*)$), then there is no need to store tmp, so relaxation does not take place. If $v \neq t$ then tmp is a sub-path that needs to be stored since there is a possibility that extending it might lead to a better path than $r*$. We now describe two cases in which tmp can never lead to a better path than the currently known $r*$. By identifying these cases, we can avoid unnecessary computations. For the first case, we concatenate tmp and $\widetilde{r}_v$. If $\mu(tmp + \widetilde{r}_v) - \beta\sigma^2(tmp + \widetilde{r}_v) > \mu(tg) - \beta\sigma^2(tg)$, then the best possible extension of tmp will exceed the upper bound $\mu(tg) - \beta\sigma^2(tg)$ and will fall in Region I in Figure 10. Since

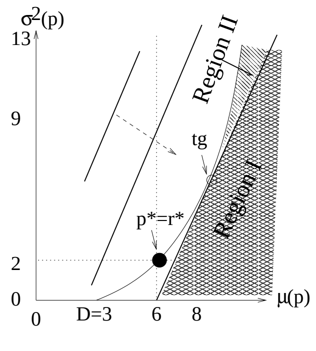


Fig. 10. Relaxing the link (u, v) in CASE-II.

every path in Region I is dominated by $r*$, there is no need to relax (u, v). For the second case, we consider tmp itself. If $\mathcal{X}_D(tmp) < \mathcal{X}_D(r*)$ and $\mu(tmp) > \mu(tg)$ (i.e., tmp falls in Region II in Figure 10), then extending tmp will lead to a path that lies to the upper-right of tmp in the (μ, σ^2) space. Since all the paths in such a region are dominated by $r*$, there is no need to relax (u, v). With these modifications, minμmaxσ can achieve good performance using a small value of k.

IV. Computing a Set of Nondominated Paths for the MP-BDCP Problem

As stated in Definition 1, the MP-BDCP problem aims at maximizing both π_B and π_D. If there is a path that maximizes both functions, then this path is the optimal solution. Otherwise, there is no solution, i.e., the optimal path w.r.t. π_B is not optimal w.r.t. π_D, or vice versa. In the latter case, a decision maker can specify a *utility function* that relates π_B and π_D, and try to find a path that optimizes this utility function. For example, one could maximize $\min\{\pi_B(p), \pi_D(p)\}$ or the product $\pi_B(p)\pi_D(p)$. Rather than optimizing a specific utility function, we focus on how to produce a set of *nondominated paths*, from which a solution can be selected according to a given utility function. Unfortunately, finding all nondominated paths is a hard problem, requiring an exponential-time algorithm [12]. Accordingly, we provide a heuristic algorithm that aims at identifying a partial set of nearly nondominated paths.

A pseudo-code of the proposed algorithm, called Approx-MP-BDCP, is shown in Figure 11. Approx-MP-BDCP starts

```
**Approx-MP-BDCP**($G = (V, E), s, t, D, B, \epsilon$)
1.  Using Heuristic-MP-DCP, find $r^*_D$
2.  Using the modified Dijkstra's algorithm for MP-BCP, find $r^*_B$
3.  **if** $r^*_D$ and $r^*_B$ are the same, **then** return $r^*_D$
4.  Set $\aleph = \{r^*_D, r^*_B\}$
5.  Set $r* = r^*_D$
6.  Using the modified Dijkstra's algorithm for MP-BCP in the
       reverse direction, find the maximum $\widetilde{\rho}[\cdot]$, the probability
       of satisfying $B$, from every node $u$ to node $t$.
7.  **While** $\pi_B(r^*_B) > \pi_B(r*) + \epsilon$
8.     Using Heuristic-MP-DCP with some modifications,[a]
       find $tmp$ such that $\pi_B(tmp) > \pi_B(r*) + \epsilon$
9.     Set $r* = tmp$
10.    Set $\aleph = \aleph \cup \{r*\}$
11. **end**
12. return $\aleph$

[a]Whenever link $(u, v)$ is considered for relaxation, check
if $\rho[u] * \rho(u, v) * \widetilde{\rho}[v] > \pi_B(r*) + \epsilon$. If so, do not relax link $(u, v)$.
```

Fig. 11. Approximate algorithm for finding a subset of the nondominated paths in the MP-BDCP problem.

by computing two paths r^*_D and r^*_B that maximize π_D and π_B, respectively (note that r^*_D is only an approximation of the most-probable delay-constrained path). If $r^*_D = r^*_B$, then this path is the single optimal solution to the MP-BDCP problem. If not, Approx-MP-BDCP proceeds to compute a set of nondominated paths. Figure 12 depicts an example of such paths in the (π_D, π_B) space (black circles indicate nondominated paths, while white ones indicate dominated paths). Nondominated paths form a staircase between r^*_D and r^*_B. Between these two, there might be several other nondominated paths. Approx-MP-BDCP tries to compute a subset of these paths by quantizing π_B using a predetermined quantization step ϵ, $0 < \epsilon < 1$. As a result, all the nondominated paths p for which $\pi_B(p) \leq \pi_B(r*) + \epsilon$ are bypassed, i.e., quantized to $r*$. In here, $r*$ is used to refer to the most recently found nondominated path in the previous iteration (initially, $r*$ is set to r^*_D). Figure 12 also illustrates the quantization process for $\epsilon = 0.2$. The basic idea here is to iteratively find the next nondominated path (indi-

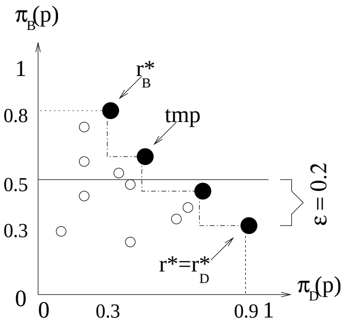


Fig. 12. Dominated and nondominated paths in the (π_D, π_B) space.

cated by tmp) by executing Heuristic-MP-DCP, which attempts to maximize π_D under the constraint $\pi_B(tmp) > \pi_B(r*) + \epsilon$. The process is repeated iteratively to find other nondominated paths so long as $\pi_B(r^*_B) > \pi_B(r*) + \epsilon$. The value of ϵ controls the number of nondominated paths that could be returned by the algorithm, and thus the number of iterations, which is at most $\mathcal{O}(\frac{1}{\epsilon})$. It also determines the goodness of the nondominated paths w.r.t. π_B, i.e., $\pi_B(\text{any nondominated path in the graph}) \leq \pi_B(\text{one of the returned nondominated paths}) + \epsilon$. Note that due to its heuristic nature, Approx-MP-BDCP does not guarantee that the returned path in each iteration is absolutely nondominated. Instead, the returned paths are, in general, nearly nondominated.

Approx-MP-BDCP combines Heuristic-MP-DCP and the solution of MP-BCP, as follows. For each node v, let $\widetilde{p}_v$ be the best path from v to the destination node t w.r.t. $\rho(i,j) \overset{\text{def}}{=} \Pr[b(i,j) \geq B]$, and let $\widetilde{\rho}[v] \overset{\text{def}}{=} \prod_{(i,j)\in\widetilde{p}_v} \rho(i,j)$. The value of $\widetilde{\rho}[v]$, which is used in computing the path tmp, can be determined by modifying Reverse-Dijkstra's algorithm in the same way we modified Dijkstra's algorithm for the MP-BCP problem in Section II. To find the path tmp that maximizes π_D under the constraint $\pi_B(tmp) > \pi_B(r*) + \epsilon$ (line 8 in Figure 11), we execute Heuristic-MP-DCP, but with the following modification. We do not relax any link (u, v) for which $\rho[u] * \rho(u, v) * \widetilde{\rho}[v] > \pi_B(r*) + \epsilon$, where $\rho[u]$ is the probability of satisfying the bandwidth constraint along the path from s to u.

V. Performance Evaluation and Discussion

We conducted extensive simulations to evaluate the performance and computational complexity of the aforementioned algorithmic solutions. Our interest is not only to assess the goodness of these solutions, but to also demonstrate the potential benefits of the probabilistic approach, in general, as a means of reducing the protocol overhead at no loss in the routing performance. Hereafter, we use the term "probabilistic approach" to refer to the probabilistic modelling of uncertainties in link bandwidth and delay (as formalized in Definition 1).

In the probabilistic approach, routers are expected to maintain and advertise two parameters for each QoS measure (e.g., mean and variance for delay, minimum and maximum for available bandwidth). As discussed in Section I, these parameters vary at a much slower pace than the instantaneous delay and bandwidth values. In our simulations, we assume that these statistical parameters are computed and advertised once at the beginning of each simulation run. Source nodes then use the one-time advertised information to determine the most-probable path w.r.t. the delay constraint, the bandwidth constraint, or both. Once this path is computed, we check its feasibility according to the *actual* (instantaneous) link values (which are not available to the path selection algorithm). If the path is feasible according to the actual values, we call the attempt a 'success'. The performance of a path selection algorithm is expressed in terms of the *success rate* (SR), which is the fraction of returned paths that are feasible.

To demonstrate the robustness of the probabilistic approach, we contrast it with the *standard* threshold-based triggered approach. In the triggered approach, the instantaneous bandwidth and delay values are advertised once they exceed certain thresholds, indicated by TH$_B$ and TH$_D$, respectively (for simplicity, we express these thresholds in absolute terms). Consider, for example, the available bandwidth over a given link. If this bandwidth changes (e.g., following the addition of a new flow or the termination of an existing one) such that the absolute difference between the new value and the most recently advertised one exceeds TH$_B$, then a new link state advertisement (LSA) is generated and advertised throughout the domain. Clearly, the smaller the values of TH$_B$ and TH$_D$, the higher is the SR of the triggered approach. But this performance gain comes at the expense of increased advertisement overhead. For the triggered approach, path selection is performed using the algorithm in [10], which was briefly described in Section I. Note that this algorithm treats the available state values as if they were exact. We compare the probabilistic and triggered approaches in terms of the *normalized* SR's and the communications overhead. To measure the communications overhead of the triggered approach, we compute the percentage of links whose bandwidth and delay values changed to the extent of triggering a state update within a given period of time. Note that in our simulations the probabilistic approach uses the one-time advertised statistical information.

Our simulations are based on random topologies that obey the recently observed power laws [39]. These topologies were generated using the BRITE topology generator [40]. In a given topology, each link (i, j) is assigned random delay and bandwidth parameters, indicated by $d(i, j)$ and $b(i, j)$, respectively. We assume that $d(i, j)$ is normally distributed with mean $\mu(i, j)$ and variance $\sigma^2(i, j)$. To produce heterogeneous link delays, we also randomize the selection of $\mu(i, j)$ and $\sigma^2(i, j)$ by taking $\mu(i, j) \sim uniform[10, 200]$ and $\sigma(i, j) \sim uniform[25, 100]$. For the link bandwidth $b(i, j)$, we take it to be uniformly distributed in the range $[lb(i, j), ub(i, j)]$. To produce heterogeneous link bandwidths, we let $lb(i, j) \sim uniform[10, 150]$ and $ub(i, j) \sim lb(i, j) + uniform[20, 50]$. In the probabilistic approach, we assume that $\mu(i, j)$, $\sigma^2(i, j)$, $lb(i, j)$, and $ub(i, j)$ are generated at the beginning of the simulation run and kept fixed thereafter. In the triggered approach, we sample the current and most recent delay and bandwidth values for every link (i, j) using the prespecified $\mu(i, j)$, $\sigma^2(i, j)$, $lb(i, j)$, and $ub(i, j)$. Whenever the current and most recent values of a link parameter differ by more than the threshold, the current value is advertised to refresh the stale link information. We also experimented with other link distributions and other ranges for $\mu(i, j)$, $\sigma^2(i, j)$, $lb(i, j)$, and $ub(i, j)$ using various network sizes, and reached similar conclusions to the ones discussed next. For brevity, we report the results obtained using 100-node random topologies.

A. Performance Evaluation for the MP-DCP Problem

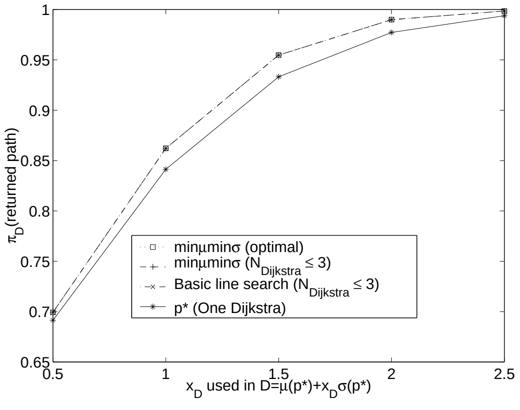
We first asses the performance and computational complexity of the proposed Heuristic-MP-DCP and compare it with the one in [19]. We then demonstrate the efficiency of the probabilistic approach over the triggered-based approaches under a delay constraint. Performance evaluations and comparisons are performed separately for the two cases.

For CASE-I the algorithm in [19] adjusts the search line in each iteration according to the extreme paths determined from the previous iteration. We refer to this approach as the *basic line search*. In contrast, our solution takes into account the contours of previously determined paths, thus avoiding the consideration of many extreme (and non-useful) paths. The computational complexity in both approaches can be expressed in the number of calls to Dijkstra's algorithm. We select the delay constraint D as follows:

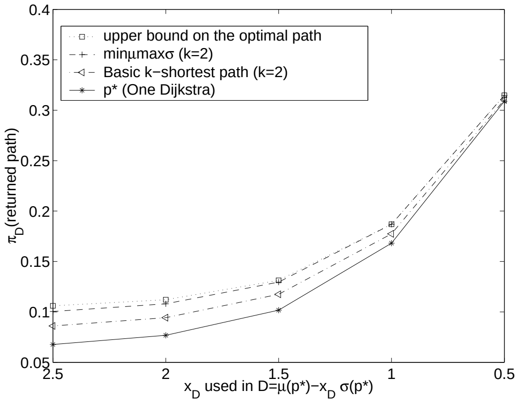
$$D = \mu(p*) + x_D \sigma(p*),$$

where $x_D \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$ is a constant that reflects the tightness of the delay constraint relative to $\mu(p*)$ (recall that $p*$ is the shortest path w.r.t. the mean). As x_D increases, so do D and π_D of the returned path. Let $N_{Dijkstra}$ be the number of calls to Dijkstra's algorithm used by the compared algorithms in CASE-I, including the two calls that are used to compute $p*$ and $q*$. If no restriction is imposed on $N_{Dijkstra}$, both algorithms are likely to find the optimal path. In this case, simulation results indicate that our line search approach requires, on average, 3.5 iterations of Dijkstra's algorithm to find the optimal path, compared to 5.3 iterations in the basic line search approach (about 51% more than ours). The worst-case complexity (averaged over several runs) is found to be 7.6 iterations for our approach, compared to 14.2 iterations for the basic line search approach (about 87% more than ours).

Instead of letting the algorithm run indefinitely until it finds the optimal path, one can impose a limit on $N_{Dijkstra}$. If after $N_{Dijkstra}$ iterations, the optimal path is still not found, the algorithm returns the most recent $r*$ and terminates the search. Simulation results show that the returned paths in this case are very close to the optimal one even when $N_{Dijkstra} \leq 3$. Figure 13(a) depicts π_D versus x_D for the optimal path (obtained by not imposing a limit on $N_{Dijkstra}$), the shortest path $p*$ w.r.t. μ (obtained in one iteration of Dijkstra's algorithm), and the paths returned by the contending algorithms with $N_{Dijkstra} \leq 3$. The differences between the returned paths and the optimal one are barely visible, indicating that line search-based algorithms are capable of achieving an almost optimal performance using no more than three executions of Dijkstra's algorithm. Under the same limit, our algorithm returns slightly better paths than the ones returned by the basic line search. As also seen in the figure, $p*$ is as good as the optimal path when D is close to or significantly larger than $\mu(p*)$. In other cases, however, the gap between $p*$ and the optimal path can be significant, particularly when the distribution of paths in the (μ, σ^2) space is not uniform.



(a)



(b)

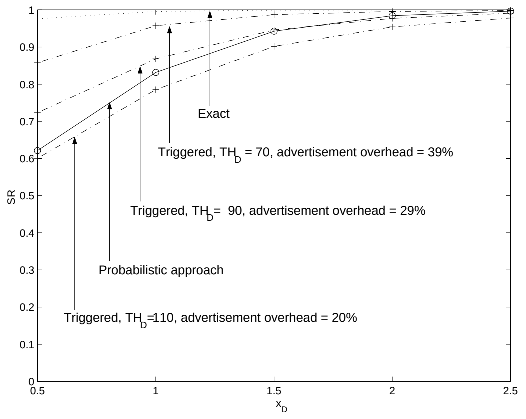
Fig. 13. π_D of various paths versus x_D: (a) CASE-I with $D = \mu(p*) + x_D\sigma(p*)$; (b) CASE-II with $D = \mu(p*) - x_D\sigma(p*)$.

For CASE-II, the algorithm in [19] uses the k-shortest-paths algorithm as is. This algorithm simply searches the (μ, σ^2) space using a line parallel to the σ^2-axis, returning k-shortest paths w.r.t. μ. From these k paths, the one that maximizes the objective function is selected. In contrast, algorithm minμmaxσ first determines a search line with a positive slope, and then uses this line along with the modified k-shortest paths algorithm. The computational complexities of both algorithms is a function of k. We select the delay constraint D as follows:

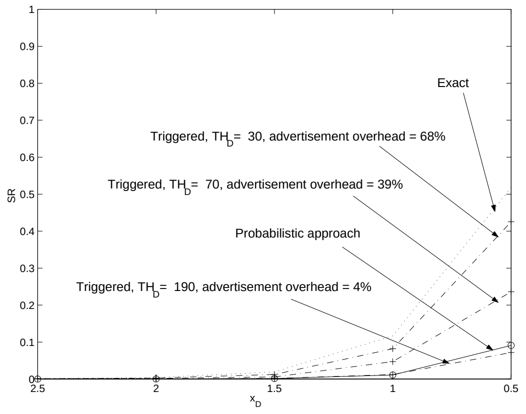
$$D = \mu(p*) - x_D\sigma(p*),$$

where $x_D \in \{0.5, 1.0, 1.5, 2.0, 2.5\}$. As x_D increases, both D and π_D decrease. Recall that minμmaxσ computes an upper bound on the optimal path. This bound becomes tighter as k increases. Hence, we first run minμmaxσ with a large k ($k = 20$), and use the computed upper bound as an approximation of the optimal π_D. Then, we contrast this value with the performance of the basic k-shortest paths algorithm and that of our algorithm with $k = 2$. The results, shown in Figure 13(b), suggest that the performance of our algorithm is near-optimal. They also indicate that our algorithm is better than the basic k-shortest paths algorithm for the same computational complexity. As D decreases, the gap between π_D(returned path) and $\pi_D(p*)$ further widens. The gap narrows as D gets close to $\mu(p*)$.

We now contrast the probabilistic and the standard triggered-based approaches. Figure 14 depicts the SR as a function of x_D for both approaches. In the probabilistic approach, we use Heuristic-MP-DCP with $N_{Dijkstra} \leq 3$ in CASE-I and with $k = 2$ in CASE-II. The performance of the triggered-based approach depends on TH$_D$; as TH$_D$ decreases, the SR performance improves at the cost of higher advertisement overhead. Hence, we show the performance of the triggered-based approach for three values of TH$_D$. For each value of TH$_D$, we



(a)



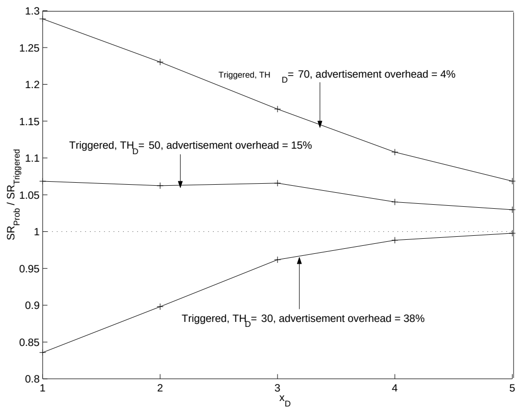
(b)

Fig. 14. Success rate versus x_D for the probabilistic and triggered approaches subject to a delay constraint D: (a) CASE-I where $D = \mu(p*)+x_D\sigma(p*)$; (b) CASE-II where $D = \mu(p*) - x_D\sigma(p*)$.

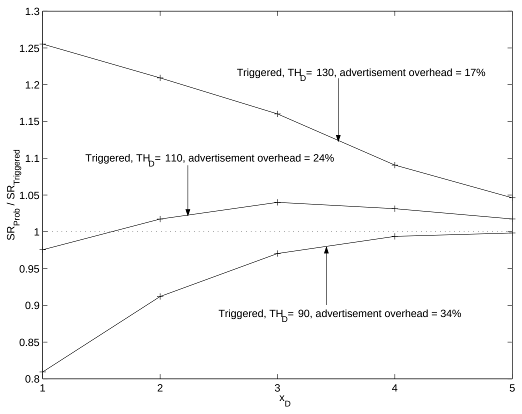
also show the corresponding percentage of links that resulted in triggered advertisements. We refer to this percentage as the *advertisement overhead*. For example, in CASE-I with TH$_D = 70$, 39% of links generated triggered updates. As a point of reference, we also depict the performance of the "exact algorithm," in which the exact link values are available to all nodes in the network (i.e., TH$_D = 0$). A 'failure' for the exact algorithm means that there is no path in the network that satisfies the required delay constraint. As shown in Figure 14(a), in CASE-I the performance of the probabilistic approach surpasses that of a triggered approach with TH$_D = 110$ (20% advertisement overhead). In other words, for a given target SR the use of the probabilistic approach in place of a threshold-based triggered approach eliminates the need to flood the delay values of 20% of the network links. This is a significant reduction in the advertisement overhead. The gain is less pronounced in CASE-II, where the probabilistic approach is shown to have roughly the same SR performance of a triggered-based approach with 4% advertised links. The reason is that in CASE-II, the probability of satisfying the delay constraint is always less than 0.5. So even if minμmaxσ succeeds in returning the optimal path w.r.t. π_D, this path is often infeasible[4].

Ideally, one would expect that for Heuristic-MP-DCP (with $N_{Dijkstra} \leq 3$ in CASE-I and $k = 2$ in CASE-II), π_D in Figure 13 should be the same as the corresponding SR in Figure 14. But apparently there is a slight difference, which can be attributed to approximation errors in the normal distribution and, evidently, the overestimation of π_D. More specifically, π_D in Figure 13 is computed under the assumption that link delays are *exactly normally distributed.* On the other hand, in the simulations used to obtain the SR, link delays are sampled from a normal distribution, which can have both negative and positive values. Whenever a negative value is obtained, we ignore it and repeat the sampling process. In effect, we are using a *truncated* normal distribution for simulating the local delays. So the end-to-end delay is not exactly normal. The same slight discrepancy is observed when link delays are sampled from other distributions (e.g., Weibull and lognormal).

Next, we study the impact of the delay variance, $\sigma^2(i,j)$, on the SR performance. We let $\sigma \overset{\text{def}}{=} \sigma(i,j)$ for all links (i.e., all links have the same delay variance). Figure 15 depicts the ratio $\text{SR}_{\text{Prob}}/\text{SR}_{\text{Triggered}}$ versus x_D for two values of σ. When this ratio is greater than one, the probabilistic approach gives better SR performance than the triggered-based approach. As seen in the figure, when $\sigma = 25$ the probabilistic approach is better than a triggered-based approach with at least 15% overhead; this overhead jumps to 24% when $\sigma = 75$. In conclusion, the reduction in the advertisement overhead achieved by using the probabilistic approach increases with σ, indicating that the gain from the probabilistic approach increases as link parameters become more dynamic.



(a)



(b)

Fig. 15. $\text{SR}_{\text{Prob}}/\text{SR}_{\text{Triggered}}$ versus x_D: (a) $\sigma = 25$; (b) $\sigma = 75$.

B. Performance Evaluation for the MP-BCP Problem

Next, we evaluate the SR of the probabilistic and triggered-based approaches subject to a bandwidth constraint. Recall from Section II that the optimal solution to the MP-BCP problem is provided through a modified version of Dijkstra's algorithm. In here, this solution represents the probabilistic approach. As indicated earlier, the link bandwidth $b(i,j)$ is uniformly distributed in the range $[lb(i,j), ub(i,j)]$, where $lb(i,j)$ and $ub(i,j)$ are themselves sampled randomly *at the beginning of the simulation run.* We select the value of the bandwidth constraint B as follows. Once the $lb(i,j)$'s and $ub(i,j)$'s are generated for all links, for the given source and destination nodes we compute the best paths w.r.t. $lb(i,j)$ and $ub(i,j)$, respectively. Let lb_{opt} and ub_{opt} denote the bandwidths of these two paths. Then B is set to:

$$B = lb_{opt} + x_B(ub_{opt} - lb_{opt})/5, \quad x_B = 0, 1, \ldots, 5$$

Intuitively, as x_B increases the SR decreases. Figure 16 depicts the SR performance as a function of x_B. The probabilistic approach is shown to be at least as good as the triggered-based approach with $\text{TH}_B = 20$ (18% advertisement overhead), i.e., by using the probabilistic approach, one can eliminate 18% of the link advertisements at no loss in the SR.

[4]Note that the optimal path w.r.t. π_D is not necessarily the same path returned by the "exact" (deterministic) algorithm.

C. Performance Evaluation for the MP-BDCP Problem

In this section, we evaluate the performance of the probabilistic approach subject to both bandwidth and delay constraints. Recall that the approximate solution for the MP-BDCP problem (Approx-MP-BDCP) uses the solutions for MP-DCP and MP-BCP to determine a partial set of nearly nondominated paths, whose number depends on the quantization factor ϵ. We let $\epsilon = 0.1$. This results in an average of 3.3 nearly nondominated paths. If any one of these returned paths is feasible according to the actual link values, we count it a 'success'. The constraints B and D are determined as follows:

$$\begin{aligned} B &\sim uniform[lb_{opt}/x_{DB}, (ub_{opt} + lb_{opt})/x_{DB}] \\ D &\sim uniform[\mu(p*), \mu(p*) + x_{DB}\sigma(p*)/2], \end{aligned}$$

where $x_{DB} = 1, 2, 3, 4, 5$. As x_{DB} increases, both constraints become looser, so more paths become feasible w.r.t. both constraints. Figure 17 depicts the performance as a function of x_{DB}. When the constraints are tight ($x_{DB} < 3$), the probabilistic approach gives roughly the same SR performance as

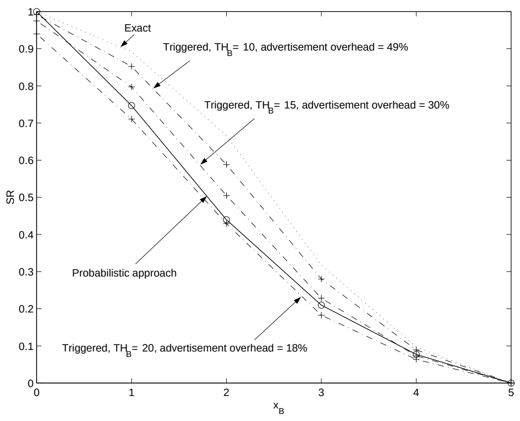


Fig. 16. SR versus x_B subject to a bandwidth constraint $B = lb_{opt} + x_B(ub_{opt} - lb_{opt})/5$.

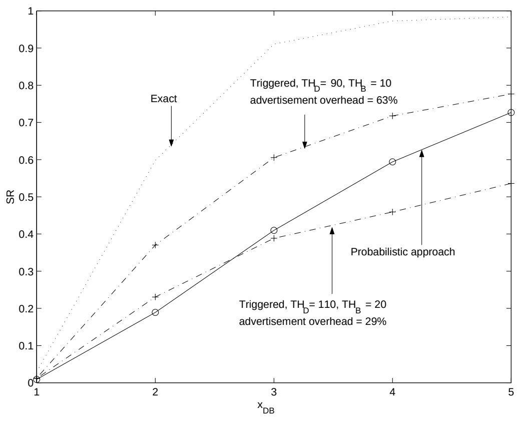


Fig. 17. SR versus x_{DB} under both bandwidth and delay constraints.

a triggered-based approach with 29% advertisement overhead. As the constraints get looser, the gain from the probabilistic approach becomes even more significant (when x_{DB} is close to 5, the reduction in the advertisement overhead is more than 50%). Note that, in general, the reduction in the advertisement overhead is more pronounced when two constraints are considered than when only one constraint is considered. This can be attributed to the fact that the overall variance (i.e., uncertainty) increases with the number of constraints. Recall from Figure 15 that the probabilistic approach is more robust to a higher variance than the triggered-based approach.

VI. Summary and Conclusions

In this paper, we investigated computationally efficient algorithmic solutions for QoS routing subject to both bandwidth and delay constraints and in the presence of link-state inaccuracies. We followed a probabilistic approach in which the bandwidth and delay parameters are modelled as random variables. The problem was then reduced to that of finding the *most probable path to satisfy given bandwidth and delay constraints*; a multi-objective problem that may not have a unique optimal solution. Accordingly, we took a divide-and-conquer approach, whereby the MP-BDCP problem was first divided into the MP-BCP and MP-DCP problems. Exact and approximate solutions for the two problems were provided and later integrated into one approximate solution for the complete MP-BDCP problem. For the MP-BCP problem, an exact polynomial-time solution is readily available through a modified version of Dijkstra's algorithm. As for the (NP-hard) MP-DCP problem, we provided approximate solutions for two different cases of the problem. In the first case ($\mu(p*) \le D$), our solution is given in algorithm minμminσ, which attempts to minimize both the mean and variance of the path delay. Simulations indicate that to find a near-optimal path, minμminσ requires, on average, three runs of Dijkstra's algorithm. For the second case ($\mu(p*) > D$), we gave an approximate solution, called minμmaxσ, which attempts to minimize the mean while maximizing the variance of the delay of the returned path. Simulations indicate that minμmaxσ achieves near-optimal performance using only $k = 2$. The presented solution for the complete MP-BDCP problem iteratively calls our solutions to MP-DCP while quantizing π_B by a factor ϵ. This process results is a set of nearly nondominated paths w.r.t. both π_B and π_D. Decision makers can select one of these paths based on a specific utility function.

Our simulations indicate that for the same SR performance, the proposed probabilistic approach significantly reduces the advertisement overhead incurred in the standard triggered-based advertisement. The eliminated overhead depends on the number and type of constraints as well as whether CASE-I or CASE-II of Heuristic-MP-DCP is being executed. It varies from 5% to about 20% when a single constraint is considered. When two constraints are considered simultaneously, the advertisement overhead can even exceed 50%, giving a compelling reason to adopt the probabilistic approach. We believe that the probabilistic approach can be easily incorporated into the semantics of the OSPF protocol. Our future work will focus on a prototype implementation of this approach within an open-source OSPF implementation (e.g., the implementation at www.zebra.org or Moy's implementation in [41]).

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments. This work was supported by the National Science Foundation under grants ANI 9733143, CCR 9979310, and ANI 0095626, and by the Center for Low Power Electronics at the University of Arizona.

References

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

[2] F.A. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem, "A review of constraint-based routing algorithms," Tech. Rep., 2002, http://wwwtvs.et.tudelft.nl/people/fernando/papers/TRreviewqosalg.pdf.

[3] A. Shaikh, J. Rexford, and K.G. Shin, "Evaluating the impact of stale link state on quality-of-service routing," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 162–176, April 2001.

[4] D. H. Lorenz and A. Orda, "QoS routing in networks with uncertain parameters," *IEEE/ACM Transactions on Networking*, vol. 6, no. 6, pp. 768 –778, Dec. 1998.

[5] J. Moy, "OSPF version 2," Standards Track RFC 2328, IETF, April 1998.

[6] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: A performance perspective," in *Proceedings of the ACM SIGCOMM '98 Conference*, Vancouver, British Columbia, Canada, August-September 1998, pp. 17–28.

[7] The ATM Forum, "Private network-to-network interface specification version 1.0 (PNNI 1.0)," March 1996.

[8] T. Korkmaz and M. Krunz, "Source-oriented topology aggregation with multiple QoS parameters in hierarchical networks," *The ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 10, no. 4, pp. 295–325, Oct. 2000.

[9] G. Apostolopoulos, D. Williams, S. Kamat, A. Guerin, R. Orda, and T. Przygienda, "QoS routing mechanisms and OSPF extensions," RFC 2676, IETF, August 1999.

[10] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, September 1996.

[11] H. Frank, "Shortest paths in probabilistic graphs," *Oper. Res.*, vol. 17, pp. 583–599, 1969.

[12] P. Hansen, "Bicriterion path problems," in *Multiple Criteria Decision Making: Theory and Applications*, G. Fandel and T. Gal, Eds., Lectures Notes in Economics and Mathematical Systems 177, pp. 109–127. Springer, Heidelberg, 1980.

[13] E. L. Lawler, *Combinatorial optimization: networks and matroids*, New York: Holt, Rinehart and Winston, 1976.

[14] R. Guerin and A. Orda, "QoS routing in networks with inaccurate information: Theory and algorithms," *IEEE/ACM Transactions on Networking*, vol. 7, no. 3, pp. 350 –364, June 1999.

[15] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Improving QoS routing performance under inaccurate link state information," in *Proceedings of the 16th International Teletraffic Congress (ITC '16)*, Edinburgh, United Kingdom, June 7-11 1999.

[16] Y. Jia, I. Nikolaidis, and P. Gburzynski, "Multiple path routing in networks with inaccurate link state information," in *Proceedings of the IEEE ICC Conference*. IEEE, 2001, vol. 8, pp. 2583–2587.

[17] W. Jianxin, W. Weiping, C. Jianer, and C. Songqiao, "A randomized QoS routing algorithm on networks with inaccurate link-state information," in *International Conference on Communication Technology (WCC - ICCT 2000)*. IEEE, 2000, vol. 2, pp. 1617–1622.

[18] R. P. Loui, "Optimal paths in graphs with stochastic or multidimensional weights," *Communications of ACM*, vol. 26, no. 9, pp. 670–676, 1983.

[19] M. I. Henig, "The shortest path problem with two objective functions," *European Journal of Operational Research*, vol. 25, no. 2, pp. 281–291, 1986.

[20] P. B. Mirchandani, "Shortest distance and reliability of probabilistic networks," *Comput. & Ops. Res.*, vol. 3, pp. 347–355, 1976.

[21] C. E. Sigal, A. A. B. Pritsker, and Solberg. J. J., "Stochastic shortest route problem," *Operations Research*, vol. 28, no. 5, pp. 1122–1129, Sept.-Oct. 1980.

[22] J. Kamburowski, "A note on the stochastic shortest route problem," *Operations Research*, vol. 33, no. 3, pp. 696–698, May-June 1985.

[23] G. H. Polychronopoulos and J. N. Tsitsiklis, "Stochastic shortest path problems with recourse," *Operations Research Letters*, vol. 10, pp. 329–334, August 1991.

[24] R. A. Sivakumar and R. Batta, "The variance-constrained shortest path problem," *Transportation Science*, vol. 28, no. 4, pp. 309–316, Nov. 1994.

[25] E. D. Miller-Hooks and H. S. Mahmassani, "Least expected time paths in stochastic, time-varying transportation networks," *Transportation Science*, vol. 34, no. 2, pp. 198–215, May 2000.

[26] S. Sen, R. Pillai, S. Joshi, and A. K. Rathi, "A mean-variance model for route guidance in advanced traveler information systems," *Transportation Science*, vol. 35, no. 1, pp. 37–49, Feb. 2001.

[27] R. K. Cheung, "Iterative methods for dynamic stochastic shortest path problems," *Naval Research Logistics*, vol. 45, pp. 769–789, 1998.

[28] A. Eiger, P. B. Mirchandani, and H. Soroush, "Path preferences and optimal paths in probabilistic networks," *Transportation Science*, vol. 19, no. 1, pp. 75–84, February 1985.

[29] P. B. Mirchandani and H. Soroush, "Optimal paths in probabilistic networks: A case with temporal preferences," *Comput. and Operations Research*, vol. 12, no. 4, pp. 365–381, 1985.

[30] I. Murthy and S. Sarkar, "Exact algorithms for the stochastic shortest path problem with a decreasing deadline utility function," *European Journal of Operational Research*, vol. 103, pp. 209–229, 1997.

[31] I. Murthy and S. Sarkar, "Stochastic shortest path problems with piecewise-linear concave utility functions," *Management Science*, vol. 44, no. 11, pp. S125—S136, Nov. 1998.

[32] "Smooth functions," http://www.maths.adelaide.edu.au/pure/mmurray/dg_hons/node4.html.

[33] "Smooth function," http://mathworld.wolfram.com/SmoothFunction.html.

[34] D. E. McDysan, *QoS & traffic management in IP & ATM networks*, McGraw-Hill, 2000.

[35] A. Papoulis, *Probability, random variables, stochastic processes*, McGraw-Hill, third edition, 1991.

[36] T. Elteto and S. Molnar, "On the distribution of round-trip delays in TCP/IP networks," in *The Proceedings of the Local Computer Networks (LCN '99) Conference*. IEEE, 1999, pp. 172–181.

[37] T. Korkmaz, M. Krunz, and S. Tragoudas, "An efficient algorithm for finding a path subject to two additive constraints," in *Proceedings of the ACM SIGMETRICS '00 Conference*, June 2000, vol. 1, pp. 318–327.

[38] E. I. Chong, S. R. Maddila, and S. T. Morley, "On finding single-source single-destination k shortest paths," in *the Seventh International Conference on Computing and Information (ICCI '95)*, July 5-8, 1995, pp. 40–47.

[39] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "Power-laws of the Internet topology," in *Proceedings of the ACM SIGCOMM '99 Conference*, Cambridge, MA, Sept. 1999, pp. 251–262.

[40] "BRITE: Boston university representative internet topology generator," http://www.cs.bu.edu/brite/.

[41] J. T. Moy, *OSPF: Complete Implementation (with CD-ROM)*, Addison Wesley, 2000.

Turgay Korkmaz (S '97 M '01 / ACM M '02) received his Ph.D. degree from Electrical and Computer Engineering at the University of Arizona in December 2001. He is currently an Assistant Professor in Computer Science at the University of Texas at San Antonio. His research interests include QoS-based routing, multi-constrained path selection, efficient dissemination of network-state information, topology aggregation, and performance evaluation of QoS-based routing protocols.

Marwan Krunz is an Associate Professor of Electrical and Computer Engineering at the University of Arizona. His research interests lie in the field of computer networks, especially in its performance and traffic control aspects. His recent work has focused on the provisioning of quality of service (QoS) over wireless links, QoS routing, traffic modeling, bandwidth allocation, video-on-demand systems, and power-aware protocols for ad hoc networks. He has published more than 50 journal articles and refereed conference papers. Dr. Krunz is a recipient of the National Science Foundation CAREER Award (1998-2002). He currently serves on the editorial board for the IEEE/ACM Transactions on Networking, the Computer Communications Journal, and the IEEE Communications Interactive Magazine. He was a Guest Co-editor for a Feature Topic on QoS Routing (IEEE Communications, June 2001) and a Special Issue on Hot Interconnects (IEEE Micro, Jan. 2002). Dr. Krunz was the Technical Program Co-chair for the 9th Hot Interconnects Symposium (Stanford University, August 2001). He has served and continues to serve on the executive and technical program committees of many international conferences. He served as a reviewer and a panelist for NSF proposals, and is a consultant for several corporations in the telecommunications industry.