Network Path Optimization Under Dynamic Conditions

Yaser Ali Enaya and Kalyanmoy Deb
Computational Optimization and Innovation (COIN) Laboratrory
Michigan State University, East Lansing, USA
{enayayas, kdeb}@msu.edu
http://www.egr.msu.edu/~kdeb/reports.shtml

COIN Report Number 2014003

***Abstract*:** Most network optimization problems are studied under a static scenario in which connectivity of the network and weights associated with the links of the networks are assumed to be fixed. However, in practice, they are likely to change with time and if the network is to be used over time under dynamic conditions, they need to be re-optimized as soon as there is a change. Since optimization process requires some finite time, there is a need for a efficient dynamic optimization strategy for solving such problems. In this study, we extend a previously proposed "Frozen-time" algorithm to network optimization by which new and optimized networks can be obtained in a computationally fast manner. We propose three different variations of the optimization strategies and show proof-of-principle simulation results on a 20-node network having 190 different source-destination paths. The results are interesting and suggest a viable further research.

Keywords: Network optimization, dynamic optimization, on-line optimization.

1. Introduction

Finding a number of optimal paths between a set of source and destination nodes in a network is one of the most important design issues that has a significant impact on a network's performance [5]. The shortest-path problem is a graph problem appears in many applications such as routing in a road network, routing data harvesting in sensor networks [18]. In real-life applications, there is an industrial demand for computing the shortest on dynamic large-scale network such as road network and sensor network whose edges are dynamically changed with the traffic for the first and in inefficiently working sensor for the second network [18]. Networks used for internet communication, road, etc. have common entities such as nodes, edges, and weights. In this study, we attempt to find the optimal paths for every source-destination pair in the network taking into a consideration the actual time of finding the optimal paths. In practice, network optimization problem is a dynamic one, simply because the weights associated with the edges often change with time. Also, certain connections may fail and the network connectivity itself can be changing with time. Since the computational time for finding optimal paths for all source-destination pairs increases exponentially with an increase in number of nodes in the network, in general, the dynamic network optimization problem becomes a computationally challenging task.

Finding optimal paths in a changing network has always been the goal, but the computational time needed to achieve an optimal network as soon as there is a change in the network is also an important matter, which has not received much attention. There is a clear trade-off between these two entities. If a large computational time is needed to find a new set of optimal paths in a network, by the time an algorithm is able to find the optimal paths, the network may have changed to a different one, which was not used at the start of the optimization. Thus, in real practice, one may have to settle down for a compromise time for optimization and a desired upper limit on inefficiency introduced due to non-optimal nature of network paths. The important question then to ask is "how much inefficiencies one should tolerate to still have a reasonably working network under dynamic conditions?". It is difficult to answer this question in the absence of any further trade-off information about the network and its change pattern. In this paper, we discuss a viable off-line optimization based "frozen-time" methodology to find such a compromise error limit on optimal paths. The procedure we adopt here is similar to a dynamic optimization methodology suggested by the second author earlier [8] in the context of a power dispatch problem and modify the method for network optimization.

Although the idea is used here for a single-objective dynamic network optimization problems, it can potentially be applied to multi-objective dynamic network optimization problems. Since the concept is practical, such a study is highly pragmatic and implementable. Moreover, the approach is potentially capable of handling faults or "jamming" in a network as a changed scenario and the proposed method will handle different kinds of changes or faults in a network in a unified manner.

In the reminder of the paper, we present a brief overview of existing studies of dynamic network optimization in Section 2. Thereafter, in Section 3, we present our proposed methodology and adaptation of the "frozen-time" algorithm for network optimization. Three different optimization methodologies are suggested as potential

optimization algorithms for finding all optimal paths in a network. Section 4 presents simulation results on a 20-node network with changes in its connection weights with time. Extensive results are shown with three different optimization methods. The procedure is choosing a compromise frozen-time from an analysis of the error versus frozen-time trade-off is presented. Finally, conclusions of this study and a number of immediate future extensions are discussed.

2. Existing Studies on Dynamic Network Optimization

The existing dynamic network optimization methods do not consider the weight changing aspect which can happen with time. However, there are many studies that consider deletion and insertion of edges and vertices after dynamic changes [2]. Also, the dynamic optimal path computation methods include answering queries for the dynamic connectivity problem, for example, a query "if there is exciting edge between two nodes in a graph, the answer will be True or False" [4]. The insertion can be divided in two types, random or non-random and in [4] they try to find the fastest algorithms for both types. In [16], they classified a dynamic shortest-path algorithm to a batch and non-batch algorithms which mean that if the dynamic algorithm is able to handle graph changes that consist of multiple edge updates at a time, then it is a batch algorithm and if it is not, then it is a non-batch algorithm. Therefore, according to this classification, our approach in this paper is a batch algorithm. But we use a different classification that is more related to applications in real life.

Based on the growing literature in dynamic network optimization, we classify the most important dynamic path application methods into three categories. Due to multi-criterion nature of network optimization problems, some studies choose multiple objectives, but constitutes a single objective by combining them in certain manner. First, a neural network is used to find the functional relationship between the desired responses and control factor values and then either a simulated annealing or a genetic algorithm is applied to determine an optimal combination of control factors [6]. To avoid getting stuck in a local minimum, authors propose integrating a hierarchical genetic algorithm and a multiple objective evolutionary algorithm to optimize the dynamic parameter design problem. Second, using a dynamic graph for wireless networks (such as in a mobile Ad-hoc network where the nodes are mobile), the network topology may change rapidly and unpredictably over time [14, 15]. Dynamic graph solutions are used to solve such problems. In the third category, simplified versions of network connections are continuously being added to make the overall network bigger and bigger to have an overall effect of different parts of the network. Social networks is a real example of this category. Social networks contain information about the relations between people or entities. With continuously adding nodes and edges, the social network graphs become bigger and bigger [10]. In [11, 12], a dynamic algorithm is applied to solve the dynamic traffic management (DTM) problem. Because of spatial correlations DTM has predefined set that may not contain well-performing strategies. To select a DTM strategy, they should be performed on several network measures, including externalities. In water distribution network the dynamic design is used which it is capable of introducing cheaper and more reliable long-term designs in comparison with normal initial design and rehabilitation [13].

Although different network application problems address different aspects of the changing aspects with time, none of the existing studies suggest any unified approach to different vagaries of the network changes. In this paper, we propose a methodology that has the potential to be one such unified approach.

3. Proposed Methodology

3.1. Frozen-Time Algorithm

Many search and optimization problems in practice change with time and therefore must be treated as on-line (or dynamic) optimization problems [8]. The change usually occurs in some parameters associated with the objective function, constraint functions, or in its variable bounds. In the case of a network optimization for finding optimal paths for all possible source-destination nodes, the change in one or more weights in the network or addition-deletion of some connections or any other minor changes with time will affect some of these optimal paths. With more and significant changes, more such source-destination pairs will get affected every time there is a change, hence requiring more computational time to find optimal paths for all possible source-destination pairs.

Ideally, such a dynamic network optimization problem must be solved instantly at every time instant, or whenever there is a change. However, an optimization task requires a *finite* amount of computational time to arrive a solution that is reasonably close to the true optimum. In such dynamic problems, there are two time frames which are intertwined: (i) computational time involved in the optimization process in arriving at a solution, and (ii) the real time in which the problem undergoes a change. Here, we shall assume equivalence of both time frames and any time spent in one frame affects the same amount in the other time frame. The relevance and accuracy of the obtained optimum in the current context largely depends on the rate at which the problem changes with time [8]. The basic idea from [9] is discussed in the next paragraph.

Let us assume that each optimization run requires a finite time G to execute and find an optimal or a near-optimal solution. Here, we assume that the problem does not change (or assumed to be constant) within a time

interval t_T, and $G \leq t_T$. We choose $\alpha = G/t_T$ to be a small value (say 0.25 or so), such that after the optimized solution is found, $(1 - \alpha)t_T$ time is spent on using the outcome for the remaining time. Figure 1 illustrates this dynamic optimization implementation procedure. The optimization problem is reformulated after every t_T time.

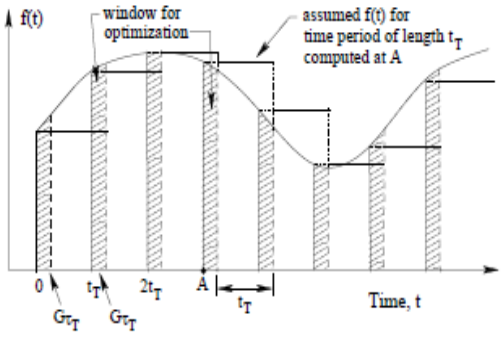

Fig. 1: The on-line optimization procedure for one objective (taken from [8]).

The shaded region indicates the duration of optimization and the white space added with next shaded region indicate the time for which the optimized solution will be implemented and used. During the t_T time window the problem is considered frozen, hence we name the procedure as the "frozen-time" approach. The extent of frozen time t_T is an important matter to fix for any problem. If we allow a large value of t_T (allowing a large number of optimization iterations to complete the optimization run, thereby allowing a near-optimal solution to be found), then, the frequency of reformulation of the optimization problem will be less. This may make a large change in the problem before the next reformulation is considered. On the other hand, if we choose a small value of t_T, a frequent change in the problem is considered (which approximates the real scenario more closely), but a lesser number of iterations are then allowed to track new optimal solutions for a problem which has also undergone a small change. Obviously, there lies a lower limit to t_T below which, albeit a small change in the problem, the number of iterations are not enough for an algorithm to track the new optimal solutions adequately. Such a limiting t_T will depend on the nature of the dynamic problem and the chosen algorithm, but importantly allows the best scenario (and closest approximation to the original problem) which an algorithm can achieve. We modify this idea from [8] and [9] and apply the frozen time approach to solve the optimal networking problem.

3.2. Optimization Algorithms

In this paper, we only consider weight changes, although the proposed procedure can be applied to other kinds of network changes as well.

Let us say that there are K different source-destination pairs possible. Let us also assume that we have a symmetric network for which the weights are the same in both forward and backward directions for any connection. So actually, we have K/2 different paths rather than a total of K paths. In our proposed approach, we first find all K/2 paths using Dijkstra's algorithm [7] for the given network and for the given set of initial weights (at $t=0$). For our calculaton purpose, we consider that a fixed proportion of the weights are changed with time in a sinusoidal manner, as shown below:

$$W_t = W_o + r \sin (2\pi (t/100) + \Theta),$$

where W_t is the new weight, W_o is the initial weight. The parameter r is a random number in the range 0 to 2.5 and Θ is an angle in the range $(-\pi)$ to π. The parameter t is the time index.

With all paths found optimally at the beginning, the network always works at its best for the first frozen-time window. Thereafter, after t_T time, the changed weights are considered and a new optimization problem is reformulated. Let us say that c% of the weights (chosen randomly) are changed every time there is an alteration in the problem. Thus, ideally all K/2 paths needs to be found again using an optimization procedure. If the frozen-time t_T is large enough to complete the optimization run fully, all paths can be re-optimized correctly; however, if the chosen frozen-time t_T is smaller than the time required to find all K/2 optimal paths, we need an optimization strategy to find as many new optimal paths as possible. Since all optimal paths will not be possible to be found in such a case, this may introduce some non-optimal and longer paths between certain source-destination pairs. Thus, considering a trade-off between the extent of non-optimal paths and frozen-time, we can arrive at a compromise value of the frozen time t_T. Figure 2 illustrates the concept. There can be two different scenarios with the trade-off curve. Figure 2 illustrates the scenario for which there is a "knee" region on the trade-off curve [19]. A knee region becomes a natural choice of a cut-off value of t_T, as choosing any other value does not make the trade-off worthwhile.

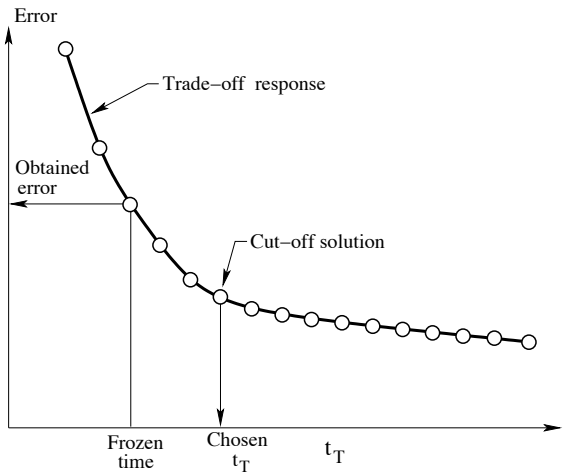

Fig. 2: The presence of a knee on the trade-off curve makes a natural choice of the cut-off value of t_T.

On the other hand, in some problems, there may not be a clear knee on the trade-off curve. In such a scenario, a cut-off value on the error can be assumed and a suitable cut-off time window t_T can be chosen from the curve, as shown in Figure 3.

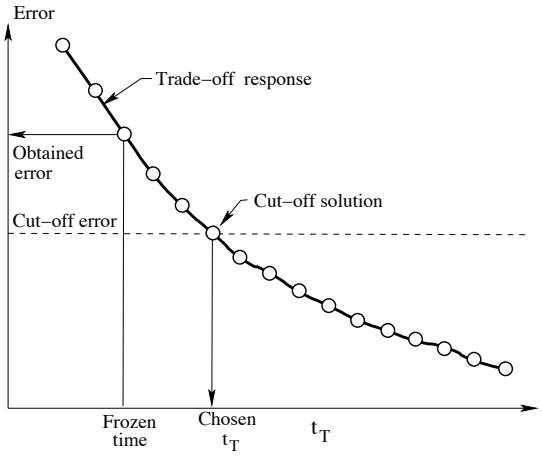

Fig. 3: The absence of a knee on the trade-off curve requires to choose a cut-off error value.

The above procedure of arriving at a compromise frozen-time will depend on the algorithm used in choosing source-destination pairs that are to be re-optimized and also the computational complexity of the optimization algorithm used for the purpose. We describe a few such algorithms used in this study.

The concept of using the above time-varying equation is to keep weight changes within a specific range and repeat them periodically after every sine cycle. In a real-world application, of course, such changes will be recorded after every t_T time and the corresponding optimization problem will be reformulated instantly. To find the number of errors (deviation from true optimal path with changed weights) for each frozen-time with a complete sinusoidal cycle, we apply each frozen-time step many times (iterations) and then calculate the average error for all iterations of that frozen-time window. We applied Dijkstra's algorithm [7] to find the optimal paths between all nodes of the network.

We apply three optimization techniques to find the minimum frozen-time window, which we discuss in the following paragraphs.

3.2.1 Method 1

The first method is to find the optimal paths just for those source-destination pairs for which their optimal paths from the last optimization run are impacted by the new weight changes. In this update method, we also ignore those source-destination pairs which are not impacted by the weight change. The idea came from the fact that the probability of a definite change in optimal path of a source-destination pair that involves an affected weight is more. However, we also realize that sources and destinations which are not included in the list of weights in a particular optimal path may also be impacted by the change in a weight, simply because the surrounding weights may offer a better path from source to destination. This may introduce some error (or deviation) in the new paths from their new optimal paths. But, the implementation of the above strategy is simple and we use this our first optimization method for quickly finding a new set of paths after every t_T time.

3.2.2 Method 2

Method 1 is computationally fast, but may introduce a large error, since it was concerned only on finding optimal paths for those sources and destinations that involved in changes in their weights and ignored the others. In Method 2, we try to improve this approach by using the rest of the allowable optimization time to find the optimal paths for those paths that are not involved with a weight change as long as it is allowed. Here we use the complete time (G= t_T) or α =1 for optimization.

3.2.3 Method 3

In this method, we associate a probability with a path based on its length, since the long paths have more probability to be impacted by a change in weight than the short paths. Thus, in this method, we apply those source-destination pairs that have longer paths first. Also, instead of sorting all paths, we use a dynamic programming approach [17] by saving the index of long paths. By that, we reduce the actual running time before even starting the probability method since the regular way to find the optimal paths between all sources and destinations needs to $O(n^2)$ computations [17]. Here, we use a matrix to save all the nodes in the network and by that we just applied one loop instead of expensive nested loops. We apply this method "on time" which means that every iteration depends on the optimal paths from the previous iteration which have an error percentage.

4. **Results**

To illustrate the working of the frozen-time algorithm and to compare the performance of three optimization methods proposed above, we consider a specific network optimization problem, shown in Figure 4 having 20 nodes with certain initial weights shown in the figure. This network is extended from the original one with 6 nodes from [7] to 20 nodes. There are K=380 different source-

destination pairs possible. Since the weights are the same in both forward and backward directions, there are 190 different paths that we would consider.

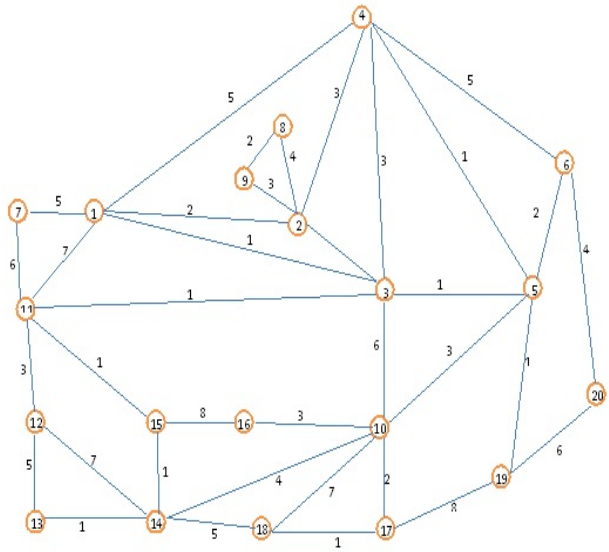

Fig. 4: Network with 20 nodes and initial weights.

4.3 Results from Method 1

The computational time to find all 190 paths take 0.097 sec on a specific computer used for this purpose. First, we consider c=50% changes in the weights after every t_T time steps. However, the fast nature of Method 1 requires around 0.080 sec in subsequent iterations. After every time steps, weights are changed and Method 1 is applied to find new paths. These paths are then compared with true optimal paths (obtained for the changed network) and the number of differences in their paths is used as an error measure. This process is continued 150 times and the error value is averaged. As shown in Figure 3, the error percentages are high. Another frozen t_T time is then considered and the procedure is repeated and the respective error percentage is computed. Figure 3 is then plotted with the corresponding t_T and error pair.

We can conclude from the figure that with those frozen times that are larger than 0.016 sec there is not a big change in the average error value. This can be explained as follows. When a large enough time is allowed for optimization, the algorithm gets enough time to find most critically changed paths and hence the average error reduces.

The shape of the variation of t_T and error is interesting to analyze and find a suitable frozen time t_T for actual implementation. With 50% changes in weights, it seems that the error continuously reduced with an increase in t_T. Thus, a choice of a suitable t_T will depend on the choice of an allowable error for operating the network. This can be achieved as follows. First, a suitable acceptable average error value (say 20% or 25%) can be chosen beforehand. Thereafter, when an offline study is performed and a response such as in Figure 5 is obtained, the actual frozen time can be obtained by using the chosen average error value from the graph.

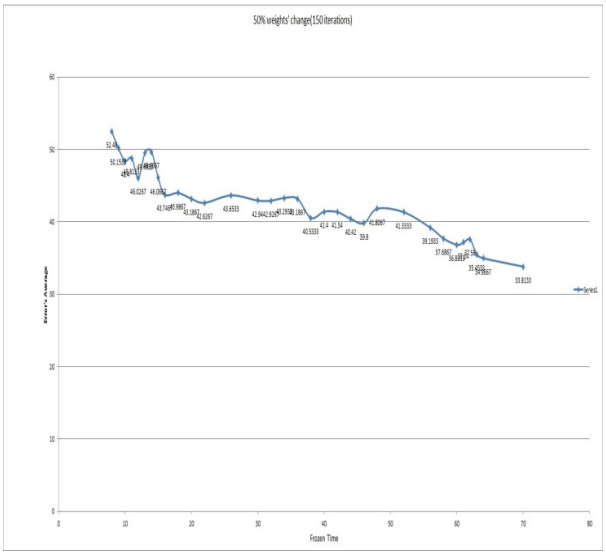

Fig. 5: 50% weights' changes with 150 iterations for Method 1.

4.4. Results from Method 2

Next, we apply Method 2 with two different weight changes. Figure 6 is plotted for c=25% weight changed percentage of all weights in the network. We notice an important insight in this figure. With an increase in t_T, the error percentage reduces sharply, but after around t_T=48 m-sec, the change in error is small. Such a "knee|" behavior [19] of t_T and error trade-off is interesting and provides a natural cut-off value for t_T.

Figure 7 is obtained with c=50% changes in weights. Here too, a cut-off t_T value of 50 m-sec is observed. What a knee behavior means is that a frozen time smaller than the cut-off time makes the error to be increasing at a fast rate, but beyond this cut-off time, the error stabilizes and does not change much. Thus, this cut-off time may make a natural choice for its implementation in practice.

4.5. Method 3 Results

Like in Method 2, we also apply Method 3 with two different weight changes – 25% and 50%. Figure 8 shows the trade-off variation between t_T and error with 25% weight changes. Figure 9 shows the same with 50% weight changes. The results are similar to that observed with Method 2. The optimal frozen time t_T for 25% and 50% weight changes is found to be approximately 47.5 m-sec and 49 m-sec, respectively.

The step-like behavior in some of these trade-off curves occurs due to that fact that although a fixed time t_T

is set, when a particular source-destination pair is started to optimize the process cannot be stopped until the optimization task is completed. Since the pairs are chosen at random (or according to the length), a significant change in t_T is needed to observe a substantial change in the average error value.

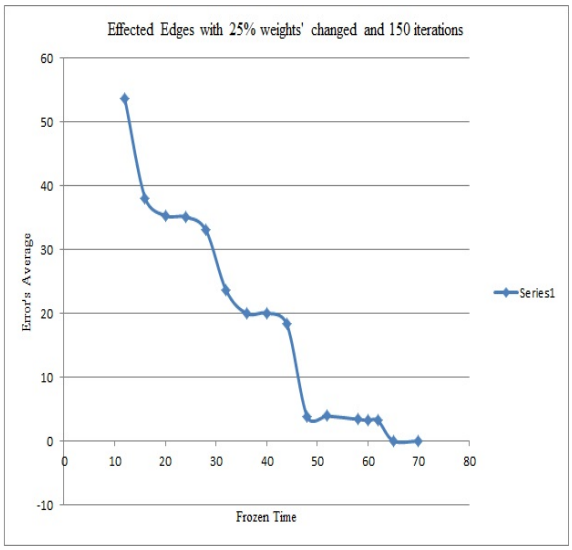

Fig.6: 25% weights' changes with 150 iterations for Method 2.

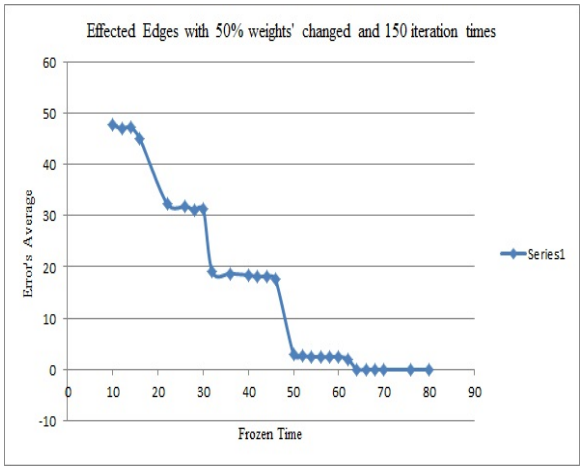

Fig. 7: 50% weights' changes with 150 iterations for Method 2.

5. Conclusions and Future Work

In this paper, we have addressed the issue of dynamic optimization a changing network for generic alterations. Although the simulations are performed for weight changes only, the methodology is applicable for other kinds of changes in a network as well. The idea proposed here is pragmatic and implementable in practice. Although the problem may change with time more frequently, the "frozen-time" approach constitutes the changed problem after every t_T time steps and the resulting problem is solved. If a small value of t_T is used, the change in the problem is expected to be small, but the allowed time for a new optimization is also small. Based on an error measure, the frozen-time approach suggests finding a trade-off between frozen time t_T and corresponding error measure. Thereafter, an analysis of the trade-off response has been suggested to choose a suitable frozen time t_T. In the event of a "knee" on the trade-off curve, a natural choice to is to choose the cut-off time t_T from the knee region.

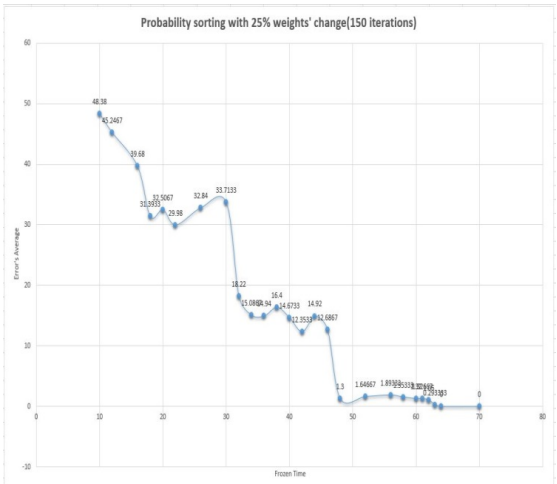

Fig. 8: 25% weights' changes with 150 iterations for Method 3.

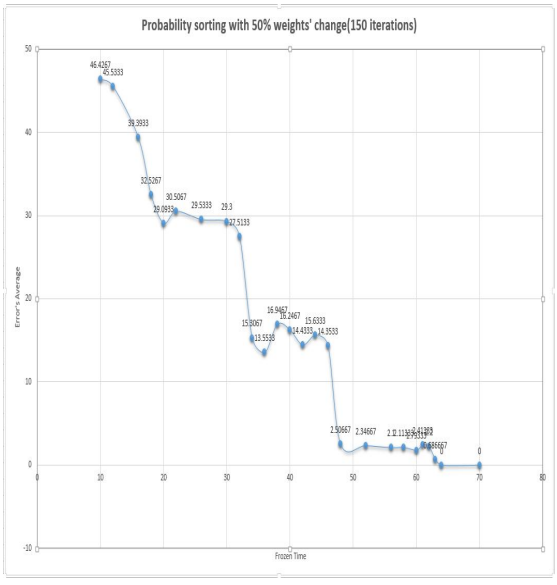

Fig. 9: 50% weights' changes with 150 iterations for Method 3.

Following conclusions can be drawn from this study:

1. Finding a set of optimal paths every time there is a change in the network is not the best computationally

tractable strategy. The use of a re-optimization in an efficient manner is needed. In this study, we have used an earlier proposed "frozen-time" algorithm with three different optimization strategies. The task in the frozen-time algorithm is to find minimum frozen-time to achieve a reasonably accurate solution for a problem. The approach is a simulation based off-line optimization strategy and is needed to be applied on a case to case basis.

2. The frozen time approach has been applied to a network with 20 nodes having a total of 190 different source-destination paths with three different optimization algorithms.

3. To arrive statistically significant results, we have simulated a dynamically changing problem for 150 iterations of the chosen frozen time.

4. Methods 2 and 3 have shown to produce trade-off curve between t_T and error measure having a knee, thereby allowing a natural choice for the critical frozen time window.

Based on the success of the frozen-time approach to the weight changed network optimization problem, we propose the following future studies.

1- For the future work, we plan to suggest further improvements on optimization methods using the dynamic programming method to reduce the time by saving the weights and the paths that are impacted by them in a matrix, so we can reach them directly without using expensive loop based methods.

2- Other optimization methodologies, such as a more efficient population-based evolutionary approach, will also be tried.

3- The frozen-time approach will be compared with other existing dynamic optimization approaches for finding advantages and disadvantages of different methods.

4- A multi-objective dynamic network optimization problem will be formulated and solved using the frozen time approach. Many applications, such as audio, video conferencing or collaborative environments and distributed interactive simulations have multiple QoS requirements such as bandwidth, packet delay, packet loss, cost etc. [3]. The method of scalarizing an objective vector into a single composite objective function is the way of converting the multi-objective optimization problem into a single objective optimization problem [3]. Since in a multi-objective problem, there will be more than one performance measure, we need to devise a single metric from multiple measures. For this purpose, a hyper-volume metric [20] can be used. The idea can be used for solving asymmetric networks for which there are at least two weights for each connection link resulting in multiple performance measures that must be considered for choosing a cut-off frozen time t_T.

5- In addition to weight changes considered in this study, a dynamic change in the connectivity of the network can also be tried. As a dynamic change, new links can be added or an existing link can be deleted, thereby simulating availability of new routes or "jamming" of some existing links in a network. The three algorithms described here can be used, but new optimization algorithms need to be devised to find reasonable solutions faster. The existing literature for such optimization methods can be used with the frozen-time approach.

The above changes for different kinds of dynamic changes in a network and the use of single and multiple performance measures will eventually make the whole approach a unified approach. We are undertaking these tasks as soon as is possible, nevertheless, this first study on the usefulness of the frozen-time approach with weight changes only in a network shows promise and gives us confidence for such a unified study.

REFERENCES

1- Annamaria Kiraly, Dynamic Multi-Objective Synthesis of Companies' Renewable Biomass and Energy Supply-Networks, *Chemical Engineering Transacations*. Vol. 35, 2013.

2- Camil Demetrescu, Dynamic Graph Algorithms, Chapman & Hall/CRC, ISBN: 978-1-58488-822-2, 2010.

3- C. Chitra, Multi-objective Optimization Solution for Shortest Path Routing Problem, *International Journal of Computer and Information Engineering,* Vol. 4, No. 2, 2010.

4- David Alberts, An empirical study of dynamic graph algorithms, *Journal of Experimental Algorithmics (JEA)* Vol. 2, 1997.

5- Fatimah Ismail, Self Organizing Multi-Objective Optimization Problem, *International Journal of Innovative Computing Information Control,* Vol. 7, No. 1, 2011.

6- Hsin-Yi Ma1, Applying Hierarchical Genetic Algorithm based Neural Network and Multiple Objective Evolutionary Algorithm to Optimize Parameter Design with Dynamic Characteristics, *Journal of Quality,* Vol. 17, No. 4, 2010.

7- James Kurose, *Computer Networking: A Top-Down Approach*, Fifth Edition, Addison-Wesley, 2011.

8- Kalyanmoy Deb, Dynamic Multi-Objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-Thermal Power Scheduling, *KanGAL Report Number 2006008*, 2007.

9- Kalyanmoy Deb, Single and Multi-Objective Dynamic Optimization: Two Tales from an Evolutionary Perspective, *KanGAL Report Number 2011004*, 2011.

10- Keehyung Kim, Multiobjective Evolutionary Algorithms for Dynamic Social Network Clustering,

Proceedings of the 12th annual conference on Genetic and evolutionary computation. Pages 1179-1186 ACM New York, NY, ISBN: 978-1-4503-0072-8, 2010.

11- Luc Wismans, Accelerating solving the dynamic multi-objective network design problem using response surface methods, *2nd International Conference on Models and Technologies for Intelligent Transportation Systems* 22-24 June, 2011, Leuven, Belgium, 2011.

12- Luc Wismans, *Towards Sustainable Dynamic Traffic Management*, http://doc.utwente.nl/81665/, 2012.

13- N. Ghajarnia, Multi-Objective Dynamic Design of Urban Water Distribution Networks, Vol. 7, No. 4, Winter 2012 (IR-WRR), 2012.

14- P. De, Dynamic Programming and Multi Objective Linear Programming approaches, *Applied Mathematics & Information Sciences – An International Journal,* Vol. 5, No. 2, Pages 253–49, 2011.

15- P. Deepalakshmi, An Ant Colony Based Multi-Objective Approach to Source-Initiated QoS Multicasting Method for Ad Hoc Networks, *Int. J. Advance. Soft Comput. Appl.,* Vol. 3, No. 2, 2011 ISSN 2074-8523, 2011.

16- Reinhard Bauer, Batch dynamic single-source shortest-path algorithms: An experimental study. *Experimental Algorithms, Vol.* 2, Pages 1–20, 2009.

17- Thomas Cormen, *Introduction to Algorithms*, Third Edition, MIT Press, 2009.

18- Xueli Liu, Dynamic Graph Shortest Path Algorithm, *WAIM 2012, LNCS 7418*, Pages 296–307, Springer-Verlag Berlin Heidelberg, 2012.

19- Kalyanmoy Deb and S. Gupta, Understanding Knee Points in Bicriteria Problems and Their Implications as Preferred Solution Principles, *Engineering Optimization*, Vol. 43, No. 11, Pages 1175-1204, 2011.

20- Eckart Zitzler, Lother Thiele, Marco Laumanns and Carlos M. Fonseca and V. G. Fonseca, Performance assessment of multi-objective optimizers: An analysis and review, IEEE Transactions on Evolutionary Computation, Vol. 7, No. 2, Pages 117--132, 2003.