Heuristically Driven Random Walks Across Large Scale Graphs

A. C. Olden, G. E. Taylor

[1]Faculty of Advanced Technology
University Of Glamorgan
Wales, CF15 7NS
Email:aolden@glam.ac.uk
Email:getaylor@glam.ac.uk

1. Introduction

The random walk is an example of application which, when considered can be seen to have foundations in a number of every day applications. (Aldious and Fill, 1996) cite the example of a chessboard where knight is moved at random to legal positions on the board.

Given a graph (G) and a starting point (s), a neighbour is selected at random and a move is made to those locations, upon which another neighbor is selected etc. until a pre-selected point is reached. This seemingly random selection of points making up this path is a *random walk* across the graph. Where the edges of the graph are weighted then the random walk becomes increasingly similar to the Markov Chain.

Increasingly, there are many graph based applications where more than a single criteria is considered for optimization. Examples can be seen in (Mooney, 2004) and (Chakraborty et al., 2005) where evolutionary algorithms (Zitzler, 1999) are applied to multi objective route selections. (Gendreau et al., 1994) applies the tabu search (Glover et al., 1993) to the vehicle routing problems. One of the key requirements of these and other optimization techniques is the generation of at least one (the tabu search) or more (evolutionary algorithms) initial randomly generated candidate solutions.

1.1 Basic notions

Let G = (V,E) be a connected graph with n nodes and m edges. The degree of any node on the graph is the incident of edges on the node. Spatially embedded graphs can be considered abstractions of real world networks. Within a spatially embedded network or graph coordinate information is stored alongside node and arc information. For instance the use of coordinate information across a real world road network allows an accurate distance measurement to be calculated between two connected nodes without the need to have such stored. The presence of the coordinate information allows such details to be inferred.

2. Random & Real World Graphs

The traditional approaches to the solution of graph based problems have been based upon randomly generated graph based structures. The work of (Cherkassky et al., 1994) for instance resulted in a series of random graph generators and solvers. This work is considered to be one of the most comprehensive within the field of shortest path analysis; a simple search of CiteSeer (2007) will result in approximately 97 citations for that work.

The use of such graphs however is not without problems. The majority of such generators fail to pick up upon the human tendency to cluster together in towns and city. There is a clear difference between random network and a real life human based network. Unsurprisingly, various algorithms react differently. The work of (Zhan, 2001) suggests that the results of previous studies into the shortest path problem may be misleading when applied to real world networks. They do however allow for the basic examination and completeness of an algorithm prior to further real world testing. Figure xxx presents a view of a random graph together with an example of how that graph may differ from a similar sized real world graph. Historically the key difference has been seen in the underlying size of the graphs. Table 1 presents a subset of graphs used in the recent 9th DIMACS Challenge on shortest path analysis. Whereas (Zhan and Noon, 2000) suggested that networks may consist of hundreds or even thousands of nodes. Table 1 demonstrates how modern networks are based upon millions of nodes. Figures 1 and 2 show how the basic structure of a graph can differ between real and randomly generated graphs. The key difference can be seen in that a real world graph will often contain many dead ends (nodes of degree 1). Random generators however have a tendency to produce more “connected” graphs.

Area	Edges	Nodes
Great Lakes	2,758,119	6,885,658
California & Nevada	1,890,815	4,657,742
North East US	1,524,453	3,897,636
North West US	1,207,945	2,840,208
Florida	1,070,376	2,712,798
Colorado	435,666	1,057,066
SF Bay Area	321,270	800,172
New York City	264,346	733,846

Table 1.Coverage Areas with Graph Sizes

3. Random Walking

Algorithm 1 presents a view of the traditional random walk across a graph. The algorithm is passed a copy of the graph structure, together with the start and end nodes. The

algorithm then enters a loop until a duplicate node is discovered or the target node reached. The algorithm adds the source node to the path when leaving the loop.

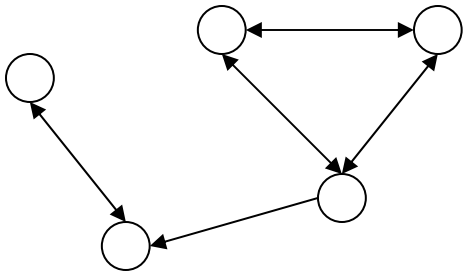

Figure 1. Example Of A Simple "Real World Graph (with Dead-end)

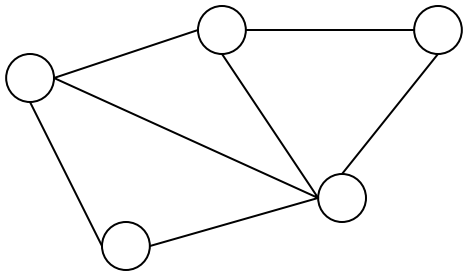

Figure 2. Example of A Complete Random Graph

Algortihm:1	Random Walk
Data	Graph G = (V,E) s=Start Node t = Destination

	Path = null
	Path = Path + s
	node=s
	While (node.Neighbours != t) && (path != cycles)
	node=Random(node.Neignbours)
	Path = Path + node
	End While
	
	Add (t) to path
	
	If (path.ContainsCycles)
	return null
	else
	return path
	End If

Algorithm 1. Traditional Random Walk

3.1 Experimental Design

Table 2 presents a series of graph sizes, ranging from 100 x 300 (nodes/edges) to 100,000 x 300,000 (nodes/edges) these graphs have been generated using the SPRAND random graph generator from Cherkassky et al (1994). A total of 10 graphs of each size have

been developed. The algorithm above has been applied to the graphs 250 times, with random selections for the source and target nodes. In effect a random work has been generated for each graph size 2,500 times. Repeat selections of source/target were disallowed.

Size	Nodes	Edges
Set 1	100	300
Set 2	1000	3000
Set 3	5000	15000
Set 4	10000	30000
Set 5	15000	45000
Set 6	100000	300000

Table 2. Randomly generated graph Sizes

3.2 Initial Results

Table 3 presents the result of the random walk algorithm on the graphs presented in Tables 2 (randomly generated graphs. The results show the Highest, Lowest and mean number of edges covered by the distinct path generated by the random walking algorithm.

Set	Minimum	Maximum	Mean (% Covered)
Set 1	3	26	7.7664 (7.8%)
Set 2	3	78	33.5144 (3.4%)
Set 3	5	256	52.9968 (1.1%)
Set 4	4	426	55.0944 (0.6%)
Set 5	5	497	71.5688 (0.5%)
Set 6	6	1343	302.2776 (0.3%)

Table 3 Network Coverage by Random Walking Across Various Graph Sizes

The results show that only a gradual increase in the network coverage is achieved through as the size of the graph increases the minimum number of nodes covered remains around the same. The highly connected nature of the graphs would logically play a major part in this.

4. Random Walking using Heuristics

The real world nature of the graphs under consideration as part of this study enables the development of heuristics to be developed for the random walk algorithm. The use of such heuristics is used to great effect in the A* shortest path algorithm (Ikeda and Imai, 1999). The graphs under considerations have either two (distance and transit time) or three (distance, transit time and road category) available. Any of which may be used to form a valid heuristic. The work of (Car, 1997) for example presents a route finding algorithm based on road hierarchy.. Others have suggested that people prefer to travel

cross the shortest path, in which either the transit time or the distance would be logical heuristics.

For the purposes of this work the distance between links is chosen as the appropriate metric for the heuristic. Algorithm 2 presents a view of a heuristic driven random walk where the nearest neighbour is selected. In order to preserve a reasonable amount of "randomness" the algorithm makes use of the degree of the node. If a current node has two edges than the one with the shortest distance is selected. If the node has three or more edges, then two are randomly selected and the edge with the shortest distance is considered.

Algortihm:2	Random Walk With Nearest Neighbour Heuristic
Data	Graph G = (V,E) s=Start Node t = Destination

	Path = null
	Path = Path + s
	node=s
	While (node.Neighbours != t) && (path != cycles)
	If node.Neighbours.Count = 1
	node=Random(node.Neignbours)
	else if node.Neigbours.Count = 2
	Select Closest Node
	Else if node.Neighbours.Count>3
	Select 2 Node At Random
	Select Closest Node
	End If
	
	Path = Path + node
	End While
	
	Add (t) to path
	
	If (path.ContainsCycles)
	return null
	else
	return path
	End If

Algorithm 3 is similar is to algorithm 2 in that it attempts to maintain a degree of "randomness". However, rather than selecting the closest node of the neighbours, the node selected is the node which has the closest distance to the target node. Such an algorithm could be used on real world graph where coordinate information for each node is available

5. Conclusions & Future Work

The results from indicates that as the size of the graph increase the percentage of coverage obtained by a random walk across a random, well connected graph remains stable. Further results for this work will evaluate the suitable of heuristic base algorithms for random walking.

The traditional approach to the random walk will often fail when applied to a real world graph. Randomly generated graphs fail to encapsulate certain notions of human behaviour, such as clustering or “dead ends”. Mechanisms need to be developed to deal will such cases, and to provide the random walk with the ability to backtrack through the graph. Such as ability of currently under development by the authors.

Algortihm:3	Random Walk With End Distance Heuristic
Data	Graph G = (V,E) s=Start Node t = Destination

```
Path = null
Path = Path + s
node=s
While (node.Neighbours != t) && (path != cycles)
        If node.Neighbours.Count = 1
                node=Random(node.Neignbours)
        else if node.Neigbours.Count = 2
                Calculate distance of Neighbours to t
                Select Closest Node To t
        Else if node.Neighbours.Count>3
                Select 2 Node At Random
                Calculate distance of Neighbours to t
                Select Closest Node To t
        End If

        Path = Path + node
End While

Add (t) to path

If (path.ContainsCycles)
      return null
else
      return path
End If
```

Further research is planned that will integrate the heuristic methods developed here into multi criteria evolutionary algorithm, tabu search and simulated annealing optimization algorithms where it is hoped that the use of heuristics will increase the rate at which suitable solutions are found whilst maintaining the diversity required in those solution.

7. References

ALDIOUS, D. & FILL, J. (1996) *Reversable Markov Chains And Random Walks On Graphs*, Book Draft.

CAR, A. (1997) Hierarachical Spatial Reasoning: Theorectical Consideration and Its Application to Modeling Wayfinding. *Department Of Geoinformation.* Technical University Of Vienna.

CHAKRABORTY, B., MAEDA, T. & CHAKRABORTY, G. (2005) Multiobjective route selection for car navigation system using genetic algorithm. *Proceedings of the 2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications.*

CHERKASSKY, B., GOLDBERG, A. & RADZIK, T. (1994) Shortest Path Algorithms: Theory and Experimental Evaluation. *Proceedings Of The 5th Annual ACM-SIAM sympoosium on Discrete algorithms.* Arlington, Virgina.

CITESEER(2007) http://citeseer.ist.psu.edu/

GENDREAU, M., HERTZ, A. & LAPORTE, G. (1994) A Tabu Search Heurtisitc For The Vehicle Routing Problem. *Management Science,* 40**,** 1276-1290.

GLOVER, F., TALLIARD, E. & DE-WERRA, D. (1993) A Users Guide To The Tabu SEarch. *Annals Of Operations Research,* 41**,** 3-28.

IKEDA, T. & IMAI, H. (1999) Enhanced A* Algortihms for multiple alighnments: optional alignments for several sequences and k-opt alignments for large cases. *Theorotical COmputer Science,* 210**,** 341-374.

MOONEY, P. (2004) Multicriteria Path Optimizartion On Graphs. *Department Of Computer Science.* National University Of Ireland Maynooth.

ZHAN, B. (2001) Three Shortest Path Algorithms on Real Road Networks: Datastrucures and Procedures. *Journel Of Geographic Information and Decison Analsyis,* 1**,** 69-82.

ZHAN, F. B. & NOON, C. E. (2000) A Comparision Between Label Setting and Label COrrecting Algorithms for Computing One-toOne shortest Paths. *Jounral Of Geographic Information and Decision Analysis,* 4**,** pp 1-11.

ZITZLER, E. (1999) Evolutionary Algorithms for Multiobjective Optimization:Methods and Applications. *Computer Engineering and Networks Laboratory.* Zurich, Swiss Federal Institute of Technology.