

ARTIFICIAL INTELLIGENCE APPROACH OF OPTIMAL ROUTE SELECTION IN TELEMATICS

Muneer-Ul-Haque, Mohammad S. Raunak, M. Asifur Rahim, Tashfin Delwar and Abul L. Haque

Department of Computer Science & Engineering
North South University, 12 Banani C/A, Dhaka 1213, Bangladesh
Email: muneeritis@hotmail.com, raunak@cs.umass.edu, asif_r@softhome.net,
tashfindelwar@hotmail.com, ahaque@northsouth.edu

ABSTRACT

This paper proposes a novel Genetic Algorithm based Artificial Intelligence approach to solve a variation of traveling salesperson problem. The task is to find the least congested path(s) between two location vertices. In real world, the congestion or traffic data between two location vertices is very dynamic. This problem becomes particularly interesting in the field of location services, or Telematics, where one activity is to use real time spatial data to suggest optimal route for its subscribers. A web based software prototype is developed using proprietary components that implements the suggested Genetic Algorithm engine to analyze and confirm the validity of the new approach. A simulation of Dhaka's dynamic and unpredictable traffic congestion scenario has been used in the prototype.

1. INTRODUCTION

One of the oldest classical problems in the algorithm theory is the Traveling Salesperson Problem (TSP) where the task is to find the optimal tour of all the vertices of a graph from a source vertex [1]. The problem is classified as NP hard. Dijkstra's classic algorithm [1, 2], presented first in 1959, ran in $O(n^2 + m)$ time [2]. The complexity has since been reduced to $O(m + n \log_2 n)$ using Fibonacci heaps [3]. Various improvements have been proposed in [4, 5].

We consider a new model of the TSP problem to find the optimal path from the source to destination that has a practical significance in real life in the field of Telematics. In Telematics, spatial data about people, vehicle or places is used to provide location services, like driving directions from one place to another, finding nearest service spots etc. The subscribers of this service usually login through the service website and ask for specific route suggestions using an interactive location map available. Our objective is to

find the least congested path from any given source to any given destination in a set of location points.

This paper proposes the use of knowledge intensive Genetic Algorithm to approximate the optimal solution to the problem viewed in the described perspective. The proposed algorithm performs with expected efficiency on sparse graphs, and shows reasonable performance for dense graphs where the number of vertices is not too large. For large number of vertices in dense graphs, the performance of the algorithm is expected to gracefully degrade. In real life Telematics, however, the location points are usually sparsely distributed.

2. PROBLEM DEFINITION AND ITS SOLUTION APPROACH

We represent the location nodes as vertices $\{V\}$ of a positively weighted directed cyclic graph $G(V, E)$ where the edge weights $\{E\}$ of the graph represent congestion between location node pairs and are dynamic in nature. Given a source $s \in V$ and a destination $d \in V$, our problem is to find, within a fixed response time, the set(s) of vertices $\{s, v_1, v_2, v_3, \dots, v_m, d\}$ with corresponding edge weights $\{e_{s1}, e_{12}, e_{23}, e_{34}, \dots, e_{md}\}$ such that $\sum e_{ij}$ (where $i = s, 1, 2, 3, \dots, m$, and $j = 1, 2, 3, \dots, d$) is minimal.

In our Genetic Algorithm approach to solve this problem, a set of random initial solutions evolve into a set of optimal solutions after multiple generations, based on theories of natural selection and gene evolution. Genetic operators like selection and mutation etc. are applied to a generation of solutions or chromosomes to transform them into better generation of solutions [8, 11, 12].

3. RELATED WORKS

There are web-based location and driving direction services like MapQuest (www.mapquest.com) and Yahoo Maps (maps.yahoo.com) that are quite popular and useful. These services use proprietary techniques (some version of Dijkstra's algorithm) to come up with the shortest path.

Genetic Algorithms using permutation operators applied to TSP ill performs in two respects, first they scale rather poorly as the number of cities n increases, and second the solution quality degrades rapidly. We have adapted the Evolutionary Divide and Conquer (EDAC) [7] algorithm that uses Genetic Algorithm to explore the space of problem subdivisions in a range of cities rather than the space of solutions themselves. The sub-tours are then patched together to form a global tour. It is much faster than the classical algorithms to compute the cycle time, including Karp's. Iterated Lin-Kernighan [7] is another sophisticated algorithm that was developed for solving large TSPs.

4. WHY GENETIC ALGORITHM?

Genetic Algorithm allows us to use dynamic congestion data that can generate sub optimal solutions within restricted response time. The least congested path may become the most congested one within seconds and other near optimal solutions can rise up ending as optimal. Where other algorithm would require repeating the procedure, Genetic Algorithm achieves this simply by reevaluating the previous generation [10].

Genetic approach requires little knowledge about the response surface [14]. This aids our search for optimal solution a great deal because we hardly have any idea about the congestion pattern among the location nodes.

Genetic Algorithm is particularly known for its usefulness in large-scale optimization problems. It is resistant to becoming trapped in local optima [15].

Genetic Algorithm bears intrinsic parallelism in its features like evaluation functions, genetic operators and their application probabilities etc. [13]. This allows division of program modules and task breakdown to distributed processing machines while attempting to solve non-linear problems like ours.

We can modify the evaluation method presented in [14, 15] to any applicable mathematical or logical function with varying complexity and can also normalize the weights to improve the accuracy of fitness computation for each solution. Our Genetic Algorithm can be customized for large number of nodes with different evaluation methods

and chromosome structure encoding [9, 10, 15]. The algorithm generates many time stamped information on users of the system and finally the optimal path for them.

5. PROPOSED SOLUTION APPROACH

The following is a pseudo-code of the algorithm.

Find_Optimal_Route (G, s, d)

1. Create a chromosome of size n with source s at chromosome [0] and destination d at chromosome [$n-1$] and the rest of the vertices randomly distributed in between. This chromosome is an initial population member for the first generation.
2. Specialize the chromosome into a valid one by matching it with static distance adjacency matrix representing G and name it candidate. Form new chromosome if specialization fails. Compare with existing solutions for duplicates if specialization succeeds. Form new chromosome if duplicate exists.
3. Evaluate the candidate with a dynamic congestion adjacency matrix by computing and assigning its fitness and other attribute values. Store the candidate as a solution chromosome.
4. Rank the solution chromosomes using a customized quick sort algorithm based on their fitness.
5. Mutate or randomly change the vertex sequence pattern of the top ranked chromosomes. Fitter chromosomes are mutated at a higher rate than the weaker ones. Then use the mutated chromosomes as initial population members for the next generation.
6. Continue until desired number of iterations is reached.
7. Return the ranked chromosomes.

The algorithm performs random search through the solution space for potential solutions and then computes its fitness and stores it into present population. The search concentrates more on the regions of the solution space where currently the best-found solutions are residing. Mutation allows new areas of the solution surface to be explored. After the iterations, the found solutions are sampled, reevaluated and ranked to create the next generation. This procedure is repeated until convergence condition is reached. Finally, the ranked solutions are returned with the best solution placed at first.

6. IMPLEMENTATION OF THE CONCEPT

We have developed a prototype software called Gypsy in which a user can log on to its website and select a city map, a particular source and destination, path computation priority (congestion or distance), path computation accuracy (decides the response time for the algorithm), and the optimization level for output path sequence within his or her desired response time. A software GPS simulator feeds dynamic congestion data to the Gypsy database to be used for optimal path calculation. The Genetic Algorithm engine computes the optimal path(s) from the user specified source to the destination and highlights the path on the map to be viewed by the user. The interactions among the software components are shown in Figure 1.

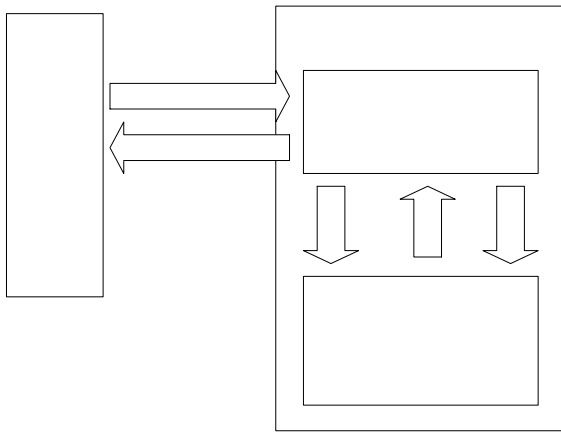


Fig. 1 Block diagram of the software components

7. ANALYSIS

7.1 Time Complexity

In our experiment with 30 vertices, the algorithm came up with the optimal solution in real time. While increasing the number of vertices from 5 to 30 with step size 5 and plotting the corresponding algorithm execution time, we have observed that the resulting graph approximately fits the n^2 polynomial curve. We have averaged the results of 20 runs per increase in the number of vertices. This is shown in Figure 2. This performance was acceptable because the worst case running time for Dijkstra's algorithm is $O(n^2)$. Furthermore, for our suggested application of the algorithm, most of the computation is done for a particular local region of a map, so the achieved performance with 30 nodes should suffice for practical implementation.

7.2 Space Complexity

For a graph with n vertices and m edges, the space requirement is $O(n^2)$. Though this representation has a large space requirement, we chose this because it reduces the time

required for knowing if a certain path exists between two vertices. Other representations like adjacency list etc. would have the added cost of list traversing. The chromosomes were encoded using arrays for simplicity. At worst case, the space requirement for list-based implementation is $O(n)$. A dynamic allocation structure can be used to free up unused spaces.

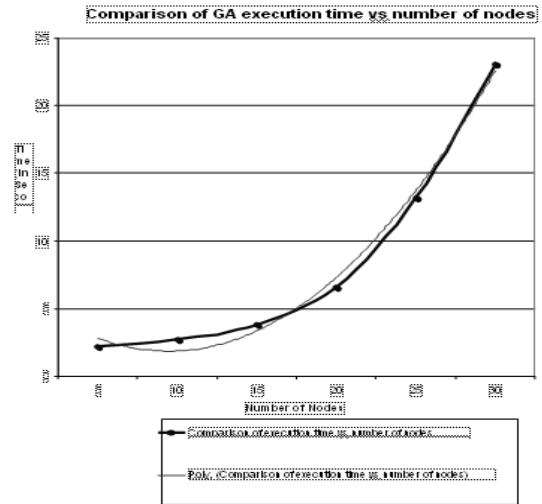


Fig. 2 Genetic Algorithm execution time versus number of nodes

8. IMPLEMENTATION

Borland C++ builder 5 and Microsoft Access were used for the software components and the database respectively. The dynamic linking libraries or ActiveX components were created as active server objects. To display the selected route on the map, a proprietary ActiveX component was developed from scratch using Visual Basic. The organization and the data flow among these components can be explained in Figure 3.

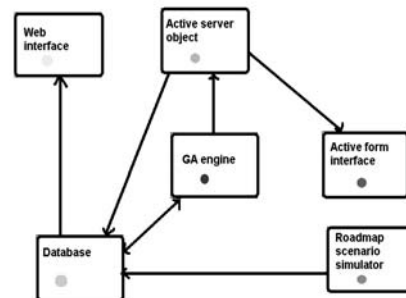


Fig. 3 Organization of software components and data flow of prototype

We provide below (in Figure 4 and Figure 5) some screenshots of the prototype for clarity. The implementation is completely web based and it uses a simulation of Dhaka's unpredictable traffic condition to show its applicability.



Fig. 4 The page that gathers user inputs and preferences.

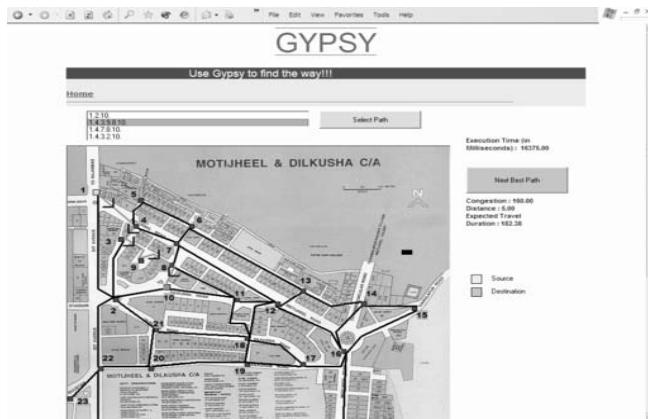


Fig. 5 The final output screen highlighting the optimal path with other useful information.

9. CONCLUSION AND FUTURE WORK

This paper presents a novel genetic algorithm approach to solve a variation of TSP with dynamic edge weights. We also related the usefulness and performance of the approach in the field of Telematics using our developed prototype, which works on a simulated traffic scenario of Dhaka. The algorithm analysis findings and the performance comparison of the approach confirmed its efficiency and applicability in the problem domain.

The method that we have used can be modified for quicker solution convergence outlined in [15]. We can also use operator selection and application probabilities for each operator as given in [9, 10, 15]. We can construct partial trees that can be used to create partially valid chromosomes. These chromosomes can be placed with existing population.

After altering with genetic operators they have better chances to eventually end up as a potential solution [8, 12].

The current web interface of the prototype can be customized for wireless application protocol (WAP) friendly interface using wireless markup language (WML). This will allow the users to obtain the service with more accurate data using their cell phones and personal digital assistants (PDA's).

REFERENCES

- [1] Cormen, T.H., C.E. Leiserson, and R.L. Rivest, *Introduc. to algorithms*, MIT Press, 1990
- [2] Dijkstra, E.W., "A note on two problems in connections with graphs", *Numerical Math. 1*, 269-271, 1959
- [3] Fredman, M.L., and R.E. Tarjan, "Fibonacci heaps and their uses in improved network algorithms", *Journal of the ACM*, vol. 34, no. 3 (July), pp. 596-615, 1987
- [4] Moriya, E., and K. Tsugane, "Optimally fast shortest path algorithms for some classes of graphs", *International Journal on Computer Math.*, vol. 70, pp. 297-317, 1998
- [5] Thorup, M., "Undirected single source shortest paths with positive integer weights in linear time", *Journal of the ACM*, vol. 46, no. 3 (May), 362-394, 1999
- [6] Valenzuela, C.L., "Evolutionary divide and conquer: a novel genetic approach to the TSP", *Unpub. Thesis*, 1995
- [7] Baker, J.E., "Reducing bias and inefficiency in the selection algorithm," In *Genetic Algorithms and Their Applications: Proceedings of the 2nd International Conference*, ed. J. J. Grefenstette, pp. 14-21, LEA, Cambridge, MA, July 1987
- [8] Bethke, A.D., "Genetic algorithms as function optimizers", *Ph.D. Thesis*, Univ. of Mich., 1981
- [9] Brindle, A. "Genetic algorithms for function optimization", *Ph.D. Thesis*, Comp. Sc. Dept., Univ. of Alberta, 1981
- [10] DeJong, K.A. "Analysis of the behavior of a class of genetic adaptive systems", *Ph.D. Thesis*, Comp. & Comm. Sc., Univ. of Michigan, 1975
- [11] DeJong, K.A. "Adaptive system design: a genetic approach," *IEEE Trans. Syst., Man, & Cyber.*, vol. 10, no. 9, pp. 566-574, 1980
- [12] Frantz, D.R. "Non-linearities in genetic adaptive search", *Ph.D. Thesis*, Comp. & Comm. Sc., Mich. Univ., 1972
- [13] Holland, J.H., *Adaptation in natural and artificial systems*, Univ. Michigan Press, Ann Arbor, 1975
- [14] Hollstien, R.B., "Artificial genetic adaptation in computer control systems", *Ph.D. Thesis*, Comp. & Comm. Sc., Univ. of Michigan, 1971
- [15] Smith, S.F., "Flexible learning of problem solving heuristics through adaptive search," *Proceedings of 8th International Conference on Artificial Intelligence*