Journal of Theoretical and Applied Information Technology
10th January 2016. Vol.83. No.1
© 2005 - 2015 JATIT & LLS. All rights reserved·

ISSN: **1992-8645** www.jatit.org E-ISSN: **1817-3195**

COMPARING THE APPROACHES OF GRAPH-REDUCTION AND LANDMARK SHORTEST PATHS

[1]FAISAL KHAMAYSEH, [2]NABIL ARMAN

[1]Asstt Prof., Department of Computer Science and Engineering, Palestine Polytechnic University

[2]Prof., Department of Computer Science and Engineering, Palestine Polytechnic University

[2]Corresponding Author

E-mail: [1]faisal@ppu.edu, [2]narman@ppu.edu

ABSTRACT

We study the different recent improvements on shortest path algorithms. We suggest improvements to the classical path computing algorithms and we implement them using random generated graphs with various sizes. Many attempts exist for improving shortest path techniques. Computing shortest path is one of the most important and recent issues in combinatorial optimization, emergency routs, NoSQL data graph representation, road and public transportation networks, and other applications. One of the big obstacles in such real-word applications is the size of the graphs that motivates the attempts of enhancing the path-finding procedures. Given a weighted graph G={V,E,w}, a heuristic method to improve conventional shortest-path for a given source node <s> is provided. In this paper, we address the complexity of shortest paths in large graphs and we present a graph structure and enhancement process of finding the shortest path in the given graph. We study the current improving algorithms and highlights the improvements over the classical ones. We evaluate our implemented techniques using randomly generated weighted graphs. The main procedure is compressing the graph without losing the graph properties. We then, compare our technique with the approach of using landmark optimization. We discuss the performance, storage, error rate in our approach compared to landmark. Our experiments show that the new technique of graph compression performs with better speedup and no path mistakes compared to fast landmark approach which leads to high overhead in most situations.

Keywords: *Candidate Subgraphs, Graph Compression, Shortest-path, landmarks, Reverse Representation.*

1. INTRODUCTION

Finding shortest paths in graph based domains is demanding issue especially in recent digital life. It is difficult to imagine real-world applications such as online social networks (like Facebook, LinkedIn, MySpace), communication, transportation and routing networks, without applying graph algorithms. The huge number of graph nodes and links makes the existing path-finding algorithm incredible in terms of time complexity. Shortest path is needed in real-world applications such as communication, transportation and routing networks. In small non-dynamic graphs, pre-processing is very beneficial. However, pre-processing procedures require much space. Applications such as NoSQL graph representation of the data network rely basically on preprocessed preparation of data items. Twitter's real graph is hundreds of millions of nodes and relationships that makes it difficult to find a simple search path unless data is prepared in a graph that simplifies the query. The advantage of such pre-processing is to help finding the intended source-destination path while obtaining tangible cost reduction.

The processes of finding the shortest path over network topology is quite expensive. It is worthy to work towards improving existing classical algorithms such as the well-known Dijkstra's algorithm for finding shortest path [1]. Many attempts exist for improving shortest path techniques using different graph representation and constraints [2]-[11]. Pre-processed paths preparations provide better performance in path finding algorithms [6], [9],[11] and [13].

We study and describe some recent existing techniques for improving the computation of shortest path [9], [11] and [12]. Recent subgraph compression technique is very important which focuses on graph reduction in order to lower shortest path computing cost. This paper focuses on the comparison between the new improved algorithm using graph compression and the landmark technique of point-to-point shortest path.

Let G={V,E,w} be a directed weighted graph with vertex (node) $v_i \in \{V\}$, edge $e_i \in \{E\}$ and w(e) is the nonnegative weight function on every graph edge $e_i \in \{E\}$. The compressed graph CG={CV, CE, w} is generated from graph G using compression procedure. {CV} is the set of vertices in CG, {CE} is a set of edges formed using compression function, and w(e) is the positive weight function applied on CG edges. The compression function transforms G into new graph while preserving the graph properties. That is, the weight-sum of the shortest path from source <s> to target <t> in the compressed graph is the same as it is in original graph G. The Complexity of shortest path goes around O(|V|log|V|) or O(|V|log|E|). Example of such variations is the running time based on Fibonacci-heap min-priority queue which is O(|V|log|V|+|E|) for the nonnegative weighted graph with w(e) is nonnegative [14].

Some recent improving attempts commit to improve shortest path finding algorithms based on search procedure by imposing a constraint functions while ignoring the large number of irrelevant nodes [10], [15]-[20]. Some researchers have focused more on overcoming the network structure rather than the algorithm itself. Some improvements introduced to find the shortest path through graph partitioning. Researchers took an advantage of the graph features to improve the search such as network properties, and sentence fusion using dependency graph structure to acquire compression [15] and [18]. Some attempts focused on simplifying the detailed graph by clustering adjacent nodes and benefiting from components on the transit edges. The main feature in these recent improvements is the possibility of partitioning the graph into a set of components or clusters and in better representations with some constraints.

The technique of landmark selects set of nodes that cover the graph using a selection algorithm for better set of selected central nodes with highest *betweenness* centrality [19]. Finding the optimal set of nodes that covers the entire graph is NP-hard [20]. Landmark approach aims at decreasing the time needed to find the frequent shortest path finding [21] and [22]. This is not very beneficial in dynamic graphs especially when the real-world applications are critical such as aviation routs, emergency communications, medical data communication networks, to mention a few. Although computing shortest path is deeply studied, landmark approach as well as many other improvements do not guarantee the best procedure especially in current real-world huge networks.

2. STRUCTURE

Different graph representations have been used to handle the given graph. In graph compression technique, the sets of V nodes and E edges with corresponding positive weights are required to be represented in two matrices with maximum $|V|^2$ elements to represent the graph in normal and reverse representations [1]. For example, figure 1 depicts a random generated graph G (V,E,w) to be represented in the matrix structure which stores all nodes and edges in two dimensional array structure where each node N_i stores the name or index of previous node N_j and the weight w of the edge (N_i,N_j). Paths are represented in the matrix structure as a single row of consecutive nodes with every subpath branched in the next rows starting from next column. For example, starting from node N1, the linear parts are represented in separate rows with every subpath occupies a consequent row as shown in table 1, for example. It is required to construct a reverse matrix representation of the graph G. The main advantage of this representation is in finding all possible ancestor nodes starting from the destination node $<s_i>$. This step determines all nodes lead to the current node.

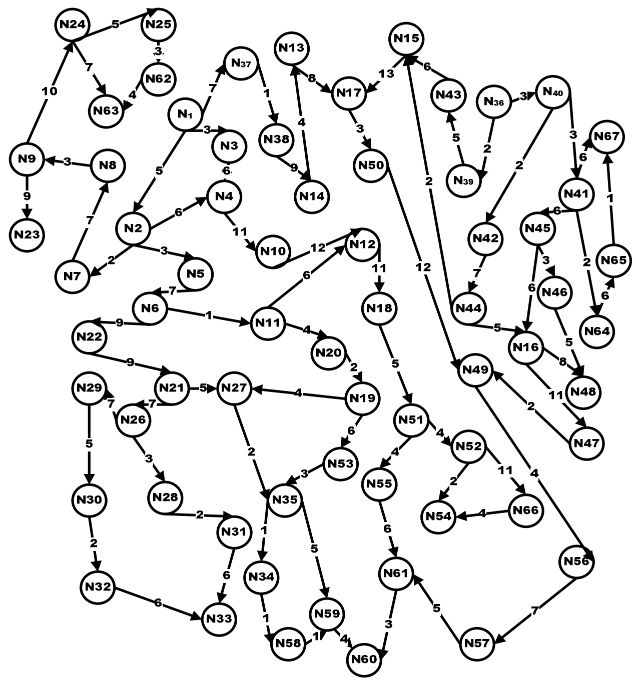


Figure 1. ph G = (V,E,w)

Journal of Theoretical and Applied Information Technology
10th January 2016. Vol.83. No.1
© 2005 - 2015 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The reverse pre-process structuring procedure limits the candidate subgraph where the algorithm is intended to work in. It is very useful in real-world path-finding applications to exclude irrelevant nodes or subgraphs in order to limit the scope. If we assume that the source node in the given graph G is N1 and the destination node is N59, the required candidate subgraph G'(V,E) is {N1, N2, N6, N11, N21, N19, N27, N35, N59} and hence, there no need to put effort in other parts of the graph. It is also required to represent the graph in linear representation for fast references of nodes as shown in table 3. For efficient implementation and to save storage, the matrix can also be represented as a linear array with |E| entries are:

{((0,0), N1), ((0,1), N2), ((0,2), N7), ((0,3), N8), ((0,4), N9), ((0,5), N24), ((0,6), N25), ((1,6), (0,8)), etc.}.

In this structuring technique, matrix stores all vertices showing all links and paths. The matrix is beneficial in finding all reachable vertices in the graph for given source and destination nodes. It takes only linear time to check path existence. We get an advantage of using this matrix in fast finding sub-paths of single in-degree and single out-degree where we apply compression technique.

TABLE I. GRAPH REPRESENTATION MATRIX

Node	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
0	*N1*	*N2*	*N7*	*N8*	*N9*	*N24*	*N25*	*N62*	*N63*
1							*(0,8)*		
2						*N25*			
3			*N5*	*N6*	*etc*				
etc					*etc*				

We also benefit from this structure in determining the graph roots or zero in-degree heads. These nodes appear in the first columns in the graph structure. The paths are represented in depth first search traversal order while common parts of the paths are stored only once using array indexing to avoid duplications of subpaths representation. When the reference of the vertex is in the form of (r_i, c_j), it means that the node is previously visited and previously represented. That is, the node (r_i, c_j) can be visited from more than one subpath. For example, if paths p1 is represented as vertices {N1,N2,…,Ni,…,Nn-1,Nn} and p2 is represented as {N1,N2,..,Ni,..,Nm}, P1 and P2 are paths in G, then the rest of path p2 is stored in the next row of path p1 starting from the column (i+1) with no elements in previous entries of the same row.

3. GRAPH REDUCTION PROCEDURE

Two major steps are required for reduction, compression and determining the candidate subgraph.

3.1 Graph Compression

The linear path that has no more than one in-degree and one out-degree in all inner nodes has to be reduced to only one edge with the corresponding weight sum. Graph reduction or compression is a task of producing new reduced graph G' produced from G by applying path compression. This procedure preserves the original graph properties including paths, weights, path heads and ends, source nodes and destinations, etc. The procedure finds all linear subpaths in which every node has no more than one out-degree and only one in-degree, adds up the subpath weights and transforms it to only one edge with its total weight. Figure 2 shows the new compressed graph G'={V,E',w'} after applying graph reduction procedures which reduces the number of nodes and edges. For example, the path p={N1,N37,N38, N14, N13, N17} with total weight = 29 in G is reduced to one edge <N1, N17> with weight = 29 preserving the same total weight of the original path. The benefit of this procedure is to help jump from head node of the linear path to its end node without calculation. The matrix structure of the graph aids in determining the linear subpaths.

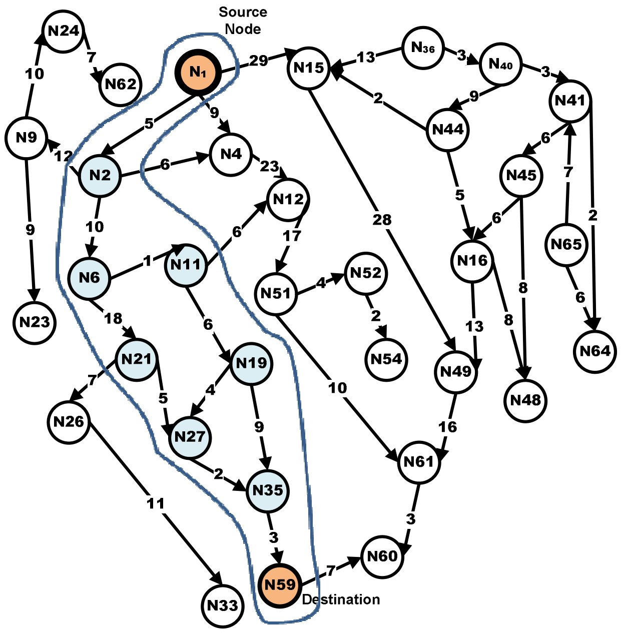


Figure 2. Compressed Graph

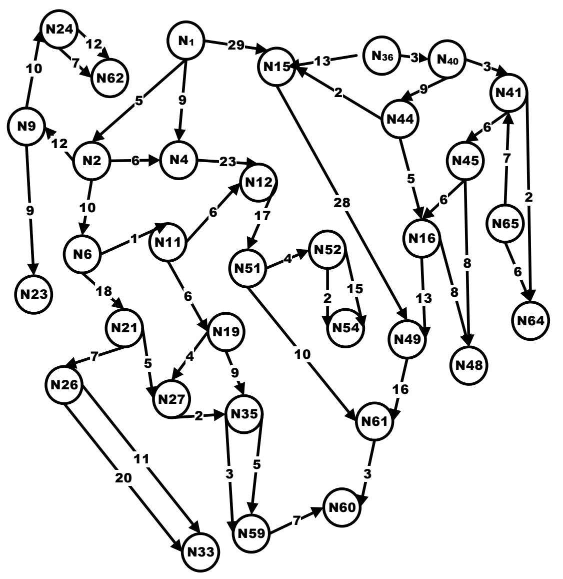
This reduction method performs well on sparse graph since it contains many linear subpaths with consequent nodes. The resultant reduced graph may include two direct edges between consecutive nodes such as N26 and N33. The second pass of the procedure removes the direct multiple edges between every two consecutive nodes. While this step is not necessary, it completes the reduction step as shown in figure 2 which creates figure 3.

3.2 Candidate Subgraph

When source and destination nodes are closed to each other or locate in a clear closed part of a big graph that can be bounded, working in the whole graph is therefore time consuming. Figure 3 shows a pictorial illustration of having candidate subgraph with reverse representation together with compression generated from the original given graph.

4. GRAPH COMPRESSION ALGORITHM

Normal structure, reverse matrix and linear representation are required for the compression stage. Reverse matrix array is used to represent the graph G where ancestors and decedents of each node v_i are reversed to represent all vertices where vertex v_i is accessible from. The optimized procedure to compress the graph and determine the candidate subgraph is then takes place where huge irrelevant parts of the graph may be excluded.



Graph G={V,E,w}

Compression technique performs path reduction using reverse matrix in the first place. The node structure includes the current vertex, minimum distance, and predecessor node. This aids in determining the direct path with its corresponding 0/1 (unmark/mark) flag of each node, the goal node, and accumulated weights. Then the algorithm saves the direct paths for nodes v_i to v_j where out-degree and in-degree of each node on this direct path equals to 1.

4.1 Compression General Steps

- Selects node N_i as a path head or parent.
- Calculates the weight sum of all consequent nodes in the same row in the original graph matrix with out-degree is not more than 1.
- Find the goal-node, which is the first node along the direct path with out-degree is other than 1.
- Save the direct path between the parent and the goal-node and the accumulated weight in the new compressed matrix.

4.2 Properties

The result of the above steps is a new reduced graph with no linear subpaths, that is, every node N_i has at least one the following properties:

- Selected source node of the shortest path.
- Selected destination of the shortest path.
- Graph root (path head): any node with in-degree equals to zero. These roots appear in column zero of the graph matrix.
- Path end: every node with no out-degree which appear as last node in each row.
- In-degree > 1in which more than one path leading to the N_i.
- Out-degree > 1 in which the node N_i appears in more than one path.

4.3 Constructing Reverse Matrix

To exclude irrelevant graph parts or nodes, we apply the selection method to determine that candidate subgraph G' with given source and destination nodes. In this step, the algorithm constructs the reverse matrix representing the graph rooted with all graph destinations to form the candidate subgraph. Candidate subgraph is determined by tracing the reverse matrix starting from the destination and going back through the matrix and marking only the candidate nodes along the backward edges only. This procedure is also possible in different ways as preferred by the programmer, e.g., copying the candidate nodes to a different reduced matrix, having mark-flag in the node structure, or by changing the weights of the excluded nodes to infinite value. After marking the visited (candidate) nodes in the subgraph, and starting from the source node, the algorithm checks the path existence in just array-row direct access time. The algorithm directly selects the direct path in the case it exists, otherwise it adds all neighbor edges of the current node by visiting all nodes listed in the next column in a breadth fashion.

The main saving of the this algorithm is to find the shortest path in the candidate part of the compressed graph excluding all nodes that do not lead to destination which limits the search to take less time than searching the entire graph. This new efficient procedure clearly reduces the search effort compared to the functionality of known conventional algorithms especially when applied on real-world problems such as communication and data networks.

Algorithm: Finding Shortest Path Using Graph Reduction (graph, mark, reverse-matrix)

```
1:  initialize mark-matrix to unmark (false
    mark)
2:  select destination vertex $t$
3:  construct a reverse-matrix
4:  start at destination vertex $t$ in reverse-
    matrix
5:  for every linear subpath from $v_i$ to $v_j$
6:          transform all inner nodes to <$v_i$,$v_j$>
            as a single edge with total weight $w_{ij}$
7:  mark all ancestor nodes to mark (true
    mark)
8:  dist= FindShortest(graph-matrix, mark-
    matrix, s) //For the selected source node s
```

Function:FindShortest(GraphMatrix, MarkMatrix,s)

```
1: for each vertex v in Graph Matrix
2:      initialize distance of v to infinity;
3:      Initialize predecessor of v to undefined
4: end for
5: for every node starting from path-heads to path-
        ends in Graph Matrix // compression
6:      add up weights of all consecutive vertices
        vi's with in-degree and out-degree no more
        than 1
7:      store total weight as an a single edge for $v_{i+1}$
8: end for
9: dist[s] := 0 ;
10:  MarkQ = set of Marked nodes in Graph Matrix
     ordered in dfs visit
11:  starting from s, select smallest nodes with
     distance using marked nodes in MarkQ only
12:  Update dist
13:  return dist
```

5. PERFORMANCE

In this paper we have studied the algorithm of reducing the graph and finding the candidate suitable subgraph to find the shortest path. The fundamental objective is to come up with better performance comparing with conventional naïve shortest path algorithms and also comparing with some recent improvements such as using landmark method. The algorithm finds the shortest path SP(s,t) between two given vertices <s> and <t> in a directed weighted graph G={V,E,w}. It obviously reduces the original graph G. This improvement reduces the number of edges |E| to the new number of edges |E'| and the number of vertices |V| to |V'| which lowers the cost significantly. This reduction results in more benefits and improved tangible performance if applied on sparse graph than applying it on dense one.

Practically, we applied the given approach on a random generated graph G={V,E,w} where the algorithm reduces the graph G to G'(E',V') in about 51% with significant saving about 49% of the total number of edges in G, for example.

In the algorithms of finding shortest paths, it is required to have $O((|V|+|E|)\log |V|)$ for each path finding applying the classical algorithms making the cost extremely high. This classical procedure is almost impossible in many real-world applications such as in communication networks. Comparing with late shortest path improvement attempts, the recent algorithm using graph compression [9] introduces more tangible performance.

6. EXPERIMENT AND EVALUATION

This paper presents an experimental study of our improving algorithm. We investigate the different graph representations, graph reduction and performance results in comparison with landmark approach. The graph compression approach provides evidence that the proposed heuristic outperforms the conventional algorithms. The performance of the algorithm shows a considerable

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

saving in the cost of random generated graphs with different sizes. The experiment shows a significant saving in performance when applied on dense graphs and more on sparse ones in most of the trials. Average performance of applying the new improving approach on set of random graphs with different density degrees is shown in figure 4. This procedure shows that the algorithm with the compression approach outperforms the improved procedures using landmark method.

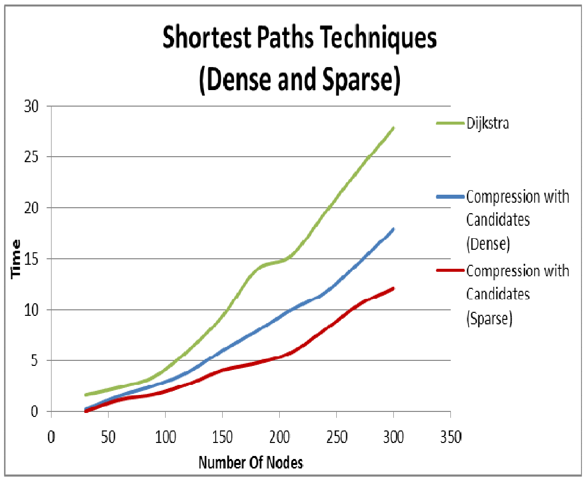


Figure 3. Performance of Shortest Path using Compression with Candidates Heuristics on Sparse and Desnse Graphs

7. GRAPH COMPRESSION USING CANDIDATE SUBGRAPH HEURISTIC VS LANDMARK APPROACH

The landmark based algorithm store complete shortest paths with reference to each landmark vertex [23]. Finding shortest path using landmark approach is fast when source node is one of the landmark vertices. The best landmark selection is the set that covers all graph nodes where it works not faster than naïve shortest path. In real random selection of source nodes, landmark approach calculates the shortest path between the source and the closest landmark point. The risk, is that shortest path between source and destination may not go through any landmark vertices. On the other hand, pre-processed shortest paths via landmark requires large storage on each landmark vertex.

Although landmark approach adds some tangible improvements on shortest path finding, there is a considerable overhead when applied on nodes out of landmark set, hence, larger k leads to less errors. The reason is that large k covers the graph in more small-subgraphs covering more source nodes. To avoid this case, compression approach performs in a better way. Figure 5 shows the same graph G with landmark selection. After structuring the graph and storing the shortest path trees, landmark approach finds the shortest path in high speed, however extra calculations needed if source point *s* exist out of selected landmark set of nodes as illustrated in figure 6. Practically and in the real world application, it is almost impossible to calculate and store the shortest path trees as shown in figure 7. The performance figures of applying landmark approach on set of random graphs with different density degrees are shown in figure 7 and figure 8 where the compression technique introduces valuable cost saving.

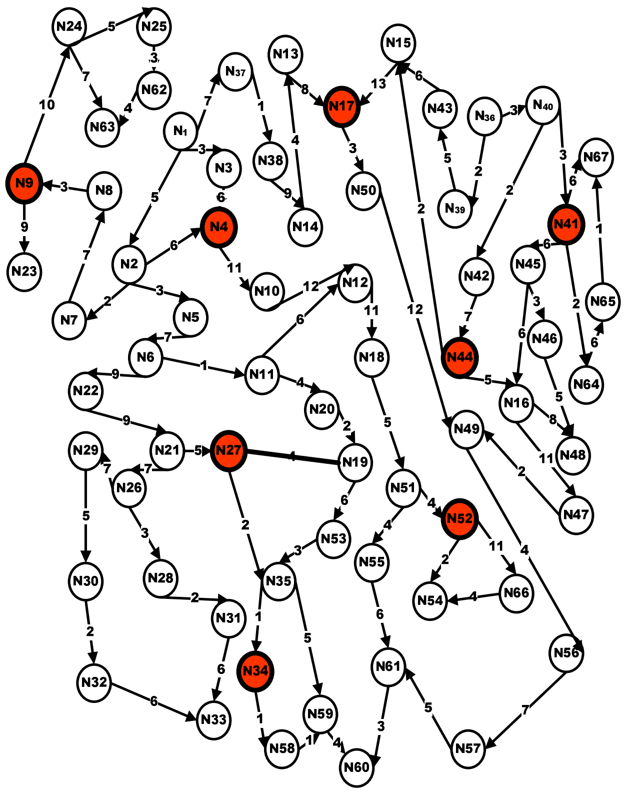


Figure 4. Landmark Graph

Compression approach requires less space and keeps preserving the properties of the original graph. Therefore, compression approach is error free. Our Experiments show a considerable saving in performance in dense graphs and more in sparse ones in most of the trials.

Landmark procedure depends on preprocessing in computing the shortest paths trees between nodes with reference to landmarks. Although the landmark approach do not guarantee the intended approximation, it has been approved to perform well in suitable context [23]. The main improvements of using landmark algorithm is to calculate shortest paths between every landmark node to each other node in the graph as illustrated in figure 6 and figure 7. The other challenge is in the way of finding the best collection of landmark nodes to cover the graph.

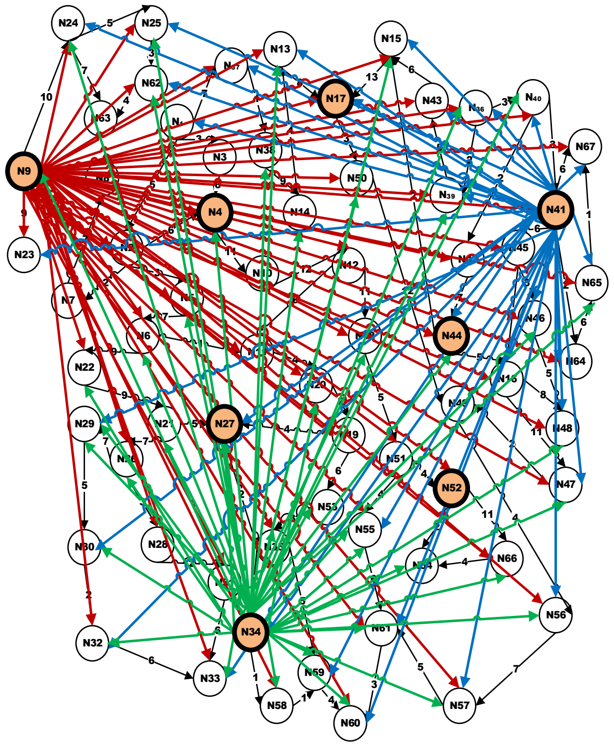


Figure 5. Graph G with 8 Landmark Points

The comparison results between the new improving algorithms show the performance in terms of running time units of the graph reduction with candidate subgraph and the landmark approach as shown in Table 2 and figure 7. These improving algorithms show better performance on different graph densities especially in sparse graphs. Shortest path computation on landmark nodes is fast, $SP(l_i,l_j)=c$, since the computation is previously calculated. The problem occurs when the given source points do not exist in selected landmark in which direct shortest path SP(s,t) requires less effort than going through landmarks.

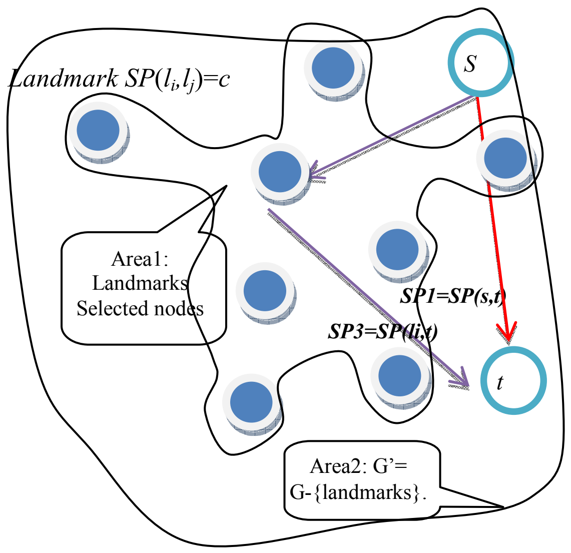


Figure 6. Landmark SP Procedure

The distance computation of SP(s,t) is performed in one of the following choices: of SP1 or SP. SP1(s,t) is the direct calculation without recognizing landmarks. SP(s,t) using $SP2(s,L_i)+SP3(L_i,t)$ in which landmark is used.

Although landmark approach is recognizable improvement and very beneficial, SP1(s,t) could be less or equal to SP2+SP3 in terms of distance based on the selected nodes:

$$SP1(s,t)<=SP2(s,l_i)+SP3(l_i,t). \qquad (1)$$

TABLE II. COMPARISON OF GRAPH COMPRESSION WITH CANDIDATES VS LANDMARK APPROACHES

Out Degree	Dijkstra	Candidate subgraph	Compression with candidate	k=15%*N	Landmark k=10%*N	k=5%*N
1	4.99	1.16	0.73	6.59	3.49	2.19
2	7.88	2.40	1.40	8.80	7.20	4.19
3	9.12	2.96	1.49	10.79	8.89	4.48
4	10.70	3.16	1.69	12.49	9.47	5.07
5	12.13	3.37	2.85	13.18	10.11	8.56

8. RESULTS

The performance of both algorithms are shown in table 2. The comparison figures between the heuristic of shortest path computation using graph compression with applied selection of candidate subgraph and the approach of using landmark show that the graph compression by excluding irrelevant nodes that do not connect source with destination outperforms landmark approach in different graph densities and sizes, and therefore outperforms the classical naïve procedure. The overhead in landmark approach is basically embodiment in the huge computations of shortest path trees that is concentrated in the selected set of landmark nodes.

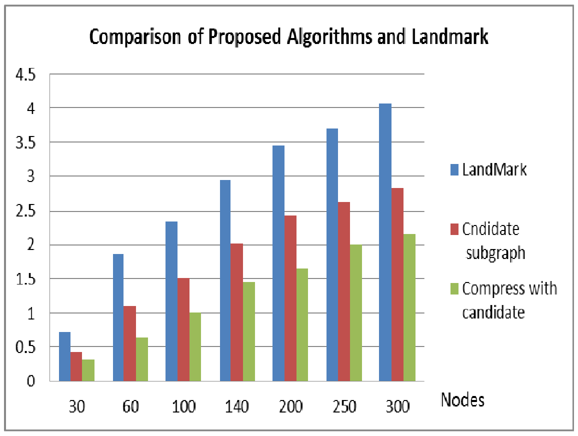


Figure 7. Comparison between Shortest Path Techniques and Lankmark Approach

For a graph G={V,E,w}, the new compression technique benefits from different graph representations and preprocessed preparation to generate a reduced graph G’ while preserving the properties of the original graph. The new approach improves finding shortest path using the candidate subgraph along with compression technique. The statistical figures result in better performance when applying the graph compression together with the exclusion of the irrelevant nodes using candidate subgraphs approach.

These experimental statistical figures show the average performance of applying the new approach on set of random graphs with different density degrees. That is, the applied new procedure for improving shortest path finding in weighted graphs outperforms the classical procedures and outperforms the approach of using landmark technique.

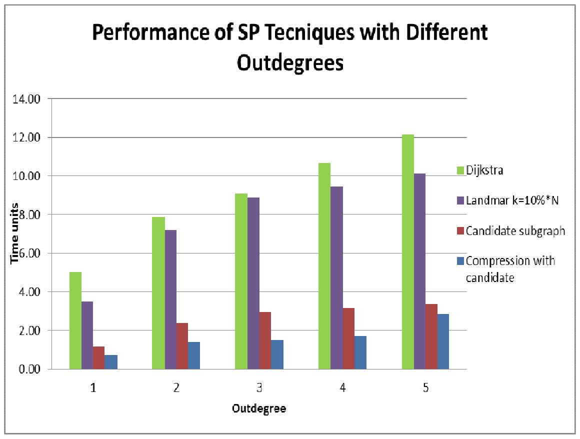


Figure 8. Performance of Shoprtest Path Techniques

9. CONCLUSION

In this paper we present and discuss the heuristic approach of finding shortest path using compression with candidate subgraph and the approach of using landmark. Given a weighted directed graph G={V,E,w}, an efficient and improved algorithm for finding shortest paths between a given source <s> and destination <t> using candidate subgraph along with graph compression is implemented and discussed.

The candidate and compression procedures are applied on random generated weighted directed graphs that vary in sizes and range from sparse to dense. The approach of using landmark to find shortest path is applied on the same graphs and compared in terms of performance. In the practical phase, we found that the algorithm of computing shortest path using graph compression with candidate subgraph technique outperforms the performance of landmark approach and other improving algorithms with the given properties and constraints. Graph compression along with candidate subgraph heuristic shows obvious improved performance on random generated sparse graphs. As a heuristic algorithm, the complexity will always be bounded by the complexity of known normal algorithms, i.e., it will not exceed $O((|V|+|E|)\log |V|)$ for each source <s> and each destination <t> while it shows tangible cost saving in the real practical applications such as in real communication and data networks.

With respect to previous improvements, we achieve notable improvements and tangible performance while preserving graph properties. The average performance of applying the graph compression approach on set of random graphs with different density degrees as sparse and dense is discussed. Landmark approach is discussed and practically applied on different graphs. A comparison between the different algorithms are established and discussed. The applied new procedure for improving shortest path finding in weighted random graphs outperforms both the classical procedure and the approach of using landmark technique especially when applied on real world applications.

ACKNOWLEDGEMENTS

This research is funded by The Scientific Research Council, Ministry of Education and Higher Education, State of Palestine under a project number of 01/12/2013, and Palestine Polytechnic University. The authors would like to thank the research assistants Ms. Walaa Nasereddin and Ms. Salma Dirbashi for their help in implementing the algorithms.

REFERENCES

[1] E. W. Dijkstra, "A note on Two Problems in Connexion with Graphs", *Numerische Mathematik 1*: 269 271. doi:10.1007/BF01386390.

[2] F. Zhang, A. Qiu, and Q. Li,"Improve on Dijkstra Shortest Path Algorithm for Huge Data". *Chinese academy of surveying and mapping*: China, 2005.

[3] F. Khamayseh and N. Arman,"*An Efficient Heuristic Shortest Path Algorithm Using Candidate Subgraphs*". International Conference on Intelligent Systems and Applications. Hammamet, Tunisia. 22-24 March, 2014.

[4] F. Simek and I. Simecek,"*Improvement of Shortest Path Algorithms through Graph Partitioning*". International Conference Presentation of Mathematics. Liberec, Czech Republic, 2011.

[5] H. N. Djidjev, G. E. Pantziou, and C. D. Zaroliagis,"Improved Algorithms for Dynamic Shortest Paths". *Algorithmica* (2000) 28: 367–389.

[6] J. B. Orlin, K. Kamesh Madduri, K. Subramani, and M. Williamson,"A faster algorithm for the single source shortest path problem with few distinct positive lengths". *J. of Discrete Algorithms*, 8, 2 (June 2010), 189-198

[7] L. Xiao, L. Chen, and J. Xiao, "A new algorithm for shortest path problem in large-scale graph". *Appl. Math*, 6(3), 657-663.

[8] F. Khamayseh and N. Arman."Improvement of Shortest-Path Algorithms Using Subgraphs' Heuristics", *Journal of Theoretical and Applied Information Technology*, 2015. Vol. 76, No.1.

[9] L. Yunpeng, J. Yichuan, Z. Yong. "A new Single-source shortest path algorithm for nonnegative weight graph", *Cornell university library*, retrieved May 9th, 2015 from http://arxiv.org/abs/1412.1870v6.

[10] N. Arman and F. Khamayseh, "A Path-Compression Approach for Improving Shortest-Path Algorithms", *International Journal of Electrical and Computer Engineering (IJECE)*, Vol.5, No.4, September 2015.

[11] Y. Huang, Q. Yi, and M. Shi, "*An Improved Dijkstra Shortest Path Algorithm*". Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013). Hangzhou, China, Paris: Atlantis Press, March 2013: 226-229.

[12] F. Khamayseh and N. Arman,"An Efficient Multiple Source Single Destination (MSSD) Heuristic Algorithm Using Nodes Exclusions", *International Journal of Soft Computing*, Vol. 10, No. 3, 2015.

[13] N. Arman, (2005). "*An Efficient Algorithm for Checking Path Existence Between Graph Vertices*". Proceedings of the 6th International Arab Conference on Information Technology (ACIT'2005), pp. 471-476, December 6-8, 2005, Al-Isra Private University, Amman, Jordan.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Dijkstra's Algorithm. Introduction to Algorithms", (Second ed.). Section 24.3: pp. 595–601. MIT Press and McGraw-Hill. ISBN 0-262-03293-7.

[15] K. Filippova. & M. Strube. "*Sentence fusion via dependency graph compression*". Proceeding of EMNLP-08, 2008, pp. 177–185.

[16] J. Zhang, J. Li, X. Fan, Z. Deng, "Research on Real-Time Optimal Path Algorithm of Urban Transport", *TELKOMNIKA Indonesian Journal of Electrical Engineering*, Vol.12, No.5, May 2014, pp. 3515 ~ 3520.

[17] W. Yahya1, A. Basuki2, J. Jiang. "The Extended Dijkstra's-based Load Balancing for OpenFlow Network", *International Journal of Electrical and Computer Engineering (IJECE),* Vol. 5, No. 2, April 2015, pp. 289~296.

[18] F. Katja, "*Multi-Sentence Compression: Finding Shortest Paths in Word Graphs*", Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 322–330, Beijing, August 2010.

[19] L. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977.

[20] W. Frank Takes and Walter A. Kosters, "*Adaptive Landmark Selection Strategies for Fast Shortest Path Computation in Large Real-World Graphs*", The Netherlands, Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014 IEEE/WIC/ACM International Joint Conferences on (Vol:1).

[21] F. Volodymyr, T. Konstantin and D. Marlon Dumas, "*Memory-Efficient Fast Shortest Path*" Estimation in Large Social Networks, Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media, Association for the Advancement of Artificial Intelligence, 2014.

[22] Q. Miao, C. Hong, C. Lijun, X. Y. Jeffrey, "*Approximate Shortest Distance Computing: A Query-Dependent Local Landmark Scheme*", Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, p.462-473, April 01-05, 2012.

[23] K. Tretyakov , A. Armas-Cervantes , L. García-Bañuelos , J. Vilo , M. Dumas, "*Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs*", Proceedings of the 20th ACM international conference on Information and knowledge management, October 24-28, 2011, Glasgow, Scotland, UK [doi>10.1145/2063576.2063834].

AUTHOR PROFILES:

Dr. Faisal Khamyseh is a Computer Science assistant professor. He received his BSc in Computer Information – Advanced Computer Careers, from Southern Illinois University, USA 1992, and MSc in Computer Science from same university in 1995, and his PhD in Computers and Information Systems from the College of Computers and Information, Helwan University, Egypt, in 2009. Currently working at Palestine Polytechnic University as instructor and head of Dept. of Information Technology and as instructor of MSc in Informatics. Dr. Khamayseh is a researcher in software engineering research unit at college of Information Technology and Computer Engineering. He is interested in Computer Algorithms, Software Engineering and E-learning.

Dr. Nabil Arman is a Computer Science professor at Palestine Polytechnic University. He received his BS in Computer Science with high honors from Yarmouk University, Jordan in 1990 and an MS in Computer Science from The American University of Washington, DC USA in 1997, and his PhD from the School of Information Technology and Engineering, George Mason University, Virginia, USA in 2000. At Palestine Polytechnic University, he worked as the MS Informatics Program Coordinator and the head of the Department of Mathematics and Computer Science. Currently, he is the Dean of the College of Information Technology and Computer Engineering. Dr. Arman is interested in Database and Knowledge-Base Systems, Algorithms, and Automated Software Engineering. He has published more than thirty refereed conference and journal papers.