TÉCNICO LISBOA

Search and Planning

Alameda Campus

IST @ 2016/2017

28th October 2016

1. Introduction

This project has two goals: (1) Develop a program in ANSI Common Lisp to solve the Asymmetric Traveling Salesman Problem. Different search strategies studied in the course should be used, and heuristics should be developed to address this kind of problem in the most efficient way; (2) Produce a study evaluating the alternatives implemented both in quantitative and qualitative terms.

2. Description of the Asymmetric Traveling Salesman Problem (ATSP)

Given a set of *n* nodes and distances for each pair of nodes, find a roundtrip of minimal total length visiting each node exactly once. In this case, the distance from node *i* to node *j* and the distance from node *j* to node *i* may be different.

3. Solving the problem

The problem is an optimization problem since it is trivial to obtain a solution. What is difficult is to obtain the optimal solution or even a good solution for large problems. So the goal is to obtain the best possible solution and if possible the optimal solution.

A suitable formulation to solve the problem as efficiently as possible should be chosen. Appropriate techniques for optimization problems should be employed.

4. Implementation

Since part of the evaluation of the program will be done automatically, it is essential that the interface specification is followed strictly.

The set of functions that implements the solver must be defined in a single file, to be compiled without errors or warnings. One of the warnings produced by compilers is the lack of a declaration of a package in the file, in order to avoid this warning, a Lisp form must be added as follows:

```
(in-package :user)
```

This form must be the first one appearing in the file.

Note also that the tests to be carried out automatically impose some reasonable spatial and temporal limits: (1) a 256 MBytes limit on the heap size; and (2) a maximum time 5 minutes for each problem on a Core i5 2.6 GHz. In other words, the temporal and spatial efficiency of the solution search process is relevant, as previously stated.

4.1. Interface

You are to implement the function *`atsp`*, that solves instances of the Asymmetric Traveling Salesman Problem problem.

This function takes as arguments: a problem in its external representation defined below and the search strategy to be used. The function should return the solution of the problem using an external representation of a solution, also defined below.

The strategies to be supported are:

```
"a*.best.heuristic"
"a*.best.alternative.heuristic"
"iterative.sampling "
"best.approach "
```

The external representation of a problem is a square array with the distance between the nodes. The size of each dimension corresponds to the number of nodes to be visited. The initial node corresponds to the element of index 0. In the example below the distance from node 0 to node 2 is 23 and the distance in the reverse direction, from node 2 to node 0, is 30:

```
#2A((9999    4   23   12)
    (   6 9999   22    7)
    (  30   26 9999   75)
    (  15   10   70 9999))
```

Figure 1: Problem representation example.

Figure 2 represents the call to `atsp`, with the problem represented in *Figure 1* that was previously assigned to the variable `prob01` and for which one wants to obtain a solution by using the search strategy `"a*.best.heuristic"`. The solution obtained is presented in *Figure 3*.

```
(astp prob01 "a*.best.heuristic")
```

Figure 2: Start search.

The external representation of a solution is an ordered *list* of all the nodes visited.

```
(0 2 1 3 0)
```

Figure 3: Solution returned by `atsp`.

4.2. Supplied Lisp code

For the implementation of this Project, you should use the search algorithms developed in ANSI Common Lisp, available on the course *site*. The code provided lies in the file '*procura.lisp*'. The code contains the implementation of various search algorithms. Their operation is described in '*procura.txt*'. The most important thing is to understand what they are for, it is not important to understand exactly how the functions are defined. This file should not be changed; if you need to change the existing definitions, you should redefine them in the file that contains the implementation carried out by your group.

5. Study

You are required to develop a program capable of solving the given problem. The problem approach should model the problem as a search in a state space, using an incremental approach and an optimisation perspective. In other words, it should return the first best solution found, the solution of lower cost, or NIL if no solution can be found in the given time.

- All search techniques provided in the file "*procura.lisp*" must be tested. Note that you must change the strategies so that the search process ends within the given time limit and returns the best solution found so far.
- The strategy *iterative sampling* with an optimization perspective must be implemented and tested.
- In addition, new strategies to facilitate/improve the resolution of problems should be developed. These may include variations to basic search algorithms, new search algorithms, hybrid strategies, macro-operators, sub-goals, etc. The problem analysis and the first results obtained should provide indications of valid approaches.

For the informed search techniques, heuristics are needed.

- At least two relevant high performance heuristics must be submitted. These heuristics should, as far as possible, be based on different ideas (heuristics that vary only in one or more constants, for example, are considered the same heuristic).

The performance of the various search strategies used, as well as of the heuristics and cutting strategies developed, should be discussed in the report, making a comparison of the results obtained for some of the instances used for testing. All of this analysis should be based on quantitative results, including: cost of solutions, search time, generated nodes, expanded nodes, average branching factor, maximum depth, solution depth, etc.

The discussion should be limited to highlighting relevant characteristics or limitations of your work. You are encouraged to create new problems, beyond those supplied as examples, which serve to illustrate interesting aspects of the discussion.

The whole decision process must be documented in the report to be delivered as part of this work assignment.

The project report must also include:

- A description of the modeling(s) of the problem and the reasons behind your decisions;
- A description of the data structures developed and the reasons underlying their development;
- The description of the heuristics developed and the reasons for its development;
- A description of the cutting strategies implemented and the reasons for its development;
- The choices made during the development of the project;
- A discussion on how to achieve better results, by indicating the type of heuristics / strategies to be developed or amendments to modeling.

Still on the report, it should be noted that the readability of the same (for which contribute not only the organization of the chapters / sections but also the spelling, the construction of sentences, and the fluidity of speech) is of paramount importance for the proper understanding of its contents. You should avoid complex sentences (using the passive voice, double negatives, etc…). All figures and graphs submitted must be legible and be properly labeled.

6. Evaluation criteria

The most important evaluation factors for the project are (not necessarily in this order):

- Correct execution and elegance of the program;
- Quality of the modeling(s) of the problem;
- Quality of heuristics and strategies developed;
- Efficiency of data structures chosen to represent the problem;

- Quality of the developed study (includes a description of work performed, the results and conclusions);
- Overall assessment.

7. Project registration and delivery

The elements of each group, at most 2, must register in the project via Fénix.

There will be two deliverables: (1) the developed code and (2) the project report.

The delivery of the developed code must be made by 23:59 on the 2nd of December 2016 as follows:

- the file containing the program must be submitted electronically using the Fénix system. The name of the Lisp file to be delivered must be "GXXX.lisp", where "XXX" is the group number, for example "G007.lisp";

The delivery of the report must be made by 23:59 on the 2nd of December 2016 as follows:

• the file containing the report in electronic format ("pdf" or "word" format) must be submitted electronically using the Fénix system. The name of the delivered file should be "GXXX.pdf" or "GXXX.doc, where “XXX” is the group number, for example "G007.pdf";

All material delivered by students (documentation and code file) must identify the group (group number supplied during registration group) and the number and name of each group member.

Attention: No deliveries are accepted after the Project deadline.

8. Project News

In case there are news concerning the project, these will be published on the course site, which should be visited daily.

Search and Planning

Alameda Campus

Assignment (2016/2017)

Group number (obtained from the Fénix system):

Name: Number:

Name: Number:

Grading:

Sum of the hours spent exclusively for developing this work:

Delivery on: 23h59 of the 2nd of December 2016.