Travelling Thief Problem

Evolutionary Computation

Prepared by William Reid, Matthew Hart, Samantha Peachey & Alec Bellati

School of Computer Science,
The University of Adelaide

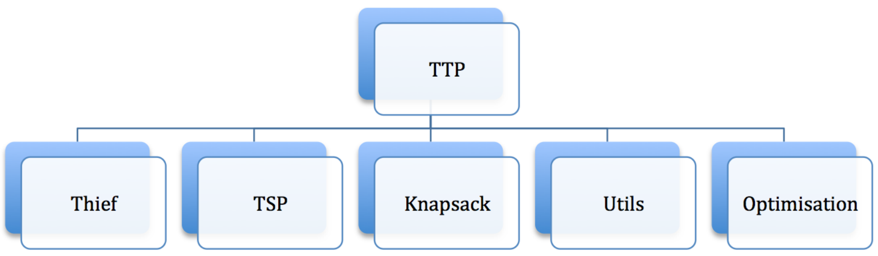
October 3, 2014

Exercise 1

Design Process

The code we were handed was implemented using integer arrays as the primary representation of nodes (cities - both 1D and 2D) to represent the nodes (cities) and items stored within this city. This design, whilst having a very low memory footprint, was structurally not intuitive and caused our group difficulties on several occasions as to what information each row/column combination was giving us. Also, in the previous assignment (The Travelling Salesman Problem) our team adopted an object oriented approach which we deemed successful, and we chose to bring that approach across to the current assignment.

Design 1



Our first design revolved around the notion of breaking each segment of this problem into its own sub-directory such that correct Java packaging techniques could be adopted. The structure was as follows:

- **TTP**: This is the top level directory that would contain the Driver file, along with TTPInstance and TTPSolution.
  - **Thief**: A subdirectory that will contain the algorithms for the thief.
  - **TSP**: A subdirectory that will contain our TSP solution from assignment 1.
  - **Knapsack**: A subdirectory that will contain all classes pertaining to solving the knapsack side of the Travelling Thief Problem.
  - **Utils**: A subdirectory containing a set of utility functions (given to use with the original code).
  - **Optimisation**: A subdirectory containing a set of Optimisation functions (given to us with the original code).

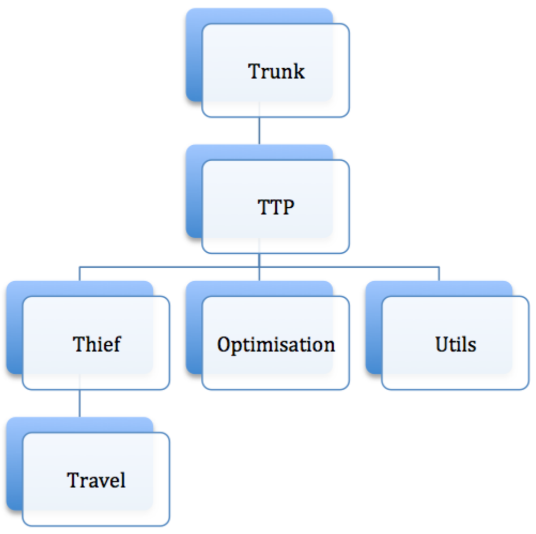
The advantage of this design was in its simplicity and substructure. Each subdirectory had a well defined subset of the problem that it was solving and it was intuitive to a developer as to what code went into each section.

The disadvantage of this design was that the object orientation was not representative of the problem. For example, in assignment 1 the City class was contained in the TSP subdirectory. However, in this problem, cities would contain items. Logically, this would imply that the Item class would reside either within the TSP subdirectory, or in a subdirectory of TSP. Unfortunately this causes some conceptual conflicts as the item class would also have to be present in the Knapsack and Thief subdirectories.

Design 2

Our second design revolved around the same segmentation and object orientation goal as the first design. However this time, problem representation played a larger focus in the design. The second design structure was as follows:

- **Trunk**: This is the top level directory that contains only the Driver.
  - **TTP**: A subdirectory that will contain both TTPInstance and TTPSolution.
    - **Thief**: A subdirectory that will contain all of the TTP algorithms within a single Thief class. This subdirectory will also house the Knapsack class as the Knapsack is directly related to the thief.
      - **Travel**: A subdirectory that will contain all of the TSP classes from the previous assignment with the addition of the Items class.
    - **Optimisation**: A subdirectory containing a set of Optimisation functions (given to us with the original code).
    - **Utils**: A subdirectory containing a set of utility functions (given to use with the original code).



The advantage of this designed was that it ticked all of the design checkboxes that we set out for the project. This adheres to our object oriented design approaches, fulfils the necessity for the separation of classes through the use of packaging but most importantly, the design correctly models the problem. In this design, the Thief has a Knapsack and is the parent of the travel subdirectory which allows the Thief to decide on the best path to take across the map of cities.

The disadvantage of this model is the fact that the Thief would like to evaluate his knapsack/travel solution and as such needs to create a TTPSolution to get this objective value. This issue was later fixed by moving the TTPSolution class out of the TTP subdirectory and into the Thief subdirectory. This design was adopted and implemented as the structure for our teams solution to the Travelling Thief Problem.