An Enhanced Estimator to Multi-objective OSPF Weight Setting Problem

Mohammed H. Sqalli, Sadiq M. Sait, and Mohammed Aijaz Mohiuddin

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

***Abstract*— Open Shortest Path First (OSPF) is a routing protocol which is widely used in the Industry. Its functionality mainly depends on the weights assigned to the links. Given the traffic demands on a network, setting weights such that congestion can be avoided is an NP-hard problem. Optimizing these link weights leads to efficient network utilization which is the main goal of traffic engineering. In this paper, Simulated Annealing iterative heuristic is applied to this problem. This will provide close-to-optimal solutions that can be used for network provisioning. For this problem, the cost function that has been used in the literature depends solely on the links utilization and therefore optimizes only the network utilization. In this paper, our goal is to optimize the number of congested links in the network in addition to the utilization. Therefore, we propose a new cost function that depends on the utilization and the extra load caused by congested links in the network. This provides the network designer with more flexibility to optimize desired parameters. Our results show less number of congested links and comparable extra load in the network when compared to results of using the existing cost function.**

Index terms - Network provisioning, Configuration management, Traffic engineering, OSPF weight setting problem, and Simulated Annealing.

I. Introduction

An internetwork is a combination of networks connected by routers. When a packet goes from a source to a destination, it passes through many routers until it reaches the router attached to the destination network. The network layer is responsible for carrying a packet from one router to another. In other words it is responsible for *host-to-host* delivery. The network layer uses services of the data link layer which is responsible for *node-to-node* delivery.

A router consults its *routing table* when a packet is ready to be forwarded. The routing table specifies the optimum path for the packet. Routing protocols are responsible for populating entries in these tables. If there is a change in the network topology, the routing protocol updates the entries in the tables. Open Shortest Path First (OSPF) [1] and Routing Information Protocol (RIP) [2] are two well known intra-domain routing protocols. These protocols are unicast routing protocols that work within an autonomous system. An autonomous system (AS) is a group of networks and routers under the authority of a single administration.

In this paper, a problem related to OSPF routing protocol is addressed, namely OSPF Weight Setting (OSPFWS). The objective is to set the OSPF weights on the network links such that, the network is utilized efficiently. OSPF routing protocol works on the basis of the weights assigned on the links of the network by the network operator. OSPF routing protocol is based on Dijkstra's algorithm [3], [4]. Based on the assigned weights, Dijkstra's algorithm finds the shortest paths for each pair of source and destination. Once shortest paths are found, they are stored in routing tables and routing is then done accordingly. This routing influences the amount of traffic that will flow on different links, and thus affects the utilization on these links. Cisco recommends that the OSPF metric (i.e., link's weight) be calculated as ref-bw divided by bandwidth (i.e., link's capacity), with ref-bw equal to 10^8 by default [5]. This states that the weights are set to the inverse of the link's capacity. However, this assumes that the flow is negligeable compared to the capacity of the link [6], which may not always be the case.

OSPF is commonly deployed by Internet Service Providers (ISPs), and thus assigning appropriate weights to links will allow ISPs to make better use of available resources. OSPFWS is a network provisioning mechanism, to be used off-line, for finding the optimal weights to be assigned to links, and which will provide an efficient use of network resources. The weights are then configured at the routers before the OSPF routing protocol is initiated. This also assumes that the network topology and the expected traffic demands are fixed. In practice, it may not be desirable to use OSPFWS as an online mechanism that reacts in real-time to changes in the topology or traffic demands. This may frequently update the weights of a large number of links, which will cause the OSPF protocol to recompute the new shortest paths based on the new weights every time a change occurs on the network. This may not be practical and may make the network instable.

The OSPFWS problem is proven to be NP-hard [7], and therefore belongs to a class of combinatorial optimization problems. In this paper, we use a well known iterative heuristic, namely Simulated Annealing (SA) [8], to address the OSPFWS problem. We have implemented a classical simulated annealing-based algorithm using two cost functions for solving this problem. The first cost function was proposed by Fortz and Thorup [7], and is based on the utilization ranges. In their work, they have used Tabu Search (TS) to solve the OSPFWS

1-4244-0143-7/06/$20.00 ©2006 IEEE.

problem.

The second cost function is a new estimator to this problem which also addresses the issue of optimizing the number of congested links. Our objective is to provide a more flexible metric to the network designer to decide about which weight setting is best for a specific network. For instance, a designer may wish to minimize the number of congested links. Our new cost function provides such possibility. The intuition behind using this metric is to minimize the number of congested links, and thus the cost of upgrading the network. We have also implemented a dynamic shortest path algorithm to find multiple equidistance shortest paths to evenly split the traffic.

The results obtained are compared with existing results of TS and Genetic Algorithm (GA) for the same problem. Comparable results were found with some exceptions. The new cost function, however, provided better results with respect to the number of congested links in most test cases.

The rest of the paper is organized as follows; first we will discuss the current related work. Then, we will state the OSPFWS problem and formulate the cost functions. The simulated annealing algorithm and related parameters used for OSPFWS problem will then be presented. The section that follows will include the experimental results and a discussion including a comparison with previous related work. The paper ends with a conclusion.

II. Related Work

Much research has been done on optimizing OSPF weights. In [7] a cost function based on the utilization ranges is formulated and Tabu search [8] is applied. Dynamic shortest path algorithm given in [9], [10], [11] was applied to find multiple equidistance shortest paths between source and destination nodes. Ericsson, Resende and Pardalos applied genetic algorithm to the same problem [12]. Other papers on optimizing OSPF weights [13], [14], [15] have either chosen weights so as to avoid multiple shortest paths from source to destination, or applied a protocol for breaking ties, thus selecting a unique shortest path for each source-destination pair. Rodrigues and Ramakrishnan [15] presented a local search procedure similar to that of Fortz and Thorup. The input to the algorithm for this problem is a network topology, capacity of links and a demand matrix. The demand matrix represents the traffic between each pair of nodes present in the topology. The methodology for deriving traffic demands from operational networks is described in [16].

Srivastava et al [6] considered three objectives: minimization of total link flow, minimization of maximum link utilization, and a composition of both. They presented an approach based on the Lagrangian relaxation-based dual method to determine the optimal weight system. They defined five performance measures to show the impact of the different objective functions on these measures.

Sridharan et al [17] used a different approach by defining a set of allowable next hops, which is a subset of the set of next hops corresponding to the shortest paths computed by the routing algorithm. This provides a mechanism to control how traffic is distributed without modifying the routing protocols, such as OSPF.

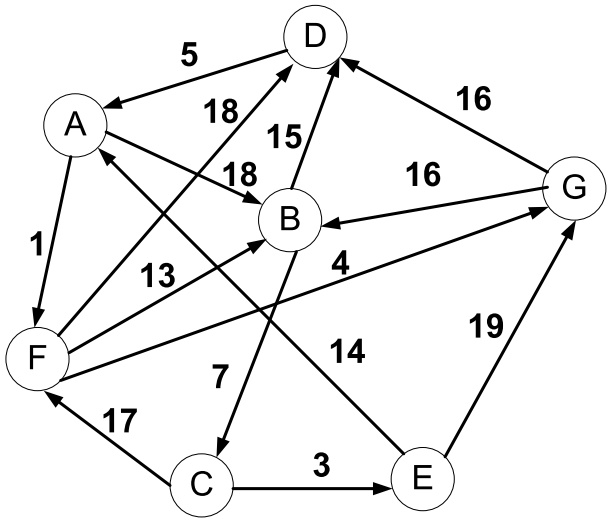

Fig. 1. Representation of a topology with assigned weights.

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 $G = (N, A)$, 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. Figure 1 depicts a topology with assigned weights in the range $[1, 20]$. A solution for this topology can be $(18, 1, 7, 15, 3, 17, 14, 19, 13, 18, 4, 16, 16)$. These elements (i.e., weights) are arranged in a specific order for simplicity. The elements are ordered in the following manner: the outgoing links from node A listed first (i.e., AB, AF), followed by the outgoing links from node B (i.e., BC, BD), and so on.

A. Mathematical Model and Cost Function

Using the above notations, the problem can be formulated as multi-commodity flow problem [12], [18]:

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

subject to these constraints:

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

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

In the experiments, Φ_a are piecewise linear functions, with $\Phi_a(0) = 0$ and a derivative, $\Phi_a^{'}(l_a)$ 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 [18].

A given weight setting will completely determine the shortest paths, which in turn determine the OSPF routing, and how much of the demand is sent over each 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 basic problem 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 five steps. 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 [19]. Dijkstra's alogrithm usually computes the distances away from source s, but since we want to compute the distance to the sink 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 us to compare costs across different sizes and topologies of networks. We applied the same normalizing scaling factor as introduced by Fortz and Thorup [18], [12].

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. Formulation of a new cost function

The cost function given in [7] is based on the range of utilization present on each link. Through experimentation it was found that this cost function does not address the optimization of number of congested links. In our work, a new cost function is proposed. 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. Consequently, the network designer will need to upgrade fewer links in the network to avoid congestion. This in turn means a less expensive upgrade. Our goal is not to change the cost function used in [7] but rather to give 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 [7].

IV. Simulated Annealing

A. Simulated annealing Algorithm

Simulated annealing is a general adaptive heuristic and belongs to the class of nondeterministic algorithms. It has been applied to several combinatorial optimization problems from various fields of science and engineering. One typical feature of simulated annealing is that, besides accepting solutions with improved costs, it also, to a limited extent, accept solutions with deteriorated costs. It is this feature that gives the

Algorithm *Simulated_annealing*$(S_0, T_0, \alpha, \beta, M, Maxtime)$;
(*S_0 is the initial solution *)
(*BestS is the best solution *)
(*T_0 is the initial temperature *)
(*α is the cooling rate *)
(*β a constant *)
(*$Maxtime$ is the total allowed time for the annealing process *)
(*M represents the time until the next parameter update *)
Begin
$T = T_0$;
$CurS$=S_0;
$BestS$=$CurS$; /* $BestS$ is the best solution seen so far */
$CurCost$=*Cost*($CurS$);
$BestCost$=*Cost*($BestS$);
$Time = 0$;
Repeat
Metropolis($CurS$, $CurCost$, $BestS$, $BestCost$, T, M);
$Time = Time + M$;
$T = \alpha T$;
$M = \beta M$
Until ($Time \geq MaxTime$);
Return ($BestS$)
End. (**of Simulated_annealing**)

Fig. 2. Procedure for Simulated Annealing Algorithm [8].

Algorithm *Metropolis*($CurS$, $CurCost$, $BestS$, $BestCost$, T, M);
Begin
Repeat
$NewS$= $Neighbor$($CurS$);
$NewCost$=*Cost*($NewS$);
Δ*Cost*=($NewCost - CurCost$);
If (Δ*Cost*< 0) **Then**
$CurS$=$NewS$;
If $NewCost < BestCost$ **Then**
$BestS$= $NewS$
EndIf
Else
If ($RANDOM < e^{-\Delta Cost/T}$) **Then**
$CurS$=$NewS$;
EndIf
EndIf
$M = M - 1$
Until ($M = 0$)
End. (**of Metropolis**)

Fig. 3. The Metropolis procedure [8].

heuristic the hill climbing capability. Initially the probability of accepting inferior solutions (those with largest costs) is large; but as the search progresses, only smaller deteriorations are accepted, and finally only good solutions are accepted. A strong feature of the simulated annealing heuristic is that it is both effective and robust. Regardless of the choice of the initial configuration it produces high-quality solutions. The detail description of the algorithm can be found in [8]. The general outline of the algorithm is given in Figure 2 and 3.

B. Simulated annealing for OSPFWS

A move for OSPFWS in annealing algorithm is done by changing the value of weight present on any one link of the network. Consider the topology in Figure 1 and its solution $(18, 1, 7, 15, 3, 17, 5, 14, 19, 13, 18, 4, 16, 16)$. Simulated annealing is blind in making moves in the search space. Changing any weight from the above solution will take us to a different solution in the search space. Thus a new solution can be $(18, 1, 7, 15, 3, 5, 5, 14, 19, 13, 18, 4, 16, 16)$ by changing the weight 17 to 5. Likewise, the algorithm proceeds as shown in Figure 4. In simulated annealing, bad solutions are accepted at higher temperatures based on a defined probability. As the temperature decreases, the probability of accepting bad solutions decreases and the algorithm becomes greedy.

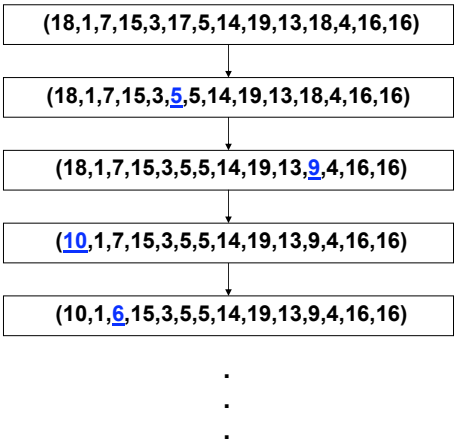

Fig. 4. Sequence of moves in simulated annealing.

In the proposed implementation, after each move the solution is evaluated with respect to the cost function. Initially, Dijkstra algorithm [19] is employed and subsequently we used the dynamic shortest path algorithm described in [9], [10], [11] to find multiple shortest paths between two nodes if they exist.

C. Simulated Annealing Parameters

Simulated annealing is basically associated with 4 different parameters: the initial temperature (T_0), alpha (α), beta (β) and M. In simulated annealing each time the Metropolis loop is called, T is reduced to αM and M is increased to βM. Initial temperature values are set such that all the initial moves are accepted. This process is repeated for all the test cases with different scaled demand values. Table I shows the values of initial temperature with r100N503a test case with different scaled demand values. In this table, SumD stands for the sum of all the demand values in demand matrix. Values of other test cases are not included due to limited space.

Through experimentation it was found that $\alpha = 0.965$, $\beta = 1.01$ and $M = 10$ are suitable parameters for both cost functions. Figures 5 and 6 show the trend of the current cost versus the number of iterations for h100N280a test case corresponding to two different cost functions. In these figures IC stands for initial cost and BC stands for best cost. Each figure includes 3 runs for each case. It is clear from these figures that regardless of the initial solution with respect to these parameters, costs of the final solutions are nearly equal to each other. Similar trends were observed for other test cases, but they are not reported here due to limited space.

TABLE I

SA T_0 VALUES FOR R100N503A

SumD	FortzCF T_0	NewCF T_0
8383	0.4	1
16766	5.4	4.2
25149	21.6	11.2
33531	29.2	18.6
41914	35	24
50297	37	34.2
58680	58.2	35.8
67063	59.2	35.8
75446	64	40
83829	65.2	56.6
92211	69.8	57.4
100594	70.4	57.8

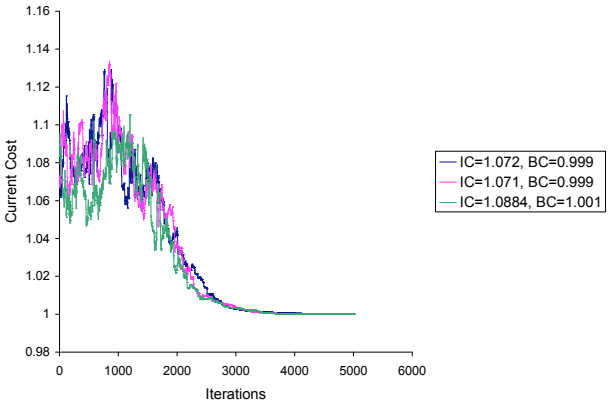

Fig. 5. Current cost curve for SA w.r.t Fortz cost function, test case h100N280a.

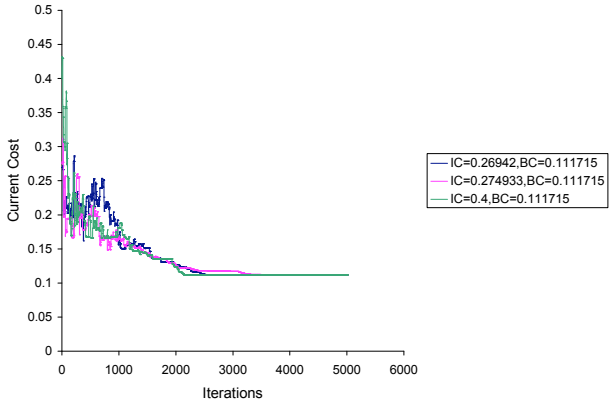

Fig. 6. Current cost curve for SA w.r.t New cost function test case h100N280a.

TABLE II

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

V. RESULTS AND DISCUSSION

A. Test cases

All the test cases are taken from [7]. Table II 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 [20], based on a model of Calvert [21] and Zegura [22]. 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. Fortz and Thorup generated the demands to force some nodes to be more active senders or receivers than others, thus modeling hot spots on the network. Their generation assigns higher demands to closely located nodes pairs. Further details can be found in [18].

B. Results

In this section we present the computational results. From hereafter Fortz cost function is referred as FortzCF and our proposed function is referred as NewCF. Results are compared with those produced by TS [7] and GA [12]. These existing results are compared to our simulated annealing implementation with existing and newly proposed cost function. We observe that with FortzCF, simulated annealing generally finds comparable results to existing ones in all the test cases. We plotted four statistics namely current cost, maximum utilization, percentage of extra load and number of congested links. By maximum utilization, we mean the link utilization (total traffic i.e., load on the link divided by the link capacity) of the maximum utilized link in the network. 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. Congested links are the links of utilization greater than 1. We are unaware of results for the extra load and the number of congested links for tabu search (TS) and genetic algorithm (GA), so we only plotted these statistics with respect to the two cost functions: FortzCF and NewCF.

In our work, C programming language is used for simulation. Simulations are ran for 5000 iterations for each test case using Pentium-4 machine. Now, we will first discuss the results obtained for the test case r100N403a. Figure 7 shows the cost curve for this test case using FortzCF for 3 algorithms: TS, GA and SA. This figure shows that SA is performing better than GA and worse than TS. Figure 8 shows the maximum utilization curve for the same test case. TS is performing better than others, but when there is no congestion on the network (i.e, $MU < 1$) the results of all algorithms are comparable. In a congested state, SA with FortzCF is doing better than GA and than SA with NewCF. Figure 9 shows the percentage of

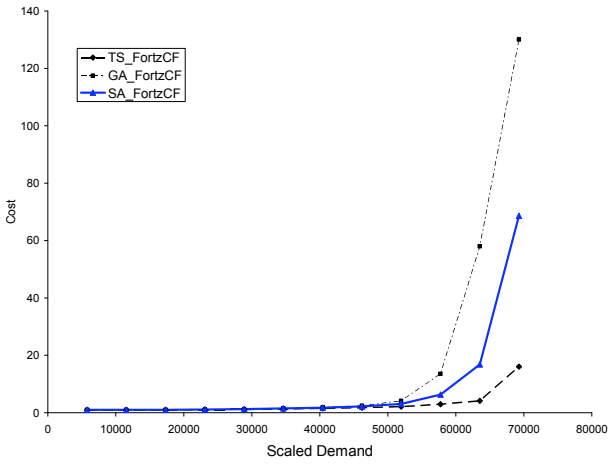

Fig. 7. Cost versus scaled Internet traffic on r100N403a network using FortzCF.

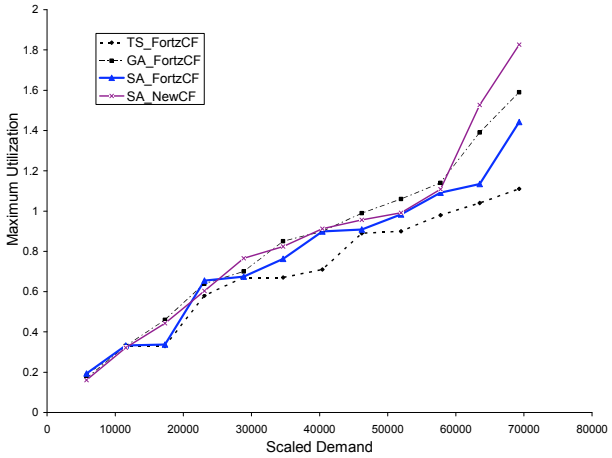

Fig. 8. Maximum utilization versus scaled Internet traffic on r100N403a network.

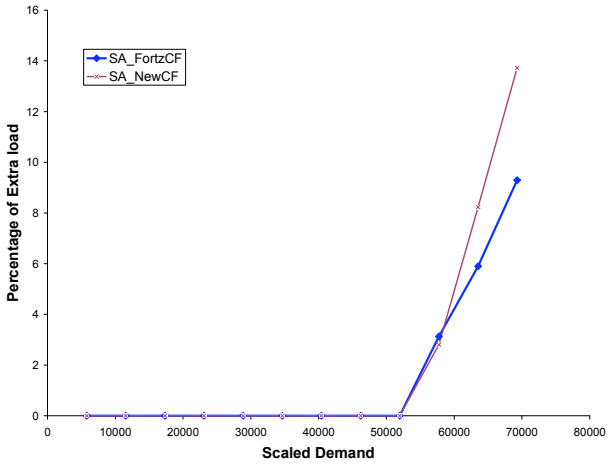

Fig. 9. Percentage of extra load versus scaled Internet traffic on r100N403a network.

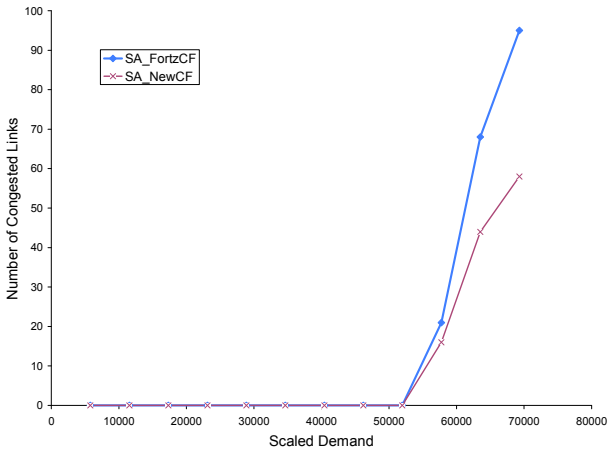

Fig. 10. Number of congested links versus scaled Internet traffic on r100N403a network.

extra load of the network for the same test case. It is clear from this figure that SA with FortzCF is performing better than SA with NewCF when the demand is high. Figure 10 shows the number of congested links in the network for the same test case. SA with NewCF is performing better than SA with FortzCF for high demand values. Figure 11 shows the progression of congested links with both cost functions. It is clear that NewCF is doing better than FortzCF. In the following paragraphs, we will discuss results obtained for the other test cases.

In terms of cost, for test cases r50N228a and r50N245a performance of SA is in between that of GA and TS. For test case h50N212a, we found SA performance to be worse than that of GA and TS. For other test cases, we found comparable results.

In terms of maximum utilization, for test cases h50N148a and h100N360a, the maximum utilization using NewCF is initially better than other algorithms and becomes comparable after MU crosses 1. For test cases r50N245a and r100N503a,

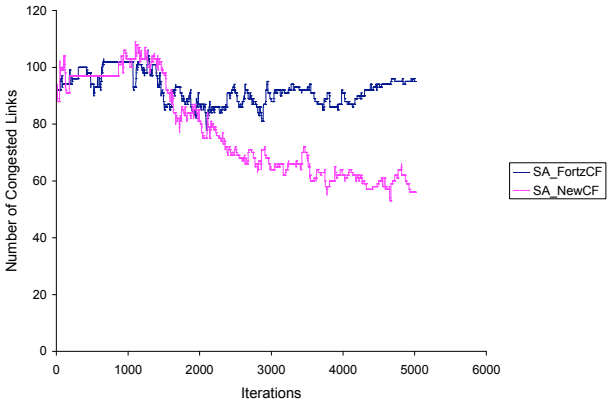

Fig. 11. Progression of number of congested links w.r.t to both cost functions.

TABLE III

COST FUNCTION VALUES CORRESPONDING TO THE HIGHEST DEMAND VALUES FOR ALL TEST CASES

Test case	Demand value	FortzCF			NewCF		
		Best	Avg	SD	Best	Avg	SD
h100N280a	4605	20.438	20.777	0.309	2.023	2.076	0.046
h100N360a	12407	20.157	20.206	0.049	2.80	2.804	0.0047
h50N148a	4928	22.908	23.01	0.102	2.702	2.743	0.042
h50N212a	3363	13.449	13.498	0.0455	1.819	1.852	0.0282
r100N403a	70000	68.669	69.65	0.104	21.569	21.756	0.178
r100N503a	100594	83.982	84.23	0.321	22.022	22.196	0.052
r50N228a	42281	32.355	32.34	0.015	11.701	11.716	0.0201
r50N245a	53562	151.033	151.2	0.199	32.239	32.963	0.7606
w100N391a	48474	12.065	12.067	0.0072	2.51	2.541	0.0307
w100N476a	63493	24.296	24.3509	0.0863	4.416	4.452	0.0507
w50N169a	25411	14.552	14.567	0.01274	3.011	3.107	0.148
w100N230a	39447	11.828	11.917	0.059	3.112	3.1189	0.0056

TS is performing better than other algorithms. For the test case r50N228a, SA and TS are equally doing better than GA. For other test cases, we find comparable results.

In terms of percentage of extra load, NewCF is providing better results than FortzCF in h100N280a and h50N148a. We found comparable results in test cases r100N503a, w50N230a, w100N476a, w50N169a and w100N391a. We found that FortzCF is doing better than NewCF in test cases r50N228a, h50N212a, h100N360a, r50N245a and r100N403a.

In terms of number of congested links, we find equal number of congested links for both cost functions in test cases w50N230a and w100N391a. In all other test cases, we found that NewCF is performing better than FortzCF. The motivation behind our new cost function is that in case of a network congestion, it is preferred to have a solution which provides less number of congested links. For instance, a network engineer may prefer to increase the capacity of 10 congested links rather than increasing the capacity of 40 congested links. Our cost function optimizes the maximum utilization when there is no congestion in the network; and in the case of congestion it first tries to optimize the number of congested links and then the maximum utilization. Table III shows the best, the average, and the standard deviation cost values for all test cases with the highest demand values.

VI. CONCLUSION

A new cost function for finding good solutions with respect to congested links and extra load is proposed in this paper. Simulated annealing iterative heuristic is used and tested on several networks using the new and an existing cost function. We found that the new cost function includes intelligence to optimize the number of congested links. The results obtained are compared with existing results of tabu search and genetic algorithm for the same problem. Comparable results were found with some exceptions. The new cost function, however, provided better results with respect to the number of congested links in most cases. This 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, less number of congested links, etc.

VII. ACKNOWLEDGEMENT

Acknowledgement goes to King Fahd University of Petroleum & Minerals (KFUPM), Dhahran, Saudi Arabia, for supporting this research work. The authors would like to thank Bernard Fortz and Mikkel Thorup for sharing the test problems.

VIII. 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.

REFERENCES

[1] J. T Moy. *OSPF: Anatomy of an Internet Routing Protocol.* Addison-Wesley., 1999.
[2] Routing information protocol at http://www.faqs.org/rfcs/rfc1058.html.
[3] Dimitri Bertsekas and Robert Gallager. *Data Networks*. Prentice Hall Series, 1992.
[4] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice Hall.
[5] Cisco Systems. Configuring OSPF. *http://www.cisco.com/univercd/*, 2004.
[6] Shekhar Srivastava, Gaurav Agrawal, Micha Pioro, and Deep Medhi. Determining link weight system under various objectives for OSPF networks using a lagrangian relaxation-based approach. *IEEE ETransactions on Network and Service Management*, 2(1):9–18, Third quarter 2005.
[7] Bernard Fortz and Mikkel Thorup. Internet traffic engineering by optimizing OSPF weights. *IEEE Conference on Computer Communications (INFOCOM)*, pages 519–528, 2000.
[8] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms and their Application to Engineering*. IEEE Computer Society Press, December 1999.
[9] Daniele Frigioni, Mario loffreda, Umberto Nanni, and Giulio Pasqualone. Experimental analysis of dynamic algorithms for the single source shortest paths problem. *ACM Journal of Experimental Algorithms*, 1998.
[10] G. Ramalingam and Thomas Reps. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, pages 267–305, 1996.
[11] Bernard Fortz. Combinatorial optimization and telecommunications. *http://www.poms.ucl.ac.be/staff/bf/en/COCom-5.pdf*.
[12] Ericsson, M. Resende, and P. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. *J.Combinatorial Optimisation conference*, 2002.
[13] F. Lin and J. Wang. Minimax open shortest path first routing algorithms in networks supporting the smds services. *IEEE Internation Conference on Communications (ICC)*, 2:666–670, 1993.
[14] A. Bley, M. Grotchel, and R. Wesslay. Desing of broadband virtual private netwokrs: Model oand heuristics for the B-WiN. *Technical Report SC 98-13 DIMACS Workshop on Robust Communication Network and Survivability*, 1998.
[15] M. Rodrigues and K. G. Ramakrishnan. Optimal routing in data networks. *Presentation at International Telecommunication Symposium (ITS)*, 1994.
[16] 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.
[17] Ashwin Sridharan, Roch Guerin, and Christophe Diot. Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks. *IEEE INFOCOM*, 2003.
[18] Bernard Fortz and Mikkel Thorup. Increasing internet capacity using local search. *Technical Report IS-MG*, 2000.
[19] E. Dijkstra. A node on two problems in connection of graphs. *Numerical Mathematics*, 1959.
[20] 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.
[21] K. Calvert, M. Doar, and E. W. Zegura. Modeling internet toplogy. *IEEE Communications Magazine*, (35):160–163, 1997.
[22] K. L Calvert E. W. Zegura and S. Bhattacharjee. How to model an internetwork. *15th IEE Conference on Computer Communications (INFOCOM)*, pages 594–602, 1996.