Volume

2

THE TRAVELLING SALESMAN PROBLEM

Heuristics

Author: Prof. Ana Inés Gómez de Castro
Universidad Complutense de Madrid

Astronomy Workshop

THE TRAVELLING SALESMAN

Astronomy Workshop

© Ana Inés Gómez de Castro
Instituto de Astronomía y Geodesia • Madrid
Teléfono (91) 3944578 • Fax (91) 3944615

Table of contents

Chapter

1

Combinatory and Heuristics

The Travelling Salesman Problem

"The star traveller" is an application designed to introduce Secondary/High School students in a basic problem of *Artificial Intelligence* (AI) most commonly known as "the travelling salesman problem". This problem is introduced through visits to other Planetary Systems.

In a first step, *"Learn to plan a trip"* , only 4 systems can be visited, thus it is quite easy to enumerate all the possible paths and evaluate the total distance covered in each one of them :

$$\mathcal{P}_4 = 4! = 4*3*2*1 =48$$

Then, the problem becomes more complicated, as the students are allowed to travel among 10 of the 155 known systems in *"Design a smart trip"*. In this case, the simple enumeration of all the possible paths:

$$\mathcal{P}_{10} = 10! = 10*9*8*7*6*5*4*3*2*1 =3.628.800$$

e.g. more than three million possible combinations!, would take a huge amount of time. Suppose that it takes just few seconds (let's say 3) to your "video-games trained students" to set a full path with our graphic interface, they would either need 12.6 days to test them all, or be VERY lucky!!.

This problem is an astronomic version of the *"Problem of the Travelling Salesman"*, a classic in Operations Research (OR). The problem of the travelling salesman, in itself, presents a "certain" degree of computational complication, as the number of possible solutions increases exponentially with the number of "planets to visit" and no algorithm has been yet found to solve it without enumerating, explicitly or implicitly, all of the feasible solutions.[1] *"The smart space traveller*" is thought as an

[1]These kinds of problems are called "**NP-hard**" (or problems of NP difficulty). Computational complexity is a rigorous mathematical discipline which shows how most optimisation problems can be grouped into classes in such a way that all problems of the same class have a similar complexity. The "**NP-hard**" problems represent the most important class.

introductory exercise to the concept of "heuristic algorithms" or algorithms capable of offering a sufficiently good solution to the problem without having to go into all of the possible solutions.

A large number of heuristics has been developed to solve this kind of problems, from very simple algorithms involving re-organisation of the sequence or path, to much more complicated algorithms, such as neuronal networks or genetic algorithms. The exercise set within this educative application includes an interface to graphically codify a simple algorithm (a greedy algorithm to set a good first approach to the optimal path). The exercise was created so that the students can test a very basic greedy algorithm and learn the complexity hidden behind any apparently reasonable approach. HOU-Spain plans to insert new capabilities as the project develops in the classrooms and feedback from teachers let us know which are the most reasonable algorithm-building graphic interfaces that are meaningful for secondary/high school students. The rest of this chapter contains some simple algorithms which can be used by the teachers to initiate students in the use of this tool and some basic artificial intelligence projects.

Heuristic to solve the travelling salesman problem.

There are different types of heuristics, according to the way in which they look for and build solutions. A possible classification divides them into constructive heuristics and iterative improvement

a) ***Constructive or pure methods:*** These methods generate the travelling salesman path by adding vertices according to some specific criterion. Once a reasonable path has been identified, these algorithms do not attempt to improve it. The most popular among these methods are the "greedy" algorithms, which generate the solution step by step, seeking for the maximum benefit at each step.

b) ***Methods of iterative or local improvement:*** These methods do not try to reach a feasible solution, rather they start from one of them (obtained perhaps through a constructive method) and then, improve it through an iterative process that ends when a given condition is fulfilled.

Simple algorithms tend to have well-defined rules of termination, and they stop at a local optimal. More complex algorithms may not have standard rules of termination and typically search for better solutions until they reach an arbitrary stopping point.

Constructive or pure algorithms

The ***nearest neighbour algorithm*** uses a very basic idea to construct a feasible route. It generates a route iteratively by selecting at each step the city closest to this you are at (this is the algorithm implemented in the graphical interface). At each step, the method needs a vertex n_0, the departure point, and another, t, that is seek among all the possible vertices (included in W), which have not yet been visited. Initially $t = n_0$ and $W = N \setminus \{ n_0 \}$.

ALGORITHM – THE NEAREST NEIGHBOUR

```
    Step 0: Select an arbitrary vertex $n_0$. Make $t = n_0$ and $W = N \setminus \{ n_0 \}$
    Step 1: As $W \neq \varnothing$, do the following:
    choose $j \in W$ so that $c_{tj} = \min \{ c_{ti} \mid i \in W \}$
    connect $t$ to $j$ and write $W = W \setminus \{ j \}$ and $t = j$.
    Step 2: Connect $t$ to the initial vertex $n_0$ to form a Hamiltonian cycle.
```

A slight variation of this algorithm consists of permitting the extension of the route in either directions, this is named the nearest bilateral neighbour algorithm.

Heuristics of iterative improvement

To improve on a suitable solution chosen at random (for example, by means of the nearest neighbour method), these heuristics change in an iterative manner the order of the cities/stars to be visited. They all fit within these simple steps:

1. Generate a feasible initial S solution. Let T(S) be its target function, for instance, the total amount of fuel spent in the path.

2. Try to find an improved possible S' solution, by means of some transformation of S.

3. If a better solution is found (*e.g.*, if T(S') < T(S) assuming that you wish to minimize the fuel used up) , replace T with T' and repeat from step 2.

4. If a better solution cannot be found, S is the local optimal solution.

What differentiates and characterises each iterative procedure is step 2. Basically, these methods find the first local optimum of the target function T(S) and stop the iteration there, preventing to reach the global optimum unless this is the first found.. Therefore, the existence of local optima is one of the greatest inconveniences of this type of heuristics.

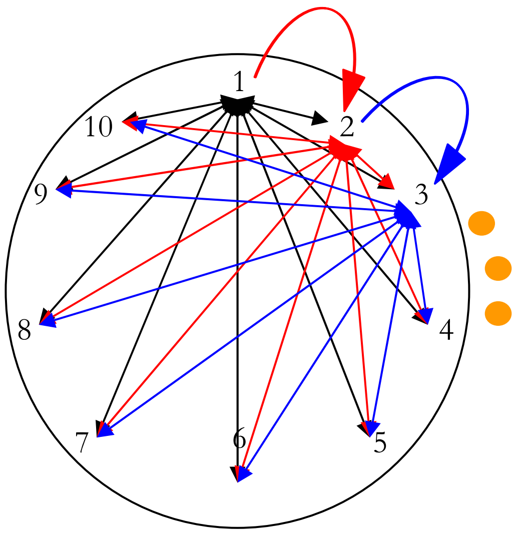
A related problem is the dependence of the goodness of the final solution (S') on a cleaver selection of the initial solution (S). Obviously, the initial solution (S) has a great influence on the possibility of falling or not falling into a local optimum that is not the global one.

Some simple iterative improvement heuristics

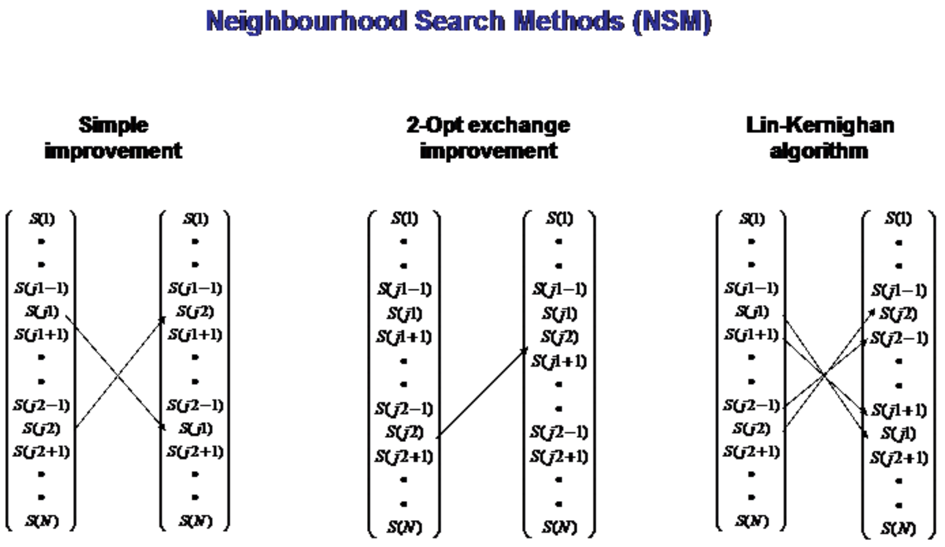
We propose three simple exchange transformations to try with the students: *Simple Improvement, 2-Opt exchange*, and *Lin-Kernighan Algorithms*. The simplest algorithm is the *2-Opt exchange*: in this algorithm, the route of the trip is reorganised, interchanging the order in which two stars are visited. The first star is selected and the voyage is reorganised, interchanging it with all the possible stars. The distance in all of these permutations is evaluated and the best one is chosen. Later, another star is chosen and the procedure is repeated. In this way, instead of calculating N! permutations, we calculate

$$N\sum_{i=1}^{i=N-1} i$$

possible paths. For example, in the case of visiting 10 stars, it would not be necessary to calculate the 10! posible permutations, rather only 10(9+8+7+6+5+4+3+2+1) =450, as indicated in the figure.



Other algorithms are based on making more complex changes, as indicated in the figure below.


Neighbourhood Search Methods (NSM)
Simple improvement
2-Opt exchange improvement
Lin-Kernighan algorithm
S(1)
S(j1−1)
S(j1)
S(j1+1)
S(j2−1)
S(j2)
S(j2+1)
S(N)