OPTIMIZING E-RETAIL DELIVERY USING GENETIC ALGORITHMS

Annapurna P. Patil¹, Aahan Singh², Abdul Rabbani Shah³, Akhil Raj Azhikodan⁴ and Archit Bhatnagar⁵

¹Professor, Department of Computer Science and Engineering
M S Ramaiah Institute of Technology, Bangalore, Karnataka, INDIA – 54
{E-mail : ¹annapurnap2@msrit.edu, ²aahansingh95@gmail.com,
³rabbanishah@gmail.com, ⁴akhil95raj@gmail.com, ⁵architb361@gmail.com}

Abstract

Genetic Algorithms are used to solve the polynomial combinatorial optimization problems. Unlike Greedy approaches to a problem, this does not perform an exhaustive search. Instead, genetic algorithms find optimal objects(solutions) at each iteration. Furthermore, the set of feasible solutions are reduced at every level until a global optimum is found. One such application for this is the Travelling Salesman Problem. A specific case would be to solve the delivery issues that E-Retailers have in metropolitan cities. Their logistics department has to find the best delivery circuit. The objective of this work is to develop a system that uses genetic algorithm to optimize the delivery service of an e-retailer and analyze the effects of using different strategies of evolutionary algorithms.

Key Words : E-Retail delivery, Genetic Algorithm, Travelling salesman problem.

1. Introduction

E Retailers is the biggest boom in the market now. The online retailers have revolutionized the way people shop. With more and more people choosing online purchasing to traditional store front purchasing, the stress on the logistics has skyrocketed. Each product has to be delivered to the hands of the users who are scattered all over the city in the shortest time. Being such a dynamic field with heavy competition, the retailers have to make sure the delivery system is never crippled and the user satisfaction is met.

Optimizations of delivery routes have multiple parameters. Each of the parameters have their own way of affecting the optimality of the route. Some of the parameters are time, distance, traffic, cost of the product, weight of the product, fragility of the product, delay in delivery, number of delivery boys and time of delivery. Manual optimization of all the parameters is arduous. Since the delivery locations vary, the routes have to be recalculated daily.

The aim of the work here is to develop a system that utilizes genetic algorithm to optimize the delivery service of an e-retailer as well as analyze the effects of using different strategies of evolutionary algorithms. It would optimize the daily fuel cost, delivery distance, time, order delay, maximize daily collections of the delivery service and number of delivery boys. The fitness function of the problem would be determined from the above mentioned factors.

2. Literature Review

The retail logistics domain is a very dynamic field which is stressed due to intense competition. The primary concerns of the logistics department are the number of drivers on the road, number of deliveries per delivery boy and the order of deliveries. Route optimization is the most important step for the logistic department.

Route optimization in the retail logistic realm can be represented as a TSP (Travelling Salesman Problem) problem. If the number of traveling salesman (delivery boys) is to be optimized, it is a MTSP (Multiple Travelling Salesman Problem).

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city, this is called the TSP. TSP in theory of computational complexity is said to be NP-Complete. TSP in problems of combinatorial optimization is said to be NP-Hard.

The generalized form of the traveling salesman problem is MTSP. In MTSP there can be M number of traveling salesmen. MTSP is also classified as NP hard optimization problem. There is no definite way to solve MTSP. We know that genetic algorithms are the best solutions to combinatorial optimization problems. Here we try a genetic algorithm based solution to the problem of delivery optimization.

In genetic algorithm, the term evolution, in a naïve sense, would refer to the process that organisms go through to become better and better in a habitat [4]. That is, according to Charles Darwin, the most important process governing the changes happening in the biosphere.

Darwin also mentioned another term called “survival of the fittest” which describes how certain individual is “surviving” or is more probable to survive in an environment if it is good at adapting to the changes demanded by that environment. In other words, an individual is more likely to die or be “wiped out” if it is not able to adapt to the environment as needed. [3] Thus it describes how the population as a whole is, in a time frame, improving by inheriting genes from the stronger species, and by having the least adapted individuals wiped out from the gene pool. In other words, evolution causes a rise in the fitness of a population. In mathematical terms, we would have a quality function as an abstract fitness measure that has to be maximized in each generation. Based on such a measure, better candidates can be chosen to breed/combine to make an even better next generation.

Mathematical functions are used to determine the fittest individuals from a gene pool. Recurring functions combine two or more of these “fitter” individuals to make a new set of evolved species. The calculations are repeated until a candidate with acceptable fitness is met. These types of algorithms are called Evolutionary Algorithms.

The evolutionary algorithms have no assumptions. As generations of the algorithm runs, the knowledge is built by selecting the best of the candidates in the gene pool. Hence evolutionary algorithms can solve majority of the optimization problem.

Evolutionary algorithms are used in real world problems mostly for Design optimization. Engineering designs are perfected by running GA (genetic algorithms) to return the perfect combination of structure and building materials. The algorithms are used in the field of molecular chemistry to identify possible molecular structures. Additionally it is used in the field of Electronics for VLSI design.

Evolutionary algorithms face a big drawback due to high computational complexity and non-deterministic termination stage.

Numerous evolutionary algorithms exist such as Evolutionary Programming, Evolution Strategies, Classifier Systems and Genetic Programming [9][10].

3. Proposed Idea

A. Objective

- 1) To translate the delivery problem as a variation of travelling salesman problem with multiple salesmen traversing the same graph (city).
- 2) To compute the most efficient path for each of the delivery person considering the factors like daily fuel cost, delivery distance, time and order delay.
- 3) To maximize daily collections of the delivery service.

- 4) To define the fitness function considering the weights of each of the factors affecting the delivery costs.
- 5) To build a genetic algorithm optimizing the fitness function defined.
- 6) To analyze the effects of application of different strategies like plus-one and comma strategy in each generation of the population.
- 7) To analyze the computational time required, completeness of the algorithm and correctness of the algorithm.

B. Algorithms

Logic - The next generation is the result of mutation of the fit members of the generation

```

t=0;
(t generation counter)
P=randomizePopulation(t);
(randomly generate the first generation)
evalFitness(P);
(find the fitness of each org in the population)
while (goal not met)
t=t+1;
(next generation)
AlphaParents=selectParents (P, t-1);
(find the best genes)
Genes=combineGenes( AlphaParents);
(combine genes)
P'=Mutate(Genes);
(bring in mutation)
goal=evaluate(P');
P=survivors(P');
End

```

Alternative Logic [3] - The population grows by replacing the most unfit members with new offspring

Create Population

While True

Choose Parents

Combine Parents (Recombination step)

Mutate Offspring (Mutation – adding better genes)

Rate Offspring (Measuring the Fitness)

Add Offspring to Population

Remove Unfit Individual (Natural Selection to remove weakest)

If Termination Criterion is met: End Loop

Next

Deliveries of goods determine several factors like profit,

workforce efficiency, warehouse utilization, customer satisfaction. The more systematic the delivery is, the more efficient and profitable the e-retailers are. With the constant reduction of product costs to keep the rival retailers at bay, delivery optimization is one of the ways the retailers can try generating profits from sales.

The logistical department of the online retailers has to solve MTSP problems every day to optimize their delivery. Given 'M' delivery boys the routes for each boy is to be optimized. This optimization is NP Hard and there is no solution for it. The closest approximation we can have to the minimum tour for each delivery boy is by running an approximation algorithm on it. Genetic algorithms work amazingly well for MTSP problems.

4. Design

Fig.1 : Flow of data in the system

A. Function- for data retrieval:

For each Destination

Get address of Destination;
 Translate address to Latitude/Longitude(lat/long)
 Append lat/long to destination array

End for

B. Constructing distance matrix and time matrix;

For each X in destination array

For each Y in destination array that is not X
 Find distance and time for X->Y
 End for

End for

C. Calculating Fitness for the offsprings;

Fitness (off spring)

For each x destination in offspring

Distance = distance (x , x+1)
 Time = time (x , x+1)

Fitness = Distance * Distancefactor

Fitness = Fitness + Time * Timefactor

Return Fitness

D. MTSP(parent)

// creates next generation

Generation = new array();

For x in 1 to GenerationSize

Temp = new Child (parent)

Temp = mutate(Temp)

Generation.append(Temp)

FitnessArray = new array()

For each child in Generation

FitnessArray.append(Fitness(child))

alphaChild = finds best fit child(FitnessArray)

if(fitness(alphaChild) == fitness(parent))

return parent

return MTSP(alphaChild)

The system takes in raw delivery destinations from a text file in line separated format. It then adds into memory each delivery location. Each location is then queried to google maps API to retrieve latitudinal, longitudinal information. A matrix called Distance Matrix is formed by the system. The matrix maps the distance from one point to another. The distance between each destination is got from the google maps API. The expected time for making the journey from point A to point B is also retrieved and stored in the time matrix. The time and the distance matrix will act as the parameters in our fitness function.

Fig. 2 : Fitness growth

In fig. 2, the growth in the fitness values of the genes over a hundred generations is shown. MTSP algorithm in our system generates new generation of off springs until there is no significant change in the fitness of the alpha parents. The system takes in a random first chromosome and runs the recursive function. Each iteration the function generates a generation of off springs, then calculates the

fitness function and selects the best of spring as the alpha parent. The growth in the fitness shows that the on progression of the algorithm the routes get optimized.

```
Alpha Parent : [[1, 2, 3], [4], [5]]
Child Generation : [[[3, 1, 2], [4], [5]], [[2, 3, 1], [4], [5]], [[3, 1, 2], [4], [5]], [[3, 2, 1], [4], [5]], [[3, 1, 2], [4], [5]], [[3, 2, 1], [4], [5]], [[1, 2, 3], [4], [5]], [[1, 3, 2], [4], [5]], [[2, 3, 1], [4], [5]], [[3, 2, 1], [4], [5]]]
fitness score : [189975, 204496, 189975, 208864, 208855, 198941, 198941, 186236, 218149, 227124, 189975, 214419, 208855, 198941, 204505, 195211, 198941, 189975, 189975, 198941, 218636, 244108, 218636, 229548, 218636, 229548, 229548, 244108, 244108, 229548]
Best Child : [[0, 1, 3, 2, 0], [0, 4, 0], [0, 5, 0]] Cost : 186236
////////////////////////////////////Alpha
Parent : [[1, 3, 2], [4], [5]]
Child Generation : [[[2, 3, 1], [4], [5]], [[1, 2, 3], [4], [5]], [[3, 2, 1], [4], [5]], [[2, 3, 1], [4], [5]], [[3, 1, 2], [4], [5]], [[2, 3, 1], [4], [5]], [[2, 3, 1], [4], [5]], [[3, 2, 1], [4], [5]], [[3, 2, 1], [4], [5]], [[3, 2, 1], [4], [5]], [[3, 2, 1], [4], [5]]]
fitness score : [189975, 204496, 189975, 208864, 208855, 198941, 198941, 185236, 218149, 227124, 189975, 214419, 208855, 198941, 204505, 195211, 198941, 189975, 189975, 198941, 218636, 244108, 218636, 229548, 218636, 229548, 229548, 244108, 244108, 229548, 244108, 229548, 229548, 244108, 244108, 229548, 229548]
Best Child : [[0, 3, 2, 1, 0], [0, 4, 0], [0, 5, 0]] Cost : 185236
```

Fig. 3 : Running of the system on 10 destinations

In figure 3, the system displays each generation and the fitness values determined for each offspring. The alpha parent acts as the parent for the next generation. The result is returned as a JSON formatted list this system can be easily hosted at a server and the returned JSON format data can be easily be parsed to make applications for any MTSP problem. The system built is a general purpose solver for MTSP given the cities. The tour is returned as a list of lists, where the number of lists indicated the optimal number of delivery boys and the list inside the list indicates the cities the delivery boy must go to in the increasing order.

5. Results

The parameters that affect the execution of the function are generation size, distance factor and time factor. The generation size indicates how many off springs will be generated at each iteration of the algorithm. The distance factor is the factor by which the distance matrix contributes to the fitness function. The time factor is the factor by which the time matrix contributes to the fitness function. These parameters need to be optimized to get the best results. By trial and error, we determined the generation size is the best set to a value approximately equal to $N < t < N^2$ where N is the number of delivery destinations and t is the generation size.

The factor by which the distance effects the fitness function is related to the time factor. For certain applications or case scenarios where time is more important than the distance traveled the time factor has to be set higher. The applications where cost of travel is significantly higher than the cost of time the distance factor is to be set higher. In our retail logistic delivery problem, the distance traveled is of a higher concern than

the time. There are multiple delivery boys so the time can be of lesser significance because as the number of delivery boys increases the cumulative available man power and time taken is optimized. After numerous trials we found out that the best results were returned by the system when the distance factor was twice the time factor. So we can say that for this problem statement the distance is twice more important to the fitness function than the time taken.

6. Comparison to Greedy Algorithms

TSP can be solved using greedy approach. Given that TSP can be solved, then a two-layer greedy algorithm can solve a Multiple-TSP problem. There is a significant difference in the solutions given by the two approaches we discuss here. Greedy approaches as well as the genetic algorithm do not guarantee the convergence of the solution to the global optima.

Given the fact that MTSP is a combinatorial optimization problem, the permutations of the possible solution are unimaginably high. A brute force solution is the only way to ensure convergence to global optima. The best we can do to solve such problems is to divide the set of solutions and run the search only of the most probable subsets of the solution where the optima might lie.

Greedy approach assumes that the choosing the most optimal next step leads incremental growth to the global optima. The solution might be true in some cases but the in the most cases in real life the solutions are never that simple. Greedy approach is easier to understand and code, but the convergence would be to a local optimum but not the global optima.

MTSP being a problem with applications to fields with such high level of competitions requires the algorithms to converge at the global optima for the best possible solution. The only way to solve such a problem and converge to the global optima is by using evolutionary algorithm. The assumptions of the genetic algorithm are that the values of fitness are continuous and the global optima is the crest of the fitness graph. There may be multiple crests which represent the other local optima. The algorithm runs but randomizing the generation with offspring from all parts of the set of solutions. The alpha parents are chosen, which are higher up the graph than the other offspring. The alpha parents are mutated to form the next generation [6]. The mutations bring in small changes to the chromosome. The small changes scatter the next generation near the alpha parent so half of the next generation would do better than the parent. Choose the next alpha parent by choosing the best offspring who is higher up the graph than the rest of the siblings. This cycle continues till we get the best alpha parent.

It is noticed that there is no way the next generation can be more optimal than the most optimal alpha parent. This can be taken as a point of exit for the algorithm. When there is

no increase in the fitness in the subsequent generations then the alpha parent is the optima.

Greedy algorithm would always converge to the same optima considering the way makes decisions at each steps. Genetic algorithms will not converge to the same solution every time as there can always be a global optimum that the genetic algorithm did not find. This global optimum can be found when the algorithm is run again on the same solution set.

7. Conclusion

MTSP is best solved with genetic algorithm. In comparison to the greedy algorithms and dynamic programming, genetic algorithms give a better tradeoff between time taken and memory used and provide a better accuracy. In figure 4, the run time for the different approaches are compared. MTSP with smaller number of cities are solved faster by greedy and dynamic methods. MTSP with higher number of cities is better solved using genetic algorithms.

Fig. 4 : Comparison of different MTSP solutions

Our proposed method is using asexual reproduction with selection and mutation which takes up lesser memory and time because only the single fittest chromosome is selected and mutated randomly discarding all the remaining chromosomes in each generation. The number of iterations for the algorithm to reach the solution is exponentially proportional to the number of cities and the number of tours (M) being formed. The MTSP problem was solved with real live data taken from the internet. Genetic optimization shows how dynamic and efficient it is for problems that have non operable range of combinations.

References

- [1] Mohammad Sedighpour, Majid Yousefi Khoshbakht, Narges Mahmoodi Darani, An Effective Genetic Algorithm for Solving the Multiple Traveling Salesman Problem, in Journal of Optimization in Industrial Engineering 8 (2011)
- [2] Felix Streichert, Introduction to Evolutionary Algorithms [Online]. Available: http://www.ra.cs.unituebingen.de/mitarb/streiche/publications/Introduction_to_Evolutionary_Algorithms.pdf
- [3] Kumara Sastry, David Goldberg, Genetic Algorithms [Online]. Available: http://webpace.ulbsibiu.ro/adrian.florea/html/Planificari/EvolutionaryComputing/Course_4/pdf/introsch4.pdf
- [4] Zakir H. Ahmed. Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator, in International Journal of Biometrics & Bioinformatics (IJBB) Volume (3): Issue (6)
- [5] Elaine Rich, Kevin Knight and Shivashankar B Nair, Artificial Intelligence, 3rd Ed, India, Tata McGraw-Hill, 2014
- [6] Jeff Heaton, Artificial Intelligence for Humans Volume 2: Nature Inspired Algorithms, 1st Ed, Heaton Research, 2014