Compilation Instructions and User Guide

Lewis, R. (2020) 'Algorithms for Finding Shortest Paths in Networks with Vertex Transfer Penalties'. *Algorithms*, vol. 13, 269. doi:10.3390/a13110269 https://www.mdpi.com/1999-4893/13/11/269/pdf

This document describes how to compile and use the algorithms proposed in the above publication. This program has been written in C++ (V11) and is available at http://rhydlewis.eu/resources/spalgs.zip. It can be compiled in Windows using Microsoft Visual Studio and in Linux using the GNU Compiler g++. Instructions on how to do the former follow.

Compilation in Microsoft Visual Studio

To compile and execute using Microsoft Visual Studio the following steps can be taken:

1. Open Visual Studio and click **File**, then **New**, and then **Project from Existing Code**.
2. In the dialogue box, select **Visual C++** and click **Next**.
3. Select the subdirectory containing the source code and click **Next**.
4. Finally, select **Console Application Project** for the project type, and then click **Finish**.

The source code can then be viewed and executed from the Visual Studio application. Release mode should be used during compilation to make the program execute at maximum speed.

Compilation with g++

Please use the **Makefile** included with this resource.

Usage

Once generated, the executable file (called **spsolver**) should be run from the command line. If the program is called with no arguments, the following usage information will be printed to the screen.

```
Shortest Path Program for Graphs with Transfers at their Vertices (Rhyd Lewis, 2020 -- www.rhydlewis.eu)

USAGE:
-i <InputFile>    (Required. File must be in DIMACS format)
----------------------
-a <int>          (Algorithm Choice
                      1: Extended-Dijkstra-V1 (Default)
                      2: Extended-Dijkstra-V1 w/ delete edges operator
                      3: Do Kirby-Potts expansion then use Dijkstra
                      4: Do Kirby-Potts expansion then use Floyd-Warshall
                      5: Do restricted expansion then use Dijkstra
                      6: Do restricted expansion then use Floyd-Warshall
                      7: Extended-Dijkstra-V2)
-single <int>     (If present, the selected algorithm is run from given source only)
-sample <int>     (If present, the selected algorithm is run from the first <int> vertices only, in turn)
-x                (Suppress problem instance checker. If present, the instance's validity is assumed)
-v                (Verbosity. If present, output is sent to screen. If -v is repeated, more output is given.)
----------------------
Notes: - If the -single option is used, the reported duration (in the output log file) includes the time taken to extract all paths
         from tree;
       - If neither -single nor -sample options are selected, the program runs until shortest paths are determined between all vertex
         pairs.
       - Shortest s-t-paths are calculated by allowing any 'in-colour' at both s and t. See Section 5 of the paper (paragraph 2).
       - The restricted expansion is only permitted in instances where all transfer costs are equal. Reasoning is explained in Section
         3.2
```

This provides the information needed to produce valid commands. Here are some examples:

```
spsolver -i graph.txt
```

This will execute the algorithm on the problem instance contained in the supplied example input file **graph.txt** using the default settings (as specified above).

```
spsolver -i graph.txt -v
```

As above, but the algorithm produces more output (the pairwise distance matrix in this case)

```
spsolver -i graph.txt -v -single 0 -a 7
```

Algorithm 7 (Extended-Dijkstra-V2) is used to produce a shortest path tree rooted at vertex 0. This is then written to the screen.

Input

An example input file **graph.txt** is included in this resource. Any input file should follow the form of this file. Here are the first and last few lines of the file.

```
c FILE graph.txt
c SOURCE This is a random shortest path graph formed using the generator of Rhyd Lewis
c
c GRAPH STATS
c Seed                          = 1
c Number of Nodes               = 100
c Density                       = 0.1
c Number of colours             = 5
c
p edge 100 939 5 1449
e 1 27 1073805 4
e 1 35 572434 4
e 1 44 381288 1
e 1 48 186321 1
e 1 56 359527 2
e 1 77 827028 2
.
.
.
t 100 3 1 484419
t 100 3 4 492769
t 100 4 1 524472
t 100 4 3 475673
t 100 5 1 483495
t 100 5 3 524725
t 100 5 4 510024
```

Lines beginning with "c" are descriptive comments and are ignored by the program. The line starting with "p" contains the word "edge", followed by the number of vertices n, edges m, colours k, and transfers T. There then follows m lines that begin with "e" and which specify the edges of the problem instance. These give the start node, end node, edge weight and edge colour respectively. This is then followed by T lines starting with the letter "t". These give the vertex of the transfer, the in-colour, the out-colour, and the weight (penalty) of the transfer respectively. Note that vertices and colours are labelled from one upwards.

Log file.

In addition to any output written to the console, each time the algorithm terminates a single line of information is also written to the file **resLog.txt**. This contains the following information, separated by tabs:

- Number of vertices in the problem instance
- Number of colours in the problem instance
- Number of edges in the problem instance
- Algorithm Choice
- Was the problem instance checker used? (0, 1)
- If –single option was selected:
    - –Single
    - Time to perform the graph expansion (if used)
    - Source node used for the shortest path algorithm
    - Time taken to run the shortest path algorithm
- Else
    - –APSP
    - Time to perform graph expansion (if used)

    - Number x of different vertices used as source vertices
    - Minimum time to produce the shortest path tree across all x sources considered
    - Maximum time to produce the shortest path tree across all x sources considered
    - Mean of these x values
    - Median of these x values
    - Standard deviation of these x values
    - Sum of these x values.
- Overall runtime (including time spent reading/checking the instances, etc.)

Copyright notice

Please direct any queries/comments to Rhyd Lewis: web: www.rhydLewis.eu, email: LewisR9@cf.ac.uk

R. Lewis (Last updated Monday, 26 October 2020)