Travelling Salesman Problem & Algorithm

(Group 1: Haonan Young Row2 P10, Jiali Wu Row1 P11, Yilang Lu Row3 P11)

catalogue

Travelling Salesman Problem & Algorithm 1

Abstract: 2

1. Background and Introduction 2

2. Content 1 3

3. Content 2 4

3.1 Nearest Neighbour Algorithm. 4

3.2 P=NP? 5

4. Further reading and study 6

Reference 8

Appendix 8

Abstract:

The travelling salesman problem (TSP) asks the following question: 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 article is divided into 3 contents, one is called the Introduction, written by Jiali Wu,which is the introduction of the TSP, and the second part is called the algorithm, written by Haonan Young, which give a brief introduction about the Nearest Neighbour Algorithm and the The P versus NP problem. The 3rd part is written by Yilang Lu, which showed a more difficult question about TSP.

Key words: TSP/Algorithm

1. Background and Introduction

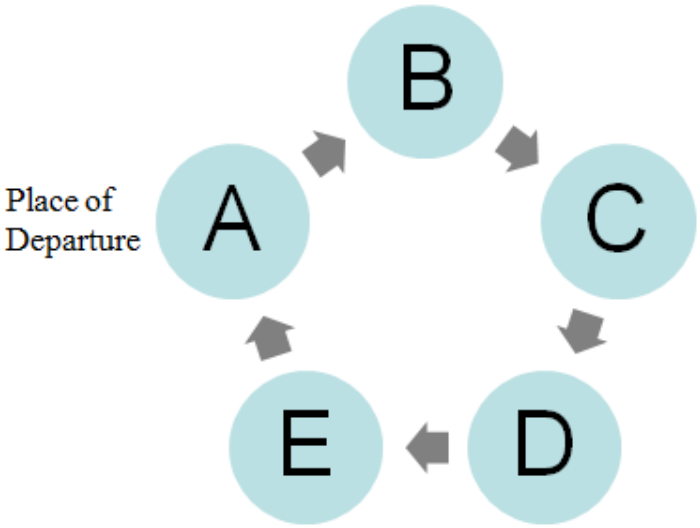
It grew out of the trio's efforts to find solutions for a classic mathematical problem—the "Traveling Salesman" problem—which has long defied solution by man, or by the fastest computers he uses.

—IBM Press Release, 1964.1

An advertising campaign by Procter & Gamble caused a stir among applied mathematicians in the spring of 1962. The campaign featured a contest with a $10,000 prize. Enough to purchase a house at the time. From the officia rules: Imagine that Toody and Muldoon want to drive around the country and visit each of the 33 locations represented by dots on the contest map, and that in doing so, they want to travel the shortest possible route. You should plan a route for them from location to location which will result in the shortest total mileage from Chicago, Illinois back to Chicago, Illinois.

Police officer Toody and Muldoon navigated Car 54 in a popular American television series. Their 33-city task is an instance of the traveling salesman problem, or TSP for short. In its general form, we are given a collection of cities and the distance to travel between each pair of them. The problem is to find the shortest route to visit each city and to return to the starting point. Be attention to a detail that you can only go to one location one time.

2. Content 1



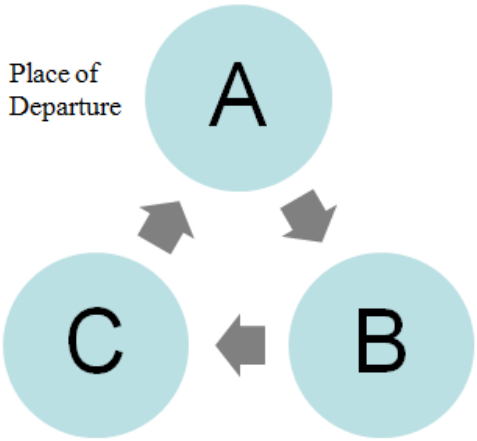
This problem is complicated and hard to us to work out, so we can solve a simpler problem first:

If we will have a trip contains **5** cities. What is the shortest route? Assuming the distance of each two cities is knew, and we have to return to the place of departure. (We can only go to one city one time.)

In this problem, we only have **5** cities, so we can enumerate all the ways to calculate the result.

Solution: In the left picture, we assume that we start at A, then we have **4** choices(B、C、D、E) to the next destination, and still assuming we get to B, then we have **3** choices(C、D、E) to the next, like this, we do the remaining steps, and then we can get **4×3×2×1=24** ways, but it's not right. Try to think, if we go the way of A → B → C → D → E → A and the way of A → E → D → C → B → A, the distance of this two way is equivalent. Thus, we have **(5-1)!/2=12** ways in all. And then, we can calculate the shortest route and get the result.

But how about ***n*** cities? **(*n*>=4)**



Be attention to the range of ***n***, why ***n*** should be larger than or equal to **4**?

Let's do an assumption, in the right picture, if there are only **3** cities to travel, we have the way of A→B→C→A and A→C→B→A, but the distance of this two ways is same. So we only have one way, it's meaningless to this problem. Hence, ***n*** cannot be smaller than **4**.

Certainly, we can easily conclude that if we have ***n*** cities to travel, we have **(n-1)!/2** ways, but if the ***n*** is large enough, it's a hard work to calculate the shortest route. For example, if ***n*=33**, then **32!/2= 131,565,418,466,846,765,083,609,006,080,000,000**, it's a horrible number. Hence, an idea came into our brain that we can use computer to help us solve this problem.

Next, we choose a computer, which is relatively faster than many others, delivering up to **$1{,}457\times10^{12}$** arithmetic operations per second. Let's assume we can arrange the

search for tours such that examining each new one requires only a single arithmetic operation. We would then need roughly **28×10^{12}** years to solve the 33-city TSP with a computer, an uncomfortable amount of time, given that the universe is estimated to be only **14×10^{9}** years old.

Obviously, it's impossible to solve this problem with a method of enumerating. Hence, we need a better and more convenient algorithm to solve it.

3. Content 2

3.1 Nearest Neighbour Algorithm.

The traveling salesman problem (TSP) asks the following question: 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? It is an NP-hard problem in combinatorial optimization, important in operations research and theoretical computer science.

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept city represents, for example, customers, soldering points, or DNA fragments, and the concept distance represents travelling times or cost, or a similarity measure between DNA fragments. The TSP also appears in astronomy, as astronomers observing many sources will want to minimise the time spent slewing the telescope between the sources. In many applications, additional constraints such as limited resources or time windows may be imposed.

In the class, I choose a easy way to find the answer,, but we cannot find a standard way to solve it. It called **nearest neighbour algorithm.**

The nearest neighbour algorithm was one of the first algorithms used to determine a solution to the travelling salesman problem. In it, the salesman starts at a random city and repeatedly visits the nearest city until all have been visited. It quickly yields a short tour, but usually not the optimal one.

Below is the application of nearest neighbour algorithm on TSP

These are the steps of the algorithm:

- **start on an arbitrary vertex as current vertex.**
- **find out the shortest edge connecting current vertex and an unvisited vertex V.**
- **set current vertex to V.**
- **mark V as visited.**
- **if all the vertices in domain are visited, then terminate.**
- **Go to step 2.**

- **The sequence of the visited vertices is the output of the algorithm.**

The nearest neighbour algorithm is easy to implement and executes quickly, but it can sometimes miss shorter routes which are easily noticed with human insight, due to its "greedy" nature. As a general guide, if the last few stages of the tour are comparable in length to the first stages, then the tour is reasonable; if they are much greater, then it is likely that there are much better tours. Another check is to use an algorithm such as the lower bound algorithm to estimate if this tour is good enough.

In the worst case, the algorithm results in a tour that is much longer than the optimal tour. To be precise, for every constant r there is an instance of the traveling salesman problem such that the length of the tour computed by the nearest neighbour algorithm is greater than r times the length of the optimal tour. Moreover, for each number of cities there is an assignment of distances between the cities for which the nearest neighbor heuristic produces the unique worst possible tour.

The nearest neighbour algorithm may not find a feasible tour at all, even when one exists.

3.2 P=NP?

The P versus NP problem is a major unsolved problem in computer science. Informally, it asks whether every problem whose solution can be quickly verified by a computer can also be quickly solved by a computer. It was essentially first mentioned in a 1956 letter written by Kurt Gödel to John von Neumann. Gödel asked whether a certain NP-complete problem could be solved in quadratic or linear time.The precise statement of the P versus NP problem was introduced in 1971 by Stephen Cook in his seminal paper "The complexity of theorem proving procedures" and is considered by many to be the most important open problem in the field. It is one of the seven Millennium Prize Problems selected by the Clay Mathematics Institute to carry a US$1,000,000 prize for the first correct solution.

The informal term quickly, used above, means the existence of an algorithm for the task that runs in polynomial time. The general class of questions for which some algorithm can provide an answer in polynomial time is called "class P" or just "P". For some questions, there is no known way to find an answer quickly, but if one is provided with information showing what the answer is, it is possible to verify the answer quickly. The class of questions for which an answer can be verified in polynomial time is called NP.

Consider the subset sum problem, an example of a problem that is easy to verify, but whose answer may be difficult to compute. Given a set of integers, does some nonempty subset of them sum to 0? For instance, does a subset of the set {−2, −3, 15, 14, 7, −10} add up to 0? The answer "yes, because the subset {−2, −3, −10, 15} adds up to zero" can be quickly verified with three additions. However, there is no known algorithm to find such a subset in polynomial time (there is one, however, in exponential time, which consists of 2n-n-1 tries), but such an algorithm exists if P =

NP; hence this problem is in NP (quickly checkable) but not necessarily in P (quickly solvable).

An answer to the P = NP question would determine whether problems that can be verified in polynomial time, like the subset-sum problem, can also be solved in polynomial time. If it turned out that P ≠ NP, it would mean that there are problems in NP (such as NP-complete problems) that are harder to compute than to verify: they could not be solved in polynomial time, but the answer could be verified in polynomial time.

Aside from being an important problem in computational theory, a proof either way would have profound implications for mathematics, cryptography, algorithm research, artificial intelligence, game theory, multimedia processing, philosophy, economics and many other fields.

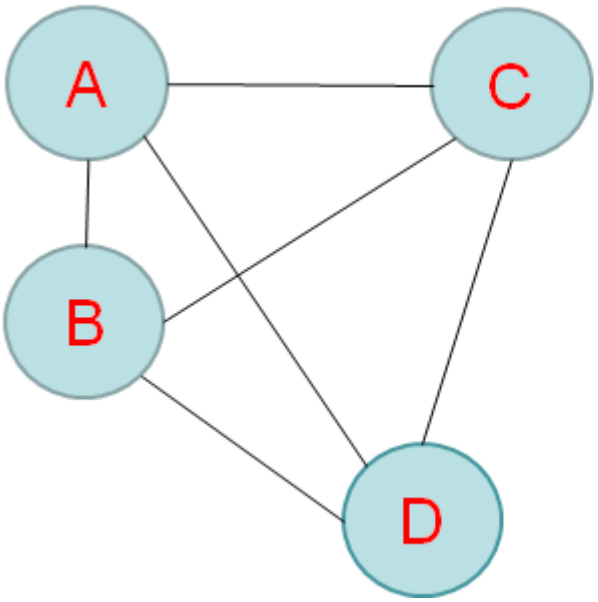
4. Further reading and study

Let's think about a more difficult problem. As we know the Traveling Salesman Problem (TSP) only has one request that is the route is the shortest. But what if it has two requests, which is more close to our daily life. Because when people decide to take a trip , they usually consider many things, like money, time and etc.

So the more difficult problem is:

If we will have a trip contains 4 cities. Using A B C D to represent 4 cities, we will start from A and finish at A. There are two requests:

(1) the route is the shortest (2) using the least money



A—B： 100km ￥200
A—C： 200km ￥300
A—D： 300km ￥100
B—C： 250km ￥200
B—D： 200km ￥300
C—D： 200km ￥400

For example: plan A is: A-B-C-D-A and plan B is A-C-D-B-A

Plan A	A-B	B-C	C-D	D-A	total
distance	100	250	200	300	850
money	200	200	400	100	900

Plan B	A-C	C-D	D-B	B-A	Total
distance	200	200	200	100	700
money	300	400	300	200	1200

Look at these two plans, you will find it is a dilemma, for A it uses less money but has a longer distance while for B it has a shorter distance but uses more money.

So the question changes to how to choose the plan. Unfortunately there is not a really useful way to solve this problem, just like the TSP is still not solved. Here I only give my idea for this problem as follow.

First we should give everything a weight. For example money is 70% and distance is 30%, different people will give a different percentage number. Why we should give everything a weight? Because both money and time is important, but we should judge which is more important. For different one it must be a different answer. This problem is based on our daily life, because when we think about our trip, we always think about many things, like money, time, weather and etc.

Second we should handle our data to show which plan is better. Here we do a unitization. We use data to subtract 10^n(n is a integer) to change the data to be a number between 1 and 100. For example C-D is 200KM, so 200÷100=2.

So for Plan A, we change the total distance to 8.5and change the total money to 9. For Plan B, we change the total distance to 7 and change the total money to 12. Then we use these numbers to multiply with the weight.

For Plan A: 8.5x30%+9x70%=2.55+6.3=8.85

For Plan B: 7x30%+12x70%=2.1+8.4=10.5

8.85 is less, so we get the consequence that Plan A is better.

However, this problem only gives 4 cities. What if there are 5 cities, 6 cities or

more? You will find the question is not only a TSP but more difficult. And what if there are three requests, four requests or more? The more requests are given the more likely with daily life the problem is. But as we know, both TSP problem and this further problem still are not solved, this is my idea.

Reference

TSP:

https://en.wikipedia.org/wiki/TSP

P versus NP problem: https://en.wikipedia.org/wiki/P_versus_NP_problem

Appendix

1. It's a great chance for me to have a presentation to share my own ideas. In my presentation, I gave a brief introduction for the algorithm of the TSP, and showed that there isn't exist the best way to solve it, but there always exists a better way to solve it, for example, I choose the nearest neighbour algorithm to show a way to solve it but it has the shortage. Then I choose to show that a deep mathematics called The P versus NP problem. I figure that's a interesting question because if we prove that, we can get a more fantastic world because no matter difficult the problem is, we can always find a quick way to do it. I did these in a short time, I personally think that I tried my best effort to prepare for it, so I did not bad in my presentation, no matter other people's evaluation. As for others comments, I will improve my skills base on the comments, maybe I should slow down my voice and so on. No matter the evaluation, I really had a great time with the classmates and the professor. (Haonan Young)

2. I enjoyed in this course, and it didn't disappoint me. I knew some interesting and attractive knowledge about Mathematics as well. Through this group work, I've learnt a lot of knowledge about traveling salesman problem (TSP), a really complicated problem. We've just knew the meaning of this problem, and only gave our superficial opinions. About this problem, there are many deeper things we need to learn and explore, and also some explanation about our presentation need to improve. Nobody's perfect. As for me, a student of Mathematics Department, the interest of learning and the spirit of exploration also need improvement.(Jiali Wu)

3. First of all I think this is a really nice course and I learn a lot these five days. I think both Mr Li's and classmates' presentations teach me a lot. For my presentation, I think my performance is great because I prepared a lot and give my idea about the problem. However there are many problems I have (I talk with classmates after class about my presentation and let them give me suggestions). I knew while I was doing my presentation, I speak a little bit fast and others cannot understand my words will. What's more, I do not say my idea clearly and also I was a little nervous. As for myself, I do know I cannot do perfectly but I strongly believe I can do better than last time. After this course I know one cannot learn many things in a shout five days, but if we learn how to learn we can learn a lot, this is my greatest achievement in this course. In the future, I want to try to find the way to use easy idea comes from daily life in difficult mathematical problems. (Yilang Lu)