Optimization Letters
https://doi.org/10.1007/s11590-024-02170-5

ORIGINAL PAPER

Efficient online sensitivity analysis for the injective bottleneck path problem

Kirill V. Kaymakov[1] · **Dmitry S. Malyshev**[2]

Received: 24 August 2024 / Accepted: 3 November 2024

Abstract

The tolerance of an element of a combinatorial optimization problem with respect to a given optimal solution is the maximum change, i.e., decrease or increase, of its cost, such that this solution remains optimal. The bottleneck path problem, for given an edge-capacitated graph, a source, and a target, is to find the max-min value of edge capacities on paths between the source and the target. For any given sample of this problem with n vertices and m edges, there is known the Ramaswamy-Orlin-Chakravarty's algorithm to compute an optimal path and all tolerances with respect to it in $O(m + n \log n)$ time. In this note, for any in advance given (n, m)-network with distinct edge capacities and k source-target pairs, we propose an $O\Big(m\alpha(m, n) + \min\big((n + k) \log n, km\big)\Big)$-time preprocessing, where $\alpha(\cdot, \cdot)$ is the inverse Ackermann function, to find in $O(k)$ time all $2k$ tolerances of an arbitrary edge with respect to some max min paths between the paired sources and targets. To find both tolerances of all edges with respect to those optimal paths, it asymptotically improves, for some n, m, k, the Ramaswamy-Orlin-Chakravarty's complexity $O\big(k(m + n \log n)\big)$ up to $O(m\alpha(n, m) + km)$.

Keywords Bottleneck path problem · Sensitivity analysis · Efficient algorithm

The work of the author Malyshev D.S. was conducted within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE).

✉ Dmitry S. Malyshev
dsmalyshev@rambler.ru

Kirill V. Kaymakov
kirill.kaymakov@mail.ru

[1] Coleman Tech LLC, 40 Mira Avenue, Moscow 129090, Russia

[2] Laboratory of Algorithms and Technologies for Networks Analysis, National Research University Higher School of Economics, 136 Rodionova Str., Nizhny Novgorod 603093, Russia

Published online: 20 November 2024

1 Introduction

1.1 Tolerances: background and significance

First of all, we present known definitions and results, concerning combinatorial optimization problems and their sensitivity analysis, following definitions and notations from [17, 34]. A *combinatorial minimization problem* is a problem of selecting a group of elements from a ground set, such that this combination yields a feasible solution and has the lowest cost among all feasible solutions. CMPs can have different objectives. The *sum objective function* minimizes the sum of the costs, whenever the *bottleneck* (or min-max) *objective function* minimizes the maximum cost among all elements in the solution. These CMPs are called *additive* and *bottleneck* (or min -max), respectively. There are many concrete additive and bottleneck CMPs, see, for example, [1, 2, 8, 10, 24].

More formally, following [34], an instance of a CMP is given by a tuple $(\mathcal{E}, F, c, f_c)$, where $\mathcal{E}$ is the finite *ground set of elements*, $F \subseteq 2^{\mathcal{E}} \setminus \{\emptyset\}$ is the *set of feasible solutions*, $c : \mathcal{E} \longrightarrow \mathbb{R}$ is the *cost function*, $f_c : F \longrightarrow \mathbb{R}$ is the *objective function*. A feasible solution $S^* \in F$ is called an *optimal solution* of the CMP if $f_c(S^*) = \min_{S \in F} f_c(S)$. The function $f_c(\cdot)$ is the *sum objective function* if $f_c(S) = \sum_{e \in S} c(e)$, and it is the *bottleneck objective function* if $f_c(S) = \max_{e \in S} c(e)$. Note that CMPs with these objectives are equivalent to the maximization ones, more precisely, to a max-min problem if those CMP is of the min-max type, by taking the costs with the opposite sign.

Sensitivity analysis is the field of finding limit cost changes, for which an optimal solution remains optimal. More specifically, upper tolerances measure the supremum increase in the cost of an element, such that a current solution remains optimal, and lower tolerances measure the corresponding supremum decrease [17].

If $\Pi = (\mathcal{E}, F, c, f_c)$ is an instance of a combinatorial minimization or maximization problem, $e \in \mathcal{E}$, and $\alpha \in \mathbb{R}$ is some constant, then, by $\Pi_{e,\alpha} = (\mathcal{E}, F, c_{e,\alpha}, f_{c_{e,\alpha}})$, a problem is denoted with

$$c_{e,\alpha}(e) = c(e) + \alpha, c_{e,\alpha}(e') = c(e') \ \forall e' \neq e.$$

Let S^* be an optimal solution of Π and $e \in \mathcal{E}$. The *upper tolerance of* e (*with respect to* S^*) is defined as

$$u_{S^*}(e) = \sup\{\alpha \in \mathbb{R}_{\geq 0} \ : \ S^* \text{ is an optimal solution of } \Pi_{e,\alpha}\}.$$

The *lower tolerance of* e (*with respect to* S^*) is defined as

$$l_{S^*}(e) = \sup\{\alpha \in \mathbb{R}_{\geq 0} \ : \ S^* \text{ is an optimal solution of } \Pi_{e,-\alpha}\}.$$

By F_{+e} and F_{-e}, we denote the sets of those feasible solutions of Π that contain and do not contain e, respectively, i.e.

$$F_{+e} = \{S \in F \ : \ e \in S\}, \ F_{-e} = \{S \in F \ : \ e \notin S\}.$$

By $f_c(F)$,$f_c(F_{+e})$, and $f_c(F_{-e})$, we denote the optimal values of the objective function on F, F_{+e}, F_{-e}, i.e.

$$f_c(F) = \min_{S\in F} f_c(S),\ f_c(F_{+e}) = \min_{S\in F_{+e}} f_c(S),\ f_c(F_{-e}) = \min_{S\in F_{-e}} f_c(S) \text{ or}$$
$$f_c(F) = \max_{S\in F} f_c(S),\ f_c(F_{+e}) = \max_{S\in F_{+e}} f_c(S),\ f_c(F_{-e}) = \max_{S\in F_{-e}} f_c(S).$$

The following statement is proved in [17], see Theorems 4 and 15 of those paper:

Statement 1 Let $\Pi = (\mathcal{E}, F, c, f_c)$ be an instance of a combinatorial minimization/maximization problem, S^* be an optimal solution of Π, $e \in \mathcal{E}$ be an element of the ground set. If Π is an additive minimization problem, then

$$u_{S^*}(e) = f_c(F_{-e}) - f_c(F),\ \text{if } e \in S^*,\ u_{S^*}(e) = +\infty,\ \text{if } e \in \mathcal{E}\setminus S^*;$$
$$l_{S^*}(e) = f_c(F_{+e}) - f_c(F),\ \text{if } e \in \mathcal{E}\setminus S^*,\ l_{S^*}(e) = +\infty,\ \text{if } e \in S^*.$$

If Π is a min-max problem, then

$$u_{S^*}(e) = f_c(F_{-e}) - c(e),\ \text{if } e \in S^*,\ u_{S^*}(e) = +\infty,\ \text{if } e \in \mathcal{E}\setminus S^*;$$
$$l_{S^*}(e) = c(e) - f_c(F),\ \text{if } e \in \mathcal{E}\setminus S^* \text{ and } \min_{S\in F_{+e}} \max_{e'\in S\setminus\{e\}} c(e') < c(e),$$
$$l_{S^*}(e) = +\infty,\ \text{otherwise.}$$

If Π is an additive maximization problem, then

$$u_{S^*}(e) = f_c(F) - f_c(F_{+e}),\ \text{if } e \in \mathcal{E}\setminus S^*,\ u_{S^*}(e) = +\infty,\ \text{if } e \in S^*;$$
$$l_{S^*}(e) = f_c(F) - f_c(F_{-e}),\ \text{if } e \in S^*,\ l_{S^*}(e) = +\infty,\ \text{if } e \in \mathcal{E}\setminus S^*.$$

If Π is a max-min problem, then

$$u_{S^*}(e) = f_c(F) - c(e),\ \text{if } e \in \mathcal{E}\setminus S^* \text{ and } \max_{S\in F_{+e}} \min_{e'\in S\setminus\{e\}} c(e') > c(e),$$
$$u_{S^*}(e) = +\infty,\ \text{otherwise;}$$
$$l_{S^*}(e) = c(e) - f_c(F_{-e}),\ \text{if } e \in S^*,\ l_{S^*}(e) = +\infty,\ \text{if } e \in \mathcal{E}\setminus S^*.$$

It follows from Statement 1 that if a tolerance is finite, then it does not depend on an optimal solution. By the same reason, an element $e \in \mathcal{E}$ belongs to some optimal solution of an additive combinatorial minimization/maximization problem if and only if its lower/upper tolerance is $+\infty$.

Tolerances give to a decision maker information about the stability of an optimal solution with respect to perturbations of its elements. They are also used to design algorithms for NP-hard and polynomially solvable combinatorial problems, like versions of the Assignment Problem [13], Travelling Salesman Problem [16, 33, 35], Vehicle Routing Problem [4], Weighed Independent Set Problem [19], see the surveys [34, 35] also. Efficiency of tolerance computations is an important problem, which not only had applications in the mentioned problems, but it is also of independent interest. Several studies are devoted to this question [6, 11, 14, 18, 22, 26, 27, 29–31].

1.2 The bottleneck path problem

In this note, we consider sensitivity analysis for the *Bottleneck Path Problem*, abbreviated as the BPP. In this problem, a simple, connected graph $G = (V_G, E_G)$ with $V_G = \{v_1, v_2, \dots, v_n\}$ and $E_G = \{e_1, e_2, \dots, e_m\}$ is given and, for every edge e_i, its capacity c_i is also given. Additionally given a source vertex $s \in V_G$ and a target vertex $t \in V_G$, the BPP is to find the value $b_G(s,t) = \max\limits_{P \in \mathcal{P}_{st}} \min\limits_{e \in P} c(e)$, where $\mathcal{P}_{st}$ is the set of all paths between s and t. A *path* is a sequence of vertices, in which any two consecutive members form an edge, without repetitions of its vertices and edges. The BPP arises as a subroutine in several flow problems, see, for example, [3] and [15]. The *online* BPP is the problem, when sources s and targets t are entered in the online regime. To the best of our knowledge, this problem was introduced in [23], where it was named the *online Multi-Pair Bottleneck Paths Problem*. To solve the online BPP, an input edge-capacitated graph can be preprocessed in $O(m + n \log n)$ time, such that any bottleneck value $b_G(s,t)$ can be computed in $O(\log n)$ time, see [23].

1.3 Our contribution: results, motivation, and methods

In this note, for any in advance given $(G = (V_G, E_G), c)$ with injective $c(\cdot)$ and $(s_1, t_1), \dots, (s_k, t_k)$, where $s_i, t_i \in V_G$, for any i, we present a preprocessing in $O\Big(m\alpha(m,n) + \min\big((n+k)\log n, km\big)\Big)$ time, allowing to compute in $O(k)$ time all $2k$ tolerances of an arbitrary edge with respect to some max min $s_i t_i$-paths, for all i. We assume that the edge capacities are pairwise distinct, and this restriction is used in justification for correctness of our algorithm, see the reasonings before Algorithm 3.

Note that computation of upper and lower tolerances of all edges for the BPP, including determination of an optimal solution, can be done in $O(m + n \log n)$ time [29]. But the approach from [29] does not allow to compute rapidly values of tolerances for individual edges. Analysis of namely individual tolerances may be important, for example, in sustainability research of flows in networks under incidents with a single edges, like breakdowns or repairs of pipelines, possibly, with respect to several source-target pairs. Moreover, to find both tolerances of all edges with respect to some optimal $s_i t_i$-paths, our algorithm asymptotically improves, for some n, m, k (e.g., $m = O(n)$), the Ramaswamy-Orlin-Chakravarty's complexity $O\big(k(m + n \log n)\big)$ up to $O(m\alpha(n,m) + km)$.

We use the formulae from [31] on sensitivity analysis for the minimum spanning tree problem (the MSTP, for short) and the algorithms from [14, 27] with the complexities $O\big(m\alpha(m,n)\big)$ and $O\big(m \log \alpha(m,n)\big)$, respectively, to compute all tolerances for this problem. But our algorithm is not completely reduced to those sensitivity analysis, involving some additional arguments. We also propose an algorithm for sensitivity analysis of the minimum spanning tree problem, which is much simpler to understand, than those in [14, 27], but with the worse complexity $O(m \log n)$.

We also conducted computational experiments on synthetic networks, using $k = 1$ and our sensitivity analysis of the MSTP. They confirmed $O(m \log n)$- and $O(1)$-theoretical complexity guaranties for preprocessing and tolerance computation times.

2 Our algorithm

2.1 Bottleneck paths and maximum spanning trees

A tree subgraph, containing all vertices of a graph, is called a *spanning tree*. The *maximum spanning tree problem*, abbreviated as the MSTP, is to find a spanning tree with the maximum sum of weights of its edges in a given edge-weighted graph. Several classic algorithms were designed to solve the MSTP, like the Boruvka's [7], the Kruskall's [25], the Prim's [28], the Chazelle's [9] algorithms. For a graph with n vertices and m edges, the Prim's algorithm can be implemented in $O(m + n \log n)$ time, using Fibonacci heaps [21]. The Chazelle's algorithm is the best known algorithm among deterministic to solve the MSTP, it has the computational complexity $O\big(m\alpha(m, n)\big)$.

The MSTP has an important connection to the BPP, shown in Statement 2. This is a well-known fact, but we did not know a corresponding reference, thus, a formal proof was presented in [23].

Statement 2 Let T be an arbitrary maximum spanning tree of (G, c). Then, for any $s \in V_G, t \in V_G$, the minimum capacity of edges on the path between s and t in T equals $b_G(s, t)$.

Statement 2 makes to think of using the following strategy for computing the tolerances: monitor changes of a MST and minima in st-paths under capacity changes of individual edges. The Ramaswamy-Orlin-Chakravarty's algorithm explores it, and it is also be applied in our algorithm. Ramaswamy, Orlin, and Chakravarty directly compute max min values on st-paths in resulting trees, but we involve more sophisticated arguments.

2.2 Lowest common ancestors and some their applications

For a given rooted tree and its vertices x and y, the *lowest common ancestor* of x and y, abbreviated as $LCA(x, y)$, is the deepest node, for which both x and y are descendants, assuming that each vertex is a descendant of itself. It was shown in [20] that with a preprocessing step in $O(n)$ time, where n is the vertex number in the tree, LCA of any pair of vertices can be found in $O(1)$ time. The approach from [5], the so-called jump pointers algorithm with preprocessing time $O(n \log n)$, is modified in [23] for computing in $O(\log n)$ time the minimum and maximum values on in online given paths of an in advance given edge-weighted tree. It can also be easily modified to return optimal edges on these paths.

LCAs are also useful to solve the following task, which are used to compute the tolerances efficiently. Assume that we are given a tree T. For any vertices x and y, by $T(x, y)$, we denote the path between x and y in T. We need to preprocess T quickly enough, such that, for any given vertices s, t of T and its edge xy, checking whether $xy \in T(s, t)$ or not can also be performed rapidly.

It can be done in $O(1)$ time with an $O(n)$ preprocessing stage. To this end, T is rooted at an arbitrary vertex r and it is preprocessed, according to [20]. The *depth of a vertex $x \in V(T)$ with respect to r*, i.e., $|T(r, x)|$, is denoted by $d_T(x)$. By breadth-first search, we find all depths in $O(n)$ time. Denote by T_x and T_y the connected components of $T \setminus \{xy\}$, containing x and y, respectively. Without loss of generality, let us assume that $d_T(y) = d_T(x) + 1$. This means that x and r belong to T_x. Clearly that $xy \in T(s, t)$ if and only if s and t lie in distinct components. Checking the fact that a vertex $z \in \{s, t\}$ belongs to T_y can be done by checking the equality $LCA(z, y) = y$. The exactly one of s and t must poses this property to satisfy $xy \in T(s, t)$. It gives an $O(1)$-time algorithm to solve the task above.

2.3 Disjoint-set data structure

A *disjoint-set data structure*, DJS, for short, is a data structure that stores a partition of a finite set into its disjoint subsets. It supports the following operations:

- *Create*(x) — creating the new singleton subset $\{x\}$ and adding it to the structure,
- *Find*(x) — finding a canonical element of those subset, which contains x,
- *Join*(x, y) — replacing the two subsets with the canonical elements x and y by their union

in near-constant time. More precisely, insertion and join can be performed in unit time in the worst case, but search can be performed in amortized time, bounded from above by a value of the inverse Ackermann function, see [32]. DJSs are useful in efficient implementation, see, for example, [12], of the Kruskal's algorithm, which orders edges by their weights, scans their sorted set, and determines whether a current edge can be added to an optimal solution or not. We use a similar idea in our algorithm.

2.4 Efficient sensitivity analysis for the MSTP

Let T be an arbitrary maximum spanning tree of the graph (G, c). It can be computed in $O\big(m\alpha(m, n)\big)$ time [9]. It was proved, see the paper [31], that, for the MSTP on (G, c) and its optimal solution T, the following relations are true:

$$
\begin{aligned}
&u_T(xy) = +\infty, \text{ if } xy \in E_T;\ l_T(xy) = -\infty, \text{ if } xy \in E_G \setminus E_T;\\
&u_T(xy) = \min\left\{c(x'y') \,:\, x'y' \in T(x, y)\right\} - c(xy), \text{ if } xy \in E_G \setminus E_T;\\
&l_T(xy) = c(xy) - \max\left\{c(x'y') \,:\, xy \in T(x', y')\right\}, \text{ if } xy \in E_T.
\end{aligned}
$$

Based on these formulae, an algorithm is designed in [14] for computing all tolerances for the MSTP on (G, c), assuming that an optimal tree has been given. It has the worst-case complexity $O\big(m\alpha(m,n)\big)$, but with $O(m)$ randomized complexity. The result from [14] is updated to the $O\big(m\log\alpha(m,n)\big)$ complexity [27]. Together with computing all tolerances, it is possible to determine the following edges in $O\big(m\alpha(m,n)\big)$ or $O\big(m\log\alpha(m,n)\big)$ time:

$$\forall xy \in E_G \setminus E_T \quad \arg \min_{\{x'y' :\ x'y' \in T(x,y)\}} c(x'y'),$$

$$\forall xy \in E_T \quad \arg \max_{\{x'y' :\ xy \in T(x',y')\}} c(x'y').$$

Indeed, computed the corresponding arg mins/arg maxs $x'y'$, called the *replacement edges*, the trees $T' = \big(T \setminus \{x'y'\}\big) \cup \{xy\}$ and $T' = \big(T \setminus \{xy\}\big) \cup \{x'y'\}$ are the maximum spanning trees of (G, c) among its spanning trees, containing $xy \in E_G \setminus E_T$ or not containing $xy \in E_T$, respectively. For any edge of (G, c), its replacement edge is unique if any, as all capacities are pairwise distinct. These observations are useful for our aims, according to Statement 2.

The algorithms from [14, 27] are quite difficult to understand. Here, we present much simpler alternative LCA- and DJS-based algorithms with the worst-case complexities $O(m\log n)$. A pseudo code for determining the replacement edges, corresponding to MSTP upper tolerances, is presented in Algorithm 1. Its computational complexity is $O(m\log n)$, which is obvious.

Algorithm 1 MSTP upper tolerances replacement edges computation

Input: A simple, edge-weighted, connected graph $G = (V_G, E_G, c)$ and some its maximum spanning tree T
Output: An array $U[]$, whose indices are edges of G and elements are the corresponding replacement edges

Build the corresponding LCA-based data structure by [23];
for $(e = xy \in E_G)$ **do**
  if $(e \in E_T)$ **then**
    $U[e] \leftarrow$'No';
  end
  else
    $U[e] \leftarrow$ find the replacement edge for e by searching the arg min on $T(x, y)$;
  end
end
return $U[]$

Now, let us consider determining the replacement edges, corresponding to MSTP lower tolerances. Let T be any MST of (G, c), rooted at an arbitrary vertex. Sort edges from $E_G \setminus E_T$ by decreasing the capacities. Then, for any $e \in E_T$, its replacement edge is the first edge $xy \in E_G \setminus E_T$ with respect to the order with $e \in E_{T(x,y)}$. Scanning the ordered set $E_G \setminus E_T$, our algorithm catches the corresponding moments, for all edges in E_T.

Let us define a multigraph (G', c'), i.e., multiple edges are allowed. For any $xy \in E_G \setminus E_T$, change xy to the edges xz and yz, where $z = LCA(x, y)$, with the same capacity $c(xy)$. The tree T is a MST of (G', c'), see [14], and, obviously, any edge of (G', c') connects an ancestor with its descendant. Put $E' = E_{G'} \setminus E_T$.

Together with T, we keep a DJS on V_T, whose all elements, i.e., subsets of V_T, are vertices of some tree T'. Initially, DJS contains all n singletons, corresponding to vertices of T, and $T' = T$. At any moment, all vertices of T' are vertex sets of some subtrees of T and all its edges are exactly edges of T, for which replacement edges have not yet been determined. Reading an edge $xy \in E'$, by $Find(x)$ and $Find(y)$, we determine the vertices $X, Y \in V_{T'}$, such that $x \in X$ and $y \in Y$. Suppose that $X \neq Y$. The invariant of the process is that Y is a descendant of X, or vice versa, in T', determined by the descendant relation between x and y in T. Supposing that Y is a descendant of X in T, we contract $T'(X, Y)$ in T' and join the corresponding subsets of V_T into a single vertex in T'. Walking through $T'(X, Y)$, for any its edge with a capacity c^*, we assign e as the replacement edge of $e' \in E_T$ with $c^{-1}(e') = c^*$.

A pseudo code for determining the replacement edges, corresponding to MSTP lower tolerances, is presented in Algorithm 2. Its computational complexity is, clearly, $O(m \log n)$.

Algorithm 2 MSTP lower tolerances replacement edges computation

Input: A simple, edge-weighted, connected graph $G = (V_G, E_G, c)$ and some its maximum spanning tree T
Output: An array $L[]$, whose indices are edges of G and elements are the corresponding replacement edges

```
Build the corresponding data structure by [20];
Put V_{G'} = V_G;
for (e = xy ∈ E_G) do
    L[e] ←'No';
    z ← LCA(x, y);
    E_{G'} ← E_{G'} ∪ {xz, yz} with c'(xz) = c'(yz) = c(xy);
end
Put E' ← E_{G'} \ E_T and sort its elements in weights decreasing;
Build the corresponding DJS data structure and T';
for (e = xy ∈ E') do
    X = Find(x), Y = Find(y);
    if (X ≠ Y) then
        Put Z ← Y, Z_∪ ← Y;
        while (Z ≠ X) do
            c* ← c(Z, P'(Z)), where P'(·) is the parenthood relation on T';
            L[e'] ← e, where c^{-1}(e') = c*;
            Join Z_∪ and P'(Z) and contract ZP'(Z) in T';
            Z ← P'(Z);
        end
    end
end
return L[]
```

2.5 Combining all together

Suppose that some vertices s, t and an edge $e = xy$ are given in the graph (G, c). Let T be an arbitrary MST of (G, c). Let P^* be an arbitrary max min st-path of (G, c) and e^* be its edge $\arg\min_{e \in P^*} c(e)$. The edge e^* is unique, as $c(\cdot)$ is injective. It is clear that $u_{P^*}(e) = +\infty \ \forall e \in E_T$ and $l_{P^*}(e) = +\infty \ \forall e \in E_G \setminus E_T$, because the corresponding changes of $c(e)$ do not break the optimality of T as a MST. By the same reason, if a tolerance (upper or lower) of e is finite, then there is a replacement edge e' and this tolerance is at least $|c(e') - c(e)|$. It follows from Statement 1 that if $u_{P^*}(e) < +\infty$, then $u_{P^*}(e) = c(e^*) - c(e)$.

Let us assume that $e \in T(s, t)$, i.e., $e \in E_T$, which can be checked in constant time, see Subsect. 2.2. If $L[e]$ ='No', i.e., removing e disconnects (G, c), then $l_{P^*}(e) = +\infty$, because any decrease of $c(e)$ keeps the optimality of T for the MSTP and the resultant graph. Suppose that there is an edge $e' = x'y' = L[e]$. It is clear that $e' \notin E_T$. Then, by Statements 1 and 2, we have $l_{P^*}(e) = c(e) - \min\left(c(e'), c(e^*)\right)$, since

$$\min_{\tilde{e} \in T(x', y') \cup \{e'\}} c(\tilde{e}) = c(e'), \text{ where } T' = \left(T \setminus \{e\}\right) \cup \{e'\}, \text{ by the maximality of } T,$$
$$\min_{\tilde{e} \in T(s,t)} c(\tilde{e}) = c(e^*).$$

Suppose that $e = xy \notin T(s, t)$. Then, $l_{P^*}(e) = +\infty$, as $T(s, t)$ exists in a new tree after any decrease of $c(e)$. If $U[e]$ ='No', then $u_{P^*}(e) = +\infty$. Suppose that there is an edge $e' = U[e]$. Then, $e \notin E_T, e' \in E_T$. If $e' \notin T(s, t)$, which can be verified in $O(1)$ time, see Subsect. 2.2, then $u_{P^*}(e) = +\infty$. Indeed, for any increase of $c(e)$, either T or $T' = \left(T \setminus \{e'\}\right) \cup \{e\}$ is an optimal solution of the MSTP. The path $T(s, t)$ exists in both these trees. Suppose that $e' \in T(s, t)$. Then, $u_{P^*}(e) = c(e^*) - c(e)$ if $c(e') = c(e^*)$, otherwise $u_{P^*}(e) = +\infty$. Indeed, we have

$$u_{P^*}(e) \geq c(e') - c(e) \geq c(e^*) - c(e), \text{ as } c(e') \geq c(e^*),$$
$$u_{P^*}(e) > c(e^*) - c(e) \longrightarrow u_{P^*}(e) = +\infty,$$

and since $c(\cdot)$ is injective and $e' = \arg\min_{\left\{\tilde{e}:\ \tilde{e} \in T(x,y)\right\}} c(\tilde{e})$ we have

$$c(e') = c(e^*) \longleftrightarrow e' = e^* \longleftrightarrow e^* \notin T'(s, t),$$
$$e^* \in T'(s, t) \longrightarrow \arg\min_{\tilde{e} \in T'(s,t)} c(\tilde{e}) \in \{e, e^*\} \longrightarrow u_{P^*}(e) = +\infty.$$

A pseudo code of an algorithm for working with a pair of a source and a target is presented in Algorithm 3. Its preprocessing stage, emphasized with the underlines, takes $O\left(m\alpha(m, n)\right)$ time. Its running time is $O(1)$ per an edge.

Algorithm 3 BPP online tolerances computation

Input: A simple, edge-capacitated, connected graph $G = (V_G, E_G, c)$, vertices $s, t \in V$ (in advance) and an edge $e = xy \in E$ (in online)
Output: The lower and upper tolerances of e with respect to a max min path between s and t

Find a maximum spanning tree T of G by [9];
Find a max min st-path P^* and its edge $e^* = \arg\min\limits_{e \in P^*} c(e)$;
Build the corresponding data structure by [20];
$(U[], L[]) \leftarrow$ apply the algorithm from [27];
if $(e \in E_{T(s,t)})$ **then**
    $u_{P^*}(e) \leftarrow +\infty$;
    if $(L[e] = \text{'No'})$ **then**
        $l_{P^*}(e) \leftarrow +\infty$;
    end
    else
        $e' \leftarrow L[e]$;
        $l_{P^*}(e) \leftarrow c(e) - \min\left(c(e'), c(e^*)\right)$;
    end
end
else
    $l_{P^*}(e) \leftarrow +\infty$;
    if $(U[e] = \text{'No'})$ **then**
        $u_{P^*}(e) \leftarrow +\infty$;
    end
    else
        $e' \leftarrow U[e]$;
        if $\left(e' \notin T(s,t) \vee c(e') \neq c(e^*)\right)$ **then**
            $u_{P^*}(e) \leftarrow +\infty$;
        end
        else
            $u_{P^*}(e) \leftarrow c(e^*) - c(e)$;
        end
    end
end
return $l_{P^*}[e], u_{P^*}[e]$

Algorithm 3 can be modified to work with k pairs $(s_1, t_1), \ldots, (s_k, t_k)$ of sources and targets. To this end, we find all $e_i^* = \arg\min_{e \in P_i^*} c(e)$, where P_i^* is a max min $s_i t_i$-path, either in $O(km)$ time or in $O(n \log n + k \log n)$ time, using LCA-based approach from Subsect. 2.2. Hence, all $2k$ tolerances of any given edge with respect to P_1^*–P_k^* can be computed in $O(k)$ time under $O\Big(m\alpha(m,n) + \min\left((n+k)\log n, km\right)\Big)$-time preprocessing. It gives an $O(m\alpha(m,n) + km)$-time algorithm for computing both tolerances of all edges with respect to P_1^*–P_k^*, sometimes improving the Ramaswamy-Orlin-Chakravarty's complexity $O\big(k(m + n \log n)\big)$, e.g., when $k = 1$ and $m = O(n)$.

3 Computational experiments

In order to verify the proposed algorithm, its software implementation was carried out, see

https://github.com/Kirundel/PHD/tree/main/SensitivityAnalysis.

An interested reader can independently justify the correctness of our algorithm on small networks. Here, we describe conditions and results of the conducted experiments to evaluate its performance. To generate growing networks for experiments, a simple cycle on vertices $1, 2, \dots, n$ was used, to which $m - n$ pairwise distinct edges ab were added, where different a and b were independently pseudo-randomly generated from the discrete uniform distribution on $\{1, 2, \dots, n\}$. Values for s and t and an edge e were selected in the same way. Edge capacities were generated independently in a pseudo-random manner with the uniform distribution on [0, 1]. Replacement edges were found by Algorithms 1 and 2. It was considered $n = 2^p$, where $5 \le p \le 15$, and $m \in \{2n, n(\log n)^2, \lfloor n^{1.5} \rfloor\}$.

Computational experiments were conducted to justify the complexity guaranties for the preprocessing and tolerance computation phases of Algorithm 3. They were made on a machine with a 4-core Intel Core i7-7700hq processor of the frequency 2.8 GHz and 24 Gb of RAM. They showed the following results, which confirm theoretical guaranties for the complexity, independently of the considered graph densities (Figs. 1 and 2):

4 Conclusion and future work

In this note, we considered the bottleneck path and sensitivity analysis problems in the form of tolerances computation for individual edges with respect to an optimal solution. The previous state-of-the-art algorithm, due to Ramaswamy, Orlin, and Chakravarty, computes an optimal solution and tolerances with respect to it in $O(m + n \log n)$ time. In this note, for any in advance given distinct-capacities

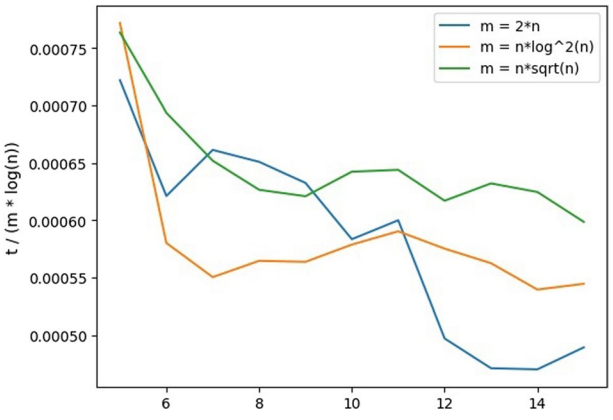

Fig. 1 Ratio of preprocessing time to $m \log_2 n$

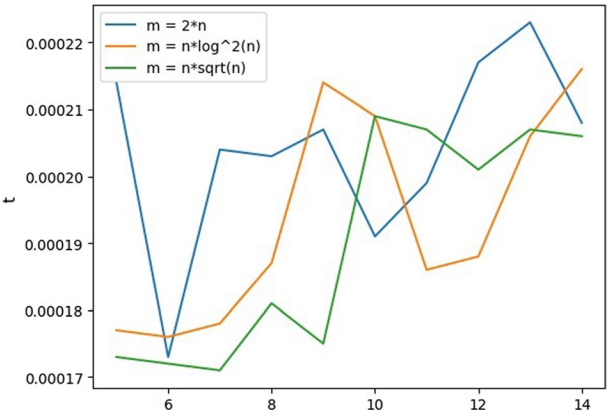

Fig. 2 Tolerances computation time

network and k source-target pairs, we propose an $O\Big(m\alpha(m,n) + \min\big((n+k)\log n, km\big)\Big)$-time preprocessing to find in $O(k)$ time all $2k$ tolerances of an arbitrary edge with respect to some max min paths between the paired sources and targets. To compute both tolerances of all edges with respect to those optimal paths, it asymptotically improves, for some n, m, k, the Ramaswamy-Orlin-Chakravarty's complexity $O\big(k(m + n\log n)\big)$ up to $O(m\alpha(n,m) + km)$. Conducted experiments with usage of our sensitivity analysis for the maximum spanning tree problem justified on synthetic data all the complexity guaranties. Developing new algorithms and improving existing ones is a challenging research problem for future work.

Author contributions All authors contributed to the study conception and design. Material preparation was performed by Kirill Kaymakov and Dmitriy Malyshev. The first draft of the manuscript was written by Kirill Kaymakov and Dmitriy Malyshev and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding The work of the author Malyshev D.S. was conducted within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE).

Data availability Our note has no associated data.

References

1. Aissi, H., Bazgan, C., Vanderpooten, D.: Min-max and min-max regret versions of combinatorial optimization problems: a survey. Eur. J. Oper. Res. **197**, 427–438 (2009)
2. Applegate, D., Cook, W., Dash, S., Rohe, A.: Solution of a min-max vehicle routing problem. INFORMS J. Comput. **14**, 132–143 (2002)
3. Baier, G., Köhler, E., Skutella, M.: On the *k*-splittable flow problem. In: Möhring, R., Raman, R. (eds.). Proceedings of European Symposium on Algorithms, 101–113, Springer (2002)
4. Batsyn, M., Goldengorin, B.I., Kocheturov, A., Pardalos, P. Tolerance-based versus cost-based branchng for the asymmetric capacitated vehicle routing problem. In: Goldengorin, B., Kalyagin, V.,

Pardalos, P. (eds.). Proceedings of the Second International Conference on Network Analysis, 1–10 Springer (2012)
5. Bender, M., Farach-Colton, M.: The level ancestor problem simplified. Theoret. Comput. Sci. **321**, 5–12 (2004)
6. Booth, H., Westbrook, J.: A linear algorithm for analysis of minimum spanning and shortest-path trees of planar graphs. Algorithmica **11**, 341–352 (1994)
7. Boruvka, O.: About a certain minimal problem. In: Proceedings of the Moravian Society of Natural Sciences. **3**, 37–58 (1926)
8. Burkard, R., Dell'Amico, M., Martello, S.: Assignment problems. SIAM, Philadelphia (2009)
9. Chazelle, B.: A minimum spanning tree algorithm with inverse-Ackermann type complexity. J. ACM. **47**, 1028–1047 (2000)
10. Cherkassky, B., Goldberg, A., Radzik, T.: Shortest paths algorithms: theory and experimental evaluation. Math. Programming Series A. **73**, 129–174 (1996)
11. Chin, F., Houck, D.: Algorithms for updating minimal spanning trees. J. Comput. Syst. Sci. **16**, 333–344 (1978)
12. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to algorithms, 4th edn. The MIT Press, Cambridge (2022)
13. Dell'Amico, M., Toth, P.: Algorithms and codes for dense assignment problems: the state of the art. Discret. Appl. Math. **140**, 17–48 (2004)
14. Dixon, B., Rauch, M., Tarjan, R.: Verification and sensitivity analysis of minimum spanning trees in linear time. SIAM J. Comput. **21**, 1184–1192 (1992)
15. Edmonds, J., Karp, R.: Theoretical improvements in algorithmic efficiency for network flow problems. J. ACM. **19**, 264–284 (1972)
16. Germs, R., Goldengorin, B., Turkensteen, M.: Lower tolerance-based branch and bound algorithms for the ATSP. Comput. Oper. Res. **39**, 291–298 (2012)
17. Goldengorin, B., Jäger, G., Molitor, P.: Tolerances applied in combinatorial optimization. J. Comput. Sci. **2**, 716–734 (2006)
18. Goldengorin, B., Malyshev, D., Pardalos, P.: Efficient computation of tolerances in weighted independent set problems for trees. Dokl. Math. **87**, 368–371 (2013)
19. Goldengorin, B., Malyshev, D., Pardalos, P., Zamaraev, V.: A tolerance-based heuristic approach for the weighted independent set problem. J. Comb. Optim. **29**, 433–450 (2015)
20. Fischer, J., Heun, V.: Theoretical and practical improvements on the RMQ-problem, with applications to LCA and LCE. In: Lewenstein, M., Valiente, G. (eds.). Proceedings of the 17th Annual Symposium on Combinatorial Pattern Matching, 36–48 (2006)
21. Fredman, M., Tarjan, R.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM. **34**, 596–615 (1987)
22. Jönker, R., Volgenant, A.: Improving the Hungarian assignment algorithm. Oper. Res. Lett. **5**, 171–175 (1986)
23. Kaymakov, K., Malyshev, D.: On efficient algorithms for bottleneck path problems with many sources. Optim. Lett. **18**, 1273–1283 (2024)
24. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack problems. Springer, Berlin (2004)
25. Kruskal, J.: On the shortest spanning subtree of a graph and the traveling salesman problem. In: Proceedings of the American Mathematical Society. **7**, 48–50 (1956)
26. Malyshev, D., Pardalos, P.: Efficient computation of tolerances in the weighted independent set problem for some classes of graphs. Dokl. Math. **89**, 253–256 (2014)
27. Pettie, S.: Sensitivity analysis of minimum spanning trees in sub-inverse-Ackermann time. J. Graph Algorithms Appl. **19**, 375–391 (2015)
28. Prim, R.: Shortest connection networks And some generalizations. Bell Syst. Tech. J. **36**, 1389–1401 (1957)
29. Ramaswamy, R., Orlin, J., Chakravarty, N.: Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs. Math. Programming Series A. **102**, 355–369 (2005)
30. Shier, D., Witzgall, C.: Arc tolerances in minimum-path and network flow problems. Networks **10**, 277–1980 (1980)
31. Tarjan, R.: Sensitivity analysis of minimum spanning trees and shortest path trees. Inf. Process. Lett. **14**, 30–33 (1982)
32. Tarjan, R., van Leeuwen, J.: Worst-case analysis of set union algorithms. J. ACM **31**, 245–281 (1984)

33. Turkensteen, M., Ghosh, D., Goldengorin, B., Sierksma, G.: Tolerance-based branch and bound algorithms for the ATSP. Eur. J. Oper. Res. **189**, 775–788 (2008)
34. Turkensteen, M., Jäger, G.: Efficient computation of tolerances in the sensitivity analysis of combinatorial bottleneck problems. Theoret. Comput. Sci. **937**, 1–21 (2022)
35. Turkensteen, M., Malyshev, D., Goldengorin, B., Pardalos, P.: The reduction of computation times of upper and lower tolerances for selected combinatorial optimization problems. J. Global Optim. **68**, 601–622 (2017)