# QoS Routing in Networks with Uncertain Parameters

**Dean H. Lorenz and Ariel Orda**
Department of Electrical Engineering
Technion—Israel Institute of Technology
Haifa 32000, Israel
{deanh@tx,ariel@ee}.technion.ac.il
Tel.: 972-4-8294-{719,646}
Fax.: 972-4-8323041

**Abstract**

We consider the problem of routing connections with QoS requirements across networks, when the information available for making routing decisions is inaccurate. Such uncertainty about the actual state of a network component arises naturally in a number of different environments. The goal of the route selection process is then to identify a path that is most likely to satisfy the QoS requirements. For end-to-end delay guarantees, this problem is intractable. However, we show that by decomposing the end-to-end constraint into local delay constraints, efficient and tractable solutions can be established. Moreover, we argue that such decomposition better reflects the interoperability between the routing and reservation phases.

We first consider the simpler problem of decomposing the end-to-end constraint into local constraints, for a *given path*. We show that, for general distributions, this problem is also intractable. Nonetheless, by defining a certain class of probability distributions, which includes typical distributions, and restricting ourselves to that class, we are able to establish efficient and exact solutions. We then consider the general problem, of combined path optimization and delay decomposition, and present efficient solutions.

Our findings are applicable also to a broader problem of finding a path that meets QoS requirements at minimal cost, where the cost of each link is some general, increasing function of the QoS requirements from the link.

*Keywords* — Routing, Networks, QoS, Delay, Metric Inaccuracy, Topology Aggregation, QoS-Dependent Costs.

# 1 Introduction

Broadband integrated services networks are expected to support multiple and diverse applications, with various quality of service (QoS) requirements. Accordingly, a key issue in the design of broadband architectures is how to provide the resources in order to meet the requirements of each connection. The establishment of efficient QoS routing schemes is, undoubtedly, one of the major building blocks in such architectures. Indeed, QoS routing has been the subject of several studies and proposals (see, e.g., [2, 4, 5, 7, 10] and references therein). It has been recognized that the establishment of an efficient QoS routing scheme poses several complex challenges. One of the major challenges results from the inherent uncertainty of the information available to the QoS routing process.

As networks grow in size and complexity, full knowledge on network parameters is typically unavailable. Indeed, each single entity in the network cannot be expected to have detailed and instantaneous access to all nodes and links. Routing must therefore rely on partial or approximate information, and still meet the QoS demands. Among the various QoS parameters, the two major ones are bandwidth and end-to-end delay. In the presence of inaccuracies, the former was shown to be polynomially solvable, while the latter poses major obstacles, such as computational intractability [5].

Our model of uncertainty assumes a probability distribution for each link that represents a tradeoff between the QoS guarantees demanded from the link and the probability that the link can meet those demands. This tradeoff is modeled by link costs, which are increasing functions of the QoS requirements. In the framework of parameter uncertainty, the “cost” corresponds to the probability of failure, *i.e.*, not being able to meet the requirements. However, it is important to note that our results hold for a more general case, in which the costs do not necessarily originate from uncertainty. The routing problem is then to establish a connection that satisfies some QoS requirements, at a minimal cost, where the cost function associated with each link increases with the QoS required from it. For simplicity, we will focus on the uncertainty perspective.

In this work we consider end-to-end delay guarantees. We explore the impact of inaccurate network information on the QoS routing process, identify useful and problematic properties in this process and present efficient solutions to the various related problems.

We proceed to discuss the possible origins of uncertainty in network parametery.

## 1.1 Origins of Uncertain Parameters

**Network Dynamics** Many parameters associated with delay requirements are affected by temporal conditions, such as congestion. Parameters advertised by a link might be based, for example, on average behavior or on worst case behavior. In either case the advertised parameters are not accurate. This inaccuracy can be eliminated by rapidly advertising the current, updated, accurate conditions. Unfortunately, this is impractical when the network is highly dynamic and changes are frequent. Thus, advertised values should be considered as uncertain. The precise probability distributions associated with each value depends on *a priori* knowledge on the frequency of updates, and the dynamics of the network.

**Aggregation in Large Networks** In interconnected networks, the sheer growth in information makes it practically impossible to maintain accurate knowledge about all nodes and links. Accordingly, proposals have been made on how to provide the needed information in a scalable form. For example, the ATM Forum PNNI standard [7] introduces a hierarchical process that aggregates information as the network gets more and more remote. However, the aggregation process inherently decreases the accuracy of the information and introduces uncertainty. The semantics of the available parameters depend on the aggregation method used. For instance, we could consider the parameters as averages, or as best or worst cases. Some aggregation schemes may advertise a possible range for each parameter, which may be considered as uniformly distributed within this range. Other schemes may imply different probability distributions, and publish specific parameters associated with these distributions, such as mean and variance.

**Hidden Information** Interconnected networks may include private networks that hide some or all of their information. One reason for this could be hiding the network's internal, proprietary, mechanisms. Typically, such networks would advertise information that contains inaccuracies or advertise ranges for specific parameters. We can interpret this information as probability distributions, based on parameters supplied by these networks, or by prior experience.

A second possible cause for hidden information in subnetworks is to maintain some degree of freedom in internal routing. For each request, the subnetwork is free to choose any internal route that satisfies the QoS requirements. The network may advertise the likelihood of path availability for each QoS requirement, in which case the QoS parameters should be treated as random variables.

**Approximate Calculation** Even the "exact" node and link parameters cannot be assumed to be truly accurate. Typically, they are just approximations of the real parameters and values, since they are based on elaborated models that cannot fully represent the intricacy of the devices. The calculated parameters are usually upper bounds (as in [8]) or incorporate some inaccurate assumptions. Approximate calculation is hence yet another source of uncertainty in the advertised parameters.

## 1.2 Goals of the Paper

Our overall goal is to investigate the impact of uncertain parameters on routing with end-to-end delay guarantees. We assume a framework where the delay guarantees that are advertised by each link are random variables with known distributions. These variables represent the probability that a link can satisfy a QoS delay requirement. We further assume that this knowledge is available to us, and that the delays on the links are independent. [1] A source node is presented with a request to establish a new connection that meets given end-to-end delay requirements. The source node seeks a path that is most likely to satisfy these requirements.

The basis for this work was laid in [5]. That study presented the framework of a network with uncertain parameters, and solved the QoS routing problem for rate demands and for delay demands under a rate-based model, as in [9]. It also presented heuristic methods for dealing with end-to-end delay requirements in models that are not rate-based, and optimal solutions for specific cases. The present study extends that framework, and achieves optimal and $\epsilon$-optimal

[1] This does not necessarily imply that probability distributions are advertised by the nodes; this point is elaborated in Section 2.1.

solutions for the general case.

The rest of the paper is organized as follows. In Section 2 we introduce terminology and definitions, and present different variants of the problem. In Section 3 we consider the simpler problem of decomposing the end-to-end constraint into local constraints, for a *given path*. We show that, for general distributions, this problem is also intractable. Nonetheless, by defining a class of probability distributions, which posses a certain convexity property, and restricting ourselves to that class, we are able to establish efficient and exact solutions. Moreover, we show that typical distributions would belong to that class. We then proceed to consider the combined problem of path selection and constraint decomposition. As a first step, in Section 4 we discuss the *restricted shortest path* problem, which is closely related to our problem. Then, in Section 5, we present a solution to the QoS routing problem, under the assumption that the end-to-end delay is partitioned along the optimal path. In Section 6 we present an efficient $\epsilon$-optimal approximation scheme. Conclusions are presented in Section 7. Due to space limits, many of the proofs and technical details are omitted from this version, and can be found in [6].

# 2 Model and Problems

This section introduces the notations and definitions that are used throughout the paper.

The network topology is known and is represented by a graph $G(V, E)$. There is a single source $s$ and a single destination $t$, and we need to establish a connection with QoS requirements, namely end-to-end delay requirements. We denote by $|\mathbf{p}|$ the number of links in a path $\mathbf{p}$.

## 2.1 Uncertain Parameters

The uncertainty lies in the delay parameter of the links. For each link $l \in E$ we are given a function $f_l(d)$, which is the probability that the link $l$ can guarantee a delay bound $d$. We denote by $\pi_D(\mathbf{p})$ the probability that an end-to-end delay bound $D$ can be guaranteed on the path $\mathbf{p}$. We shall assume that the functions $\{f_l(d)\}_{l\in E}$ are known and that the delays are independent.

These parameters may be (partially) advertised as in any link-state routing scheme. The distributions can be either an *interpretation* of available parameters as probability distributions, or may be deduced from prior experience, or be *a priori* assumptions. For instance, we may consider the delays as uniformly distributed around the advertised value, where the size of the region is determined by the update threshold. The exact mechanisms for obtaining these distributions are out of the scope of this study.

Note that even in a dynamic network, the functions $f_l(d)$ do not change very rapidly, rather the dynamics are incorporated in the probabilities. Hence, using an uncertainty perspective allows less frequent updates which are a key problem when dealing with QoS parameters. Also note that as link parameters reflect the chances of successful connection setup, they might vary with the source, and possibly with the flow specification.

## 2.2 Problems definition

As mentioned, we need to satisfy a given end-to-end delay constraint. Observe that we do not seek a shortest path, but rather a path that is most likely to satisfy our delay constraints. This can be formalized as follows.

**Problem MP (Most Probable path)** *Given an end-to-end delay constraint $D$, find a path $\mathbf{p}^*$, such that for every other path $\mathbf{p}$, $\pi_D(\mathbf{p}^*) \geq \pi_D(\mathbf{p})$.*

It is important to notice that Problem MP is different from finding the path that is most likely to be shortest.[2] Indeed, it is possible that the path $\mathbf{p}^*$ would unlikely be the shortest path despite being the most likely to satisfy our delay constraints.

After an optimal path is selected, a reservation phase takes place. Usually, this involves decomposing the end-to-end delay constraints into local constraints, each imposed on a link along the path. Such a decomposition allows evaluating the delay requirements in term of link resources (e.g. rate [9]) . Thus, the total delay guarantee, $D$, should be partitioned into a set of guarantees $S_D(\mathbf{p}) = \{D_l\}_{l\in\mathbf{p}}$ such that $\Sigma_{l\in\mathbf{p}} D_l = D$.

**Definition 2.1** *Given a path $\mathbf{p}$, and a set of link delay requirements $S_D(\mathbf{p}) = \{D_l\}_{l\in\mathbf{p}}$, define*

$$\pi\left(\{D_l\}_{l\in\mathbf{p}}\right) = \Pr\{d_l \leq D_l \quad \forall l \in \mathbf{p}\} = \prod_{l\in\mathbf{p}} f_l(D_l).$$

[2] When considering $d_l$ as a random variable with a distribution $f_l$.

Different partitions may lead to different probabilities of success, and, as we shall see, finding the best partition (for a given path) is a difficult problem.

**Problem OP (Optimal Partition)** *Given a path* $\mathbf{p}$ *and a delay* $D$*, find a partition* $S_D^*(\mathbf{p}) = \{D_l^*\}_{l \in p}$*, s.t.* $\pi(S_D^*(\mathbf{p})) \geq \pi(S_D(\mathbf{p}))$*, for every (other) partition* $S_D(\mathbf{p}) = \{D_l\}_{l \in p}$.

The assumption that the delay is partitioned imposes a new restriction on the solution of Problem MP. It also changes the probability space, since each event is now determined by the solution to Problem OP. This leads to a revised problem as follows.

**Problem OP-MP (Optimally Partitioned MP)** *Given an end-to-end delay constraint* $D$*, find a path* $\mathbf{p}^*$ *s.t. for every other path* $\mathbf{p}$, $\pi(S_D^*(\mathbf{p}^*)) \geq \pi(S_D^*(\mathbf{p}))$.

Problem OP-MP is identical to Problem MP except that the delay constraint is partitioned. As argued above, partitioning the delay is a requirement that often rises from the actual way in which delay guarantees are requested and provided.

## 2.3 Example

We illustrate the problems through the following example. Consider the network of Figure 1, where the probability distribution of each link is listed.

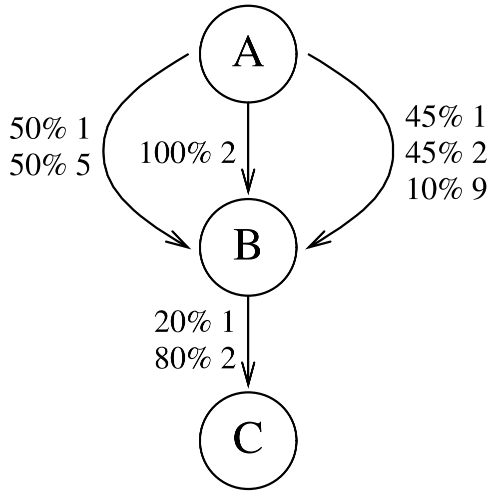


Figure 1: Example

There is a single source $A$ and a single destination $C$. All paths go through $B$ and the routing problem is merely choosing which of the three links will be used to traverse from $A$ to $B$. We will assume that the end-to-end delay bound is $3$.

It can be easily verified that the probability of satisfying the end-to-end bound is $0.5$ for the left link, $0.2$ for the middle link and $0.54$ for the right link. Thus the solution to Problem MP is the path through the right link.

If we assume the delay requirement must be later partitioned into local constraints then we must solve Problem OP for each path. The optimal partition on the right path is $(1, 2)$, which leads to a probability of success of $0.45$. The same partition is optimal for the left path too, leading to a (higher) probability of success of $0.5$. The optimal partition for the middle path is $(2, 1)$ which has a probability of success of only $0.2$. Note that the naive, "equal-partition", solution to Problem OP, $(1.5, 1.5)$, leads to a very poor probability of success of $0.1$ on the left path and even worse on the other paths. Next, observe that the *middle* path has the shortest expected delay yet it is a bad choice for both problems MP and OP-MP. Finally, note that the solution to Problem OP-MP in this case is the left path, while the solution to Problem MP, *i.e.*, the right path, is significantly inferior.

## 2.4 Practical Considerations

In practice, there are further restrictions to the above problems.

One such restriction is on the minimal probability of success, *i.e.*, $\pi\left(S_D\left(\mathbf{p}\right)\right) > p_{\min}$; in other words, we do not consider paths with a probability of success smaller than $p_{min}$. This imposes a restriction on the minimum probability for each link $f_l\left(d\right) > p_0$.[3] Since $f_l\left(d\right)$ is monotonic increasing, we get a restriction on the minimal delay $t_l$ allocated to each link. This means that $f_l\left(t_l\right) > p_0$ implies $D_l \geq t_l$ for each $l \in \mathbf{p}$. Note that this restriction always holds, as $p_0 > 0$. Accordingly, in the following we shall assume that for every link $l$, we have a positive minimal delay $t_l$. Moreover, we shall assume that there is a non zero probability $p_0$, such that for every link $l \in E$, we have $f_l\left(t_l\right) > p_0$, that is, the probability that the link can

[3]Typically, $p_0$ is greater than $p_{\min}$, since the probability of a path is determined by all its links. For instance, if $k$ links have $p_0$ as their probability of success then we must have $p_0 \geq \sqrt[k]{p_{\min}}$.

guarantee a delay of $t_l$ is positive.

A second restriction is regarding the granularity of the delay values. In practice, the delays cannot be broken into arbitrarily small pieces. We denote the smallest possible change in delay by $\delta$, *i.e.*, $\delta$ is the resolution we have for the delay. The smaller $\delta$ is the more accurate the solution is, but generally at the cost of higher complexity in order to solve the problem (or even just to represent the solution). We shall assume, without loss of generality, that $\delta = 1$, and that all delays $(d_l, t_l, D_l, D)$ are integers.

## 2.5 Relation to Shortest Path Algorithms

The probability of success on a path is a product of the probability of success on all links. This is readily transformed to a usual sum by defining a proper cost function. Given a probability distribution $f_l(d_l)$, we define a corresponding cost function $c_l(d_l) \equiv -\log f_l(d_l)$. The cost associated with each link is positive and decreases as the delay allocated to the link increases.

The standard shortest path problem (and consequently its corresponding algorithms) has an *optimal sub-structure* property [1]: a shortest path between two vertices contains within it other shortest paths. This property is a hallmark of the applicability of both *dynamic-programming* and *greedy* methods.

Specifically, the above property means that if $s \rightsquigarrow u \rightsquigarrow v \rightsquigarrow t$ is a shortest path from $s$ to $t$, then $u \rightsquigarrow v$ is a shortest path from $u$ to $v$. Thanks to this property one does not have to check all possible paths, and the complexity reduces from exponential growth to polynomial. Unfortunately, this property *does not* hold in our framework, making it impossible to apply standard shortest path algorithms to our problems. This is illustrated through the example of Section 2.3. Suppose we wish to solve Problem MP through a sub-problem, *i.e.*, the best path from $A$ to $B$. The left link is best if the delay bound, $D_{AB}$, on the segment $AB$ is less than 2 and the middle link is best otherwise. Recall, however, that the *right* link is the optimal solution to Problem MP, yet it is not optimal for *any* choice of $D_{AB}$. Thus, Problem MP does not posses an optimal sub-structure property.

Partitioning the delay, however, introduces a similar, useful property to Problem OP-MP, as follows: if $s \rightsquigarrow u \rightsquigarrow v \rightsquigarrow t$ is a solution of Problem OP-MP with total delay $D$, and it is partitioned into link delay constraints, such that $D^{'}$ is allocated to the sub-path $u \rightsquigarrow v$, then $u \rightsquigarrow v$ is a solution for total delay $D^{'}$ with the same delay partitioning. This enables us in certain circumstances to solve Problem OP-MP by greedy and dynamic programming methods. Going back to the example, we see that the left link is an optimal sub-path for the delay bound $D_{AB} = 1$, which is indeed the optimal delay partition on the segment $AB$.

The solution of Problem OP may also introduce some optimal sub-structure property. This is especially true if we use a predefined (non-optimal) solution of Problem OP. For instance, partitioning the delay with equal probabilities for all links allows us to use standard shortest path solutions, as shown in [5].

## 2.6 General Costs

Guaranteeing QoS requires allocating resources along the path, hence the cost of providing such QoS guarantees corresponds to the cost of reserving such resources. Accordingly, each link may advertise the costs of providing various delay guarantees. A QoS routing mechanism for such a network should choose a path that can satisfy the end-to-end delay requirements at minimal cost.

Hence, the cost function $c_l\left(d_l\right) \equiv -\log f_l\left(d_l\right)$ can be viewed in a broader scope than uncertainty. In other words, all the previously defined problems can be redefined in terms of cost alone without restricting ourselves to costs functions that originate from parameter uncertainty. We only require that the cost is decreasing with the delay. This requirement is very reasonable, since we should pay less if we get a worse QoS bound.

Obviously, such costs provide important tools for resource management. For instance, one may associate higher costs with congested links in order to direct traffic away from them. In other words, less cost may reflect *network* optimality rather than *user* optimality.[4] Our results are applicable for this general cost model as well as for the uncertainty model.

[4]Note, that the network perspective can be incorporated within the uncertainty model, by "distorting" the distributions.

# 3 Solution to Problem OP

In this section we explore Problem OP, and establish an efficient solution for a wide class of probability distributions. Due to space limits, in this version we just outline the results, and the reader is referred to [6] for the full details.

Problem OP can be shown to be NP hard, through a reduction to the *0-1 knapsack problem* [1]. However, by restricting ourselves to a certain class of distributions, exact and efficient solutions can be obtained.

First, we define a family of probability distributions for which we can present efficient solutions. This family consists of distributions with a certain convexity property.

**Definition 3.1** *For $c_l(d_l) \equiv -\log f_l(d_l)$, let $\Psi$ be the set of probability functions $f(d)$ s.t. $\Delta c(d) \equiv c(d) - c(d-1)$ is (strictly) monotonic increasing with $d$.*

$\Psi$ includes all probability distributions with convex cost functions $c(d)$. In [6] we investigate to what extend the assumption $f_l \in \Psi$ limits the choice of $f_l$. We proceed to outline the main conclusions.

It is important to note that the assumption is needed only in the region $(t_l, D)$. In other words, we are only interested in the region $p_0 \leq f_l(d)$. Typically, the minimal probability of success, $p_0$, on each link, should be close to $1$ to allow a reasonable probability of success for the whole path.

**Proposition 3.1** *If a probability distribution $f_l(d)$ is concave, then $f_l \in \Psi$.*

A probability distribution is concave if its density function is monotonic decreasing. This is true for many distributions, given that $p_0$ is large enough. For instance, the exponential distribution is concave for any choice of $p_0$, and the normal distribution is concave for $p_0 \geq 0.5$. In [6], we show that the uniform distribution also belongs to $\Psi$.

Next, we show that any optimal partition is an *equilibrium* point, in the sense that moving a single delay unit from one link to another does not improve the overall probability of success. The following lemma states this property, in terms of the "cost" functions $\{c_l\}_{l\in\mathbf{p}}$, as defined in Section 2.5, and the partition $\{D_l\}_{l\in\mathbf{p}}$.

**Lemma 3.1** *If $\{D_l\}_{l\in\mathbf{p}}$ is an optimal solution to Problem OP then for any $e,l \in \mathbf{p}$ we have $\Delta c_e(D_e) \leq \Delta c_l(D_l+1)$.*

**Proof:** Otherwise we can reduce the total cost by adding 1 to $D_l$ at the expense of $D_e$. The cost will reduce because $((c_l(D_l)+c_e(D_e)) - (c_l(D_l+1)+c_e(D_e-1))) = \Delta c_e(D_e) - \Delta c_l(D_l+1)$. □

Lemma 3.1 leads to an important corollary.

**Corollary 3.1** *If $\{D_l\}_{l\in\mathbf{p}}$ is an optimal solution to Problem OP then there is a threshold $\alpha$ s.t. for all $l \in \mathbf{p}$ we have $\Delta c_l(D_l) \leq \alpha \leq \Delta c_l(D_l+1)$.*

**Proof:** Set $\alpha = \max_{l\in\mathbf{p}} \Delta c_l(D_l)$, and the result follows. □

Focusing on cost functions in $\Psi$, $\Delta c_l$ is monotonic increasing hence $\alpha$ determines $D_l$. We denote the delay allocated to the link $l$ for a specific $\alpha$ by $D_l(\alpha)$, and from the definition of $\alpha$ we have $D_l(\alpha) = \sup\{d : \Delta c_l(d) \leq \alpha\}$. The corresponding delay allocated to the whole path is $D_{\mathbf{p}}(\alpha) = \sum_{l\in\mathbf{p}} D_l(\alpha)$. It can be verified that for distributions in $\Psi$, $S_D(\mathbf{p}) = \{D_l(\alpha)\}$ is a solution to Problem OP for $D = D_{\mathbf{p}}(\alpha)$.

As a consequence of the above result, we can employ a *greedy* scheme (Figure 2). Specifically, we distribute the total delay piece by piece, giving each piece to the link where it most improves the probability of success. Such a strategy involves $D$ iterations, in each of which we select the optimal link and add 1 to its allocated delay.

**Algorithm GREEDY-OP** ($\{f_l\}_{l\in\mathbf{p}}, D$)

1 Let $S_{D^0}(\mathbf{p}) = \{t_l\}_{l\in\mathbf{p}}$, *i.e.*, $D_l^0 = t_l \quad \forall l \in \mathbf{p}$

2 If $D^0 = \sum_{l\in\mathbf{p}} D_l^0 > D$ then **fail**[5]

3 While $D^{i-1} = \sum_{l\in\mathbf{p}} D_l^{i-1} < D$ :

$$S_{D^i}(\mathbf{p}) \leftarrow \begin{cases} D_l^i = D_l^{i-1} & \text{if } l \neq l^i, \\ D_{l^i}^i = D_{l^i}^{i-1} + 1 & \text{otherwise;} \end{cases}$$

where $l^i$ is determined such that $\pi(S_{D^i}(\mathbf{p}))$ is maximal

Figure 2: Algorithm GREEDY-OP

[5] It is impossible to satisfy the minimum delay for each link.

**Theorem 3.1** *If $f_l \in \Psi$ for all $l \in \mathbf{p}$, then Algorithm* GREEDY-OP *solves Problem OP within $O(D \log |\mathbf{p}|)$ steps.*

This result can be further improved by searching for the threshold $\alpha$, *i.e.*, we find an $\alpha$ for which $D_{\mathbf{p}}(\alpha) = D$. We assume that the cost functions have bounded variations ($|\Delta c_l| \leq A$), therefore we must have $\alpha \in (-A, 0)$. We may also assume[6] that changes in $D_l(\alpha)$ are of the same magnitude as changes in $\alpha$, therefore a change of $1$ in $D_{\mathbf{p}}(\alpha)$ requires a change of $|\mathbf{p}|^{-1}$ in $\alpha$.

This means that the *effective* size of our search space is of order $O(|\mathbf{p}| A)$. $D_{\mathbf{p}}(\alpha)$ is a monotonic function, thus we can employ a binary search with $O(\log(|\mathbf{p}| A))$ iterations (Figure 3). Computing $D_{\mathbf{p}}(\alpha)$ requires $O(|\mathbf{p}|)$, hence the search can be implemented in $O(|\mathbf{p}| \log(|\mathbf{p}| A))$.

**Algorithm BINARY-SEARCH-OP** $(\{f_l\}_{l \in \mathbf{p}}, D)$

If $\sum_{l \in \mathbf{p}} t_l > D$ then **fail**
$\hat{D} \leftarrow D - \sum_{l \in \mathbf{p}} t_l, \quad \hat{D}_l \leftarrow t_l + \frac{\hat{D}}{|\mathbf{p}|}$
$L \leftarrow \min_{l \in \mathbf{p}} \Delta c_l \left(\hat{D}_l\right), \quad H \leftarrow 0$
Repeat
    $\alpha \leftarrow \frac{L+H}{2}$
    if $D_{\mathbf{p}}(\alpha) = D$ then **return** $D_l = D_l(\alpha) \quad \forall l \in \mathbf{p}$
    if $D_{\mathbf{p}}(\alpha) > D$ then $H \leftarrow \alpha$ else $L \leftarrow \alpha$

Figure 3: Algorithm BINARY-SEARCH-OP

**Theorem 3.2** *If $f_l \in \Psi$ for all $l \in \mathbf{p}$, then Algorithm* BINARY-SEARCH-OP *solves Problem OP in $O(|\mathbf{p}| \log(|\mathbf{p}| A))$.*

It is possible to solve Problem OP by extracting $\alpha$ directly from the equation $D = D_{\mathbf{p}}(\alpha)$. This can be done numerically in the general case, and analytically for certain distributions. One such distribution is the uniform, which we proceed to discuss.

## 3.1 Uniform Distribution

It can be verified that uniform distributions belong to $\Psi$. We proceed to show that we can get an implicit expression for the optimal partition $D_{\mathbf{p}}(\alpha)$.

[6] See [6] for a more detailed discussion.

**Theorem 3.3** *Let $f_l(d)$ be uniform, $f_l$=$\mathbf{U}(t_l, t_l+\delta_l)$,* i.e.,

$$f_l = \begin{cases} 0 & \text{if} \quad 0 < d < t_l, \\ \frac{1}{\delta_l}(d-t_l) & \text{if} \quad t_l \le d \le t_l+\delta_l, \\ 1 & \text{if} \quad d > t_l+\delta_l; \end{cases}$$

*for every $l \in \mathbf{p}$. Let $\tau$ be a delay s.t. $\sum_{l\in\mathbf{p}} t_l + \sum_{l\in\mathbf{p}} \min(\tau, \delta_l) = D$.*

*Then, $S_D = \{D_l = t_l + \min(\tau, \delta_l)\}_{l\in\mathbf{p}}$ is a solution to Problem OP.*

**Proof:** The derivative, $c_l'(d)$, is given by

$$c_l'(d) = -\frac{f_l'(d)}{f_l(d)} = \begin{cases} 0 & \text{if} \quad 0 < d < t_l, \\ \frac{-1}{d-t_l} & \text{if} \quad t_l < d < t_l+\delta_l, \\ 0 & \text{if} \quad t_l+\delta_l < d; \end{cases}$$

hence for $\alpha = \frac{1}{\tau}$ we get $D_l\left(\frac{1}{\tau}\right) = \sup\left\{d : c_l'(d) \le \frac{1}{\tau}\right\} = t_l + \min(\tau, \delta_l)$. By the assumption on $\tau$ we have, for $\alpha = \tau^{-1}$,

$$D_{\mathbf{p}}(\alpha) = \sum_{l\in\mathbf{p}} D_l\left(\frac{1}{\tau}\right) = \sum_{l\in\mathbf{p}} t_l + \sum_{l\in\mathbf{p}} \min(\tau, \delta_l) = D$$

Since $f_l(d) \in \Psi$ and $D_{\mathbf{p}}(\alpha) = D$ we have that

$$S_D = \left\{D_l(\alpha) = D_l\left(\frac{1}{\tau}\right) = t_l + \min(\tau, \delta_l)\right\}_{l\in\mathbf{p}}$$

is a solution to Problem OP. □

Note that, since $c_l(d)$ is differentiable, we may apply the previous lemmas on $c_l'(d)$ rather than $\Delta c_l(d)$.

Theorem 3.3 leads to an alternative algorithm for solving Problem OP. The idea is to seek the $\tau$ stated in the theorem. From the proof of Theorem 3.3 (the calculation of $c'(d)$), we can realize how we should partition the delay: all links should get exactly the same addition to their minimal requirement, *i.e.*, $D_l - t_l$ is constant. We only have to compensate for the fact that this addition might be greater than $\delta_l$. All excess allocation beyond $\delta_l$ can be evenly distributed among the other links.

Such an algorithm was presented in [5], without the above proof of optimality. The complexity is $O(|\mathbf{p}|)$.

# 4 Restricted Shortest Path Problem

In this section we review the well-known restricted shortest path problem and a (non-standard) variant of its dynamic programming solution. We then discuss the relation between this problem and Problem OP-MP. These observations shall help us solve Problem OP-MP, in the next section.

We begin by formally presenting the problem.

**Problem RSP (Restricted Shortest Path)** *Given a network $G(V, E)$, a delay and a cost for each link $\{d_l, c_l\}_{l \in E}$, and a maximal delay $D$, find a path $\mathbf{p}^*$, such that $\sum_{l \in \mathbf{p}^*} d_l \leq D$ and $\sum_{l \in \mathbf{p}^*} c_l \leq \sum_{l \in \mathbf{p}} c_l$, for any other path $\mathbf{p}$ that satisfies the restriction $\sum_{l \in \mathbf{p}} d_l \leq D$.*

Problem RSP is NP-hard [3], yet has a pseudo-polynomial solution, based on dynamic programming. The version of the algorithm presented in Figure 4 is not standard, but it simplifies our subsequent discussion. At iteration $n$, we calculate the optimal path from $s$ to each vertex $v$, with a delay limit of $n$, and store its cost $\mathcal{C}(n, v)$. The cost, $\mathcal{C}(n, u, v)$, is calculated similarly to $\mathcal{C}(n, v)$, except that the last link on the optimal path to $v$ is assumed to be $(u, v)$. At each iteration, after we have $\mathcal{C}(n, u)$ for each vertex $u$, we find $\mathcal{C}\left(n + d_{(u,v)}, u, v\right)$ for all outgoing links from $u$.

Finding the optimal path involves calculating $\min_u \mathcal{C}(n, u, v)$ and choosing the *parent*, $\theta(n, v)$, of $v$ on the optimal path. We add a loop $(v, v)$, to each vertex, with a delay of $1$ and a zero cost, *i.e.*, $d_{(v,v)} = 1$, $c_{(v,v)} = 0$. This addition exempts us from handling the special case, where $\mathcal{C}(n, v) = \mathcal{C}(n - 1, v)$.

Since $\left| \bigcup_{v \in V} \Theta(v) \right| = |E|$, calculating $\min_{\Theta(v)} \mathcal{C}(n, u, v)$ requires a total of $O(|E|)$ for *all* vertices. At each iteration we go over all links in $O(|E|)$, and the number of iterations is $D$. Thus, the complexity is $O(D|E|)$.

**Algorithm** **DYNAMIC-RSP** $\left(G\left(V,E\right),\left\{d_l,c_l\right\}_{l\in E},D\right)$

$\mathcal{C}\left(0,v\right)\leftarrow\infty\quad\forall v\neq s$
$\mathcal{C}\left(0,s\right)\leftarrow 0$
$\Theta\left(v\right)\leftarrow\left\{u:\left(u,v\right)\in E\right\},\ \theta\left(v,0\right)\leftarrow\text{NIL}\quad\forall v\in V$
$n\leftarrow 0$
Repeat until $n\geq D$
    for all $l=\left(u,v\right)\in E$ do
        $\mathcal{C}\left(n+d_l,u,v\right)\leftarrow\mathcal{C}\left(n,u\right)+c_l$
    $n\leftarrow n+1$
    for all $v\in V$ do
        $\mathcal{C}\left(n,v\right)\leftarrow\min_{\Theta(v)}\mathcal{C}\left(n,u,v\right)^{7}$
        $\theta\left(n,v\right)\leftarrow\arg\min_{\Theta(v)}\mathcal{C}\left(n,u,v\right)$

Figure 4: Algorithm DYNAMIC-RSP

## 4.1 Application to Problem OP-MP

Problem OP-MP can be shown to be NP-hard [6], essentially through a reduction to Problem RSP. The crucial difference between Problem RSP and Problem OP-MP is that we do not have a single pair $\left(d_l,c_l\right)$ for each link $l=\left(u,v\right)$. Rather, we have a complete function $c_l\left(d_l\right)$. This means that we cannot calculate $\mathcal{C}\left(n,u,v\right)$ in $O\left(1\right)$. We must find the minimal among all possibilities for $d_l$ and the corresponding $c_l\left(d_l\right)$. In the worst case, this would mean that at each iteration $m$, $0\leq m\leq n-1$, we recalculate $\mathcal{C}\left(n,u,v\right)$ according to $\mathcal{C}\left(m,u\right)+c_{(u,v)}\left(n-m\right)$. This implies a total complexity of $O\left(D^2\left|E\right|\right)$.

A better bound can be achieved for discrete probability distributions. Assume there is a bound $K$ such that, for each link $l\in E$, there are no more than $K$ possible delays, that is, $f_l\left(d_l\right)$ (and $c_l\left(d_l\right)$) are discrete functions with no more than $K$ points. Under this assumption, calculating $\mathcal{C}\left(n,u,v\right)$ requires finding a minimum among $K$ possibilities and can be done in $O\left(K\right)$, leading to a total complexity of $O\left(DK\left|E\right|\right)$. Alternately, we could replace each link $l$ with a set of $K$ links $\left\{l_k\right\}_{1\leq k\leq K}$, each with a specific delay $d_{l,k}$. We then get a graph $\tilde{G}$ with $|\tilde{E}|=K\left|E\right|$, and the related complexity is again $O(D|\tilde{E}|)=O\left(DK\left|E\right|\right)$.

Yet a more careful analysis of Problem OP-MP enables to considerably reduce the complexity of the solution, for the wide class of (both discrete and continuous) distributions in $\Psi$. This is shown in the next section.

[7] For all *existing* $\mathcal{C}\left(n,u,v\right)$.

# 5 Problem OP-MP

In this section we solve Problem OP-MP using dynamic programming methods. The solution uses a modification of Algorithm DYNAMIC-RSP.

Suppose we have a data structure that contains $\mathcal{C}(n,u,v)$, for all $0 \leq n \leq D$. Clearly, this structure must be updated $D$ times, once for each calculation of $\mathcal{C}(m,u)$, $0 \leq m \leq D-1$. We will denote by $\mathcal{C}^{(m)}(n,u,v)$, the value of $\mathcal{C}(n,u,v)$ after iteration $m$.

For specific $n$ and $m$, where $n > m$, $\mathcal{C}(m,u)$ may affect $\mathcal{C}(n,u,v)$, only if we allocate a delay of $(n-m)$ to the link $(u,v)$. In this case, we define $\mathcal{C}_m(n,u,v)$ to be the calculated $\mathcal{C}(n,u,v)$, that is $\mathcal{C}_m(n,u,v) \equiv \mathcal{C}(m,u) + c_{(u,v)}(n-m)$. We also define $\mathcal{C}_m(n,u,v) \equiv \infty$ for $n \leq m$.

In each iteration we update, for all $n$, $\mathcal{C}^{(m)}(n,u,v) = \min\left\{\mathcal{C}^{(m-1)}(n,u,v), \mathcal{C}_m(n,u,v)\right\}$. Note that $\mathcal{C}(n,u,v)$ cannot change after iteration $n$, and for each $n$, there is some $0 \leq m < n$ for which $\mathcal{C}(n,u,v) = \mathcal{C}_m(n,u,v)$. We will denote this $m$ by $base(n,u,v)$.

We are now ready to present Algorithm DYNAMIC-OP-MP (Figure 5). The algorithm employs a data structure that holds $base(n,u,v)$, and supports three operations: INIT - initialize the structure; UPDATE - update the structure at each iteration; and GET - get the value of $base(n,u,v)$ for a specific $n$. Note that for each vertex, we need to hold not only its parent on the optimal path, but also the delay allocated to the last hop.

The algorithm is essentially the same as Algorithm DYNAMIC-RSP. The main difference is in calculating $\mathcal{C}(n,u,v)$. After each calculation of $\mathcal{C}(n,u)$, we use the UPDATE procedure to update our data structure. At Lines 11-13, we extract the value of $base(n,u,v)$ from our data structure, and calculate $\mathcal{C}(n,u,v)$. We use $base(n,u,v)$ at Line 17 to calculate the delay allocated to the last hop and then store this delay with the parent at Line 18.

With an efficient implementation of $base(n,u,v)$, the time complexity of Algorithm DYNAMIC-OP-MP is $O(|E|\, D \log D)$.

**Algorithm DYNAMIC-OP-MP** $(G(V,E), \{f_l\}_{l\in E}, D)$

$\mathcal{C}(0,v) \leftarrow \infty \quad \forall v \neq s$
$\mathcal{C}(0,s) \leftarrow 0$
$\Theta(v) \leftarrow \{u : (u,v) \in E\}$, $\theta(v,0) \leftarrow$ NIL $\quad \forall v \in V$
for all $l = (u,v) \in E$ do
    INIT$(u,v)$
$n \leftarrow 0$
Repeat until $n \geq D$
    for all $l = (u,v) \in E$ do
        UPDATE$(n,u,v)$
    $n \leftarrow n+1$
    for all $l = (u,v) \in \Theta(v)$ do
        $base(n,u,v) \leftarrow$GET$(n,u,v)$
        $\mathcal{C}(n,u,v) \leftarrow \mathcal{C}_{base(n,u,v)}(n,u,v)$
    for all $v \in V$ do
        $\mathcal{C}(n,v) \leftarrow \min_{\Theta(v)} \mathcal{C}(n,u,v)$ [8]
        $u \leftarrow \arg\min_{\Theta(v)} \mathcal{C}(n,u,v)$
        $d \leftarrow n - base(n,u,v)$
        $\theta(n,v) \leftarrow (u,d)$

Figure 5: Algorithm DYNAMIC-OP-MP

## 5.1 Implementing $base(n,u,v)$

In the general case, maintaining our data structure requires $O(D)$ in each iteration, which results in a total complexity of $O(D^2|E|)$. However, we will show that, for probability distributions that belong to $\Psi$, this can be done in $O(\log D)$, reducing the total complexity of Algorithm DYNAMIC-OP-MP to $O(D \log D |E|)$.

As described above, at each iteration we perform the update

$$\mathcal{C}^{(m)}(n,u,v) = \min\left\{\mathcal{C}^{(m-1)}(n,u,v), \mathcal{C}_m(n,u,v)\right\}$$

for all $n$. Since $0 \leq n \leq D$, we need $O(D)$ for the update, however the next lemma (see Appendix for proof) shows that we can improve upon this for distributions in $\Psi$.

**Lemma 5.1** *If $f_{(u,v)}(d) \in \Psi$, and there exists an $n_0$, for which $\mathcal{C}^{(m)}(n_0,u,v) = \mathcal{C}_m(n_0,u,v)$, then $\mathcal{C}^{(m)}(n,u,v) = \mathcal{C}_m(n,u,v)$ for all $n \geq n_0$.*

Intuitively, Lemma 5.1 means that $\mathcal{C}^{(m-1)}(n,u,v)$ and $\mathcal{C}_m(n,u,v)$ have at most a single intersection point. At each iteration we only need to search for the next intersection point. This

[8]For all *existing* $\mathcal{C}(n,u,v)$.

can be done, using a binary search, in $O(\log D)$. In each update we seek the smallest $n_0$ that satisfies the condition of Lemma 5.1, and set $base(n, u, v) \leftarrow m$, for all $n \geq n_0$.

We implement $base(n, u, v)$ through a balanced binary sort tree. The tree is kept balanced by maintaining a subset of a *full* tree, that is, each level splits the interval exactly by half. We also maintain an in-order linked list of all "live" nodes in the tree. During the search, we create all the nodes on the path leading to $n_0$ and update the linked list. When $n_0$ is found we trim the tree by simply connecting node $n_0$ to node $D$.

Procedure INIT (Figure 6) performs the initialization.

**Procedure INIT** $(u, v)$

1 $tree = tree(u, v)$, $last = last(u, v)$
2 $last \leftarrow 0$
3 $tree[0].base \leftarrow 0$
4 $tree[0].next \leftarrow D$
5 $tree[D].base \leftarrow 0$
6 $tree[D].next \leftarrow D$

Figure 6: Procedure INIT

Line 1 just simplifies the notation. Each tree is initialized with two nodes. Node $D$ is needed because each search for $n_0$ begins by checking if $n_0 < D$. The initial $base$ value assumes the allocation to the link is $D_l = D$. Node $0$ is needed for the beginning of the linked list, that is, our initial $base$ value for each iteration is $0$. The variable $last$ is used by GET and is explained later.

Procedure UPDATE (Figure 7) is the core of the algorithm. The procedure traverses through the search tree, creating new nodes and trimming the tree whenever needed. This procedure is called at each iteration, after $\mathcal{C}(m, u)$ is calculated, for all outgoing links $(u, v)$, from vertex $u$. The search for $n_0$ is done in the *repeat* loop, where $n$ is the currently visited node and $(L, H)$ is the current interval.

**Lemma 5.2** *Procedure* UPDATE *updates the tree in* $O(\log D)$.

**Proof:** We assume the tree is correct before the UPDATE call. At each iteration of the *repeat* loop (Line 5), we compare what can be achieved from the node's $base$ (using $\mathcal{C}(base, u)$) with what can be achieved from $m$ (using $\mathcal{C}(m, u)$). If the current $base$ is better than $m$, we must

**Procedure** UPDATE **$(m, u, v)$**

1 $tree = tree\,(u, v)$
2 if $\mathcal{C}_m\,(D, u, v) > \mathcal{C}_{tree[D].base}\,(D, u, v)$ then **return**[9]
3 $L \leftarrow 0,\ H \leftarrow D$
4 $n \leftarrow D$
5 repeat
6     if $\mathcal{C}_{tree[n].base}\,(n, u, v) \leq \mathcal{C}_m\,(n, u, v)$ then
7         $L \leftarrow n$
8     else
9         $tree[n].base \leftarrow m$
10         $tree\,[n]\,.next \leftarrow H$
11         $H \leftarrow n$
12     if $(H - L) \leq 1$ then **return**
13     $n \leftarrow \left\lfloor \frac{L+H}{2} \right\rfloor$
14     if $tree\,[L]\,.next = H$ then
15         $tree\,[n]\,.next \leftarrow H$
16         $tree\,[L]\,.next \leftarrow n$
17         $tree\,[n]\,.base \leftarrow tree\,[L]\,.base$

Figure 7: Procedure UPDATE

have $n_0 > n$, hence we should keep searching in the right branch, *i.e.*, in the interval $(n, H)$. If $m$ is better, we must have $n_0 < n$, hence we should keep searching in the left branch, *i.e.*, in the interval $(L, n)$. In the latter case, we must also update the $base$ to $m$, and trim the tree, *i.e.*, update $next$ to $H$. Thus, we have established that we search in the right direction, and update the nodes along the way correctly.

We need to show how we create new nodes. We have an indication that $n$ is a new node if $L$ and $H$ are neighbors in the linked list (Line 14). In this case, we insert the node $n$ in the list, and set the default $base$ to the base of $L$. Note that the new node will be examined (and updated if needed) on the next *repeat* iteration.

Each node is updated (and if needed, created) in $O\,(1)$ and the number of nodes is bounded by $D$. Since the tree is balanced, the search is done in $O\,(\log D)$ iterations. □

GET should simply return the value of $tree\,[n]\,.base$, and can be easily implemented by traversing the tree in $O\,(\log D)$. However, if we save the last returned value between calls we can implement GET in $O\,(1)$ as follows (Figure 8).

The value of $last$ represents the closest intersection point before $n$, and is always smaller than $n$. $last$ is initialized to 0 by INIT and is updated at each iteration.

[9] $\mathcal{C}\,(m, v)$ cannot improve $\mathcal{C}\,(n, u, v)$ for any $n$.

**Procedure GET** $(n, u, v)$

```
1  tree = tree(u, v) , last = last(u, v)
2  If tree[last].next ≤ n then last ← tree[last].next
3  return tree[last].base
```

Figure 8: Procedure GET

We are now ready to calculate the complexity of Algorithm DYNAMIC-OP-MP.

**Theorem 5.1** *Algorithm* DYNAMIC-OP-MP *solves Problem OP-MP within* $O(|E| D \log D)$ *steps.*

**Proof:** By Lemmas 5.1 and 5.2, we conclude that we can maintain a data structure for $base(n, u, v)$, that can be updated in $O(\log D)$. Clearly, the INIT procedure can be done in $O(|E|)$ for all links. Each iteration calls UPDATE and GET once for each link in $O(|E|(\log D + 1))$. There are $D$ iterations, thus we get a total time complexity of $O(|E| D \log D)$, as claimed. □

## 5.2 Uniform Distribution

Our solution to Problem OP-MP uses the fact that the delay is partitioned, to establish an optimal substructure property, which allows the use of dynamic programming. However, it does not make use of the partition strategy *per se*. Indeed, the dynamic programming algorithm adds at each iteration a single link to an optimally partitioned path. It is obvious that the delay allocated to that link may result in a non-optimal partition of the whole path. This means that only a specific delay may be allocated to the added link. Thus, we can perform an UPDATE at each iteration in $O(1)$.

For uniform distributions, we can efficiently solve Problem OP and use this result in the solution of OP-MP. By Theorem 3.3, the optimal partition is $S_D(\mathbf{p}) = \{t_l + \min(\tau, \delta_l)\}_{l \in \mathbf{p}}$, where $\tau$ is a delay such that

$$\sum_{l \in \mathbf{p}} t_l + \sum_{l \in \mathbf{p}} \min(\tau, \delta_l) = D.$$

The delay $\tau$ is a common property for all links along the optimal path (with optimal partition). This implies that any optimal sub-path should be continued only with the same $\tau$.

The initial value of $\tau$ is determined by the delay allocated to the first link of each path, hence we must consider all possible delay allocations for any links outgoing from $s$.

The dynamic programming algorithm inspects all possible delay allocations to any link outgoing from $s$, hence all possible values of $\tau$ are considered. We might expect a complexity of $O(|E|\, D)$ for the solution, however this is not the case. The problem arises when $\tau$ is greater than $\delta$ for a link outgoing from $s$. In any optimal solution, the maximal delay allocated to a link is $\delta$, hence an allocation of $\delta$ may correspond to different choices of $\tau$. This means that a path starting with $\delta$ corresponds to several different values of $\tau$ $(D \geq \tau \geq \delta)$ and each UPDATE will require more than $O(1)$.

This problem can be circumnavigated if we assume $\tau \leq \delta_l$, for all $l \in E$,[10] in which case the optimal partition becomes $S_D(\mathbf{p}) = \{t_l + \tau\}_{l \in \mathbf{p}}$. This can be implemented using dynamic programming, by saving $\tau$ for each optimal path, and checking all possible values of $\tau$ for each outgoing link from $s$ [6]. Thus, with the above assumption we can solve Problem OP-MP in $O(|E|\, D)$.

In the following Algorithm UNIFORM-OP-MP (Figure 9), we relax the (unreasonable) assumption $\tau \leq \delta_l$ for all $l \in E$. If $\tau > \delta_l$ for some link, then the cost on that link is zero $(f_l(t_l + \delta_l) = 1)$, however we must allocate delay to the link. When $\tau$ is already known, we can simply allocate $t_l + \min(\delta_l, \tau)$ for the next hop. However, when such a link is the first link in the path, we must determine $\tau$. We should consider all possible values for $\tau$, and in the worst case this could change the complexity to $O(|E|\, D^2)$. To overcome this problem, we must make sure that paths do not start with such links.

If we sort all links by increasing values of $\delta_l$, *i.e.*, $0 \equiv \delta_0 \leq \delta_1 \leq \cdots \leq \delta_{|E|}$, then we can solve the problem $|E|$ times, where at iteration $i$, we assume $\delta_{i-1} \leq \tau < \delta_i$. For all links that have $\delta_l \leq \delta_{i-1}$, we set $d_l = t_l + \delta_l$, with probability $1$. We can now delete all those links from the graph and add their delay to the remaining links. For each remaining link $(u, v)$ we add a new link $(s, v)$, which has the same $\delta$ as $(u, v)$, but with $t_{(s,v)} = t_{(u,v)} + T(u)$, where $T(u)$ is the delay allocated to the links we removed. $T(u)$ is the minimal distance (on those links) from $s$ to $u$ w.r.t. $(t_l + \delta_l)$ and can be computed using a (standard) shortest path algorithm on a

[10] We shall relax this assumption later.

graph consisting of the links we removed.

**Algorithm UNIFORM-OP-MP** ($G, \{f_l = \mathbf{U}(t_l, t_l + \delta_l)\}_{l \in E}, D$)

1 sort the links by increasing values of $\delta_l$

2 for $i = 1$ to $|E|$ do

3 $\quad \tilde{E} \leftarrow \{l \in E : l \leq i\}$

4 $\quad T(V) \leftarrow$ SHORTEST-PATH$\left(G\left(V, \tilde{E}\right), \{t_l + \delta_l\}_{l \in \tilde{E}}\right)$

5 $\quad \bar{E} \leftarrow \left\{(s, v) : (u, v) \in E \setminus \tilde{E}\right\}$

6 $\quad f_{(s,v)} \leftarrow \mathbf{U}\left(t_{(u,v)} + T(u), t_{(u,v)} + T(u) + \delta_{(u,v)}\right)$ [11]

7 $\quad \mathbf{p}_i \leftarrow$ OP-MP$\left(G\left(V, E \cup \bar{E}\right), \{f_l\}_{l \in E \cup \bar{E}}, D\right)$, assuming $\tau \leq \delta_l$, for all $l \in E$

8 choose the best $\mathbf{p}_i$

Figure 9: Algorithm UNIFORM-OP-MP

**Theorem 5.2** *Algorithm* UNIFORM-OP-MP *solves Problem OP-MP, for uniform distributions, within* $O\left(|E|^2 (|V| + D)\right)$ *steps.*

**Proof:** Finding $T(V)$ requires $O(|V||E|)$.[12] In the worst case, the number of links is doubled, hence the complexity of finding each optimal path in Line 7 is $O(|E| D)$. Thus, the complexity of each iteration is $O(|E|(|V| + D))$. We perform $|E|$ such iterations, hence the total complexity is $O\left(|E|^2 (|V| + D)\right)$. □

In practice, it is possible to take advantage of the limits on $\tau$ in each iteration. We may also assume that $|V| = O(D)$, hence the complexity of the Algorithm UNIFORM-OP-MP is $O\left(|E|^2 D\right)$.

# 6 Approximation Scheme for Problem OP-MP

In this section we present an efficient approximation scheme for Problem OP-MP. Our approximation is an adaptation of Fully Polynomial Approximation Schemes (FPAS) for Problem RSP, which are based on dynamic programming principles. We use Algorithm DYNAMIC-OP-MP and detail only the needed adjustments.

[11] There may be parallel links.

[12] E.g. using the Bellman-Ford algorithm.

First, we switch the roles of cost and delay, *i.e.*, we iterate over all possible costs instead of delays. Since we have cost *functions* it is not obvious that this can be done. However, we show that, for probability functions in $\Psi$, it is indeed possible to reverse the functions. Next, we use the minimal allowed probability of success to bound the maximal cost. Finally, we establish a relation between the granularity of the cost and the error and complexity of computation.

## 6.1 Changing $c(d)$ to $d(c)$

Recall that we assumed a minimal probability of success $p_0$ for each link $l$, implying a minimal delay $t_l$ allocated to it (where $f_l(t_l) > p_0$). This means that there is a *maximal* cost, $c_0$, for each link, *i.e.*, $c_0 = -\log(p_0)$. The *maximal* probability of success for a link is bounded by $1$, which implies a minimal cost of $0$. We will denote the minimal delay, which achieves this cost, by $T_l$.

For probability distributions in $\Psi$, the cost functions $c_l(d_l)$ are monotonic non-increasing and convex. This implies that the cost functions are strictly monotonic, hence the inverse function $d_l(c_l)$ exist. It also implies that the inverse functions are monotonic non-increasing and convex. Thus, we may apply Algorithm DYNAMIC-OP-MP to $d_l(c_l)$. Formally we define the inverse function $d_l(c_l)$ as follows.

**Definition 6.1** *Given a probability function $f_l(d_l) \in \Psi_1$, define*

$$\begin{array}{l} d_l(c_l) : (0, c_0) \rightarrow (t_l, T_l) \\ d_l(c_l) \equiv c_l^{-1}(d_l) \end{array} .$$

We shall denote by $\eta$ the equivalent of $\delta$ (Section 2.4) for costs, *i.e.*, we assume from here on that all costs $(c_l, c_0)$ are integer multiples of $\eta$. We will later analyze the impact of $\eta$ on the error and complexity.

**Remark 6.1** *The discretization, $\delta$, of the delay may be regarded as an* a priori *assumption,* i.e.*, the delay demands from each link cannot be given more accurately. We may* choose *to increase $\delta$ to improve performance, but then we have to consider the error due to this discretization. On the other hand, the cost discretization, $\eta$, is* not *inherent to the problem and is always a* *matter of choice. If the delay discretization is inherent, then $d(c)$ must have discrete values both for $c$ and for $d(c)$. This might complicate many of the claims regarding the function $d(c)$ (for instance, it might not be strictly convex). In order to avoid such complications, we shall assume that the discretization $\delta$ is finer than the discretization $\eta$. Specifically, we assume that $|\Delta c_l(d_l)| \leq \eta$, in the region $d_l \in (t_l, T_l)$, for all $l \in E$. The total error due to discretization would depend both on $\delta$ and on $\eta$.*

## 6.2 Application to Algorithm DYNAMIC-OP-MP

The algorithm can be readily used with $d(c)$ instead of $c(d)$. Hence, we should expect a complexity of $O(|E| C_{\max} \log C_{\max})$, where $C_{\max}$ is the maximal total cost. Since there is a bound on the minimal probability of success, $p_{\min}$, we have

$$C_{\max} = \left\lceil \frac{-\log p_{\min}}{\eta} \right\rceil .$$

This result can be refined, as we must change the "stop" condition. In the original algorithm, the condition was $n \geq D$. An equivalent condition would be $n \geq C$, however the *actual* stop condition should be $\mathcal{D}(n, v) \leq D$ for some $v \in V$. We denote by $\hat{C}^*$ the discrete cost of the optimal path, *i.e.*,

$$\hat{C}^* \equiv \left\lceil \frac{-\log \pi\left(S_D^*(\mathbf{p}^*)\right)}{\eta} \right\rceil .$$

The resulting complexity is $O\left(|E| \hat{C}^* \log \hat{C}^*\right)$.

## 6.3 Impact of $\eta$

We now inspect the impact of the cost discretization on the error. We denote by $\mathbf{p}^d$ the discrete path found by the algorithm and by $D_l^d, C_l^d$ the corresponding delays and costs. Similarly we denote the optimal solution by $\mathbf{p}^*, D_l^*, C_l^*$. We use the notation $\hat{C}_l^*$ for the discrete costs corresponding to the optimal costs, $\hat{C}_l^* = \left\lceil \frac{C_l^*}{\eta} \right\rceil$. The relation between the total costs on the path is given by $C^* + |\mathbf{p}^*| \eta \geq \eta \sum_{l \in \mathbf{p}^*} \hat{C}_l^* = \eta \hat{C}^*$.

For each link $l \in \mathbf{p}$, we have $\eta \hat{C}_l^* \geq C_l^*$, hence $d_l\left(\hat{C}_l^*\right) \leq D_l$, therefore $\sum_{l \in \mathbf{p}^*} d_l\left(\hat{C}_l^*\right) \leq D$, which means that $\left\{\hat{C}_l^*\right\}_{l \in \mathbf{p}^*}$ is a legal (non-optimal) solution. Since $\mathbf{p}^d$ is an optimal discrete solution, we must have $\sum_{l \in \mathbf{p}^*} \hat{C}_l^* \geq \sum_{l \in \mathbf{p}^d} C_l^d$, hence $C^* + |\mathbf{p}^*| \eta \geq \eta \sum_{l \in \mathbf{p}^*} \hat{C}_l^* \geq \eta \sum_{l \in \mathbf{p}^*} C_l^d = \eta C^d$. Thus, the difference in actual cost between the optimal solution and the discrete solution is bounded by $\eta C^d - C^* \leq |\mathbf{p}^*| \eta \leq |V| \eta$ and, in terms of the corresponding probabilities of success, we get $\log \frac{\pi^*}{\pi^d} \leq |V| \eta$.

By setting $\eta = \frac{\epsilon}{|V|}$, we get $\log \frac{\pi^*}{\pi^d} \leq \epsilon$, and using $e^\epsilon \leq 1 + 2\epsilon$, for small $\epsilon$, we have $\frac{\pi^*}{\pi^d} \leq 1 + 2\epsilon$, *i.e.*, an $\epsilon$-approximation. The value of $\eta$ affects the order of $C_{\max}$, as $C_{\max} = O\left(\frac{-\log p_{\min}}{\eta}\right)$. We have assumed $p_{\min} = O(1)$, hence $C_{\max} = O\left(\eta^{-1}\right) = O\left(\frac{|V|}{\epsilon}\right)$. Implementing this into the complexity, we get $O\left(|E| C_{\max} \log C_{\max}\right) =$

$$O\left(\frac{|E|\,|V|}{\epsilon} \log \frac{|V|}{\epsilon}\right).$$

Thus, we have achieved an efficient $\epsilon$-approximation scheme which is summarized in the following theorem.

**Theorem 6.1** *Algorithm* DYNAMIC-OP-MP *can be used* [13] *to find ,within* $O\left(\frac{|E||V|}{\epsilon} \log \frac{|V|}{\epsilon}\right)$ *steps, a path* $\mathbf{p}^\epsilon$ *that satisfies*

$$\frac{-\log \pi\left(S_D\left(\mathbf{p}^\epsilon\right)\right)}{-\log \pi\left(S_D^*\left(\mathbf{p}^*\right)\right)} \leq 1 + \epsilon.$$

# 7 Conclusions

This work investigated the effects of uncertain parameters on QoS routing with end-to-end delay requirements. We have defined the routing problem and an important variant: *optimally partitioned most probable path* (OP-MP). We have discussed these problems in the context of shortest path problems and indicated their difficulties.

[13] with the above modifications.

We first focused on the *optimal delay partition problem* (OP). Although Problem OP is intractable in the general case, we established an efficient and exact solution for a wide class of probability distributions, including exponential and normal distributions. Moreover, a further improvement, in term of complexity, was presented for uniform distributions.

Next, we considered Problem OP-MP with the above class of distributions, and established a pseudo-polynomial solution, based on dynamic programming. It is remarkable that changing the parameters of each link, from a specific delay with a specific probability of success, as in Problem RSP, to a complete probability distribution function, as in Problem OP-MP, only adds a factor of $\log D$ to the complexity of the solution. Finally, we established a fully polynomial $\epsilon$-optimal approximation to Problem OP-MP.

Our solutions are applicable to a broader domain than that of uncertain parameters. Specifically, they apply to any link costs that are (convex) increasing function of the quality of service guarantees provided by the link. Such a cost scheme is not only reasonable, but also useful for resource management, as it allows taking network objectives into consideration.

Establishing a connection with QoS guarantees requires the invocation of a QoS routing scheme, and, subsequently, a reservation protocol for setting up the flow. The QoS guarantees are usually provided by each link along the path. This usually requires to decompose the end-to-end QoS requirements into local (link) requirements. While reservation protocols, such as RSVP, provide the mechanisms for signaling, they do not provide a partition policy. Therefore, the QoS routing problem that we considered, OP-MP, is not only more solvable than (the more straightforward) MP but, more importantly, seems to be the *actual* problem that we should solve. Indeed, our algorithms for solving Problem OP-MP return both the optimal path and the optimal partition of QoS requirements along it. We also solve the optimal partition problem, OP, which, as argued above, is relevant independently of the path selection process.

While almost any network control function has to handle some degree of uncertainty, traditional approaches, such as dealing with average values, were often enough to circumvent the problem. However, large-scale broadband networks require more sophisticated solutions. Indeed, the size and architecture of such networks increases the degree of uncertainty, while the stricter service demands complicate the impact of imprecisions. While we believe that our results offer valuable insight towards the construction of a proper analysis and design methodology, it is clear that much is yet to be done and understood.

We are currently working on extending our results to multicast routing. This includes investigating optimal partitions of end-to-end QoS demands on given multicast trees, as well as finding optimal trees.

# APPENDIX

## Proof of Lemma 5.1

$\mathcal{C}_m(n,u,v)$ can be viewed as a (monotonic decreasing) function of $n$. We proceed to show that any two such functions $\mathcal{C}_{m_1}(n,u,v), \mathcal{C}_{m_2}(n,u,v)$, have at most a single intersection point.

**Lemma A.1** *If $l=(u,v)$, $f_l \in \Psi$ and $m_1 > m_2$ and if for some $n_0$, we have $\mathcal{C}_{m_1}(n_0,u,v) < \mathcal{C}_{m_2}(n_0,u,v)$ then for every $n \geq n_0$ we have $\mathcal{C}_{m_1}(n,u,v) < \mathcal{C}_{m_2}(n,u,v)$.*

**Proof:** The assumption $m_1 > m_2$, implies $n-m_1 < n-m_2$. Combining this with the assumption $f_l \in \Psi$ implies $\Delta c_l(n-m_1) < \Delta c_l(n-m_2)$, for all $n \geq n_0$. Hence,

$$\sum_{i=n_0+1}^{n} \Delta c_l(i-m_1) < \sum_{i=n_0+1}^{n} \Delta c_l(i-m_2),$$

implying

$$c_l(n-m_1) - c_l(n_0-m_1) \leq c_l(n-m_2) - c_l(n_0-m_2). \tag{A.1}$$

Observe that

$$\begin{array}{ll} \mathcal{C}_m(n_0,u,v) + (c_l(n-m) - c_l(n_0-m)) & = \\ (\mathcal{C}(m,u) + c_l(n_0-m)) + (c_l(n-m) - c_l(n_0-m)) & = \\ \mathcal{C}(m,u) + c_l(n-m) & = \\ \mathcal{C}_m(n,u,v). & \end{array} \tag{A.2}$$

Now, by considering the assumption $\mathcal{C}_{m_1}(n_0,u,v) < \mathcal{C}_{m_2}(n_0,u,v)$, and adding Equation A.1 using Equation A.2, we get $\mathcal{C}_{m_1}(n,u,v) < \mathcal{C}_{m_2}(n,u,v)$, as required. □

We are now ready to prove Lemma 5.1, which we quote again:

**Lemma 5.1** *If $f_{(u,v)}(d) \in \Psi$, and there exists an $n_0$, for which $\mathcal{C}^{(m)}(n_0,u,v) = \mathcal{C}_m(n_0,u,v)$, then $\mathcal{C}^{(m)}(n,u,v) = \mathcal{C}_m(n,u,v)$ for all $n \geq n_0$.*

**Proof:** By definition,

$$\mathcal{C}^{(m)}(n,u,v) = \min\left\{\mathcal{C}^{(m-1)}(n,u,v), \mathcal{C}_m(n,u,v)\right\} = \min_{0\leq k\leq m} \mathcal{C}_k(n,u,v),$$

therefore we have, for all $0 \leq k \leq m-1$, $\mathcal{C}_m(n_0,u,v) \leq \mathcal{C}_k(n_0,u,v)$. Hence, by Lemma A.1 we have, for all $n > n_0$ and for all $0 \leq k \leq m-1$, $\mathcal{C}_m(n,u,v) \leq \mathcal{C}_k(n,u,v)$, for all $0 \leq k \leq m-1$. Thus, for all $n \geq n_0$, we have

$$\mathcal{C}^{(m)}(n,u,v) = \min_{0\leq k\leq m} \mathcal{C}_k(n,u,v) = \mathcal{C}_m(n,u,v).$$

$\square$

# References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[2] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. Internet Draft, April 1998. work in progress.

[3] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.

[4] R. Guérin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. Internet Draft, April 1997. work in progress.

[5] R. Guérin and A. Orda. QoS-Based Routing in Networks with Inaccurate Information: Theory and Algorithms. In *IEEE INFOCOM'97*, pages 75–83, Kobe, Japan, April 1997.

[6] D. H. Lorenz and A. Orda. QoS Routing in Networks with Uncertain Parameters. Research Report EE Pub. No. 1094, Department of Electrical Engineering, Technion, Haifa, Israel, June 1997. Conference version in Proceedings of IEEE INFOCOM'98, San Francisco, CA, March 1998. May be obtained from "ftp://ftp.technion.ac.il/pub/supported/ee/Network/lor.qosr97.ps".

[7] Private Network-Network Interface Specification v1.0 (PNNI). ATM Forum Technical Committee, March 1996.

[8] A. K. Parekh and R.G. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Multiple Node Case. *IEEE/ACM Transactions on Networking*, 2:137–150, 1994.

[9] S. Shenker, C. Partridge, and R. Guerin. Specification of Guaranteed Quality of Service – RFC No. 2212. Internet RFC, November 1997.

[10] Z. Wang and J. Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE JSAC*, 14(7):1288–1234, September 1996.