International Journal of Innovation Scientific Research and Review
Vol. 05, Issue, 05, pp.4530-4534, May 2023
Available online at http://www.journalijisr.com
SJIF Impact Factor 2023: 6.599

Research Article

ISSN: 2582-6131

COMPARISON OF THE USE OF DYNAMIC PROGRAMMING ALGORITHM AND GENETIC ALGORITHM IN SOLVING THE TRAVELLING SALESMAN PROBLEM

*** Charisma Tubagus Setyobudhi**

Department of Computer Science, Faculty of Technology and Engineering, DiponegoroUniversity,Jl. Prof. Soedharto SH, Tembalang, Semarang-Indonesia.

Received 19th March 2023; Accepted 20th April 2023; Published online 31st May 2023

ABSTRACT

Artificial Intelligence (AI) is a topic of great interest in research. One of the well-known problems is the TSP problem. The Traveling Salesman Problem (TSP) is a classic problem that is quite difficult to find a solution to. TSP problems are included in the NP Hard Problem category. Currently, several solution approaches have been found in the TSP. The well-known algorithms used for the solution approach are DP (Dynamic Programming) and GA (Genetic Algorithm). In this paper, the author wants to discuss the performance differences between the two algorithms. In the end, the author argues which algorithm is suitable for the TSP problem.

Keywords: AI, TSP, Dynamic Programming, Genetic Algorithm.

INTRODUCTION

The travelling salesman problem (TSP) is a classic problem for solving graph problems. The TSP involves a salesman who has to visit each city on the graph once before returning to the starting point, where the travel costs incurred during the trip are the minimum [1]. The TSP problem is widely used in real-world cases, for example, in the vehicle routing problem and railway travelling salesman problem[2]. This TSP problem was found in 1930 by Irish Mathematician, William Rowan Hamilton and British Mathematician Thomas Kirkman. TSP has several application domains, including the planning, logistics, and manufacturing of microchips. Hence, finding a correct and efficient method for TSP is very important for researchers.

RELATED WORKS

Colony Optimization [3] algorithm, Particle Swarm Optimization [4], Evolutionary Algorithm [5], Agent Based Evolutionary Search [6], VNS Algorithm [7], Genetic Algorithm [8,17], Novel Hybrid Penguins Search[9], Bee Colony [10,15], Genetic Annealing[11], Memetic Hunting Search [12], Harmony Search [13], Simulated Annealing[14], Reinforcement Learning on Genetic Algorithms [15] TSP is a complex problem that is categorised as an NP Hard problem. The NP Hard problem is a problem in which the algorithm to solve it can be derived to become another algorithm that can solve the NP polynomial. In addition to TSP, there are several other problems that can be categorised as NP Hard, namely the subset sum, maximum clique problem, minute colour problem, and longest path problem.

THEORETICAL BACKGROUND

Dynamic Programming

Dynamic Programming is an optimization technique in programming when encountering a problem that can be solved recursively.

In Dynamic Programming, when one encounters a sub-problem that can be solved repeatedly, Dynamic Programming can be used. Every time we solve a sub-problem in a bigger problem, we can store it in an array so that when we need a solution to a sub-problem that has been solved before we can retrieve the solution constantly, O(1).There are two types of DP approach techniques in Dynamic Programming:

1. Top Down (Memoization)

Using this technique, we first depart from the main problem and then break it down into smaller sub-problems. Typically, a top-down approach uses recursive techniques to solve this problem.

2. Bottom Up (Tabulation)

If we use this technique, we start with small problems first and then accumulate them into larger ones. Typically, a bottom-up approach uses regular iteration techniques to solve this problem.

Genetic Algorithm

Artificial Intelligence is a field in computer science that studies how computers and machines act intelligently. There are many branches of artificial intelligence, one of which is genetic algorithms. A genetic Algorithm (GA) is an Artificial Intelligence method that is often used to solve optimisation problems. Optimisation problems aim to find optimal solutions from among other solutions. The goal of the genetic algorithm is to find a solution from several possible solutions generated through an iterative process [2]. The genetic algorithm is a metaheuristic algorithm used to solve complex problems [3]. In addition to genetic algorithms, optimisation problems can be solved using dynamic programming and machine learning methods. Genetic algorithms can generally be used because the method

***Corresponding Author: Charisma Tubagus Setyobudhi,**
Department of Computer Science, Faculty of Technology and Engineering, Diponegoro University,Jl. Prof. Soedharto SH, Tembalang, Semarang-Indonesia.

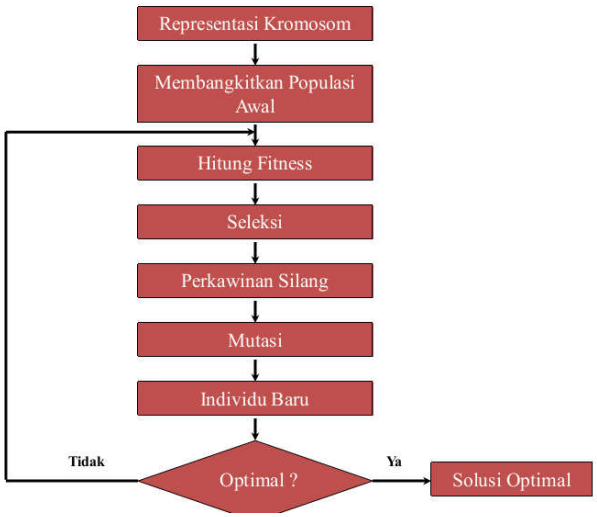

Fig 1. Flowchart of Genetic Algorithm

1. Chromosome Representation
   In this stage, the solution is expressed in terms of chromosomes. Chromosomes are the forerunners of the solution being sought. The form of chromosome representation can use binary numbers, decimal numbers or other numbers depending on the problem being faced MembangkitkanPopulasi Awal
2. Calculating the value of Fitness
   The fitness value is the value of an individual that is used to determine which individual is the strongest among other individuals. This fitness value generally uses a formula based on individual suitability to achieve the goals of the search process.
3. Selection
   During the selection stage, individuals with high fitness values were selected and pressed.
4. Crossover
   At this stage, crossover or crossbreeding is the stage in which two individuals are selected, and then the chromosomes are exchanged to produce new individuals to be included in the population.
5. Mutation
   Mutation is the stage where an individual's chromosome will be changed (mutated) to produce a new individual
6. New Individual
   New individuals that have been produced in the previous stages are then included in the population so that they will be processed further in the process of finding solutions

PROPOSED METHODS

In this study, we present two techniques for solving the TSP: Dynamic Programming and Genetic Algorithm. In each subsection, we elaborate on the methods and algorithms used in each implementation. Hence, we hope that the reader will understand the mechanics and logical processes implemented using both techniques.

Dynamic Programming

In the search for TSP solutions using Dynamic Programming techniques, several methods or functions are implemented, namely,

1. findSolution()

```
public int findSolution(){
startTime = getCurrentTimeMillis();
      solution = doTSP(1,0);
endTime = getCurrentTimeMillis();
elapsedTime = endTime - startTime;
      return solution;
}
```

2. doTSP()

```
public int doTSP(int mask, int pos){
if( mask == VISITED_ALL)
         return graph[pos][0];
if( DP[mask][pos] != -1){
         return DP[mask][pos];
      }

ans = Integer.MAX_VALUE;

for(int city = 0; city < N; city++){
         iteration++;
if( graph[pos][city] > 0 && ((mask & (1 << city)) == 0)){
costPath = graph[pos][city] + doTSP( mask | (1 << city), city);
ans = min(costPath, ans);
         }
      }
      return (DP[mask][pos] = ans);
   }
```

Genetic Algorithm

Genome

To represent the genome in the genetic algorithm, the authors used a class called SalesmanGenome. Inside SalesmanGenome, several methods and functions have been used in genetic algorithms. The first function is calculate Fitness()

```
public int calculateFitness(){
      fitness = 0;
currentCity = startingCity;
      for each (gene in genome) {
         fitness += travelPrices[currentCity][gene];
currentCity = gene;
      }
      fitness      +=      travelPrices[genome.get(numberOfCities-2)]
[startingCity];
      return fitness;
   }
```

The second function used is randomSalesman()

```
private List<Integer>randomSalesman(){
[      result = new ArrayList<Integer>();
for(int i=0; i<numberOfCities; i++) {
         if(i!=startingCity)
result.add(i);
      }
      shuffle(result);
      return result;
   }
```

And the last and not least important third function is compareTo()

```
@Override
   public int compareTo(Object o) {
      genome = (SalesmanGenome) o;
if(this.fitness>genome.getFitness())
         return 1;
      else if(this.fitness<genome.getFitness())
         return -1;
      else
         return 0;
   }
```

SalesmanTSP

SalesmanTSP is the main class in which a genetic algorithm is used to determine the optimal solution for the TSP. In this SalesmanTSP class there are several main functions, namely: create Generation()

```
public
List<SalesmanGenome>createGeneration(List<SalesmanGenome>
population){
      generation = new ArrayList<>();
currentGenerationSize = 0;
while(currentGenerationSize<generationSize){
         parents = pickNRandomElements(population,2);
         children = crossover(parents);
children.set(0, mutate(children.get(0)));
children.set(1, mutate(children.get(1)));
generation.addAll(children);
currentGenerationSize+=2;
      }
      return generation;
   }
```

1. crossOver()

```
public     List<SalesmanGenome>crossover(List<SalesmanGenome>
parents){
      // housekeeping
      random = new Random();
      breakpoint = random.nextInt(genomeSize);
      children = new ArrayList<>();

      // copy parental genomes - we copy so we wouldn't modify in
case they were
      // chosen to participate in crossover multiple times
      parent1Genome                    =                    new
ArrayList<>(parents.get(0).getGenome());
      parent2Genome                    =                    new
ArrayList<>(parents.get(1).getGenome());

      // creating child 1
for(int i = 0; i<breakpoint; i++){
newVal = parent2Genome.get(i);
         swap(parent1Genome,parent1Genome.indexOf(newVal),i);
      }
children.add(new
SalesmanGenome(parent1Genome,numberOfCities,travelPrices,start
ingCity));
      parent1Genome = parents.get(0).getGenome(); // reseting the
edited parent

      // creating child 2
for(int i = breakpoint; i<genomeSize; i++){
newVal = parent1Genome.get(i);
         swap(parent2Genome,parent2Genome.indexOf(newVal),i);
      }
children.add(new
SalesmanGenome(parent2Genome,numberOfCities,travelPrices,start
ingCity));
      return children;
   }
```

2. mutate()

```
public SalesmanGenomemutate(SalesmanGenome salesman){
      random = new Random();
      mutate = random.nextFloat();
      if(mutate<mutationRate) {
         genome = salesman.getGenome();
swap(genome,                    random.nextInt(genomeSize),
random.nextInt(genomeSize));
         return new SalesmanGenome(genome, numberOfCities,
travelPrices, startingCity);
      }
      return salesman;
   }
```

3. optimize()

```
public SalesmanGenomeoptimize(){
      population = initialPopulation();
globalBestGenome = population.get(0);

for(int i=0; i<maxIterations; i++){
         iteration++;
         selected = selection(population);
         population = createGeneration(selected);
globalBestGenome = min(population);

if(globalBestGenome.getFitness() <= targetFitness)
            break;
      }
      return globalBestGenome;
   }
```

EXPERIMENTAL RESULT

The experiment was conducted using four test case graphs. For each of these, we attempted to execute both algorithms to determine the TSP length. Four graphs are presented below.

Figure 2. Case 1 of TSP Problem

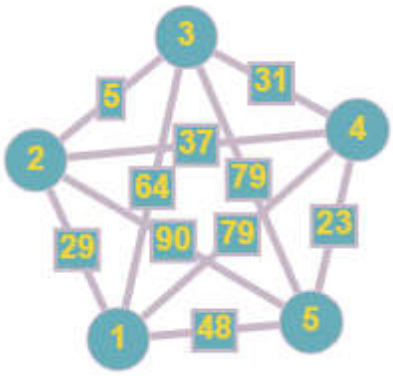

Figure 3. Case 2 of TSP Problem

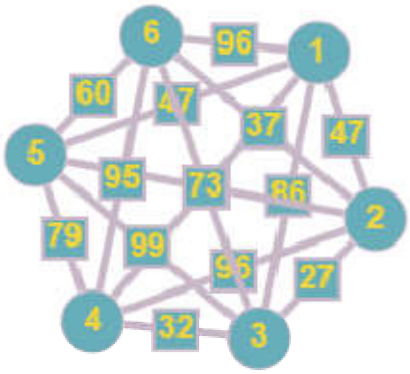

Figure 4. Case 3 of TSP Problem

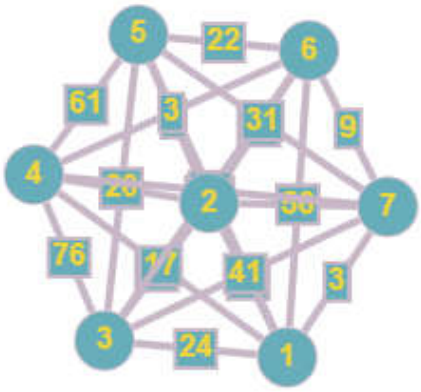

Figure 5. Case 4 of TSP Problem

The results of the experiments are listed in the following table.

Table 1. Comparison Table of Result

Case	Graph	Iteration and Elapsed Time	
		Dynamic Programming	AlgoritmaGenetika
1	Graph: 0 20 42 25 20 0 30 34 42 30 0 10 25 34 10 0	Length: 85 Iteration: 40 Elapsed Time(ms):0	Length: 85 Iteration: 20401 Elapsed Time(ms):259
2	Graph: 0 29 64 79 48 29 0 5 37 90 64 5 0 31 79 79 37 31 0 23 48 90 79 23 0	Length: 136 Iteration:145 Elapsed Time(ms):0	Length: 136 Iteration: 22901 Elapsed Time(ms): 218
3	Graph: 0 47 86 30 47 96 47 0 27 96 56 37 86 27 0 32 99 73 30 96 32 0 79 95 47 56 99 79 0 60 96 37 73 95 60 0	Length:233 Iteration : 456 Elapsed Time(ms): 0	Length:233 Iteration:25401 Elapsed Time(ms):256
4	Graph: 0 53 24 17 51 43 3 53 0 64 48 6 74 56 24 64 0 76 20 91 41 17 48 76 0 61 3 89 51 6 20 61 0 22 31 43 74 91 3 22 0 9 3 56 41 89 31 9 0	Length:113 Iteration:1309 Elapsed Time(ms):0	Length:113* Iteration: 27901 Elapsed Time(ms): 196

RESULT AND ANALYSIS

The test results show that the two algorithms, dynamic programming and genetic algorithms, can produce an optimum solution. However, if the graph size increases, the genetic algorithm produces an uncertain result. This is because genetic algorithms contain elements of randomness in generating populations and their mutations. The genetic algorithm can only produce a cost path that is close to the actual solution. In addition, genetic algorithms tend to take longer in the search process than in dynamic programming. This is shown by the number of iterations, and the time required is more than that required for dynamic programming.

CONCLUSION

In conclusion, dynamic programming algorithms are better used for TSP cases than genetic algorithms. Although the genetic algorithm is much more intuitive to implement than the dynamic programming technique, the performance of dynamic programming outperforms that of the genetic algorithm. Depending on the implementation, the genetic algorithm might perform similarly performance than the dynamic programming.

ACKNOWLEDGMENT

The authors would like to thank the people who supported this study mentally and financially.

REFERENCES

[1] Zaeri, M.S., Shahrabi, J, Pariazar, M, Morabbi, “A. A combined spatial cluster analysis - travelling salesman problem approach in location-routing problem: A case study in Iran”. 2007 IEEE.

[2] Hu, Bin, Raidl, Gunther R. “Solving the Railway Traveling Salesman Problem via a Transformation into the Classical Travelling Salesman Problem”. 2008. IEEE

[3] Cai, and Zhaoquan. “Multi-direction Searching Ant Colony Optimization for Traveling Salesman Problems”.2008. IEEE

[4] Song, Weitang, Zang, Shumei. “A novel adaptive particle swarm optimization to solve traveling salesman problem”.2009. IEEE

[5] Wang, Xuan, Zhang, GhanNian, Li, Yuan-xiang. “A novel adaptive particle swarm optimization to solve traveling salesman problem”. 2009. Springer.

[6] Da Zhi, Wang., Xin, Liu Shi. “An Agent-based Evolutionary Search for Dynamic Travelling Salesman Problem”. 2010. IEEE

[7] Piriyaniti, Ittiporn, Pongchairerks, Pisut. “Novel VNS Algorithms on Asymmetric Travelling Salesman Problem”.2010. IEEE

[8] Rai, Kartik., Madan, Lokesh., Anand, Kislay. "Research Paper on Travelling Salesman Problem And it's Solution Using Genetic Algorithm". 2014. IJIRT Volume 1 Issue 11

[9] Mzili, Ilyass, Bouzidi, and Morad. "A novel hybrid penguins search optimization algorithm to solve travelling salesman problem".2015.IEEE

[10] Bai, Qiuying., Li, Giuzhi., Sun, Qiheng. "A novel hybrid penguins search optimization algorithm to solve travelling salesman problem".2015. 8th International Conference on BioMedical Engineering and Informatics.

[11] Chen, Muhao., Gong, Chen., Li, Xialong., Yu, Zongxin. "Research on Solving Traveling Salesman Problem Based on Virtual Instrument Technology and Genetic Annealing Algorithm".2015.IEEE

[12] Agharghor, Amine., Riffi, Mohammed Essaid., Chebihi, Faycal. "A Memetic Hunting Search Algorithm for the Travelling Salesman Problem".2016. IEEE

[13] Tongchan, Tanapat, Pornsing, Choosak, Tonglim, Tongtang. "A Memetic Hunting Search Algorithm for the Travelling Salesman Problem".2017. IEEE

[14] Zhou, Ai Hua., Zhu, Li Peng., Hu, Bin., Deng, Song., Song, Yan., Qiu, Hongbin, Pan, Sen. "Traveling-Salesman-Problem_Algorithm_Based_on_Simulated Annealing and Gene Expression Programming".2018. MDPI.

[15] Xutong, Li., Yan, Zheng. "Artificial Bee Colony Algorithm and Its Application in Traveling Salesman Problems".2019. IEEE

[16] Biswas, B. Mitra, A. Sengupta, S. "A Study of Travelling Salesman Problem Using Reinforcement Learning Over Genetic Algorithm".2020. Turkish Journal of Computer and Mathematics Education

[17] Liu, Junjun., Li, Wenzheng. "Greedy Permuting Method for Genetic Algorithm on Travelling Salesman Problem".2018. IEEE

[18]. Taoshen, Li, Zhihui, GE. A Multiple-QoS AnyCast Routing Algorithm-based Adaptive Genetic Algorithm. 3rd International Conference on Genetic and Evolutionary Computing. 2009

[19]. Simon, Philomina., Sathya, S Siva. Genetic Algorithm for Information Retrieval. IAMA. 2009

[20]. Cheng, Chi-Tsun., Leung Henry. Genetic algorithm-inspired UUV path planning based on dynamic programming. IEEE Transactions on Systems, Man and Cybernetics Part C Application and Reviews Vol 42 No 6 November. 2012

[21]. Roberge, Vincent., Tarbouchi, Mohammed., Labonte, Gilles. Comparison of parallel Genetic Algorithm and Particle Swarm Optimization for real-time UAV path planning. IEEE Transactions on Industrial Informatics, 9, no. 1, February. 2013

[22]. Chaurasiya, Arjun Prasad, Sah, Roshan, and Dr. V. Sivakumar. Energy-efficient routing for underwater acoustic sensor networks using a genetic algorithm. 2022

[23]. Shakhtareh, Hazim., Sawalmeh Ahmad., Alenezi, Ali H., Sharief, Abdul Razeq, Almutiry, Muhammad, Ala Al-Fuqaha. Mobile-IRS Asisted Next-Generation UAV Communication Networks. 2022

[24]. Jiao, Ziyuan., Niu, Yida., Zhang, Zeyu., Zu, Song Chun., Zhu, Yixin., Liu, Hangxin. Planning Sequential Task on Contact Graph. 2022

[25]. Virgolin, Marco, Pissis, and Solon P. The symbolic Regression is NP Hard. 2022

****