Optimizing OSPF Routing Using Accelerated Iterative Heuristic

Sadiq M. Sait, Mohammed H. Sqalli, and Syed Asadullah
Computer Engineering Department
King Fahd University of Petroleum & Minerals
Dhahran 31261, Saudi Arabia
{sadiq,sqalli,sasad}@kfupm.edu.sa

Abstract: The problem of setting Open Shortest Path First (OSPF) weights on links such that congestion can be avoided is proved to be NP-hard. Many iterative heuristics have been applied to solve the OSPF weight setting (OSPFWS) problem. As the size of any combinatorial optimization problem increases, it becomes more difficult to find an optimum solution using sequential algorithms. In this paper, we investigate the parallelization of Tabu Search and apply two variants of a Parallel Tabu Search (PTS) heuristic on the OSPFWS problem. It is shown through experimental results that both PTS approaches produced better solutions quality compared to the sequential heuristics; specifically for larger topologies. In one approach, we propose a new design for our parallel cooperative search algorithm, which performs better than the conventional parallel heuristic.

Key–Words: Open Shortest Path First (OSPF), OSPF Weight Setting Problem, NP-hard, Iterative Heuristics, Parallel Tabu Search.

1 Introduction

Open Shortest Path First (OSPF) is an intra autonomous system routing protocol which computes the shortest paths based on the weights assigned to the links. Routing on the Internet defines the traffic flows over the selected shortest paths. Traffic engineering aims at providing the required Quality of Service (QoS) to the users by efficiently utilizing the available resources and managing these traffic flows.

OSPF uses the link weights as its routing metric. The major networking vendor Cisco, assigns the link weights inversely proportional to the link capacity. This is termed as inverse capacity OSPF in the literature [1, 2]. Other methods of weight assignment such as unit OSPF and random OSPF also exist. Given a set of traffic demands between each source and destination node, the OSPF weight setting problem consists of determining suitable OSPF link weights so as to optimize a certain criterion (cost function), aiming at avoiding congestion in the network. This Problem is NP-hard [1].

The application of iterative heuristics to solve the OSPFWS problem was first attempted by Fortz and Thoroup [3]. They applied tabu search using their proposed cost function [1]. We call this cost function FortzCF. Sqalli et al. proposed a new cost function (NewCF) and applied Simulated Annealing [4], Simulated Evolution [5], and Tabu Search [6] using both cost functions. They proved experimentally that the new cost function minimizes the number of congested links. Ericsson et al. attempted Genetic Algorithm [2] as well as hybrid GA for the same problem.

In this paper, we investigate parallelizing the Tabu Search iterative heuristic to solve the OSPFWS problem which, to our knowledge, has not been attempted yet. Enormous work has been done in the area of parallelization of iterative heuristics in general and Parallel Tabu Search (PTS) [7, 8] in particular to solve a range of combinatorial optimization problems. Different implementation strategies are also proposed in the literature which include, using dynamic Tabu Search parameters at different processors, using a cluster in some hierarchical architecture to enhance the search, integration of different parallelization strategies, etc. A detailed survey of various parallelization strategies and their application to one or more classical or specific optimization problems can also be found in the literature [9].

In this paper, we present a Parallelized Tabu Search (PTS) [10] heuristic using two strategies to solve the OSPFWS problem by using the two cost functions available in the literature. The objective is to achieve better solutions quality in a given time, which could not be achieved by sequential heuristics; specifically for larger topologies and higher traffic demands.

The rest of the paper is organized as follows; The OSPFWS problem statement and the cost functions proposed in the literature are presented in Section 2. The two Parallel Tabu Search algorithms are discussed in Section 3. This is followed by the experimental results and conclusion.

2 Problem Statement

The OSPF weight setting problem can be stated as follows: Given a directed network of nodes and arcs $G = (N, A)$, a demand matrix D, and capacity C_a for each arc $a \in A$, 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. When routing is perfromed 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 are computed. This is done for all destination nodes. The aggregated partial loads for all destinations on a particular arc give the total load l_a on that arc. The cost of sending traffic through this arc is given by $\Phi_a(l_a)$. The cost value depends on the utilization of the arc and is given by the linear function proposed by Fortz and Thoroup [1].

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

The Fortz cost function is given in equation 2.

$$\Phi = \sum_{a \in A} \Phi_a(l_a) \tag{2}$$

The objective is to minimize Φ, subject to these constraints:

$$l_a = \sum_{(s,t) \in NXN} f_a^{(s,t)} \; a \in A, \tag{3}$$

$$f_a^{(s,t)} \geq 0 \tag{4}$$

In constraint 3, for traffic between a source and a destination pair (s,t), $f_a^{(s,t)}$ indicates the amount of traffic flow that goes over arc a. The detailed steps showing the formulation of this cost function can be found in the literature [1, 4].

The New cost function [4] is shown in equation 5,

$$\Phi = MU + \frac{\sum_{a \in SetCA} (l_a - c_a)}{E} \tag{5}$$

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.

We have applied our proposed parallel heuristics on both cost functions presented in this section.

3 Parallel Tabu Search (PTS)

In conventional parallel implementations of tabu search, all the processors including the master independently generate an initial solution and compute its cost. After searching its part of the current neighborhood, repeatedly for a certain number of iterations, each slave process reports its best move back to the master. The master process selects the best among the received best moves (subject to tabu conditions). If the stopping criteria are met then the search stops; otherwise the master broadcasts the selected move back to the slaves. The selection criterion for the best move and also the slaves to which this best move is broadcasted, depend on the implementation strategy. The search continues, where each processor sets the received solution from the master as its current solution and explores the neighborhood. The master also broadcasts the tabu list pertaining to the selected solution and all slaves update their individual tabu lists.

Two strategies, namely, PTS-Star and PTS-Ring were implemented to parallelize Tabu Search. The implementation aspects of the two strategies mainly differ in the way the processors coordinate during the periodic solution exchanges. From the functional aspect, they differ in the search space each processor works on after each solution exchange. In this section, we discuss the two strategies in detail.

3.1 PTS-Star

In the PTS-Star strategy, the parallel cluster consists of $n + 1$ processors with 1 master and n slave processors/nodes. Each processor starts by generating a random initial solution. Due to the independent random number generation by each entity the initial solution generated by a processor is different from the other. Each slave processor starts Tabu Search and continues for K iterations. After each move the current and best cost are computed. At the end of the K^{th} iteration, the best cost and solution are sent to the master. The master waits to receive the cost and solution from all slaves, after which it compares the received costs and picks the best cost among the n slaves. Once the comparison is done and the best cost is selected, the master broadcasts the corresponding best solution to all slaves. Each slave processor receives the new best solution from the master and sets this received solution as the current solution. With this current solution, Tabu Search is started for another K iterations.

This process of parallel Tabu Searches with processor coordination by periodic solution exchange continues until the termination criterion is met. The master is the central point of coordination between all slaves, which means that all communications go

through the master and each processor is directly connected only to the master. The processor arrangement resembles a star topology and hence the name PTS-Star. The structure of the PTS-Star algorithm is shown in Figure 1.

Algorithm: PTS-Star
Begin:
Slave:
1: Generate Initial Solution.
2: Continue TS for K Iterations.
3: Send the bestsol and bestcost to master.
Master:
4: Receive individual bestcost from each slave.
5: Compare the received costs.
6: Select the bestsol among all slaves.
7: **if** termination criteria is met: **then**
Save the results.
Stop.
8: **else**:
Broadcast bestsol to all slaves.
9: **GoTo**: 4
Slave:
10: Receive the solution from master.
11: Update current solution based on the received solution.
12: **if** termination criteria is met: **Stop**.
13: **else**: **GoTo**: 2
End:

Figure 1: Structure of the PTS-Star algorithm

With multiple processors working on the solution and exploring the search space, the convergence is expected to be faster and the final solution quality is also expected to be better by visiting more points (solutions) in the search space.

This technique however has the limitation that, after every periodic solution exchange, all processors start Tabu Search with the same current solution, and hence searching in similar regions of the search space. To prevent this redundant search, some technique is required where different processors can coordinate to explore a larger search space. The next strategy PTS-Ring addresses this issue.

3.2 PTS-Ring

The primary objective of devising this new strategy is to improve the solution quality by making the processors work in different regions of the search space. This is done by making different set of processors start the search from different starting points (solutions) after each solution exchange, thereby diversifying the search considerably. It is practically important to search a new region when the neighborhood search fails to improve the best solution for a while; and diversification techniques can help achieve this goal. In the PTS-Ring strategy, the parallel cluster consists of n+1 processors with 1 master and n slaves. Each slave node S_i has a rank i associated with it. A node exchanges its solution only with its neighbors. Neighborhood is defined as a set of three nodes with ranks $i-1$, i, $i+1$; hence the neighbors of node S_i are (S_{i-1}, S_{i+1}). Further, the neighbors of the last node S_n are (S_{n-1}, S_1) while the neighbors of the first node S_1 are (S_n, S_2). This arrangement of nodes, with the first and last node being neighbors, completes a ring structure. The implementation logic of the PTS-Ring strategy is explained below.

Each processor starts by generating a random initial solution. Each slave processor starts Tabu Search and continues for K iterations (moves). After every move, the current and best cost are computed and at the end of K moves, the best cost and solution are sent to the master. The master receives the cost and solution from all slaves. For each slave processor, the master compares the slave's cost with its neighbors and sends the best solution in its (slave's) neighborhood to this slave processor. Hence, any slave S_i receives the best solution among the three processors (S_{i-1}, S_i, S_{i+1}). Each slave processor receives the new best solution from the master and sets this received solution as the current solution. With this current solution, Tabu Search is started for another K iterations. The process continues until the termination criterion is met.

The structure of the PTS-Ring algorithm is shown in Figure 2.

Algorithm: PTS-Ring
Begin:
M=Master.
S={ S_1, S_2,...S_n }; n=No. of Slave Processors.
N(s): Neighbour of s.
Ring Arrangement:
N(S_i)=(S_{i-1}, S_{i+1}).
N(S_1)=(S_n, S_2).
N(S_n)=(S_{n-1},S_1).
Slave:
1: Generate Initial Soln.
2: continue TS for K Iterations.
3: Send the bestsol and bestcost to M.
Master:
4: Receive costs from all slaves.
5: **if** termination criteria is met: **then**
Save the results.
Stop
6: **else**:
for each slave S_i. **do**
7: Compare cost of S_i and N(S_i).
8: Select the best solution.
9: Send the bestsol to S_i.
EndFor
EndIf **GoTo**: 4
Slave:
10: Receive the solution from master.
11: Update current solution based on the received solution.
12: **if** termination criteria is met: **Stop**
13: **else**: **GoTo**: 2
End:

Figure 2: Structure of the PTS-Ring algorithm

4 Results

In this section, we present the experimental results for the two Parallel Tabu Search algorithms. The test cases and demand matrices are the same as used in the literature [1].

Experimental results have been recorded for the following four performance metrics: Cost, Maximum Utilization (MU), Percentage of Extra Load (PXLoad), and Number of Congested Links (NOCL).

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 the 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 the 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. The statistics for these performance metrics are plotted with respect to both parallel strategies PTS-Star and PTS-Ring, using the two cost functions: FortzCF and NewCF.

4.1 PTS-Star

In this section, we first present the results of the PTS-Star strategy in which we show the Cost comparison versus the number of processors for four test cases. In Figure 3, the results for the maximum demand are shown. It can be seen here that the cost improves with the number of processors for all cases but this improvement for the cases h100N360a and w100N476a is very minimal. The total demand and also the average load per arc (Total Demand/Number of Links) on these topologies is less when compared to the random graphs (r100N403a, r100N503a). The cost of a solution is dependent on the utilization of the links, and lower input demands lead to lesser utilization. Hence, when the input demands are low, a large set of solutions (weight combinations) will result in links with low utilization and hence, a lower cost. These solution sets representing a good solution, form a slightly bigger fraction of the search space and can also be explored by sequential heuristics. Hence, for topologies with lower input demands, the improvement due to parallelization is not significant when compared to sequential heuristics.

The other two cost curves for r100N403a and r100N503a graphs show a significant improvement in cost with the increase in the number of processors. The same trend is also observed for Demand-11. The random graphs used in this work are the graphs with large sizes and higher demands when compared to any other topology. None of the sequential heuristics mentioned in this paper have achieved results close to PTS for these topologies. The final costs achieved using PTS are even upto one third of the best achieved using the sequential Tabu Search heuristic (TS). TS has also the best results among all sequential heuristics [6].

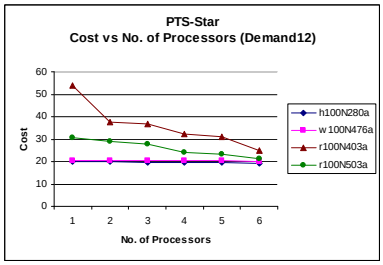

Figure 3: Cost vs. # of Processors Using PTS-Star.

Time comparison for the PTS-Star strategy is shown in Figure 4 for Demand-12. The target cost is the best cost achieved by the sequential heuristic in 1 hour (3600 sec). The runtime for the parallel heuristics is also 1 hour. With the increase in the number of processors, the parallel heuristics generally take lesser time to reach the target cost. The time taken by the parallel heuristic to achieve the target cost is recorded for different number of processors. The h100N280a topology shows an almost linear speedup with the increase in the number of processors. Other topologies show faster convergence with the increase in the number of processors when compared to the sequential heuristic, but the speedup is not linear. It was observed that the average time consumed in synchronization and message passing between processors was about 25% of the total time. Hence, the communication overhead could be one of the reasons for not achieving faster speedups. However, further investigation is required to make a conclusive statement about the same.

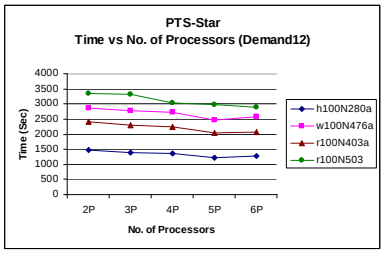

Figure 4: Time vs. # of Processors Using PTS-Star.

4.2 PTS-Ring

Cost comparison for the above mentioned four topologies is shown here for the PTS-Ring strategy. The results for Demand-12 are shown in Figure 5. It can be

seen that, with the increase in the number of processors, the cost improves considerably for the two test cases r100N403a, r100N503a. The PTS-Ring strategy also could not achieve a linear speedup with the increase in the number of processors, but was again able to achieve costs which are about three times better than the sequential heuristics.

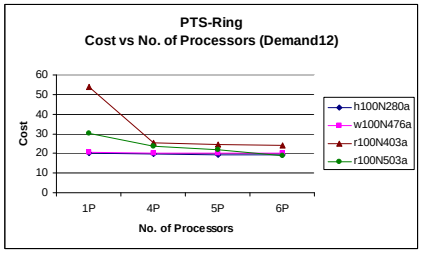

Figure 5: Cost vs. # of Processors Using PTS-Ring.

4.3 PTS-Star Versus PTS-Ring

Results for the Fortz cost function show that the costs achieved by PTS-Ring are slightly better than PTS-Star for all the test cases and demands. This improvement is expected and can be attributed to the diversification feature of the Ring approach. The slave processors, in PTS-Ring, start from slightly different solutions after every solution exchange; thereby covering a wider search space and achieving better solutions when compared to the ring approach. The cost curves for the FortzCF are shown in Figure 6 for Demand-12.

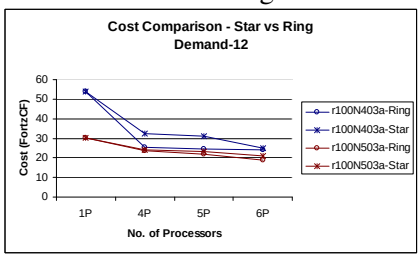

Figure 6: Cost Results of PTS For FortzCF.

Figure 7 shows the cost curves for Demand-12 using the new cost function (NewCF). The PTS-Ring Strategy, with a better diversification mechanism, again produced better results than the PTS-Star Strategy for the NewCF which can be seen from these results.

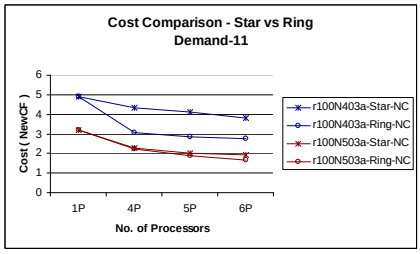

Figure 7: Cost Results of PTS For NewCF.

The comparisons of other performance metrics for both cost functions are shown in the following subsections.

4.4 Number of Congested Links

The comparison of the Number of Congested Links (NOCL) for both cost functions for the topology r100N403a is shown in Figure 8. It can be observed that for FortzCF, there is no improvement of the NOCL with the increase in the number of processors. We have seen in previous results that there is a significant improvement in cost with the number of processors. Hence, we can conclude that the improvement in cost is due to reducing the extra load on the network at the expense of the number of congested links. FortzCF attempts to load balance the congestion on the network among all links by reducing the load from the more congested links and placing it on the lesser congested links. However, for the NewCF, there is a minimization in the NOCL. Among the two strategies, the PTS-Ring minimizes the NOCL better than the PTS-Star for the same cost function, as it also minimizes the overall cost better for both cost functions. The best results for the NOCL were obtained when optimizing weights using the NewCF and the Ring Topology.

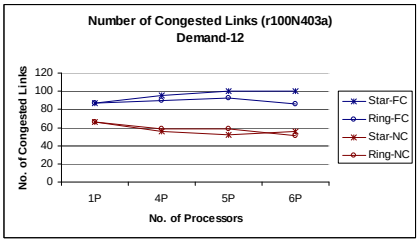

Figure 8: NOCL Results of PTS Using Both CFs.

4.5 Percentage of Extra Load

We now discuss the results for the Percentage of extra load (PXLoad). In the case of Demand-12 as shown in Figure 9, the values of PXLoad are better for FortzCF when optimized using the PTS-Ring strategy compared to the NewCF; and PTS-Ring performs better than PTS-Star. Hence, it can be said that when weights are optimized using the FortzCF, the links which are over-utilized are not highly congested; while with the NewCF, a lesser number of links are congested but each individual congested link has a higher load.

4.6 Maximum Utilization

The results for the Maximum Utilization (MU) are shown in Figure 10. From this figure, it can be seen that the Maximum Utilization in the case of FortzCF is better than that of the NewCF.

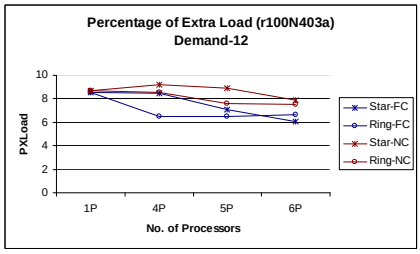

Figure 9: PXLoad Results of PTS Using Both CFs.

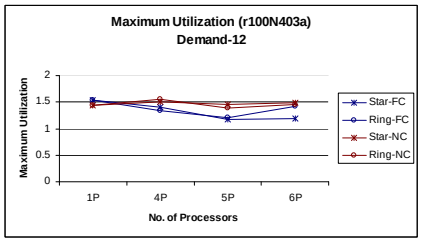

Figure 10: MU Results of PTS Using Both CFs.

From the above results it was observed that the PTS-Ring strategy produced better results for both cost functions and the NewCF produced very good results for minimizing the Number of congested links. The percentage of extra load was found to be best (Minimum) using the FortzCF and the Ring topology. However, in many cases, the NewCF also produced values close to the FortzCF. For both topologies, a significant improvement in the performance metrics was observed using parallelization.

As the size of any combinatorial optimization problem increases, not only its search space increases exponentially but also it becomes more complex as the fraction of solutions that are close to the optimal one becomes smaller. Sequential heuristics often fail to explore this small fraction from the huge search space. Having seen the results of sequential heuristics, PTS-Star, and PTS-Ring in terms of speedup and final cost, it is evident that, as the size and routing complexity of the network increases, there is a significant improvement in the solution quality with parallelized heuristics.

5 Conclusion

Two different Parallel Tabu Search iterative heuristics were implemented on several networks using two cost functions. Results for four performance metrics are reported for FortzCF and NewCF. Both PTS heuristics produced the desired results in the case of large topologies in terms of cost. The PTS-Ring strategy which was designed to induce diversification into the search was found to achieve better cost than any other approach including the PTS-Star for the same run time. However, linear speedup could not be achieved in any strategy. Both the strategies produced better results for smaller topologies also, but did not provide a significant improvement in cost as achieved for the larger topologies. Hence, it can be concluded that if the problem size is large, parallel Tabu Search or any other parallel algorithm can be efficiently applied to the OSPFWS problem to achieve a solution quality which can not be achieved by using sequential algorithms within the same or comparable time frame.

Acknowledgements: Acknowledgement goes to KFUPM for supporting this research work. This material is based in part on work supported by a KFUPM project under Grant No. SAB-2006-10. The authors wish to thank Bernard Fortz and Mikkel Thorup for sharing the test problems.

References:

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

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

[3] B. Fortz, J. Rexford, and M. Thorup. Traffic engineering with traditional ip routing protocols. *IEEE Communicatoins Magazine*, 2002.

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

[5] 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*, 2006.

[6] Mohammed H. Sqalli, Sadiq M. Sait, and Syed Asadullah. Minimizing the Number of Congested Links in OSPF Routing. *ATNAC*, 2008.

[7] A. Bortfeldt, H. Gehring, and D. Mack. A parallel tabu search algorithm for solving the container loading problem. *Parallel Computing 29*, 2003.

[8] Ahmad Al-Yamani, Sadiq M. Sait, Habib Youssef, and Hassan Barada. Parallelizing tabu search on a cluster of heterogenous workstations. *Journal of Metaheuristics*, May 2002.

[9] T.G. Crainic, M. Toulouse, and M. Gendreau. Towards a taxonomy of parallel tabu search heuristics. *INFORMS Journal of Computing*, 9(1), 1997.

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