Lagrange Relaxation Based Algorithm for DiffServ/MPLS Network Dimensioning with QoS Guarantees

Jaudelice C. de Oliveira
Department of Electrical and Computer Engineering
Drexel University, Philadelphia, PA 19104
Email: jco24@drexel.edu

Caterina Scoglio
Broadband and Wireless Networking Laboratory
School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
Email: caterina@ece.gatech.edu

Abstract—**In this paper, a new approach to dimensioning DiffServ/MPLS networks taking into account per-class bandwidth reservation and fault tolerance cushion weights is proposed. Our dimensioning algorithm is based on Lagrange relaxation and it uses a bandwidth constraint model in order to provide per-class bandwidth reservation, and link costs which translate to backup LSP cushion weights. We propose an optimization formulation for our policy and also a polynomial time heuristic capable of approximating the optimal results. Besides a-priori dimensioning, a modified version of our dimensioning policy can also be run weekly/monthly in order to redimension LSPs according to the stored traffic history. Both proposed heuristics have polynomial complexity and provide bounds on the value of the optimal objective function, which can be used to evaluate the accuracy of the results.**

I. Introduction

The dimensioning of an Multiprotocol Label Switching (MPLS) network includes the assignment of bandwidth resources to a set of pre-selected Label Switched Paths (LSPs) and mapping the same onto a physical network of links with capacity constraints. The dimensioning process also determines the link capacity thresholds associated with the use of some bandwidth reservation scheme for service protection. Service protection controls the Quality of Service (QoS) provided for certain service types by restricting access to bandwidth, or by giving priority access to one type of traffic over another. Such methods are essential to prevent starvation of low-priority flows, to guarantee a minimum amount of resources for flows with short duration, to improve the probability of acceptance for flows with high bandwidth requirements, to maintain network stability by preventing performance degradation in case of a local overload, etc. [1].

Another important aspect to be considered in MPLS networks dimensioning is fault tolerance. Besides Traffic Engineering, traffic protection is another major advantage of using MPLS. In pure IP networks, each failure has to be flooded by the Internet Gateway Protocol (IGP) through the whole network. After the IGP has converged, new shortest paths can be calculated and the failure is repaired. Failure to properly capacity plan a network will lead to congestion.

Previous work, such as NetScope [2] and RATES [3] try to automate network configuration in order to maximize the utilization. The first one estimates the demands using measurements and uses the off-line algorithm described in [4] to alleviate overloaded links. The latter uses the online algorithm described in [5] to minimize the interference between different routes in the network for a specific set of ingress-egress nodes. Both do not take into account demand QoS requirements and only try to minimize the maximum load of certain links in the network. To the best of our knowledge, a dimensioning algorithm which included bandwidth reservation with per Class-Type (suitable for Differentiated Services) guarantees or both bandwidth reservation and fault tolerance concomitantly was not yet proposed.

In this paper, a new approach to dimensioning DiffServ/MPLS networks taking into account per class bandwidth reservation and fault tolerance is proposed. The resulting network dimensioning problem can be modeled as a multi-commodity network flow optimization. Our proposed dimensioning algorithm uses a Lagrange based relaxation of a modified cost function. The algorithm includes a bandwidth constraint model in order to provide per-class bandwidth reservation, and also link weights which translate to backup LSP fault tolerance cushion weights. A modified version of our policy can also be run weekly/monthly in order to redimension LSPs according to the traffic history. Both proposed heuristics have polynomial complexity and provide bounds on the value of the optimal objective function, which can be used to evaluate the accuracy of the results.

The rest of this paper is organized as follows: We first present, in Section II, our proposed formulation to the network dimensioning problem. An optimal solution to this problem is provided, but with exponential running time. In Section III, we propose a new algorithm based on Lagrange relaxation of a modified cost function. In Section IV, a modified version of our dimensioning heuristic is now an online redimensioning tool. It can be run periodically to possibly update the capacity and routing of LSPs according to the traffic demand. A numerical example is discussed in Section V, and finally our conclusions are summarized in Section VI.

II. Proposed Dimensioning Problem Formulation

A. Bandwidth Reservation Constraints

In this section we define the bandwidth constraint model to be used in our dimensioning algorithm.

The fundamental requirement for DiffServ-aware Traffic Engineering (DS-TE) is to be able to enforce different bandwidth constraints for different sets of traffic classes. DS-TE may support up to 8 CTs: CT_c, $c = 0, \cdots, 7$. By definition, each CT is assigned either a Bandwidth Constraint (BC), or a set of BCs. Therefore, DS-TE must support up to 8 BCs: BC_b, $b = 0, \cdots, 7$.

The *Russian Doll Model* (RDM), [6], is recommended by the IETF Traffic Engineering Working Group in RFC 3564 ([7]). Other models have been proposed, such the the *Maximum Allocation Model* (MAM),

The RDM may be defined as follows [6]:

- Maximum number of BCs is equal to maximum number of CTs = 8;
- All LSPs from CT_c must use no more than BC_b (with $b \leq c \leq 7$, and $\mathrm{BC}_b \leq \mathrm{BC}_{b-1}$, for $b = 1, \cdots, 7$), i.e.:

All LSPs from CT_7 use no more than BC_7;

All LSPs from CT_6 and CT_7 use no more than BC_6;
All LSPs from CT_5, CT_6 and CT_7 use no more than BC_5;
$\cdots$
All LSPs from CT_0, CT_1, CT_2, CT_3, CT_4, CT_5, CT_6 and CT_7 use no more than BC_0.

To illustrate the model, assume only three CTs are activated in a link and the following BCs are configured: $BC_0 = 100$, $BC_1 = 80$, and $BC_2 = 50$. Fig. 1 shows the model in a pictorial manner (nesting dolls). CT_0 could be representing the best-effort traffic, while CT_1 the non-real-time traffic, and CT_2 the real-time traffic. Following the model, CT_0 could use up to 100% of the link capacity given that no CT_1 or CT_2 traffic would be present in that link. Once CT_1 comes into play, it would be able to occupy up to 80% of the link, and CT_0 would be reduced to 20%. Whenever CT_2 traffic would also be routed in that link, CT_2 would then be able to use up to 50% by itself, CT_1 would be able to use up to 30% by itself, while CT_0 could use up to 20% alone.

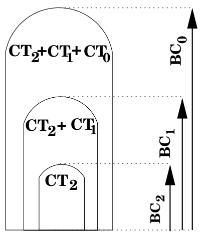

Fig. 1. Russian Doll model with three active CTs.

Next we will explain our proposal on how to calculate these cushion weights.

B. Fault Tolerance Cushion Weights

Besides Traffic Engineering, fault tolerance is another major advantage of using MPLS. There are two kinds of approaches for this problem: global repair mechanisms like edge-to-edge backup LSPs, and local repair mechanisms such as a local detour LSP [8].

To take backup LSP cushion bandwidth into account in our dimensioning algorithm, we propose to use cushion weights in our problem formulation. We propose to calculate these cushion weights based on how frequently a link is selected as a backup path link. Our proposed algorithm is illustrated in Fig. 2.

Cushion Weights Algorithm

1. For every demand between node pairs belonging to the topology in consideration do:
   - Run Dijkstra and determine the shortest path between the demand's source and destination.
   - Prune the selected shortest path from the topology and run Dijkstra again in order to select an alternative path.
   - For the links, ℓ_l, that compose the newly selected path, increment utilization counter $U(\ell_l)_c = U(\ell_l)_c + 1$.
2. For each link ℓ_l calculate the respective cushion weight $\gamma(\ell_l)$ as a function of its utilization in backup paths:
$\gamma(\ell_l) = 1 + \frac{U(\ell_l)_c}{\text{total number of demands}}$

Fig. 2. Algorithm for calculation of cushion weights.

Once all the $\gamma(\ell_l)$s are calculated, we are ready to formulate our dimensioning problem, which will take into account bandwidth reservation for different Class-Types and also fault tolerance through the use of cushion weights. Next we describe our formulation and heuristic algorithm.

C. Problem Formulation

Consider a given physical network topology $T(V, L)$, with a set of nodes V and a set of links L. Suppose the network designer is considering a total of N CTs activated in the network.

In order to simplify the notation, we define matrix M (with dimensions $V \times V$) containing a mapping of the topology $T(V, L)$, assigning links to node-pairs. Each element of matrix M, $M(i, j)$, denotes the physical link $\ell_l \in L$ (with $l = 1, 2, \cdots, \mathcal{L}$ and where $\mathcal{L}$ represents the cardinality of L) that connects nodes $(i, j) \in V$. Note that the matrix may be symmetric since the physical link connecting nodes (i, j) may be the same as the one connecting nodes (j, i):

$$M(i,j) = \begin{cases} \ell_l, \text{ if link } \ell_l \text{ connects nodes } i \text{ and } j; \\ 0, \text{ otherwise.} \end{cases} \tag{1}$$

We define the following notation:

- $\ell_l \in L$: denotes a physical link labeled ℓ_l for which the mapping to the corresponding node-pair (i, j) is given in M;
- $C_{ph}(\ell_l)$: denotes the total capacity of link ℓ_l;
- $C_n(\ell_l)$: denotes the portion of link ℓ_l dedicated to the overlaying MPLS network n; $n = 0, \cdots, N-1$ (MPLS network for CT_n).

To bring the dimensioning model closer to reality, we assume that capacity is available in capacity modules, which implies that the optimization variable $C_{ph}(\ell_l)$ is integer. We can easily further restrict the set of possible integers to commercially available values such as 10 Mbps, 100 Mbps, 155 Mbps, and 622 Mbps.

We define D_n as the $V \times V$ traffic matrix for CT_n (with null diagonal elements). Each element in D_n can be calculated with a Gaussian approximation $D_n = \mu_n + \alpha\sigma_n$, where μ_n is the mean, σ_n is the the standard deviation, and α is a multiplier that controls the extend to which the estimated demand accommodates variability of the traffic. In a Gaussian approximation for the rate distribution, we expect the rate estimation to be exceeded with probability $1 - G(\alpha)$, where G is the cumulative distribution of the standard normal distribution. Depending on the requirement on the probability that the traffic will exceed the capacity, we select the appropriate α.

We also define the matrix $R_n^{i,j}$ containing the set of possible routes for $D_n(i, j)$. Suppose the number of possible routes is given by r. The value of r can be different for each CT and for each node-pair (i, j). Each element in $R_n^{i,j}$ is defined as follows:

$$R_n^{i,j}(h,l) = \begin{cases} 1 & \text{if link } \ell_l \text{ belongs to route } h, (h = 1, \cdots, r); \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

We define $x_n^r(i, j)$ as the portion of demand $D_n(i, j)$ routed over route r. In our previous example, assume $D_0(a, b) = 10$ Mbps and $D_1(a, b) = 20$ Mbps. Therefore, $x_0^1(a, b)$ could be configured as 4 Mbps, while $x_0^2(a, b)$ could be 6 Mbps, meaning that 4 Mbps of $D_0(a, b)$ are routed over route 1 (ℓ_1-ℓ_2-ℓ_3), while 6 Mbps are routed over route 2 (ℓ_4-ℓ_5). $x_1^1(a, b) = 20$ Mbps, since there is only one route for $D_1(a, b)$.

To include bandwidth reservation in our formulation, we define a vector of bandwidth constraints, BC, to be used in the Russian Doll model: $\mathrm{BC} = [\mathrm{BC}_0, \mathrm{BC}_1, \cdots, \mathrm{BC}_7]$.

We define the following objective function:

$$F = \sum_{\forall \ell_l \in L} \gamma(\ell_l) \cdot C_{ph}(\ell_l) \tag{3}$$

where $\gamma(\ell_l)$ is a per-link per-capacity unit cost coefficient which is used to introduce fault management considerations into the formulation.

As constraints, we must ensure that the variables $x_n^r(i,j)$ are positive $\{1\}$; we must ensure that after a demand is partitioned into more than one route option, the total demand value is still the same $\{2\}$; we must also consider that the total bandwidth for CT_n on link ℓ_l is given by the sum of all portions of traffic belonging to CT_n which are routed over ℓ_l $\{3\}$; and that the total capacity on a physical link $C_{ph}(\ell_l)$ is given by the sum of all existing CT traffic on that link $\{4\}$; and finally, we also enforce the bandwidth constraints given by the Russian Doll model $\{5\}$.

Dimensioning Optimization Formulation

GIVEN $T(V,L)$, D_n, $R_n(i,j)$, BC_n, $\gamma(\ell_l)$, and N

FIND $C_{ph}(\ell_l)$ (L integer variables)

MINIMIZING $F = \sum_{\forall \ell_l \in L} \gamma(\ell_l) \cdot C_{ph}(\ell_l)$

SUBJECT TO

$$\{1\}\ x_n^r(i,j) \geq 0, \forall n, r$$

$$\{2\} \sum_r x_n^r(i,j) = D_n(i,j)$$

$$\{3\} \sum_{\forall P_{i,j}(\ell_l)} x_n^r(i,j) = C_n(\ell_l)$$

$$\{4\} \sum_{n=0}^{N-1} \gamma(\ell_l) \cdot C_n(\ell_l) = C_{ph}(\ell_l)$$

$$\{5\} \sum_{n=y; y=0,\cdots,N-1}^{N-1} C_n(\ell_l) \leq \mathrm{BC}_y \cdot C_{ph}(\ell_l)$$

Fig. 3. Our dimensioning optimization formulation.

Fig. 3 contains a summary of our proposed problem, including the constrains, and where $\forall P_{i,j}(\ell_l)$ means for all paths between node-pair (i,j) routed through physical link ℓ_l. It is easy to see that the complexity of the optimization formulation will grow tremendously with the complexity of the network topology. Unfortunately, we cannot hope for an algorithm that could find the optimal solution to this problem and also would run in exponential time. Our heuristic, which will be described next, is based on the Lagrange relaxation technique [9].

III. Proposed Dimensioning Heuristic

A. Lagrange Relaxation based Subproblem

Lagrange relaxation has the property of providing bounds on the value of the optimal objective function and, frequently, of quickly generate good, though not necessarily optimal, solutions with associated performance guarantees [10]. The optimal value of a Lagrangian problem is a lower bound (in minimization problems) on the optimal value of the original problem. We propose to use this technique to solve our dimensioning problem. Our **La**grange **R**elaxation based **D**imensioning **A**lgorithm (**LaRDA**) is based on the heuristic of minimizing the modified objective function $\mathcal{F}' = F + \sum_i w_i \cdot \text{constraint}_i$, where w is a Lagrange multiplier that our algorithm needs to find. If all ws are equal to zero, and all the constraints in Fig. 3 are respected, we find the optimal solution to the original problem. If the constraints are not being fulfilled, we need to increase the value(s) of w(s) so that we increase their importance in the new objective function $\mathcal{F}'$. In this section we show how to find the values of w_i that lead to best results.

In our formulation, we have two sets of Lagrange multipliers, each set containing as many multipliers as the number of links considered in the dimensioning formulation (in the constraints that are being relaxed). Since our formulation is a multicommodity flow problem, the Lagrange multipliers are nonnegative and are associated with the bundle constraints. Note that the new objective function will be subject to only one constraint (nonnegativity constraint).

With the help of the Lagrange relaxation, we have an algorithm that is able to find an estimation of the optimal solution and a bound which tells us how far it is from the optimal solution. In summary, first we eliminate constraints $\{4\}$ and $\{5\}$ by incorporating them into the other constraints and the objective function. We then relax resulting constraining conditions $\{2\}$ and $\{3\}$ by including them into the new objective function. If the solution found is not feasible for the constraining conditions, we increase the constraining conditions' weight (by increasing the value of the correspondent multiplier) in the modified objective function, forcing the solution to approach the optimum. For a more detailed description of Lagrange relaxation methods, refer to [9]. Next, we describe our heuristic in detail.

B. Description of LaRDA

In **LaRDA**, we use the subgradient optimization technique to solve the Lagrange multiplier problem previously discussed. We let w_{1b}^0 and w_{2b}^0 be any initial choice of values. Using subgradient optimization, we determine the subsequent values of w_{1b} and w_{2b} as follows [10]:

$$w_{1b}^{k+1} = [w_{1b}^k + \theta_{1b}^k (\sum_r x_n^r(i,j)^k - D_n(i,j))]^+ \tag{4}$$

$$\begin{aligned} w_{2b}^{k+1} = [w_{2b}^k + \theta_{2b}^k (&\sum_{n=y}^{N-1} \sum_{\forall P_{i,j}(\ell_l)} x_n^r(i,j) - \\ &-\mathrm{BC}_y \cdot \sum_{n=0}^{N-1} \sum_{\forall P_{i,j}(\ell_l)} \gamma(\ell_l) \cdot x_n^r(i,j)]^+ \end{aligned} \tag{5}$$

In these expressions, the notation $[\beta]^+$ denotes the positive part of β, that is $\max(\beta, 0)$.

$x_n^r(i,j)^k$ represents any solution to the Lagrangian subproblem when $w = w^k$. The variable θ^k is the step length at the k-th iteration which indicates how far we move in the gradient direction. For convergence, we would like to find $\theta^k \to 0$ and $\sum_{j=1}^{k} \theta^j \to \infty$, resulting in:

$$\theta_{1b}^k = \frac{\lambda_k \left[\mathcal{F}_{\text{UB}} - \mathcal{L}(w^k)\right]}{\left\| \sum_r x_n^r(i,j)^k - D_n(i,j) \right\|^2} \tag{6}$$

$$\theta_{2b}^k = \frac{\lambda_k \left[\mathcal{F}_{\text{UB}} - \mathcal{L}(w^k)\right]}{\left\| \sum_{n=y}^{N-1} \sum_{\forall P_{i,j}(\ell_l)} x_n^r(i,j) - \text{BC}_y \cdot \sum_{n=0}^{N-1} \sum_{\forall P_{i,j}(\ell_l)} \gamma(\ell_l) \cdot x_n^r(i,j) \right\|^2} \tag{7}$$

In the above equations, $\mathcal{F}_{\text{UB}}$ is an upper bound on the optimal objective function value, and λ_k is a scalar chosen chosen strictly between 0 and 2.

Initially, the upper bound $\mathcal{F}_{\text{UB}}$ is the objective function value of any known feasible solution to the optimization problem. As the algorithm proceeds, it updates this value ($\mathcal{F}_{\text{UB}}$) when it generates a better (i.e. lower cost) feasible solution. We start the algorithm with $\lambda_k = 2$ and then reduce λ_k by a factor of 2 whenever the best Lagrangian objective function value found so far has failed to increase in a specified number of iterations, which we choose to be 4 (based on our experience with case studies). Fig. 4 contains a summary of **LaRDA**. After the optimal Lagrange multipliers are found, $C_n(\ell_l)$ is calculated for each physical link, and finally, $C_{ph}(\ell_l)$ is calculated.

```
**La**grange **R**elaxation based **D**imensioning **A**lgorithm (**LaRDA**)
($w_{ab}^0, \lambda_k, k = 1$)

1. $\theta_{ab}^k$ is calculated as in Equations 6 and 7;
2. $w_{ab}^{k+1}$ is calculated as in Equations 4 and 5;
3. Go to step 6 if one of the following conditions holds:
     • Total number of iterations is equal to MAX=150
     • $\mathcal{F}_{\text{UB}} = \mathcal{L}(w^k)$
     • $\lambda_k < \epsilon$
4. If $\mathcal{F}_{\text{UB}}$ does not reduce in 4 iterations, reduce $\lambda_k$ by a factor of 2;
5. $k = k + 1$; Go back to step 1.
6. Use the values found for $x_n^r(i,j)$ and calculate $C_n(\ell_l)$
   and $C_{ph}(\ell_l)$ as follows:
       $C_n(\ell_l) = \sum_{\forall P_{i,j}(\ell_l)} x_n^r(i,j)$
       $C_{ph}(\ell_l) \geq \gamma(\ell_l) \cdot \sum_{n=0}^{N-1} C_n(\ell_l)$
7. Return $C_{ph}(\ell_l)$s;
```

Fig. 4. Our heuristic's algorithm (**LaRDA**).

MAX and ϵ are constant values which translate to the maximum number of iterations to be run and how small λ_k can be, respectively. Both are used as stop criteria for **LaRDA** and can be defined by the user. Based on our experience running **LaRDA**, we suggest MAX=150. Another stopping criterion that could be used with **LaRDA** is to calculate how much the lower bound is improving when compared to the previous value found. The algorithm could then stop when the improvement is not significant, which means the bound is good enough.

The use of subgradient optimization for solving the Lagrangian multiplier problem is attractive for several reasons, which include the fact that it permits us to exploit the underlying network flow structure, and moreover, the formulas for updating the Lagrange multipliers w_{ab} are rather trivial computationally and very easy to encode in a computer program. The Lagrangian relaxation solution can also be combined with integer programming by using Lagrangian relaxation to develop an improved start point for the integer programming formulation [10].

C. Complexity and Goodness of Solution

The answer to the question of how good are the bounds provided by Lagrange relaxation is problem specific and mostly empirical. In [11], G. Cornuejols, M. Fisher, and G. Nemhauser and in [9], M. Fisher, describe their computational experience with Lagrange relaxation and provide overwhelming evidence that its bounds are extremely sharp. For a total of 33 uncapacitated network design problems solved in [11], 66.6% of the problems resulted in very sharp bounds (99.9% accuracy). That speaks for the goodness of the solution of our Lagrange relaxation based heuristic. Regarding computational complexity, the overall **LaRDA** algorithm has polynomial time complexity. The algorithm is linear with respect to the number of iterations k. In a worst case scenario analysis, regarding the number of nodes V, the number of demands between node-pairs grows with V^2 and considering that so do the number of LSPs, if we also consider that the ratio between the number of nodes and links is constant, then the computational complexity grows with V^5. Therefore, we can estimate that the overall algorithm is $O(k \times V^5)$

IV. Proposed Redimensioning Heuristic

After some period of time, the network administrator may decide to check on the accuracy of the previously considered traffic load used for network dimensioning. The administrator now uses our **La**grange **R**elaxation based **R**edimensioning **A**lgorithm (**LaRRA**), in which Bandwidth Constraints are variables rather than input data, and finds the new values of $C_n(\ell_l)$ and new values for BC_n. The new parameters found for the Russian Doll model would also be taking into account the cushion needed for fault protection. This procedure would be repeated periodically in order to keep the virtual network dimensioning up-to-date with the traffic load offered. Note that in the redimensioning policy we define variables $C_n(\ell_l)$ and BC_n as integer variables. The formulation is very similar, so we will omit it due to the space constraints.

V. Numerical Examples

Consider the Abilene topology shown in Fig. 5. Now consider that nodes v_1, v_2, v_5, v_6, v_7, v_9 and v_{12} (Seattle, Sunny Yale, Kansas City, Houston, Chicago, Atlanta, and New York nodes, respectively) are active and have traffic demand. The topology $T(V, L)$ is known and three CTs are used: CT_0, CT_1, and CT_2. $\text{BC}_0 = 100\%$, $\text{BC}_1 = 60\%$, and $\text{BC}_2 = 40\%$. The cushion weights to be used in our cost function are then calculated according to our proposed algorithm in Fig. 2: $\gamma(\ell_1) = 1.08$, $\gamma(\ell_2) = 1.03$, $\gamma(\ell_3) = 1.08$, $\gamma(\ell_4) = 1.05$, $\gamma(\ell_5) = 1.08$, $\gamma(\ell_6) = 1.03$, $\gamma(\ell_7) = 1.11$, $\gamma(\ell_8) = 1.05$, $\gamma(\ell_9) = 1.03$, $\gamma(\ell_{10}) = 1.08$, $\gamma(\ell_{11}) = 1.08$, $\gamma(\ell_{12}) = 1.03$, $\gamma(\ell_{13}) = 1.05$, $\gamma(\ell_{14}) = 1.03$, $\gamma(\ell_{15}) = 1.05$.

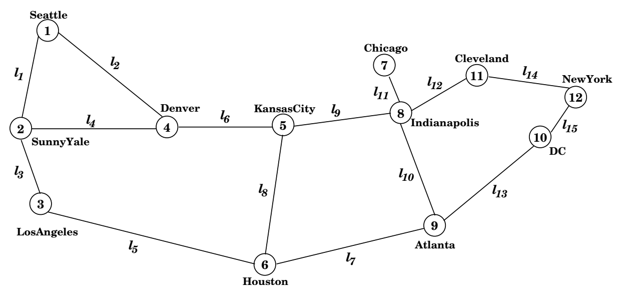

Fig. 5. Abilene backbone network.

Using a Gaussian approximation, the following demands (shown in Table I) are calculated based on the given mean and standard deviation of each traffic:

TABLE I

TRAFFIC DEMANDS FOR NUMERICAL EXAMPLE.

$d_0(1,2) = 100$	$d_0(1,7) = 20$	$d_0(1,9) = 30$	$d_0(1,12) = 30$
$d_0(7,12) = 40$	$d_0(9,1) = 30$	$d_0(9,7) = 30$	$d_0(9,12) = 40$
$d_0(5,6) = 100$	$d_0(7,1) = 20$	$d_0(2,6) = 100$	$d_0(6,9) = 100$
$d_0(12,1) = 30$	$d_0(12,7) = 40$	$d_1(1,2) = 0$	$d_1(1,7) = 10$
$d_1(1,9) = 30$	$d_1(1,12) = 30$	$d_1(7,12) = 30$	$d_1(9,1) = 30$
$d_1(9,7) = 30$	$d_1(9,12) = 30$	$d_1(5,6) = 0$	$d_1(1,12) = 30$
$d_1(2,6) = 0$	$d_1(6,9) = 0$	$d_1(12,1) = 30$	$d_1(12,7) = 30$
$d_2(1,2) = 0$	$d_2(1,7) = 10$	$d_2(1,9) = 40$	$d_2(1,12) = 40$
$d_2(5,6) = 0$	$d_2(7,1) = 10$	$d_2(2,6) = 0$	$d_2(6,9) = 0$
$d_2(12,1) = 40$	$d_2(12,7) = 40$	$d_2(7,9) = 40$	$d_2(12,9) = 40$
$d_2(7,12) = 40$	$d_2(9,1) = 40$	$d_2(9,7) = 40$	$d_2(9,12) = 40$
$d_0(7,9) = 30$	$d_0(12,9) = 40$	$d_1(7,9) = 30$	$d_1(12,9) = 30$

The following routing options are given for each active node-pair. Note that $R_k^{i,j} = R_k^{j,i}$ and therefore the similar routing options were omitted.

$$R_0^{1,2} = \ell_1$$
$$R_0^{1,7} = \ell_2 - \ell_6 - \ell_9 - \ell_{11};\ \ell_1 - \ell_3 - \ell_5 - \ell_7 - \ell_{10} - \ell_{11}$$
$$R_0^{1,9} = \ell_2 - \ell_6 - \ell_9 - \ell_{10}$$
$$R_0^{1,12} = R_1^{1,12};\ \ell_1 - \ell_3 - \ell_5 - \ell_7 - \ell_{13} - \ell_{15}$$
$$R_0^{2,6} = \ell_3 - \ell_5;\quad R_0^{7,9} = \ell_{11} - \ell_{10}$$
$$R_0^{7,12} = \ell_{11} - \ell_{12} - \ell_{14};\ \ell_{11} - \ell_{10} - \ell_{13} - \ell_{15}$$
$$R_0^{9,12} = \ell_{13} - \ell_{15};\ \ell_{10} - \ell_{12} - \ell_{14}$$
$$R_1^{1,7} = \ell_2 - \ell_6 - \ell_9 - \ell_{11};\quad R_1^{1,9} = \ell_2 - \ell_6 - \ell_9 - \ell_{10}$$
$$R_1^{1,12} = \ell_2 - \ell_6 - \ell_9 - \ell_{12} - \ell_{14}$$
$$R_1^{7,9} = \ell_{11} - \ell_{10};\quad R_1^{9,12} = \ell_{13} - \ell_{15}$$
$$R_1^{7,12} = \ell_{11} - \ell_{12} - \ell_{14}$$
$$R_2^{1,7} = R_1^{1,7};\ \ell_1 - \ell_3 - \ell_5 - \ell_7 - \ell_{10} - \ell_{11}$$
$$R_2^{1,9} = R_1^{1,9};\ \ell_1 - \ell_3 - \ell_5 - \ell_7$$
$$R_2^{1,12} = R_0^{1,12};\quad R_2^{7,12} = R_0^{7,12}$$
$$R_2^{7,9} = \ell_{11} - \ell_{10};\ \ell_{11} - \ell_9 - \ell_8 - \ell_7$$
$$R_2^{9,12} = \ell_{13} - \ell_{15};\ \ell_{10} - \ell_{12} - \ell_{14}$$
$$R_0^{5,6} = \ell_8;\quad R_0^{6,9} = \ell_7$$

The results for the physical link capacities found by our optimization formulation are shown in Table II and graphically illustrated in Fig. 6, where the links get thicker as their load increases. Our heuristic found the results in 23 iterations. The dimensioned network has bandwidth constraints of $\text{BC}_0 = 100\%$, $\text{BC}_1 = 60\%$, and $\text{BC}_2 = 40\%$ to provide service protection.

TABLE II

RESULTS FOR NUMERICAL EXAMPLE.

$C_{ph}(\ell_1) = 120$	$C_{ph}(\ell_2) = 500$	$C_{ph}(\ell_3) = 120$
$C_{ph}(\ell_4) = 0$	$C_{ph}(\ell_5) = 110$	$C_{ph}(\ell_6) = 500$
$C_{ph}(\ell_7) = 120$	$C_{ph}(\ell_8) = 110$	$C_{ph}(\ell_9) = 540$
$C_{ph}(\ell_{10}) = 400$	$C_{ph}(\ell_{11}) = 570$	$C_{ph}(\ell_{12}) = 470$
$C_{ph}(\ell_{13}) = 270$	$C_{ph}(\ell_{14}) = 470$	$C_{ph}(\ell_{15}) = 270$

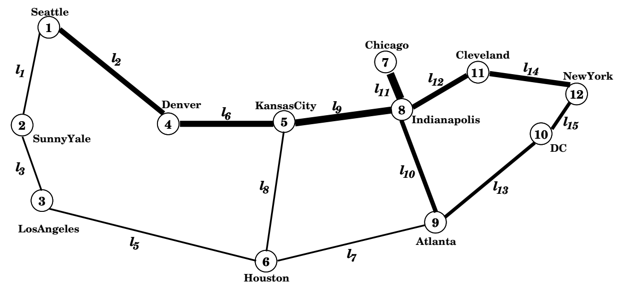

Fig. 6. Abilene backbone network.

VI. CONCLUSIONS

The two heuristics proposed in this paper can be categorized as an off-line Traffic Engineering tool (**LaRDA**) and an online resource optimization tool (**LaRRA**). They solve problems that can be modeled as a multi-commodity network flow optimization. Lagrange relaxation is a common technique to solve these problems. Our formulation minimizes the cost of the total network and includes the delay constraints of the traffic demands for each active Class-Type and also fault tolerance considerations. Both proposed heuristics have polynomial complexity and provide bounds on the value of the optimal objective function, which can be used to evaluate the accuracy of the results. The proposed heuristics are efficient polynomial algorithms with a comprehensive approach to network planning.

REFERENCES

[1] J. Boyle et al., *Applicability Statement for Traffic Engineering with MPLS*, IETF, RFC 3346, August 2002.

[2] A. Feldmann and J. Rexford, "IP Network Configuration for Intradomain Traffic Engineering," *IEEE Network*, vol. 15, no. 5, pp. 46–57, September/October 2001.

[3] P. Aukia et al., "RATES: A Server for MPLS Traffic Engineering," *IEEE Network*, vol. 14, no. 2, pp. 34–41, September/October 2000.

[4] B. Fortz and M. Thorup, "Internet Traffic Engineering by Optimizing {OSPF} Weights," in *Proceedings of IEEE INFOCOM'00*, Israel, 2000, pp. 519–528.

[5] M. Kodialam and T. V. Lakshmann, "Minimum Interference Routing with Applications to Traffic Engineering," in *Proceedings of IEEE INFOCOM'00*, Israel, 2000, pp. 884–893.

[6] F. Le Faucheur, *Russian Dolls Bandwidth Constraints Model for Diff-Serv-aware MPLS Traffic Engineering*, IETF Internet Draft, Work in Progress, March 2004.

[7] F. Le Faucheur et al., *Requirements for Support of Diff-Serv-Aware MPLS Traffic Engineering*, IETF, RFC 3564, July 2003.

[8] S. De Cnodder et al., *Backup Record Route for Fast Reroute Techniques in RSVP-TE*, IETF Internet Draft, Work in Progress, February 2002.

[9] M. L. Fisher, "The Lagrangian Relaxation Method for Solving Integer Problems," *Management Science*, vol. 27, no. 1, pp. 1–18, January 1981.

[10] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[11] G. Cornuejols, M. L. Fisher, and G. L. Nemhauser, "Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms," *Management Science*, vol. 23, no. 8, pp. 709–808, April 1977.