Optimal Partition of QoS Requirements with Discrete Cost Functions

Danny Raz, *Member, IEEE* and Yuval Shavitt, *Member, IEEE*

***Abstract*—The future Internet is expected to support applications with quality of service (QoS) requirements. To this end, several mechanisms are suggested in the IETF; the most promising among them is DiffServ. An important problem in this framework is how to partition the QoS requirements of an application along a selected path. The problem which is, in general, NP-complete, was solved for continuous convex cost functions by Lorenz and Orda. This paper concentrates on discrete cost functions, which better model the existing and upcoming mechanisms in the Internet. We present efficient exact and approximated solutions for various conditions of the problem. We also show that although the more complex problem of QoS sensitive routing with discrete cost functions is hard, it has a fully polynomial approximation scheme.**

***Index Terms*—Approximation algorithms, differentiated services (DiffServ), dynamic programming, multicast, quality of service (QoS), restricted shortest path (RSP).**

I. Introduction

THE FUTURE Internet is expected to support applications with quality of service (QoS) requirements. To this end, mechanisms are required to support signaling for connection establishment that include QoS routing and resource allocation. A promising application that is currently being deployed and needs QoS support is IP telephony [4]. To support IP telephony, one needs to guarantee the overall end-to-end delay, in order to allow acceptable service level to the end user.

DiffServ [1], [15] is a technology that is suggested to be used to enable the QoS support over the Internet for applications with QoS constraints like IP telephony. In this framework, routers at the edge of the network mark packets to provide them with a designated priority level or service class. Each type of service has, in our case, a bound on the delay inflicted on packets through the network. Service providers may publish different prices per type of service. An IP telephony call will typically traverse multiple networks, each with its own service classes and pricing scheme. In this environment, we need to find a route that satisfies the end-to-end delay bound requirement, and a partition of the end-to-end requirement along the selected route such that the cost of using the route is minimized. Note that, in many cases, the routing is given by some general best-effort underlying routing protocol such as BGP, and the application may only optimize its cost via partition.

The QoS routing problem is to find a minimal cost path (or a multicast tree) in the network that can support the connection QoS requirements (such as delay). Along the selected path, resources (bandwidth, buffer space) should be optimally allocated to support the required QoS at a minimal cost. The latter can be formulized as an optimization problem for the partition of the end-to-end QoS requirements to local requirements along a path (or a multicast tree).

In general, the partition problem is intractable. The special case where the link cost functions, i.e., the function that describes the cost of allocating a QoS parameter on a link, are continuous convex cost functions was addressed recently by several works. Kodialam and Low [10] dealt with multicast trees for the strongly convex case. Lorenz and Orda [13] presented polynomial algorithms both for trees and paths for weakly convex cost functions and addressed the QoS routing problem [12].

This paper concentrates on discrete cost functions, and presents efficient exact and approximated solutions for various cases. We first show that even the simplest possible discrete case, i.e., two level cost functions, is still intractable. We give an efficient dynamic programming solution for the special case where the QoS parameter domain is integer, but not necessarily convex. We present a sublinear algorithm for the homogeneous convex case—the case where all the cost functions are identical. Both solutions are demonstrated to be easily distributed with low communication and storage complexity. The same techniques are also used to establish similar algorithms for the multicast problem.

For the general discrete cost functions case, we show a simple reduction of the QoS partition and the QoS routing problems to the restricted shortest path problem [8]. Using this reduction, one can easily derive an ϵ-approximation algorithm both for the QoS partition and routing problems in the unicast case. However, this reduction does not apply to the multicast case. Thus, we present a different fully polynomial approximation algorithm for the QoS partition problem that works both for the unicast and multicast cases. Namely, we prove that for any approximation parameter ϵ, our approximation algorithm finds a solution with cost not greater than $1+\epsilon$ times the optimal cost, both for paths and trees. Moreover, we show that our approximation can also solve a more general class of nondiscrete cost functions.

The discrete model used in this work lends itself more easily for practical purposes than its continuous counterpart. For example, as discussed above, in the Internet, *DiffServ* is suggested as a framework for QoS provisioning [1], [15]. In DiffServ, each packet can be classified to one of finitely

Manuscript received October 15, 1999; revised April 15, 2000.

D. Raz is with Bell Laboratories, Lucent Technologies, Holmdel, NJ 07733 USA. He is also with the Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000, Israel (e-mail: danny@cs.technion.ac.il).

Y. Shavitt is with Bell Laboratories, Lucent Technologies, Holmdel, NJ 07733 USA. He is also with the Department of Electrical Engineering - Systems, Tel Aviv University, Tel Aviv 69978, Israel (e-mail: shavitt@ieee.org).

Publisher Item Identifier S 0733-8716(00)09229-5.

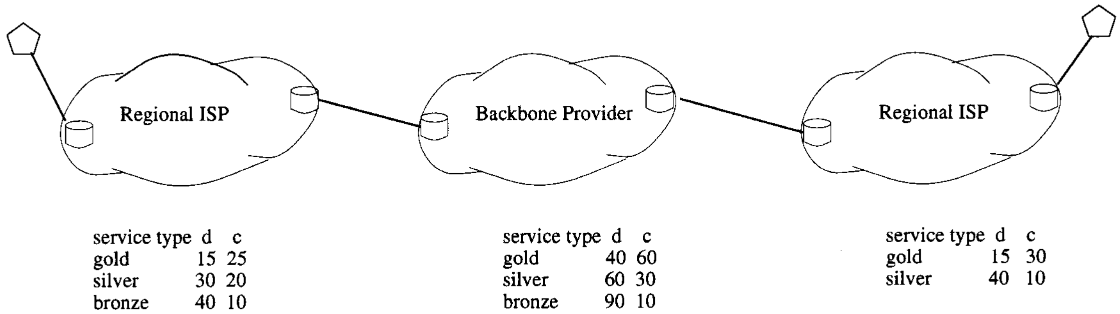


Fig. 1. An example of the use of DiffServ for IP telephony (based on [11]).

many service classes. Each administrative domain,[1] termed differentiated services (DS) domain in [1], determines internally its service classes and defines the QoS associated with each class. This is being published as part of a service level agreement (SLA) which associates a cost to each service class. The administration of the DS domain is responsible for ensuring that adequate resources are provisioned and/or reserved to support the SLA's offered by the domain. For example, an ISP can offer three levels of service above best-effort: gold, silver, and bronze. For any ingress–egress pair, the SLA guarantees a specific delay bound for each service level. The guarantee is achieved by using priority mechanisms, and class based routing in the DS domain.

Consider an application like IP telephony, that requires a delay bound of say 120 ms and traverses three DS domains (see Fig. 1). We need to partition the delay bound requirement among the three DS domains that comprise the path, in a way that results in a minimal cost. The cost and the delay bound for each service level at each DS domain appear below the domain in Fig. 1. A naive approach will partition the delay equally among the three domains. This results, in this example, with a cost of 80 units. An optimal partition, with a cost of only 65 units, is to select the low level service from the backbone (with 90 ms delay bound) and the highest level service (with 15 ms delay bound) in each of the regional ISPs. Even in this simple example, there are multiple choices for the partition. A service provider that can choose the cheapest one has an obvious advantage.

The support of QoS has been the subject of excessive research. The specific aspect of resource allocation in this context has also been excessively studied, in particular, a similar framework was studied by [14], [5], [13], [10]. The reader is referred to [3] for a survey on QoS multicast routing algorithms, although from a slightly different perspective.

The rest of the paper is organized as follows. In the next section we detail our model and define families of discrete cost functions. In Section III we prove that the QoS partition problem is NP-hard. The paper then focuses on the unicast case: in Section IV, we solve the problem for the special case of integer functions; and in the next section we give an approximation algorithm for general discrete cost functions. In Section VI we extend these results to the multicast case. In Section VII we describe an approximation algorithm for the QoS routing problem, and in Section VIII we show that our approximation results also hold for nondiscrete cost functions.

[1]An administrative domain is a subnetwork that is administrated by a single organization, e.g., an ISP or a corporate network.

II. Model

A network is represented by a graph $G(V, E)$, each link $e \in E$ is associated with a discrete cost function c_e: $\mathcal{Q} \rightarrow \boldsymbol{R}$, that assigns a real positive value to each QoS parameter value. To simplify the discussion, we sometimes refer to the QoS parameter as delay.

In the unicast case, a path $\boldsymbol{p}$ of length n between two end nodes is given, and the QoS requirement is additive. Given a bound, $\hat{Q}$, on the end-to-end QoS requirement, the QoS partition problem is to find a vector $X = (x_1, \cdots, x_n)$, s.t., $\sum_{i=1}^{n} x_i \leq \hat{Q}$, and $\sum_{i=1}^{n} c_i(x_i)$ is minimal. Note that, the case of bottleneck QoS requirement is trivial [13], and the multiplicative case can be easily reduced to the additive case by using the logarithm of the requirement [2], [13].

In the multicast case, a multicast tree $\boldsymbol{T}$ is given. The tree has n nodes one of which is designated as the root. The QoS partition problem is to find a vector $X = (x_1, \cdots, x_n)$, s.t., $\sum_{i \in \boldsymbol{p}} x_i \leq \hat{Q}$, for all paths, $\boldsymbol{p}$, from the root to the leaves, and $\sum_{i=1} c_i(x_i)$ is minimal.

The QoS partition problem is called homogeneous if all the links have the same cost function.

A. Discrete Cost Functions

A general discrete cost function associates a cost with each discrete level of QoS. In the most general case, there may be infinitely many discrete QoS levels. We concentrate on the case where link i has l_i QoS levels, $q_{i_1}, \cdots, q_{i_{l_i}}$. In such a case, $c_i = (c_i(q_{i_1}), \cdots, c_i(q_{i_{l_i}}))$. Note that the representation of the discrete cost functions causes the input size to depend on the possible number of QoS levels, Q.

A convenient way to visualize a cost function is to consider a step function where the cost of a QoS parameter q is the cost of the biggest discretely defined QoS parameter $q_{i_j} \leq q$ (see Fig. 2). However, technically the function is defined only for the discrete points q_i, and it is easy to see that an optimum partition is always at these points, since sliding rightwards on a step increases the QoS parameter without decreasing the cost.

Next we define some special cases of cost functions.

Definition 1: A cost function is called *integer* if it is defined only on a finite number of points $q_1, q_2 \cdots, q_l$.

Definition 2: A cost function is called *fully integer* if it is defined on a finite number of consecutive points starting at 1 (or 0) $q_1 = 1, q_2 = 2 \cdots, q_l = l$.

Note that, by scaling, any discrete finite cost function (defined on the rationals) can be translated into an integer function. How-

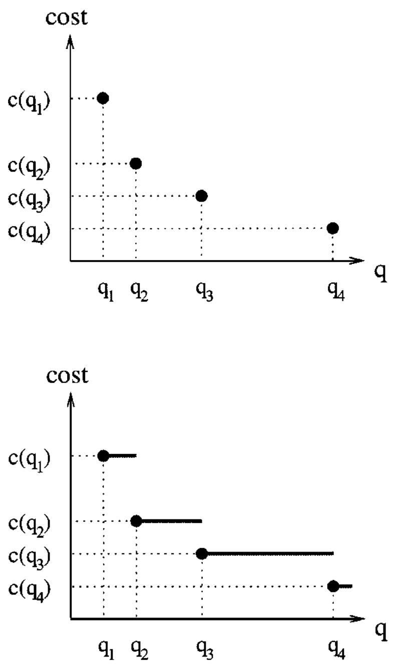


Fig. 2. A discrete cost function, and its representation as a step function.

ever, this may increase the cost of representing a set of functions exponentially, and thus translate a polynomial solution for an integer function to a pseudopolynomial solution.

Definition 3: A cost function c is called *convex* if for every three points $q_i < q_j < q_l$ we have

$$c(q_j) \leq \frac{(q_j - q_i)c(q_i) + (q_l - q_j)c(q_l)}{q_l - q_i}.$$

Definition 4: A cost function c is called *strongly convex* if for every three points $q_i < q_j < q_l$ we have

$$c(q_j) < \frac{(q_j - q_i)c(q_i) + (q_l - q_j)c(q_l)}{q_l - q_i}.$$

The above definition requires that an intermediate point q_j is below the straight line connecting any two points one to its left (q_i) and one to its right (q_l).

III. Hardness Results

In this section we prove that, in general, the QoS partition problem is NP-complete even if the discrete functions are convex. In particular, we show that even if the cost functions are the simplest nontrivial possible, containing only one-step functions, but different for every link, the problem is intractable. A similar but weaker (since it considers a wider class of problems) proof was given in [9].

Lemma 1: Let the cost function for link l be[2]

$$c_l(i) = \begin{cases} a_l & i < a_l \\ 0 & i \geq a_l \end{cases}$$

then determining whether the optimal solution to the QoS partition with limit $\hat{Q}$ for a path of length n is $\hat{Q}$ is equivalent to solving the *subset sum* problem [6, problem SP13] with a set of items $a_1, \cdots, a_n$ and a bound $B = \sum a_i - \hat{Q}$.

Proof: It is easy to show by comparing the problem definitions that the optimal cost of the above QoS partition with limit $\hat{Q}$ for a path of length n is $\hat{Q}$ if and only if there exists a subset $S \subset \{1, 2, \cdots n\}$, with $\sum_{i \in S} a_i = \hat{Q}$. □

We will show in Section V that although the problem is NP-complete, good approximation algorithms can be used to solve it.

IV. Exact Solutions

In this section, we solve the QoS partition problem for integer cost functions. We first present a polynomial dynamic programming algorithm for the general case, and then give a sublinear solution for the homogeneous case.

A. The General Case

In this section we use dynamic programming to solve the QoS partition problem for a collection of integer cost functions. The only requirement is that all these functions can be defined on the same integer scale with no significant increase in their representation. We do not impose any other requirements on the functions, in particular, they need not be convex.

Let $cost(k, d)$ be the optimal cost of partitioning the QoS requirement d along the path $l_1, \cdots, l_k$. Clearly, $cost(k, d)$ can be calculated by the following recursive formula

$$cost(k, d) = \min_{0 \leq i \leq d} cost(k-1, d-i) + c_k(i). \qquad (1)$$

The minimal cost for the partition of requirement $\hat{Q}$ along a path is thus given by calculating $cost(n, \hat{Q})$.

Theorem 1: The complexity of calculating the QoS partition in the case of general integer cost functions is $O(nQ\hat{Q})$, and the memory requirement is $O(n\hat{Q})$. The proof appears in [9, Section 3.3].

B. Convex Cost Functions

For the case where the cost functions are fully integer and (weakly) convex, one can apply the algorithm by Lorenz and Orda [13] to find a solution in $O(n \log n \log \hat{Q})$. Note that every monotone function, and thus every convex function, has at most $\hat{Q}$ different values for q_i, $1 \leq i \leq \hat{Q}$.

In this section, we consider the case where the cost functions are fully integer and convex, but require them all to be identical. We give an optimal algorithm with constant time complexity, i.e., $O(1)$. To this end, we first prove the following lemmas.

Lemma 2: The optimal QoS partition in the homogeneous fully integer strongly convex case results in all the QoS parameters taken from at most two successive values.

Proof: Suppose to the contrary that the lemma does not hold. Then the optimal partition contains, at least, two QoS values,[3] $i < j$, s.t. $i + 1 < j$. By applying Definition 4 twice we get

$$\begin{aligned} &c(i+1) + c(j-1) \\ &\quad < \frac{c(i) + (j-i-1)c(j)}{j-i} + \frac{(j-i-1)c(i) + c(j)}{j-i} \\ &\quad = c(i) + c(j). \end{aligned}$$

□

[2]In our notation, c_l is defined in the points $q_1 = 0$, $q_2 = a_i$, and $c_l(0) = a_l$, $c_l(a_l) = 0$.

[3]Note that for fully integer functions $q_i = i$.

Corollary 1: In the optimal QoS partition in the homogeneous fully integer strongly convex case at least one link is allocated $\lfloor \hat{Q}/n \rfloor$.

Similarly, we can prove the lemma for the weakly convex case:

Lemma 3: There exists an optimal QoS partition in the homogeneous fully integer convex case where all the QoS parameters are taken from at most two successive values.

Corollary 2: There exists an optimal QoS partition in the homogeneous fully integer convex case where at least one link is allocated $\lfloor \hat{Q}/n \rfloor$.

An optimal QoS partition is calculated as follows. $q_i = \lfloor \hat{Q}/n \rfloor$. The number of links that allocated q_i is given by finding the maximal x that solves the inequation $\hat{Q} \geq x q_i + (n-x) q_{i+1}$. Since $c()$ is a fully integer function (assume normalized to the integers) we have $q_{i+1} = q_i + 1$ and thus $x = \lfloor n(q_i+1) - \hat{Q} \rfloor$. $n - x$ links are allocated at q_{i+1}.

C. Distributed Implementation

The dynamic program for the general integer cost functions (Section IV-A) can be easily distributed. Node i along the path can calculate $cost(i, d)$ for $1 \leq d \leq \hat{Q}$, based on the $\hat{Q}$ values passed to it from node $i-1$. When the calculation reaches the end node, it selects its optimal value, and passes back the optimal portion left for the path's prefix. This process continues until it reaches the originator. The total number of messages is only $2n$, while the bit complexity is $O(n\hat{Q})$. The storage requirement at each node is $O(\hat{Q})$. Note that the reservation is done in the reverse direction, thus the origin can start transmission after a two-way handshake. A formal description of the more complex multicast case is given in Section VI-B.

The homogeneous case of Section IV-B can be calculated at the source node since the cost functions are known and equal. Once the two QoS parameter values that should be used are determined, a reservation message with a counter stating how many reservations should be made for each value can propagate along the path toward the destination.

D. Discrete Cost

All the results in this section can be applied to the dual case, where the cost is discrete and one wishes to find the best delay a certain cost can buy. This simple extension is omitted.

V. Approximations

In this section we give a fully polynomial approximation scheme for the QoS partition problem with general discrete cost functions. We assume here that the cost values are all integers. In this case we can rephrase the problem as follows.

Definition 5: (The QoS partition problem with general discrete cost functions.) Given n sets $S_i = \{a_{i_1}, a_{i_2}, \cdots a_{i_{l_i}}\}$ of objects, with specific delays and costs, $delay(a_{i_j}) = d_{i_j} \in Z^+$ and $cost(a_{i_j}) = c_{i_j} \in Z^+$, and a delay bound $D \in Z^+$, find a subset containing n objects, each from a different set, such that their total delay is bounded by D, and the total cost is minimized.

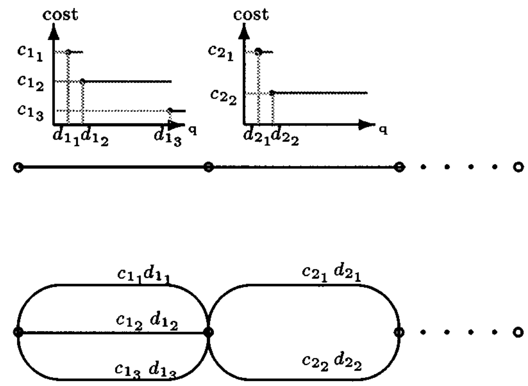


Fig. 3. The reduction of the QoS partition problem for general discrete cost functions to the restricted shortest path problem.

We assume that the cost function is nonincreasing, i.e., for all $i, d_{i_j} > d_{i_{j'}} \rightarrow c_{i_j} \leq c_{i_{j'}}$. Denote by $c_{i_{\max}}$, the maximum cost of any element in the set S_i, and by $d_{i_{\max}}$ the delay associated with this element. Let $c_{\max} = \max_i c_{i_{\max}}$ be the maximum cost overall links. Clearly $\sum_i d_{i_{\max}} \leq D$, otherwise there is no feasible solution. We denote by m the over all number of elements, that is: $m = \sum_{i=1}^{1=n} l_i$, where l_i is the number of elements in the set S_i.

It was already observed by Lorenz and Orda [12] that there is a straightforward reduction from the QoS partition problem with general discrete cost functions to the restricted shortest path problem. Thus one can derive a fully polynomial approximation scheme for the QoS routing problem using Hassin's results [8]. For completeness, we state and prove the following claim in our notation.

Claim 1: Given an instance of the QoS partition (routing) problem with general discrete cost functions, one can construct a bi-criteria[4] graph G' such that the cost of the restricted shortest path problem in G' equals the cost of the QoS partition (routing) problem.

Proof: We replace the ith link by a set of l_i parallel links, each corresponds to a specific working point in the discrete cost function. More formally, given an instance of the QoS partition problem, we build the graph G' with $n+1$ nodes where nodes $i-1$ and i are connected by l_i parallel links, with costs c_{i_j} and delays d_{i_j} (see Fig. 3). Since any simple path from node 0 to node n must choose exactly one of the edges between nodes $i-1$ and i, a path with a delay bounded by D and cost C in G' defines a set with delay bounded by D and cost C in the QoS partition problem. □

Note that the same reduction holds for the QoS routing problem with general discrete cost functions, where the solution for the restricted shortest path problem determines both the links and the appropriate partition (see Section VII).

There are two problems when applying Hassin's algorithm in this way. The first one is that the solution is complex and it is difficult to implement. The more significant problem is that this result does not translate to multicast trees. Thus, the reduction does not hold for the QoS partition problems on trees. To this end, we develop a different algorithm that can be generalized to multicast trees. Note that although Hassin's algorithm cannot

[4]A graph where each edge is associated with both a cost and a delay.

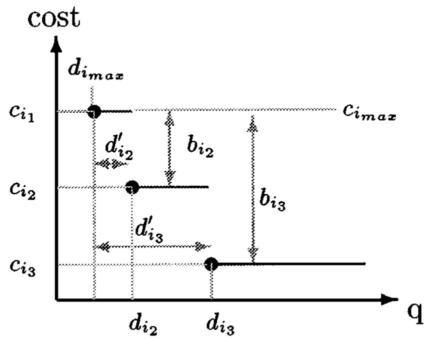


Fig. 4. The transition from the QoS partition problem with general discrete cost functions to the benefit discrete QoS partition problem.

Pseudo Polynomial [PP] $(\{S_i\}_{i=1}^n, d_{i_j}, b_{i_j}, D)$

1. $b_{max} = \max_{i_j} b_{i_j}$.
2. for $p = 1$ to $n \cdot b_{max}$:
3. $A(1,p) = \infty$; if $p = b_{1_j}$ then $A(1,p) = d_{1_j}$.
4. for $i = 2$ to n:
5. for $p = 1$ to $n \cdot b_{max}$
6. $A(i,p) = \min\{\min_{1 \le j \le l_i}\{d_{i_j} + A(i-1, p - b_{i_j})\}, A(i-1,p)\}$.
7. return the largest p such that $A(n,p) \le D$.

Fig. 5. Algorithm pseudoplynomial.

be applied to multicast trees, there might be other dynamic programming algorithms that solve this problem without the benefit formulation we present in this paper.

We begin by defining a variant of the problem, called the benefit discrete QoS partition problem, and proving that it has a fully polynomial approximation algorithm. We then show how to use this algorithm to achieve a fully polynomial approximation scheme for the QoS partition problem with general discrete cost functions.

Definition 6: (The benefit discrete QoS partition problem.) Given n sets $S_i = \{a_{i_1}, a_{i_2}, \cdots a_{i_{l_i}}\}$ of objects, with specific sizes and profits, $size(a_{i_j}) = d_{i_j} \in Z^+$ and $profit(a_{i_j}) = b_{i_j} \in Z^+$, and a delay bound $D \in Z^+$, find a subset of at most n objects, each from a different set, such that their total size is bounded by D, and the total profit is maximized.

The main idea here is that the total profit for a given delay D represents the amount of cost one can save by allowing D more units of delay along the path, starting with any feasible (but maybe costly) initial solution (see Fig. 4). The objective is thus to gain as much savings as possible for every unit of delay. In the example of Fig. 1, the initial costly solution consists of selecting the highest level of service in all the three DS domains which results with an end-to-end delay bound of 70 ms at a cost of 115 units. Reducing the service level by one in the backbone network, for example, results in yet a feasible partition (with delay bound of 90 ms), and saves 30 units.

First we show that the benefit discrete QoS partition problem has a pseudopolynomial algorithm that uses dynamic programming. Then we use this algorithm in order to achieve the polynomial approximation scheme for the QoS partition problem.

Claim 2: Algorithm PP (Fig. 5) is a pseudopolynomial algorithm for the benefit discrete QoS partition problem that works in $O((m+n)nb_{\max})$ time, where $b_{\max} = \max_{i_j} b_{i_j}$.

Benefit Discrete [BD] $(\{S_i\}_{i=1}^n, d_{i_j}, b_{i_j}, \epsilon, D)$

1. $K = \frac{\epsilon b_{max}}{n}$
2. for each a_{i_j}: $b'_{i_j} = \lfloor \frac{b_{i_j}}{K} \rfloor$.
3. $B' = PP(\{S_i\}_{i=1}^n, d_{i_j}, b'i_j, D)$.
4. if $(KB' < b_{max})$ output b_{max}, otherwise output KB'.

Fig. 6. Algorithm benefit discrete.

Proof: Define $A(i,p)$, for $1 \le i \le n$, and $1 \le p \le n \times b_{\max}$, to be the delay of the set with minimal delay that has at most i objects, each from a different set $S_1, \cdots, S_i$, with benefit of exactly p. $A(i,p)$ is ∞ if no such set exists. Clearly,

$$A(i,p) = \min\left\{ \min_{1 \le j \le l_i} \{d_{i_j} + A(i-1, p - b_{i_j})\}, \right. \\ \left. A(i-1,p) \right\}.$$

Now, the largest p such that $A(n,p)$ is smaller than D is the optimal solution for the QoS partition problem. Since we need to compute $n \cdot b_{\max}$ different values, and to compute $A(i,p)$, we need $l_i + 1$ steps. The overall complexity of the algorithm is $O((m+n)nb_{\max})$. □

Next we show how to use this algorithm in order to achieve an ϵ-approximation in polynomial time. We assume that the size of all elements is smaller than the bound D, since elements with bigger sizes cannot be used, and may as well be omitted.

Claim 3: Let B_{BD} be the profit outputted by algorithm benefit discrete (see Fig. 6). Then

$$B_{BD} \ge (1-\epsilon)B_{opt}.$$

Proof: For every element a_{i_j}, the profit considered by the algorithm may be smaller than the actual profit divided by K (as defined in line 1 of Fig. 6), but by no more than 1, i.e., $b'_{i_j} \ge (b_{i_j}/K) - 1$.

Let $profit_K(U)$ be the solution for the instance of U scaled by K. Therefore, for any set of elements U, $profit(U) - Kprofit_K(U) \le nK$. The set S' computed by the PP algorithm must have at least the same profit on the scaled elements as any other set, including the set computed by the optimal algorithm, S^*. Therefore,

$$B_{BD} \ge Kprofit_K(S') \ge Kprofit_K(S^*) \\ \ge profit(S^*) - nK.$$

The first inequality is by line 4 of algorithm BD, and the second inequality is due to the optimality of PP on the scaled elements. Substituting K, we get

$$B_{BD} \ge B_{opt} - \epsilon b_{\max}.$$

Since $B_{opt} \ge b_{\max}$,

$$B_{BD} \ge (1-\epsilon)B_{opt}.$$

□

Since the running time of algorithm benefit discrete is $O((n+m)n\lfloor(b_{\max}/K)\rfloor) = O((n+m)n\lfloor(n/\epsilon)\rfloor)$, Theorem 2 follows.

Theorem 2: Algorithm benefit discrete is a fully polynomial approximation algorithm for the benefit discrete QoS partition problem.

General Discrete [GD] $(\{S_i\}_{i=1}^n, d_{i_j}, c_{i_j}, \epsilon, D)$

$C_{GD} = \sum_i c_{i_{max}}$
repeat as long as none of the sets S_i is empty:
  for each a_{i_j}: $b_{i_j} = c_{i_{max}} - c_{i_j}$; $d'_{i_j} = d_{i_j} - d_{i_{max}}$.
  $C'_{GC} = BD(\{S_i\}_{i=1}^n, d'_{i_j}, b_{i_j}, \epsilon/n, D - \sum_i d_{i_{max}})$.
  if $(\sum_i c_{i_{max}} - C'_{GC} < C_{GD})$ then $C_{GD} = \sum_i c_{i_{max}} - C'_{GC}$.
  remove the element c_{max} from its set.

Fig. 7. Algorithm general discrete.

Now we can describe the algorithm for the QoS partition problem with general discrete cost functions. It works as follows (see Fig. 7). Given an approximation parameter ϵ, we construct a benefit discrete QoS partition problem that basically captures the amount of "saving" one can get starting from an obvious feasible solution. We find an ϵ/n-approximation to this value, and compute the resulting cost. We then iteratively remove the most expensive cost from one of the links and recompute the best cost. At the end, we choose the best cost out of the (polynomially many) costs that we may have. The following claim follows immediately from the definitions.

Claim 4: For any feasible solution U to the benefit discrete QoS partition problem with b_{i_j}, and d'_{i_j}, there exists a set U', which is a feasible solution for the original QoS partition problem with general discrete cost functions such that $profit(U) + cost(U') = \sum_i c_{i_{\max}}$.

This last claim proves that the algorithm finds a feasible set. We have to show both that the cost found by the algorithm is an ϵ-approximation of the optimal cost, and that the algorithm is polynomial.

We already proved that the saving created by the algorithm is almost as good as the saving created by any other algorithm. The problem is that this does not give a full proof for the approximation ratio, as the optimal cost may be much smaller than the total saving. However, if the optimal cost is at least $\sum_i c_{i_{\max}}/n$, we do have an ϵ-approximation since we used BD with $\epsilon' = \epsilon/n$. If the optimal cost is smaller, we will show that the biggest step in one of the links is not used in the optimal solution; hence, we may delete it from the problem and start with a better upper bound. Thus, one of the iterations of the algorithm will find a solution which is an ϵ-approximation of the optimal cost. Since all sets are feasible, and we choose the one with minimal cost, the output of algorithm general discrete is within an ϵ factor of the optimal cost.

For the formal proof we need the following two lemmas.

Lemma 4: If $(C_{opt}/\sum_i c_{i_{\max}}) \geq (1/k)$, and S' is a set with $profit(S') \geq (1-\epsilon/k)B_{opt}$ for the benefit discrete QoS partition problem with $b_{i_j} = c_{i_{\max}} - c_{i_j}$ and $d'_{i_j} = d_{i_j} - d_{i_{\max}}$, then $cost(S') \leq (1+\epsilon)C_{opt}$, for any constant $k > 1$.

Proof: By Claim 4, $cost(S') = \sum_i c_{i_{\max}} - profit(S')$, therefore,

$$cost(S') \leq \sum_i c_{i_{\max}} - (1-\epsilon/k)B_{opt}.$$

Replacing again B_{opt} by $\sum_i c_{i_{\max}} - C_{opt}$, we get

$$cost(S') \leq \sum_i c_{i_{\max}} - (1-\epsilon/k)\left(\sum_i c_{i_{\max}} - C_{opt}\right).$$

Using $\sum_i c_{i_{\max}} \leq kC_{opt}$, we get

$$cost(S') \leq \epsilon C_{opt} + C_{opt}\left(1 - \frac{\epsilon}{k}\right) \leq (1+\epsilon)C_{opt}.$$

□

Lemma 5: If $(C_{opt}/\sum_i c_{i_{\max}}) < (1/n)$, then the element with the maximal cost cannot be in the set that achieves optimal cost.

Proof: Clearly,

$$C_{opt} < \frac{\sum_i c_{i_{\max}}}{n} \leq c_{\max}.$$

□

Now, let O be the set that achieves an optimal cost. Let a_O be the element with the maximal cost in O. Since algorithm GD deletes elements in order according to their cost values, there is an iteration in which a_O is the biggest element. By Lemma 5, in this iteration, $(C_{opt}/\sum_i c_{i_{\max}}) \geq (1/n)$ because the element with the maximal cost is in the optimal set. By Lemma 4, with $k = n$, the cost found in this iteration is bounded by $(1+\epsilon)C_{opt}$. Since the algorithm chooses the best cost overall iteration its output is at least as good.

The running time of algorithm general discrete is bounded by $O(m^2 n(n^2/\epsilon))$ since we run the BD algorithm at most m times, and the complexity of BD is $O(mn\lfloor(b_{\max}/K)\rfloor) = O(mn\lfloor(n/\epsilon')\rfloor)$. However, we can replace the exhaustive search for the best $c_{i_{\max}}$ by a binary search using Lemma 5. If $1 - \epsilon C_G D < 1/n \sum_i c_{i_{\max}}$, we need to reduce $C_{\max}$; and if there is no feasible solution, we went too far and need to increase $C_{\max}$. This complicates the description of the algorithm but reduces the running time to $O(mn^3 \log m/\epsilon)$. Altogether we have proven the following theorem.

Theorem 3: Algorithm GD is a fully polynomial approximation algorithm for the QoS partition problem with general discrete cost functions.

Going back to the example of Fig. 1, $n = 3$, $m = 3 + 3 + 2 = 8$, and for $\epsilon = 1\%$ $(mn^3 \log m/\epsilon) = 64\,800$. In general, the number of DS domains that a connection traverses in the US is almost always below 5 [7]. Assuming a full utilization of the DS field in the IP header enables us to support 256 service classes, which translates to $m = 256 * 5 = 1280$, we get for $\epsilon = 1\%$ $(mn^3 \log m/\epsilon) = 1.6E8$. Although this is well within the computing capabilities of today's hardware, improving the algorithm complexity is an important research direction.

VI. Multicast

A. Exact Solutions

In this section we solve the QoS partitioning problem for a collection of integer cost functions in a multicast tree. As in the unicast case, the only requirement is that all these functions can be defined on the same integer scale with no significant increase in their representation. We do not impose any other requirements on the functions, in particular, they need not be convex. We begin by presenting a polynomial dynamic programming solution.

Let $l = (u, v)$ be a link in the multicast tree such that u is the parent of v, and let $\mathcal{N}_l$ be the group of tree links connected to v except l, namely the group of links leading to v's children. Remember that here n denotes the number of tree links. Let $cost(l, d)$ be the optimal cost of partitioning the QoS requirement d in the subtree of node v and the link l. Clearly, $cost(l, d)$ can be calculated by the following recursive formula

$$cost(l, d) = \min_{i \le d} \sum_{e \in \mathcal{N}_l} cost(e, d-i) + c_l(i) \tag{2}$$

and the minimal cost for the partitioning of requirement $\hat{Q}$ in the tree, $\boldsymbol{T}$, is given by calculating $\sum_{e \in \mathcal{N}} cost(e, \hat{Q})$, where $\mathcal{N}$ are the set of links emanating from the tree root.

Theorem 4: The cost of calculating the QoS partitioning in the case of general integer cost functions in a multicast tree is $O(nQ\hat{Q})$, and the memory requirement is $O(n\hat{Q})$.

Proof: For the calculation we need to keep a table of $cost(l, d)$ where $1 \le l \le n$ and $1 \le d \le \hat{Q}$. This requires a storage of $O(n\hat{Q})$ numbers. The calculation of each entry is done using (2). For each entry, the calculation cost is at most $Q\mathcal{N}_l$, thus the overall calculation complexity is $\sum_{l \in \boldsymbol{T}} \hat{Q}Q\mathcal{N}_l = nQ\hat{Q}$. □

B. Distributed Implementation

The dynamic program for general integer cost functions (Section VI-A) can be easily distributed. The root floods the tree with START messages. The START message carries in each link it traverses the cost function of the immediate upstream parent. A leaf that receives the START message from link l calculates its $cost(l, \cdot)$ entries and sends them on link l to its parent. A node that receives the $cost$ values from all its children calculates its $\hat{Q}$ entries and sends them to its parent. When the root receives the $cost$ calculations from all its children, it initiates the reservation phase by sending a BUDGET message. This message carries the QoS remaining budget downstream. A node that receives the BUDGET message uses the appropriate entry in the cost table it calculated before locating the QoS parameter allocation in its upstream link. It asks its upstream neighbor to allocate this amount using the RESERVE message, and sends a BUDGET message with the remainder of the budget downstream.

The total number of messages is only $4n$, while the bit complexity is $O(n\hat{Q})$. The storage requirement for each link is $O(\hat{Q})$, thus, for a node with $\mathcal{N}$ children the storage requirement is $\mathcal{N}\hat{Q}$. The time complexity is $O(3h)$ where h is the tree hight.

Figs. 8 and 9 give a formal description of the algorithm. Each node has the following variables: p the index of the parent link; $cost(l, \cdot)$ a vector with the optimal cost calculated by the node downstream link l; $cost(\cdot)$ a vector with the results of the local optimal cost calculation; $value(\cdot)$ a vector with the QoS parameter that gives the best cost; and $gotit(\cdot)$ a binary vector to monitor the receipt of $cost$ calculations from the child links. In addition, a node holds a discrete cost function, $c_l()$, for each of its downstream links l, and a list of its downstream links $\mathcal{N}$.

Distributed Multicast Partitioning

For START$(c_l(\cdot))$ from link l
  $p \leftarrow l$
  $c_p(\cdot) \leftarrow c_l(\cdot)$
  foreach $i \in \mathcal{N}_l$
    send START$(c_i(\cdot))$ on link i
    $gotit(i) \leftarrow false$
  if $|\mathcal{N}_p| = 0$
    foreach $j < \hat{Q}$
      $cost(j) = \min_{i<j} c_l(j)$
      $value(j) = \arg\min_{i<j} c_l(j)$
    send $cost(l, \cdot)$ on link l
For $cost(\cdot)$ from link l
  $cost(l, \cdot) \leftarrow cost(\cdot)$
  $gotit(l) \leftarrow true$
  if $\forall i \in \mathcal{N}_p : gotit(i) = true$
    foreach $j < \hat{Q}$
      $cost(j) = \min_{i<j} \sum_{e \in \mathcal{N}_p} cost(e, d-i) + c_p(i)$
      $value(j) = \arg\min_{i<j} \sum_{e \in \mathcal{N}_p} cost(e, d-i) + c_p(i)$
    send $cost(\cdot)$ on link p
For BUDGET(d)
  send RESERVE$(value(d))$ on link p
  foreach $i \in \mathcal{N}_l$
    send BUDGET$(d - c_p(value(d)))$ on link i
For RESERVE(d) from link l
  reserve d delay guarantee on link l

Fig. 8. Distributed multicast partitioning—a nonroot node algorithm.

Distributed Multicast Partitioning

For a request with demand $\hat{Q}$
  foreach $i \in \mathcal{N}$
    send START$(c_i(\cdot))$ on link i
    $gotit(i) \leftarrow false$
For $cost(\cdot)$ from link l
  $cost(l, \hat{Q}) \leftarrow cost(\hat{Q})$
  $gotit(l) \leftarrow true$
  if $\forall i \in \mathcal{N} : gotit(i) = true$
    $cost = \sum_{e \in \mathcal{N}} cost(e, \hat{Q})$
    foreach $i \in \mathcal{N}$
      send BUDGET$(\hat{Q})$ on link i
For RESERVE(d) from link l
  reserve d delay guarantee on link l

Fig. 9. Distributed multicast partitioning—the root algorithm.

C. Approximations

As mentioned before, in the multicast tree case we cannot use the reduction of Claim 1, since it will require finding a restricted shortest tree rather than a restricted shortest path.[5]

Thus, in order to derive a fully polynomial approximation scheme, we follow the steps we took in the path case: starting from a feasible (but possibly costly) partition on the tree, we try to use the extra delay we have in order to save as much cost as possible. We use many of the notations from Section V, and we assume that the multicast tree is a binary tree. It is easy to verify that for any multicast tree $\boldsymbol{T}$, with n nodes, there exists a binary multicast tree $\boldsymbol{T'}$, with at most $2n$ nodes, with exactly the same optimal cost.[6] The amount of saving is expressed in the following definition.

[5]To the best of our knowledge, approximating a restricted shortest tree for graphs with costs and delays is an open problem.

[6]This is done by replacing the outgoing edges at any node that has more than two children, by a binary tree in which these children are the leaves, and all the links to internal nodes have both zero cost and zero delay.

Pseudo Polynomial Multicast [PPM] $(,\{S_i\}_{i=1}^{n}, d_{i_j}, b_{i_j}, D)$

1. $b_{max} = \max_{i_j} b_{i_j}$.
2. for $p = 1$ to nb_{max}:
3.   for all leaves l:
4. $A(l,p) = 0$.
5. for $i = 2$ to n:
6.   for $p = 1$ to nb_{max}
7. $A(i,p) = \min_{0 \le l \le p}\{\max\{\min\{A(i_L,l), \min_{a \in S_{i_L}} A(i_L, l - profit(a)) + d(a)\}, \min\{A(i_R, p-l), \min_{a' \in S_{i_R}} A(i_R, p - l - profit(a')) + d(a')\}\}\}$
8. return the largest p such that $A(n,p) \le D$.

Fig. 10. Algorithm pseudoplynomial multicast.

Definition 7: (The benefit discrete QoS partition problem on trees.) Given a tree $\boldsymbol{T}$, with n nodes, sets $S_i = \{a_{i_1}, a_{i_2}, \cdots a_{i_{l_i}}\}$ of objects, with specific sizes and profits, $size(a_{i_j}) = d_{i_j} \in Z^+$ and $profit(a_{i_j}) = b_{i_j} \in Z^+$, and a delay bound $D \in Z^+$, find a subset of at most n objects, $a_{i_{j_i}}$, each from a different set, such that $\sum_{i \in \boldsymbol{p}} d_{i_{j_i}} \le D$, for all paths, $\boldsymbol{p} \in \boldsymbol{T}$, and $\sum_{i=1} b_{i_{j_i}}$ is maximized.

Next we prove that the benefit discrete QoS partition problem on trees has a pseudopolynomial algorithm that uses dynamic programming. An algorithm similar to the dynamic programming algorithm from Section VI-A is not good enough because it is polynomial in the delay bound D and not the maximal cost $c_{\max}$. In the multicast tree case, the rules of the delay and the cost (benefit) are not symmetric since the cost (benefit) is computed over the entire tree while the delay bound is true for any path in the tree. Recall that we assume that the multicast tree $\boldsymbol{T}$ is a binary tree. We also assume that node n is the root of the tree, and that i_L and i_R are the indices of the left and right children of node i.

Claim 5: Algorithm PPM is a pseudopolynomial algorithm for the benefit discrete QoS partition problem on multicast trees, that works in $O((m+n)nb_{\max}^2)$ time, where $b_{\max} = \max_{i_j} b_{i_j}$.

Proof: Define $A(i, p)$, for $1 \le i \le n$, and $1 \le p \le n \cdot b_{\max}$, to be d iff there exists a QoS partition of the subtree rooted at node i with minimal delay bound d and profit p, and no other partition of this subtree with benefit p has a smaller delay bound. $A(i, p)$ is ∞ if no such partition exists. At node i, one can choose the amount of profit and delay from each of the sets associated with the left and the right children of i. One can also choose not to choose any of these elements. In any case, the relation defined by line 7 of Fig. 10 holds. Therefore, the value of $A(n, p)$, is the best possible delay bound for the given tree $\boldsymbol{T}$ with profit p, and the algorithm outputs the optimal value.

Since the computation of $A(i, p)$ requires at most $b_{\max}(l_i + 1)$ steps, we need to compute it for n values of i and $nb_{\max}$ values of p. The computation complexity of line 7 of Fig. 10 is $n^2 b_{\max}^2 (l_i + 1)$. Thus, the overall complexity of the algorithm is $O((m+n)nb_{\max}^2)$. $\square$

We now apply the rounding technique from Section V to achieve an approximation scheme for this case. We use Algorithm BD with a call to PPM rather than PP (this version is called BDM[7]). This results in an ϵ-approximation algorithm that runs in time $O((m+n)n(\lfloor(b_{\max}/K)\rfloor)^2) = O(mn^3/\epsilon^2)$.

Fig. 11 presents a slightly modified version of the GD algorithms for the multicast trees case. Applying Claim 4 and Lemmas 4 and 5 for GMD, we get the following theorem.

[7]The formal description of BDM is omitted.

General Discrete for Multicast trees [GDM] $(\{S_i\}_{i=1}^{n}, d_{i_j}, c_{i_j}, \epsilon, D)$

$C_{GD} = \sum_i c_{i_{max}}$
repeat as long as none of the sets S_i is empty:
  for each a_{i_j}: $b_{i_j} = c_{i_{max}} - c_{i_j}$; $d'_{i_j} = d_{i_j} - d_{i_{max}}$.
  $C'_{GC} = BDM(\{S_i\}_{i=1}^{n}, d'_{i_j}, b_{i_j}, \epsilon/n, D - \sum_i d_{i_{max}})$.
  if $(\sum_i c_{i_{max}} - C'_{GC} < C_{GD})$ then $C_{GD} = \sum_i c_{i_{max}} - C'_{GC}$.
  remove the element c_{max} from its set.

Fig. 11. Algorithm general discrete for multicast trees.

Theorem 5: Algorithm general discrete multicast is a fully polynomial approximation algorithm for the QoS partition problem on multicast trees with general discrete cost functions. It has time complexity of $O(mn^4 \log m/\epsilon^2)$.

VII. The QoS Requirements Routing Problem

In this section we formally define the QoS requirements routing problem with discrete cost functions. We then show that it is an NP-hard problem and show how to obtain a fully polynomial approximation scheme for this problem.

Definition 8: (The QoS requirements routing problem with discrete cost functions.) Given a graph $G = (V, E)$, each edge is associated with a discrete cost function, and a delay bound D, find a path $\boldsymbol{p}$, and a partition such that the delay along the chosen path is bounded by D, and the cost along this path is the minimal possible.

Clearly this problem is NP-complete since the QoS partition problem with general discrete cost functions is a special case of it (when the graph G is a line). Even the simpler problem of finding the best path without specifying the QoS partition along it is NP-complete. This follows from a straightforward reduction to the restricted shortest path problem [8].

Claim 1 provides a constructive way to construct an instance of the restricted shortest path problem from a given instance of the QoS routing problem with general discrete cost functions. Using it, and the approximation results from [8], we get the following theorem.

Theorem 6: The QoS requirements routing problem with discrete cost functions has a fully polynomial approximation scheme.

VIII. General Cost Functions

In this section we discuss general cost functions. That is, we do not assume any properties such as monotonicity or convexity. Such a function, which is defined for all cost values, is represented by a constant number of bytes. It computes the minimal delay guarantee for a given cost in polynomial time in the representation of the input number. Note that although the discrete functions defined in Section II look similar to the general cost

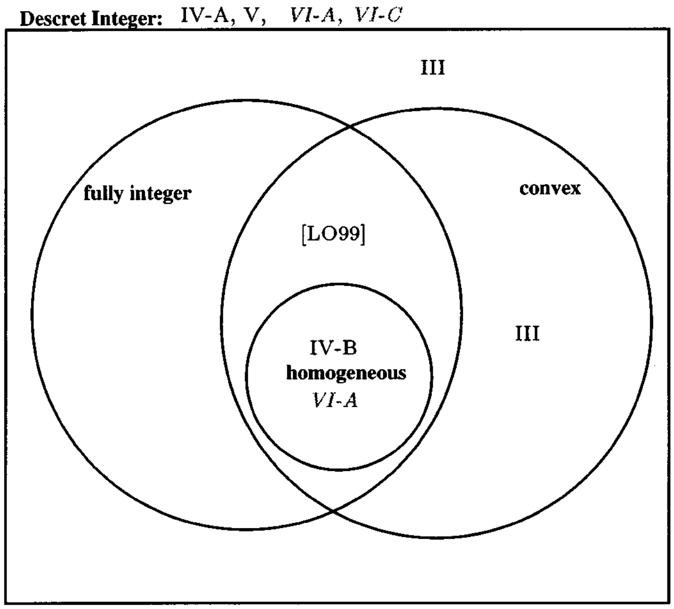


Fig. 12. Graphical view of our results in the problem domain.

functions, the representation of such a function is linear in the number of points and not constant. For example, the function $cost(d) = 1/d$ is defined for every $d > 0$, but if we want to represent it by a discrete function we would have to explicitly give the cost value of each possible delay. To emphasize this point we use here f_i: $Cost \rightarrow Delay$ for the reverse function $cost^{-1}$; thus, for the above example, $f_i^{-1}(d) = 1/d$. For such functions we can prove the following theorem.

Theorem 7: For a set of n functions f_i: $Cost \rightarrow Delay$, such that for each such a function either f_i^{-1}, or a pair $(c_i, d_i = f(c_i))$, with $d_i < D/n$ is given, a graph G, a delay bound D, and an approximation parameter ϵ, one can do the following.

1) Given a path i, j, find an ϵ-approximation algorithm for the QoS partition problem.
2) Given a multicast tree $\boldsymbol{T}$, find an ϵ-approximation for the QoS partition problem on multicast trees.

Proof: First observe that given the reverse function f_i^{-1}, one can compute $f_i^{-1}(D/n) = c_i$, and use the pair $(c_i, D/n)$. These pairs define a feasible solution, both for the path and multicast tree problems with cost bounded by $c_{\max} = \sum_{i=1}^{n} c_i$. Therefore, no optimal solution will use any cost bigger than $c_{\max}$. Observe now that the pseudopolynomial algorithms PP and PPM work also if each step function has $c_{\max}$ steps. Thus, one can compute a discrete function for each link with $l_i = c_{\max}$ points, and therefore the results from Section V (unicast) and Section VI-C (multicast) can be used, with $m = \sum c_{\max} = nc_{\max}$. □

A similar proof can be used to prove an ϵ-approximation scheme for the routing problem with general cost functions. However, it requires either to generalize our results from Section V to the QoS routing problem, or to show that Hassin's algorithm can be used. This is beyond the scope of this paper.

IX. Discussion

In this paper we studied QoS partition and routing problems. We concentrated on discrete cost functions that are both theoretically interesting and have practical applications in IP networks. Fig. 12 depicts graphically the problem domain, where the enclosing rectangle represents the domain of discrete integer cost functions. Each subdomain holds the numbers of the sections where it is discussed (multicast treatment is in italics), sections with results that apply to the entire domain are placed above the rectangle.

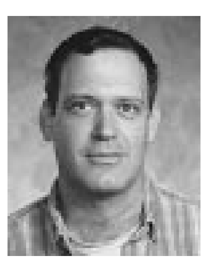
An interesting direction, with possible practical significance, is studying a distributed implementation of the approximation algorithms presented here (in the same spirit of the results of Sections IV-C and VI-B). The main open problem, however, remains the multicast trees routing problem.

Acknowledgment

The authors would like to thank the editors and the reviewers for their useful comments, which improved the paper considerably.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," Internet RFC 2475, Nov. 1998.
[2] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1992.
[3] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high-speed networks: Problems and solutions," *IEEE Network Mag.*, vol. 12, Nov./Dec. 1998.
[4] M. Guizani, A. Rayes, and M. Atiquzzaman, Eds., "Special issue on internet telephony," in *IEEE Commun. Mag.*, Apr. 2000.
[5] V. Firoiu and D. Towsley, "Call admission and resource reservation for multicast sessions," in *Proc. IEEE INFOCOM'96*, Mar. 1996.
[6] M. R. Garey and D. S. Johnson, *Computer & Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H. Freeman, Nov. 1979.
[7] R. Govindan and A. Reddy, "An analysis of internet inter-domain topology and route stability," in *Proc. IEEE INFOCOM'97*, Apr. 1997, pp. 851–858.
[8] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Math.Operations Res.*, vol. 17, no. 1, pp. 36–42, Feb. 1992.
[9] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches*. Cambridge, MA: MIT Press, 1988.
[10] M. Kodialam and S. H. Low, "Resource allocation in a multicast tree," in *Proc. IEEE INFOCOM'99*, Mar. 1999, pp. 262–266.
[11] B. Li, M. Hamdi, D. Jiang, X.-R. Cao, and Y. T. Hou, "QoS-enabled voice support in the next-generation internet: Issues, existing approaches and challenges," *IEEE Commun. Mag.*, vol. 38, pp. 54–61, Apr. 2000.
[12] D. H. Lorenz and A. Orda, "QoS routing on networks with uncertain parameters," *IEEE/ACM Trans. Networking*, vol. 6, pp. 768–778, Dec. 1998.
[13] ——, "Optimal partition of QoS requirements on unicast paths and multicast trees," in *Proc. IEEE INFOCOM'99*, Mar. 1999, pp. 246–253.
[14] R. Nagarajan, J. F. Kurose, and D. Towsley, "Heirat: The heidelberg resource administration technique design philosophy and goals," in *Proc. IFIP Workshop Performance Analysis ATM Systems*, Jan. 1993.
[15] W. Weiss, "QoS with differentiated services," *Bell Labs Tech. J.*, vol. 3, no. 4, pp. 48–62, Oct.–Dec. 1998.



Danny Raz (M'99) received the doctoral degree from the Weizmann Institute of Science, Israel, in 1995.

From 1995 to 1997, he was a Postdoctoral Fellow at the International Computer Science Institute, (ICSI) Berkeley, CA, and a Visiting Lecturer at the University of California, Berkeley. Since October 1997 he has been with the Networking Research center at Bell-Labs, Lucent Technologies. His primary research interest is the theory and application of management related problems in IP networks.

Yuval Shavitt (S'89–M'96) received the B.Sc. degree in computer engineering (cum laude), the M.Sc. degree in electrical engineering, and the D.Sc. degree from the Technion Israel Institute of Technology, Haifa, Israel, in 1986, 1992, and 1996, respectively.

From 1986 to 1991, he served in the Israel Defense Forces, first as a System Engineer and the last two years as a Software Engineering Team Leader. He spent the summer of 1992 as a summer student at IBM T. J. Watson Research Center. After graduation he spent a year as a Postdoctoral Fellow at the Department of Computer Science at Johns Hopkins University, Baltimore, MD. Since 1997, he has been a member of Technical Staff at Bell Labs, Lucent Technologies, Holmdel, NJ. His recent research focuses on active networks and their use in network management QoS routing, and Internet mapping and characterization.