Efficient Compression of Large Route Datasets using Pattern Matching

T2

Kishan Patel

Supervisor: Dr. Panagiotis Angeloudis

Department of Civil & Environmental Engineering, Imperial College London

Imperial College London

INTRODUCTION

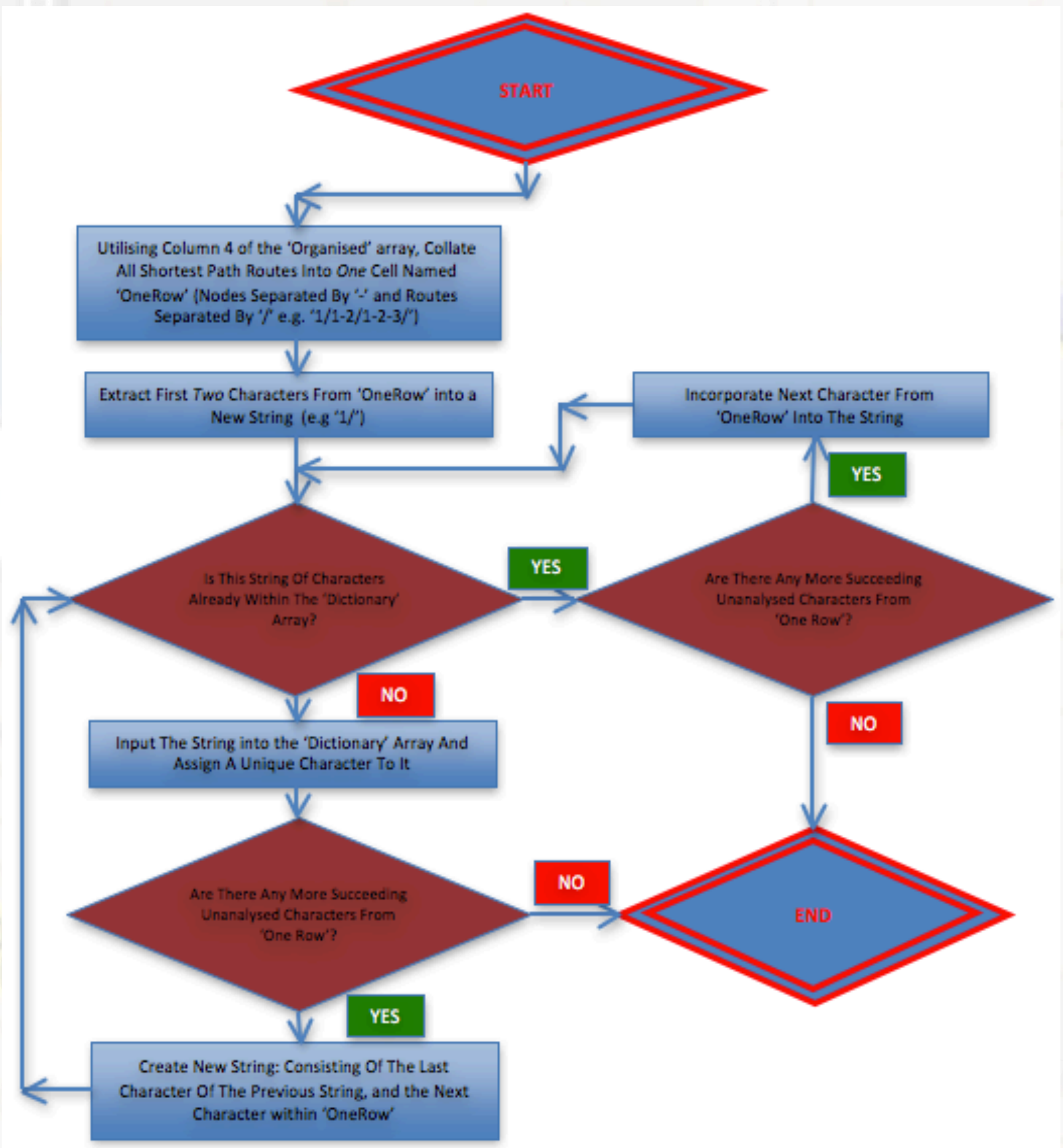
Transport users may require the fastest route between an origin and destination, which would involve the use of a shortest path algorithm, such as Dijkstra's. Given a sizeable number of travellers seeking routes at one time-point, 'on-demand' path generation would become infeasible, as algorithmic runtime will create delays within a queue of route requests. This research looks into ways to retrieve information to the user 'on-demand' after dataset compression techniques have been applied, given that all routes within a network are generated beforehand. These algorithms can enable a reduction in processing requirements and computer storage space within embedded/mobile devices, whilst simultaneously providing useful information quickly through decompression procedures.

EXISTING COMPRESSION METHODS

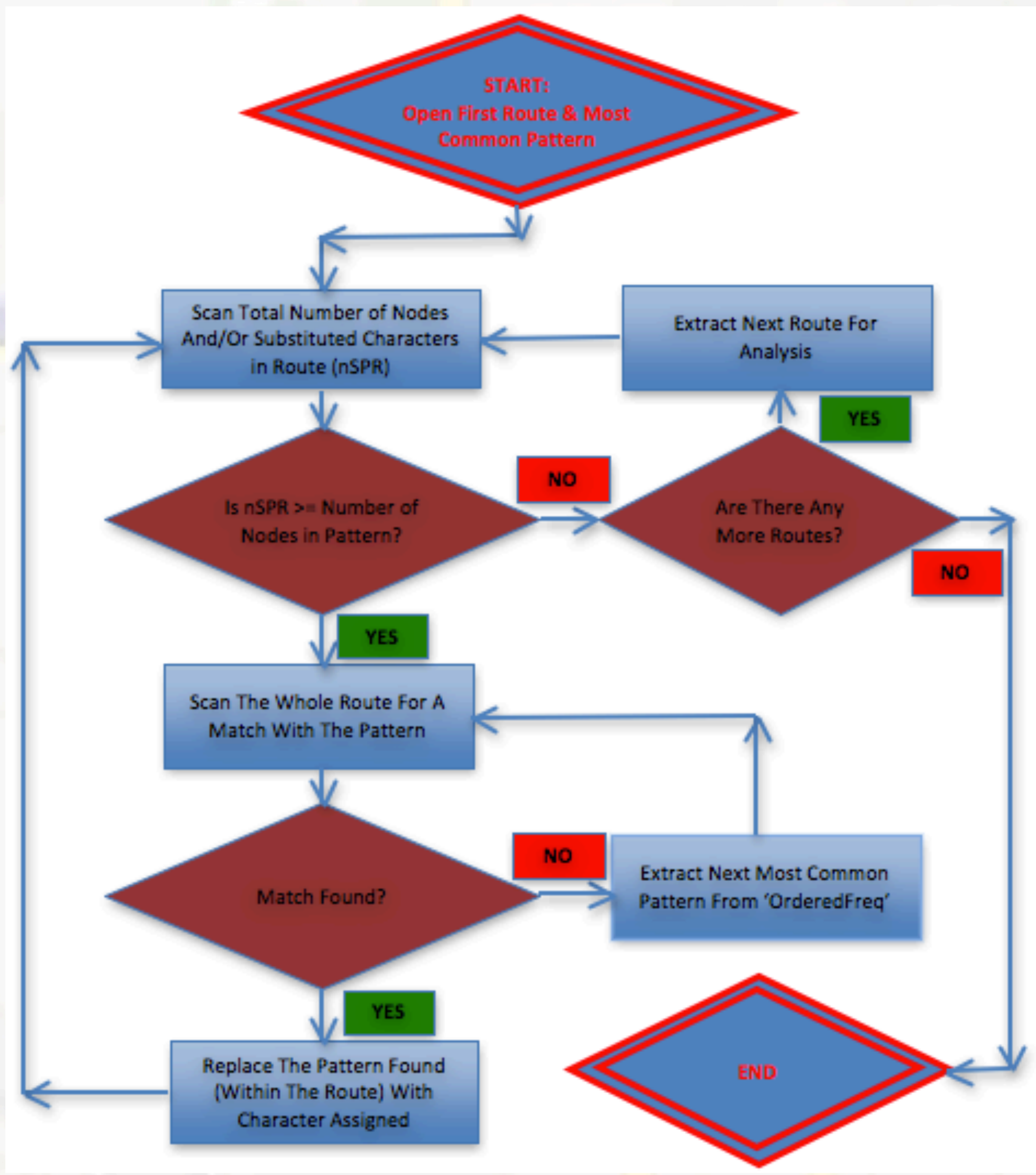
A list of compression methods were initially read over and understood, to account their suitability with this research. These included the Discrete Cosine Transform (DCT), Principal Component Analysis (PCA) and CUR Matrix Decomposition. These methods were all found not to be suitable, due to there uniqueness around image/audio compression. However, the LZW Algorithm was discovered to be highly applicable, whereby the algorithm itself 'slices' data sequentially, and references it into a dictionary.

METHODOLOGY

LZW Algorithm



Simple Substitution Algorithm (SSA)



COMPRESSION MEASUREMENTS

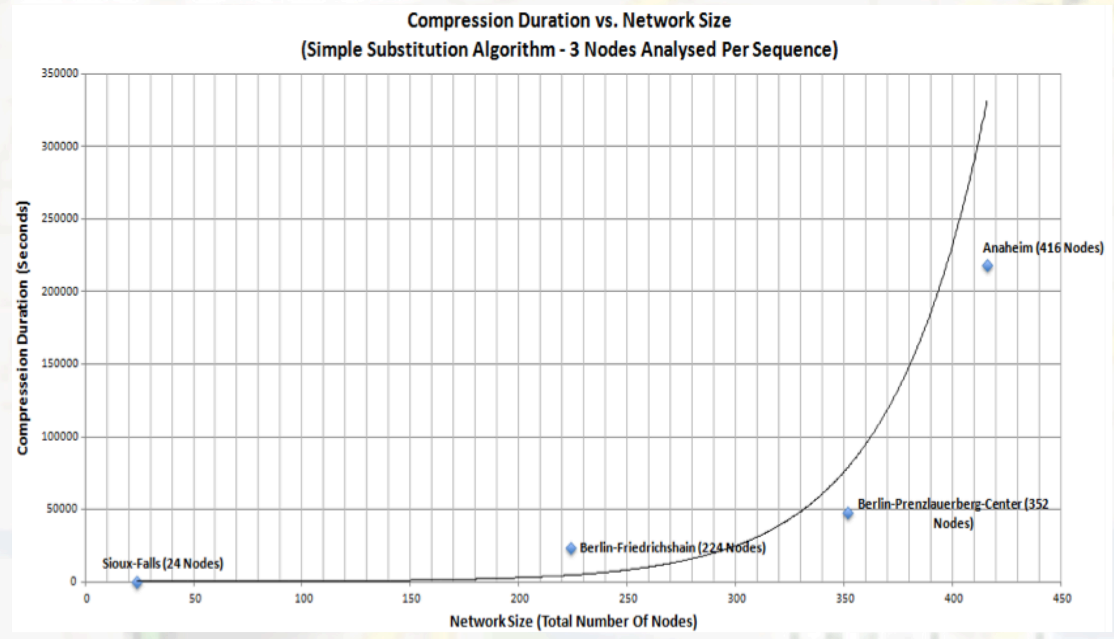
$$CR = \frac{\textit{Number of data points in original dataset}}{\textit{Number of data points in compressed dataset}}$$

$$ALTCR = \frac{\textit{Number of data points in compressed dataset}}{\textit{Number of data points in original dataset}}$$

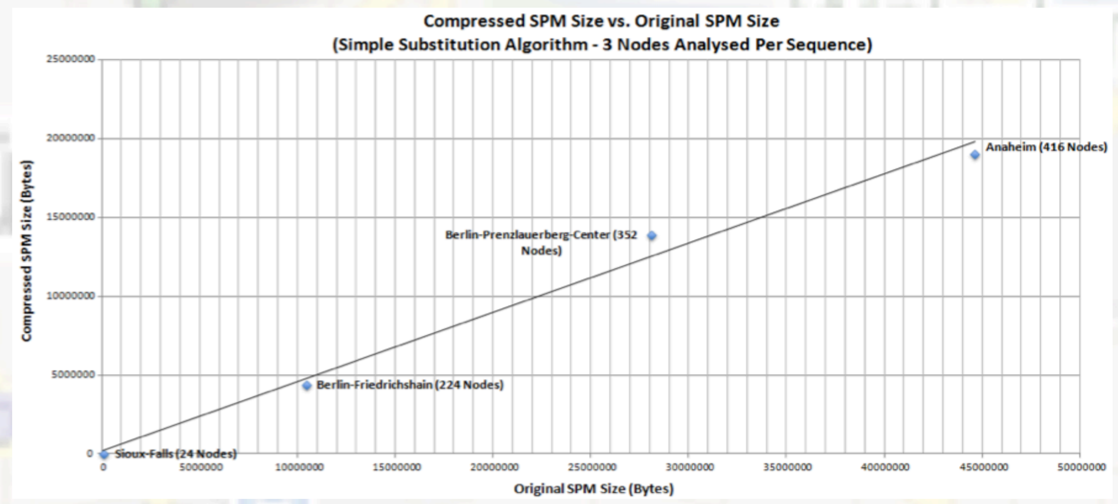
$$EDC\ (\%) = 100(1 - ALTCR) = 100\left(1 - \frac{1}{CR}\right)$$

RESULTS

SSA: Compression Duration Results		Compression Duration (Seconds)		
Network Name	Total Number of Nodes	3 Nodes Analysed Per Sequence	4 Nodes Analysed Per Sequence	5 Nodes Analysed Per Sequence
Sioux-Falls	24	25.046	12.838	7.718
Berlin-Friedrichshain	224	23,565	29,281	30,587
Berlin-Prenzlauerberg-Center	352	47,753	58,105	64,651
Anaheim	416	217,980	530,701	750,778



SSA: Compression Size Results			Compressed SPM Size (Bytes)		
Network Name	Total Number of Nodes	Original SPM Size (Bytes)	3 Nodes Analysed Per Sequence	4 Nodes Analysed Per Sequence	5 Nodes Analysed Per Sequence
Sioux-Falls	24	80,232	13,824	13,824	16,128
Berlin-Friedrichshain	224	10,453,292	4,415,488	5,318,656	4,816,896
Berlin-Prenzlauerberg-Center	352	28,083,114	13,877,248	14,372,864	14,372,864
Anaheim	416	44,625,714	19,036,160	20,168,648	20,057,334



LZW Algorithm: Compression Size Results & Analysis								
Network Name	Total Number of Nodes	Original SPM Size (Bytes)	Compressed SPM Size (Bytes)	Compression Ratio (CR)	Alternative Compression Ratio (ALTCR)	Extent of Data Compression (EDC [%])	Time Take To Generate 500 Routes (Seconds)	Average Decoding Speed For 500 Routes (Seconds)
Sioux-Falls	24	80,232	2,934	27.346	0.037	96.343	28.306	89.977
Berlin-Friedrichshain	224	10,453,292	1,690,566	6.183	0.162	83.827	45.211	372.616
Berlin-Prenzlauerberg-Center	352	28,083,114	6,199,466	4.530	0.221	77.925	49.661	503.765
Anaheim	416	44,625,714	11,580,422	3.854	0.260	74.050	51.798	645.950

LZW Algorithm: Compression & Decompression Duration Results				
Network Name	Total Number of Nodes	LZW Compression Duration (Seconds)	LZW Original Decompression Duration (Seconds)	LZW Alternate Decompression Duration (Seconds)
Sioux-Falls	24	43.233	10.601	0.052
Berlin-Friedrichshain	224	531,082.855	45,317.720	4.424
Berlin-Prenzlauerberg-Center	352	1,257,058.427	139,893.669	3.997
Anaheim	416	2,035,121.435	278,347.892	5.239

DISCUSSION & CONCLUSION

Through the results shown, it can be seen that:

•LZW Algorithm achieves a much higher 'extent of data compression' compared to the SSA

•LZW Algorithm takes a much longer time to complete than the SSA

The reasons for this include the nature of the LZW Algorithm: taking each small 'slice' of data and checking if it has been referenced within a dictionary. Ultimately, the LZW Algorithm achieves better compression because all sliced data has a corresponding substituted character, unlike the SSA which contains a mixture of substituted characters and 'leftover' nodes. An important note is that processing power of the computer also has an effect upon the compression duration results. Conclusively, both algorithms achieved a substantial amount of compression for the networks analysed. The implications could mean that significant compression could allow the storage of shortest path route data within embedded/mobile devices that would not need an 'online' internet connection (hence function 'offline'). The most interesting aspects of this study include how the format of data (saved within MATLAB) can drastically affect the number of bytes utilised for storage space. Information saved in 'char' format took up far less space than information saved within 'cell' or 'numeric' format.

ACKNOWLEDGEMENTS

I would like to thank Dr. Angeloudis and Nils Goldbeck for their help and support throughout this project.