International Core Journal of Engineering Volume 7 Issue 1, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0039

Genetic Algorithm on MTSP within Polar Coordinates and Time Consumption with Respect to Various Variables

Haocheng Ren[1], Haixiang Huang[2], Zhangxian Song[3], Hengfei Liu[4]

[1]College of LSA, University of Michigan Ann Arbor, 48109, U.S;

[2]College of Engineering, South China Agriculture University, Guangzhou, 510642, China;

[3]College of Software Engineering, Shandong University, 250100, China;

[4]College of Engineering, University of Sheffield, S10 2TN, U.K.

Abstract

Multi-Agent Travelling Salesman Problem also known as MTSP, which consists of at least two salesmen and a specific number of cities (city # > agent #), extends the issue of classical travelling salesman problem (TSP). Under the conditions of a set of cities, a starting point, where all salesman or agents are, and a cost measure, the objective is to determine combinatorial routes for all agents, minimizing the total cost of M routes. A general genetic algorithm (GA) based in the polar coordinate system is introduced to solve MTSP in this paper. The main structure focuses on making sub problems. The paper will address how GA can solve MTSP by dividing it into many TSPs, and how solution time scales with various variables such as population size and generation.

Keywords

Genetic Algorithm, Polar Coordinates System, Multi-Agent Travelling Salesman Problem, Variables Control, Solution Time.

1. Introduction

Classic TSP is a mathematical question on route finding in the way of combinatorial optimization [1]. It can be solved by many algorithms such as simple Brute Force, Dynamic Programming and Simulated Annealing [2][3][4]. Multi-Agent Travelling Salesman Problem (MTSP) is an evolution of TSP with more than one agent to come up with a minimized combined path, which follows the same constraint as TSP has that each city is allowed to be visited once by a single agent [5].

The MTSP fits the real-life applications much more appropriately than TSP [6]. One of them is that cashiers from the same accounting firm have the responsibility to visit the companies in other cities for a business trip to do some financial auditing. The MTSP here works out a plan of efficiency to help the accounting firm to save time plus budgets by reducing the total distance of the whole trip.

In this paper, the solution of a general MTSP is based on Genetic Algorithm (GA) inspired from Darwin's theory that follows the pattern of natural selection [7]. Although the strategic plan has received extensive attention, the research on it is still limited. Our purpose is to review the existing literature on MTSP, solution programming, and data and graph analysis.

The rest of paper put emphasis on three parts. We propose this algorithm under the Polar Coordinate System. More details about how GA selects parents, processes the mutation and generates a new population would be elaborated in section III. Before that, section II will introduce the way cities fit into a polar coordinate system and how agents pick their cities to visit. After being done with the implementation of GA, scientific experiment – control variable – would be performed to see the pattern of our solution time corresponding to the population size and generation (iteration).

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0039

2. Graph Model

A center city is the first key element as our starting point. The way to choose this point follows the properties of Geometry. Disregarding the outliers, we pick four furthermost points: left, right, bottom and top. As shown in the Fig. 1, four lines passing those points as a rectangle. A red dot center of this shape can be found via the intersection of two diagonals. We can decide our starting city by finding the closest point to the red dot.

The world map can represent all locations in terms of latitude and longitude or more easily we are able to simplify the geographical representation into the Cartesian coordinate system through x and y. However, the drawback of this setting is that it easily limits the number of agents up to four since there are four quadrants. A better approach could be using the polar coordinate system. Let's take Fig. 2 as a huge pizza shared by 8 people (agents). We cut this cake evenly for 8 pieces and distribute them to people. Each person is "responsible" for eating her own piece with some pepperoni on it. By this means, no matter how many agents we have, just need to make sure to find a center, set up a polar coordinate system based on this origin and evenly arrange the whole map.

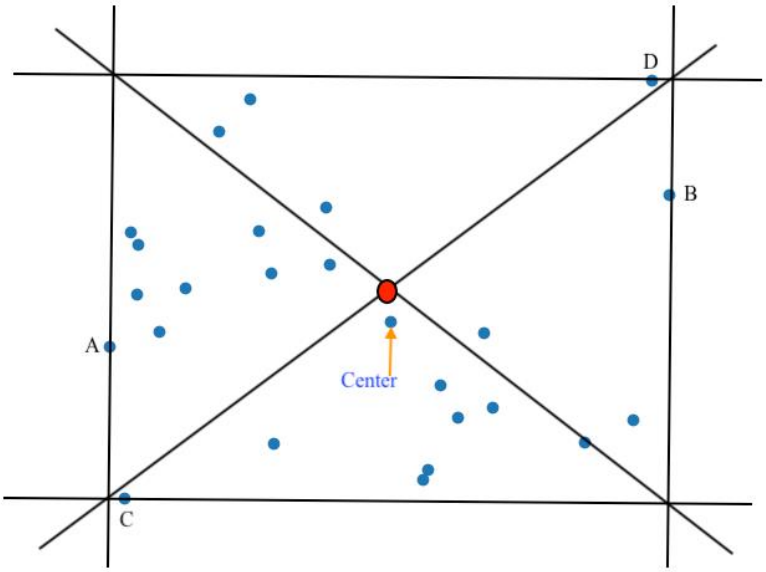

Figure 1. Points randomly generated

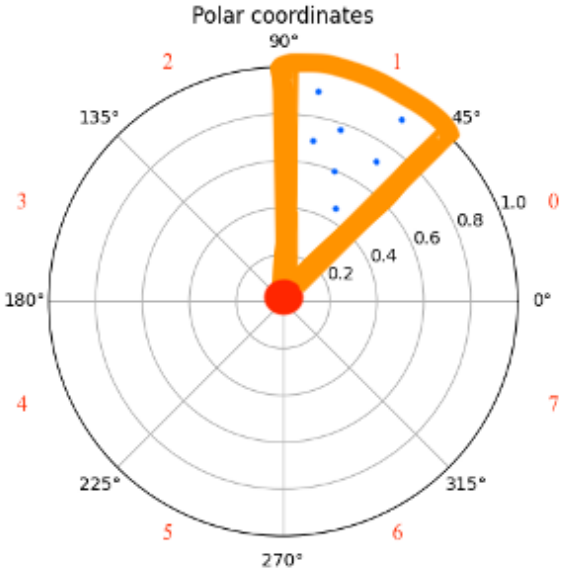

Figure 2. The Polar Coordinate System

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0039

In the Cartesian system, we can also divide the whole graph like the way we do in polar, but it is not easy to know which city belongs to which agent's task via (x, y) comparing to the polar system whose coordinates consist of (r, θ). Through the formula $SECTION\ DEGREE = \frac{360}{\#\ of\ agents}$, we get how much degree each agent is staying with. θ of the city tells us which agent would come to visit.

The distance between cities is given as:

$$Distance = \sqrt{r_1^2 + r_2^2 - 2r_1r_2\cos(\theta_1 - \theta_2)}$$

Since the paper is dealing with general MTSP, the iteration is applied here to generate random points for agents one by one until we have enough cities. While doing so, we make it certain that each agent has roughly the same amount of cities to visit.

3. Algorithm Implementation and Presentation

3.1 GA demonstration and some terminologies

As mentioned in intro, GA is an evolution on Darwin's theory. The process of GA keeps track of genetics pattern. The chromosome is the route or path, in which the gene is the city. Due to one of our constraints, the route here always starts from the origin (0, 0). All of those chromosomes, which would be ranked by fitness score, consist of a huge population also known as generation. Each chromosome has its fitness score. The higher score it has, the larger possibilities it would be selected into the mating pool to reproduce the next generation, which follows the rules of natural selection in genetics. Mutation does not occur that often, but still keeps little odds to appear. This variation could be bad that creates a route with worse total cost or good that leads to a more favorable offspring than normal breeding. GA puts all these pieces together and repeat the whole process creating a new generation over and over again until the top chromosome in the population converges. To solve MTSP, we use this process to conquer single TSP first and eventually combine all the results together as our final solution.

3.2 Setting up population and fitness

Since it is not possible for the computer to know the best path in the first time, randomness coms to play a role. Algorithm generates routes by randomly arranging cities in each section in Fig. 2 for instance. We can set the parameter for population size in advance which should be less than cities# factorial due to the permutation. Then, the fitness calculation contributes to ranking routes in the current generation. In general, the algorithm prefers the routes with less total cost. Therefore, we take the inverse of this total distance cost as the fitness score of each path. The less cost the path has, the higher fitness score it would achieve.

3.3 Survival of the fittest

In this stage, GA is going to pick the fine chromosomes and construct a mating pool for the next generation. Complying with Darwin's survival of the fittest, a number of top parents are selected first who are also called elites here [8]. For the rest of spots in the pool, we employ the approach "Roulette wheel selection" [9]. The first step is to calculate the percentage of each route's fitness score relative to the sum of the total score. Then, randomly generate a number within range from 0 to 100 and compare it with all routes' fitness percentage via an ascending ranking order. The principle here is that, for example, one of the fitness percentages is 98% which means there would be 98% chance to pick a number from 0 to 100 that is less than or equal to this percentage. The higher fitness a path has, the higher chance it will be chosen. With all procedures above, the mating pool would mainly consist of parents or chromosomes with better characteristics to carry on.

3.4 Reproduction and Mutation

In this stage, offspring are created from the mating pool. The common approach we take is called crossover. We pick two parents from the pool first.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0039

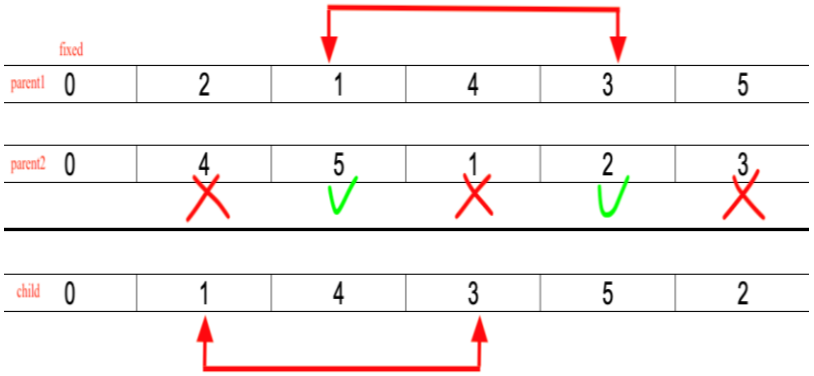

Figure 3. Crossover Reproduction

In Fig. 3, a sub portion from parent1 is randomly selected to descend for the child. If there are still some spots left in the child, then we go to parent2 to pick the rest gene that does not appear in the previous sub portion.

Mutation occurs during the stage of breeding, which assists with keeping away from the local converging through the way of providing new path. By doing so, it allows us to investigate other possibilities of solution. We cannot set the mutation rate too high; otherwise, the randomness would avoid a global convergence. Here in programming our mutation rate is 0.01. Since we are not allowed to drop any city in the routes, the swap between random two cities will be taken when there is a chance to mutate.

3.5 Iteration or Repeating the generation

This is the last stage for solving one agent's route-finding problem. We put all pieces mentioned above together and keep iteration. The new generation is created over and over again as we want. As we have many generations like 500 or 1000, there will be a good path with relatively less cost in the distance. To tackle MTSP, we just need a for loop to solve each individual agent's TSP by the procedures above. Finally, the complete route should look Fig 4. below.

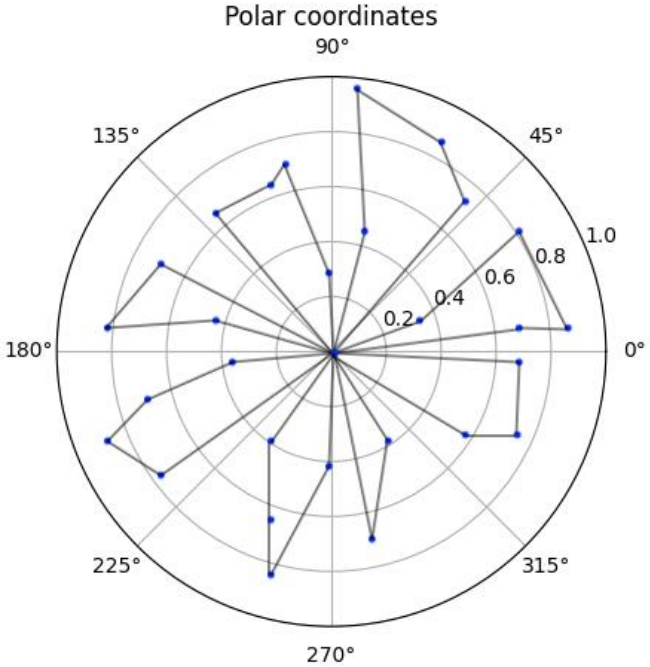

Figure 4. Combined routes for MTSP

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0039

4. Experiment

Variable control would be conducted here to how solution time scales with various variables (population and generation) when the mutation rate and elite size are always fixed, 0.01 and 5 respectively.

4.1 Population:

The numbers of population size we are going to use are 10, 15… 40 while the agent number is 6, the generation size is 100, and the city number is 40. As shown in Fig. 5, it has the trend of polynomial time.

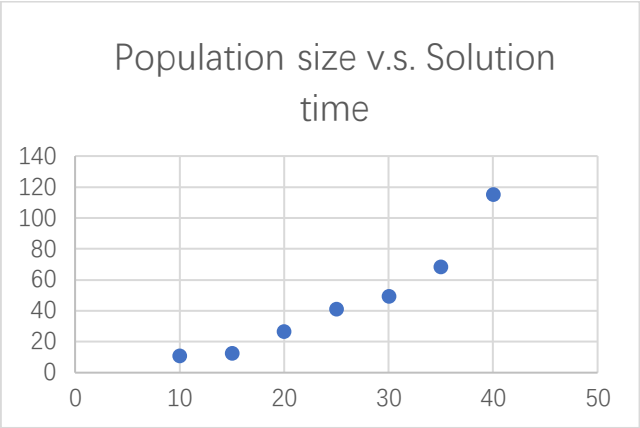

Figure 5. Population size vs Solution time

4.2 Generation:

The numbers would be 100, 200, 300, 400, 500, 600, 700, and 1000, while the agent number is 6, the population size is 15, and the city number is 40. The graph of Fig. 6 is likely to be linear.

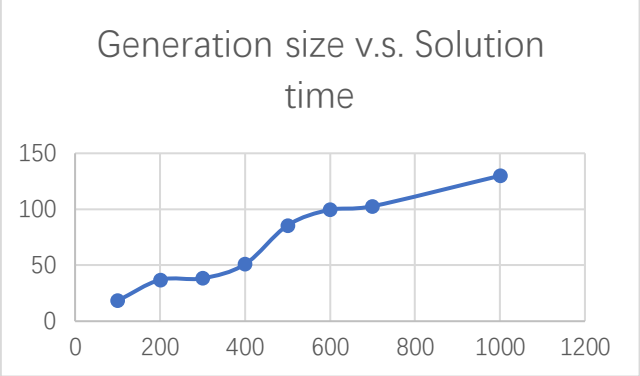

Figure 6. Generation size vs Solution time

5. Conclusion

Our algorithm divides a big problem into several sub problems and solve each one in the way of TSP. An optimal solution shall never be guarantee by Genetic Algorithm, but instead it is a good approach at finding the path in a huge search space, i.e. huge amounts of cities. When we increase the number of cities, the solution time increase too, but it is always relatively good. Instead, the future research should still focus on the situation when there is no city in a specific range of an agent.

References

[1] Durbin, R., Willshaw, D. (1987) An analogue approach to the travelling salesman problem using an elastic net method. Nature 326, 689–691. https://doi.org/10.1038/326689a0.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202101_7(1).0039

[2] 125 Summer 2020 (2020) 13.Case Study: Solving the Traveling Salesman Problem. Available at: https:// www2.cs.sfu.ca/ CourseCentral/125/tjd/tsp_example.html (Accessed:).

[3] ScienceDirect (2011) A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem. Available at: https://doi.org/10.1016/0377-2217(94)00299-1 (Accessed:).

[4] ScienceDirect (2011) Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. Available at: https://doi.org/10.1016/j.asoc.2011.01.039 (Accessed:).

[5] C. Yang and K. Y. Szeto, (2019) "Solving the Traveling Salesman Problem with a Multi-Agent System," IEEE Congress on Evolutionary Computation (CEC), Wellington, New Zealand, 2019, pp. 158-165, doi: 10.1109/ CEC.2019.8789895.

[6] He J. (2014) Solving the Multiobjective Multiple Traveling Salesmen Problem Using Membrane Algorithm. In: Pan L., Păun G., Pérez-Jiménez M.J., Song T. (eds) Bio-Inspired Computing - Theories and Applications. Communications in Computer and Information Science, vol 472. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-45049-9_27.

[7] O. Matei and P. Pop, (2010) "An efficient genetic algorithm for solving the generalized traveling salesman problem," Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, 2010, pp. 87-92, doi: 10.1109/ICCP.2010.5606458.

[8] S. A. Kazarlis, A. G. Bakirtzis and V. Petridis, (1996) "A genetic algorithm solution to the unit commitment problem," in IEEE Transactions on Power Systems, vol. 11, no. 1, pp. 83-92, Feb. 1996, doi: 10.1109/ 59.485989.

[9] Lipowski, Adam, and Dorota Lipowska. (2012) "Roulette-wheel selection via stochastic acceptance." Physica A: Statistical Mechanics and its Applications 391.6: 2193-2196.