INFODIDACT (2008), 25–35
tmcs@inf.unideb.hu
http://tmcs.math.klte.hu

Teaching
Mathematics and
Computer Science

The single-source shortest paths algorithms and the dynamic programming

Zoltán Kátai

Abstract. In this paper we are going to present a teaching—learning method that help students look at three single-source shortest paths graph-algorithms from a so called "upperview": the algorithm based on the topological order of the nodes, the Dijkstra algorithm, the Bellman-Ford algorithm. The goal of the suggested method is, beyond the presentation of the algorithms, to offer the students a view that reveals them the basic and even the slight principal differences and similarities between the strategies. In order to succeed in this object, teachers should present the mentioned algorithms as cousin dynamic programming strategies.

Key words and phrases: optimal path algorithms, graph theory, teaching methods, programming techniques.

ZDM Subject Classification: B10, C70, K30, P50, Q30, Q60.

1. Introduction

Comenius, [1] considered the founder of modern teaching, made the following statement regarding teaching methods: *"To teach means scarcely anything more than to show how things differ from one another in their different purposes, forms, and origins. ... Therefore, he who differentiates well teaches well."* In this paper we are going to present a teaching—learning method that help students look at three famous single-source shortest paths graph-algorithms from a so called "upperview": the algorithm based on the topological order of the points, the Dijkstra algorithm, the Bellman-Ford algorithm. The goal of the suggested method is, beyond the presentation of the algorithms, to offer the students a view that reveals them the basic and even the slight principal differences and similarities between the strategies. According to the Comenius principle this is essential, if we want to master this field of graph theory and programming.

We use the notion of "upperview" in the following sense [10]:

(1) We can see the entities being analyzed "next to each other".

(2) Only those elements can be seen which are essential for the analysis.

(3) The similarities and differences are obvious, the connections are striking.

The purpose of the method is to offer the student a position where to have such a view to the analyzed object. In order to succeed in this, teachers should present the mentioned algorithms as cousin dynamic programming strategies.

As an illustration of the didactic value of the "upperview method", consider the following exercise. Given three rational numbers (11/119, 2/21, 5/51), look for similarities, differences and relations among them. At first sight there is not any evident relation among the three numbers. However, if we determine the common denominator of them, we obtain: 33/357, 34/357, 35/357. Having this form it is obvious that the three numbers are successive members of an arithmetic progression with common difference 1/357. In the case of the mentioned graph-algorithms the common denominator can be considered the common strategy they use, namely, the dynamic programming method. The "common denominator method" can be seen as an application of the "upperview method".

2. The shortest paths problem and the dynamic programming

The single-source shortest paths graph-algorithms solve the following problem: Given a weighted digraph ($G(V,E,w)$, V: set of nodes, E: set of arcs, $w\colon E \to R$ weight function) with n nodes $1, 2, \ldots, n$ and m arcs, determine the shortest paths from the node s (source node) to all the other nodes (destination nodes). The notation $\mathrm{wsp}(v)$ represents the weight of the shortest path from the source node s to the node v ($v \in V$).

If the graph is acyclic, then the best algorithm is based on the topological order of the nodes. The complexity of this algorithm is $O(n + m)$ [6]. If G contains cycles but no arcs with negative weight, the best solution for the single-source shortest paths problem is the Dijkstra algorithm. The complexity of the Dijkstra algorithm is $O(n^2)$ (using special date-structures this complexity can be improved) [6]. The Bellman-Ford algorithm solves efficiently the problem in the case when the graph has cycles and negative arcs, but there are not feasible cycles with negative weight (the sum of the weights of the arcs that constitute the cycle is negative). The complexity of the Bellman-Ford algorithm is $O(nm)$ [6].

In the book entitled Introduction to Algorithms [6] the authors are trying to find a kind of common denominator among the discussed algorithms by presenting them as applications of the so called successive approximation method. (Interestingly, the authors consider the Dijkstra algorithm mainly a greedy strategy, and in the case of the two other algorithms they do not make any connection with a specific programming technique.) Sniedovich [11] goes further and presents the mentioned algorithms as dynamic programming strategies on the base of the fact that the successive approximation method is a dynamic programming strategy (Other references: [4, 9]). (According to Sniedovich [11], a careful reading of Dijkstra's original paper [3] "leaves no room for doubt" that methodologically the Dijkstra algorithm is based on Bellman's Principle of Optimality and technically it constitutes a successive approximation procedure for the solution of the functional equation of dynamic programming associated with the single-source shortest paths problem.)

There are to strategies to transpose the functional equation of dynamic programming into an algorithm: the direct method ("direct-conversion" of the functional equation into an iterative procedure) and the successive approximation methods (after an initial approximation, the $(s \rightarrow v)$ "distances" $(v \in V)$ are successively updated (improved) either by the functional equation itself or by an equation related to it) [11].

The didactical method presented in this paper brings the three shortest paths algorithm much closer to each other, and, consequently, transmits to the students a much clearer "upperview" than the methods mentioned above.[1]

3. The principals of the dynamic programming

The dynamic programming is characterized by the following principals:

(1) The dynamic programming strategy can be considered as an implementation of the principal of the optimality: the optimal solution is built by optimal sub-solutions. Consequently, the dynamic programming follows this strategy: starts from the optimal solutions of the trivial sub-problems and builds the

[1] This research was supported by the Research Programs Institution of the Sapientia University.

optimal solutions of the more and more complex sub-problems and eventually of the original problem. (Bottom-up way) In the case of the shortest path problem this principal can be announced as it follows: Parts of any shortest path are also shortest paths.

Consequences of the principal of the optimality in the case of shortest paths problem [6]:

- The optimal paths with more arcs can be built from optimal paths with fewer arcs.
- The optimal paths constitute a rooted tree. All the three algorithms build this optimal-paths-tree as one which grows progressively from the root node s. (At each step the tree is extended with a new arc which binds to the tree a new node.)
- If the node u is the predecessor of the node v on the optimal path from s to v, then:
$$\mathrm{wsp}(v) = \mathrm{wsp}(u) + w(u,v).$$
- For all arcs (u,v) it is true that:
$$\mathrm{wsp}(v) \leq \mathrm{wsp}(u) + w(u,v).$$

(2) The basic principle of optimality is expressed by a recursive formula (named by Bellman as the dynamic programming functional equation), which describes mathematically the way in which the solutions of the more and more complex sub-problems are built from the simpler sub-problems' optimal solutions.

In the case of the shortest paths problem the functional equation has the following form (for each node v, pred(v) denote the set of the immediate predecessors) [11]:

$$\begin{aligned}
&\mathrm{wsp}(s) = 0 \text{ (we assume that } \mathrm{pred}(s) = \emptyset)\\
&\mathrm{wsp}(v) = \infty, \text{ if } \mathrm{pred}(v) = \emptyset, \text{ and } v \neq s\\
&\mathrm{wsp}(v) = \min\{\, \mathrm{wsp}(u) + w(u,v) \mid u \in \mathrm{pred}(v)\,\}, \text{ if } \mathrm{pred}(v) \neq \emptyset
\end{aligned}$$

(3) The dynamic programming method records the optimal values of the optimal solutions of the already solved sub-problems (the optimal values of the object function to be optimized concerning the respective sub-problems).

According to the successive approximation method all the three algorithms we are going to present use a distance array $d[1..n]$ (the weights of the arcs can also be seen as distances), where the element $d[v]$ stores the "current optimal weight" (in the actual stage of the algorithm) of the path from the node s to the node v. Initially all the elements of the array d are set with ∞, excepted $d[s]$ which is set with 0. During the algorithm the values of the elements $d[v]$ $(v = 1, \ldots, n)$ are updated (hopefully decreased) according to the following procedure (updating operation on the basis of the arc (u, v)):

```
if $d[v] > d[u] + w(u, v)$ then
   $d[v] = d[u] + w(u, v)$
end if
```

In the end of the algorithms the array d will contain the "wsp-values" of the nodes ($d[v] = wsp(v)$ for all $v = 1, \ldots, n$).

An other classification of the dynamic programming strategies is based on the way the "wsp-values" are calculated. The so called "pull-approach" calculates direct (not through an updating process) the "wsp-value" of the current node on the basis of the already calculated "wsp-values" of its immediate predecessors. This approach is an immediate application of the functional equation, and can be used only for the acyclic graphs. In this case the array $d[1..n]$ can be replaced with the array $\mathrm{wsp}[1..n]$, which elements are calculated according to the topological order of the nodes [2, 8, 9, 11].

The key idea in the case of the "push-approach" is to propagates any improvement that has been made in the current node u (the $d[u]$ value has been updated) to its immediate successors (if v is a successor of the u, than $d[v]$ is updated on the basis of the arc (u, v)). The algorithm ends when any other improvements cannot be performed. All the three algorithms we are presenting apply the "push-approach" [5, 7, 9, 11].

4. Shortest path algorithms as dynamic programming strategies

According to the above exposed ideas we can state that:

- If node u is the immediate predecessor of node v on the optimal path from s to v, then the $\mathrm{wsp}(v)$ value can be figured out ($d[v]$ reaches its minimum) only after the $\mathrm{wsp}(u)$ value has been determined ($d[u]$ has already been reached its minimum).
- The $\mathrm{wsp}(v)$ value is calculated by updating $d[v]$ on the basis of the arc (u, v) (after $d[u]$ has already been reached the $\mathrm{wsp}(u)$ value).

Consequently the algorithms, in order to determine the "wsp-values" for the all nodes, have nothing else to do than to update the array d on the basis of the arcs of the shortest paths tree in their topological order (topological-optimal_tree-arc-sequence). According to this order, arc i precedes arc j if arc i precedes arc j on any shortest path of the graph. (Evidently, the topological-optimal_tree-arc-sequence is not unique.) This updating process in effect is a dynamic programming solution-building process.

Since the shortest paths tree is unknown (this tree represents the solution of the problem) the algorithms generate arc-sequences which contain, as sub-sequence, the topological-optimal_tree-arc-sequence necessary for the dynamic programming building process. The basic difference between the presented algorithms is the way they generate a proper arc-sequence.

4.1. The algorithm based on the topological order of the nodes

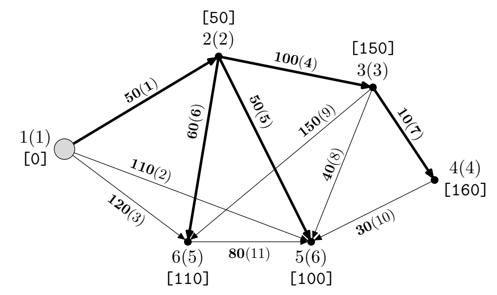
In the case that the graph is acyclic it is obvious that if we sort the arcs of the graph on the basis of the topological order of their start points this arc-sequence will contain, as sub-sequence, a topological-optimal_tree-arc-sequence. Consequently, this algorithm traverses the nodes of the graph (starting with node s) according to their topological order, and updates the $d[v]$ value of all immediate successors of the current node u ($v \in \mathrm{succ}(u)$) on the basis of the arc (u, v) (Figure 1(a))

Remarks:

- At the moment we have arrived to a node, the corresponding element in array d already stores its "wsp-value". The algorithm only confirms this optimal value.
- The succession the "wsp-values" of the nodes are determined (confirmed) is "predestinate" by the topological order of this nodes. The topological order of the nodes can be established by a DFS procedure.
- The algorithm attempts to approximate with each arc at most ones.

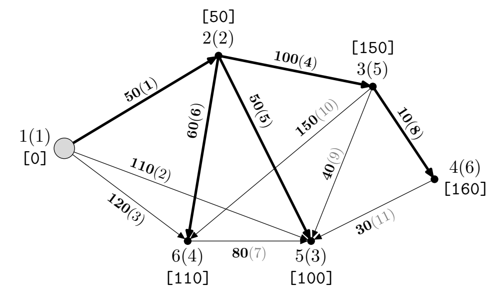
4.2. The Dijkstra algorithm

In the case of the Dijkstra algorithm the cycles are welcome, but arcs with negative weight are not allowed. In this situation it can be observed that along the shortest paths the "wsp values" of the nodes are in ascending order. Consequently, the Dijkstra algorithm traverses the nodes of the graph according to the ascending



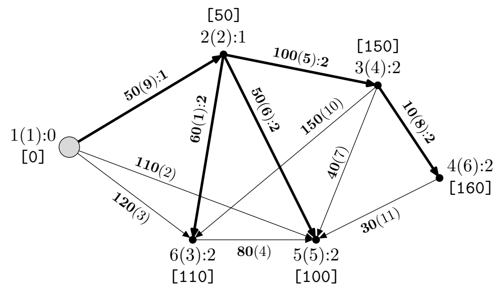
$(\mathbf{1},\mathbf{2}), (1,5), (1,6), (\mathbf{2},\mathbf{3}), (\mathbf{2},\mathbf{5}), (\mathbf{2},\mathbf{6}), (\mathbf{3},\mathbf{4}), (3,5), (3,6), (4,5), (6,5)$

(a)



$(\mathbf{1},\mathbf{2}), (1,5), (1,6), (\mathbf{2},\mathbf{3}), (\mathbf{2},\mathbf{5}), (\mathbf{2},\mathbf{6}), (6,5), (\mathbf{3},\mathbf{4}), (3,5), (3,6), (4,5)$

(b)



1: $(2,6), (1,5), (1,6), (6,5), (2,3), (2,5), (3,5), (3,4), (\mathbf{1},\mathbf{2}), (3,6), (4,5)$
2: $(\mathbf{2},\mathbf{6}), (1,5), (1,6), (6,5), (\mathbf{2},\mathbf{3}), (\mathbf{2},\mathbf{5}), (3,5), (\mathbf{3},\mathbf{4}), (1,2), (3,6), (4,5)$

(c)

Figure 1. The strategies of the Topological (a) / Dijkstra (b) / Bellman-Ford (c) algorithms. The “topological-optimal_tree-arc-sequence”, as sub-sequence of the generated arc-sequence, is bolded.

order of their “wsp-values”, and updates the $d[v]$ values of all immediate successors of the current node u ($v \in \text{succ}(u)$) on the basis of the arc (u, v). This arc-sequence contains, as sub-sequence, a topological-optimal_tree-arc-sequence (Figure 1(b))

Remarks:

- It is evident that in this case the order the “wsp-values” of the nodes are determined (confirmed) is unpredictable. Therefore Dijkstra’s algorithm determines this order on the fly (during the algorithm): If the node v is the out-neighbour of the current (confirmed) shortest-path-tree that is “closest” to the source node (according to the current values of the array d), then $d[v]$ is confirmed as the “wsp-value” of the node v ($\text{wsp}(v) = d[v]$). (This greedy choice can be justified as follows: if the other out-neighbours are farther from the node s than the node v, then through this nodes cannot lead shortest paths from s to v than $d[v]$). In the other words, v is the next node where the shortest path is confirmed. If the last update of the element $d[v]$ was performed on the basis of the arc (u, v), then node v is linked to the shortest-paths-tree through this arc. We could say that $\text{wsp}(v)$ is the next value in the ascending succession of the “wsp-values” of the nodes.
- The algorithm attempts to approximate with each arc at most ones.

5. The Bellman-Ford algorithm

The Bellman-Ford algorithm solves the single-source shortest paths problem efficiently when the graph contains cycles and “negative-arcs”, but there are not feasible cycles with negative weight. In this case it is unknown any method that generates such arc-sequence (containing each arc at most ones) which contains, as sub-sequence, a topological-optimal_tree-arc-sequence. Therefore this algorithm goes through (in arbitrary order) all the arcs of the graph (and attempts to approximate with them) again and again until the arc-sequence generated in this way finally will contains, as sub-sequence, a topological-optimal_tree-arc-sequence (Figure 1(c)).

Remark:

- It needs at most $(n-1)$ tours. During a last extra-tour the algorithm realizes that all elements of the array d have reached their optimal value (“wsp-value”). (There were not any updates.)

6. Teaching method for the single-source shortest paths algorithms

According to the presented material we suggest the following didactical method:

- The teacher presents the principals of the dynamic programming with respect to the single-source shortest paths problem.
- The teacher helps students to understand that
  - the "wsp-values" of the nodes can be determined by updating the elements of the array d on the basis of the arcs of the shortest paths tree in their topological order.
  - this arc-sequence is unavailable directly.
  - the updating process effectuated according to the "topological-optimal_tree-arc-sequence" constitutes dynamic programming.
- The teacher helps students to realize that the "topological-optimal_tree-arc-sequence" can be generate as sub-sequence of
  - a complete arc-sequence sorted by the topological order of the start-points of the arcs (acyclic graph).
  - a complete arc-sequence sorted on the basis of the "wsp-values" of the start-points of the arcs (graph without "negative arcs").
  - "multiple arc-sequence" (graph without feasible "negative cycles").
- The teacher explains how the three algorithms generate the above mentioned arc-sequences, and update the array d on the basis of them.
- The students are invited to identify the specific characteristics of the three algorithms. For example:
  - The first algorithm can predict the node-succession the "wsp-values" are determined (confirmed).
  - The second algorithm determines (confirms) the "next wsp-value" on the fly with greedy choices.
  - The third algorithm has to generate a "multiple arc-sequence".
- The students implement the algorithms.

7. Conclusions

The "upperview character" of the teaching method described above can be illustrated as follows. We denote the three graph-algorithms with the following initials: T, D and BF (Topological, Dijkstra, Bellman-Ford). As this letters at firs sight do not have any commonality, the students may not find any common feature in the algorithms the letters represent. The way the presented method transforms the "image of the algorithms" (in the students' mind) into a so called "common denominator form" can be illustrated with the "fractions" below:

$$GT/DP \qquad GD/DP \qquad GBF/DP$$

DP denotes the common basis of the algorithms: Dynamic Programming. GT, GD and GBF denote the ways the algorithms generate "topological-optimal_tree-arc-sequence"-s.

As the above illustration also emphasizes, there are more common features in the presented algorithms than the fact that all of them apply the successive approximation variant of the dynamic programming method. The presented method accentuates that the core idea in all the three algorithms is to generate a topologically ordered sequence of the arcs of the shortest paths tree (in order to make updating operation on the basis of this arc-succession), and all the three algorithms provide these arc-sequences as sub-sequences of "proper ordered complete arc-sequences" or of a "multiple arc-sequence". This approach transmits to the students a much clearer "upperview" than the traditional ones.

References

[1] Comenius, *Orbis sensualium pictus*, 1653.

[2] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, 1957.

[3] E. W. Dijkstra, A note on Two Problems in Connexion with Graphs, *Numerische mathematik* **1** (1959).

[4] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Whinston, NY, 1976.

[5] E. V. Denardo and B. L. Fox, Shortest-route methods: 1. Reaching, pruning, and buckets, *Operations Research* **27** (1979).

[6] T. H. Cormen, C. E. Leirserson, R. L. Rives, *Introduction to Algorithms*, The Massachusetts Institute of Technology, 1990.

[7] D. Bertsekas, *Linear Network Optimization*, The MIT Press, Cambridge, MA, 1991.

[8] M. Sniedovich, *Dynamic Programming*, Marcel Dekker, NY, 1992.

[9] E. V. Denardo, *Dynamic Programming*, Dover, Mineola, NY, 2003.

[10] Z. Kátai, “Upperview” algorithm design in teaching computer science in high schools, *Teaching Mathematics and Computer Science* **3**, no. 2 (2005), 221–240.

[11] M. Sniedovich, Dijkstra’s algorithm revisited: the dynamic programming connexion, *Control and cybernetics* **35**, no. 3 (2006), 599–620.

ZOLTÁN KÁTAI
SAPIENTIA–EMTE
TÎRGU-MUREŞ
ROMANIA

E-mail: katai_zoltan@ms.sapientia.ro

(Received May, 2008)