Electronic Colloquium on Computational Complexity, Report No. 158 (2019)

Sorting Can Exponentially Speed Up Pure Dynamic Programming

Stasys Jukna[1], Hannes Seiwert*

Institute of Computer Science, Goethe University, Frankfurt am Main, Germany

Abstract

Many discrete minimization problems, including various versions of the shortest path problem, can be efficiently solved by dynamic programming (DP) algorithms that are "pure" in that they only perform basic operations, as min, max, +, but no conditional branchings via if-then-else in their recursion equations. It is known that any pure $(\min, +)$ DP algorithm solving the minimum weight spanning tree problem on undirected n-vertex graphs must perform at least $2^{\Omega(\sqrt{n})}$ operations. We show that this problem *can* be solved by a pure $(\min, \max, +)$ DP algorithm performing only $O(n^3)$ operations. The algorithm is essentially a $(\min, \max)$ algorithm: addition operations are only used to output the final values. The presence of both min and max operations means that now DP algorithms can sort: this explains the title of the paper.

Keywords: Spanning tree, MST problem, dynamic programming

1. Introduction

A discrete 0-1 optimization problem is specified by giving a finite set E of *ground elements* together with a family $\mathcal{F} \subseteq 2^E$ of subsets of these elements, called *feasible solutions*. The problem itself is, given an assignment of nonnegative real weights to the ground elements, to compute the minimum or the maximum weight of a feasible solution, the latter being the sum of weights of its elements. Note that we only need $(\min, +)$ or $(\max, +)$ operations to *define* such problems.

For example, in the *assignment problem* E is the set of all edges of a complete bipartite graph and $\mathcal{F}$ is the family of all perfect matchings in it, each viewed as set of its edges. In the *MST problem* (minimum weight spanning tree problem) on a connected graph $G = (V, E)$, feasible solutions are spanning trees of G, etc.

Dynamic programming (DP) is a fundamental algorithmic paradigm for solving such optimization problems. Many DP algorithms are *pure* in that they only perform basic operations, as min, max, +, −, in their recursion equations, but no *conditional branchings* via if-then-else or argmin/argmax, or other additional operations. In particular, the recursions then do not depend on the actual input weightings.

*Corresponding author.

Email addresses: `stjukna@gmail.com` (Stasys Jukna), `seiwert@thi.cs.uni-frankfurt.de` (Hannes Seiwert)

[1]Research supported by the DFG grant JU 3105/1-2 (German Research Foundation). Affiliated with the Institute of Data Science and Digital Technologies, Faculty of Mathematics and Informatics of Vilnius University, Lithuania.

ISSN 1433-8092

Notable examples of pure DP algorithms are the Bellman–Ford–Moore algorithm for the shortest s-t path problem [1, 5, 13], the Floyd–Warshall algorithm for the all-pairs shortest paths problem [3, 15], the Held–Karp DP algorithm for the traveling salesman problem [6] and the Dreyfus–Levin–Wagner algorithm for the weighted Steiner tree problem [2, 10]. The Viterbi $(\max, \times)$ DP algorithm [14] is also a pure $(\min, +)$ DP algorithm via the isomorphism $h : (0, 1] \to \mathbb{R}_+$ given by $h(x) = -\ln x$.

There are, however, important optimization problems that can be efficiently solved using greedy-type algorithms, but cannot be efficiently solved by pure $(\min, +)$ or $(\max, +)$ DP algorithms. One of such problems, is the famous *MST problem* on an undirected connected graph $G = (V, E)$, which we have already mentioned above: given an assignment $x : E \to \mathbb{R}_+$ of nonnegative real weights to the edges of G, compute the minimum weight $\mathrm{mst}(x) = \mathrm{mst}_G(x)$ of a spanning tree of G:

$$\mathrm{mst}(x) = \min\{x(T) \colon T \text{ is a spanning tree of } G\}\,,$$

where $x(T) = \sum_{e\in T} x(e)$; here and throughout $\mathbb{R}_+$ stands for the set of all nonnegative real numbers. In the *directed* version of the MST problem, known as the *minimum arborescence problem*, the underlying graph G is directed and one seeks for the minimum weight of an arborescence of G; an *arborescence* of a digraph G is a directed tree in which all vertices of G are reachable by directed paths from one fixed root vertex.

That every pure $(\min, +)$ DP algorithm for the minimum arborescence problem on the complete n-vertex graph $G = K_n$ must perform $2^{\Omega(n)}$ operations was proved by Jerrum and Snir in their seminal paper [8]. As we have recently shown in [9], even the simpler *undirected* MST problem requires $2^{\Omega(\sqrt{n})}$ operations. So, pure $(\min, +)$ DP algorithms for both these problems must perform an *exponential* in n number of operations.

Therefore, the following result of Fomin, Grigoriev and Koshevoy [4] came as a surprise. Using ideas from the electrical engineering (Kirchhoff's effective conductance formula and the star-mesh transformation to compute effective conductances), they show that both (the directed and the undirected) MST problems *can* be solved by pure $(\min, +, -)$ DP algorithms performing only $O(n^3)$ operations. That is,

- *subtraction* can exponentially speed up pure $(\min, +)$ DP algorithms.

In this paper, we show that, in fact, the MST problem can already be solved by a pure $(\min, \max, +)$ DP algorithm performing only $O(n^3)$ operations (Theorem 2 below). Hence, already the *monotone* max operation, instead of the *non-monotone* subtraction $(-)$ operation, can exponentially speed up pure $(\min, +)$ DP algorithms. The presence of *both* min and max operations means that now DP algorithms can sort; this explains the title of this paper:

- already *sorting* can exponentially speed up pure $(\min, +)$ DP algorithms.

Note that $(\min, +, -)$ operations can be (easily) simulated by $(\min, \max, +)$ operations because $\max(x, y) = -\min(-x, -y)$, but not vice versa.

2. Our results

Let $G = (V, E)$ be an undirected connected graph. Given a weighting $x : E \to \mathbb{R}_+$ of the edges of G, the *min-max distance* between two vertices u and v, which we denote by $\mathrm{dist}_x(u, v)$, is the minimum, over all paths from u to v in G, of the maximum weight of an edge along this path:

$$\mathrm{dist}_x(u, v) = \min_P\ \max\{x(f) \colon f \in P\}\,,$$

where the minimum is taken over all paths P in G between the vertices u and v. That is, the min-max distance between vertices u and v is the minimum number d for which there is a path in G between u and v with all edges of weight at most d. The min-max distance $\mathrm{dist}_x(e)$ of an edge $e = \{u, v\}$ is the min-max distance between its endpoints u and v. Note that the min-max distance of any edge does not exceed its weight (the edge itself is a path between its endpoints), but may be smaller, that is, we always have $\mathrm{dist}_x(e) \leq x(e)$.

The following theorem relates min-max distances to the MST problem.

Theorem 1. *Let $G = (V, E)$ be an undirected n-vertex graph, and $T = \{e_1, \dots, e_{n-1}\}$ be a spanning tree of G. Then for every weighting $x : E \to \mathbb{R}_+$, we have*

$$\mathrm{mst}(x) = \mathrm{dist}_{x_0}(e_1) + \mathrm{dist}_{x_1}(e_2) + \cdots + \mathrm{dist}_{x_{n-2}}(e_{n-1})\,,$$

where $x_0 = x$, and each next weighting $x_i : E \to \mathbb{R}_+$ is obtained from x by setting the weights of edges $e_1, \dots, e_i$ to zero.

Theorem 1 allows us to efficiently solve the MST problem by a pure DP algorithm performing only min, max and + operations. Namely, we can fix an *arbitrary* spanning tree T of G; this tree T will be used for *all* arriving weightings $x : E \to \mathbb{R}_+$ of the edges of G. When an input weighting x arrives, compute the min-max distances of the $n - 1$ edges of the (fixed) tree T under the corresponding modifications of the weighting x by the Floyd–Warshall DP algorithm. By Theorem 1, the sum of all these distances is then exactly the minimum weight of any spanning tree of G with respect to the input weighting x. This yields a pure $(\min, \max, +)$ DP algorithm performing $O(n^4)$ operations. Some additional savings (see Section 4 for details) lead to the following theorem.

Theorem 2. *The MST problem on every undirected connected graph on n vertices can be solved by a pure $(\min, \max, +)$ DP algorithm performing $O(n^3)$ operations.*

Remark 1. Hu [7] reduced the problem of computing all min-max distances to the MST problem. When an input weighting x of the edges arrives, find a spanning tree T_x of G of minimal x-weight. Then, with respect to this weighting, the min-max distance between any pair of vertices of G is the maximal weight of an edge along the (unique) path in the tree T_x between these vertices. That is, all min-max distances in the graph G and in the minimum spanning T are *identical*. This result was re-discovered (with a more detailed proof) by Malpani and Chen [12, Theorem 2.1].

Our Theorem 1 does the *converse* reduction: it reduces the MST problem to the min-max distance problem.

Remark 2. That the MST problem is related to the min-max distances was observed already by Maggs and Plotkin [11]. They consider the case when weights of edges are distinct; hence, for every such weighting $x : E \to \mathbb{R}_+$, the minimum weight spanning tree T_x is unique. They show that then $T_x = \{e \in E : \mathrm{dist}_x(e) = x(e)\}$. This result also gives a $(\min, \max, +)$ DP algorithm for the MST problem: use the Floyd–Warshall DP algorithm to compute the min-max distances $\mathrm{dist}_x(e)$ of all edges e, and then sum up the weights of all edges for which $\mathrm{dist}_x(e) = x(e)$ holds.

The main difference of this algorithm from that given by Theorem 2 (besides the restriction to distinct weights, which is not crucial) is that it essentially uses *conditional branchings*: if $\mathrm{dist}_x(e) = x(e)$ then accept e else reject e. Thus, the DP algorithm in [11] is *not* a pure DP algorithm. In contrast, our algorithm uses no conditional branchings: it just performs

$(\min, \max)$ operations to compute the min-max distances of $n-1$ edges (of one, fixed in advance, spanning tree), and then just uses $+$ operations to output the sum of these values. Thus, Theorem 2 removes the need of conditional branchings in the Maggs–Plotkin DP algorithm, and does this without increasing the total number of performed operations.

3. Proof of Theorem 1

Since each next weighting in Theorem 1 sets the weight of one single edge to zero, it is enough to consider what happens after each such setting.

Lemma 1. *Let $G = (V, E)$ be an undirected connected graph. Then for every weighting $x : E \to \mathbb{R}_+$, and for every edge $e \in E$, we have*

$$\mathrm{mst}(x) = \mathrm{mst}(x') + \mathrm{dist}_x(e)\,, \tag{1}$$

where $x' : E \to \mathbb{R}_+$ is the weighting obtained from x by giving zero weight to the edge e, and leaving other weights unchanged.

Lemma 1 immediately yields Theorem 1 because after the weights of all edges $e_1, \ldots, e_{n-1}$ of the tree T are set to zero, we have an optimal spanning tree T of *zero* weight, that is, $\mathrm{mst}(x_{n-1}) = x_{n-1}(T) = 0$; recall that all weights are *nonnegative*.

Proof of Lemma 1. We prove (1) by showing the inequalities

$$\mathrm{mst}(x') \leq \mathrm{mst}(x) - \mathrm{dist}_x(e) \tag{2}$$

and

$$\mathrm{mst}(x) \leq \mathrm{mst}(x') + \mathrm{dist}_x(e) \tag{3}$$

separately. To show (2), let T be a spanning tree of G of minimal x-weight. If $e \in T$, then $x'(T) = x(T) - x(e)$. Since $x(e) \geq \mathrm{dist}_x(e)$ and $x(T) = \mathrm{mst}(x)$, inequality (2) trivially holds in this case.

Assume now that $e \notin T$. We claim that there is an edge $f \in T$ of weight $x(f) \geq \mathrm{dist}_x(e)$ such that $T^* = T - f + e$ is a spanning tree of G. To show this, take the (unique) path P in the tree T between the endpoints of e. Let $f \in T$ be an edge of that path of maximal weight $x(f)$. By the definition of $\mathrm{dist}_x(e)$, *every* path between the endpoints of e must contain an edge of x-weight at least $\mathrm{dist}_x(e)$. Hence, $x(f) \geq \mathrm{dist}_x(e)$. The removal of the edge f from T cuts the tree T into two connected components. Since the set $P + e$ forms a cycle, the edge e lies between these two components. Thus, $T^* = T - f + e$ is a spanning tree of G, and inequality (2) follows:

$$\begin{aligned} \mathrm{mst}(x') &\leq x'(T^*) = x'(T) - x'(f) + x'(e) \\ &= x(T) - x(f) \leq x(T) - \mathrm{dist}_x(e) \\ &= \mathrm{mst}(x) - \mathrm{dist}_x(e)\,. \end{aligned}$$

To show (3), we use the fact that the x'-weight $x'(e) = 0$ of the edge e is the *smallest* possible weight (all weights are nonnegative). So, $e \in T$ holds for at least one spanning tree T of G of minimal x'-weight; fix such a tree T.

We claim that there is an edge f of G of weight $x(f) \leq \mathrm{dist}_x(e)$ such that $T^* = T - e + f$ is a spanning tree of G. Indeed, by the definition of $\mathrm{dist}_x(e)$, there is a path P in G between the endpoints of e such that $x(f) \leq \mathrm{dist}_x(e)$ holds for all edges $f \in P$. The path P does not need to lie in the tree T, but at least one edge $f \in P$ must cross the cut induced by the edge e of T, that is, must lie between the two connected components of T after the edge e is removed. Thus, $T^* = T - e + f$ is also a spanning tree of G.

So, since $x'(T) = x(T - e)$ holds, inequality (3) follows:

$$\begin{aligned} \mathrm{mst}(x) &\leq x(T^*) = x(T-e) + x(f) = x'(T) + x(f) \\ &\leq x'(T) + \mathrm{dist}_x(e) = \mathrm{mst}(x') + \mathrm{dist}_x(e)\,. \end{aligned} \qquad \square$$

4. Proof of Theorem 2

Let $G = (V, E)$ be an undirected connected graph with $V = \{1, 2, \ldots, n\}$. Our goal is to show that the MST problem on G can be solved by a pure DP algorithm performing $O(n^3)$ $(\min, \max, +)$ operations.

First, we can easily reduce the MST problem on G to the MST problem on the complete graph K_n on V. For an input weighting $x : E \to \mathbb{R}_+$, compute the maximum weight $M = \max\{x(e)\colon e \in E\}$ with $|E| - 1 = O(n^2)$ max operations. Then give the weight M to every non-edge of G. Under the resulting weighting $y : K_n \to \mathbb{R}_+$, we have $\mathrm{mst}(x) = \mathrm{mst}(y)$. What we achieved is that now *all* pairs of distinct vertices, not only the edges of G, are weighted edges.

Now, given a weighting $x : K_n \to \mathbb{R}_+$, the *max-length* of a walk is the weight of its heaviest edge. Hence, the min-max distance $\mathrm{dist}_x(e)$ of an edge $e = \{i, j\}$ is the minimal max-length of a walk between i and j. Note that this minimum will always be achieved on some simple *path* between i and j: every walk between i and j contains a path between i and j. The min-max distances $\mathrm{dist}_x(e)$ of *all* edges e of K_n can be simultaneously computed by the Floyd–Warshall DP algorithm [3, 15] as follows.

A *k-walk* is a walk using only vertices from $\{1, \ldots, k\}$ as inner vertices. As subproblems, we take $D^k_{i,j}$ = the minimum max-length over all k-walks P between vertices i and j. Initial values are the weights $D^0_{i,j} = x(i, j)$ of the edges $\{i, j\}$ of K_n. Every k-walk between i and j either does not go through the vertex k, or does. So, the recurrence is:

$$D^k_{i,j} = \min\left\{D^{k-1}_{i,j},\ \max\{D^{k-1}_{i,k}, D^{k-1}_{k,j}\}\right\}$$

Then $D^n_{i,j} = \mathrm{dist}_x(i, j)$ is the min-max distance between i and j. Hence, all min-max distances $\mathrm{dist}_x(i, j)$ can be simultaneously computed with $N = O(n^3)$ min and max operations.

According to Theorem 1, we only have to compute min-max distances $\mathrm{dist}_{x_0}(e_1), \ldots, \mathrm{dist}_{x_{n-2}}(e_{n-1})$ of $n-1$ edges $e_1, \ldots, e_{n-1}$ (of a fixed spanning tree T), and add them together. This gives us a pure DP algorithm solving the MST problem on any n-vertex graph by performing $O(nN + n - 1) = O(n^4)$ $(\min, \max, +)$ operations.

But, since in our case each next weighting differs from the previous one on only one edge, we can reduce the total number of operations to $O(n^3)$. Compute all min-max distances $\mathrm{dist}_x(i, j)$ under the initial weighting x using the Floyd–Warshall algorithm, as above. After that, it is enough just to update these weights. Namely, the next to x weighting x' only sets the weight of one edge $e = \{a, b\}$ to 0, and leaves the weights of other edges unchanged.

Every path from a vertex i to a vertex j either goes through the edge e, or not. If a path of minimal x'-max-length does not go through e, then $\mathrm{dist}_{x'}(i, j) = \mathrm{dist}_x(i, j)$. If a path of minimal x'-max-length goes through e, then $\mathrm{dist}_{x'}(i, j)$ is the minimum of $\max(\mathrm{dist}_x(i, a), \mathrm{dist}_x(b, j))$

and $\max(\text{dist}_x(i,b), \text{dist}_x(a,j))$, because the edge $e=\{a,b\}$ can be entered from both its endpoints. Thus, $\text{dist}_{x'}(i,j)$ is the minimum of $\text{dist}_x(i,j)$ and $\max(\text{dist}_x(i,a), \text{dist}_x(b,j))$ and $\max(\text{dist}_x(i,b), \text{dist}_x(a,j))$.

We thus can compute the min-max distances between all pairs of vertices under the next to x weighting x' performing only $K = O(n^2)$ additional $(\min, \max)$ operations. Since we only have to update the distances $n-2$ times, the total number of performed operations is $N + (n-2)K + n - 1 = O(n^3)$. □

References

[1] R. Bellman. On a routing problem. *Quarterly of Appl. Math.*, 16:87–90, 1958.

[2] S.E. Dreyfus and R.A. Wagner. The Steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[3] R.W. Floyd. Algorithm 97, shortest path. *Comm. ACM*, 5:345, 1962.

[4] S. Fomin, D. Grigoriev, and G. Koshevoy. Subtraction-free complexity, cluster transformations, and spanning trees. *Found. Comput. Math.*, 15:1–31, 2016.

[5] L.R. Ford. Network flow theory. Technical Report P-923, The Rand Corp., 1956.

[6] M. Held and R.M. Karp. A dynamic programming approach to sequencing problems. *SIAM J. on Appl. Math.*, 10:196–210, 1962.

[7] T.C. Hu. The maximum capacity route problem. *Oper. Res.*, 9:898–900, 1961.

[8] M. Jerrum and M. Snir. Some exact complexity results for straight-line computations over semirings. *J. ACM*, 29(3):874–897, 1982.

[9] S. Jukna and H. Seiwert. Greedy can beat pure dynamic programming. *Inf. Process. Lett.*, 142:90–95, 2019.

[10] A.Y. Levin. Algorithm for the shortest connection of a group of graph vertices. *Sov. Math. Dokl.*, 12:1477–1481, 1971.

[11] B.M. Maggs and S.A. Plotkin. Minimum-cost spanning tree as a path-finding problem. *Inf. Process. Lett.*, 26(6):291–293, 1988.

[12] N. Malpani and J. Chen. A note on practical construction of maximum bandwidth paths. *Inf. Process. Lett.*, 83(3):175–180, 2002.

[13] E.F. Moore. The shortest path through a maze. In *Proc. Internat. Sympos. Switching Theory*, volume II, pages 285–292, 1957.

[14] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Information Theory*, 13(2):260–269, 1967.

[15] S. Warshall. A theorem on boolean matrices. *J. ACM*, 9:11–12, 1962.