A Parallel Multi-Phase Implementation of Simulated Annealing for the Traveling Salesman Problem *

Dasaradh R. Mallampati
Pooja P. Mutalik
Roger L. Wainwright

Department of Mathematical and Computer Sciences
The University of Tulsa
Tulsa, Oklahoma 74104

Abstract

This paper describes and analyses a new parallel algorithm using simulated annealing for finding a good solution to the Traveling Salesman Problem. This algorithm combines the strong points of three recent implementations [1,2,5] with some new features. An initial tour is generated and partitioned among a ring of processors. Each processor receives two disconnected parts (tiers) of the tour. The algorithm is subdivided into three phases. In phase one, 2-opting is performed separately within each of the two tiers of the tour. During the second phase remote-swapping is performed between cities from the two different tiers of the tour. During phase three, synchronization of the cities is accomplished by each processor shifting a quarter of its cities in a clock-wise direction to its neighboring node. This is called a quarter-spin. Results show this algorithm is superior over recent implementations. For the datasets tested, this algorithm yielded improvements ranging from 32% to 56% compared to three recent implementations. The significance of this algorithm is the manner in which cities from different parts of the tour are combined to form new tours. The multiple phases within the algorithm allows for a better mixture of cities compared to previous algorithms.

1 Introduction

Given a set of n points in a plane corresponding to the location of n cities, find the minimum distance closed path that visits each city exactly once. This is called the Traveling Salesman Problem (TSP). In this paper it is assumed each city is directly connected to every other city by euclidean distance. In recent years, researchers have primarily used two different techniques for determining a near optimal local minimum solution to the traveling salesman problem: genetic algorithms [3,4,9], and simulated annealing (SA) [1,2,5]. In this paper we are concerned with the simulated annealing method for locating a near optimal solution.

2 Simulated Annealing

Simulated annealing is a stochastic computational technique for finding near globally-minimum-cost solutions to large optimization problems. The method of simulated annealing is an analogy with thermodynamics, specifically in the manner that metals cool (anneal), or in the way liquids freeze and crystallize. For more information the reader is referred to Metropolis [8] and Kirkpatrick [6].

In the Traveling Salesman Problem, simulated annealing starts with an arbitrary initial tour. The objective function, which is the total length of the tour, is analogous to the current energy state of the system. Changes from one state to another (ie., a new tour from a previous tour) that result in a reduced tour length are always accepted. This is a analogous to slow cooling. However, changes that increase the tour length are only accepted with probability $p(d,T) = \exp(-d/T)$, where d is the change in the tour length, and T is the "temperature" of the system. The temperature parameter controls the annealing process. By occasionally allowing the energy of the system to rise slightly before cooling again is analogous to occasionally allowing a longer tour. This may allow the system to avoid falling into a local minima where it cannot get out. Braschi [2] gives an excellent discussion of simulated annealing applied to the Traveling Salesman Problem.

* Research partially supported by OCAST Grant AR0-038 and Sun Microsystems, Inc.

Reprint: Proceedings of the Sixth Distributed Memory Computing Conference
April 28 - May 1, 1991, Portland, Oregon.

0-8186-2290-3/91/0000/0488/$01.00 © 1991 IEEE

3 Parallel TSP Using SA

The algorithm for implementing the Traveling Salesman Problem using simulated annealing has two phases. Phase 1 is concerned with the distribution of the tour among the processors and the interaction among the "internal cities" within each processor. In phase 2 some interconnection scheme is used to move or redistribute the cities of the tour among the processors. Allwright and Carpenter [1], Braschi [2], and Felton, Karlin and Otto [5] have investigated the Traveling Salesman Problem on a hypercube using simulated annealing. Furthermore, each of these researchers have taken different approaches to these two phases. A detailed review of each of these three implementations including advantages and disadvantages of each is given in [7].

4 A Parallel Multi-Phase Implementation

In our parallel multi-phase implementation we have combined the strong points of each of the previous approaches and added some new features. Our algorithm uses a ring topology hypercube arrangement. An initial tour is generated and partitioned among the processors. Each processor receives two disconnected parts (tiers) of the tour as shown in Fig. 1a. The algorithm is subdivided into three phases. In phase one, 2-opting is performed separately within each of the two tiers of the tour (see Fig 1b). During the second phase remote-swapping is performed between cities from the two different tiers of the tour (see Fig. 1c). During phase three, synchronization of the cities is accomplished by each processor shifting a quarter of its cities in a clock-wise direction to its neighboring node. This is called a quarter-spin (see Fig 1d).

Phase one of the algorithm performs local minimization whereas phase two enables two adjacent cities to be brought together which would otherwise end up in separate tiers. These two phases, even implemented collectively, do not eliminate completely the local minima problem. Therefore, a quarter-spin is introduced to give rise to new combinations that will enhance the performance of 2-opting and remote-swapping. A quarter-spin results in each node retaining 50% of its cities and receiving 50% new cities. The quarter-spin begins by node 0 initiating a data transfer by sending a quarter of its cities to its neighbor. Before node 0 receives a quarter of the cities from its other neighbor in the ring, it begins to rearrange its data internally and starts 2-opting on the first half. Similarly, other nodes perform double duty (message passing and 2-opting). As a result, the idle time of the processors in our algorithm and the amount of data transferred is much less than that of other algorithms. Our algorithm obtains a good mix of the cities of the tour. The simulated annealing technique surrounds the three phases of the algorithm. The parallel multi-phase algorithm is described below:

1. Compute an initial tour (path), p.
2. Choose the initial temperature parameter, T
3. *Loop*
4. *Loop*
5. *Loop*
   Compute new state, p' using 2-opting on the top tier of the tour.
   p = f(p',p)
   Until equilibrium
6. *Loop*
   Compute new state, p' using 2-opting on the bottom tier of the tour.
   p = f(p',p)
   Until equilibrium
7. *Loop*
   Compute new state, p' using remote-swapping between cities from the two tiers of the tour
   p = f(p',p)
   Until equilibrium
8. Shift clockwise a quarter of the cities to the neighboring processor.
   Until equilibrium
9. Decrease T
   Until crystallization

function f(p',p)
1. *If* p' is a better path than p, *then return*(p')
2. d = length of path p' - length of path p
3. *return*(p') with probability minimum(1,exp(-d/T)) otherwise *return*(p)
4. *end function*

5 Results

Table I shows the results of our experiments. We implemented Felton's algorithm, a modified version of Allwright's algorithm, and our Multi-phase algorithm. For comparison purposes we also show results using the Multi-phase algorithm without 2-opting tiers, and without remote-swapping. The Multi-phase algorithm without remote-swapping is similar to Braschi's algorithm. The

modified Allwright algorithm is an improvement over his original algorithm. Allwright's algorithm was modified to shift the cities using a quarter-spin. This is an improvement over his random spin. Allwright takes two cities from each of the two tiers to perform his 2-opting. In our remote-swapping, three cities are taken from each of the two tiers and only the middle cities get swapped. In most cases the end result is identical to Allwright's 2-opting, except that remote-swapping is a faster approach. The Multi-phase algorithm without remote-swapping represents an improvement to Braschi's algorithm. We refer to this algorithm as the Braschi-like algorithm.

Each algorithm was tested using datasets of 320 and 1024 cities. The cities were randomly placed in a square of length 500 on a side. The same initial tour was used in each algorithm. In each case the algorithms were run on an iPSC hypercube using 32 processors. In each case the initial temperature was set to 1.0. The temperature decreased by 0.01 at each step until a final value of 0.01. Furthermore, each algorithm was implemented such that the temperature in the system was decreased either (a) after one iteration per loop or (b) after equilibrium. Equilibrium is achieved when no changes occur during an iteration or 10 iterations, whichever comes first.

Results show the Multi-phase algorithm to be vastly superior to Felton's algorithm, modified Allwright's algorithm, and our Braschi-like algorithm (Multi-phase without remote-swapping) in both examples tested. Furthermore, the execution time for the Multi-phase algorithm was essentially the same as the other algorithms for 320 cities, and only slightly higher for 1024 cities. Results also show how essential remote-swapping is to the excellent performance of the Multi-phase algorithm. Remote-swapping allows for a good mixture of the cities. Note in Table I the performance of the Multi-phase algorithm with out remote-swapping.

Table I shows the results of the Multi-phase algorithm when each of the three phase loops were allowed to execute until equilibrium. This had little effect on the 320 city example. Neither the CPU time nor the tour length was effected very much. However, in the larger example of 1024 cities, an improvement was noted. The Multi-phase algorithm improved nearly 4% over the one iteration version. This cost 40% more in CPU time, however. This suggests more analysis may be needed in developing a better equilibrium strategy.

6 Summary

A new parallel algorithm using simulated annealing for finding a good solution to the Traveling Salesman Problem is presented. Results show the new Multi-phase algorithm to be vastly superior to three recently implemented algorithms. The solution from the Multi-phase algorithm represents a 44%, 56%, and 52% improvement over Felton's algorithm, modified Allwright's algorithm, and the Braschi-like algorithm, respectively, for 320 cities. An improvement of 39%, 30%, and 55% occurred respectively, in the case of 1024 cities. The remote swapping between the two tiers is the key to the excellent performance of the Multi-phase algorithm. The multiple phases within this algorithm allows for an excellent mixing of the cities compared to previous algorithms.

References

[1] J. Allwright and D. Carpenter, A Distributed implementation of Simulated Annealing for the Traveling Salesman Problem, *Parallel Computing* 10 (1989) 335-338.

[2] B. Braschi, Solving the Traveling Salesman Problem Using The Simulated Annealing on a Hypercube, *Proceedings of the Fourth Conference on Hypercubes, Concurrent Computers, and Applications*, March, 1989.

[3] L. Davis, Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publisher, 1987.

[4] K.A. De Jong and W.M. Spears, Using Genetic Algorithms to Solve NP- Complete Problems, *Proceedings of the Third International Conference on Genetic Algorithms*, June, 1989, pp. 124-132.

[5] E. Felton, S. Karlin and S. Otto, The Traveling Salesman Problem on a Hypercubic, MIMD Computer, *Proceedings of the 1985 International Conference on Parallel Processing*, August, 1985.

[6] S. Kirkpatrick, C.D. Gelatt Jr. and M.P. Vecchi, Optimization by Simulated Annealing, *Science*, May, 1983, vol. 220, pp 671-680.

[7] D. Mallampati, P. Mutalik, R. Wainwright, A Parallel Multi-Phase Implementation of Simulated Annealing for the Traveling Salesman Problem, Computer Science Technical Report No. CS-91-12, University of Tulsa.

[8] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equation of State Calculation by Fast Computing Machines *J. Chem. Phys.* vol. 21, 1953, p. 1087.

[9] D. Whitney, T. Starkweather and D. Fuquat, Schedu ing Problems and Traveling Salesman: The Genetic Edg Recombination Operator, *Proceedings of the Third Inte national Conference on Genetic Algorithms*, June, 1989.

Table I

Comparison of Four Parallel TSP SA Algorithms on 320 and 1024 Cities

Algorithm	Number of Cities	Initial Tour Dist.	Final Tour Distance			
			One Iteration	CPU Sec.	Equilib-rium	CPU Sec.
Felton	320	78,546	52,109	106		
Modifed Allwright	320	78,546	66,813	272		
Multi-Phase w/o 2-opting	320	78,546	34,146	264	33,633	269
Multi-Phase w/o Remote-Swap	320	78,546	61,607	269	61,622	273
Multi-Phase	320	78,546	29,339	272	30,836	272
Felton	1024	258,624	150,708	285		
Modified Allwright	1024	258,624	126,734	274		
Multi-Phase w/o 2-opting	1024	258,624	106,138	287	103,834	440
Multi-Phase w/o Remote-Swap	1024	258,624	202,273	295	202,056	317
Multi-Phase	1024	258,624	91,323	336	87,827	471

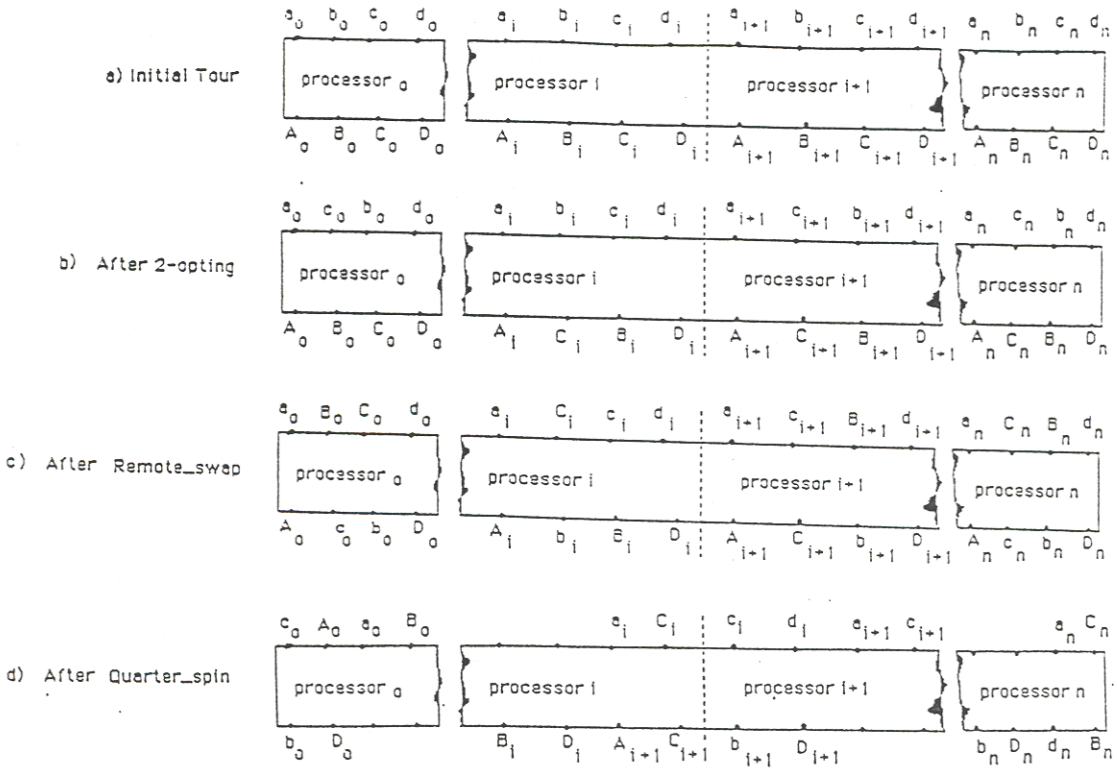

Fig 1 : Successive steps in Multi-phase Algorithm