Bulletin of the *Transilvania* University of Braşov
Series III: Mathematics and Computer Science, Vol. 2(64), No. 1 - 2022, 193-196
https://doi.org/10.31926/but.mif.2022.2.64.1.13

SHORTEST PATHS IN A DIGRAPH WITH AN UNDERESTIMATED ARC WEIGHT

Laura CIUPALĂ[1], Adrian DEACONU[2] and Luciana MAJERCSIK[3]

Abstract

There are many real world problems that can be modeled and solved as shortest path problems. Among them, single-pair shortest path problems appear most frequently. Sometimes the weighted digraph, in which a shortest path problem is stated, suffers a minor data change (for instance, an arc weight might increase by a given amount a) due to the changes occurring in the corresponding real life problem. In this case, one needs to solve a shortest path problem in the modified digraph. If shortest paths are already determined in the initial digraph, these can be used as a starting point when determining shortest paths in the modified digraph or a known shortest path algorithm can be applied, from scratch, in the modified digraph. We will describe an algorithm that determines shortest paths from a given source node s in the new weighted digraph starting from the already known shortest paths in the original weighted digraph.

2000 *Mathematics Subject Classification:* 90B10, 90C90.
Key words: graph algorithms, shortest paths, incremental algorithms.

1 Introduction

The literature on graph algorithms is extensive([1], [2], [3]) and one of the reasons is the fact that graphs have widespread and diverse applications, such as reallocation of housing, assortment of steel beams, the paragraph problem, compact book storage in libraries, the money-changing problem, cluster analysis, concentrator location on a line, the personnel planning problem, single-duty crew scheduling, equipment replacement, asymmetric data scaling with lower and upper bounds, DNA sequence alignment, determining minimum project duration,

[1]Department of Mathematics and Computer Science, Faculty of Mathematics and Computer Science, *Transilvania* University of Braşov, Romania, e-mail: laura_ciupala@yahoo.com
[2]Department of Mathematics and Computer Science, Faculty of Mathematics and Computer Science, *Transilvania* University of Braşov, Romania, e-mail: a.deaconu@unitbv.ro
[3]Faculty of Mathematics and Computer Science, *Transilvania* University of Braşov, Romania, e-mail: luciana.majercsik@gmail.com

just-in-time scheduling, dynamic lot sizing etc. ([1]). Around 1960, researchers designed independently several shortest path algorithms: the first label-setting algorithm was developed by Dijkstra in 1959, by Dantzig in 1960 and by Whiting and Hillier also in 1960.

Shortest paths are extensively studied because they arise both when modeling and solving real world problems and as subproblems in more complex optimization problems, such as determining a maximum flow, a minimum flow or a minimum cost flow.

Let $G = (N, A)$ be a digraph, defined by a set N of n nodes and a set A of m arcs. The weight function $w : A \to \mathbb{R}_+$ associates to each arc $(x, y) \in A$ a weight $w(x, y)$, that can represent its length, cost, time, penalty etc. So, $G = (N, A, w)$ is a weighted digraph.

In the weighted digraph $G = (N, A, w)$, the single-source shortest path problem is to determine a shortest path from the given source node s to every node in N.

There are two types of algorithms that solve the single-source shortest path problem: label-setting algorithms and label-correcting algorithms. Both types are based on the notion of distance labels. During each iteration of any shortest path algorithm, there is a numerical value, named the distance label assigned to each node. If this value is infinite, it means that a path from the source node to that node is still to be found; otherwise, this represents the weight of some path from the source node to that node. The aim of the shortest path algorithms is to reduce the distance labels values to minimum, i.e. to the shortest paths weights.

The shortest path algorithms are based on the following optimality conditions:

Theorem 1. *[1](Shortest Path Optimality Conditions) For every node $y \in N$, let $d[y]$ denote the length of some directed path from the source node s to y. Then the numbers $d[y]$ represent the shortest path distances if and only if they satisfy the following shortest path conditions:*

$d[y] \leq d[x] + w(x, y)$, for all arcs $(x, y) \in A$.

2 Single-source shortest path problem in a digraph with an underestimated arc weight

In addition to being subproblems when dealing with more complex network optimization problems, the shortest paths problems arise when modeling and solving real life situations. In this case, data may slightly vary. This slight change in data implies small variation in the weighted digraph in which the problem is modeled and solved as a shortest path problem. For instance, a common change in a practical problem may cause an increase of an arc weight in the corresponding weighted digraph. If shortest paths from the source node to the other nodes in the original weighted digraph are already determined, then they can be used as a starting point when computing shortest paths in the modified digraph, instead of starting from scratch. We adopt this approach because it is the most efficient.

Let $G = (N, A, w)$ be the initial weighted digraph in which shortest paths from the source node s to the other nodes are already determined, let d be the array containing shortest paths weights and let p be the array containing the predecessors in the shortest paths. Let $\hat{G} = (N, A, \hat{w})$ be the weighted digraph that differs from G only by the weight of a given arc (k, l), which is larger than the initial weight of (k, l). So, $\hat{w}(x, y) = w(x, y)$, for each $(x, y) \in A \backslash \{(k, l)\}$ and $\hat{w}(k, l) = w(k, l) + a$, where a is a given positive amount.

There are two possible cases:

Case 1: $p[l] \neq k$. In this case, the shortest paths in G are, obviously, shortest paths in $\hat{G}$, too, and they have the same weights as in G.

Case 2: $p[l] = k$. In this case, in the modified digraph it is possible to have different shortest paths than in the initial digraph. These modified shortest paths can be only from the source node to node l and to the descendents of l in the predecessor subgraph $G' = (N, A')$ of G, where $A' = \{(p[x], x) | p[x] \neq 0\}$. We refer to these nodes towards which it is possible to have different shortest paths as affected nodes. They can be determined by applying the following recursive function with the effective parameter $\{l\}$.

```
Function affectedNodes(X)
Begin
    if X = ∅ then
        return φ;
    Y = ∅;
    for x ∈ X do
        Y = Y ∪ adjacentList[x];
    return Y∪ affectedNodes(Y);
end;
```

Suppose that we already know the array d with the shortest paths weights and the predecessor array p. All we have to do is to search for shorter paths towards the affected nodes. Because the weight function has positive values, we can use an approach similar to Dijkstra's algorithm. It may start with the set W of nodes towards which we are still looking for shortest paths containing all the affected nodes, instead of starting with $W = N$ as Dijkstra's algorithm does. The affected node set is generally a subset of the node set N, which implies that our algorithm performs less iterations than Dijkstra's algorithm.

The algorithm that solves the single-pair shortest path problem in the modified weighted digraph $\hat{G}$ starting with the known arrays d and p is the following:

```
UnderestimatedArc Algorithm;
Begin
    W = {l}∪affectedNodes({l});
    while W ≠ ∅  do
```

```
    begin
        select a node x from W such that d[x] = min{d[y]|y ∈ W};
        W = W\{x};
        for y ∈ adjacentList[x] ∩ W do
            if d[y] > d[x] + w(x, y) then
            begin
                d[y] = d[x] + w(x, y);
                p[y] = x;
            end;
    end;
end.
```

Theorem 2. *The UnderestimatedArc algorithm solves the single-pair shortest path problem in the modified weighted digraph $\hat{G}$ with an underestimated arc weight $O(m + n \log n)$ time.*

Proof. The only shortest paths in $\hat{G}$ that can differ from the shortest paths in G are those whose destination nodes are in the set of affected nodes. This set is determined by calling the recursive function affectedNodes for the effective parameter $\{l\}$. The recursive function determines in $O(m)$ time the set of descendents of the nodes in the set given as parameter. Then, the algorithm recomputes the values of the arrays d and p corresponding to the affected nodes in a way similar to Dijkstra's algorithm. Because in the worst case, the set of affected nodes contains $O(n)$ nodes, it follows that the time complexity of the UnderestimatedArc algorithm is the same as Dijkstra's algorithm, which is $O(m + n \log n)$ when implemented efficiently with Fibonacci heaps.

□

Remark 1. *Although the UnderestimatedArc algorithm has the same time complexity as Dijkstra's algorithm, $O(m + n \log n)$, it is more efficient than Dijkstra's algorithm because it does not compute the distance labels or the predecessors for the nodes that cannot be affected by the increase in the weight of the arc (k, l). This implies that our algorithm performs less iterations than Dijkstra's algorithm and it is more efficient in practice.*

References

[1] Ahuja, R., Magnanti, T., Orlin, J., *Network Flow. Theory, Algorithms and Applications*, Prentice Hall, New Jersey, 1993.

[2] Bang-Jensen, J., Gutin, G., *Digraphs, Theory, Algorithms and Applications*, Springer-Verlag, London, 2001.

[3] Cormen, T., Leiserson, C., Rivest, R., Stein, C., *Introduction to Algorithms*, MIT Press, 2009.