Vehicle Routing Problem
Based On Improved Particle Swarm Optimization

INFO 6205 - Final Project

AUGUST 28, 2017

HUIZHE WANG
001614841

Table of Contents

Abstract

The vehicle routing problem (VRP) is a combinatorial optimization and integer programming problem. It talks about the optimal set routes for a fleet of vehicles to traverse in order to deliver to a given set of customers. It generalizes the well-known travelling salesman problem (TSP).

Vehicle Routing Problem (VRP) is an important content in the study of logistics management. Choosing the right vehicle routing can accelerate the speed of response to customer requirement, improve the quality of service, enhance customer, and decrease the cost of service operation.

According to different emphasis, the vehicle routing problem has varieties of classification. The project mainly studies the shortest path problem in VRP. The method used in this project is Particle Swarm Optimization (PSO), which is a computational method which was proposed by Eberhart and Kennedy in 1995. The programming language is Java and programming environment is Eclipse.

Key words: PSO, VRP, Java, shortest path problem

1 Introduction

1.1 Vehicle Routing Problem (VRP)

Vehicle Routing Problem (VRP) is an integer NP-complete programming problem. It is a generic name given to a class of problems to determine a set of vehicle routes, in which each vehicle departs from a given depot, serves a given set of customers, and returns back to the same depot. Various types of service appear in practical situations, while physical delivery of goods is the most common one.

The basic VRP involves a single depot, a fleet of identical vehicles that stations at the depot, and a set of customers who require delivery of goods from the depot. The objective of basic VRP is to minimize the total routing cost, subject to maximum working time and maximum capacity constraints on the vehicles. Besides the basic VRP, many VRP variants may appear since there are many possibilities in real-life problem settings and characteristics, for example: the number of depots, type of vehicle, and customer requirements.

This project only talks about the shortest path problem in basic VRP using Particle Swarm Optimization (PSO) algorithm.

1.2 Particle Swarm Optimization (PSO)

As is well-known, gregarious insects would like to foraging, against the enemy, and build their nests by groups. An individual can only do simple task, and a group can complete complex tasks, the "intelligence" displayed by group called swarm intelligence. The new algorithm, designed from the cooperation by gregarious insects, is called swarm intelligence algorithm.

Particle swarm optimization algorithm is a new kind of swarm intelligence optimization algorithm. The idea comes from the artificial life and evolutionary computation theory. While some other population-based search methods had been successfully applied in broader area of VRP, such as genetic algorithm and ant colony optimization, the application of PSO

on VRP is still rare. However, this algorithm is more simple and effective than others with many advantages. Particles could search according to the situation in the solution space, it is easy to implement, there are no many parameters need to be adjusted, and convergence speed is fast, etc.

In the PSO, a solution of a specific problem is being represented by multi-dimensional position of a particle and a swarm of particles is working together to search the best position which correspond to the best problem solution. In each PSO iteration, every particle moves from its original position to a new position based on its velocity, where particles' velocity is influenced by the cognitive and social information of the particles. The cognitive information of a particle is the best position that has been visited by the particle, i.e. position that provides the best objective function, and the most common social information of the particles is called the global best position, the best position that has been visited by all particles in the swarm.

The remainder of the paper is organized as follows: First, in section 2, this report will talk about the application of the PSO algorithm and the setup for the VRP in a formal way. Parts of codes and results are presented in sections 3. The last section summarizes the findings and concludes the report.

2 PSO framework for solving VRP

2.1 Original Particle Swarm Optimization algorithm

The original PSO use the inertia weight w. The inertia weight is used to control the impact of previous histories of velocities on the current velocity. The particle adjusts its trajectory based on information about its previous best performance and the best performance of its neighbors. The inertia weight w is, also, used to control the convergence behavior of the PSO. A number of different alternatives for the definition of w have been proposed. These alternatives vary from constant values to different ways of increasing or decreasing of w during the iterations. The equation of original PSO are as follows:

$$(1) v_{ij}(t+1) = w * v_{ij}(t) + c_1 * rand_1 * (pbest_{ij} - x_{ij}(t)) + c_2 * rand_2 * (gbest_{ij} - x_{ij}(t))$$

(2) $x_{ij}(t+1) = x_{ij} + v_{ij}(t+1)$

Where,

- $v(t+1)$ = updated velocity
- w = inertia weight
- v_{ij}=initial velocity
- c1 = social parameter
- c2 =cognitive parameter
- pbest = personal or local best
- gbest = global best
- x_{ij} = initial Particle position
- $x_{ij}(t+1)$ =updated particles position
- $rand_1$, $rand_2$= Random numbers between 0 and 1

With the formula mentioned above, the steps in IPSO would be as follows:

1) Initialize a population (array) of particles with random positions and velocities on‘d’ dimensions in the problem space.
2) For each particle, evaluate the desired optimization fitness function in d variables.
3) Compare particle's fitness evaluation with particle's pbest. If current value is better than pbest, then set pbest value equal to the current value, and the pbest location equal to the current location in d-dimensional space.
4) Compare fitness evaluation with the population's overall previous best. If current value is better than gbest, then reset gbest to the current particle's array index and value.
5) Change the velocity and position of the particle according to equations (1) and (2).
6) Loop to step (2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations (generations).

2.2 Improved Particle Swarm Optimization algorithm

This project used an improved Particle Swarm Optimization algorithm to solve VRP.

Different from the original PSO, the "swapping operator" is defined for solving this kind of problem based on PSO. Take a solution sequence of this problem with n nodes, S = (Li), i=l ... n. S_0 (i1, i2) can be define as exchanging node N_i1 and node N_i2 in solution S. Then S'=S+ S_0 (i1, i2) can be defined as a new solution on which operator S_0 (i1, i2) acts.

Suppose there is a vehicle routing problem with Five nodes, solution: S = (6, 7, 8, 9, 10). The Swap Operator is S_0 (3, 4), then:

S' = S + S_0 (3, 4) = (6,7,8,9,10) + (3, 4) = (6,7,9,8,10)

There are some concepts should be followed:

(1) A Sequence of swap SS is having one or more Swap Operators. SS = (SO_1, SO_2, SO_3, ..., SO_n), where SO_1, SO_2, SO_3, ..., SO_n are Swap Operators.

(2) Swap Sequence acting on a solution can be called as all the Swap Operators of the swap Sequence act on the solution in order.

(3) Various Swap Sequences acting on the same solution may create the same new solution. All these Swap Sequences are called the equivalent set of Swap Sequences.

Formula (1) in the original PSO has already no longer been suitable for the VRP problem. It can be update as follows:

$$V_{id} = V_{id} \oplus \alpha * (P_{id} - X_{id}) \oplus \beta * (P_{id} - X_{id}) \quad (3)$$

Where α, β are random number between 0 and 1. α *(Pid-Xid) means all Swap Operators in Basic Swap Sequence (Pid-Xid) should be maintained with the probability of α. β*(Pgd-Xid) means all Swap Operators in Basic Swap Sequence (Pgd-Xid) should be maintained with the probability of β. From here it can be seen that the bigger the value of α the greater the influence of Pid is, for more Swap Operators in (Pid-Xid) will be maintained, it is also the same as β*(Pgd-Xid). The steps are as follows.

Step 1: Each of the particles gets a random solution and a random Swap Sequence, namely velocity.

Step 2: If the algorithms are ended, go to step 5.

Step 3: For all the particles in position Xid, calculating the next position Xid'.

- Calculating difference between Pid and Xid, according to the method that has been Shown above, A=Pid-Xid, where A is a basic sequence
- Calculating B=Pgd-Xid, B is also a basic sequence,
- Calculating velocity Vid according to formula (3), and then transform swap Sequence Vid to a basic Swap Sequence,
- Calculating new solution Xid = Xid + Vid = + (4)

  Equation (4) means that Swap Sequence Vid acts on solution Xid to get a new solution.
- If the new solution is superior to Pid then update Pid.

Step 4: If there is new best solution; which is Superior to Pgd then update Pgd. Go to step 2.

Step 5: Draw the global best solution.

2.3 Setup for Vehicle Routing Problem (VRP)

The VRP concerns the service of a delivery company. How things are delivered from one or more depots which has a given set of home vehicles and operated by a set of drivers who can move on a given road network to a set of customers.

In this project, a set of *n* customers (total number is 48) require a delivery service from only one depot. A vehicle (ignore its capacity) is stationed at the depot. The depot and customer locations are known. Each customer is visited exactly once and the route must start and end at the depot.

3 Codes and Results

For completing this project, there are main functions need to done.

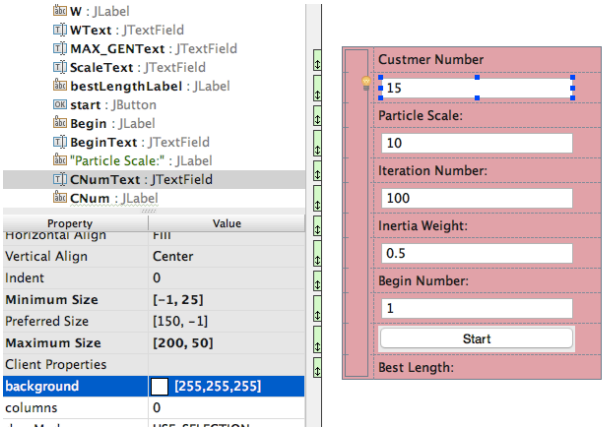
(1) New PSO algorithm.

```
float ra = 0f;
float rb = 0f;
ArrayList<SO> Vi;
ArrayList<SO> Vii = new ArrayList<SO>();

// update speed

// Vii=wVi+c1*ra*(Pid-Xid)+c2*rb*(Pgd-Xid)
Vi = listV.get(i);

// get SO(X,Y) of particle i
// wVi+
len = (int) (Vi.size() * w);

for (j = 0; j < len; j++) {
    Vii.add(Vi.get(j));
}

// Pid-Xid
ArrayList<SO> a = minus(pBest[i], Particle[i]);
ra = random.nextFloat();

// ra(Pid-Xid)+
len = (int) (a.size() * ra);
for (j = 0; j < len; j++) {
    Vii.add(a.get(j));
}

// Pgd-Xid
ArrayList<SO> b = minus(gBest, Particle[i]);
rb = random.nextFloat();

// rb(Pid-Xid)+
len = (int) (b.size() * rb);
for (j = 0; j < len; j++) {
    SO tt= b.get(j);
    Vii.add(tt);
}

// Save new Vii
listV.set(i, Vii);

// update location
// Xid'=Xid+Vid
add(Particle[i], Vii);
```

(2) Multi-threading to calculate particle routes.

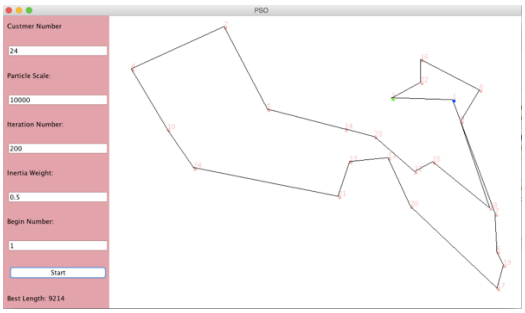
```
// iteration
for (t = 0; t < MAX_GEN; t++) {
    // calculate particle routes using multithreading
    ArrayList<Callable<Void>> runnables = new ArrayList<>();
    // Every particle
    for (i = 0; i < scale; i++) {
        if(i == bestNum) continue;
        final int ii = i;
        runnables.add(new Callable<Void>() {
            @Override
            public Void call() {
                search(ii);
                return null;
            }
        });

    }
    try {
        List<Future<Void>> futures = executorService.invokeAll(runnables);
        for (Future<Void> future : futures) {
            future.get();
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (ExecutionException e) {
        e.printStackTrace();
    }
```

(3) Draw the route in UI.

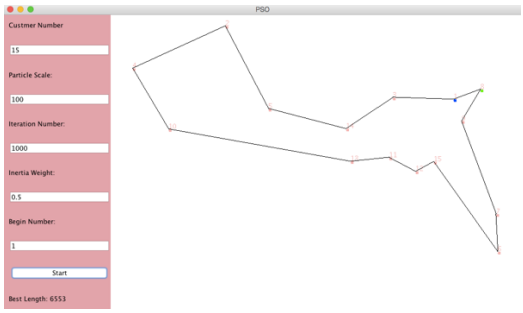



```
class GUICanvas extends Canvas {
    public GUICanvas() {
        // set background
        setBackground(Color.WHITE);
    }

    public void paint(Graphics graphics) {
        try {
            graphics.setColor(Color.PINK);
            for (int i = 0; i < cNum; i++) {
                graphics.fillOval( x: x[i] / 10,  y: y[i] / 10,  width: 6,  height: 6);
                graphics.drawString(String.valueOf(i + 1),  x: x[i] / 10,  y: y[i] / 10);
            }

            if (isStarted == true) {
                graphics.setColor(Color.BLACK);
                for (int j = 0; j < cNum - 1; j++) {
                    try {
                        Thread.sleep( millis: 200);
                        graphics.drawLine( x1: x[bestTour[j]] / 10,  y1: y[bestTour[j]] / 10,  x2: x[bestTour[j + 1]] / 10,  y2: y[bestTour[j + 1]] / 10);

                    } catch (InterruptedException ex) {
                        ex.printStackTrace();
                    }
                }
                graphics.drawLine( x1: x[bestTour[0]]/10, y1: y[bestTour[0]] / 10,  x2: x[bestTour[cNum - 1]] / 10,  y2: y[bestTour[cNum - 1]] / 10);

                graphics.setColor(Color.BLUE);
                graphics.fillOval( x: x[bestTour[0]] / 10,  y: y[bestTour[0]] / 10,  width: 6,  height: 6);
                graphics.setColor(Color.GREEN);
                graphics.fillOval( x: x[bestTour[cNum - 1]] / 10,  y: y[bestTour[cNum - 1]] / 10,  width: 6,  height: 6);

            }
        } catch (Exception e) {
            e.printStackTrace();
        }
```

After run it, we can see the result as follows (cNum = 24 and 15 respectively):




```
Best Generation: 85
Best Length: 9214
Best Route:
0 6 5 18 16 19 10 12 20 23 9 3 1 4 13 22 11 14 17 8 7 15 21 2
```




```
Best Generation: 15
Best Length: 6553
Best Route:
0 2 13 4 1 3 9 12 10 11 14 5 6 8 7
```

Run No	Customer Number	Particle Scale	Inertia Weight	Best Length
1	15	1000	0.5	7035
2	15	1000	0.85	7936
3	15	5000	0.5	6644
4	15	5000	0.85	7035
5	15	10000	0.5	7009
6	15	10000	0.85	7013
7	24	1000	0.5	9330
8	24	1000	0.85	13732
9	24	5000	0.5	9081
10	24	5000	0.85	13636
11	24	10000	0.5	9224
12	24	10000	0.85	12585

As shown above, the smaller inertia weight is, the shorter best length is. When the particle scale is small, there are big influence to the best length. However, when the particle scale is big enough, there is little influence to the best length.

4 Conclusion

The improved Particle swarm optimization algorithm is hereby used to solve the vehicle routing issue. Here the algorithm performs well for small numbers of customers. This can be improved to work for more numbers of customers. In this project, there is only one route for vehicles. This algorithm can also be applied for more route as a further work.

References

(1) Angeline P. Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference. The 7th Annual Conference. On Evolutionary Programming, San Diego, USA, 1998.

(2) Kang-Ping Wang, lan Huang, Chun-Guang Zhou, Wei Pang, “Particle Swarm Optimization For Travelling Salesman Problem”, IEEE 2003.

(3) Kennedy J, and Spears W. Matching algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA, 1998.

(4) Kennedy J, Eberhart R. Particle Swarm Optimization, IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: 1942-1948, 1995.

(5) KP. Wang, L. Huang, C.G. Zhou, W. Pang, Particle swarm optimization for traveling salesman problem, International conference on Machne Learning and Cybernatics 3 (2003) 1583-1585