Service Planning in Multi-Layer Networks Considering Physical Constraints

Shu Zhang, Lothar Kreft and Ulrich Killat
Institute of Communication Networks, Hamburg University of Technology
Email: s.zhang, kreft, killat@tu-harburg.de

***Abstract*—In the daily work of network operators, some traffic engineering tasks are often encountered, e.g., to create new logical links over the physical layer considering the efficient utilization of network resources; to establish new end-to-end paths across the network with minimum cost in order to support emerging data transfer services; to install physical disjoint paths for some critical services where fault tolerance is desired, etc. Since these tasks are by nature interrelated, we propose an integrated optimization framework to solve them as a unified planning problem. Both an Integer Linear Programming model and a Simulated Annealing based optimization method are discussed in this paper. Because optimization in multi-layer networks is known to be much more complicated than that in a single layer, special care has been taken in our model to alleviate the scalability problem. The framework has been implemented as a commercial tool for traffic planning. The numerical tests have shown that the corresponding tasks in real scale network can be efficiently handled.**

***Keywords*-Physical Disjoint; SRLG; Traffic Engineering; ILP; Simulated Anealing**

I. INTRODUCTION

One of the major challenges in short term network management is to establish a number of new end-to-end paths with dedicated resource assignment for the emerging request of data transfer services, using the currently available spare resources in the network. A network operator usually does not handle each service directly. Instead, a fixed path with a bulk of bandwidth allocated at each hop is provided to an aggregation of individual data transfer services with some common properties, e.g., same source and destination, similar QoS requests, and same protection mechanism. From the operator's point of view, the aggregation of services is considered as an abstract *demand* to be routed across the network.

The state of the art solution is to calculate a minimum-cost solution at the moment for each upcoming demand using a shortest path algorithm. However, such a greedy approach is known to be sub-optimum in case of multiple demands, because the demands are interrelated due to the competition for common network resources. The situation becomes even more complicated when multiple network layers are taken into consideration. In practical solving approaches nowadays, the consideration of inter-layer relations is mostly intuitive or based on personal experiences of the planner. This may result in worse solutions than planning in a single layer only. Furthermore, some critical user requests could be left unguaranteed. A typical problem raised by network operators is to ensure physical disjoint paths for the protection of important services in a multi-layer setting. This problem and our solution based on a two-layer reference model will be extensively discussed in this paper.

A great number of researches have been carried out on multi-layer optimization problems, typical works includes [1][2][3] in which multi-layer problems are implicitly discussed in the background of WDM networks, and [4][5] where general purpose multi-layer planning models are suggested. One common conclusion is that multi-layer optimization is much more complicated than single layer ones, and to strictly model multi-layer data structures may result in prohibitive scale when dealing with practical problems. In this paper, we propose an approach which reduces the multi-layer model as much as possible into a single layer one. We will formulate a core TE model for the standard single layer planning problem, where the critical multi-layer features appear as extra constraints, and the non-critical features are shifted into heuristic algorithms executed before or after the main TE process.

The rest of the paper is organized as follows. Section II introduces the problem settings and the Integer Linear Programming (ILP) formulation of our Traffic Engineering model. In Section III, a greedy planning method and its extension to a Simulated Annealing (SA) based method are presented as alternative heuristic solutions. Some numerical results in solving the ILP and the heuristic models for a test scenario are presented in Section IV. The conclusions are given in the last part.

II. THE PROBLEM DEFINITION AND THE ILP MODEL

A. Problem Setting

Let's consider the task to route a set D of end-to-end demands (with given resource requests) across a network described by a graph $G(N, E)$, where N is the set of nodes and E is the set of links. The objective is to find a minimum-cost solution where all constraints are held. Without loss of generality, our multi-layer network model is defined by following properties:

1) There are two layers in the network, physical layer $G_{phys}(N_{phys}, E_{phys})$ and logical layer $G_{log}(N_{log}, E_{log})$. Since logical nodes are a subset of physical nodes($N_{log} \subseteq N_{phys}$), we define $N_{phys} = N$;

2) The logical layer occupies a part of the physical resources. Therefore, there are some remaining free resources at both physical and logical layer.
3) Free resources in the logical layer can be directly used to route an end-to-end demand, while free resources in the physical layer must be converted into logical links before being used to route any demand.
4) The routing of each logical link in the physical layer is known; new logical links can be arbitrarily created when there are enough resources in each of its physical hops.
5) The settings of all existing logical links and routed demands remain constant. The capacity of logical links, as well as the routing of demands and logical links cannot be changed.

Note that the last property originates from the practical request of the network operators. The purpose of this conservative constraint is to make sure that no active services could be disturbed due to the accommodation of new demands. Consider another extreme case: Free reconfiguration of all logical links is allowed. In this case, we can setup an analytical model where all free resources in logical links are returned to the corresponding links at the physical layer. After this step, all logical links can be safely removed from the graph since they can no longer influence the routing decisions. Finally, the optimization will be carried out in a topology identical to the physical network, and thus becomes equivalent to a single layer TE problem. After the optimization, we only have to modify the related logical links in the original network according to the solution. With this approach, the sub-optimality of resource utilization due to the ”bundle effect” is not an issue, and may therefore result in better resource effectiveness than our definition above. But, such kind of solution may require a large number of reconfigurations, which is a tedious task and in many cases a major source of error.

In the following part of this paper, we will focus on our problem setting with the properties presented above, in a network G defined as follows.

$$\begin{aligned} G &= G_{phys} \cup G_{log} = G(N_{phys} \cup N_{log}, E_{phys} \cup E_{log}) \\ &= G(N, E_{phys} \cup E_{log}) \\ &= G(N, E),\ E = E_{phys} \cup E_{log} \qquad (1) \end{aligned}$$

This equation explains our attempt to convert most multi-layer optimization features into a single layer model, as shown in Fig.1. Both logical and physical links in Fig.1a which have spare resources at the moment are represented by an *abstract* link in Fig.1b, with capacities equal to their spare resources. In the real operation, spare resources on a physical link must be organized into logical link(s) to be eligible for the routing of demands. However, since the operation of creating a new logical link on a selected physical segment is not an optimization issue, it is taken off from our optimization model without compromising the optimality. Here, we consider the spare resources also as *abstract*, and no longer differentiate between physical and logical links that can eventually be used to support demands.

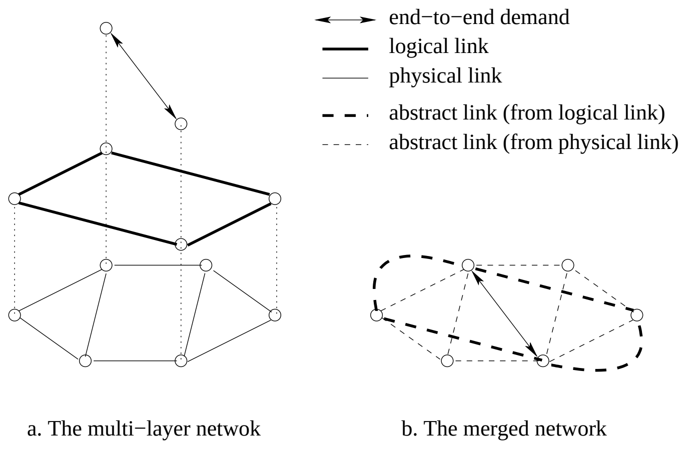
end−to−end demand
logical link
physical link
abstract link (from logical link)
abstract link (from physical link)
a. The multi−layer netwok
b. The merged network

Figure 1. Merge of the physical and logical layer

Note that both Eq.1 and Fig.1 are missing the information of the routing of logical links over physical links, which must be specially modeled.

Following the requests of network operators, we define 2 types of end-to-end demands:

1) Type 1 (D_1): requiring a single end-to-end path with dedicated resource allocation.
2) Type 2 (D_2): requiring a pair of physical disjoint end-to-end paths with dedicated resource allocation along both paths, so that any single failure in the physical layer can be tolerated.

The objective of the optimization is to find a routing solution for each demand, while the link load does not exceed the limit of the available capacity, and the cost due to resource consumption is minimized. In our planning model, all available free resources at any layer become resources in the abstract links of the merged network, based on which a traffic engineering algorithm is carried out. Eventually, the creation of logical links is accomplished according to the results of the optimized routing of demands.

B. The Routing of Demands

In order to establish an end-to-end connection for type 1 demand, a set of flow continuity equations are established. We define a set of binary variables:

$$x_1(i,e) = \begin{cases} 1 & \text{if demand } d \in D_1 \text{ traverses link } e \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

Assume $u, v \in N$ are the end nodes of demand i, and $e(m,n)$ denotes a directional link e from node m to n, then the flow continuity constraint is as follows:

$$\delta_{u,j} + \sum_{e(m,j)\in E} x_1(i,e) = \sum_{e'(j,n)\in E} x_1(i,e') + \delta_{v,j},\ \forall j \in N$$
$$\delta_{a,b} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Eq.3 means that the traffic entering any node j must be equal to that leaving the node, with the exception at the source and destination nodes of the demand.

For type 2 demands, the flow continuity equations are in principle the same. Here, a pair of disjoint paths for each demand is required. We will show that the physical disjointness in the multi-layer model can be well modeled by disjoint conditions of nodes and Shared Risk Link Groups ($SRLG$) [8][9][10] in the layer-merged model.

While node disjoint condition is obvious, an $SRLG$ may originate from two cases in a multi-layer network. The first case is due to the routing of logical links over the same physical link, as shown in Fig.2a. A single event which destroys the physical link L will also destroy logical link P_1 and P_2. Since they share the risk of the same event, we state that the set of links $\{L, P_1, P_2\}$ forms a shared risk link group S. Consider disjoint paths calculation: If one path traverses one of the links in a risk group, the other path should avoid taking any of the links in the same group. Given the routing of all logical links, we can find as many risk groups as the number of physical links in the network, each containing a physical link and all the logical links routed over it. These groups can be obtained by a deterministic analysis process, denoted as *routing related groups*.

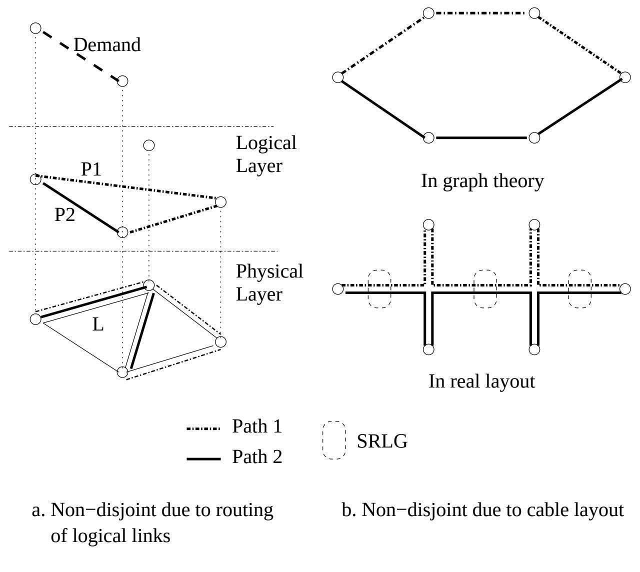

Figure 2. Situations of non-disjoint paths

The second case originates from cable layout. As shown in Fig.2b, although a pair of disjoint paths can be calculated, it may still be risky due to the layout of the physical links. I.e., because cables can be placed in the same bundle or same duct, a single event that destroys one cable should also destroy all others in the same place.

Considering both cases in the optimization model, a preprocess before running the main optimization is required: firstly, an analysis should be made to find out all such bundles/ducts, referred to as a *risk area*; then the *routing related groups* traversing the same risk area are merged to become an SRLG. Note that any standalone physical link is considered as an SRLG with only one physical link. With this model, the physical disjoint constraint in a multi-layer problem is converted to the equivalent condition in the merged single-layer network: the two paths of a type 2 demand should not traverse the same SRLG.

To establish a pair of paths for every type 2 demand, we define a similar binary decision variable as Eq.2:

$$x_2(i,e,p) = \begin{cases} 1 & \text{the } p\text{th path of } d(\in D_2) \text{ traverses } e \\ 0 & \text{otherwise} \end{cases}$$
$$p \in \{1,2\} \quad (4)$$

Under this definition, the flow continuity condition is very similar Eq.3, with the same equation set for every $p \in \{1,2\}$, while the SRLG disjointness of the paths is guaranteed by Eq.5:

$$x_2(i,e,p) + x_2(i,e',p') \le 1 \quad (5)$$
$$\forall i \in D, \forall p,p' \in \{1,2\}, p \ne p', \forall e,e' \in S, \forall S$$

The above constraint means that $x_2(i,e,p)$ and $x_2(i,e',p')$ cannot both be 1, which implies the path p and p' of the demand i must not traverse links in the same SRLG S. However, Eq.5 does not check disjointness of paths at each nodes. The following two equations ensures node disjointness, where the variable $u(i,n,p)$ tracks the intermediate nodes of a path (see Eq.6), and the node-disjointness constraint (see Eq.7) is similar to Eq.5.

$$u(i,n,p) \text{ is binary, }, \quad \forall i \in D_2, n \in N, p \in [1,K]$$
$$x_2(i,e(m,n),p) \le u(i,n,p), \quad \forall i \in D_2, e \in E, p \in [1,K]$$
$$n \text{ is not an end node of } i \quad (6)$$
$$\sum_{p\in\{1,2\}} u(i,n,p) \le 1, \quad \forall i \in D_2, n \in N \quad (7)$$

Note that the SRLG and node-disjointness equations discussed in this section are not restricted to a pair of disjoint paths, the formulation can be naturally extended to the K disjoint paths model.

C. Other Constraints and the Optimization Objective

1) Link utilization: Link utilization should not exceed capacity, where $R(i)$ is the resource required by demand i, and $Cap(e)$ indicates the spare capacity at link e.

$$\sum_{i_1\in D_1} x_1(i_1,e)\cdot R(i) + \sum_{i_2\in D_2, p\in\{1,2\}} x_2(i_2,e,p)\cdot R(i_2)$$

$$\leq Cap(e),\ \forall e \in E \tag{8}$$

2) Objective: As in standard TE problems, we wish to minimize the total cost to support the demands. Let $Price(i, e)$ indicate the cost of demand i taking a unit of resource at link e, then the objective function can be formulated as:

$$\begin{aligned}&Minimizing : Cost\\ &Cost = \sum_{i_1 \in D_1, e \in E} x_1(i_1, e) \cdot R(i) \cdot Price(i_1, e) +\\ &\sum_{i_2 \in D_2, e \in E, p \in \{1,2\}} x_2(i_2, e, p) \cdot R(i) \cdot Price(i_2, e)\end{aligned} \tag{9}$$

With this optimization objective and all above constraints, the default model to solve the TE problem has been established. It is a linear problem and can be solved by an LP solver. Since we have made no compromise on any feature, the optimum solution of the problem can be obtained.

3) Relaxations: For the default model, all the constraints hold strictly. If there are any violations, e.g., the network resource is not sufficient to support all demands, or a pair of strictly disjoint paths does not exist for some type 2 demands, the result of the optimization will be "infeasible". Practically, a planner may wish to know more. One of the typical FAQ is: We know it is hard to keep all constraints, but if we tolerate some violations, what can still be achieved?

Here we discuss two kinds of tolerances. The first one is to allow some demands eventually be left unrouted. Define a binary variable $e_x(i)$ as in Eq.10, and a punishment cost $C_{ex}(i)$ to indicate the increment of total cost if demand i cannot be routed.

$$e_x(i) = \begin{cases} 1 & \text{if demand } i \text{ is routed} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Now, the flow continuity condition (Eq.3) should be slightly modified, so that the incoming/outgoing traffic is no longer guaranteed to be 1 at the end nodes of each demand. Instead, it depends on the value of $e_x(i)$:

$$\begin{aligned}\delta_{i,u,j} + \sum_{e(m,j) \in E} x_1(i, e) &= \sum_{e'(j,n) \in E} x_1(i, e') + \delta_{i,v,j},\\ \forall j \in N,\ \delta_{i,a,b} &= \begin{cases} e_x(i) & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}\end{aligned} \tag{11}$$

The other tolerance is related to the physical disjointness. We define two integer variables: $e_s(i, S)$ indicates that the paths of demand i traverse the same SRLG S, and $e_n(i, n)$ indicates the paths of demand i traverse the same node n, both taking 0 for no violation and positive integer values for so many times of violation. Besides, punishment costs $C_{es}(i, k)$ and $C_{en}(i, n)$ represent the increment of cost when these kinds of violation happen. We need to modify the SRLG and node disjoint constraints (Eq.5, 7) as follows:

$$x_2(i, e, p) + x_2(i, e', p') \quad \leq \quad 1 + e_s(i, S), ... \tag{12}$$

$$\sum_{p \in \{1,2\}} u(i, n, p) \quad \leq \quad 1 + e_n(i, n), ... \tag{13}$$

In the objective function, we set that each violation brings an extra punishment cost in addition to the regular cost. Therefore, the optimization of minimizing the objective can proceed in the direction of reducing violations too. The formulation is as follows:

$$\begin{aligned}&&&Minimizing : Cost + CostT\\ Cost &= &&\text{... (as defined in Eq.9)}\\ CostT &= &&\sum_{i \in D_1 \cup D_2} (1 - e_x(i)) \cdot C_{ex}(i)\\ &+ &&\sum_{i \in D_2} \left[\sum_{k \in S} e_s(i, k) \cdot C_{es}(i, k) \right.\\ &+ &&\left. \sum_{n \in N} e_n(i, n) \cdot C_{en}(i, n) \right]\end{aligned} \tag{14}$$

The values of C_{ex}, C_{es} and C_{en} are very critical to the result of the optimization. The relationship between these punishment costs indicates the tradeoffs among several factors: to support more demands, to save cost, as well as to reduce the amount of rule violation. In our model, we have made the clear setting:

$$C_{es} \approx C_{en} \gg C_{ex} \gg Cost \tag{15}$$

Here, $Cost$ is the regular cost due to resource utilization. This setting implies that respecting the disjoint condition is most important. If we set C_{es} and C_{en} to $Infinite$, or remove the tolerance terms in Eq.12 and Eq.13, then no violation is allowed. Under this principle, a decision which can accommodate more demands is always better than any other solution with less regular cost but also less demands being routed.

III. THE SIMULATED ANNEALING (SA) MODEL AS ALTERNATIVE SOLUTION

In the previous section, we have modeled every feature of the optimization problem as an ILP. Therefore, theoretically the optimum solution can always be obtained. However, the disjoint routing problem with SRLG constraints is proved to be NP-complete [9], and our numerical results that will be introduced in the next section also tend to confirm this property. As an alternative, we introduce our basic greedy algorithm for the same planning problem; then, the greedy algorithm is taken as the core element of a meta heuristic model, for which we chose the Simulated Annealing (SA) model. Since it aims at searching for a satisfactory solution rather than the optimum one, the SA approach can efficiently avoid the difficulties of an NP-complete problem.

A. The greedy algorithm

As discussed in the introduction, a straightforward solution to plan multiple demands in a network is to route all demands sequentially. This is a fully deterministic process, which can be expressed by the pseudo code of Algorithm 1.

Algorithm 1 Greedy TE Algorithm (GTA)

```
Given a fixed sequence $S$ of demands $D(i), i = 1..N_S$
for $i = 1$ to $N_S$ do
  for each link $l$ in the network do
    if $l$ has sufficient resource to accommodate $D(i)$ then
      set the cost of $l$ as its cost for $D(i)$
    else
      set the cost of $l$ as $Infinite$
    end if
  end for
  calculate a minimum-cost solution $sol(i)$ for $D(i)$
  if $sol(i)$ exists then
    record $sol(i)$ in the solution set $Sol$
    for each link $l$ in $sol(i)$ do
      add the cost of $l$ to cost $C$
      update resource utilization at $l$
    end for
  else
    record no solution for $D(i)$ in $Sol$
    add the punishment cost to $C$
  end if
end for
Return the solution set $Sol$ and cost $C$
```

Here, if $D(i)$ is type 2, then a pair of paths with minimum cost sum should be calculated. If the SRLG condition is not considered, know methods like Suurballe's algorithm [6][7] can guarantee the optimum solution. To the best of our knowledge, no heuristic algorithm has been found to be able to guarantee a minimum cost SRLG-disjoint solution. In our greedy algorithm, the *trap avoidance* algorithm suggested by the authors of [10] is used.

B. The simulated annealing algorithm

Because each shortest path is adaptively calculated according to the available network resources at the moment, the above greedy method is to some extent optimized. The quality of the solution is generally better than the intuitive solution of a human planner. An open issue is that GTA (Algorithm 1) depends on a given sequence of demands. With a different sequence, a different set of paths and different overall cost will be obtained. A simple method to take care of the observation is to repeat the same operation: Randomly modify the sequence, and call GTA; the best solution that occurs in this process is taken as the final solution. We refer to this method as *Random Solution Generation (RSG)*.

However, according to our tests, if we combine GTA with some well-known meta heuristics, solutions with better quality (more demands accommodated, less cost) than that of RSG can be obtained in the same calculation time. Here, we present our model with simulated annealing which helps to decide on a suitable sequence for GTA. The pseudo code is shown in Algorithm.2.

Algorithm 2 Simulated Annealing based TE

```
Start with a current sequence $S$ and cost $C \leftarrow GTA(S)$
$S_{best} \leftarrow S$, $C_{best} \leftarrow C$
for reset $round = 0$ to $R$ do
  $S \leftarrow S_{best}$, $C \leftarrow C_{best}$
  for $schedulestep = 1$ to $N$ do
    sequence $S' \leftarrow neighbor(S)$
    $C' \leftarrow GTA(S')$
    if $C' < C$ then
      $S_{best} \leftarrow S'$, $C_{best} \leftarrow C'$
    end if
    the current temperature $t \leftarrow T(i)$
    if $P_{trans}(C, C', t)) > random()$ then
      $S \leftarrow S'$, $C \leftarrow C'$
    end if
  end for
end for
Return $S_{best}$ and $C_{best}$
```

The major functions in the algorithm are:

1) The function $GTA(S)$ is the greedy method Algorithm.1 with the given demand sequence S. The cost C of its solution is then passed to the simulated annealing process.
2) The SA algorithm will reset for R times. At the beginning of each reset, the current state is set to the best solution obtained so far, i.e., the sequence which brings the lowest overall cost.
3) The function $neighbor(S)$ is designed to move a fraction of randomly chosen demands in the sequence to the front of the modified sequence. Therefore, the new sequence S' is similar to the original S, and theoretically the whole solution space can be explored by this operation without preference of any specific pattern.
4) There will be N steps till the temperature drops from the initial T_{max} to 0. The function $T(i)$ controlling the temperature dropping according to the time i is referred to as a *cooling schedule*. Here, an exponential schedule $T(i) = \alpha^i T_{max}, 0 < \alpha < 1$ is chosen based on our tests.
5) If a neighbor state S' is better (with lower cost) than its original state S, then a state transition to S' will definitely take place. When S' is worse than S, the key idea of *simulated annealing* is to allow the transition according to probability, so that the searching process may have chances to let the current state move out from the *local optima*. The probability of a transition

to a worse state is reduced with the dropping of temperature, therefore the state tends to stabilize around some good solutions. When the temperature reaches 0, the SA becomes a pure greedy algorithm. According to our tests, the exponential transition probability formula suggested by Kirkpatrick et al.[11] has shown good performance, i.e., $P_{trans}(C, C', t) = exp(\frac{C-C'}{K_B t})$.

According to our tests, it is capable of obtaining a satisfactory solution within much less time than that required by the ILP model (Section IV), and the solution quality is also better than that obtained by running the *Random Solution Generation (RSG)* for the same time. Although the final solution obtained by SA is inherently sub-optimum. its quality is significant better than the path-oriented methods (even combined with ILP), in which the paths for each demand are selected from a set of pre-calculated candidates.

IV. Numerical Result

Here we show the optimization result obtained with the topology of X-WiN network [12], which is a German scientific research network with nodes located in its major cities. The physical layer of our test case consists of 54 major nodes and 81 links from X-WiN network. The capacities of links range from 1Gbit/s to 20Gbit/s, the same setting as established in X-WiN. Then, 100 logical links are randomly generated using the resource of physical links, which organizes 70% to 80% of the physical resources into the logical layer. Finally, 20% to 80% (random even distribution) of the total capacity at each logical link is marked as occupied to emulated the current network usage.

A test has been carried out to show the influence of SRLG conditions (in the above model, SRLGs are only due to logical links routed over the same physical link). We repeatedly generate type 2 demand with random source and destination nodes. The resource request of such demands are set to 0 so that resource shortage in the network is not a problem. At first, by ignoring all SRLG relationship and only considering the pure graph information like Fig.1b, we use Suurballe's algorithm to find a pair of disjoint paths for each demand. This is roughly what a planner can do in a single-layer TE model. Then, we check if such a solution violates the SRLG-disjoint condition. In our experiment of 10 thousand times random demands generation, 74.2% solutions obtained in this way are in fact non-disjoint, i.e., the 2 paths in a solution traverse at least one common physical link.

Now, we test the different solving approaches of the TE model. To avoid the mixed effect of performance measuring, the test is set to find a minimum cost solution for x type 2 demands, each with randomly generated source and destination. Violation of the disjoint condition is not allowed in this test. The resource requests of each demand are randomly generated and evenly distributed. The range of distribution is adjusted each time to let the overall demand slightly hit the bottleneck of network resources, i.e.: In the solution obtained by simulated annealing, roughly 90% of the demands can eventually be routed. Then, the result is compared to the greedy solution, as well as the optimum solution obtained by the ILP model. All methods follow the principle of accommodating as many demands as possible (Section II-C3).

The quality of the solutions obtained by the different solving approaches is shown in the following two figures. For ILP solutions, the result is obtained when the gap reaches $\leq$1%, i.e., at most 1% away from the optimum solution. Fig.3 shows the total number of demands routed in the final solution, and Fig.4 shows the comparison of average routing cost for each demands using the greedy solution as reference.

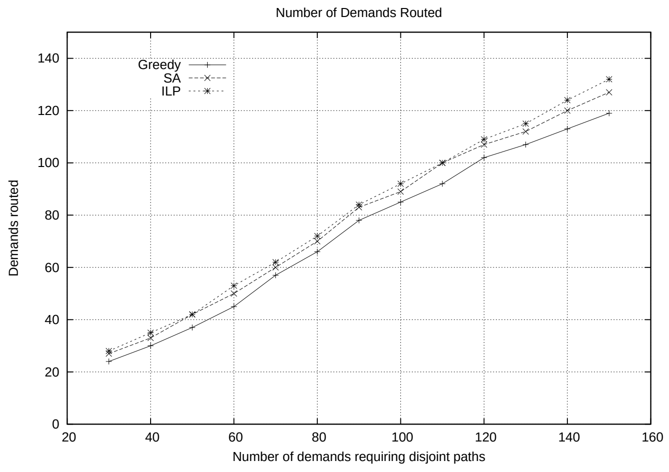

Figure 3. The average cost per routed demands by each method

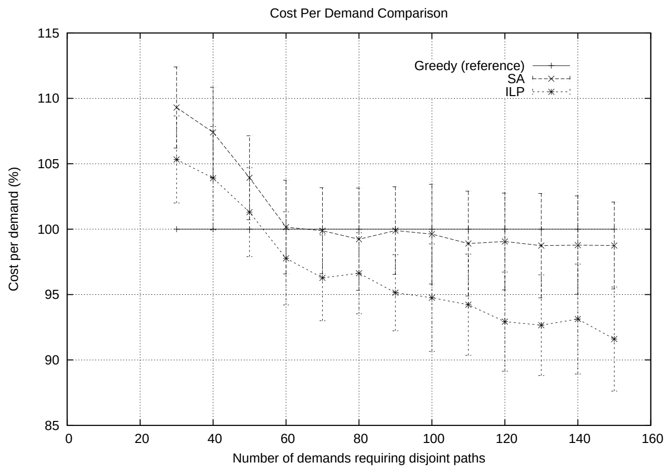

Figure 4. Number of accommodated demands by each method

From the results, it is clear that with the increment of the amount of demands to be considered in one planning task, more demands can be routed with even reduced average cost. The performance of SA is close to that of ILP, and significantly better than that of the greedy solution. Fig.5 shows the solving time of the ILP model and the simulated annealing model with respect to the amount of demands. The solving time of SA is multiplied by 10 to display the curve more clearly. The LP solver is ILOG CPLEX version 11.0.0, and we are using a PC with 3GHz processor and 1G memory.

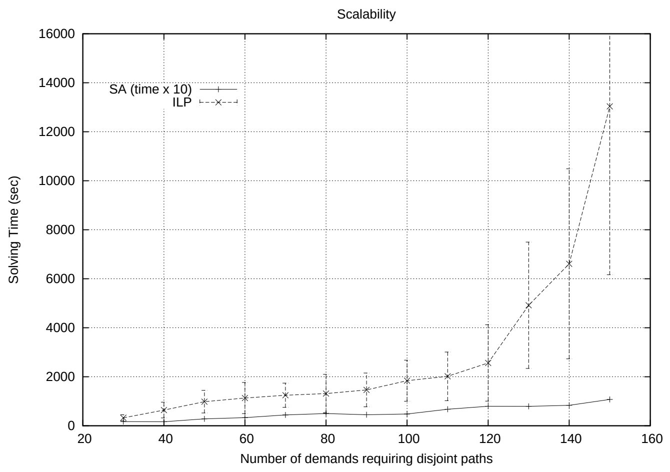

Figure 5. The solving time of the basic TE model

The SRLG disjoint routing problem has been proved to be NP-complete [9]. Indeed, Fig.5 shows a tendency of exponential growth of solving time of the ILP model with the increasing number of demands. The calculation time is also related to whether the bottleneck has been reached, and to which extent. When network resources are adequate, less calculation time (e.g., around 1 hour for 150 demands) is observed in our tests. In comparison, the simulated annealing model has demonstrated a linear increment of solving time due to its fixed cooling schedule and the linear scalability of the greedy sequential method $GTA(S)$ in its core (Algorithm 2). The execution time of the greedy method takes only several seconds within the tested range, therefore it is not shown in the figure.

V. Conclusion

In this paper, we discussed the traffic engineering problem of disjoint route allocation in multi-layer networks, and suggested an analytical model to take the disjoint conditions in physical layer into an integrated optimization approach. Our ILP model is introduced for the basic problem setting, and then extended to support choices like conditional violation of disjoint constraints and best-effort accommodation of demands. Our model has considered the scalability issue often encountered in the multi-layer planning. Besides, a simulated annealing based planning method has also been suggested as an alternative solution approach, aiming at obtaining satisfactory solutions when the solving time of the ILP model is prohibitively high.

References

[1] J. Hu and B. Leida, *Traffic Grooming, Routing, and Wavelength Assignment in Optical WDM Mesh Networks*, In Proc. of IEEE INFOCOM 2004, vol. 1, March 2004, pp.495-501.

[2] K.Zhu, H. Zhu and B. Mukherjee, *Traffic Grooming in Optical WDM Mesh Networks*, Springer Science 2005

[3] E. Kublilinskas and M. Pioro, *An IP/MPLS over WDM network design problem*, in Proc. of International Network Optimization Conference (INOC) 2005, Lisbon, Portugal, 20-23 March, 2005. volume 3, pp.718-725.

[4] S.Orlowski and R. Wessäly, *An Integer Programming Model for Multi-layer Network design*, Konrade-Zuse Center, Berlin. ZIB-Report 04-49 Dec.2004

[5] P. Belotti, A. Capone, G. Carello and F. Malucelli, *Multi-layer MPLS Network Design: The Impact of Statistical Multiplexing*, Computer Networks, Vol.52, Issue 6, pp.1291-1307, April 2008.

[6] J. W. Suurballe, *Disjoint Paths in a Network*, Networks vol.4 (1974) 125-145

[7] J. W. Suurballe and R. Tarjan, *A Quick Method for Finding Shortest Pairs of Disjoint Paths*, Networks, vol.14 (1984) pp.325-336

[8] D. Xu, Y. Xiong, and G. Li, *Trap Avoidance and Protection Schemes in Networks with Shared Risk Link Groups*, IEEE Network, 18(13):36-41, May-June 2004.

[9] J. Hu, *Diverse routing in optical mesh networks*, IEEE Transaction of Communications, Vol.51, pp.489-494, 2003.

[10] Lu Shen, Xi Yang and B. Ramamurthy *Shared Risk Link Group (SRLG) - Diverse Path Provisioning Under Hybrid Service Level Agreements in Wavelength Routed Optical Mesh Networks*, IEEE/ACM Transaction on Networking, Vol.13, No.4, August 2005, pp.918-931.

[11] S. Kirkpatrick, C. D. Gelatt Jr. and M. P. Vecchi *Optimization by Simulated Annealing*, Science, 13 May 1983: Vol.220. no.4598, pp.671-680

[12] German Research Network (Deutsches Forschungsnetz DFN), *The Scientific Network X-WiN*, http://www.dfn.de/xwin/