Minimizing the Number of Congested Links in OSPF Routing

Mohammed H. Sqalli, Sadiq M. Sait, and Syed Asadullah

Computer Engineering Department
King Fahd University of Petroleum & Minerals
Dhahran 31261, Saudi Arabia
E-mail: {sqalli,sadiq,sasad}@kfupm.edu.sa

Abstract— **Efficient network utilization using available resources is the main goal of traffic engineering and routing is the core criteria which regulates traffic over Internet links. Open Shortest Path First (OSPF) is a routing protocol which is widely used in the industry and uses the link weights as its routing metric. Optimizing these link weights leads to efficient routing and better network utilization. Setting weights on links for given traffic demands such that congestion can be avoided is an NP-hard problem. In this paper, Tabu Search, a non-deterministic iterative heuristic is applied to this problem using two different cost functions proposed in the literature. Moreover, we present the results for two additional performance metrics *viz.* Number of Congested Links and Percentage of Extra Load. This provides the network designer with more flexibility to optimize desired parameters based on the requirements. Our results show superior performance of Tabu Search over other heuristics.**

Keywords: *Traffic Engineering, Routing, Open Shortest Path First (OSPF), OSPF Weight Setting Problem, Utilization, Tabu Search.*

I. Introduction

Traffic over the Internet is growing considerably, due to more frequent network expansions [1]. With an increasing trend of web based applications and worldwide spread of the Internet for an increasing number of purposes, a traffic boom in the near future is almost certain. These developments highlight the importance of traffic engineering. Managing traffic flows on the network to most efficiently utilize the available resources and enhance the Quality of Service (QoS) is the main goal of traffic engineering. Routing manages these traffic flows on the Internet. The objective of traffic engineering is to prevent traffic imbalance by efficiently mapping traffic onto available resources.

Routers are network devices that forward packets between source and destination nodes. Due to redundant connectivity between nodes, multiple paths exist between a given source and destination pair. The task of a router is to find shortest paths for each source to destination. Routers take the decision of the best path the packet should take by maintaining a routing table which is periodically updated by exchanging network state information with other routers.

The Internet, because of its huge size, is complex. To reduce its complexity, it is divided into Autonomous Systems (AS). An AS is a collection of networks under the control of one single entity or organization with a common routing policy. There are Interior Gateway Protocols (IGP) for managing routing inside an AS and Exterior gateway Protocols (EGP) for managing routing between ASs [2]. Open Shortest Path First (OSPF) is an Interior Gateway routing protocol that uses the Dijkstra's Algorithm for finding the shortest paths. The routing metric used in OSPF is the link weights where the sum of the OSPF weights on the links in the path gives the cost of that path. The path with the least cost is the shortest path. No standard criteria has been defined to assign these weights. OSPF implementation of Cisco assigns the link weights inversely proportional to the link bandwidth.

In this paper, a problem related to OSPF routing protocol is addressed, namely the OSPF Weight Setting (OSPFWS) problem. This problem is proven to be NP-hard [3]. We have used a Tabu Search (TS) algorithm [4] to solve the OSPFWS problem by using the two cost functions available in the literature. Tabu search has been implemented previously by Fortz and Thorup [3] for the cost function proposed by them and results were reported for cost and maximum utilization. The maximum utilization refers to the utilization of the link that has the highest utilization in the network. In this paper, we extend their work and apply tabu search for OSPFWS problem using the new cost function proposed by Sqalli et al. [5] We also present results for the two additional performance metrics *Viz.* the number of congested links and the percentage of extra load. We further show that the Number of Congested Links can be optimized by using the new cost function.

The rest of the paper is organized as follows; a discussion of the related work is presented in Section II. We state the OSPFWS problem and formulate the cost functions in Section III. The Tabu Search algorithm and related parameters used for OSPFWS problem will be presented in Section IV. This is followed by the experimental results and discussion including comparison with previous related work in Section V. Finally, we conclude in section VI.

II. Related Work

The consideration of even traffic splitting in OSPF weight setting was first done by Fortz and Thorup [3]. They also established that this problem is NP-hard. They initially proposed a local search heuristic and also applied Tabu Search iterative heuristic [2][6] to improve the results of local search. They tested their heuristics on AT&T backbone network and on synthetic networks.

Ericsson et al. [7] proposed a Genetic Algorithm (GA) and used the set of test problems considered in [3]. A hybrid GA was also proposed by them [8] which makes use of the dynamic shortest path algorithm to recompute shortest paths after the modification of link weights. Sridharan et al. [9] developed another heuristic for a slightly different version of the problem, in which the flow is split among a subset of the outgoing links on the shortest paths to the destination IP address.

Sqalli et al. [5] proposed a new cost function based on the utilization and the extra load caused by congested links in the network. The new cost function optimizes the number of congested links in addition to the utilization. They also compared their results with [3][2] and achieved better results in terms of minimizing the number of congested links. They also implemented Simulated Annealing (SA) [5] and Simulated Evolution (SimE) [10] for the Fortz Cost Function (FortzCF) proposed in [3].
Fortz and Thorup further extended their work [11] to a slightly

different problem area *viz.* to find robust solutions taking into account single link failure scenarios. A set of links was considered as critical, and in each iteration one of these links was failed based on the maximum utilization among critical links. The cost of normal topology and the resulting failed topology was averaged and the search was driven to find a solution which minimizes the average cost.

The input to these algorithms for this problem is a network topology, capacity of links and a demand matrix. The demand matrix represents the traffic between each source-destination pair of nodes present in the topology. The methodology for deriving traffic demands from operational networks is described in [12].

III. Problem Statement

The OSPF weight setting (OSPFWS) problem can be stated as follows: Given a network topology and predicted traffic demands, find a set of OSPF weights that optimize network performance. More precisely, given a directed network $G = (N, A)$, where N represents nodes and A represents arcs, a demand matrix D, and capacity C_a for each arc $a \in A$, we want to determine a positive integer weight $w_a \in [1, w_{max}]$ for each arc $a \in A$ such that the objective function or cost function Φ is minimized. w_{max} is a user-defined upper limit. The chosen arc weights determine the shortest paths, which in turn completely determine the routing of traffic flow, the loads on the arcs, and the value of the cost function Φ. The quality of OSPF routing depends highly on the choice of weights. As an example of this problem, Figure 1 depicts a topology with assigned weights in the range $[1, 20]$. One solution for this topology is $(18, 1, 7, 15, 3, 17, 5, 14, 19, 13, 18, 4, 16, 16)$. Each element of this vector represents the weight assigned to the respective arc in this topology. For instance, the first arc (A, B) has a weight of 18, the second arc (A, F) has a weight of 1, etc.

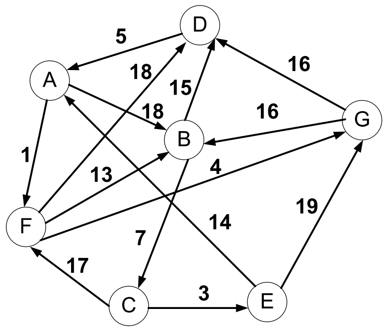

Fig. 1. Representation of a topology with assigned weights.

A. Cost Functions

Using the above notations, the problem can be formulated as a multi-commodity flow problem [3][7] using the following cost function, i.e., FortzCF:

$$minimize \quad \Phi = \sum_{a \in A} \Phi_a(l_a) \tag{1}$$

subject to these constraints:

$$l_a = \sum_{(s,t) \in NXN} f_a^{(s,t)} \; a \in A, \tag{2}$$

$$f_a^{(s,t)} \geq 0 \tag{3}$$

Constraint (2) defines the load on each arc a.

Φ_a are piecewise linear functions, with $\Phi_a(0) = 0$ and a derivative, $\Phi_a^{'}(l_a)$ [3] given by:

$$\Phi_a^{'}(l) = \begin{cases} 1 & \text{for } 0 \leq l/c_a < 1/3, \\ 3 & \text{for } 1/3 \leq l/c_a < 2/3, \\ 10 & \text{for } 2/3 \leq l/c_a < 9/10, \\ 70 & \text{for } 9/10 \leq l/c_a < 1, \\ 500 & \text{for } 1 \leq l/c_a < 11/10, \\ 5000 & \text{for } 11/10 \leq l/c_a < infinity \end{cases} \tag{4}$$

B. Formulation of Cost Function

In this section, we enumerate the steps to compute the cost function Φ for a given weight setting $\{w_a\}_{a \in A}$ and a given graph $G = (N, A)$ with capacities $\{c_a\}_{a \in A}$ and demands $d_{st} \in D$. This procedure is also described in [3]. Two cost functions are discussed in this paper.

When routing is done using OSPF the assigned link weights completely determine the shortest paths, and hence the traffic flows. Based on these traffic flows the partial loads on each arc for a given destination is computed. This is done for all destination nodes. The aggregated partial loads for all destinations on a particular arc gives the total load l_a on that arc. The load on each arc gives us the link utilization on this arc, which in turn gives us a cost given in the equation 1.

The first task is to compute the arc loads l_a resulting from the given weight setting $\{w_a\}_{a \in A}$. The arc loads are computed in the steps shown below. For all demand pairs $d_{st} \in D$, one destination t at a time is considered and partial arc loads $l_a^t \; \forall \; t \in \bar{N} \subseteq N$, are computed. $\bar{N}$ represents the set of destination nodes.

1) Compute the shortest distances d_u^t to t from each node $u \in N$, using Dijkstra's shortest path algorithm [13]. Dijkstra's alogrithm usually computes the distances away from source s, but since we want to compute the distance to the source node t, the algorithm will be applied on the graph obtained by reversing all arcs in G.
2) Compute the set A^t of arcs on shortest paths to t as,
$$A^t = \{(u, v) \in A : d_u^t - d_v^t = w_{(u,v)}\}.$$
3) For each node u, let δ_u^t denote its out degree in $G^t = (N, A^t)$, i.e.,
$$\delta_u^t = \mid \{v \in N : (u, v) \in A_t\} \mid$$
If $\delta_u^t > 1$, then the traffic flow is split at node u to balance the load.
4) The partial loads l_a^t are computed as follows:
   (a) Nodes $v \in N$ are visited in the order of decreasing distance d_v^t to t.
   (b) When visiting a node v, for all $(v, w) \in A^t$, set $l_{(v,w)}^t = 1/\delta_v^t (d_{vt} + \sum_{(u,v) \in A^t} l_{(u,v)}^t)$
5) The arc load l_a is now summed from the partial loads as:
$$l_a = \sum_{t \in \bar{N}} l_a^t$$

The evaluated costs are normalized to allow comparison of costs across different sizes and topologies of networks. We applied the same normalizing scaling factor as introduced by Fortz and Thorup [3][7].

Let Ψ be the cost when all the flow was sent along the hop-count shortest paths and the capacities matched the loads. Let $\Delta(s, t)$ be the hop-count distance between s and t. Ψ is calculated as,

$$\Psi = \sum_{(s,t) \in NXN} (\; D[s, t] \; . \; \Delta(s, t)) \tag{5}$$

The normalized cost function is,

$$\Phi^* = \Phi / \Psi \tag{6}$$

C. New cost function

The cost function proposed by Fortz and Thorup [6] is based on the range of utilization present on each link. This cost function specifically targets at optimizing routing by balancing traffic flows. Sqalli et al. [5], proposed a new cost function based on maximum utilization and extra load on the network. The following equation shows the proposed cost function:

$$\Phi = MU + \frac{\sum_{a \in SetCA} (l_a - c_a)}{E} \tag{7}$$

This function contains two terms. The first term is the maximum utilization (MU) in the network. The second term represents the extra load on the network divided by the number of edges present in the network to normalize the entire cost function. The motivation for the use of this function is to reduce the number of congested links if any in the network.

Optimizing the number of congested links has an advantage that, in case of congestion in a network, the network designer will need to upgrade capacity of fewer links in the network to avoid congestion. If a designer decides on a capacity expansion, it would result in a less expensive upgrade. The first cost function [3] aims at balancing traffic flows over all links and the second [5] aims at minimizing the number of congested links. This gives more flexibility to the designer to choose the cost function that will fulfill the desired objectives. The choice of using one or the other will depend ultimately on the target to be achieved. We discuss the results related to this function in more details in a later section. Results show that this function reduces the number of congested links when the traffic demand is high, with some tradeoff on the maximum utilization. The results of the final solutions with respect to this function are compared with those of the cost function given in [6].

IV. Tabu Search

A. Tabu Search Algorithm

Tabu search (TS) is an iterative heuristic that has been applied for solving a range of combinatorial optimization problems in different fields [4]. Tabu search starts with an initial feasible solution and carries out its search by making a sequence of random moves or perturbations. A tabu list is maintained that stores the attributes of a certain number of recent moves. This list prevents bringing the search process back to already visited states, and hence prevents the search from getting stuck in a local optima. In each iteration, a subset of neighbor solutions called the candidate list is generated by making a certain number of moves and the best move (the move that resulted in the best solution) is accepted, provided it is not in the tabu list. Otherwise, if the said move is in the tabu list, the best solution is checked against an aspiration criterion and if satisfied, the move is accepted. Thus, the aspiration criterion can override the tabu list restrictions. It is desirable in certain conditions to accept a move even if it is in the tabu list, because it may take the search into a new region due to the effect of intermediate moves. The behavior of tabu search heavily depends on the size of tabu list as well as on the chosen aspiration criterion. Different sizes of tabu list result in short-term, intermediate term, and long-term memory components that can be used for intensifying or diversifying the search. The aspiration criterion determines the extent to which the tabu list can restrict the possible moves. If a tabu move satisfies the aspiration criterion, then the move is accepted and tabu restriction is overridden. The structure of the Tabu Search algorithm is given in Figure 2. The detailed description and related references for tabu search can be found in [4].

Ω	:	Set of feasible solutions.
S	:	Current solution.
S^*	:	Best admissible solution.
$Cost$	:	Objective function.
$\aleph(S)$	:	Neighborhood of $S \in \Omega$.
$\mathbf{V}^*$	:	Sample of neighborhood solutions.
T	:	Tabu list.
AL	:	Aspiration Level.

```
     Begin
1.   Start with an initial feasible solution $S \in \Omega$.
2.   Initialize tabu lists and aspiration level.
3.   For fixed number of iterations Do
4.     Generate neighbor solutions $\mathbf{V}^* \subset \aleph(S)$.
5.     Find best $S^* \in \mathbf{V}^*$.
6.     If move $S$ to $S^*$ is not in T Then
7.       Accept move and update best solution.
8.       Update tabu list and aspiration level.
9.       Increment iteration number.
10.    Else
11.      If $Cost(S^*) <$ AL Then
12.        Accept move and update best solution.
13.        Update tabu list and aspiration level.
14.        Increment iteration number.
15.      EndIf
16.    EndIf
17.  EndFor
     End.
```

Fig. 2. Structure of the short-term Tabu Search (TS) algorithm.

B. Tabu Search for OSPFWS

Initial Solution: The initial solution is randomly generated where each link is assigned a random weight between 1 - 20. The idea behind using small weights (1 - 20) is to increase the chance of even splitting due to multiple shortest paths from a node to some destination [3].

Move: The move in our implementation is an assignment of a random weight (value between 1 - 20) to a randomly selected link. For example, if the current solution is $(3, 7, 19, 13, \mathbf{18}, 16, 1, 7, 15, 5, 14, 18, 4, 16)$, a random move can be a change of weight on the link 5 from 18 to 9. This move will take us to a different solution in the search space i.e. $(3, 7, 19, 13, \mathbf{9}, 16, 1, 7, 15, 5, 14, 18, 4, 16)$

Tabu Move: A move is considered tabu if the randomly selected link is found in the tabu list.

Aspiration Criterion: The aspiration criterion is the best cost i.e a move is accepted even if it is tabu only if it meets the aspiration criterion i.e. if it improves the current cost beyond the best cost achieved so far.

Tabu List: The tabu list is a single dimensional array which contains the edge numbers on which recent moves were performed. The topologies used in the experiments are taken from [6] and are of varying sizes, starting from 50 nodes, 148 arcs to 100 nodes, 503 arcs. Hence, a static tabu list size would have not been appropriate. Due to the fact that a move was based on arc weights and the tabu list stored these recent moves, it is intuitive to have a tabu list size proportional to the number of arcs in the network. After initial experimentation and analysis, the tabu list size was set to be the square root of the number of edges in the topology.

Candidate List: The candidate list contains the solution resulting from each random move, each time starting from the original solution. After the initial experimentation, the appropriate size of the candidate list was found to be 10.

Termination criteria: The termination criteria for all simulations was the simulation time. For small test cases (all topologies with 50 nodes), the simulation time was approximately 40 minutes and for large test cases (all topologies with 100 nodes) the simulation time was 60 minutes.

Hardware & Software: All simulations were run on Intel machines with Pentium4, 3.0GHz processor and 1GB RAM. Operating System used was Windows XP. The source code was compiled using Microsoft Visual C++ package.

V. Results

A. Test cases

The cases used in our experiments have been taken from [6]. Same test cases were used in previous works for the implementation of GA, SA, SimE for this problem. Table I shows the characteristics of the test cases. For each test case, the table lists its network type, number of nodes (N), number of arcs or edges (A). The *2-level hierarchical networks* are generated using the GT-ITM generator [14], based on a model of Calvert [15] and Zegura [16]. In hierarchical networks, local access arcs have capacities equal to 200, while long distance arcs have capacities equal to 1000. In *random networks* and *waxman networks* capacities are set to 1000 for all arcs. Further details can be found in [3].

TABLE I
TEST CASES

Test Code	Network type	N	A
h100N280a	2-level hierarchical graph	100	280
h100N360a	2-level hierarchical graph	100	360
h50N148a	2-level hierarchical graph	50	148
h50N212a	2-level hierarchical graph	50	212
r100N403a	Random graph	100	403
r100N503a	Random graph	100	503
r50N228a	Random graph	50	228
r50N245a	Random graph	50	245
w100N391a	Waxman graph	100	391
w100N476a	Waxman graph	100	476
w50N169a	Waxman graph	50	169
w50N230a	Waxman graph	50	230

Table II shows different levels of demand values for six topologies. the demands are denoted as D4 - D12. The value of demand shown in various columns in the table is the total demand on the network for each topology at different demand levels. All these demands were used in our experiments.

B. Performance Metrics

Experimental results have been recorded for the following four performance metrics:

1) Cost
2) Maximum Utilization
3) Percentage of Extra Load
4) Number of Congested Links.

The utilization of the link is the ratio of load on the link to its capacity. If the utilization of the link is more than one, the link is congested. The Maximum Utilization is the utilization of the maximum utilized link in the network. In other words it is the utilization of the link having highest degree of congestion. The extra load on a particular link is the load present in excess to its capacity. If the load on the link is less than its capacity ($utilization < 1$), then extra load on that link is zero. The percentage of extra load is the sum of the extra load present in the network divided by the sum of capacities of congested links. As explained earlier, congested links are the links which have a utilization greater than 1 (i.e., load on the link exceeds its capacity). The statistics for these performance metrics are plotted with respect to the four heuristics GA, SA, SimE and TS using the two cost functions: FortzCF and NewCF. The results for GA [7], SA [5] and SimE [10] have been taken from the literature and are compared to our implementation of TS.

C. Discussion of the Results

In this section we present the experimental results for Tabu Search. We will first discuss the results obtained for the test case r100N403a.

Figure 3 shows the cost curve for r100N403a test case using FortzCF for the above mentioned four algorithms. It can be seen that for lower demands all heuristics perform almost equally well. For higher demands GA performs worse than the other algorithms followed by SA. The results for SimE and TS are comparable however TS outperforms SimE for Demand D10, D11 and slightly for D12.

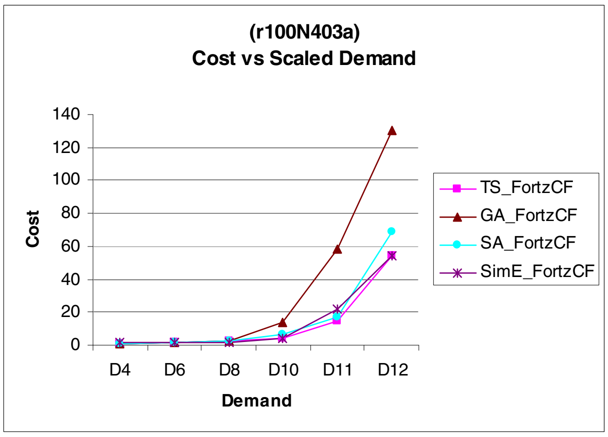

Fig. 3. Cost versus scaled demand on r100N403a network using FortzCF.

Figure 4 shows the comparison of Maximum Utilization in the network for the four heuristics for both cost functions. The results for TS_FortzCF and TS_NewCF are either better or comparable to others for most of the demands. For higher demands SA_NewCF and GA_FortzCF perform worse than other algorithms. For lower demands the results for all heurists are comparable.

The next performance metric is the Number of Congested Links (NOCL). The results for this metric are not reported in literature for GA. Hence, we Present the results for three heuristics in the Figure 5. It can be observed from the figure that the NewCF for all three heuristics minimizes the number of congested links better than the FortzCF. The performance of SimE_NewCF is the best which is followed by TS_NewCF and SA_NewCF.

Finally the results for Percentage of Extra Load are shown in Figure 6. The extra load is a very good measure of the amount of congestion in the network. Results show that SA_NewCF does not perform well for this test case. The results for other heuristics are close to each other while TS_NewCF clearly performs the best. The results for other heuristics are comparable.

We now discuss the results for other test cases. The cost comparison for the case r50N245a shows that TS performs better than

TABLE II
DEMAND TABLE

Demand	r100N403a	r100N503a	w100N391a	w100N476a	h100N280a	h100N360a
D4	23099	33531	16158	21164	1535	4136
D6	34648	50297	24237	31747	2303	6203
D8	46198	67063	32316	42329	3070	8271
D10	57747	83829	40395	52911	3838	10339
D11	63522	92211	44434	58202	4221	11373
D12	69297	100594	48474	63493	4605	12407

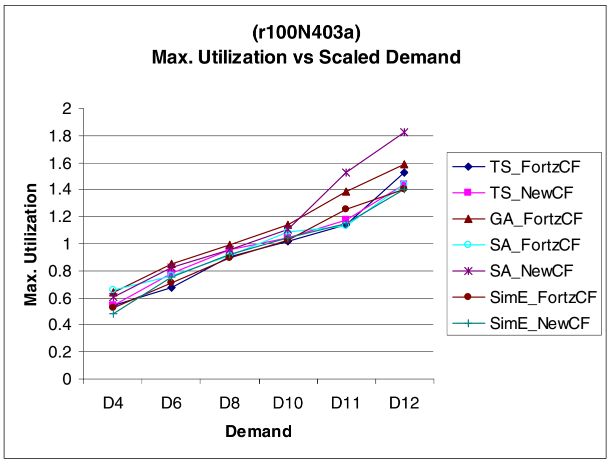

Fig. 4. Maximum Utilization versus scaled demand on r100N403a network using FortzCF and NewCF.

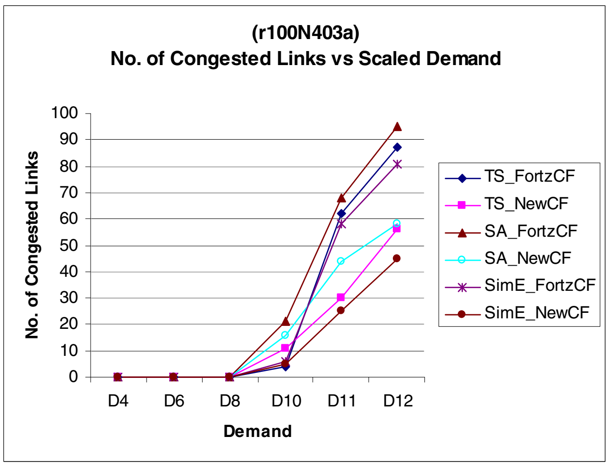

Fig. 5. Number of Congested Links versus scaled demand on r100N403a network using FortzCF and NewCF.

all other heuristics for all demands. The results for w100N391a and w100N476a are comparable for all heuristics. However the results for TS are slightly better than others for medium and higher demands. For the other two test cases, i.e.; h100N280a and h100N360a, the results for TS, GA and SA are comparable and better than SimE.

The NOCL comparison for r50N245a again shows that for all heuristics NewCF performs better than FortzCF. TS has better results than other heuristics for this test case. The results for w100N391a and w100N476a are comparable for all heuristics. The results for h100N280a and h100N360a are comparable for all heuristics while SA_NewCF slightly performs better than others for higher demands in h100N360a.

The results for PXLoad for the test case r50N245a shows best performance from TS_NewCF and TS_FortzCF. The results for w100N476a and h100N360a are comparable for all heuristics. w100N391a shows SimE_FortzCF performing slightly better than others and h100N280a shows NewCF performing better than FortzCF for most heuristics.

The cost comparison shows that Tabu Search was found to perform the best in all test cases, most importantly for higher demands when there is additional load on the network and higher possibility of links getting congested. The comparison of NOCL for all test cases shows better results with NewCF when compared to FortzCF. The results for TS, SA and SimE are comparable in most of the test cases. The results for PXLoad for NewCF is better than or comparable to FortzCF in most of the test cases while also minimizing the number of congested links. This makes the NewCF a very efficient cost function for solving the OSPFWS problem.

VI. CONCLUSION

Tabu Search iterative heuristic is implemented on several networks using the two cost functions. Results for all four performance metrics are reported for both cost functions including two new performance metrics FortzCF. It was found that NewCF better optimizes number of congested links than FortzCF while the results for percentage of extra load are mostly comparable for both cost functions with some exceptions. The results obtained are compared with existing results of genetic algorithm, simulated annealing and simulated evolution for the same problem. Cost comparison shows that Tabu Search was performing better than any other heuristic. With the new cost function also providing results better than or comparable to the Fortz cost function, it provides more flexibility to the network designer to choose between the different cost functions depending on the target to be achieved; e.g., lower maximum utilization, fewer number of congested links, or smaller percentage of extra load.

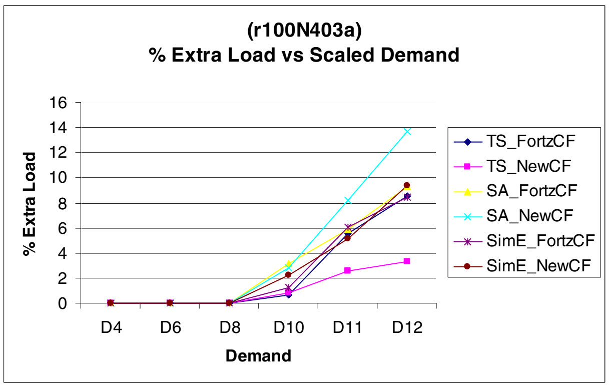

Fig. 6. Percentage Extra Load versus scaled demand on r100N403a network using FortzCF and NewCF.

VII. Appendix

This appendix provides the notation used in the OSPFWS problem formulation.

A. Notations

G	Graph.
N	Set of nodes.
n	A single element in set N.
A	Set of arcs.
A^t	Set of arcs representing shortest paths from all sources to destination node t.
a	A single element in set A. It can also be represented as (i, j).
s	Source node.
v	Intermediate node.
t	Destination node.
D	Demand matrix.
$D[s,t]$	An element in the demand matrix that specifies the amount of demand from source node s to destination node t. It can also be specified as d_{st}.
w_{ij}	Weight on arc (i, j). If $a = (i, j)$, then it can also be represented as w_a.
c_{ij}	Capacity on arc (i, j). If $a = (i, j)$, then it can also be represented as c_a.
Φ	Cost function.
$\Phi_{i,j}$	Cost associated with arc (i, j). If $a = (i, j)$, then it can also be represented as Φ_a.
δ_u^t	Outdegree of node u when destination node is t.
$\delta^+(u)$	Outdegree of node u.
$\delta^-(u)$	Indegree of node u.
l_a^t	Load on arc a when destination node is t.
l_a	Total load on arc a.
$f_a^{(s,t)}$	Traffic flow from node s to t over arc a.
MU	It is the maximum utilization of the network. It is defined as the utilization of the link whose value is more than that of other links.
MC	It is the maximum cost of link present in the network.
$\mid A \mid$	Number of arcs.
$SumD$	It is the sum of all the demand values in demand matrix.
$SetCA$	It is the set of congested arcs.

B. Assumptions

1) A single element in the set N is called a "Node".
2) A single element in the set A is called a "Arc".
3) A set $G = (N, A)$ is a graph defined as a finite nonempty set N of nodes and a collection A of pairs of distinct nodes from N.
4) A "directed graph" or "digraph" $G = (N, A)$ is a finite nonempty set N of nodes and a collection A of ordered pairs of distinct nodes from N; each ordered pair of nodes in A is called a "directed arc".
5) A digraph is "strongly connected" if for each pair of nodes i and j there is a directed path $(i = n_1, n_2, ..., n_l = j)$ from i to j. A given graph G must be strongly connected for this problem.
6) A "demand matrix" is a matrix that specifies the traffic flow between s and t, for each pair $(s, t) \in\ NXN$.
7) $(n_1, n_2, ..., n_l)$ is a "directed walk" in a digraph G if (n_i, n_{i+1}) is a directed arc in G for $1 \leq i \leq l - 1$.
8) A "directed path" is a directed walk with no repeated nodes.
9) Given any directed path $p = (i, j, k, ..., l, m)$, the "length" of p is defined as $w_{ij} + w_{jk} + ... + w_{lm}$.
10) The "outdegree" of a node u is a set of arcs leaving node u i.e., $\{(u, v) : (u, v) \in A\}$.
11) The "indegree" of a node u is a set of arcs entering node u i.e.,$\{(v, u) : (v, u) \in A\}$.
12) The input to the problem will be a graph G, a demand matrix D, and capacities of each arc.

VIII. Acknowledgement

Acknowledgement goes to King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia, for support of this research work. This material is based in part on work supported by a KFUPM research project under Fast Track Grant No. SAB-2006-10. The authors wish to thank Bernard Fortz and Mikkel Thorup for sharing the test problems.

References

[1] K.G. Coffman and A.M. Odlyzko. Internet growth: Is there a moore's law for data traffic? *Handbook of Massive Data Sets*, pages 47–93, 2001.

[2] Bernard Fortz, J. Rexford, and Mikkel Thorup. Traffic engineering with traditional ip routing protocols. *IEEE Communicatoins Magazine*, pages 118–124, 2002.

[3] Bernard Fortz and Mikkel Thorup. Increasing internet capacity using local search. *Technical Report IS-MG*, 2000.

[4] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms and their Application to Engineering*. IEEE Computer Society Press, December 1999.

[5] Mohammed H. Sqalli, Sadiq M. Sait, and Mohammed Aijaz Mohiuddin. An enhanced estimator to multi-objective ospf weight setting problem. *Network Operations and Management Symposium, NOMS*, 2006.

[6] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing ospf weights. *IEEE INFOCOM*, 2000.

[7] M Resende Ericsson and P Pardalos. A genetic algorithm for the weight setting problem in ospf routing. *Combinatorial Optimisation conference*, 2002.

[8] L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, , and M. Thorup. A hybrid genetic algorithm for the weight setting problem in ospf/is-is routing. *AT&T Labs Research Technical Report, TD-5NTN5G*, 2005.

[9] A. Sridharan, R. Gurin, and C. Diot. Achieving near-optimal traffic engineering solutions for current ospf/is-is networks. *Sprint ATL Technical Report TR02-ATL-022037, Sprint Labs.*

[10] Sadiq M. Sait, Mohammed H. Sqalli, and Mohammed Aijaz Mohiuddin. Engineering evolutionary algorithm to solve multi-objective ospf weight setting problem. *Australian Conference on Artificial Intelligence*, pages 950–955, 2006.

[11] Bernard Fortz and Mikkel Thorup. Robust optimization of ospf/is-is weights. *Proceedings of the International Network Optimization Conference*, pages 225–230, 2003.

[12] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reigold, Jennifer Rexford, and Fred True. Deriving traffic demands for operational ip networks: Methodology and experience. *IEEE/ACM Transactions on Networking,*, 9(3), 2001.

[13] E Dijkstra. A node on two problems in connection of graphs. *Numerical Mathematics*, 1959.

[14] E. W. Zegura. Gt-itm: Georgia tech internetwork topology models (software),. *http://www.cc.gatech.edu/faq/Ellen.Zegura/gt-itm/gt-itm.tar.gz*, 1996.

[15] K. Calvert, M. Doar, and E. W Zegura. Modeling internet topology. *IEEE Communications Magazine*, pages 160–163, 1997.

[16] E. W. Zegura, K.L.Calvert, and S.Bhattacharjee. How to model an internetwork. *IEEE Conf.On computer Communications(INFOCOM)*, pages 594–602, 1996.