A Randomized Cost Smoothing Approach for Optical Network Design[1]

Alpár Jüttner[†‡@], Tibor Cinkler[*], Balázs Dezső[†]

[†] Department of Operations Research, Eötvös University, Budapest, Pázmány P. s. 1/C, Hungary H-1117.
[‡] Ericsson Research, Hungary,Budapest Laborc u. 1. H-1071
[*] High-Speed Networks Laboratory, Department of Telecommunications and Media Informatics Budapest University, Technology and Economics Magyar tudósok krt 2, H-1117 Budapest, Hungary
[@] e-mail: alpar@cs.elte.hu

ABSTRACT

In designing infocommunications networks the cost of optical ports and links grows in discrete steps as the capacity is being increased. This cost function is referred to as “step function” or “staged capacity cost”.

If a sequential algorithm is used to design the networks it often results in sub-optimal solution due to the so called ”long path problem”, where the weighted shortest path algorithms rather choose very long paths where such links are chosen where no additional capacity step (and therefore no additional cost step) has to be made.

In this paper we propose and compare methods that perform randomised smoothing of these staged capacity cost functions to allow decomposition of the network design problem to a sequence of weighted shortest path searches, that is the mostly used approach. The problem can be interpreted as an Unsplittable Multi-Commodity Flow Problem with staged capacity costs.

1 INTRODUCTION

When designing an optical network not only the topology but the demand routing and link/port capacities have to be determined as well. For example in an SDH/SONET network the capacities, i.e., the interface speeds take values of 155,52; 622,08; 2 488,32; 9 953,28 and 39 813,12 Mbps, i.e., they always multiply by exactly four. In OTN networks the capacities take values of 2.666, 10.709 and 43.018, i.e. values always multiply by a bit more than four (4,017). Furthermore in OTN and in any other CWDM or DWDM system one or more wavelengths can be used in parallel for demands of larger capacity as well as the number of wavelengths to be used in a WDM system varies in steps of 8, 16, 24, 32, 40, 48, 64, 80, 96, 120 etc. wavelengths per fiber. This shows that for all optical networks we face staged capacity costs.

We propose simple yet efficient approximation approaches to handle this problem. The methods presented and proposed in this paper can be used for green-field design, network extension or configuration (e.g., VPN, leased λ, leased line, etc. services) purposes as will be formulated in Section 1.1. Section 2 presents the known and proposed solution approaches while Section 3 presents the results of evaluations.

1.1 General Network Planning Problem (GNPP)

The network is represented by a directed graph $G_{net} = (N, L)$ with the set N of nodes and the set L of the possible links. The demands are also represented by a directed graph $G_{dem} = (N, D)$ over the same node set N and a function $dem : D \longrightarrow \mathbb{R}_+$. Each demand is represented by an edge $d \in D$, where the tail and the head of d respectively show the source and the target of the demands, while $dem(d)$ is the amount of the demand. Moreover, we are given a monotone increasing load dependent cost function on each link, denoted by $cost : L \times \mathbb{R}_+ \longrightarrow \mathbb{R}_+$, so the cost of establishing a link l with capacity c (or upgrading l to capacity c) is $cost(l, c)$.

Then the task is to assign a route p_d to each demand d in such a way that the obtained configuration minimizes the total cost

$$\sum_{l\in L} cost(l, \mathit{traffic}(l)), \quad \text{where} \quad \mathit{traffic}(l) := \sum_{d:l\in p_d} dem(d). \tag{1}$$

This definition models several usual network optimization problems. Some examples are shown below.

[1]The work reported in this paper was supported by COST293 and by the Mobile Innovation Center, Hungary.

Routing configuration. Let us assume that we are given a network with given link capacities $cap(l)$ and the set of demands we want to carry over this network while keeping the capacity constraints or minimizing the total overload. This problem can be modeled by the framework above by defining the to the cost function.

$$cost(l,t) := \max(0, t - cap(l)) \tag{2}$$

If we want to minimize the number of overloaded links instead of the total overload, we can use the following formula.

$$cost(l,t) := \begin{cases} 0, & \text{if } t \leq cap(l) \\ 0, & \text{if } t > cap(l) \end{cases} \tag{3}$$

Network Design. This case is a green field network planning problem: we want to plan a new network carrying our traffic and we want to minimize the installation costs. Then L will consist of all possible links, while $cost(l,t)$ is defined to be the cost of installing a link of capacity t between the source and the destination of l. As we can choose only from some fix-capacity equipment, the cost will be a step function for each possible link l.

Network Extension. This case is similar to the previous one, but now we have some already existing links, which can be used with no extra installation cost. In this scenario, the cost $cost(l,t)$ of an existing link l with capacity $cap(l)$ are 0 for all $t \leq cap(l)$.

2 SOLUTION METHODS

2.1 Local search

The following simple observation plays a central role in solving GNPP.

Claim 1 *Let us assume, that we have fixed a route for all but one demands, i.e. we fixed p_d for all $d \in D \setminus d'$. Then the optimal route for the last demand d' can be found by searching for a shortest path (using e.g. the Dijkstra algorithm) according to the following auxiliary length function.*

$$len(l) := cost(l, traffic_{d'}(l) + dem(d')) - cost(l, traffic_{d'}(l)), \tag{4}$$

where

$$traffic_{d'}(l) := \sum_{d: d \neq d' \wedge l \in p_d} dem(d). \tag{5}$$

Using the claim above a simple heuristic can be given by starting with an arbitrary solution, then in each iteration removing a route from the configuration and replacing it by the locally best path. We refer this as the *Local Search (LS) method.*

It is easy to see that this method finds the theoretically optimal solution if the cost function is linear in the second variable. However, it works poorly for the two most typical cost functions: the concave and the staged costs, especially for the latter one.

2.2 Cost Function Smoothing (CFS) Algorithm

The main weakness of the LS scheme is that it considers the cost function as a black box and queries its values only for the current traffic on the links. So, it does not take into consideration how much traffic can be still allocated on the link before we actually reach the next stage in the cost function.

A solution proposed by [3] is to smooth the actual cost function by convolving it with a "Gauss-like" function. This smoothed cost function $cost_\delta(l,t)$ should be parametrized, in such a way that for large δ values it provides a very smooth (close to linear for practical t values) function while $cost_\delta(l,t) \longrightarrow cost(l,t)$ as $\delta \longrightarrow 0$.

Then the CFS algorithm is the same as LS with the exception that we use $cost_\delta(l,t)$ instead of the original cost function. We start with a large δ value and decrease it exponentially during the execution of the algorithm.

A concrete example for such a smoothing given in [3] is as follows.

$$cost_\delta(l,t) := (cost(l,\cdot) * h_\delta(\cdot))(t) = \int_{-\infty}^{\infty} cost(l,\xi) h_\delta(t-\xi) d\xi, \tag{6}$$

where

$$h_\delta(t) := \frac{1}{\delta}h(\frac{t}{\delta}), \quad \text{and} \quad h(t) := \begin{cases} 1 - 2t^2 & \text{for } |t| \leq 1/2 \\ 2(|t|-1)^2 & \text{for } 1/2 \leq |t| \leq 1 \\ 0 & \text{for } 1 \leq |t| \end{cases} \tag{7}$$

The advantage of this construction is that a convolution integral is actually a polynomial for constant or linear segments of the cost functions, so it is fast to evaluate for staged of piecewise linear cost functions.

2.3 Randomized Cost Smoothing (RCFS) Algorithm

In this approach is somewhat similar to the Cost Function smoothing but instead of modifying the cost function, we apply a random "uncertainty" when querying its values. Namely we change Equation (4) to the following.

$$len(l) := cost(l, traffic_{d'}(l) + dem(d') + R_\delta) - cost(l, traffic_{d'}(l)), \tag{8}$$

where R_δ is a certain random variable of standard deviation δ. Natural choices for R_δ could include the Gaussian distribution $N(0, \delta^2)$ and the exponential distribution. If R_δ can also take negative values, then $len'(l) := \max(0, len(l))$ should be used in order to avoid negative lengths. Note that Dijkstra algorithm reads the length of each link only once, so this gives a consistent length function in each iteration.

As the *expected value* of $cost(l, t + R_\delta)$ is the same as $cost(l, t)$ smoothed/convolved by the probabilistic distribution function of the random variable R_δ, this approach indeed performs a "randomized smoothing". On the other hand, an expected advantage of this scheme is that similarly to other metaheuristics like Simulated Annealing or the Evolutionary Algorithms, the randomness may provide higher freedom for the algorithms when choosing a replacement of route, thus it may have a higher chance to avoid the local optima that are far from the global optimum.

3 EVALUATION

Due to the space limitation, here we only demonstrate the behaviour of the algorithms on a single network topology but our tests showed that the presented results are quite representative.

The test network (see Fig. 1) is a two connected planar graph consisting of 50 nodes and 84 bidirectional links. It was generated by `lgfgen`, a random graph generator of LEMON [4].

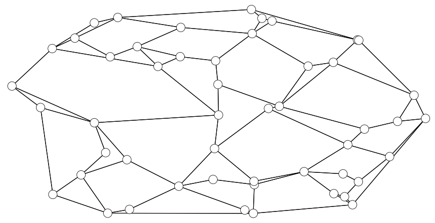

Figure 1: Text network example with 50 nodes and 84 links

The cost function is a step function also proportional to the physical length of the link, i.e,

$$cost(l, t) := length(l) step\_fn(t), \quad \text{where} \quad step\_fn(t) := \begin{cases} 0 & \text{for } t = 0, \\ 1 & \text{for } 0 < t \leq 1, \\ 2 & \text{for } 1 < t \leq 4, \\ 4 & \text{for } 4 < t \leq 10, \\ 8 & \text{for } 10 < t \leq 100, \\ M & \text{for } 100 < t. \end{cases} \tag{9}$$

Two different types of demand matrices were chosen for the tests.

Homogeneous demands. In this scenario, we have a homogeneous traffic matrix, i.e. we have the same amount of traffic between any pair of nodes.

Big trunks. In this scenario only 245 (out of the 2450 possible) random pairs of nodes was chosen as traffic sources and destinations but the demands are 10 times larger than in the previous case.

Table 1: Cost obtained by the different algorithms

	Homogeneous demands	Big trunks
SHORTEST	91101	89575
LS	83371	72601
CFS	59890	56158
RCFS/Gauss	86722	69444
RCFS/Exp	63342	52104

We ran LS, CFS, RCFS methods on these examples, and we also computed the cost of the simple shortest path routing (SHORTEST) as a reference. In case of RCFS, both Gaussian and exponential randomization has been tested. The costs of the obtained solutions are presented in Table 1.

The algorithms were implemented in C++, heavily based on the LEMON library [4]. The tests were made on a Laptop equipped with 2 GHz Centrino Duo processor and running Linux (OpenSuse 10.2) operating system.

The running time of SHORTEST (less 0.1 second) and LS (a few seconds) was obviously much less than those of CFS and RCFS. Actually the running time of these algorithms depends on the number of iteration they do. Thus the number of iterations was chosen in such a way that the resulted running time was around 40 seconds. The other parameters were tuned to provide the best results.

It is worth mentioning that RCFS takes 2 000 000 iterations within 40 seconds while CFS takes only 400 000 within the same time. This is because the cost function calculation must be performed at each iteration of the algorithms thus it turns to be the most resource consuming part of the algorithms. Thus the easier calculation of the randomized cost function makes the running time of an iteration of RCFS much smaller than those of CFS.

Considering the two versions of RCFS, the results show that the Gaussian version performed quite poorly. In fact in case of homogeneous demands, this version is even outperformed by the simple LS method. The possible explanation of the better performance of the exponential distribution based version is that this cost randomization can be interpreted as a stochastic prediction of the amount of traffic that will be allocated on the link in the subsequent iterations.

Comparing CFS and RCFS, we obtained that for an homogeneous traffic matrix the winner is CFS, while for uneven traffic with big trunks RCFS outperforms CFS. The reason for this is that for an homogeneous traffic matrix the traffic can be almost continuously distributed on the links and the cost function smoothing seems to perform really well. On the other hand if there are fewer but bigger unsplittable traffic flows, the problem becomes a rather combinatorial packing problem, and in this case the RCFS method is able to scan a larger portion of the search space.

4 CONCLUSION

This paper presented a novel approach for the Unsplittable Multi-Commodity Flow Problem with the staged capacity costs. This class of problems plays an important role in solving optical network planning problems. Our numerical tests showed that for uneven traffic matrices — which is the typical case in practice — the proposed method performs better than the tested reference methods.

References

[1] R.K. Ahuja, J.B. Orlin. *A Capacity Scaling Algorithm for the Constrained Maximum Flow Problem*, Networks 25 (1995) pp. 89-98.

[2] T. Cinkler, A. Kern, I. Moldován and Gy. Sallai. *Dimensioning Transport Networks for VPNs over Capacities with Stepwise Costs*, NETWORKS 2006. 12^{th} International Telecommunications Network Strategy and Planning Symposium, 2006.

[3] Per Lindberg. *Network optimisation with successive smooth approximation*, 11th Nordic Teletraffic Seminar, Stockholm, 1993.

[4] LEMON: A C++ Library for Efficient Modelling and Optimization in Networks, `http://lemon.cs.elte.hu`