CORC Report 2002-08

Combined Network Design and Multi-Period Pricing: Modeling, Solution Techniques and Computation

Daniel Bienstock[1], Olga Raskina
Dept. of IEOR
Columbia University
New York, NY

Iraj Saniee, Qiong Wang
Bell Laboratories
Lucent Technologies
Murray Hill, NJ

December, 2002 (Version 2003-05-05)

Abstract

In this paper we describe an efficient algorithm for solving novel optimization models arising in the context of multi-period capacity expansion of optical networks. We assume the network operator must make investment decisions over a multi-period planning horizon while facing rapid changes in transmission technology, as evidenced by a steadily decreasing per-unit cost of capacity. We deviate from traditional, and monopolistic, models in which demands are given as input parameters, and the objective is to minimize capacity deployment costs. Instead, we assume the carrier sets end-to-end prices of bandwidth at each period of the planning horizon. These prices determine the demands that are to be met, using a plausible and explicit price-demand relationship; the resulting demands must then be routed, requiring an investment in capacity. The objective of the optimization is now to simultaneously select end-to-end prices of bandwidth and network capacities at each period of the planning horizon, so as to maximize the overall net present value of expanding and operating the network. In the case of typical large-scale optical networks with protection requirements, the resulting optimization problems pose significant challenges to standard optimization techniques. The complexity of the model, its highly nonlinear nature and the large size of realistic problem instances motivates the development of efficient and scalable solution techniques. We show that while general-purpose non-linear solvers are typically not adequate for the task, a specialized decomposition scheme is able to handle large-scale instances of this problem in reasonable time, producing solutions whose net present value is within a small tolerance of the optimum.

[1] Partially funded by NSF awards 29132-5538 (CORC), CCR-0213840, and DMI-0200221.

1 Introduction

In this paper we describe a novel model and develop efficient algorithms for addressing capacity expansion and allocation combined with bandwidth pricing, in the context of designing resilient optical transport networks.

A network is modeled as a set of nodes, representing cities and/or metropolitan areas, which are connected by links, representing physical routes owned by the network operator; transmission systems deployed on links are used to carry traffic. The carrier, or network operator, makes investments for deploying those systems and incurs periodic operating expenses, while collecting revenue from carrying demand for customers.

We consider network planning over a time horizon that spans many years. During such a period several generations of transmission technologies will typically emerge. Though a newer system may have a higher deployment and operating cost, the magnitude of capacity improvement can be expected to far outpace that of cost increase. As a result, a new technology results in a lower cost per-unit of capacity than previous ones, thus making it an attractive candidate for new deployment.

In addition, after a new technology becomes available, its deployment cost decreases over time due to a learning effect. As a result, there may be an economic incentive to delay the deployment of a new technology in order to exploit savings from future cost reductions.

Such a dynamic technology environment immediately poses two interesting problems. First, timing: when should the operator start to deploy the new systems and phase out old technologies. Second, sizing: how much capacity should be deployed on each link at each time period.

Traditional network planning problems have been discussed extensively in the literature, and it would be impractical to provide a thorough review here. See [1], [3], [5], [6], [8], [15], [23], [29] and the references therein. Typically, the critical decisions involve adding capacity, at minimum cost, to existing capacity levels; so that one can route known demands (fully or partly, with a penalty for shortfalls). [21], [28] consider models in which demands are imperfectly known.

While such models rationalize decision-making from an engineering perspective, they reflect a monopolistic approach to decision-making under price regulation. Consequently more comprehensive approaches have been proposed in recent studies to integrate capacity planning into the overall business optimization strategy, in particular in the context of optical networks [24].

In [24], instead of assuming fixed forecast demands between pairs of nodes, an explicit nonlinear price-demand relationship is taken into consideration and demands are treated as flexible quantities that are determined by prices. Prices, which must be chosen for every pair of nodes, are critical decision variables, playing two roles: they determine the revenue for each unit of service rendered, and through the price-demand relationship they affect the amount of deployed capacity, and thus, incurred deployment cost. Consequently, capacity planning is driven not by the need to meet a fixed demand target, but by the desire to generate higher profits. In summary, the objective of the optimization model is to simultaneously choose network capacities and prices of network services, so as to maximize the overall net present value of operating the network; i.e. we want to maximize the discounted total revenue minus total cost over the planing horizon. An important qualitative observation in [24] is that, as the price-elasticity of demand grows larger than 1, the observed (optimal) capacities become large.

The study in [24], while innovative, incorporated several limitations: the networks that were considered were of small size and were simple, the number of time-periods in the model was small, and the underlying network design problem was simplified. We expand on these limitations next.

First of all, in [24] the planning model was limited to a five-node ring network, over five time periods. The small size of the network and small number of time periods obviated the need for truly effective optimization algorithms. We note that in the case of large-scale optical networks (e.g. national networks, quite likely entailing hundreds of nodes), the optimization problem poses significant computational challenges. In comparison with standard minimum-cost capacity planning models, the new approach introduces a price variable and a demand variable, for each period for each node-pair; thus the total number of price (and demand variables) is on the order of N^2T. Here T is the number of planning periods, while N is the number of nodes in the network. In addition, each pricing variable appears in a nonlinear term in the objective function. As a result, when planning a node network with possibly hundreds of nodes, typical for national or international optical transport networks, over a 10-period horizon, we obtain an optimization model that could have even millions of variables that explicitly appear in the nonlinear component of the objective.

Another modeling component of current interest, not considered in [24], is that of survivability. In the context of traditional network design problems, survivability was considered in [2], [8], [7], [29], and others. Survivability modeling seeks to ensure reliable communications in the presence of possible failures. When considering optical transport networks, one way to achieve this end is to have multiple non-overlapping paths between every node pair, with enough capacity allocated so that recovery from a link failure is guaranteed. Several specific routing and protection schemes that need different formulations will be considered in the following Sections. In terms of our optimization model, the survivability feature calls for additional variables used to specify the rerouting of traffic when a network failure occurs. Formulations of some capacity efficient survivability schemes, such as shared protection, require the introduction of new flow variables whose number is again on the order of N^2T.

A third feature not addressed in [24], and implicit in the prior paragraph, concerns routing schemes – in the case of a ring networks, routing is trivial. Typically, standard network design models have assumed that any path may be used to route traffic (for exceptions, see [9], [10]). In realistic settings, however, one cannot route using arbitrarily paths – for example, allowable paths are typically limited in length, or they may be constrained by an underlying routing protocol. In general, the decision as to which paths can be used may be defined somewhat idiosyncratically, and for optimum flexibility in the design of an algorithm, it should be assumed that an allowable path family is given as an input to the problem. The restriction to a (potentially large) given path family will almost certainly increase the difficulty of the model.

Clearly, a decision-support tool that could handle large, complex networks over many time periods, while incorporating survivability and routing features, and, of course, the price/demand model, would be of great strategic importance – it would serve as a means to forecast future demand trends, and to ascertain the interplay between different pride/demand models and estimations of future per-unit capacity costs.

In this paper we present the development of such a tool. First, we extend the model in [24] so as to incorporate all the features we described above. Second, we develop and test an optimization algorithm geared for large instances. Our algorithm is able to handle very large problems endowed with all of the above modeling features in reasonable time, while providing feasible solutions whose net present value is guaranteed to be within a small tolerance of the optimum. Here we stress once more that the optimization problems we tackle are truly difficult. As we will show, general-purpose commercial solvers prove unequal to the task. Finally, our solution techniques are *robust* in the sense that they can handle a broad range of models without modification.

The rest of the paper is organized as following. In Section 2 we present a detailed description of our overall problem. In Section 3, we present a mathematical formulation of the optimization model. Our solution techniques are presented in Section 4 and Section 5 presents computational experiments.

2 Modeling

In this section we discuss several key concepts and assumptions of the pricing and capacity planning model presented in this paper. The scope of our study is outlined in Figure 1 and is explained as follows. First, the model we use differs fundamentally from the norm in that demand is not given in advance. Rather, based on a given price-demand relationship, we make pricing decisions so as to generate corresponding amounts of demands and, as a result, revenues. Next, routing and protection decisions relate generated demands to link capacity requirements. These requirements are satisfied by the deployment of transmission systems, whose availability, cost, and capacity are pre-specified as a technology "roadmap," the principal ingredient of which is that it models (per unit) capacity costs that decrease over time.

Thus, pricing decisions and capacity planning are interrelated, as installed capacity must be sufficient to route the generated demands; and the issues of pricing, routing/protection, and capacity planning are embedded in the overall optimization framework, whose objective is to maximize the net present value of the total cash flow of all planning periods.

We note that the general topic of combining resource allocation with revenue management has been taken up in other problem settings, notably in supply-chain management; our price-demand relationship described below (and close variants) has been studied in that context. See [13], [11] and references therein for recent results.

In the rest of this section we discuss the precise details of our models.

Throughout this paper, we will model the network by a graph $G = (N, L)$, where N is the set of nodes, and L is the set of physical links (rights-of-way). Pairs of nodes in the network are denoted by $\sigma \in N \times N$. We also denote the length of the planning horizon by T, and index each period by $t = 1, \cdots, T$.

Figure 1: Model framework

We will first describe the individual components of our model; a complete formulation that puts together these components is given in Section 3.

2.1 Price-Demand Relationship

We first introduce the notion of the price elasticity of demand.

Let d_σ^t be the demand between node-pair σ in period t, which relies on price per unit of demand p_σ^t. The price-elasticity of demand is defined as the negative ratio of the percentage change in demand to the percentage change in price, i.e.,

$$\epsilon_\sigma^t = -\frac{\partial d_\sigma^t}{\partial p_\sigma^t}\frac{p_\sigma^t}{d_\sigma^t}$$

Following [24], we assume ϵ_σ^t is constant and solve the above differential equation to arrive at the following price-demand relationship,

$$d_\sigma^t = A_\sigma^t (p_\sigma^t)^{-\epsilon_\sigma^t} \tag{1}$$

The revenue R_σ^t collected from node-pair σ in period t is then

$$R_\sigma^t = p_\sigma^t d_\sigma^t = (A_\sigma^t)^{1/\epsilon_\sigma^t}(d_\sigma^t)^{1-1/\epsilon_\sigma^t} \tag{2}$$

In this paper we assume $\epsilon_\sigma^t > 1$ for all σ and t. Therefore, R_σ^t is an increasing and concave function of d_σ^t, i.e., the carrier always receives more revenue by carrying more demand, but the slope of revenue growth declines as demand volume increases.

In our model, d_σ^t will be a decision variable, with price implicitly determined through the revenue function (2).

2.2 Routing and Protection

In fiber transport networks, node-pair demands give rise to required link capacities through routing and protection design. The latter is of vital importance: it is mandatory to allocate capacity to protect carried traffic so as to be able to weather a failure, such as a fiber cut or equipment outage. Typically, all traffic affected by a failure must be re-routed. There are many possible schemes for ensuring survivability, each characterized by a particular trade-off between efficiency of capacity use and complexity of the implementation. See, e.g. [2].

In this paper we consider several routing/survivability schemes under the common assumption that for each node pair, there is a pre-selected set of disjoint paths to carry the demand; we must choose which paths will be used and how much to route on each path. We will assume that at most one failure can occur, and that in the event of a failure we must be able to reroute all affected traffic.

There is an added modeling benefit that ensues from regarding the set of allowable paths as an input to the optimization. The reason for this is that in a realistic application, the question of which paths can be used to route traffic may be outside the scope of the optimization process, e.g. it may be determined by some qualitative service requirements. It is worth discussing this point with a little more detail. In the traditional network design literature, in particular in work with an optimization focus (esp. integer programming aspects of network design) it is frequently assumed that the optimization algorithm has complete control over routing decisions. That is to say, the optimization engine produces the paths used to route demands. From a networking standpoint, this is usually not a realistic assumption, as the optimization may route demands using an excessive amount of "splitting" over many paths, or may produce overly long and circuitous paths, in particular when there is a survivability requirement. A more satisfactory framework is one where a limited path family is input to the optimization engine: then the problem becomes that of choosing paths (and corresponding traffic amounts) from among this family. For example, the path family may embody simple structural restrictions (such as limited path length), or may reflect idiosyncratic business objectives of the network operator, or may describe constraints that are imposed on the network operator, such as being forced to use an existing protocol to route traffic (see [9]).

In fact, all of the above restrictions may well arise at the same time. This makes their incorporation into a mathematical programming model problematic, since, typically, it is impossible or extremely costly to describe the allowable set of paths through the use of e.g. linear inequalities. Thus, in this paper we assume that an explicit path family is given – the precise nature of the family is irrelevant to the design of our solution procedures.

Let $\mathcal{P}(\sigma)$ be the set of candidate paths for demand pair σ – recall that d^t_σ measures the amount of demand for σ at time t. These paths, as discussed above, are assumed to be pairwise-disjoint. Also recall that d^t_σ is a **variable:** it will be determined by the optimization. Denote f^t_r as the amount of flow carried on path r in the normal (non-fault) condition, and $\phi^t_{r'}(r)$ as the amount of flow diverted from route r to r' when r is disrupted by failure ($r, r' \in \mathcal{P}(\sigma)$, and $\phi^t_{r'}(r) = 0$ if $r' = r$). It follows that to carry all demand under the normal condition,

$$\sum_{r \in \mathcal{P}(\sigma)} f^t_r = d^t_\sigma, \tag{3}$$

and to restore all traffic when failure occurs,

$$\sum_{r' \in \mathcal{P}(\sigma)} \phi^t_{r'}(r) = f^t_r, \ \forall r \in \mathcal{P}(\sigma). \tag{4}$$

There are two alternate models that can be built on these equations. In the first, and simplest, we assume that there are given constants θ^t_r and $\eta^t_{r'}(r)$ such that

$$f^t_r = \theta^t_r d^t_\sigma, \text{ where } \sum_{r \in \mathcal{P}(\sigma)} \theta^t_r = 1, \text{ and } \theta^t_r \geq 0.$$

$$\phi^t_{r'}(r) = \eta^t_{r'}(r) * f^t_r, \text{ where } \sum_{r' \in \mathcal{P}(\sigma)} \eta^t_{r'}(r) = 1, \text{ and } \eta^t_{r'}(r) \geq 0.$$

In the second model f_r^t and $\phi_{r'}^t(r)$ are decision variables to be optimized by the model, subject to (3) and (4). The second approach will, of course, typically produce better solutions, at the cost of increased problem size.

Regardless of which approach we take, on link l at time t the total bandwidth that is consumed can be expressed as the sum of two terms:

$$\left(\sum_{\sigma} \sum_{r \in \mathcal{P}(\sigma): l \in r} f_r^t \right) + z_l^t. \tag{5}$$

In this expression the summation in parentheses is the amount of bandwidth used for carrying traffic on primary routes, while z_l^t is the amount of ("redundant") bandwidth used for protecting traffic on other routes when a failure occurs.

There are different ways for planning backup capacity, resulting in different formulas for z_l^t. One approach is to set aside dedicated protection capacity for each demand, i.e.,

$$z_l^t = \sum_{\sigma} \bar{z}_l^t(\sigma), \text{ where } \bar{z}_l^t(\sigma) \geq \phi_{r'}^t(r), \ \forall l \in r', \ \forall r, r' \in \mathcal{P}(\sigma) : r \neq r'. \tag{6}$$

This ensures that for each demand d_σ^t, when any route r fails, every link l on a protection route r' has enough capacity to carry $\phi_{r'}^t(r)$, which is the amount of traffic rerouted to r'.

A more efficient but complicated approach is to allow protection bandwidth to be shared by different demands. In this case, we first define for each pair l, l' of distinct links,

$$\tilde{z}_l^t(l') = \sum_{\sigma} \sum_{r \in \mathcal{P}(\sigma): l' \in r} \sum_{r' \in \mathcal{P}(\sigma): l \in r'} \phi_{r'}^t(r),$$

as the the amount of redundant bandwidth consumed on link l to protect traffic against the failure of l'. The equation shows that when l' fails, all paths r that contain the link are disconnected, and flows on them are rerouted to some backup paths r. The redundant bandwidth needed on l is then the sum of rerouted traffic, $\phi_{r'}^t(r)$, for all r that contains link l. To satisfy rerouting requirements in the presence of any link failure, the redundant capacity on link l should be the maximum of protection bandwidth set aside for each failure scenario, i.e.,

$$z_l^t \geq \tilde{z}_l^t(l').$$

for each $l' \neq l$.

To summarize, we formulate flow values in two ways: we either make them pre-specified fractions of the corresponding demand variables (in which case the flows do not appear in the formulation as explicit variables), or include them in the optimization model as decision variables. There are also two ways to derive required link capacity from flows; one allows the sharing of protection bandwidth by multiple demands and the other does not. The combination of these options generates four routing and protection schemes. The two that do not allow sharing of link protection capacity are called single demand protection with fixed percentage (SDP), and single demand protection with no fixed percentage (SDN). The other two that allow sharing are called multiple demand protection with fixed percentage (MDP), and multiple demand protection with no fixed percentage (MDN).

2.3 Capacity Planning

Our model deploys capacity on the links of the network – at each time period, there must be enough capacity to cover the required bandwidth on each link, as described in equation (5). Of course, the combined pricing/routing/capacity optimization model that we will describe makes all decisions simultaneously, in order to maximize net present value of profit; but in this section we focus on the capacity component. First we will provide a broad description of our model.

In a tactical network design setting, 'capacity" is deployed by means of discrete transmission "systems." Typically there are a number of available different system types (e.g., technologies), with different cost/performance profiles. In addition, deployed capacity systems must be “configured” – this entails making decisions about configuration of sub-components of the systems (ports, interfaces, cables, etc.). The configuration of each deployed system affects the actual capacity of that system and its cost, and also constrain the configuration of other transmission systems. In network design models, the discrete nature of the transmission systems, as well as their configuration, is handled by relying on mixed-integer programming formulations. These formulations heavily draw on precise technical knowledge of the existing transmission technologies. In addition, the models assume fairly accurate knowledge of demand levels, which are not decision variables.

In this paper, however, we do not consider a tactical network design model. On the contrary, ours is a strategic, multi-year model devised to extract long-range information. Namely, we would like to predict demand levels, as a function of our projected price/demand relationship. The usefulness of this type of model is that it can be used to predict future profit levels, future cash flows, resource needs, and so on. An important point is that when using, say, a ten-year model, it is neither feasible nor desirable to use a precise description of the transmission technologies – in particular, we cannot even know what technologies will be available during the later stages of the planning horizon. As a result, a detailed mixed-integer programming model becomes an unnecessary hindrance – restricting the model to the set of technologies that are known at the start of the planning horizon would limit the scope of the model.

Nevertheless, we *would* like to incorporate into our model the empirical fact that per-unit capacity costs are expected to decrease. This is due to improving economies of scale, a result of improving technologies. Our numerical data reflects these effects through a simple “technology roadmap,” that posits a progressive decrease of per-unit capacity costs as time goes by. Thus, to some degree, our experiments reflect the interplay between our price/demand relationship and the technology roadmap.

For additional discussion, please refer to [25]. Also see [12] (especially Chapter 11) for related material from an economic viewpoint. We will now describe our detailed capacity model.

For a link l and time period t, we denote by c_l^t the cost of deploying one unit of capacity on link l at time t. This cost is paid at time t. Having deployed capacity at time t, this capacity can be kept in the link for use during future time periods, or can be retired (at zero cost) at future time periods under a retirement schedule of our choice. On the other hand, each unit of capacity which is not retired incurs a maintenance/operating cost during each future time period that it remains in operation. For time periods $1 \leq s < t \leq T$, the cost incurred by each unit of capacity deployed in link l at time s and still in operation at time t is denoted by $c_l^{s,t}$ (which is paid at time t).

We will use two types of variables:

(i) For each link l and time period $1 \leq t \leq T$, set y_l^t = total capacity deployed on link l at time t, and

(ii) For each link l and time periods $1 \leq s < t \leq T$, set $y_l^{s,t}$ = amount of capacity installed on l at time s, which we still keep in the network at time t.

Using these variables, the cost that we incur for at time t deploying and maintaining this capacity configuration on link l is

$$c_l^t y_l^t \; + \; \sum_{s<t} c_l^{s,t} y_l^{s,t}.$$

Further, we have to have enough capacity in link l at time t so as to handle the load on this link. Using equation (5), this is expressed as:

$$y_l^t \, + \sum_{s<t} y_l^{s,t} \; - \; \left(\sum_{\sigma} \sum_{r \in \mathcal{P}(\sigma): l \in r} f_r^t \right) \, - \, z_l^t \; \geq \; 0$$

where the first two terms indicate the total capacity we have available on l at time t. Finally, we need constraints to describe the possibility of capacity retirement. Put in other words, we want to mandate that the amount of capacity that can be carried over from one period to the next should not exceed the amount that is currently in use. Thus, we have,

$$y_l^{s,s+1} \leq y_l^s,$$

for each l and $1 \leq s < T$, and

$$y_l^{s,t} \leq y_l^{s,t-1},$$

for each l and $1 \leq s < t-1 \leq T$.

3 The formulation

In this Section we provide the complete nonlinear programming formulation of our model, summarizing what we discussed above. In the sequel, we will refer to this formulation as **MULTIREV**.

Our formulation describes decisions over T periods. For $1 \leq t \leq T$, let h_t denote the discount factor used to weigh cash flows at time t. The objective function is formulated as

$$\max F(d,y) \equiv \sum_{t=1}^{T} h_t [\sum_{\sigma} (A_\sigma^t)^{1/\epsilon_\sigma^t} (d_\sigma^t)^{(1-1/\epsilon_\sigma^t)} - \sum_l (c_l^t y_l^t + \sum_{s<t} c_l^{s,t} y_l^{s,t})] \tag{7}$$

In the above equation, items inside the square parenthesis reflect the cash flow in period t, where the first summation refers to the revenue (see Section 2.1), aggregated over all node pairs, and the second one refers to the cost (see Section 2.3), aggregated over all links. Thus, the objective, which we seek to maximize, is the sum of discounted cash flows, or net present value.

Our formulation has two broad sets of constraints. The first set models the fact that capacity purchased in a given period can be reused in the next period (and in future periods) while at the same time allowing the possibility of "retiring" capacity at any time.

$$y_l^{s,s+1} - y_l^s \quad \leq \quad 0 \tag{8}$$

for each l and $1 \leq s < T$, and

$$y_l^{s,t} - y_l^{s,t-1} \quad \leq \quad 0 \tag{9}$$

for each l and $1 \leq s < t-1 \leq T$.

The second set of constraints requires that in any period there should be enough capacity on each link to carry and protect all traffic.

$$y_l^t + \sum_{s<t} y_l^{s,t} - \left(\sum_\sigma \sum_{r \in \mathcal{P}(\sigma): l \in r} f_r^t \right) - z_l^t \quad \geq \quad 0 \tag{10}$$

Flow variables f_r^t and protection bandwidth z_l^t are related to demand variables d_σ^t in different ways, depending on routing and protection schemes.

1. In the case of Single Demand Protection with Given Percentages (SDP), we have:
   For simplicity of demonstration, the formulation of z_l^t below, as well as that for Case 3, is based on the assumption that paths $r \in \mathcal{P}(\sigma)$ are disjoint from each other. This restriction can be easily removed by grouping non-disjoint paths and define new flow variables associated with groups. This extension will not affect our main point, which is the linear structure of constraints.

   $$f_r^t = \theta_r^t d_\sigma^t, \quad \text{and} \quad \forall\, t, l, \quad z_l^t \geq \sum_\sigma \max_{r,r' \in \mathcal{P}(\sigma): l \in r'} [\eta_{r'}^t(r) f_r^t]. \tag{11}$$

   where $\theta_r^t, \eta_{r'}^t$ are constants which are an input to the algorithm. Here the f_r^t are *not* variables – rather, we substitute the first equation in (11) into e.g. (10). The z_l^t are decision variables.

2. In the case of Multiple Demand Protection with Given Percentages (MDP), the quantities $\theta_r^t, \eta_{r'}^t$ and f_r^t are as in the previous case, but instead we impose

   $$z_l^t \geq \sum_\sigma d_\sigma^t \left[\sum_{r \in \mathcal{P}(\sigma): l' \in r} \left(\sum_{r' \in \mathcal{P}(\sigma): l \in r'} \eta_{r'}^t(r) \right) \right] \quad \forall l' \neq l. \tag{12}$$

3. In the case of Single Demand Protection without Given Percentages (SDN), f_r^t are decision variables, and we impose the constraint

$$\sum_{r \in \mathcal{P}(\sigma)} f_r^t = d_\sigma^t. \tag{13}$$

To specify z_l^t, we define another set of decision variables, $\phi_{r'}^t(r)$, representing the amount of traffic rerouted to route r' if the primary path r fails. We impose

$$\sum_{r' \in \mathcal{P}(\sigma)} \phi_{r'}^t(r) = f_r^t.$$

Let $\tilde{\phi}_{r'}^t(\sigma) = \max_{r \in \mathcal{P}(\sigma): r' \neq r} \phi_{r'}^t(r)$. Then we also impose

$$z_l^t = \sum_\sigma \sum_{r' \in \mathcal{P}(\sigma): l \in r'} \tilde{\phi}_{r'}^t(\sigma). \tag{14}$$

4. In the case of Multiple Demand Protection and without Given Percentages (MDN) f_r^t are decision variables constrained by Equation (13), and z_l^t the constraint we impose is

$$z_l^t \geq \sum_\sigma \sum_{r \in \mathcal{P}(\sigma): l' \in r} \sum_{r' \neq r: l \in r'} \phi_{r'}^t(r), \tag{15}$$

where $\phi_{r'}^t(r)$ is defined as in the previous case.

We emphasize that the decision variables appearing in our model can be divided into two categories. Those variables in the first category, $d = \{d_\sigma^t\}$, are nonlinear in the objective function but linear in all constraints, while those in the second category, $w = \{f_r^t, y_l^t, y_l^{s,t}, z_l^t, \phi_{r'}^t(r)\}$, are linear in both the objective and constraints. We will exploit this problem structure in developing our algorithms.

4 The algorithm

In this section we describe a computationally efficient algorithm to solve **MULTIREV**. One of our goals in the design of this algorithm is that it should be *scalable*: its performance should deteriorate gracefully as problem size increases. Note that an instance of **MULTIREV** can be very large even when the underlying network is of moderate size. For example, a network with 50 nodes and 70 links considered over 14 time periods can result in a formulation with approximately 60000 variables for the MDN model. The formulation resulting from a larger network with 150 nodes and 800 links over the same time horizon will already have over one million variables for the same model. The large number of variables is primarily due to the complex routing component of our model – every pair of nodes, in every time period, constitutes a commodity, which must be routed using the given path families.

The highly nonlinear objective function in **MULTIREV** strongly suggests the use of an approach that relies, at the core, on the use of Newton's method or a similar second-order method (see [22], [16]). The attractiveness of such an approach is however tempered by the risk that the complex structure of the constraints, and the large size of the models, will conspire to turn the necessary matrix computations into an insurmountable obstacle. This, in fact, is precisely what we found when testing general-purpose solvers, which proved unusable, as will be detailed in Section 5.

Thus, the challenge at hand is to develop a second-order method that intelligently leverages the structure of the model so that the critical computational matrix algebra is only carried out on "small," sparse matrices.

In this section we provide a high-level description of an algorithm that successfully relies on this approach, with enough details for the purposes of this paper. For a complete description of the algorithm and related details, see [25].

To motivate our algorithm, note that the objective function to be maximized in **MULTIREV**, $F(d, y)$ (see equation (7)) can be written as

$$F(d, y) \quad = \quad R(d) \; - \; c^T y, \tag{16}$$

where R is the sum of present values of all revenues (and is thus nonlinear) and $c^T y$ is the sum of present values of all capacity purchase and maintenance costs, a linear function.

The key insight that we will use is that, for a given demand vector d, the capacity vector y is chosen so that the demands d can be routed at minimum cost. More precisely, for a fixed demand vector $d = \hat{d}$, y is determined from $\hat{d}$ by the solution of the **linear program** $\boldsymbol{LP(\hat{d})}$ given by

$$\begin{aligned} C(\hat{d}) \quad &= \quad \min c^T y \\ s.t. \quad & \quad \text{constraints } (8) - (15) \end{aligned} \tag{17}$$

where in (17) we are fixing demands at $\hat{d}$. This statement of the linear program is perhaps misleadingly simple, as we have other decision variables besides capacities (for example, percentages in the cases where these are variables). Nevertheless, what we have is that, intuitively, having fixed demands we should then choose capacities as cheaply as possible. The following is a consequence of standard results in linear programming theory (see [27] for background).

Theorem 1 *Consider the function $C(d)$ which is the value of the linear program $LP(d)$.*

(a) $C(d)$ is a piecewise-linear convex function of d.

(b) The space of all demand variables is the union of a finite collection of polyhedral regions $\mathcal{F}_1, \mathcal{F}_2, \cdots \mathcal{F}_K$ such that for any such region $\mathcal{F}_j$ there is a vector κ_j with the following property: for any $d \in \mathcal{F}_j$, we have that $C(d) = \kappa_j^T d$.

Technical Note The regions in (b) have disjoint *relative interiors* but, for example, may overlap on lower dimensional faces.

According to the theorem, inside each region the cost function is *linear* in the demands. In view of this result, we can recast our overall optimization problem as

$$\max_{1 \leq j \leq K} \max_{d \in \mathcal{F}_j} \{R(d) \; - \; \kappa_j^T d\} \tag{18}$$

Based on this observation, we can now state a 'prototype' algorithm.

Step 1. Find an initial feasible solution, contained in region $\mathcal{F}_j$ for some j.

Step 2. Use a Newton-like method, optimize $R(d) - \kappa_j^T d$ over $\mathcal{F}_j$. Let $\hat{d}$ be the optimal solution.

Step 3. Use a first-order step to find a better solution than $\hat{d}$ in a different region $\mathcal{F}_k$. If no such improvement is found, **Stop**: $\hat{d}$ is optimal. Otherwise, reset $\mathcal{F}_j \leftarrow \mathcal{F}_k$, and go to Step 2.

End.

The correctness of this ideal algorithm follows from the fact that, as we will see below, the objective function is a concave function of the demands. The essential simplification embodied by this algorithm is that the cost function is now linear in the demands, while, at the same time, the dimensionality of the problem has been substantially decreased. Nevertheless, Step 2 is difficult to implement: it entails constrained nonlinear optimization. Another way to put this is that, while running (say) Newton's method, a step may take us from a feasible solution in region $\mathcal{F}_j$ to outside of this region. In spite of this difficulty, one key property enjoyed by this algorithm is that, when we are in the relative interior of the given region $\mathcal{F}_j$, we can cast our problem as an unconstrained optimization problem in the appropriate space of variables. Our actual algorithm, given below in Section 4.2, uses this point of view. Another potential difficulty lies in the fact that, quite likely, the total number of regions could be very large – but we do not need to enumerate the regions in advance; they are simply 'discovered' in the course of the algorithm. We will return to this point later.

In order to develop a complete algorithm, we also need a starting-point heuristic to carry out Step 1, and we also need a termination criterion that, unlike that in Step 3, allows us to stop the algorithm early when it has achieved sufficient accuracy. The heuristic is outlined next, while the termination criterion is described in Section 4.3.

4.1 Outline of heuristic

The starting-point heuristic is quite simple and we will only outline it here (for full details see [25]). The heuristic consists of performing the following two steps.

(H.a) Solve the optimization problem obtained by restricting **MULTIREV** to *one demand at a time.* That is to say, we focus on a single demand, and then solve the optimization problem: this involves choosing prices and amounts for this demand over the entire planning horizon, and also simultaneously choosing capacities so as to feasibly route the demand during each of the time periods; all of this done so as to maximize the present value of profit. This task is carried out for every demand, separately.

(H.b) The solutions obtained in **(H.a)** are simply put together. This means that the prices, demands, flows (and, when appropriate, percentages) obtained in **(H.a)** are used verbatim, and we simply add (for each time period and for each link) all capacities computed in **(H.a)** so as to obtain a feasible solution.

This heuristic is admittedly myopic – it ignores potential savings that would result from intelligent capacity decisions and how this would impact the pricing process. But, on the other hand, the heuristic already does incorporate some basic understanding of the nonlinear price/demand relationship. As may be expected, on small, simple problem instances the heuristic performs fairly well; less so on larger and more complex instances (see [25] for details). In any case, all we require from the heuristic is that it produce a feasible starting point, and it is the job of the core optimization algorithm to refine this initial solution.

One point that should be clear is that each of the optimization tasks in **(H.a)** is a significantly simpler problem than **MULTIREV** – while the latter has on the order of N^2T variables, each of the single-demand problems solved in **(H.a)** has $O(N+T^2)$ variables (in fact, this can be reduced to $O(T^2)$) and is in addition far simpler. In the case of a large network this has a major impact on problem difficulty, and as a result many solution approaches are likely to be effective; we used a first-order method (see [22]) to carry out **(H.a)** to reasonable accuracy with small computational overhead. Briefly, first-order methods are algorithms that, at each iteration, compute a step direction which maximizes the inner product with the gradient, while maintaining feasibility. Once the step direction has been computed, a line-search is conducted so as to compute the next iterate (which maximizes the objective along the step-direction). First-order methods are notorious for slow convergence to an optimal solution in the case of highly nonlinear problems (such as ours) – but the risk of slow convergence is mitigated by the relatively much smaller size of the problem that we consider, and by the fact that we do not really need to converge to a very precise solution.

4.2 Core of the algorithm

Now we return to the critical part of the algorithm, which primarily corresponds to Step 2 in the above prototype. In order to make this approach effective, we need to change a bit our definition of 'region'. Rather than viewing regions as defining subsets of the demand space only, the regions we will use involve all variables. However, the (critical) component that, in each given region, the cost function is a linear function of the demands, will be maintained. From a standard nonlinear programming point of view, our method is (essentially) an active-set method that operates in a reduced space of coordinates. It differs from a standard active-set method in that it treats the demand variables in a special way, and that it only focuses on the capacity constraints. In this last regard, the method also bears some resemblance to Lagrangian relaxation schemes, though it always maintains a primal feasible solution. Details can be found in [25]; a similar algorithm was employed in [4]. See [22] and [14] for general background. Additional comments concerning the design of our method are given later.

The constraints of our optimization problem, including nonnegativity, can be written as:

$$-Pw + Qd \quad \leq \quad 0 \tag{19}$$

where w is the vector containing all variables other than demand variables. Hence w includes not only the capacity variables, but also the flow variables, and all other variables as needed in the various versions of our model. Consequently, with a slight abuse of notation, we denote the cost incurred by w as

$$c^T w.$$

This is an abuse of notation in the sense that in equation (16) the vector c was restricted to the capacity variables; but we would rather not introduce new notation.

Definition 1 *A* **working set** *consists of a pair* (I, J, K)*, where* I *is a set of rows of the matrix* $[P\ \ Q]$*,* J *is a set of* w*-columns,* K *is a set of* d*-columns, and such that the submatrix of* P *indexed by* I *and* J *has full column rank.*

Definition 2 *Let* (I, J) *be a working set. We say that a vector* $(\hat{w}, \hat{d})$ *is* **consistent with** (I, J) *if*

(i) $-P\hat{w} + Q\hat{d} \ \leq\ 0$ *with equality precisely for those rows* $i \in I$*.*

(ii) $\hat{w}_j > 0$ *if and only if* $j \in J$*,*

(iii) $\hat{d}_k > 0$ *if and only if* $k \in K$*,*

Now we have the following (straightforward) results:

Lemma 2 *Suppose* (w^*, d^*) *is an optimal solution to the overall nonlinear optimization problem. Then, without loss of generality,* (w^*, d^*) *is consistent with some working set* (I^*, J^*, K^*)*.*

Proof. We have (w^*, d^*) is feasible; define I^* to be the set of constraints satisfied by (w^*, d^*) with equality. Our optimization problem has an objective that is linear in the variables w; using standard linear programming facts we have that J^* can be chosen as desired (the set of columns in a basis of the submatrix of P indexed by I^*). K^* is the set of positive demand variables. ■

Lemma 3 *Let* (I, J, K) *be a working set. Then there is a vector* $\kappa = \kappa_{I,J,K} \in R^K$ *with the property that, for any vector* $(\hat{w}, \hat{d})$ *consistent with* (I, J, K)*,*

$$c^T\hat{w} \ \ = \ \ \kappa^T\hat{d}_K. \tag{20}$$

Proof. Let $\bar{Q}$ denote the submatrix of Q corresponding to the row set I. By definition of working set and consistency,

$$\hat{w}_J = Z^{-1}\bar{Q}\hat{d},$$

where Z is an appropriate submatrix of the submatrix of P indexed by I and J, and where $\hat{w}_J$ is the subvector of $\hat{w}$ indexed by the column set J. As a result,

$$c^T\hat{w} \ \ = \ \ c_J^T\hat{w}_J \ = \ c_J^T Z^{-1}\bar{Q}\hat{d}, \tag{21}$$

and the result follows since $d_k = 0$ for $k \notin K$. ■

As a consequence of Lemmas 2 and 3 we can now state the critical result:

Lemma 4 *Let* (I, J, K) *be a working set. Then if* (w, d) *is consistent with* (I, J, K) *its objective value in the overall optimization problem is*

$$R(d_K) - \kappa_{I,J,K}^T d_K,$$

where d_K *is the subvector of* d *indexed by the columns* K*, and, with a slight abuse of notation* $R(d_K)$ *is the revenue associated with* d*.* ■

Using the above definitions and results, we state our core algorithmic step.

Routine $\Omega(\hat{w}, \hat{d}, \hat{I}, \hat{J}, \hat{K})$

Initialization. We are given a working set $(\hat{I}, \hat{J}, \hat{K})$ and a vector $(\hat{w}, \hat{d})$ that is consistent with $(\hat{I}, \hat{J}, \hat{K})$.

A. Using Newton's method, solve the **unconstrained** optimization problem

$$u = \max R(d_K) - \kappa^T_{I,J,K} d_K,$$

with solution $\bar{d}_K$.

B. Let $\bar{w}_J$ be the (unique) solution to:

$$-P_{I,J}\bar{w}_J \, + \, Q_{I,K}\bar{d}_K \, = \, 0,$$

where $P_{I,J}$ and $Q_{I,K}$ are, respectively, the submatrices of P and Q indexed by row set I and column sets J and K.

C. Define $\bar{d}$ to be a vector of demands that equals d_K on the column set K and is zero otherwise; and similarly define the w-vector $\bar{w}$ so that it agrees with w_J on the column set J and is zero otherwise.

D. If $(\bar{w}, \bar{d})$ is consistent with $(\hat{I}, \hat{J}, \hat{K})$ then $(\bar{w}, \bar{d})$ is feasible, and is optimal over all solutions consistent with $(\hat{I}, \hat{J}, \hat{K})$. Routine **exits** and outputs $(\bar{w}, \bar{d})$.

E. Otherwise, $(\bar{w}, \bar{d})$ violates a constraint (possibly nonnegativity); hence there is a maximum value $0 < \lambda < 1$ such that

$$(\breve{w}, \breve{d}) \doteq (1 - \lambda)(\hat{w}, \hat{d}) + \lambda(\bar{w}, \bar{d})$$

is feasible. Since $\lambda < 1$, $(\breve{w}, \breve{d})$ satisfies with equality some inequalities that are slack for $(\hat{w}, \hat{d})$. Then, without loss of generality, $(\breve{w}, \breve{d})$ is consistent with a working set $(\breve{I}, \breve{J}, \breve{K})$.

We reset $(\hat{w}, \hat{d}) \rightarrow (\breve{w}, \breve{d})$ and $(\hat{I}, \hat{J}, \hat{K}) \rightarrow (\breve{I}, \breve{J}, \breve{K})$, and **go to A**.

END.

Comment: essentially this method can be viewed as a Newton method that operates on reduced coordinates; but the choice of which coordinates to eliminate is specific to problem **MULTIREV**. The choice we make is "good" in that, by removing the capacity inequalities, the problem naturally tends to decompose on separate problems that reflect the demand structure (this is a feature shared with Lagrangian relaxation schemes for solving e.g. multicommodity flow problems). A complete decomposition is not always possible because of the multi-period structure of the problem, but nevertheless a substantial simplification of the problem is attained. We have:

Lemma 5 *Routine Ω is finite and always exits as in* **D**.

Proof Sketch. Any time that the routine executes step E one more tight inequality is found, and henceforth the routine operates in a lower dimensional face. Thus the routine must terminate in step **D**. ■

4.3 Termination criterion

The primary termination criterion we use relies on a simple, yet experimentally effective upper bound on the optimum discounted profit. This upper bound, furthermore, is already obtained when running procedure Ω and hence entails no additional computational burden.

The key insight to understand the upper bounding procedure lies in Theorem 1. To restate it, recall that the profit function associated with a particular demand vector d is of the form $R(d) - C(d)$, where R is the (discounted) revenue function and C is the (discounted) cost of routing d. Theorem 1 states that $C(d)$ is convex; and that the space of all demands is the union of a collection of polyhedral regions $\mathcal{F}_j$, such that C is linear when restricted to any one such region. As a result, we have:

Theorem 6 *Let V^* denote the optimal value of* **MULTIREV**. *Suppose $\mathcal{F}_j$ is one of the regions, and let κ_j be a vector such that $C(d) = \kappa_j^T d$ for all $d \in \mathcal{F}_j$. Write*

$$V_j^* \quad = \quad \max_d \left\{ R(d) \, - \, \kappa_j^T d \right\}. \tag{22}$$

Then:

$$V_j^* \quad \geq \quad V^*. \tag{23}$$

Proof. First, we stress that in (22) the maximum is taken over **all** d, not just all $d \in \mathcal{F}_j$.

Further, as a technical point, both maxima *are* attained, without loss of generality, if all capacity costs are positive. In this case $R(d) - C(d)$ goes to $-\infty$ as $|d| \rightarrow +\infty$.

Now consider some other region $\hat{d} \in \mathcal{F}_k$. Then once more $C(d)$ is linear over all d in this region, i.e. if $d \in \mathcal{F}_k$ then $C(d) = \kappa_k^T d$ for some vector κ_k. In summary,

$$C(d) \quad = \quad \kappa_j^T d, \ \ \forall\, d \ \in \ \mathcal{F}_j, \tag{24}$$

$$C(d) \quad = \quad \kappa_k^T d, \ \ \forall\, d \ \in \ \mathcal{F}_k. \tag{25}$$

Since C is in addition convex, we have

$$\kappa_k^T d \quad \geq \quad \kappa_j^T d, \ \ \forall\, d \ \in \ \mathcal{F}_k. \tag{26}$$

(in order to see this, it may help to think of the line segment joining a point in $\mathcal{F}_j$ and a point in $\mathcal{F}_k$). Thus, for any $d \in \mathcal{F}_k$,

$$R(d) \, - \, C(d) \, = \, R(d) \, - \, \kappa_k^T d \quad \leq \quad R(d) \, - \, \kappa_j^T d, \tag{27}$$

and now (23) follows by definition of V_j^* in (22). ■

Suppose we temporarily think of a working set (I, J, K) as defining a region $\mathcal{F}_j$. Then, each time we execute Step **A** in procedure Ω we are computing a new upper bound on the value of **MULTIREV**; and we can use this step to keep track of the *best* upper bound found so far.

Now note that, as shown in Lemma 4, the cost function $C(d)$ is linear over the points consistent with any given working set; consequently, if somewhat informally, we can say that each polyhedral region $\mathcal{F}_j$ is a union of working sets. More precisely: it is the projection to the demand space of those points that are consistent with a finite family of working sets, all of which define the same linear function $\kappa_{I,J,K}^T d$. Hence Theorem 6 applies, and the upper bounding procedure described in the previous paragraph can be used.

4.4 The complete algorithm

Using Sections 4.2 and 4.4 we can now systematically lay out a formal algorithm to solve **MULTIREV** derived from our prototype algorithm given above. As the procedure iterates, it will keep track of two values: v^F, the *maximum* objective value attained by any feasible solution found by the algorithm so far, and v^U, an upper bound on the value of the problem. For full details, see [25].

I. Use the starting-point heuristic to find an initial feasible solution $(\hat{w}, \hat{d})$. Without loss of generality, $(\hat{w}, \hat{d})$ is consistent with some working set $(\hat{I}, \hat{J}, \hat{K})$. Let v^F be the objective value associated with this solution, and set $v^U \leftarrow +\infty$.

II. Run routine $\Omega(\hat{w}, \hat{d}, \hat{I}, \hat{J}, \hat{K})$, which outputs a vector $(\bar{w}, \bar{d})$. Let $\bar{u}$ be the smallest value u computed in any execution of step **A** during this run of Ω. Reset $v^U = \min\{v^U, \bar{u}\}$.

III. Reset v^F to be the objective value of the solution $(\bar{w}, \bar{d})$. If $v^U - v^F$ is small, **exit**.

IV. Perform a first-order step from $(\bar{w}, \bar{d})$, to find a better solution, $(\hat{w}, \hat{d})$. Go to **I.**

END.

5 Computational experiments

We tested our algorithms using network data obtained from Lucent. This data involved international long-distance networks. The data included the (geometrical) length of links.

First, we used five medium-sized networks; by varying some of the numerical parameters we generated a total of 1350 problem instances of **MULTIREV** on which we ran our implementation.

In addition, in order to stress-test our algorithm, we used five large networks to generate a set of 50 problem instances.

The medium-sized problems were solved on a 336 MHz UltraSPARC machine with 1.7GB of RAM, while the tests using larger networks were conducted on a 1.89 GHz Xeon with 3GB of RAM.

We will next describe how we constructed the problem instances.

5.0.1 Cost data.

We generated capacity investment and maintenance data that are consistent with our modeling goals, as described in Section 2.3. In order to do so, for each link l we generated the per-unit capacity investment at time t, c_l^t and the per-unit maintenance costs, $c_l^{t,s}$ (see Section 2) for $1 \leq t < s \leq T$. The values c_l^t should be non-increasing (e.g. $c_l^{t+1} \leq c_l^t$) so as to model the emergence of improved technologies. Finally, in order to make the data realistic, we wanted to incorporate into the costs a dependence on the geometrical length of the links.

To achieve these ends, we set, first,

$$c_l^t = \gamma^{t-1} \Delta_l. \tag{28}$$

Here, Δ_l is the geometrical length of link l, and $0 < \gamma < 1$ is a constant chosen in a manner described below. The import of equation (28) is that in any given time period, the unit capacity investment costs are proportional to link lengths, and that they become progressively cheaper as time goes by. Note that ideally one might want some proportionality constant in (28) – but the objective of **MULTIREV** (see equation (7)) can be scaled without changing the nature of the problem.

Second, we generated maintenance costs of the form

$$c_l^{t,s} = c_l^t \, \mu \, \alpha^{s-t}$$

for any link l, and each pair of time periods $t < s$, where $\mu < 1$ and $\alpha > 1$. This models the case where (per unit) maintenance costs are a fixed fraction of (per unit) investment costs, but become more expensive with age. For our experiments, we set $\mu = 0.05$ and $\alpha = 1.05$.

5.0.2 Price-demand data.

As discussed in Section 2.1 we used a price-demand relationship of the form $d_\sigma^t = A_\sigma^t (p_\sigma^t)^{-\epsilon_\sigma^t}$ for any given service σ. Denoting by N the number of demands (e.g., the number of node pairs) we set $A_\sigma^t = \frac{A}{N}$ for all t and σ, where the parameter A was randomly chosen using a procedure described below. This procedure was also used to create random values to the parameters ϵ_σ^t.

5.0.3 Numerical parameters

In summary, to completely specify a problem instance, we had to assign values to three types of parameters: γ (in eq. (28)), A, and the ϵ_σ^t.

In order to do so, we used the following procedure.

1. First, choose a triple $(\hat{A}, \hat{\gamma}, \hat{\epsilon})$ from among the values in Table 1.
2. Second, having chosen $(\hat{A}, \hat{\gamma}, \hat{\epsilon})$, we then choose A randomly, from a uniform distribution in the interval $(\frac{4\hat{A}}{5}, \frac{6\hat{A}}{5})$. Similarly, γ is chosen from a uniform distribution in $(\hat{\gamma} - 0.05, \hat{\gamma} + 0.05)$. Finally, each ϵ_σ^t is chosen from a uniform distribution in $(1, 2\hat{\epsilon} - 1)$.

$\hat{A}$	$\hat{\gamma}$	$\hat{\epsilon}$
50000	0.95	1.3
500000	0.9	1.4
5000000	0.85	1.5

Table 1: Data parameters

Note that using this procedure, the mean of A is $\hat{A}$, the mean of γ is $\hat{\gamma}$, and the mean of ϵ_σ^t (for each σ and t) is $\hat{\epsilon}$. All possible triples $(\hat{A}, \hat{\gamma}, \hat{\epsilon})$ in Table 1 were used (a total of 27 combinations) and for each triple, 50 problem instances were generated as indicated, for a grand total of 1350 data sets.

Finally, the time horizon T was set to 14 in all cases, and we used the discount factor $h_t = 0.86^{t-1}$ for every time period t except for period T, where we let h_T take a larger value to account for the terminal value of installed network capacity. In our example, $h_T = 2.0$, which is calculated based on the assumption of a 7% continuous cash flow growth after the planning horizon.

In Table 2 we describe basic properties of the medium-sized data sets and the resulting instances of **MULTIREV**. Here the last two columns show the minimum and maximum number of paths per node pair. The small number of paths used is due to the sparsity of the networks. The paths were generated using the *SPIDER* code [18], a tool that generates paths so as to achieve various survivability criteria. The columns headed "Vars", "Const" and "Nonz" describe, respectively, the number of variables, constrains and nonzeros in the optimization problem.

Set	#Nodes	#Links	Vars	Const	Nonz	#Paths (min)	#Paths(max)
snet1	14	22	3584	2310	17472	2	3
snet2	38	48	12782	2940	96222	2	3
snet3	47	55	20909	5775	196623	2	4
snet4	50	64	23870	6720	223272	2	3
snet5	70	94	43680	9870	431382	2	3

Table 2: medium-sized problems

5.1 Test results

We can now describe the results of our computational tests. Table 3 shows the average, maximum and minimum percentage error ("GAP") yielded by our algorithm on different parameter groups, while Table 4 presents running-time information, both for runs using the SDP model. [Here the percentage error is the relative gap between the upper and lower bounds computed by the algorithm.]

Table 3 clearly shows that the problem becomes easier to solve as the elasticity ϵ and the yearly reduction rate γ parameters decrease. On the other hand the scaling constant A does not affect the performance. This observation supports the natural assumption that the problem becomes harder to solve as the demand-price curve (which is essentially defined by the elasticity) becomes steeper. In addition, using a smaller value of γ induces a large decrease in cost from one year to another and helps to reduce the problem degeneracy, thus allowing a faster convergence.

The summary performance of the other three models on the same data sets is reflected in Tables 5 and 6. Table 5 shows the average and boundary optimality gaps for SD model without percentages (SDN) and MD model with (MDP) and without (MDN) percentages. Table 6 shows the average running time for the models.

Generally the problems become harder to solve as problem size grows. Nevertheless, as we will see in subsection 5.3, our algorithm can handle large instances of the models without substantial degradation of solution quality or running time. However, we will first compare the performance of the different models from the perspective of the practical model that we discuss in this paper.

net	Gap (%) (%)	$\hat{A}$			$\hat{\gamma}$			$\hat{\epsilon}$		
		50000	500000	5000000	0.95	0.9	0.85	1.3	1.4	1.5
snet1	Ave	0.00081	0.00077	0.00082	0.001	0.00078	0.00062	0.00098	0.00075	0.00065
	Max	0.0085	0.0082	0.0088	0.0098	0.0081	0.0068	0.0096	0.0082	0.0071
	Min	0	0	0	0	0	0	0	0	0
snet2	Ave	0.006	0.0061	0.0061	0.008	0.006	0.0041	0.0086	0.0062	0.004
	Max	0.051	0.052	0.051	0.063	0.051	0.039	0.071	0.053	0.04
	Min	0	0	0	0	0	0	0	0	0
snet3	Ave	0.019	0.018	0.016	0.044	0.017	0.009	0.039	0.018	0.01
	Max	0.078	0.076	0.075	0.088	0.076	0.064	0.089	0.075	0.065
	Min	0	0	0	0	0	0	0	0	0
snet4	Ave	0.019	0.023	0.021	0.031	0.021	0.011	0.033	0.021	0.012
	Max	0.11	0.125	0.14	0.25	0.13	0.1	0.21	0.12	0.096
	Min	0	0	0	0	0	0	0	0	0
snet5	Ave	0.021	0.024	0.025	0.041	0.023	0.012	0.033	0.022	0.013
	Max	0.193	0.198	0.21	0.29	0.2	0.176	0.28	0.196	0.18
	Min	0.011	0.014	0.009	0.018	0.013	0.007	0.019	0.014	0.007

Table 3: Performance summary - SDP model

net	$\hat{A}$			$\hat{\gamma}$			$\hat{\epsilon}$		
	50000	500000	5000000	0.95	0.9	0.85	1.3	1.4	1.5
snet1	15	15	14	16	15	15	16	16	14
snet2	45	47	45	47	46	44	46	46	44
snet3	100	100	99	102	99	99	100	100	98
snet4	150	152	151	153	151	148	153	152	148
snet5	213	212	214	215	213	213	214	214	212

Table 4: Average running time (sec) - SDP model

One of the goals of our study is to analyze the impact of survivability on profit. In particular, we would like to study the effect on profit of using fixed percentages, and to compare the SD and MD models, as they were designed to provide different degrees of survivability, at potentially higher cost. Table 7 shows the (percent) decrease in the optimal objective value when fixed percentages are used and Table 8 demonstrates the average increase in total profit when the MD model is used instead of the SD model.

The average improvement is about 2% of the overall profit for all cases considered. Thus a weaker survivability model offers a considerable increase in profit. On the other hand the decrease in profit resulting from using fixed percentages is also about 2% on average. However, it is evident from Tables 5 and 6 that reducing the size and complexity of the problem by using fixed percentages does not provide a considerable decrease in the running time or noticeable improvement of the algorithm performance. This in turn implies that it is most likely not worthwhile to utilize this technique.

5.2 Comparison to other solvers

In order to evaluate our algorithm we compared it against LOQO [30] and SNOPT [17], two well-known and highly regarded general-purpose nonlinear solvers. Recall that Table 2 describes statistics on the optimization problems.

In Table 9 we show the performance comparison of our algorithm and solvers on these problems. In each cell we show the average optimality gap (in percents) followed by the average running time, in seconds. A "-" indicates that the solver did not converge, or could not run due to excessive memory requirements.

Model	Gap (%)	snet1	snet2	snet3	snet4	snet5
SDN	Ave	0.32	0.35	0.48	0.47	0.65
	Min	0.22	0.17	0.2	0.21	0.4
	Max	0.44	0.61	0.88	0.85	1.13
MDP	Ave	0.27	0.29	0.33	0.33	0.41
	Min	0.16	0.18	0.16	0.18	0.2
	Max	0.39	0.43	0.52	0.59	0.64
MDN	Ave	1.19	1.77	2.01	2.2	2.3
	Min	0.82	1.16	1.29	1.56	1.45
	Max	1.56	2.47	2.89	3.08	3.89

Table 5: Performance summary - SDN, MDP, MDN models

Model	snet1	snet2	snet3	snet4	snet5
SDN	10	79	119	108	284
MDP	11	80	126	130	304
MDN	13	93	142	158	332

Table 6: Running time (sec) summary - SDN, MDP, MDN models

It seems clear that the general-purpose solvers cannot directly handle the larger medium-sized models. What is more, in all of the cases where the solvers were not able to converge either no feasible solution was found, or the optimality gap was unacceptably large.

In those cases where LOQO was able to solve the problem, our algorithm significantly outperformed it in terms of running time while still providing a solution with negligible optimality error.

5.3 Larger problems

In order to study the performance of our algorithm in a more comprehensive fashion, we also carried out tests involving fifty data sets arising from large, dense networks.

Table 10 describes the five larger problem instances that we generated.

The numerical parameters for these runs were chosen using the same recipe as for the medium-sized problems; but since these larger problem instances were likely to be far more demanding problem instances, we only used the triple $\hat{A} = 50000$, $\hat{\epsilon} = 1.5$, and $\hat{\gamma} = 0.95$ – recall that these values generated the hardest instances on the medium-sized data sets. For the same reason the performance was tested on the largest model (MDN). The triple $(\hat{A}, \hat{\gamma}, \hat{\epsilon})$ was used to generate ten random triples (A, γ, ϵ) as explained above, for each network. Thus, we considered a total of 50 large problem instances.

5.3.1 Comparing to other solvers on the large models

Based on the comparisons given above, it is reasonable to expect that LOQO and SNOPT will be unable to directly handle the larger problems. In fact, our tests showed that the underlying computational linear algebra (in particular, Cholesky factorizations) was a critical stumbling block: all runs using either solver either produced no output (even after a very long time had elapsed) or were aborted, by the solver. In some cases this resulted from a 'crash', or core-dump, and it is difficult to say exactly why this happened, but it appears likely that this was due to excessive memory requirements, despite having 3GB of physical memory available.

However, hypothetically, a nonlinear solver endowed with an extremely efficient Cholesky factorization *might* be able to mount a challenge against our method. There are large numbers of nonlinear solvers (both academic and commercial codes) that are available; we would like to tackle our hypothetical question

Model	snet1	snet2	snet3	snet4	snet5
SDN/SDP	1.5	1.7	1.6	1.8	1.85
MDN/MDP	1.6	1.9	1.86	1.91	1.9

Table 7: Decrease in profit (%) when fixing percentages

Model	snet1	snet2	snet3	snet4	snet5
SDP/MDP	1.92	1.79	1.98	2.01	2.02
SDN/MDN	2.4	2.34	2.22	2.3	2.19

Table 8: Change in profit (%) – SD vs. MD comparison

without engaging in wholesale testing of every code. Fortunately, a simple test turned out to provide a negative answer to our question.

CPLEX [20] is known to have a very fast Cholesky factorization scheme that makes effective use of sparsity, and is thus a natural algorithm to test on the larger problems. However, CPLEX does *not* handle general nonlinear optimization problems, making a direct comparison impossible. On the other hand, CPLEX *does* solve (convex) quadratic minimization problems. To accommodate this issue, we **replaced** the nonlinear objective in **MULTIREV** with its second order Taylor series approximation, evaluated at the solution computed by our heuristic, yielding, after changing the sign of the objective, a single convex quadratic program. While this negates a direct comparison with our algorithm, it makes it possible to provide a clear (negative) answer to our hypothetical question, as we will see next. Table 11 shows a running-time comparison of our algorithm (run on the nonlinear problem) with CPLEX (run on the quadratic relaxation of the problem).

As we can see from this table, CPLEX's efficiency deteriorates rapidly as the network size and density increase, while our algorithm shows stable and scalable performance. From this test, we conclude that is rather unlikely that a general-purpose second-order method will dramatically out-speed our implementation. Of course, it is conceivable that a first-order method might "work" – but we are skeptical, given the large size, highly nonlinear objective, and complex constraints of **MULTIREV**.

Table 12 illustrates the optimality gap and the running time of our algorithm on the large problem instances.

6 Conclusion

Our numerical results indicate that our approach leads to a viable algorithm that provides a high quality solution within a reasonable time-frame, and scales well with problem size. Note that the specific form of the nonlinear component of the objective function is not essential for the development of the algorithm. The idea of projecting out the capacity variables and working only with the demand or flow variables can thus be extended to different pricing models as long as the resulting objective can be formulated as a separable concave function. We expect that this approach will prove efficient for other network design problems with linear or concave objective functions.

Further, we have demonstrated that the integration of pricing and design can be efficiently accomplished, and that it should be possible to handle more detailed design models (for example, using more sophisticated protection requirements).

An interesting extension would be to incorporate stochastic components into the price/demand relationship (as in [13], [11]). Another would be to create a hybrid tactical-strategic network layout model that serves to capacitate networks in a medium-term setting while incorporating both economic considerations and a more accurate model of capacity systems. Some work in this direction is described in [25]

Solver	snet1	snet2	snet3	snet4	snet5
LOQO	0.000001/30	0.000001/600	0.000001/720	0.000001/900	–
SNOPT	0.000001/90	–	–	–	–
Our Code	0.0008/15	0.006/46	0.018/100	0.021/152	0.023/213

Table 9: Performance comparison on real problems - Average optimality gap (%) and time (sec)

net	#Nodes	#Links	Vars	Const	Nonz	#Paths(min)	#Paths(max)
bnet1	100	500	745500	406000	5299700	3	6
bnet2	150	900	1659000	889350	15340500	3	7
bnet3	200	1500	4336500	1850100	40267500	3	9
bnet4	250	2000	6746250	2852500	69618500	3	10
bnet5	300	4500	18053700	5558700	311100300	4	12

Table 10: Larger problem instances

References

[1] D. Alevras, M. Grötschel, R. Wessäly, Cost-efficient network synthesis from leased lines, *Annals of Operations research*, 76, pp. 1-20, 1998.

[2] D. Alevras, M. Grötschel, R. Wessäly, Capacity and survivability models for telecommunication networks, in *Proc. Euro/Informs Meetiing*, Barcelona, pp. 187-199, 1997.

[3] A. Balakrishnan, T.L. Magnanti, R.T. Wong, A Decomposition Algorithm For Local Access Telecommunications Network Expansion Planning, *Operations Research*, 43, pp. 58-76, 1995.

[4] D. Bienstock, Computational study of a family of mixed-integer quadratic programming problems, *Mathematical programming*, 74, pp. 121-140, 1996.

[5] D. Bienstock, O. Günlük, S. Chopra and C.Y. Tsai, Mininum cost capacity installation for multicommodity flows, *Mathematical programming*, 81, pp. 177-199, 1998.

[6] D. Bienstock, O. Günlük, Capacitated network design - polyhedral structure and computation, *Mathematical programming.*

[7] D. Bienstock, G. Muratore, Strong inequalities for capacitated survivable network design problems, *Mathematical programming*, 89, no. 1, Ser. A, pp 127-147, 2000.

[8] D. Bienstock, I. Saniee, ATM network design: traffic models and optimization-based heuristics, *Telecommunication systems*, 16:3,4, pp. 399-421, 2001.

[9] A. Bley, M. Grötschel and R. Wessäly, Design of Broadband Virtual Private Networks: Model and Heuristic for the B-WiN, Preprint SC 98-13, ZIB (Berlin), 1998.

[10] O. Günlük, B. Brockmüller and L. Wolsey, Designing Private Line Networks - Polyhedral Analysis and Computation, CORE - Discussion Paper Series, pp. 96-47.

[11] X. Chen and D. Simchi-Levi, Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The finitie horizon case, MIT OR Center Technical Memorandum, April 2002.

[12] A. Dixit and R. Pindyck, *Investment under Uncertainty*, Princeton University Press (1994).

[13] A. Federgruen and A. Heching, Combined pricing and inventory control under uncertainty, *Operations Research* 47, pp. 454-475, 1999.

[14] A.V. Fiacco and G.P. McCormick, *Nonlinear Programming: Sequential Unconstrained Optimization Techniques*, Wiley, New York (1968).

Solver	bnet1	bnet2	bnet3	bnet4	bnet5
CPLEX (QP Approx)	2341	7982	14324	18943	–
Our Code	1943	2457	2691	2893	3103

Table 11: Average running times on large models (sec)

Gap (%)	bnet1	bnet2	bnet3	bnet4	bnet5
Ave Gap (%)	2.91	2.99	3.15	3.26	3.51
Max Gap (%)	4.56	4.87	4.91	5.1	5.08
Min Gap (%)	1.71	1.82	1.83	1.93	1.96
Ave Time (sec)	1943	2457	2691	2893	3103

Table 12: Performance summary - Optimality gap and running time on large models

[15] O. Günlük, A branch-and-cut algorithm for capacitated network design problems, *Mathematical programming* 86, 17-39, 1999.

[16] P. Gill, W. Murray, M.Wright, Numerical linear algebra and optimization, *Addison-Wesley*, 1991.

[17] P. Gill, W. Murray, M.Saunders, User's guide for SNOPT 5.3

[18] R. D. Davis, K. Kumaran, G. Liu, I. Saniee, SPIDER: A simple and flexible tool for design and provisioning of protected lightpaths in optical networks, *Bell Labs Technical Journal*, January-June 2001.

[19] CLAPACK library, *www.netlib.org/clapack*

[20] ILOG CPLEX, Incline Village, NV.

[21] A. Lisser, A. Ouorou, J.-Ph. Vial, and J. Gondzio, Capacity Planning Under Uncertain Demand in Telecommunications Networks, http://hec.info.unige.ch/recherche/99.13.pdf.

[22] D. Luenberger, Introduction to linear and nonlinear programming, *Addison-Wesley*, 1989.

[23] T.L. Magnanti, P. Mirchandani, R. Vachani, Modeling and solving the two-facility capacitated network loading problem, *Operations Research*, 43, pp. 142-157, 1995.

[24] S. G. Lanning, D. Mitra, Q. Wang, M. H. Wright, Optimal planning for optical transport networks, *Philosophical Transactions Royal Society London A*, Vol.358, No.1773, pp. 2183-2196, August 2000.

[25] O. Raskina, *Two large-scale network design problems*, Ph.D. Thesis, Columbia University (2002).

[26] I. Saniee, Optimal routing design in self-healing communications networks, *Int. Trans. Op. Res.*, vol 3, no. 2, pp. 187-195, 1996.

[27] A. Schrijver, Theory of linear and integer programming, *Wiley-Interscience*, 1999.

[28] S. Sen, R. Doverspike, and S. Cosares, Network planning with random demand, *Telecommunication Systems* 3, pp. 11-30, 1994.

[29] M. Stoer and G. Dahl, A polyhedral approach to multicommodity survivable network design, *Numerische Mathematik*, 68, pp. 149-167, 1994.

[30] R. Vanderbei, LOQO User's manual, *Statistics and Operations Research, Technical report No SOR-97-08*, 1997.