

# CAB203 Graphs Project Sample Report

Matthew McKague

This sample project report is based on the Transporter Transportation Problem from Lecture 9, written up as though it were a task in the project. This write-up is on the longer side; not all the tasks will need a write-up this long.

## 1 Transporter Transportation Problem

The *transporter transportation problem* (TTP)  $(L, s, e)$  is defined by a set  $L$  of *links*, and starting point  $s$  and an ending point  $e$ . Each link is a pair of points  $(a, b)$  representing a link between two cities  $a$  and  $b$ . Links are symmetric so that a link  $(a, b)$  is taken to be the same as  $(b, a)$ , although only one of each is present in  $L$ . The links are given in a CSV file with two columns, which we model as a set of pairs, where each pair corresponds to a single row. In this way  $L$  corresponds directly to the CSV file.

The objective of a TTP is to find a sequence of intermediate cities  $c_1, c_2, \dots, c_n$  so that there is a link between  $s$  and  $c_1$ , between  $c_n$  and  $e$ , and between each  $c_j$  and  $c_{j+1}$  so that teleporters can be used to transport goods from  $s$  to  $e$  through the sequence of intermediate cities via links. The solution will be such a sequence of minimal length.

The transporter transportation problem can be solved by finding a shortest path in a graph (see [3] for an introduction to graphs) where vertices are cities and edges are links. More specifically, the graph is given by  $G = (V, E)$  where  $E$  is a *symmetric* (see [1] for a definition) version of  $L$ :

$$E = L \cup \{(b, a) : (a, b) \in L\} \quad (1)$$

and  $V$  is the set of first items from each pair in  $E$ , which is all cities since  $E$  is symmetric:

$$V = \{a : (a, b) \in E\}. \quad (2)$$

In  $G$  we find a shortest path from  $s$  to  $e$ . If no such path exists then there is no solution to the corresponding TTP. Otherwise, suppose that we find a shortest path  $s = v_1, v_2, \dots, v_n = e$ . Then the solution to the TTP is given by  $v_2, \dots, v_{n-1}$ .

The Python implementation closely follows the description above, and so variables  $L$ ,  $E$  and  $V$  are sets. We first use the `csv` module to read the CSV file into a set  $L$  of rows, where we convert each row from a list (as returned by the `csv.reader` object) to a tuple, which is hashable and can be stored in a set, unlike lists. Sets  $E$  and  $V$  are computed using Python set comprehensions analogous to equations (1) and (2). The `solveSPP` function from the `graphs` module [2] is used to find a shortest path and the intermediate vertices are finally returned using a slice.

## References

- [1] Matthew McKague, *CAB203 Lecture 6*. [https://canvas.qut.edu.au/courses/1979/files/1701727/download?download\\_frd=1](https://canvas.qut.edu.au/courses/1979/files/1701727/download?download_frd=1) QUT, 2023.
- [2] Matthew McKague, *graphs.py*. [https://canvas.qut.edu.au/courses/1979/files/1755763/download?download\\_frd=1](https://canvas.qut.edu.au/courses/1979/files/1755763/download?download_frd=1) QUT, 2023.
- [3] Vitaly Voloshin, *Introduction to graph theory*. New York: Nova Science Publishers, 2009.