International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:6, 2007

Parallel Branch and Bound model using Logarithmic Sampling (PBLS) for Symmetric Traveling Salesman Problem

Sheikh Muhammad Azam, Masood-ur-Rehman, Adnan Khalid Bhatti and Nadeem Daudpota

***Abstract*—**Very Large and/or computationally complex optimization problems sometimes require parallel or high-performance computing for achieving a reasonable time for computation. One of the most popular and most complicate problems of this family is "Traveling Salesman Problem". In this paper we have introduced a Branch & Bound based algorithm for the solution of such complicated problems. The main focus of the algorithm is to solve the "symmetric traveling salesman problem". We reviewed some of already available algorithms and felt that there is need of new algorithm which should give optimal solution or near to the optimal solution. On the basis of the use of logarithmic sampling, it was found that the proposed algorithm produced a relatively optimal solution for the problem and results excellent performance as compared with the traditional algorithms of this series.

***Keywords*—**Parallel execution, symmetric traveling salesman problem, branch and bound algorithm, logarithmic sampling.

I. Introduction

THE traveling salesman problem (TSP) is one of those problems to whom the mathematicians and the computer scientists had given loads of attention just because of its easiness in description and the immense difficulty for the solution. In 1920's this problem was considered initially. In simple wording the problem can be described as: "if a traveling salesman wants to visit exactly once each of n cities starting from the city S (the starting city), also keeping in the mind that the cost of traveling from city i to city j is cij , and then comes back to the initializing city, what is the minimal cost for the entire trip traveling salesman can take"? [1],[2]
If there are n cities, then a traveling salesman can have of n! possible ways to start form any arbitrary city and end on the same. Now coming on to get the minimal path we consider an undirected graph G = (n, m) consisting of n nodes (cities) and m links together with costs for each link is cij. The traveling salesman problem (TSP) is to find a tour of minimum cost.
For this sort of problem there are three types of solutions [5].

Sheikh Muhammad Azam is with the COMSATS Institute of Information Technology Abbottabad, NWFP, Pakistan. (e-mail: azam@ciit.net.pk & phone: 0092 333 6541406).
Masood-ur-Rehman is with the COMSATS Institute of Information Technology Abbottabad, NWFP, Pakistan. (e-mail: masoodqau@ciit.net.pk & phone: 0092 300 9507971).
Adnan Khalid Bhatti is with the COMSATS Institute of Information Technology Abbottabad, NWFP, Pakistan. (e-mail: adnankb@ciit.net.pk & phone: 0092 300 9845504).
Nadeem Daudpota is with the COMSATS Institute of Information Technology Abbottabad, NWFP, Pakistan. (e-mail: daudpota@ciit.net.pk & phone: 0092 992 383591).

A. Exact Algorithms

Exact algorithms guarantee to get an optimal solution, but the only drawback for it is that it takes exponential number of repetition of steps. An exact algorithm for TSP it to solve it by using branch and bound algorithm which takes care of all possible tours and neglects only those about which its sure that they could not produce optimal tour [8],[9]. At the end the best so far gives the optimal solution. These algorithms have complexity in exponentials.

B. Approximation Algorithms

Algorithms of this type have polynomial worst-case time complexity, supplying a suboptimal solution with a guaranteed bound on the degree of sub optimality. [8],[13]. Nearest neighbor, minimum link and triangle inequality methods are examples of approximation algorithms.

C. Heuristic Algorithms

These algorithms supply suboptimal solutions but they do not have any bound on their quality. They do not necessarily provide polynomial running times, but on the basis of observation and experiments they often provide a successful tradeoff between solution optimality of the problem and the speed of computing the problem. There are many approaches for these sorts of algorithms but out of those two approaches that have been used for TSP are: local search and simulated annealing.

"Heuristic algorithms" is of the most important and promising research area in which there had been a lot of work done in the recent past. "These heuristics, utilizing analogies with natural or social systems are used to derive non-deterministic heuristic methods and obtain very good results in NP-hard combinatorial optimization problems"[3]. A heuristic is a commonsense rule or set of rules which is used to increase the probability of solving some problems. These type of algorithms are basically general foundation for the research methodologies [5],[6],[7].

II. Analysis of Tradeoff Between Optimality and Speed

The TSP problem belongs to the group of combinational optimization problems which are called as NP-complete problems. Distinctively, if a person can get a solution by using some good and efficient algorithm which could assure to have the optimal solution within a polynomial number of steps, for the traveling salesman problem, then it is all possible that

efficient algorithms could be found for all other problems in the NP-complete class. But to date, no one has been able to find a polynomial-time algorithm to solve the Traveling Salesman Problem. But it does not mean that it is impossible to solve a large instance of this type of problems. Many large scale practical optimization problems have been continuously solved to optimality repeatedly. In the year 1994, Applegate, et. al. gave a solution of a traveling salesman problem that produces model for the production of printed circuit boards having 7,397 holes (cities), Then, in the year 1998, the same authors gave optimal solution of a problem for more that 13,509 largest cities in the U.S.[1] So, the computational record of specific illustrations of TSP problems coming from practical applications is very much optimistic.

The question is how mathematicians and the computer scientists handle such problems? Obviously, they will not consider a brute force approach for such problem. In one example of an 16 city traveling salesman problem -- the problem of Homer's Ulysses (a Greek hero) attempting to visit the cities described in *The series of travel* exactly once -- there are 653,837,184,000 different routes counting the numbers for all such roundtrips to find a shortest roundtrip took 92 hours while been computed on a powerful workstation[2]. In place of enumerating all possibilities of the complete round trips, the successful algorithms for the solution of the Traveling Salesman Problems have been able to eliminate most of the roundtrips that are use less for the computing without the consideration, resulting in the decrease of the time taken for the computation [4]. We will be concentrating on the combination of exact algorithms and Approximations algorithms with a hope in decreasing the computational complexity and at the same time getting optimal or suboptimal solution.

III. Our Proposed Algorithm

"Parallel Branch and Bound model using Logarithmic Sampling (PBLS) for STSP"

PBLS is an approximation algorithm proposed for finding optimal or nearly optimal tours for Symmetric Traveling Salesman Problem. Basically the idea lays its foundation on the Branch and Bound method which is the most famous and widely used [12] approach for solving a given NP-hard discrete optimization problem for the best solution. In simpler words B & B strategy refers to all state space search methods in which a tree search algorithm is basically used. That is the problem of finding a minimum tour in a graph G(V,E) starting and ending at the same node is reduced to performing search in a tree. This tree is known Branch and Bound tree BBT and each node of BBT is actually a tour which includes some particular edges or does not include some particular edges based upon some criteria. Detailed discussion on concepts and various methods employed for B & B strategy can be found in [10],[11] and [12].

We have focused upon attacking the STSP for very large values and tried to devise an algorithm which can be programmed to run in parallel and tries to find optimal tour. To reduce the size of our search matrix, we have applied the concept of **Logarithmic Sampling** with the assumption that logarithmic sampling for larger number of cities will reduce our search space drastically keeping the hope of getting the optimal or nearly optimal tour from the reduced set of search space set. For this purpose we will take Base-2 Logarithm of the total number of cities i.e., let $k = log_2 n$. Now for each city, k minimum cost edges are kept separate and thus reducing the original n^2 element of the search matrix to $n \times k$ which in turn reduces the **(n-1)! / 2** possible tours in the original graph to approximately $(\log_2 n)^{(n-1)}$ tours only.

Doing all this makes the algorithm computable in a reasonable amount of time on a trivial PC without loosing the hope of getting good results, although for very large values of n a distributed parallel processing model can also be programmed.

Let **G (V,E)** be Symmetric Weighted Graph where
$V = \{v1, \ldots, vn\}$ be a set of **n** cities,
$E = \{(r,s) : r,s \in V\}$ be the edge set,
and **Crs = Csr** be a cost measure associated with edge $(r,s) \in E$.
M be the **n** × **n** symmetric matrix containing all edges **Є E**.
Let $k = log_2 n$ and **S** be the set containing first k minimum cost edges for every city $v \in V$.

So taking $v \in V$ be the root of BBT and starting a tour.

We will now perform a complete search on all the possible paths but restricting our search with following rules so as to avoid exploring the paths which are not of our interest:

1) K threads will be started exploring different paths spawning from the k minimum cost edges of the root v from set S,
2) Any subsequent nodes in the path during search of every thread minimum cost edge will be preferred over others when there are more that one children of any node.
3) Threads will share a common memory location called **Upper Bound** where to place the cost of best tour found so far. Initially it will contain infinity.
4) Whenever a thread computes a path it compares its total cost value with Upper Bound. Upper Bound will be assigned this path value if it is lesser than the value of upper bound.
5) Let **Best Path** be the list where we maintain the path sequence of latest value of Upper Bound.
6) Search will continue in this way until there is no path unexplored.
7) As the search terminates we will have the best solution found so far in the list **Best Path** with its cost in **Upper Bound**.

Here is the pseudo code for the function which each of threads will be executing. For this purpose we define a structure **VisitedList** which possesses attributes like:
array of nodes; *// Each node is capable of storing an element of set S.*
FirstElement; *// means the root node for which the thread has started executing.*

```
TotalCost;   // holds the cost of the tour.
Count; // used to count the number of cities(nodes) visited so far.

VisitStep(X)
{ // X is the current node where we are and want to move through to find a tour
        if(VisitedList.Count = = 0)
        // means if list is emty then add the X in it
                VisitedList.Add(X);
        L = GetNextMinLink(X);
        // It returns next minimum cost node for X
        from set S and if all K has been returned then return null
        if(L = = null)
        // no more branches exists for this node
                return;
        else if(VisitedList.Count = = n)
        // that is a tour is complete
                return VisitedList;
        else if(VisitedList.Exists(L) = = true)
        // node L is already visited
                VisitStep(X);
        // so visit other branches of parent of L which is X
          else
          {
                VisitedList.Add(L);
          // so L is not already visited so add it into list of visited nodes
                C = Cost(X,L);
        // get cost of traveling from node X to L
                VisitedList.UpdateTourCost(C);
              // add cost C to the cost of tour so far
                VisitStep(L);
    // keep visiting to branches recursively unless base conditions are not met
          }
}
```

Here we solve an example to illustrate the process.

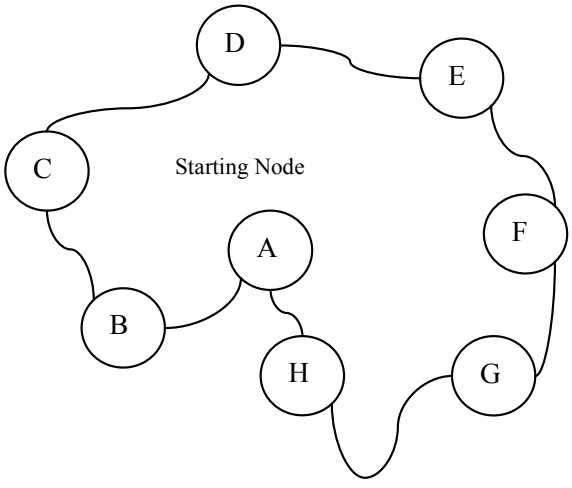

Fig. 1 Graph showing STSP with 8 cities

Let there be 8 cities numbered A-H as shown in fig 1. Its symmetric version of TSP problem, so every city will have edges connecting it to the all other. So adjacency matrix of the graph of 8 cities along with edges and their weights will be 8 by 8 matrix as shown in table I.

TABLE I
ADJACENCY MATRIX FOR THE GRAPH IN FIG. 1

	A	B	C	D	E	F	G	H
A	0	11	24	25	30	29	15	15
B	11	0	13	20	32	37	17	17
C	24	13	0	16	30	39	29	22
D	25	20	16	0	15	23	18	12
E	30	32	30	15	0	9	23	15
F	29	37	39	23	9	0	14	21
G	15	17	29	18	23	14	0	7
H	15	17	22	12	15	21	7	0

TABLE II
TABLE I AFTER APPLYING THE LOGARITHMIC SAMPLING

A	11,B	15,G	15,H
B	11,A	13,C	17,G
C	13,B	16,D	22,H
D	12,H	15,E	16,C
E	9,F	15,D	15,H
F	9,E	14,G	21,H
G	7,H	14,F	15,A
H	7,G	12,D	15,A

II. Conclusion

An n=8 instance of STSP is taken. Its optimal solution was known to be 100. We applied our algorithm on that. Table I is the original adjacency matrix, and after applying logarithmic sampling we got an 8 by 3 matrix as shown in Table II. Fig. 2 shows the search process in the Table II matrix using the rules we mentioned in our algorithm. One can see the second path computed was the one required and all the subsequent paths will be just ignored as no other will give cost less that 100.

In worst case if we had to compute all the paths before getting the best one even then beauty of applying logarithmic sampling is that **(n-1)! / 2** possible tours in the original graph of STSP when solved using exact algorithms are reduced to approximately $\mathbf{(\log_2 n)^{(n-1)}}$ tours only. So if we talk in terms of time complexity comparison between the two approaches, it is obvious that **O((n-1)! / 2)** in case of exact algorithm is replaced with $\mathbf{O((\log_2 n)^{(n-1)})}$ plus the sorting cost after applying logarithmic sampling for PBLS. A comparison between number of possible tours in both the cases is shown in the graph in fig. 3.

III. Future Work

As this is a simple implementation requiring threading, a more complex version for highly large values of n can be implemented. So as our next step we plan to propose a distributed parallel processing model using remoting facility of dot NET framework.

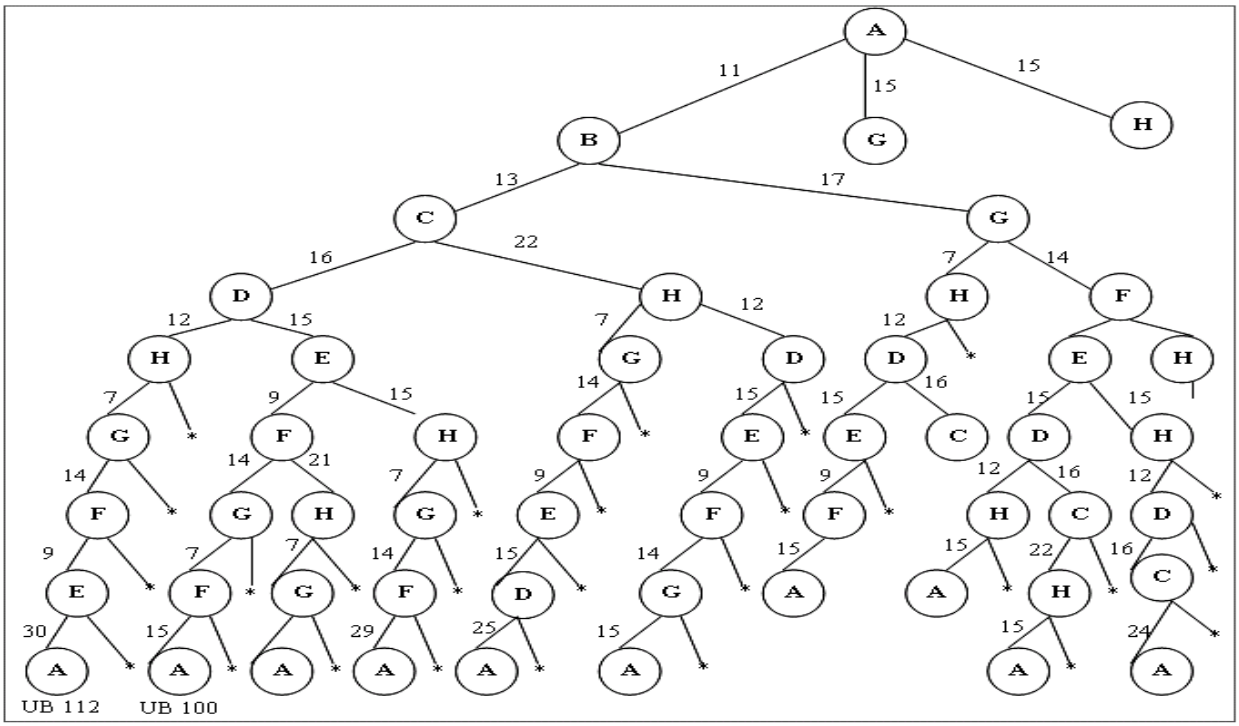

Fig. 2 Search process on the matrix in table ii. All possible branches will be explored. Here only branch 'b' of 'a' is shown. Upper bound is maintained after the completion of each successful path.

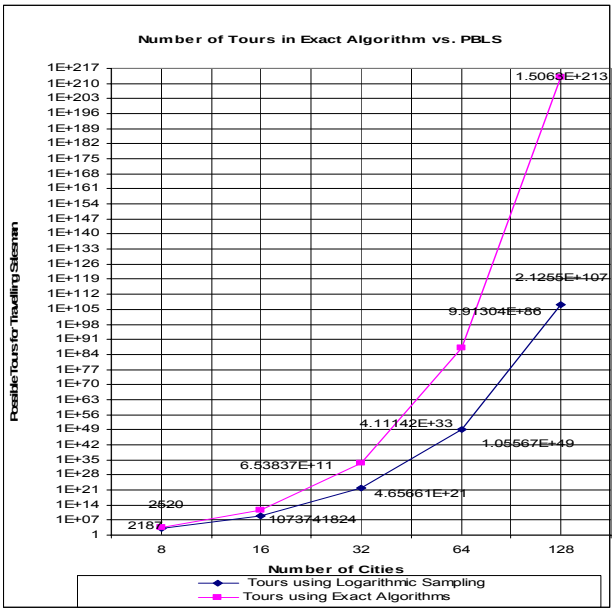

Fig. 3 Graph illustrating the comparison of pbls and exact algorithms with respect to number of tours

REFERENCES

[1] Applegate, R.E. Bixby, V. Chvatal, and W. Cook (1994) "*Finding cuts in the TSP*" a preliminary report distributed at The Mathematical Programming Symposium, Ann Arbor, Michigan, August 1994.

[2] D. Applegate, R.E. Bixby, V. Chvatal, and W. Cook (1998) "*On the solution of traveling salesman problems*" Documenta Mathematica - Extra Volume, ICM III 645-658.

[3] Irina Dumitrescu and Thomas St¨utzle," *Combinations of Local Search and Exact Algorithms*", S. Cagnoni et al. (Eds.): EvoWorkshops 2003, LNCS 2611, pp. 211–223, 2003.

[4] Chantal Korostensky, and Gaston H. Gonnet, "*Using traveling salesman problem algorihms for evolutionary tree consturction*" Institute of scientific computing , 8092 ETH Zurich, Switzerland, January 23,2000.

[5] Bapna, S. De, S."*An intelligent search strategy for solving the symmetric traveling salesman problem*" Systems, Man, and Cybernetics, 1991. 'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference, pp579-583 vol.1

[6] Jano I . van Hemert and Neil B. Urquhart, "*Phase transition properties of clustered traveling salesman problem instances generated with evolutionary computation*" Parallel Problem Solvers from Nature VIII, pages 150–159, 2004.

[7] Chuan-Kang Ting; Sheng-Tun Li; Chungnan Lee; **"***TGA: a new integrated approach to evolutionary algorithms***"** Evolutionary Computation, 2001. IEEE Proceedings of the 2001 Congress,Volume 2, 27-30 May 2001 PP:917 – 924.

[8] Chris Walshaw, "*A Multilevel Lin-Kernighan-Helsgaun Algorithm for the Travelling Salesman Problem*" Mathematics Research Report: 01/IM/80, September 27, 2001.

[9] Irina Dumitrescu and Thomas St¨utzle, **"***Combinations of Local Search and Exact Algorithms*"S. Cagnoni et al. (Eds.): EvoWorkshops 2003, LNCS 2611, pp. 211–223, 2003.

[10] Freisleben, B.; Merz, P.; "*A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems*" Evolutionary Computation, 1996., Proceedings of IEEE International Conference on,20-22 May 1996 Page(s):616 – 621.

[11] Jens Clausen, "*Branch and bound algorithms principles and examples* ", department of comp sc., university of Copenhagen, Universitetsparken 1, DK-2100 Corpenhagen, Denmark.(March 12,1999)

[12] R. C. T. Lee, R. C. Chang, S.S. Tseng, and Y. T. Tsai, *An Introduction to the Design and Analysis of Algorithms*, 2nd Ed, FLAG Publishers, 2002.

[13] Grigni, M.; Koutsoupias, E.; Papadimitriou, C.; "*An approximation scheme for planar graph TSP*" Foundations of Computer Science, 1995. Proceedings., IEEE ,36th Annual Symposium,23-25 Oct. 1995 pp:640 – 645.