Route Planning in a Weakly Dynamic Undirected Graph

Jean-Yves Colin
LITIS, Université du Havre, IUT
76610 Le Havre, France
Email: jean-yves.colin@univ-lehavre.fr

Ahmed Salem Ould Cheikh
Université de Nouakchott
Nouakchott, Mauritanie
Email: ahdsalem@univ-nkc.mr

Moustafa Nakechbandi
LITIS, Université du Havre, IUT
76610 Le Havre, France
Email: moustafa.nakechbandi@univ-lehavre.fr

***Abstract*—In this paper, we study weakly dynamic undirected graphs, and we propose an efficient polynomial algorithm that computes in advance alternative shortest paths for all possible configurations. No additional computation is then needed after any change in the problem because shortest paths are already known in all cases. We use these results to compute critical values for the traversal duration and to pre-compute best paths, and we apply this result to one delivery problem for trucks from one regional storehouse to several local stores when one possible point has a variable traversal duration.**

Keywords: graph theory; shortest path problem; adaptative scheduling

I. Introduction

In a static problem, all data are supposed known from the start. The real world is not static however, and the solutions to static problems are not always useful. Thus, dynamic problems, in which some of the data are neither static nor perfectly known in advance, must be studied [1] [15].

Several approaches have been developed [5]. Full dynamic algorithms usually deal with problems that can statically be solved in polynomial time. They start from an optimal solution to the static problem, and try to maintain it in an environment with many consecutive or simultaneous changes. Their authors present a lot of innovative data structures to reach this goal [8], [11] [13].

When the delay between the occurrence of a change, and the moment a solution is needed, is very short, or when the problem itself is NP-hard, even faster solutions are needed, although they do not try to find optimal solutions. Re-optimization algorithms use an initial solution, and are usually approximation algorithms which provide faster results than classical algorithms with the same guaranty on the approximation ratios, or which even guarantee better approximation ratios than classical static algorithms with the same speed performances [5] [10]. Others papers propose meta-heurisics, for example swarm intelligence algorithms, such as ant colony algorithms [2] [3].

Another approach uses probabilities. Probabilities are applied either to one or several parameters of the model (say a weight) instead of using a fixed value for this parameter, or the topology or structure itself of the model (say the presence or not of a constraint) [4] [9] [12]. The probabilistic combinatorial optimization algorithms proposed again usually compute a solution to an initial problem, then either do a robustness analysis of this solution in the probabilistic space, or do a quick re-optimization of this solution as soon as the real instance of the problem is known .

All these approches are used to study many problems, including route planning algorithms and transportation problems. For an overview on the problems, we refer to [16]. For example, the particular problem of finding point-to-point shortest paths on a large size road network with times depending traffic is studied in [6].

In this paper, we will deal with full dynamic problems that may be represented by a weakly dynamic undirected graph. We first define what we call weakly dynamic graphs. Then we propose an efficient polynomial algorithm that computes in advance shortest paths for all possible configurations of this graph. We use these results to compute critical values for the variable part of the graph and to pre-compute alternative shortest paths. Because the number of critical values is small, the number of alternatives is small too. No additional re-optimization is then needed after any change in the problem because shortest paths are already known in all cases. We apply this result to one delivery problem for trucks from one regional storehouse to several local stores when one possible point has a variable traversal duration.

II. Problem Definition

We first present a definition of weakly dynamic graph. This definition extends the definition presented in [7] and [14].

Definition 1: A *weakly dynamic* undirected (resp. directed) graph is an undirected (resp. directed) weighted graph in which there is one unstable weighted edge (resp. arc). That edge (resp. arc) has a variable positive weight that may change at any time. All other edges (resp. arcs) of this graph are stable, with known weights that never change.

In the rest of this paper, we will consider a weakly dynamic undirected graph $G(V, E)$, with V being the set of vertices of G and E being the set of edges. Each edge (i, j) of E has a positive stable weight $p_{i,j}$, except for an edge (x_1, x_2) of E that has a variable positive weight x that may change at any time. The cost or length of a path at a given moment is the sum of the weights of all of its edges at this moment. Figure 1 presents an example of a small weakly dynamic undirected graph.

We now want to find a set of shortest paths between a given vertex s_0 and each vertex s_n of G. We note $C_{s_0,s_n} = (s_0, s_1, ...s_n)$ a shortest path of adjacent vertices between s_0 and s_n, if there is at least one. We also want to know the length $d(s_0, s_n)$ of each shortest path C_{s_0,s_n} computed.

III. Main Results

A. The Proposed Algorithm

The proposed algorithm computes the shortest paths with the following three consecutive steps:

1) it first computes shortest paths $Cs(s_0, s_n)$ that do not include the variable edge (x_1, x_2), between vertex s_0 and each vertex s_n of the graph. It computes the constant length $ds(s_0, s_n)$ of each path $C(s_0, s_n)$ found, too,
2) If $ds(s_0, x_1) = ds(s_0, x_2)$ (i.e. the two vertices x_1 and x_2 of the variable edge are at the same distance from s_0), the algorithm stops because the paths $Cs(s_0, s_n)$ found in step 1 are shortest paths regardless of the value of x (cf. Corollary 1 later). Else it computes shortest paths $Cs(x_1, s_n)$ that do not include the variable edge (x_1, x_2), between vertex x_1 and each other vertex s_n of the graph, or shortest paths $Cs(x_2, s_n)$ that do not include the variable edge (x_1, x_2), between vertex x_2 and each other vertex s_n of the graph, depending on whether x_1 or x_2 is closer to s_0 (cf. Theorem 1 and Theorem 2 later). It computes their lengths too.
3) If $ds(s_0, x_1) < ds(s_0, x_2)$, then, for each vertex s_n of the graph, it computes a path $Cv(s_0, s_n) = Cs(s_0, x_1), Cs(x_2, s_n)$ that does include the variable edge (x_1, x_2), between vertex s_0 and vertex s_n. Its length $dv(s_0, s_n)$ is

$$dv(s_0, s_n) = ds(s_0, x_1) + x + ds(x_2, s_n) \qquad (1)$$

else if $ds(s_0, x_1) > ds(s_0, x_2)$, then, for each vertex s_n of the graph, it computes a path $Cv(s_0, s_n) = Cs(s_0, x_2), Cs(x_1, s_n)$ that does include the variable edge (x_1, x_2), between vertex s_0 and vertex s_n. Its length $dv(s_0, s_n)$ is

$$dv(s_0, s_n) = ds(s_0, x_2) + x + ds(x_1, s_n) \qquad (2)$$

For each vertex s_n, it then compares the length $ds(s_0, s_n)$ of the shortest path that do not include the variable edge (x_1, x_2) and the length $dv(s_0, s_n)$ of the computed path that includes the variable edge (x_1, x_2). A positive *critical value* $cv(s_0, s_n)$ of x for each vertex s_n is computed, if it exists, that states when either $Cs(s_0, s_n)$ or $Cv(s_0, s_n)$ is a shortest path between s_0 and s_n.

Because the weights are all positive, one may use Dijkstra's algorithm to compute efficiently the shortest paths in the different steps.

We will use the example of Figure 1 to illustrate the computation of shortest paths from vertex 0 to all other vertices (or from all other vertices to vertex 0).

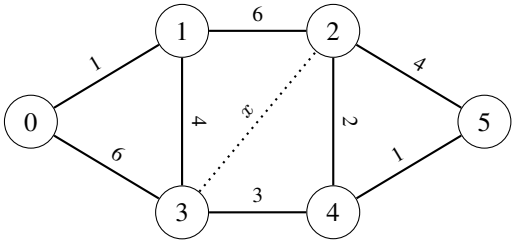

Figure 1. Example of weakly dynamic undirected graph. The dotted edge $(2, 3)$ is the variable edge of this graph, and is valuated with the variable value x.

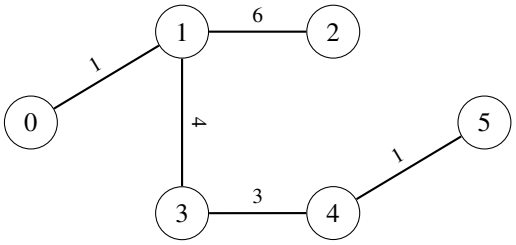

Figure 2. Shortest paths to vertex 0 if the variable edge $(2, 3)$ is not considered at all.

Table I
Distances from or to vertex 0 without the variable edge

vertex n	0	1	2	3	4	5
$ds(0, n)$	0	1	7	5	8	9

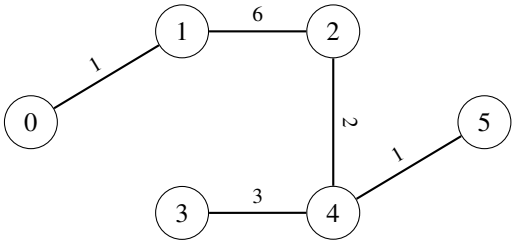

Figure 3. Shortest paths to vertex 2 if the variable edge is not considered at all.

The first step computes shortest paths from vertex 0 to all other reachable vertices without using the variable edge. An example of result is presented in Figure 2.

The distances $ds(0, n)$ of shortest paths from, or to, vertex 0 are presented in Table I.

One notes that vertex 3 is closer to vertex 0 that vertex 2 is. So step 2 now computes shortest paths from vertex 2 to all other reachable vertices, again without using the variable edge.

An example of result is presented in Figure 3.

The distances $ds(2, n)$ of shortest paths from, or to, vertex 2 are presented in Table II.

The length $dv(0, n)$ from vertex 0 to a vertex n of the computed path $Cv(0, n)$ that includes the variable edge $(2, 3)$ depends on the exact value of x. More precisely, $dv(0, n)$ is

Table II
DISTANCES FROM OR TO VERTEX 2 WITHOUT THE VARIABLE EDGE

vertex n	0	1	2	3	4	5
$ds(2,n)$	7	6	0	5	2	3

Table III
DISTANCES FROM OR TO VERTEX 0 WITHOUT AND WITH THE VARIABLE EDGE

vertex n	0	1	2	3	4	s_5
$ds(0,n)$	0	1	7	5	8	9
$dv(0,n)$	5+x+7	5+x+6	5+x+0	5+x+5	5+x+2	5+x+3

Table IV
DISTANCES AND CRITICAL VALUES FROM OR TO VERTEX 0 WITHOUT AND WITH THE VARIABLE EDGE

vertex n	0	1	2	3	4	5
$ds(0,n)$	0	1	7	5	8	9
$dv(0,n)$	5+x+7	5+x+6	5+x+0	5+x+5	5+x+2	5+x+3
$cv(0,n)$	-	-	2	-	1	1

then

$$dv(0,n) = ds(0,3) + x + ds(2,n) \quad (3)$$

because vertex 3 is closer to vertex 0 than vertex 2 is (cf. Theorem 1 and Theorem 2 later).

Table III presents the shortest distance $ds(0,n)$ to 0 without the variable edge, and the computed distance $dv(0,n)$ to 0 with the variable edge.

Step 3 of the algorithm builds Table III and compares the constant result $ds(0,n)$ and the variable result $dv(0,n)$ for each vertex n. For vertex 1 for example, it is obvious that there is no positive value of x such that $dv(0,1) < ds(0,1)$. For vertex 2 however, one can note that $dv(0,2) < ds(0,2)$ if $x < 2$, and that $dv(0,2) > ds(0,2)$ if $x > 2$. Else they are equal.

Definition 2: We call *critical value* $cv(s_0, s_n)$ of vertex s_n of V from vertex s_0 in a weakly dynamic graph G, the positive value such that a shortest path must use the variable edge if x is inferior to this critical value, and must not use the the variable edge if x is superior to this critical value. Thus, if $ds(s_0, x_1) < ds(s_0, x_2)$, we have

$$cv(s_0, s_n) = ds(s_0, s_n) - ds(s_0, x_1) - ds(x_2, s_n) \quad (4)$$

else, if $ds(s_0, x_1) > ds(s_0, x_2)$, we have

$$cv(s_0, s_n) = ds(s_0, s_n) - ds(s_0, x_2) - ds(x_1, s_n) \quad (5)$$

Table IV presents the computed distances to 0 without and with the variable edge, and the critical values cv associated to each vertex n, if any.

One can note that a critical value $cv(s_0, s_n)$ may not exist for each vertex s_n. If it exists and x is equal to this value for a given node, then a shortest path to this vertex that does not include the variable edge has the same length than a shortest path to this vertex that includes the variable edge.

B. Analysis

The following theorems may be proved.

Theorem 1: In any weakly dynamic undirected graph with a positive variable edge (x_1, x_2), if a shortest path from a vertex s_0 to s_n includes the variable edge (x_1, x_2) in one direction (from x_1 to x_2 for example), then any other shortest path from vertex s_0 to another vertex s_m that includes the variable edge (x_1, x_2) will do so in the same direction (x_1 to x_2 in the example)

Theorem 2: Let the shortest paths from s_0 to x_1 (resp. x_2) that do not include the variable edge (x_1, x_2) have a length of $ds(s_0, x_1)$ (resp. $ds(s_0, x_2)$). If $ds(s_0, x_1) < ds(s_0, x_2)$ then all shortest paths that include the variable edge will do so in the direction from x_1 to x_2. If $ds(s_0, x_1) > ds(s_0, x_2)$ then all shortest paths that include the variable edge will do so in the direction from x_2 to x_1.

Corollary 1: If $ds(s_0, x_1) = ds(s_0, x_2)$ then no shortest path from s_0 to a vertex s_n that includes the variable edge is shorter than the shortest path that do not include the variable edge, for any positive value x. So it is never necessary in this case to use the variable edge if there is a path froms_0 to s_n that does not include it.

Theorem 3: the paths computed by the algorithm are all shortest paths, depending on the value x of the variable edge.

Theorem 4: The algorithm has an overall complexity of $O(n^2)$.

Theorem 5: the number of critical values in a weakly dynamic undirected graph is inferior or equal to the number of vertices of the graph.

C. Using The Critical Values

One interest of critical values appears when one consider Theorem 5 and the set of all critical values found when computing shortest paths between a vertex s_0 and other vertices. That is, if we sort in ascending order all the critical values $cv(s_0, s_n)$ in a problem, one can remark that the computed set of shortest paths from vertex s_0 to all other vertices in the graph is the same for all values of x between two consecutive critical values. For example, in the example of Figure 1, when considering shortest paths to or from vertex 0, there is only a total of two critical values for the variable weight x as noted in Table IV: 2 appears once, and 1 appears twice. So there are three intervals for x: values from 0 to 1, values from 1 to 2 , and finally values above 2. Figure 4 presents shortest paths to vertex 0 if $x \leq 1$. Figure 5 presents shortest paths to vertex 0 if $1 < x \leq 2$. And Figure 6 presents shortest paths to vertex 0 if $2 \leq x$.

In any weakly dynamic undirected graph, the value x may change several times. But as long as it stays in the same interval of critical values, the shortest paths computed for this particular interval do not change and are still shortest paths. As soon as x leaves an interval of critical values and enters another one, then the shortest paths for the new interval the variable weight x is in now are the correct ones. Furthermore, the number of critical values is finite, so all the intervals of

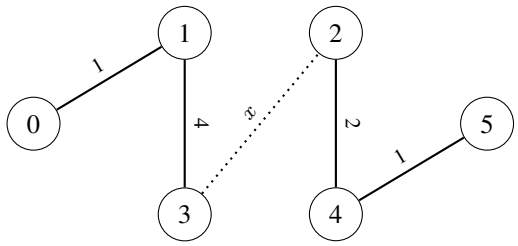

Figure 4. Shortest paths to vertex 0 if $x \leq 1$.

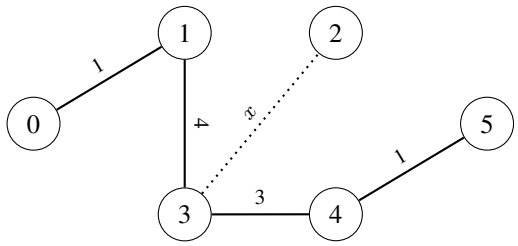

Figure 5. Shortest paths to vertex 0 if $1 < x \leq 2$.

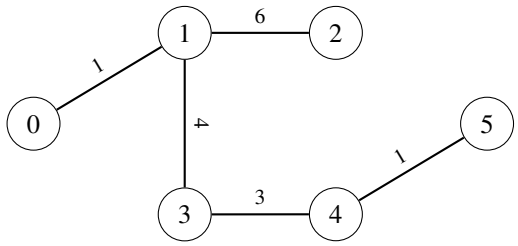

Figure 6. Shortest paths to vertex 0 if $2 < x$.

x and their corresponding sets of shortest paths may be pre-computed. These sets are alternative sets of shortest paths, statically computed, to be used depending on the current value of the variable weight,

Thus, the proposed algorithm can be used to efficiently pre-compute shortest paths for all possible values of x from a given vertex to all other vertices. It can be used too to efficiently pre-compute shortest paths from all vertices of a graph to a given target vertex, Once the alternative sets of shortest paths have been pre-computed for all intervals of critical values of x, they may be directly used and there is no need for later computations.

The next part presents an example of application to logistics.

IV. Application

We now study the following delivery problem. A fleet of trucks in an area must be managed. Each truck has to do one unique trip from the regional warehouse to one local delivery point. Because in this problem each truck may be considered independently from all the other ones, we shall limit ourself here to the particular case of one truck. One particular location in the area is known to have a variable traversal duration because of traffic jams, work in progress, or a lift bridge, for example. The traversal duration of this location may change one or more times during the trip. The position of the truck is known at all time, thanks to a GPS.

How to direct, and if necessary redirect, the truck, so that it follows the shortest path according to its current location and the current conditions?

This problem can be represented by a weakly dynamic undirected graph, with vertex s_0 being the location of the destination of the truck, s_n being its starting point, and variable edge (x_1, x_2) representing the particular location with a variable traversal duration x, The proposed algorithm can then be used to efficiently compute the critical values of x between all location and the destination, then the intervals of x, and finally alternative shortest paths from all locations according to the intervals.

We will use again the example of Figure 1 to illustrate this problem, with vertex 0 being the destination, and vertex 5 being the starting point. The proposed algorithm found 2 critical values, 1 and 2. This gave us three intervals for x: from 0 to 1, from 1 to 2, and 2 and above. The corresponding alternative shortest paths were presented in Figure 4, Figure 5, and Figure 6.

These alternative paths are all given to the truck at the start, with the current value of x. The truck may then check what interval x is currently in, and decide what path it must follow. During the trip, the value of x is monitored, either directly by the truck, or from somewhere else. As soon as x changes from the interval it was in, to another one, its value is communicated to the truck. The truck may then, without any re-computation, decide what is the current path it must follow now, according to the location it is currently at. If, for example, the truck started when x had a value of 0.5, so x was in the first interval, and the truck is currently at vertex 2, from vertices 5 then 4, Then the truck must now go from vertex 2 to vertex 3. However, if x changes to 4, x is now in the third interval, the one with values above 2. The set of shortest paths associated to this interval must then be consulted. Doing so, the truck immediately knows that its next destination is vertex 1, from vertex 2, in this case.

The return trip may be managed the same way, by computing too the critical values of x, the intervals and their alternative shortest paths, from all vertices to the regional warehouses' vertex 5.

One last important point must be considered, however. What if x has the pathological behavior of continually switching between low and high values, so that the truck alternatively follows paths that send it along a edge from a to b, then back from b to a, without end, in a ping-pong effect. This is a well known problem in dynamic problems. In this case one may use a simple heuristic that states that once a path that does not include the variable edge must be followed, because x reached a value high enough to make including it a poor decision, it must be followed to the end, even if x becomes smaller later.

V. Conclusion

In this paper, we studied weakly dynamic undirected graphs, and we proposed an efficient polynomial algorithm that computes in advance alternative shortest paths for all possible configurations. No additional computation is then needed after any change in the problem because shortest paths are already known in all cases. We used these results to compute critical values for the traversal duration and to pre-compute best paths, and we applied this result to one delivery problem for trucks from one regional storehouse to several local stores when one possible point has a variable traversal duration.

We are currently working on extending this result to weakly dynamic graphs with two or more variable edges. We also are studying how to apply this result to other kinds of graphs, such as graphs with directed arcs and cycles.

References

[1] M. Alivand, A.A. Alesheikh and M.R. Malek, "New method for finding optimal path in dynamic networks". World Applied Sci. J.,2008, 3: 25-33 (2008).

[2] H.R. Bajgan, and R.Z. Farahani. "Using colony system and neighborhood search for dynamic vehicle routing problem". American Journal of Operational Research, vol. 2, no.4, pp. 31-44 (2012).

[3] S. Balev, F. Guinand, Y. Pigné, "Maintaining Shortest Paths in Dynamic Graphs". In proceedings of the International Conference on Non-Convex Programming: Local and Global Approaches Theory, Algorithms and Applications (NCP'O7). 2007, December 17-21, Rouen (2007).

[4] D.J. Bertsimas. Probabilistic combinatorial optimization problems. PhD thesis, Massachusetts, Institute of Technology, (1988).

[5] N. Boria and V. T. Paschos, "Optimization in dynamic environments", Cahier du Lamsade 314, Université Paris-Dauphine, (2011).

[6] A. Casteigts, P. Flocchini. W. Quattrociocchi, and N. Santoro. "Time-varying graphs and dynamic networks". In ADHOC-NOW, pages 346–359, (2011).

[7] J.-Y. Colin, M. Nakechbandi and A.S. Ould Cheikh, "Searching Shortest Paths in Weakly Dynamic Graphs", ECCS'12 : European Conference on Complex Systems, sept. 3-7 2012, Brussels (2012).

[8] C. Demetrescu and G.F. Italiano, "A new approach to dynamic all pairs shortest paths", J. ACM, 2004, 51(6):968–992 (2004).

[9] D. Fulkerson, Expected critical path lengths in PERT networks. Operations Research 1962, 10 808 - 817 (1962).

[10] P. Jaillet. Probabilistic traveling salesman problems. PhD thesis, Massachusetts Institute of Technology, (1985).

[11] H. Mao, "Pathfinding Algorithms for Mutating Graphs", Computer Systems Lab 2007-2008 (2008).

[12] P.B Mirchandani and H. Soroush. "Optimal paths in probabilistic networks. A case with temporary preferences". Computers & Operations Research, 1985, 12:365 - 383 (1985).

[13] A. Orda and R. Rom. "Shortest-path and minimum-delay algorithms in networks with time dependent edge length". J. ACM, 1990, 37(3), 607–625 (1990).

[14] A. S. Ould Cheikh, J.-Y. Colin and M. Nakechbandi, "Problème de transport dans un graphe faiblement dynamique", GOL'12 : 1st International IEEE Conference on Logistics Operations Management, 17-19 October, Le Havre (2012).

[15] G. Ramalingam and T. Reps. "On the computational complexity of dynamic graph problems", Theoret. Comput. Sci. 1996, 158 233-277 (1996).

[16] P. Sanders and D. Schultes "Engineering Fast Route Planning Algorithms", LNCS 4525, pp. 23–36, (2007).