A HIGH PERFORMANCE ALGORITHM FOR SOLVING LARGE SCALE TRAVELLING SALESMAN PROBLEM USING DISTRIBUTED MEMORY ARCHITECTURES

Khushboo Aggarwal1,Sunil Kumar Singh2, Sakar Khattar3

1,3 UG Research Scholar, Bharati Vidyapeeth's College of Engineering
2 Assistant Professor, Computer Science Department, Bharati Vidyapeeth's College of Engineering

ABSTRACT

In this paper, we present an intelligent solution system for travelling salesman problem. The solution has three stages. The first stage uses Clustering Analysis in Data Mining to classify all customers by a number of attributes, such as distance, demand level, the density of customer, and city layout. The second stage introduces how to generate feasible routing schemes for each vehicle type. Specifically, a depth-first search algorithm with control rules is presented to generate feasible routing schemes. In the last stage, a genetic programming model is applied to find the best possible solution. Finally, we present a paradigm for using this algorithm for distributed memory architectures to gain the benefits of parallel processing.

Keywords: Travelling Salesman problem, Genetic algorithms, fitness functions

1. INTRODUCTION

Travelling Salesman Problem is a problem in computational mathematics aimed at finding the shortest path visiting every city in a given set of cities, along with known distances for each pair of cities, and return to the starting position. For a given complete graph of n nodes, with nodes representing the set of cities, and with edge lengths representing the distance between each pair of respective cities, Travelling Salesman Problem aims at finding the most efficient Hamiltonian Cycle a salesman can follow through each of the n cities present.

The Travelling Salesman Problem has been categorized as NP-Hard, and an ideal solution for the problem cannot be found in polynomial time. There is no known efficient algorithm to calculate the best solution for the problem, and the time required in such a solution would increase exponentially with increase in problem size, that is, time to solve the problem in such a case would be an expression that would contain size of the problem n as an exponent. Thus, we seek to find an efficient solution close to the ideal solution, which can be found in polynomial time.

Using Distributed Memory Architectures for solving the Travelling Salesman Problem decreases the computation time required to compute the solution by making use of parallel computing. We categorize the set of cities into different classes based on their respective region. Each different class is then fed to an individual processor which is a part of the Distributed Memory Architecture. Different processors then operate on different data and use genetic algorithm to find the optimum path solution for the given region. A single processor then combines the resultant outputs of different regions to give the final solution to the Travelling Salesman Problem.

The major advantage of using Distributed Memory Architecture is the faster computation of the final solution by dividing the main problem into independent sub problems, which are then solved concurrently in different parallel processors, thus decreasing computation time. It also facilitates easy scalability of the problem. For large scale problems, number of processors operating parallel in the Distributed Memory Architecture can be increased, to work on a larger set of cities, thus restricting the increase in computation time to a minimum level. A disadvantage of using Distributed Memory Architecture is that the quality of final solution is degraded when the solutions of different regions are combined, since there is no co-relation between the data of different classes processed by different processors, and the intermediate path solutions of different regions are completely independent of each other.

2. LITERATURE SURVEY

A. The Traveling Salesman problem (TSP)

The Traveling Salesman Problem (TSP) is an NP-hard problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pair wise distances, the task is to find a shortest possible tour that visits each city exactly once.
With metric distances: In the metric TSP, also known as delta-TSP or Δ-TSP, the intercity distances satisfy the triangle inequality. This can be understood as "no shortcuts", in the sense that the direct connection from A to
B is never longer than the detour via C.

$Cij<=Cik+Ckj$

Exact algorithms: The most direct solution would be to try all permutations (ordered combinations) and see which one is cheapest (using brute force search). The running time for this approach lies within a polynomial factor of O (n!), the factorial of the number of cities, so this solution becomes impractical even for only 20 cities. One of the earliest applications of dynamic programming is an algorithm that solves the problem in time O (n22n) [2].
The dynamic programming solution requires exponential space. Using inclusion–exclusion, the problem can be solved in time within a polynomial factor of 2n and polynomial space. Improving these time bounds seems to be difficult. For example, it is an open problem if there exists an exact algorithm for TSP that runs in time O (1.9999n).

B. Genetic Algorithms

Genetic algorithms are an optimization technique based on natural evolution. They include the survival of the fittest idea into a search algorithm which provides a method of searching which does not need to explore every possible solution in the feasible region to obtain a good result. Genetic algorithms are based on the natural process of evolution. In nature, the fittest individuals are most likely to survive and mate; therefore the next generation should be fitter and healthier because they were bred from healthy parents. This same idea is applied to a problem by first 'guessing' solutions and then combining the fittest solutions to create a new generation of solutions which should be better than the previous generation. We also include a random mutation element to account for the occasional 'mishap' in nature.

3. ALGORITHM

Customer classification

Customer classification is the primary task in the first stage to solve the PRP. In this stage, clustering analysis in Data Mining is used to classify all customers into some areas by some attributes, such as distance, demand level, the density of customer, city layout, and others. If the customers in one area are considered to have some similar parameters, then the computing complexity can be greatly reduced. Nevertheless, customer classification is still an important and complicated task in the approach. We are using neural networks to cluster.[4]

Generating vehicle routing scheme (dfs)

A feasible vehicle routing scheme refers to a vehicle's routing plan that starts with the depot and ends with the last customer that either uses up the full vehicle load or satisfies the given time window. Only after all feasible routing schemes are obtained can we construct a PRP optimization model. When a vehicle leaves from the central depot, it must choose one of all customers as its serving target. After the customer is served, the vehicle will choose one of the remaining un-served customers as its delivery stop. This process continues until the travel time or the capacity constraint is violated.

A routing scheme is a path spanning through the state space from an initial state to a goal state. In this case, the central depot is a search node corresponding to the initial state, and the last customer a vehicle will serve within the travel time or the load constraint is the goal state. Thus, the generation of vehicle routing schemes is turned into the path searching through the state-space. We adopt the depth-first search strategy in Artificial Intelligence to generate the routing schemes, and use the travel time and capacity to control the search depth

Optimizing the problem (genetic algorithm)

Once the feasible routing schemes are identified, it is necessary to identify the best scheme. We develop an integer programming model to accomplish the task. The integer model aims to minimize the total vehicle operating cost subject to several constraints. Optimization using Genetic Algorithms.

Algorithms exist that can solve this problem exactly. However, as the number of cities grow, the time required to find the solution grows exponentially. When you get above, say, 100 cities, you need a rather powerful computer if you want the solution this side of Christmas.

If you want a fast answer and you can live with not having the exact solution, the way to solve the TSP problem is by heuristics. A common heuristic method is to start in a city and then travel to the nearest city and so on, until all cities have been visited.
This program uses a genetic algorithm as a heuristic approach. A genetic algorithm mimicks natural evolution: Out of a population of individuals (which in fact each represent a proposed solutions in the form of a route between all cities) the fittest are selected to mate, creating new individuals (new proposed routes).

Basically, with the proper criteria and methods for fitness, selection, and reproduction, some of the new individuals represent better solutions (in this case less overall distance). These individuals are then selected for a new generation, and so on. That is how a genetic algorithm works.[3]

4. PSEUDOCODE

```
ALGORITHM: GATSP

Begin
Step 1: Classify the cities according to their distance from the central depot.

Class A = 0<dist<3
Class B = 3<dist<6
Class C= 6<dist<9
Class D= 9<dist<12
Class E= 12<dist<15

Step 2: For each class X do the following

Step 3: Apply DFS(X) where X is a set of cities in the class to generate the initial population P0

Step 4: Using P0 as the first generation, Apply GA to the P0 and continue for 512 generations.
```

The new generations are made using the GreedyCrossOver Technique.

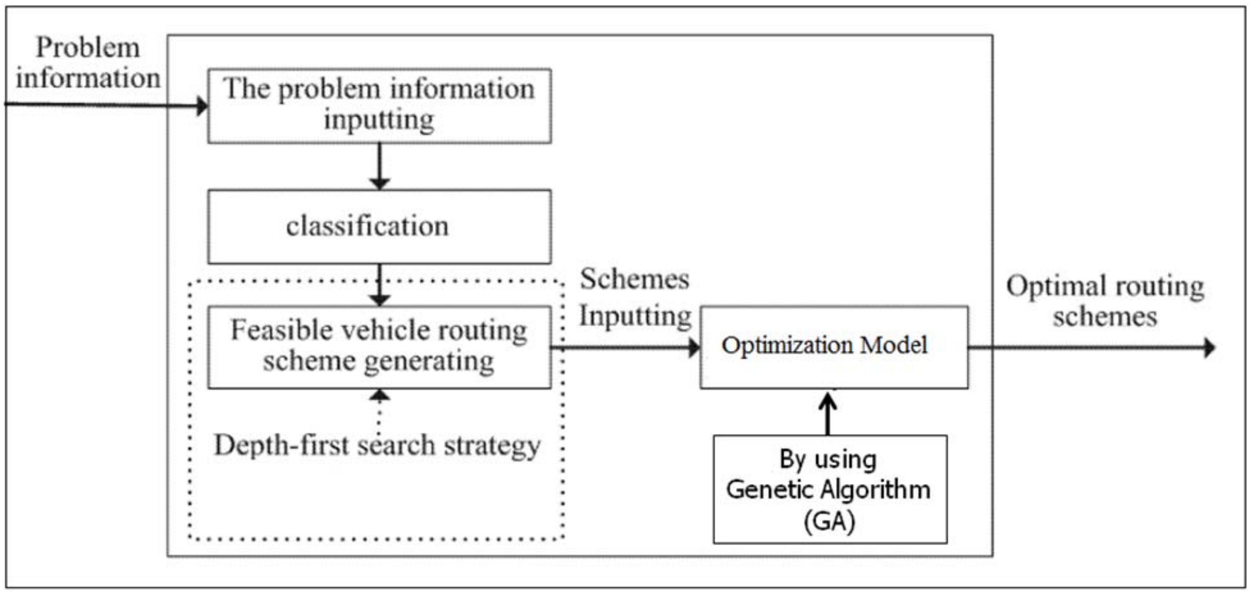
Fitness function used is

Fitness=x4+y3+z2+u

x= Distance
y=road conditions
z=traffic
u=drivers experience

Step 5: Combine the solution of each class to give the final result.

Step 6: Calulate the final distance for the path and display


Problem information
The problem information inputting
classification
Feasible vehicle routing scheme generating
Depth-first search strategy
Schemes Inputting
Optimization Model
By using Genetic Algorithm (GA)
Optimal routing schemes

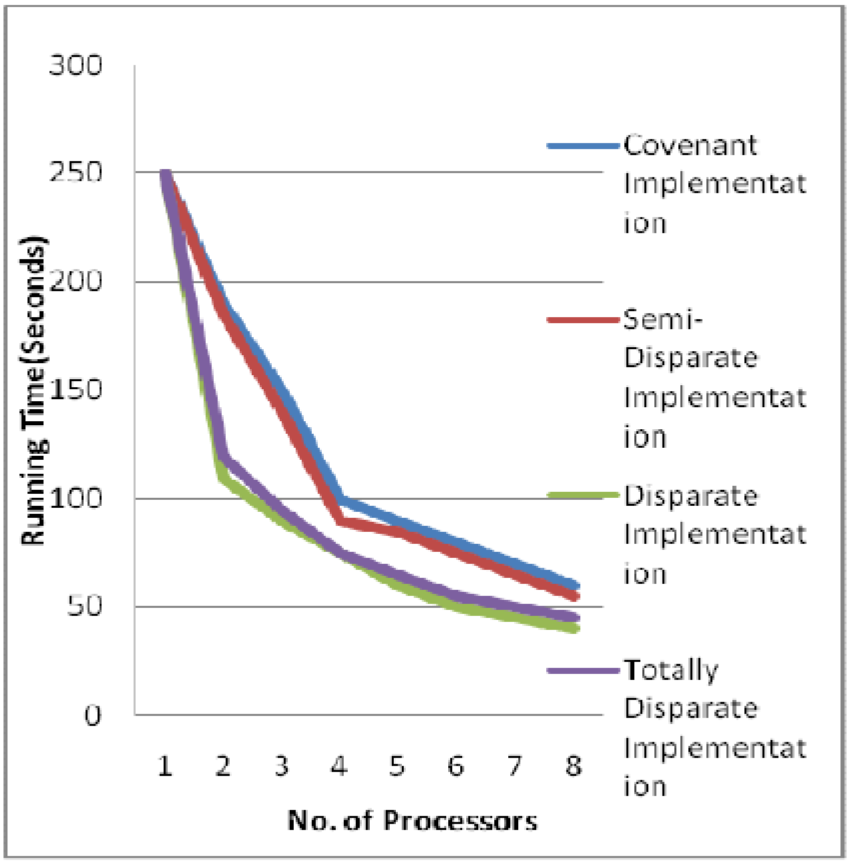
5.IMPLEMENTATION ON DISTRIBUTED MEMORY ARCHITECTURE

Many architectures for implementation of genetic algorithms on distributed architectures have been proposed in [5]. These include covenant architecture in which all the processors communicate after each generation of the algorithm. Another extreme is the disparate implementation in which the processors do not communicate at all withing successive generation and give the final result by just combining them in the last stage. Between these two extremes a trade off of performance and search quality exists. The more the communication between the processors the better the quality of a solution. Thus the use of the architecture is dependent on the application for which we use the algorithm. In case of criticial applications like protein structure of medicines etc. we need the search to be accurate and thus, we use a covenant architecture of implementation. [6]

However, In our case, the performance of the Travelling Salesman Algorithm is more critical. This is because as the number of cities increases the problem becomes intractable. Thus, we use a disparate implementation. This will give us a nearly good solution in the minimum time possible.[7]

6. PERFORMANCE ANALYSIS

We reviewed the performance of the above algorithm on each of the proposed architectures and plotted the results as a graph between the running time and the number of processors used in the model.
It can be seen from the above graph that the running time in a disparate implementation is greatly reduced, and thus, the performance of this implementation is better than a covenant implementation. However there exists a tradeoff between performance and search quality. The choice of the architecture is dependent in the application. If we are using the GA to predict a critical problem where search quality is a crucial factor we prefer a covenant implementation despite its slow running time.



7. TRADE OFF BETWEEN APPLICATION IN COVENANT AND DISPARATE IMPLEMENTATIONS OF PARALLEL GENETIC ALGORITHM

The disparate implementation of the algorithm is better suited to the protein structure prediction as the population is large and its running on a covenant system will take much more time. Both the semi-disparate and disparate implementations show much lesser time complexity and an analysis leads to the result that the semi-disparate is more effective. Thus using a parallel genetic algorithm on distributed memory architecture is the best possible alternative, running in lesser time and yielding better results.

8. CONCLUSION

The algorithm we proposed was a three stage algorithm. The algorithm uses the depth first search algorithm to find the best path through the search space. It then uses the genetic algorithms for optimizing the solutions. We also studied the possibility of using this algorithm on distributed memory architecture. When used with this architecture, the algorithm showed a significant improvement over the other algorithms.

REFERENCES

[1] Artificial intelligence for network routing problems steven willmott and boi faltings from: AAAI technical report WS99-03.

[2] Performance of Intelligent routing in wireless networks Ioannis n. Kassabalidis, arindam K. Das, mohamed A. El-sharkawi, robert J. Marks II university of washington, department of electrical engineering

[3] A swarm intelligent routing algorithm for mobile networks martin roth and stephen wicker wireless intelligent systems laboratory school of electrical and computer engineering,Cornell university

[4] An intelligent solution system for a vehicle routing problem in urban distribution xiangpei hu and minfang huang, international journal of innovative computing, information and control ICIC international °c 2007 ISSN 1349-4198 volume 3, number 1, february 2007

[5] [PLG87] C. B. Petty, M.R. Leuze and J.J.Grefensttete. a Parallel Genetic Algorithm,in proc of 2nd int. conference on Genetiv Algorithms and its applications,1987

[6] Ricardo Bianchini, Christopher Brown, Parallel Genetic Algorithms on Distributed Memory Architecture

[7] A.-A. Tantara, N. Melaba,, E.-G. Talbia, B. Parentb, D. Horvathb, A parallel hybrid genetic algorithm for protein structure prediction on the computational grid