Global Sensitivity Analysis for Bottleneck Assignment Problems

Elad Michael\textsuperscript{a,∗}, Tony A. Wood\textsuperscript{a}, Chris Manzie\textsuperscript{a}, Iman Shames\textsuperscript{b}

\textsuperscript{a}Department of Electrical and Electronic Engineering, University of Melbourne, Parkville, Victoria, 3010, Australia
\textsuperscript{b}School of Engineering, Australian National University, Acton, ACT, 0200, Australia

Abstract

In assignment problems, decision makers are often interested in not only the optimal assignment, but also the sensitivity of the optimal assignment to perturbations in the assignment weights. Typically, only perturbations to individual assignment weights are considered. We present a novel extension of the traditional sensitivity analysis by allowing for simultaneous variations in all assignment weights. Focusing on the bottleneck assignment problem, we provide two different methods of quantifying the sensitivity of the optimal assignment, and present algorithms for each. Numerical examples as well as a discussion of the complexity for all algorithms are provided.

Keywords: Assignment, Robustness and sensitivity analysis, Bottleneck

1. Introduction

In the classic assignment problem, we seek a one to one matching of a set of agents to a set of tasks which optimises some assignment cost. For example, Nam and Shell (2015) consider a vehicle routing problem, assigning destinations to vehicles in order to minimise the time taken for all vehicles to reach their destinations within a road network. Assignment problems have a wide variety of applications, such as resource allocation (Harchol-Balter et al. (1999)), scheduling (Carraresi and Gallo (1984), Adams et al. (1988)), and

\textsuperscript{∗}Corresponding author

Email addresses: elad.michael@gmail.com (Elad Michael), wood.t@unimelb.edu.au (Tony A. Wood), manziec@unimelb.edu.au (Chris Manzie), iman.shames@anu.edu.au (Iman Shames)

Preprint
the aforementioned autonomous vehicle routing (Shames et al. (2017), Nam and Shell (2015), Arslan et al. (2007)). Pentico (2007) review a variety of assignment objectives, such as the sum of all the assigned weights (linear assignment) or the difference between the maximum and minimum weight assignment (balanced assignment problem). Here we focus on the bottleneck assignment problem (BAP), which minimises the maximum assigned weight within the assignment. The BAP is particularly applicable in minimum time scenarios, ensuring that a team of agents working in parallel complete their tasks in minimum time. For the bottleneck assignment problem there are many algorithms which provide solution in polynomial time, reviewed in Pentico (2007) and Pferschy (1997).

In many applications the weight associated with the assignment of an agent to a task carries with it some uncertainty. Measurement errors, communication delays, or a dynamic environment may contribute to an inaccurate estimate of an assignment weight. If the uncertainty is known a-priori a minimax approach may be used, identifying an assignment which minimises the assignment cost for the worst case realisation of the uncertainty. However, this is typically a conservative solution. In applications where such a-priori knowledge is unavailable/impractical, or a conservative solution undesirable, we instead employ sensitivity analysis.

Sensitivity analysis characterises how perturbations to the inputs affect the output. In the case of uncertainty in the assignment weights, a sensitivity analysis informs the decision maker in which ways the optimal assignment may change, given perturbations to the measured weights. We adopt Type II sensitivity analysis from Koltai and Terlaky (2000), or equivalently the sensitivity of the optimal basis as in Jansen et al. (1997). Type II sensitivity analysis characterises the set of weight perturbations to which the optimal assignment is invariant. In the sensitivity analysis of linear programs, as in Jansen et al. (1997), the perturbations are restricted to an individual element of the canonical weight vector, while the remaining elements are held fixed. However, due to the additional structure of a BAP, we are able to extend the sensitivity analysis to simultaneous perturbations in all of the weights. We consider this a more applicable formulation of assignment sensitivity, as perturbations to the measured weights occur simultaneously if tasks are carried out in parallel.
1.1. Literature Review

Due to the assignment problem’s popularity in network theory, there exist various methods incorporating uncertainty in the assignment weights. Roughly characterised, we have those which compute robust assignments with \textit{a priori} knowledge of uncertainty, and those which characterise some notion of sensitivity of the assignment to perturbations. Volgenant and Duin (2010) presents several robust optimisation approaches to the bottleneck assignment problem, with interval assumptions on the possible weight values. Fu et al. (2015) and Kasperski and Zieliński (2013) begin with a set of possible scenarios, and compute robust minimax assignment solutions by blending those scenarios. Without \textit{a priori} knowledge of the uncertainty, Lin and Wen (2003) and Volgenant (2006) present the sensitivity of the linear assignment to perturbations in a single weight, based on the dual variables arising from a primal-dual optimisation. Ramaswamy et al. (2005) provide similar results in the shortest path and maximum cut network problems, which can be shown to be equivalent formulations to the linear assignment problem. However, they do not consider simultaneous weight perturbations in multiple edges. Both Nam and Shell (2015) and Lin and Wen (2007) present sensitivity analysis algorithms for the linear assignment, allowing for perturbations in an assigned edge as well as in edges adjacent to the assigned edge (i.e., a row/column of the weight matrix). However, these analyses do not take into account simultaneous perturbations in multiple assigned edges. Wood et al. (2020) present an assignment and collision avoidance algorithm, which utilises a bottleneck assignment sensitivity estimation. However, they consider only positive weight perturbations, and only in the edges which are assigned.

Most similar to the analysis presented in this paper are the results from Sotskov et al. (1995), which discusses the largest uniform bound on all of the input weight perturbations to maintain optimality for both the linear and bottleneck assignment problems. The results from Sotskov et al. (1995) will coincide with the smallest magnitude perturbation bound in our results, discussed in Section 2 and Section 5.

Previously, Michael et al. (2019) presented a preliminary sensitivity analysis which allowed for simultaneous perturbations in all assignment weights. In this paper we build on these results, providing a complete theoretical framework for understanding the sensitivity of the bottleneck assignment problem, as well as improved algorithmic results. For a given bottleneck assignment, the methods we provide construct intervals within which the
assignment weights may vary while preserving the optimality of the assignment. We further show these are the 
lexicographically largest set of allowable perturbations, as defined in Section 2. We provide algorithms to compute 
these intervals, as well as numerical examples and complexity analyses.

1.2. Organisation of Paper

Section 2 formalises the assignment problem, as well as the definitions of sensitivity used in this work. Section 3 establishes the main theoretical tools used, and applies them to solve the problem of bottleneck edge sensitivity. Using these, the bottleneck assignment sensitivity problem is addressed in Section 4. In Section 5, we discuss the complexities of the provided algorithms. Conclusions are presented in Section 6.

2. Preliminaries and Problem Formulation

Let \( G = (V, \mathcal{E}) \) be a bipartite graph with \( V = V_A \cup V_T \) the set of vertices such that \( V_A \cap V_T = \emptyset \), and the edge set \( \mathcal{E} \subseteq V_A \times V_T \). Define a weight matrix \( W \in \mathbb{R}^{n \times m} \) over the extended reals i.e., \( \mathbb{R} := \mathbb{R} \cup \{-\infty, +\infty\} \), where \( n := |V_A| \) and \( m := |V_T| \) such that \( m \leq n \), and \( w_{ij} \) is the weight associated with edge \((i, j)\), or \(+\infty\) if \((i, j) \notin \mathcal{E}\). We also define a set of binary decision variables \( \pi := \{\pi_{ij} \mid \pi_{ij} \in \{0, 1\}, (i, j) \in \mathcal{E}\} \). The bottleneck assignment problem can be formulated as

\[
\min_{\pi_{ij} \in \{0, 1\}} \max_{(i, j) \in \mathcal{E}} \pi_{ij}w_{ij} \tag{1a}
\]

subject to

\[
\sum_{i \in V_A} \pi_{ij} = 1, \ j \in V_T \tag{1b}
\]

\[
\sum_{j \in V_T} \pi_{ij} \leq 1, \ i \in V_A. \tag{1c}
\]

If \( \pi_{ij} = 1 \) then we say that vertex \( i \) is assigned to vertex \( j \) or the edge \((i, j)\) is assigned. Constraints (1b) and (1c) ensure that every vertex is assigned to at most one other vertex, and that all vertices from the smaller vertex set \( V_T \) are assigned. Along with the set of binary decision variables \( \pi \), we define an assignment \( \Pi = \{(i, j) \mid \pi_{ij} = 1\} \) to be the set of edges which are assigned in \( \pi \). We refer to an assignment which is an optimiser of (1) as a bottleneck assignment.
Definition 1 (Feasible Assignments Set). For a bipartite graph $G$ with weights $W \in \mathbb{R}^{n \times m}$, define $\mathcal{P}^{n \times m}$ as the set of all possible assignments $\Pi$ satisfying the constraints (1b)-(1c).

We make the following standing assumption throughout this paper.

Assumption 1. The set $\mathcal{P}^{n \times m}$ as defined in Definition 1 is nonempty.

We define a mapping $\text{BAP}: \mathbb{R}^{n \times m} \rightarrow \mathcal{P}^{n \times m}$ which maps the weight matrix to the set of optimisers of (1).

Definition 2 (Allowable Perturbation). For a given bipartite graph $G$ with weights $W$, let $\Pi \in \text{BAP}(W)$, i.e. $\Pi$ is an optimiser of (1) over the weights $W$. A perturbation $P \in \mathbb{R}^{n \times m}$ is allowable with respect to $\Pi$ if $\Pi \in \text{BAP}(W + P)$. If $\Pi \not\in \text{BAP}(W + P)$, then $P$ is not allowable with respect to $\Pi$.

For a given graph $G$ and weights $W$ with optimiser $\Pi \in \text{BAP}(W)$, identifying if $\Pi$ is an optimiser over the perturbed weights $W + P$ is, in general, equivalent to solving a new assignment problem. In this sensitivity analysis, we provide a sufficient but not necessary condition for the invariance of the optimal assignment, in the form of an interval test.

Let $\Lambda \subseteq \mathbb{R}^{n \times m}$ be an $n \times m$ array of intervals over the extended reals. For each edge $(i, j) \in E$, let $\Lambda_{ij} := [-\lambda_{ij}, \lambda_{ij}]$ be the interval corresponding to edge $(i, j)$. For any perturbation $P \in \mathbb{R}^{n \times m}$, we write $P \in \Lambda$ if

$$p_{ij} \in [-\lambda_{ij}, \lambda_{ij}], \quad \forall (i, j) \in E.$$  

Finally, we label $\Lambda$ allowable relative to $\Pi$ if for all $P \in \Lambda$, $P$ is an allowable perturbation relative to $\Pi$ as in Definition 2. Throughout this paper we will require that $0^{n \times m} \in \Lambda$, or equivalently $\lambda_{ij}, \bar{\lambda}_{ij}$ are non-negative. To order the intervals we use the lexicographic ordering.

The lexicographic ordering is inspired by alphabetical ordering of words, such as in a dictionary. Let $\Lambda \subseteq \mathbb{R}^{n \times m}$ be an array of intervals. Define a mapping $\rho_k(\Lambda)$ which returns the $k$-th minimal magnitude upper or lower bound of any of the intervals in $\Lambda$, which will be used to order these intervals.

Definition 3 (Lexicographic Ordering of Interval Arrays). For two arrays of intervals, $\Lambda_a, \Lambda_b \subseteq \mathbb{R}^{n \times m}$, $\Lambda_a \succ \Lambda_b$ if $\rho_1(\Lambda_a) > \rho_1(\Lambda_b)$. If $\rho_1(\Lambda_a) = \rho_1(\Lambda_b)$,
then $\Lambda_a, \Lambda_b$ are compared by $\rho_2(\Lambda_a), \rho_2(\Lambda_b)$, and so on. Let the operator $\text{lexmax}(I)$ return the lexicographic maximal element of $I$ denoted by $\Lambda^* \in I$ where $I$ is a set of interval arrays and $\Lambda^* \succeq \Lambda, \forall \Lambda \in I$, with $\succeq$ denoting the lexicographic ordering.

Lexicographic maximisation thus maximises the minimum element, then the second minimum, and so on.

**Problem 1.** For a given graph $G = (V, E)$ with weights $W$ and optimiser $\Pi \in \text{BAP}(W)$, construct the lexicographically largest array of intervals $\Lambda^* := \{[-\Delta_j, \Delta_j, \lambda_{ij}, \lambda_{ij}] \mid \lambda_{ij}, \lambda_{ij} \geq 0, \forall (i, j) \in E\}$ such that any perturbation $P \in \Lambda^*$ is allowable with respect to $\Pi$. For a given graph $G$ with weights $W$ and $\Pi \in \text{BAP}(W)$, let

$$L_0 := \{\Lambda \subseteq \mathbb{R}^{n \times m} \mid 0^{n \times m} \in \Lambda\}, \quad (2)$$
$$L_A := \{\Lambda \subseteq \mathbb{R}^{n \times m} \mid \Pi \in \text{BAP}(W + P), \forall P \in \Lambda\}. \quad (3)$$

Problem 1 is then equivalent to finding $\Lambda^* \subseteq \mathbb{R}^{n \times m}$ such that

$$\Lambda^* = \text{lexmax} (L_0 \cap L_A). \quad (4)$$

Before solving Problem 1, or equivalently constructing $\Lambda^*$ from (4), we will solve a simpler related problem. To this aim, we define bottleneck edges below.

**Definition 4 (Bottleneck Edges).** For a bipartite graph $G$ with weights $W$, the maximal weight edges in all bottleneck assignments are the bottleneck edges of $G$.

We define a new mapping $E : \mathbb{R}^{n \times m} \to E$ which takes the weights and returns the bottleneck edges. We define a perturbation $P \in \mathbb{R}^{n \times m}$ as edge allowable relative to a bottleneck edge $e^* \in E(W)$ if $e^* \in E(W + P)$. Note that we use either $e$ or $(i, j)$ to denote an edge interchangeably. We formulate the problem of the bottleneck edge sensitivity similarly to the bottleneck assignment sensitivity.

**Problem 2.** For a given bipartite graph $G = (V, E)$ with weights $W$ and $e^* \in E(W)$, construct the lexicographically largest array of intervals $\Lambda^* := \{[-\Delta_j, \Delta_j, \lambda_{ij}, \lambda_{ij}] \mid \Delta_j, \lambda_{ij} \geq 0, \forall (i, j) \in \mathcal{E}\}$ such that any perturbation $P \in \Lambda^*$ is edge allowable with respect to $e^*$. 
For a bipartite graph $G$ with weights $W$ and $e^* \in E(W)$, let

$$
L_E := \{ \Lambda \subseteq \mathbb{R}^{n \times m} | e^* \in E(W + P), \quad \forall P \in \Lambda \}. \tag{5}
$$

Problem 2 is then equivalent to finding $\Lambda^* \subset \mathbb{R}^{n \times m}$ such that

$$
\Lambda^* = \text{lexmax} (L_0 \cap L_E). \tag{6}
$$

This problem will be addressed in Section 3. The theoretical tools used to solve Problem 2 will be central to solving Problem 1. However, before proceeding any further, we introduce exclusive sets which will be used throughout the paper.

### 2.1. Exclusive Sets

We begin by defining the central theoretical tool of this paper, which we term the exclusive set with respect to some $e \in \mathcal{E}$, denoted $S_e$.

**Definition 5 (Exclusive Set).** For a bipartite graph $G$ we define $S_{e^*}$ to be an exclusive set with respect to an edge $e^* \in \mathcal{E}$ if it satisfies the following properties:

1. $S_{e^*} \subseteq \mathcal{E}$;
2. $e^* \not\in S_{e^*}$;
3. $\mathcal{P}^{n \times m} = \mathcal{P}_{e^*} \cup S_{e^*}$ where $\mathcal{P}_e \subset \mathcal{P}^{n \times m}$ is the subset of assignments which assign edge $e$.

For a bipartite graph $G$, define $S_e$ to be the set of all exclusive sets for $e \in \mathcal{E}$.

The third property of an exclusive set $S_{e^*}$, as defined above, is critical in connecting the exclusive set to the bottleneck assignment problem. In words, we may rephrase the property as: *All assignments over the graph either contain the edge $e^*$, or an edge from the exclusive set.* The connection between the bottleneck assignment problem and the exclusive set is made clear in Lemma 1.

**Lemma 1.** For a bipartite graph $G$ with weights $W$, let $e^* \in \mathcal{E}$. There exists an assignment $\Pi \in \mathcal{P}_{e^*}$ and an exclusive set $S_{e^*} \in S_{e^*}$ that satisfy

$$
w_{e^*} = \max_{e \in \Pi} w_e \tag{7}
$$

$$
w_{e^*} \leq \min_{e \in S_{e^*}} w_e \tag{8}
$$

(7)
if and only if $e^* \in E(W)$, i.e., the edge $e^*$ is a bottleneck edge, and $\Pi$ is a bottleneck assignment.

Proof. Appendix A.

Lemma 1 establishes the equivalence of $e^*$ being a bottleneck edge and the existence of an exclusive set and assignment $\Pi$ satisfying (7) and (8) relative to $e^*$. This tool allows a reformulation of (4) and (6), in which we replace the sets $L_A$ and $L_E$ defined in (3) and (5), respectively, with sets re-defined by the conditions from Lemma 1.

3. Bottleneck Edge Sensitivity

We now apply Lemma 1 to reformulate and solve Problem 2. For a bipartite graph $G$ with weights $W$, let $e^* \in E(W)$. By Lemma 1, there exist an assignment $\Pi$ and an exclusive set $S_e^*$ satisfying (7) and (8), respectively, relative to $e^*$. We define the following sets of interval arrays:

$$A_\Pi := \{ \Lambda \subset \mathbb{R}^{n \times m} \mid w_{e^*} - \Delta_e \geq w_e + \bar{\lambda}_e, \quad \forall e \in \Pi \},$$ (9)

$$X_{S_e^*} := \{ \Lambda \subset \mathbb{R}^{n \times m} \mid w_{e^*} + \lambda_e \leq w_e - \Delta_e, \quad \forall e \in S_e^* \}. $$ (10)

The main result of this section is presented below.

Theorem 1. For a given bipartite graph $G$ with weights $W \in \mathbb{R}^{n \times m}$, let $e^* \in E(W)$. There exists an assignment $\Pi \in \mathcal{P}_{e^*}$ and an exclusive set $S_{e^*}$ satisfying (7) and (8), respectively, such that

$$\text{lexmax}(L_0 \cap X_{S_{e^*}} \cap A_\Pi) = \text{lexmax}(L_0 \cap L_E)$$ (11)

where $L_0$, $L_E$, $A_\Pi$, $X_{S_{e^*}}$ are defined in (2), (5),(9), and (10), respectively.

Proof. Appendix B.

Remark 1. In Theorem 1, we note that (9)-(10) are defined only by the bounds $\Delta_e$, $\forall e \in S_{e^*} \cup \{e^*\}$ and $\bar{\lambda}_e$, $\forall e \in \Pi$. All other upper and lower bounds are not involved in the set definitions, and are thus unbounded in the lexicographic maximisation. We use $\lambda_e = \infty$ to denote that any positive perturbation is allowable, similarly using $-\Delta_e = -\infty$ for negative perturbations.
In Theorem 1 we reformulate Problem 2, using auxiliary variables $S_e^* \in S_e^*$ and $\Pi \in \mathcal{P}_e^*$. In the following section we provide an algorithm to construct $\Lambda^* = \text{lexmax}(L_0 \cap X_{S_e^*} \cap A_{\Pi})$, for a given exclusive set $S_e^*$ and assignment $\Pi$. Finally, we conclude with algorithms to construct the particular exclusive set $\hat{S}_e^* \in S_e^*$ and assignment $\hat{\Pi} \in \mathcal{P}_e^*$ such that

$$\text{lexmax}(L_0 \cap X_{\hat{S}_e^*} \cap A_{\hat{\Pi}}) \succeq \text{lexmax}(L_0 \cap X_{S_e^*} \cap A_{\Pi}), \quad \forall S_e^* \in S_e^*, \quad \forall \Pi \in \mathcal{P}_e^*,$$

completing the solution to Problem 2.

3.1. Constructing Edge Allowable Perturbation Intervals

In this section we construct $\text{lexmax}(L_0 \cap X_{S_e^*} \cap A_{\Pi})$ for a given exclusive set and assignment.

**Algorithm 1**: Algorithm for $\text{lexmax}(L_0 \cap X_{S_e^*} \cap A_{\Pi})$

**Data:** $G, S_e^*, \Pi, e^*$

**Result:** $\Lambda$

1. $\Lambda \leftarrow \{-\underline{\lambda}_e, \overline{\lambda}_e\} \mid \underline{\lambda}_e \leftarrow \infty, \overline{\lambda}_e \leftarrow \infty, \forall e \in E$;
2. $\underline{\lambda}_e^* \leftarrow \min_{e \in S_e^*} \frac{w_e - w_{e^*}}{2}$;
3. for $e \in S_e^*$ do
4. $\underline{\lambda}_e \leftarrow w_e - w_{e^*} - \underline{\lambda}_e^*$;
5. end
6. $\underline{\lambda}_\Pi \leftarrow \min_{e \in \Pi \setminus e^*} \frac{w_{e^*} - w_e}{2}$;
7. for $e \in \Pi \setminus e^*$ do
8. $\overline{\lambda}_e \leftarrow w_{e^*} - \underline{\lambda}_e^* - w_e$;
9. end
10. return $\Lambda$

**Lemma 2.** For a bipartite graph $G$ with weights $W$, let $\Pi \in \mathcal{P}_e^*$ and $S_e^*$ be an assignment and exclusive set satisfying (7)-(8), respectively relative to $e^* \in E(W)$. With $G, W, \Pi, S_e^*$, and $e^*$ as inputs to Algorithm 1, the output $\Lambda$ satisfies

$$\Lambda = \text{lexmax}(L_0 \cap X_{S_e^*} \cap A_{\Pi})$$

where $A_{\Pi}$ and $X_{S_e^*}$ are given by (9)-(10).
Proof. Appendix C.

Remark 2. For a bipartite graph $G = (V, E)$, Algorithm 1 has complexity $O(|E|)$.

3.2. Constructing the Exclusive Set and Assignment

Algorithm 1 constructs the lexicographically largest set of allowable perturbation intervals given an exclusive set and bottleneck assignment. We now show how to select the exclusive set $\hat{S}_{e^*}$ and bottleneck assignment $\hat{\Pi}$ such that

$$\text{lexmax}(L_0 \cap X_{\hat{S}_{e^*}} \cap A_{\hat{\Pi}}) \succeq \text{lexmax}(L_0 \cap X_{S_{e^*}} \cap A_{\Pi}), \quad \forall S_{e^*} \in S_{e^*}, \quad \forall \Pi \in \mathcal{P}_{e^*}.$$ 

We begin with the bottleneck assignment $\hat{\Pi}$.

Definition 6 (Lexicographic Assignment). Given a weight matrix $W$, an assignment $\Pi$ is a lexicographic assignment if $\phi_1(\Pi', W) \geq \phi_1(\Pi, W)$ for any other assignment $\Pi'$ and if there exists an $i \in \{2, \ldots, |\Pi|\}$ such that

$$\phi_j(\Pi, W) = \phi_j(\Pi', W), \quad \forall j \in \{1, \ldots, i-1\} \implies \phi_i(\Pi', W) \geq \phi_i(\Pi, W),$$

where $\phi_i(\Pi)$ returns the $i$-th largest element of the set $\{w_{ij} | (i, j) \in \Pi\}$.

Informally, the assignment in which the largest weight is minimised, the second largest weight is minimised, ... is the lexicographic assignment. The lexicographic assignment can be seen as a continuation of the bottleneck assignment objective, and can be computed in polynomial time, Burkard and Rendl (1991).

Lemma 3. For a given set of weights $W$ and an exclusive set $S_{e^*}$, the lexicographic assignment $\hat{\Pi}$ with maximal weight edge $e^*$ satisfies

$$\text{lexmax}(L_0 \cap X_{\hat{S}_{e^*}} \cap A_{\hat{\Pi}}) \geq \text{lexmax}(L_0 \cap X_{S_{e^*}} \cap A_{\Pi}), \quad \forall \Pi \in \mathcal{P}_{e^*}.$$ 

Proof. Appendix D.

We now turn to the choice of the exclusive set $\hat{S}_{e^*}$ which yields the lexicographically largest array of intervals from Algorithm 1.

In Algorithm 2, the initial bottleneck edge is found, removed from the graph by setting its weight to $\infty$, and the subsequent bottleneck edge is
Algorithm 2: Algorithm to Construct $\hat{\mathcal{S}}_{e^*}$

**Data:** $W, e^*$

**Result:** $\hat{\mathcal{S}}_{e^*}$

1. $\hat{\mathcal{S}}_{e^*} \leftarrow \{\}$;
2. $w_{e^*} \leftarrow \infty$; // $\infty$ may be any suitably large constant
3. $e \leftarrow \mathbf{E}(W)$;
4. while $w_{e^*} \neq \infty$ do
5.     $\hat{\mathcal{S}}_{e^*} \leftarrow \{\hat{\mathcal{S}}_{e^*}, e\}$;
6.     $w_{e^*} \leftarrow \infty$;
7.     $e \leftarrow \mathbf{E}(W)$;
8. end
9. return $\hat{\mathcal{S}}_{e^*}$

found. If the new bottleneck edge has finite weight, it is added to the set $\hat{\mathcal{S}}_{e^*}$. This repeats until the bottleneck edge has weight $\infty$, in which case there are no assignments left with the remaining edges of the graph. In order to prove the output of Algorithm 2 has the requisite properties, discussed in Lemma 4, we place the following restriction on graph $\mathcal{G}$.

**Assumption 2.** For the bipartite graph $\mathcal{G}$ with weights $W$, the bottleneck edge $e \in \mathbf{E}(W)$ at each iteration of Algorithm 1 is unique.

Assumption 2 is trivially satisfied if all edge weights are distinct.

**Lemma 4.** Given a weight matrix $W$ with a bottleneck edge $e^* \in \mathbf{E}(W)$, the set of edges returned from Algorithm 2 is an exclusive set $\hat{\mathcal{S}}_{e^*}$ relative to $e^*$ as defined in Definition 5. If Assumption 2 holds, then for a given bottleneck assignment $\Pi \in \mathcal{P}_{e^*}$ the exclusive set returned from Algorithm 2 satisfies

$$\text{lexmax}(\mathcal{L}_0 \cap \mathcal{X}_{\hat{\mathcal{S}}_{e^*}} \cap \mathcal{A}_\Pi) \geq \text{lexmax}(\mathcal{L}_0 \cap \mathcal{X}_{\mathcal{S}_{e^*}} \cap \mathcal{A}_\Pi), \quad \forall \mathcal{S}_{e^*} \in \mathcal{S}_{e^*}.$$

**Proof.** Appendix E. \hfill \Box

**Corollary 1.** Given a weight matrix $W$ with a bottleneck edge $e^* \in \mathbf{E}(W)$, let $\hat{\mathcal{S}}_{e^*}$ and $\hat{\Pi}$ be the exclusive set returned from Algorithm 2 and the lexicographic assignment, respectively. With $\mathcal{G}$, $\hat{\mathcal{S}}_{e^*}$, $\hat{\Pi}$, and $e^*$ as inputs, the array of intervals $\Lambda^*$ returned from Algorithm 1 satisfies $\Lambda^* = \text{lexmax}(\mathcal{L}_0 \cap \mathcal{L}_E)$, and thus is the solution of Problem 2.
Proof. By Theorem 1, we have that there exists an exclusive set and assignment such that (11) is true. By Lemma 2, we have that $A^*$ from Algorithm 1 is the lexicographically largest array of intervals satisfying (9)-(10) for a given exclusive set and assignment. By Lemmas 3 and 4, we have that $\hat{S}_e$ from Algorithm 2 and the lexicographic assignment are inputs which lexicographically maximise the $A^*$ output from Algorithm 1, therefore $A^*$ is the solution to Problem 2. \hfill \Box

Example 1. Consider a scenario where the number of agents and tasks is $n = m = 3$. We consider a case where all edges have finite weight, although in general this is not necessary. Consider the following weight matrix

$$W = \begin{bmatrix} 2 & 91 & 63 \\ 26 & 89 & 93 \\ 48 & 60 & 71 \end{bmatrix}.$$ 

The bottleneck edge of this weight matrix is $e^* = (1, 3)$, with weight 63. We define the set $\hat{S}_e$ as it is returned from Algorithm 2, $\hat{S}_e = \{(2, 2), (1, 2), (2, 3)\}$. With the lexicographic assignment $\hat{\Pi} = \{(2, 1), (3, 2), (1, 3)\}$ and the aforementioned $\hat{S}_e^*$, the lexicographically largest allowable perturbation intervals array is given by

$$A^* = \text{lexmax}(L_0 \cap \mathcal{A}_{\hat{S}_e^*} \cap A_{\hat{\Pi}}),$$

where $L_0, \mathcal{A}_{\hat{S}_e^*}, A_{\hat{\Pi}}$ are defined as in Theorem 1. The elements of interval array $A^*$ are presented in Table 1.

| $A^*$ |
|------------------|
| $(-\infty, \infty)$ |
| $[-15, \infty)$ |
| $[-1.5, 13]$ |
| $(-\infty, 35.5]$ |
| $[-13, \infty)$ |
| $[-17, \infty)$ |
| $(-\infty, \infty)$ |
| $(-\infty, 1.5]$ |
| $(-\infty, \infty)$ |

It is simple to verify that any perturbation $P \in A^*$ satisfies $e^* \in E(W + P)$.

4. Bottleneck Assignment Sensitivity

In this section we solve Problem 1 by providing Algorithm 3 to construct $A^* = \text{lexmax}(L_0 \cap A_{\hat{\Pi}})$. In the previous section, we separated the construction of $\hat{S}_e^*$ and $\hat{\Pi}$ from the construction of $A^*$. However, for Problem 1, we
construct $\Lambda^*$ directly, relying on the previously established theory of exclusive sets only in the proof of the results. We first make an assumption on the structure of the weighted graph, analogous to Assumption 2.

**Assumption 3.** The bottleneck edges for all constructed graphs in Algorithm 3 are unique.

This assumption may be relaxed, however, it will lead to a quite cumbersome set of edge cases being included in the logic of Algorithm 3 and the proof of the following theorem without providing much insight into the nature of the problem.

**Theorem 2.** For a bipartite graph $G$ with weights $W$, let $\Pi \in \text{BAP}(W)$. If Assumption 3 holds, then the array of intervals $\Lambda^*$ returned from Algorithm 3 satisfies

$$\Lambda^* = \text{lexmax} (L_0 \cap L_A),$$

for $L_A$ defined in (3). Equivalently, $\Lambda^*$ is the solution to Problem 1.

**Proof.** For a given bipartite graph $G$ with weights $W$ and $\Pi \in \text{BAP}(W)$, let $\Lambda^*$ be the array of intervals returned from Algorithm 3. We first prove by contradiction that there exists no lexicographically larger $\Lambda' \in L_0 \cap L_A$, by showing that a lexicographically larger set of intervals must contain an unallowable perturbation. We then show that $\Lambda^* \in L_0 \cap L_A$, which completes the proof. With the addition of Lemma 5 in Appendix F, the proof of this Theorem follows a similar line of reasoning as that of Lemma 4. For the complete proof, see Appendix G. \qed

In Algorithm 3, we use $B[e_j]$ to indicate the element of $B \in \mathbb{R}^{n \times m}$ corresponding to edge $e_j$. We also use $\infty$ to denote a sufficiently large number such that when applied to a weight element the corresponding edge can be considered “removed” from the graph.

Prior to presenting Algorithm 3, we first discuss the key steps and motivations. For the given assignment $\Pi$, we focus on a single edge $e \in \Pi$. Clearly, if there exists a perturbation $P \in \Lambda$ and an assignment $\Pi' \neq \Pi$ such that $w_{e'} + p_{e'} < w_e + p_e$, $\forall e' \in \Pi'$, then $P$ cannot be an allowable perturbation and by extension $\Lambda$ is not allowable. Algorithm 3 thus begins with all bounds of $\Lambda$ unset (labeled with $\infty$), and iterates the following steps:
1. For each $e \in \Pi$ and for each $e' \in \mathcal{E}$, compute the lexicographically largest $\overline{\lambda}_e, \overline{\lambda}_e'$ such that $w_e + \overline{\lambda}_e = w_{e'} - \overline{\lambda}_{e'}$, for whichever of the two $\overline{\lambda}_e, \overline{\lambda}_e'$ are unset. Store the resulting value in $B_e[e']$. (Update function in line 5)

2. For each $e \in \Pi$, find the bottleneck edge $b_e = \text{E}(B_e)$. The edge $b_e$ is a maximal weight edge of an assignment $\Pi'$ for which there exists $p \leq B_e[e']$ such that $w_e + p \geq w_{e'} - p$, $\forall e' \in \Pi'$. Let $\hat{e}$ be the edge for which the weight of $\text{E}(B_{\hat{e}})$ is less than or equal to the weight of $\text{E}(B_{b_e})$, $\forall e \in \Pi$, and let $\hat{b} = \text{E}(B_{\hat{e}})$. (Lines 6 & 8)

3. Update the bounds such that $w_{\hat{e}} + \overline{\lambda}_{\hat{e}} = w_{b_{\hat{e}}} - \overline{\lambda}_{b_{\hat{e}}}$. (Lines 9-17)

We conclude by implementing Algorithm 3 on the same example as in Section 3, to highlight the differences between the measures of sensitivity.

Example 2. Using the same weight matrix as in Example 1, we perform a sensitivity analysis for assignment $\Pi = \{(2, 1), (3, 2), (1, 3)\}$. The array of intervals $\Lambda^*$ returned from Algorithm 3 is shown in Table 1.

We can see in this case that either the allowable perturbation intervals are identical to the ones in Example 1, or strictly larger. However, this will not be true in general. Although the intervals $\Lambda^*$ corresponding to the bottleneck assignment sensitivity will be lexicographically larger, they may include more bounded edges than the intervals for the bottleneck edge sensitivity. For comparison, we will show the results from Sotskov et al. (1995), over the same example. Further explanation of these results are included in Section 5.

5. Computational Complexity

In this section we will provide a brief accounting of the complexity of the algorithms provided, as well as a comparison to the results from Sotskov et al. (1995). We first note that, in the computation of the bottleneck edge sensitivity as well as the bottleneck assignment sensitivity, we are iteratively running a bottleneck assignment solver on a graph, removing or modifying one edge weight at a time. Unsurprisingly, we can take advantage of the information from the previous iteration, to dramatically reduce the computational complexity. Using an augmenting path solver, such as from Punnen and Nair

\footnote{If both of $\overline{\lambda}_e, \overline{\lambda}_e'$ are unset, the result is the mean of $w_e$ and $w_{e'}$ for each of $\overline{\lambda}_e, \overline{\lambda}_e'$, and thus can be represented by the single number $B_e[e']$}
Algorithm 3: Algorithm to Construct \( \Lambda^* = \text{lexmax} (\mathcal{L}_0 \cap \mathcal{L}_A) \)

Data: \( G, W, \Pi \)

Result: \( \Lambda \)

1. \( \Lambda \leftarrow \{ [\lambda_e, \overline{\lambda}_e] \mid \lambda_e \leftarrow \infty, \overline{\lambda}_e \leftarrow \infty, \forall e \in E \} ; \)
2. \( B_e \leftarrow 0^{n \times m}, \forall e \in \Pi ; \) // Matrices of 0 the same size as \( W \)
3. while True do
   4.   for \( e \in \Pi \) do
      5.     \( B_e \leftarrow \text{update}(B_e, \Lambda^*, W) ; \)
      6.     \( b_e \leftarrow \mathbf{E}(B_e) ; \)
   7.   end
   8.   \( \hat{e} \leftarrow \text{argmin}_{e \in \Pi} B_e[b_e] ; \) // Edge \( \hat{e} \in \Pi \) has tightest bound \( B_e[b_e] \)
   9.   if \( B_e[b_e] = \infty \) then
      10.      return \( \Lambda \) ;
   11.   else if \( \overline{\lambda}_{\hat{e}} = \infty \land \Delta_{\hat{e}} = \infty \) then
      12.      \( \overline{\lambda}_{\hat{e}}, \Delta_{\hat{e}} \leftarrow B_e[b_e] ; \) // Neither of \( \overline{\lambda}_{\hat{e}}, \Delta_{\hat{e}} \) have been previously set
   13.   else if \( \overline{\lambda}_{\hat{e}} = \infty \) then
      14.      \( \overline{\lambda}_{\hat{e}} \leftarrow B_e[b_e] ; \) // Bound \( \overline{\lambda}_{\hat{e}} \) has not been previously set
   15.   else if \( \Delta_{\hat{e}} = \infty \) then
      16.      \( \Delta_{\hat{e}} \leftarrow B_e[b_e] ; \) // Bound \( \Delta_{\hat{e}} \) has not been previously set
   17.   end
   18. end
Function update

Data: \( B_e, \Lambda^*, W \)

\[
\begin{align*}
B_e & \leftarrow \{ B_e[e'] \leftarrow \infty | \forall e' \in \mathcal{E} \}; \\
\text{for } e' \in \mathcal{E} \text{ do} & \\
\text{if } \overline{\lambda}_e = \infty \land \lambda_{e'} = \infty & \text{ then} \\
B_e[e'] & \leftarrow \frac{w_{e'} - w_e}{2}; \\
\text{else if } \overline{\lambda}_e = \infty & \text{ then} \\
B_e[e'] & \leftarrow w_{e'} - \lambda_{e'} - w_e; \\
\text{else if } \lambda_{e'} = \infty & \text{ then} \\
B_e[e'] & \leftarrow w_{e'} - w_e - \overline{\lambda}_e; \\
\text{else if } w_e + \overline{\lambda}_e > w_{e'} - \lambda_{e'} & \text{ then} \\
B_e[e'] & \leftarrow -\infty; \\
\end{align*}
\]

end

Table 2: Lexicographically Largest Allowable Perturbation Intervals
\[
\begin{array}{ccc}
(-\infty, \infty) & [-15, \infty) & (-\infty, 13] \\
(-\infty, 50] & [-13, \infty) & [-17, \infty) \\
(-\infty, \infty) & (-\infty, 16] & (-\infty, \infty)
\end{array}
\]

We begin with the computational complexity of the bottleneck edge sensitivity. To compute the bottleneck edge sensitivity of a given \( e^* \in \text{BAP}(W) \), we compute

- The lexicographic assignment \( \hat{\Pi} \).
- The exclusive set as returned from Algorithm 2.
- The lexicographically largest \( \Lambda^* \) defined by the previous two steps, as in Algorithm 1.
Table 3: Sensitivity Results using Sotskov et al. (1995)

|       |       |       |
|-------|-------|-------|
| [-13,13] | [-13,13] | [-13,13] |
| [-13,13] |       |       |
| [-13,13] |       |       |

For a bipartite graph $G = (V_1 \cup V_2, \mathcal{E})$, let $n = |V_1| + |V_2|$ and $m = |\mathcal{E}|$. Algorithm 2 requires $O(m)$ iterations of a bottleneck assignment solver, although as noted previously, each iteration is only a single augmenting path search with complexity $O(n)$. Note this is not true without Assumption 2. The complexity of computing the intervals $\Lambda^*$ is then $O(L(n,m) + nm + m)$, where $L(n,m)$ is the computational complexity of constructing the lexicographic assignment. For a simple example, we assume the graph is square and dense, i.e. $|V_1| = |V_2|$ and $\mathcal{E} = V_1 \times V_2$, and that the bottleneck edge of each subgraph of $G$ is unique. Then the lexicographic assignment may be computed as $O(n)$ iterations of the augmenting path solver, which yields a complexity of $O(n^2 + n^3 + n^2) = O(n^3)$. This simplification shows that the computation of the exclusive set in Algorithm 2, which requires $O(n^2)$ iterations of the bottleneck assignment solver, dominates the complexity.

For the complexity of the bottleneck assignment sensitivity, we note that there are $O(m)$ iterations of Algorithm 3, and at each iteration we update $O(n)$ of the constructed graphs $B_e, \forall e \in \Pi$. The complexity is then $O(nB(n,m) + n^2m)$, for $B(n,m)$ the complexity of the bottleneck assignment solver over the given graph. Using the complexity of the bottleneck assignment algorithm from Punnen and Nair (1994), and assuming a square and dense graph as before, this yields a complexity of $O(n^{3.5} + n^4) = O(n^4)$.

In the introduction, we mentioned the most similar work found in the literature to the presented was Sotskov et al. (1995). In their work, they construct a “sensitivity radius” $\sigma(G)$, which we may interpret in the notation of this paper as the maximum scalar $\sigma \in \mathbb{R}^+$ such that all perturbations $P_{\sigma} := \{p_{ij} \in [-\sigma, \sigma] \mid (i,j) \in \mathcal{E}\}$ are allowable. Letting $\Lambda_{\sigma}$ be the array of intervals with lower and upper bound $-\sigma, \sigma$, we then have that $\rho_i(\Lambda^*) = \rho_i(\Lambda_{\sigma})$ and $\rho_i(\Lambda^*) \geq \rho_i(\Lambda_{\sigma}) , \forall i \in \{2, ..., m\}$ for $\Lambda^*$ the array of intervals computed in Algorithm 3. The computation of the “sensitivity radius” requires solving a single bottleneck assignment, and so has complexity $O(B(n,m))$ or $O(n^{2.5})$ with the same simplifications as previously discussed.
6. Conclusion

In this paper we propose two frameworks for assessing the sensitivity of a bottleneck assignment problem, as well as algorithms for the computation of the sensitivity within each framework. The analysis provided is driven primarily by the characterisation of “exclusive sets”, and the connection of these sets to the bottleneck assignment problem. The combination of the theory of exclusive sets, along with recursive applications of the bottleneck assignment problem, allow for these algorithms to be run with off the shelf assignment solvers and a minimum of additional programming. The sensitivity analysis provided can be used to certify the optimality of a solution, if for example the true assignment weights can be shown to be contained by the provided intervals, or as a measure of the robustness of an assignment solution. Further research should concentrate on the expansion of this type of sensitivity analysis to other formulations of the assignment problem, as well as other formulations of the “largest” set of intervals besides the lexicographic ordering.

Acknowledgements

This research was supported by the University of Melbourne Graduate Research Scholarship.

Appendix A. Proof of Lemma 1

First, assume that edge $e^* \in E(W)$, i.e., it is a bottleneck edge of the graph $G$ with associated weight matrix $W$. Then there exists at least one assignment $\Pi$ where $e^*$ is the maximal weight edge, so $\Pi$ satisfies (7). Because $e^*$ is a bottleneck edge of the graph, we have that there is no assignment in which all edges have weight less than $w_{e^*}$. Therefore, all assignments which do not assign $e^*$ contain at least one edge with weight greater than or equal to $w_{e^*}$. Let $S_{e^*}$ be the set of all edges with weight greater than or equal to $w_{e^*}$, excluding $e^*$. Thus, from Definition 5, $S_{e^*}$ is an exclusive set corresponding to $e^*$ and satisfies (8).

To prove the converse, let the assignment $\Pi$ and the edge set $S_{e^*}$ satisfy (7)–(8), respectively. To obtain a contradiction, we assume there exists an assignment $\Pi'$ that does not assign $e^*$ and only assigns edges with strictly lower weights. We have from the definition of an exclusive set that
\[ P^{n \times m} \setminus P_{e^*} \subseteq \bigcup_{e \in S_{e^*}} P_e, \] i.e., all assignments which do not include \( e^* \), assign at least one edge from \( S_{e^*} \). Therefore, there is at least one edge within \( S_{e^*} \), which is assigned in \( \Pi' \). However, as \( S_{e^*} \) satisfies (8), we conclude that \( \Pi' \) must contain an edge with weight greater than the weight of \( e^* \), which is a contradiction. \( \square \)

Appendix B. Proof of Theorem 1

We will prove this by showing that given \( \Lambda^* = \text{lexmax}(L_0 \cap L_E) \), we are able to construct an assignment \( \Pi \) and an exclusive set \( S_{e^*} \) that satisfy (7) and (8) over \( \widetilde{W} = W + P, \forall P \in \Lambda^* \). Therefore, we will have that \( \Lambda^* \in A_{\Pi} \) and \( \Lambda^* \in X_{S_{e^*}} \). We will then show there exists no lexicographically larger \( \Lambda \) in \( L_0 \cap X_{S_{e^*}} \cap A_{\Pi} \), completing the proof.

We begin by constructing two particular perturbations \( P_1, P_2 \in \mathbb{R}^{n \times m} \), with \( P_1[e] \) the perturbation corresponding to the edge \( e \),

\[
P_1[e] := \begin{cases} 
-\Delta e & \text{if } e \neq e^* \\
\Delta e & \text{if } e = e^*
\end{cases}
\]

\[
P_2[e] := \begin{cases} 
\Delta e & \text{if } e \neq e^* \\
-\Delta e & \text{if } e = e^*
\end{cases}
\]

Every edge in \( P_1 \) is perturbed as negatively as possible, except \( e^* \), and the opposite in \( P_2 \). By (5), we have that \( e^* \in E(W + P), \forall P \in \Lambda^* \), including perturbations \( P_1 \) and \( P_2 \). By Lemma 1, we have that there exists an assignment \( \Pi \) and an exclusive set satisfying (7)-(8) for each of the perturbations \( P_1 \) and \( P_2 \). We let \( S_{e^*} \) be defined as the exclusive set satisfying (8) for \( W_1 := W + P_1 \), and \( \Pi \) be defined as the assignment satisfying (7) for \( W_2 := W + P_2 \). By the construction of perturbations \( P_1 \) and \( P_2 \) we have that (7)-(8) are satisfied by \( \Pi \) and \( S_{e^*} \) for all \( P \in \Lambda^* \), and thus \( \Lambda^* \in A_{\Pi} \) and \( \Lambda^* \in X_{S_{e^*}} \). Therefore,

\[
\Lambda^* = \text{lexmax}(L_0 \cap L_E) \preceq \text{lexmax}(L_0 \cap X_{S_{e^*}} \cap A_{\Pi}).
\]

To complete the proof, note that for any \( \Lambda \in L_0 \cap X_{S_{e^*}} \cap A_{\Pi} \), we have \( e^* \in E(W + P), \forall P \in \Lambda \) by Lemma 1. Therefore,

\[
\text{lexmax}(L_0 \cap X_{S_{e^*}} \cap A_{\Pi}) \preceq \text{lexmax}(L_0 \cap L_E),
\]

which completes the proof. \( \square \)
Appendix C. Proof of Lemma 2

We show that the lexicographically maximal set of bounds Λ is determined from the definitions of (9) and (10), and is returned by Algorithm 1. We first note that in the algorithm there are two nearly identical steps, in lines 2-5 when the bounds for the edges $e \in \mathcal{S}_e^*$ are determined, and lines 6-9 when the bounds for the edges $e \in \Pi$ are determined. Additionally, in lines 2-5 the upper bound for $e^*$ is determined while in lines 6-9 the lower bound. Importantly, we note that the two sections are entirely independent, i.e. they do not affect each other. We show that there exist no lexicographically larger bounds by focusing on lines 2-5, noting that the same argument holds for lines 6-9.

Lines 2-5 of Algorithm 1 yield the minimal magnitude bound
\[
\bar{\lambda}_{e^*} := \min_{e \in \mathcal{S}_e^*} \frac{w_e - w_{e^*}}{2}.
\] (C.1)

There exists an edge $e \in \arg \min w_e, \forall e \in \mathcal{S}_e^*$ with identical lower bound, i.e.
\[
\lambda_e := \min_{e \in \mathcal{S}_e^*} \frac{w_e - w_{e^*}}{2}.
\]

The difference in weight between these two edges is $w_e - w_{e^*}$, and we have from (10) that $w_{e^*} + \bar{\lambda}_{e^*} \leq w_e - \lambda_e$, so $\bar{\lambda}_{e^*} = \lambda_e = \frac{w_e - w_{e^*}}{2}$ is the lexicographically maximal distribution of their allowable perturbation. For all subsequent edges $e \in \mathcal{S}_e^*$, we have from the definition of $\chi_{\mathcal{S}_e^*}$ that
\[
w_{e^*} + \bar{\lambda}_{e^*} \leq w_e - \lambda_e
\]
where $\bar{\lambda}_{e^*}$ is determined by (C.1). Therefore, the maximum $\lambda_e$ is clearly
\[
\lambda_e = w_e - (w_{e^*} + \bar{\lambda}_{e^*}),
\]
which are the bounds determined in Algorithm 1. \qed

Appendix D. Proof of Lemma 3

From Algorithm 1, we once again note that the bounds of the interval array Λ are generated in two independent sets, corresponding to the edges in
the provided bottleneck assignment and exclusive set. As the exclusive set is held fixed and so are the corresponding boundaries, we therefore focus on the boundaries corresponding to the edges in the bottleneck assignment \( \bar{\Pi} \) in lines 6-9.

Algorithm 1 returns the the minimal magnitude bounds

\[
\lambda_{e^*} = \bar{\lambda}_e = \frac{w_{e^*} - w_e}{2}
\]

for \( e := \arg\max_{\Pi \in \bar{\Pi}} \{ w_e \} \). This bound is directly proportional to the difference \( w_{e^*} - w_e \), and as all bottleneck assignments by definition have maximal edge weight \( w_{e^*} \), the bound is maximised by the assignment which minimises \( w_e \). Over all assignments, the lexicographic assignment minimises the second largest weight. Similarly, for all subsequent weights, we have that the lexicographic assignment minimises the \( i \)-th largest weight, and thus maximises the \( i \)-th largest bound \( \bar{\lambda}_e := w_{e^*} - \lambda_{e^*} - w_e \), and therefore

\[
\text{lexmax}(\mathcal{L}_0 \cap \mathcal{X}_{S_{e^*}} \cap \mathcal{A}_{\bar{\Pi}}) \succeq \text{lexmax}(\mathcal{L}_0 \cap \mathcal{X}_{S_{e^*}} \cap \mathcal{A}_\Pi), \quad \forall \Pi \in \mathcal{P}_{e^*}.
\]

\[\square\]

**Appendix E. Proof of Lemma 4**

An exclusive set relative to an edge \( e^* \) is defined by three properties, laid out in Definition 5. The set of edges returned from Algorithm 2 satisfies the first two properties from the definition trivially. To see that it satisfies the final property, note that each edge in the set \( S_{e^*} \) constructed in Algorithm 2 is as well as the bottleneck edge \( e^* \) are assigned the weight \( \infty \). The algorithm terminates when the bottleneck edge of the graph has weight \( \infty \). Therefore, there are no assignments remaining in the graph which do not assign either the bottleneck edge \( e^* \) or an edge in \( S_{e^*} \), which is precisely the final property defining an exclusive set with respect to \( e^* \).

Next, we prove with \( \bar{S}_{e^*} \) as an input, the output array of intervals \( \Lambda^* \) of Algorithm 2 is the lexicographically largest set of allowable perturbations over all exclusive sets \( S_{e^*} \in S_{e^*} \). Let \( \Lambda' \) be the lexicographically largest array of intervals which satisfies

\[
\rho_i(\Lambda') = \rho_i(\Lambda^*), \quad \forall i \in \{1, \ldots, k\}.
\]
for some $k \in \{1, ..., nm - 1\}$. Note that, by Assumption 2, the bottleneck edge at every iteration of the while loop is unique, so for $\{e_1, e_2, ..., e_k\}$ the bottleneck edges from the first $k$ iterations of the while loop in Algorithm 2, we have $\overline{\lambda}_{e_i} = \overline{\lambda}_{e_i}^*$. Assume that $\rho_{k+1}(\Lambda') > \rho_{k+1}(\Lambda)$. There then exists an allowable perturbation such that $w_e + p_e < w_{e^*} + p_{e^*}$ for some $e \in S_{e^*}$. Because $e$ was returned in the $k+1$ iteration of the while loop in Algorithm 2, we know it is the maximal weight edge in some assignment $\Pi$, and therefore there exists a perturbation $P$ under which all edges of $\Pi$ would have weight less than $e^*$, and thus $P$ would not be allowable. □

Appendix F. An Auxiliary Lemma

**Lemma 5.** The sequence of bounds determined within Algorithm 3 is non-decreasing.

**Proof.** The bound determined on the $i$-th iteration of Algorithm 3 is the minimum weight bottleneck edge over $E(B_e)$, $\forall e \in \Pi$. We therefore show that the sequence of bottleneck edges from each of these sets $B_e$ is non-decreasing, completing the proof of the lemma. Focusing on a single set $B_e$ for some $e \in \Pi$, we can separate the iterations into two cases: either the minimum weight bottleneck is from $B_e$, or it is from some other set $B_{e'}$ corresponding to an edge $e' \in \Pi$, with $e' \neq e$.

In the first case, the minimum weight bottleneck edge is $b = E(B_e)$ with weight $w_b$. Assume, previous to the $i$-th iteration, we have $\overline{\lambda}_e = \infty$. Then for any edge $e'$ satisfying $B_e[e'] \geq w_{b}$, we have that $B_e[e']$ is increased or remains constant after updating. The subsequent bottleneck edge of $B_e$ is drawn from this set, so the subsequent bottleneck edge weight is greater than or equal to $w_b$. If instead we have $\overline{\lambda}_e \neq \infty$, then only $w_b$ is set to $\infty$, so the subsequent bottleneck edge weight from $B_e$ is greater than or equal to $w_b$.

In the second case, the minimum weight bottleneck edge is $b' = E(B'_e)$ with weight $w_{b'}$. Therefore, we have that the weight $w_{b'} \leq w_b$ for $w_b$ the weight of $b = E(B_e)$, as the bound is determined by the minimum of these bottleneck edge weights. By careful (tedious) examination of the update function, one may observe that if the update of the weight $B_e[b']$ results in a strict decrease then $w_{b'} > B_e[b']$. We therefore have that $B_e[b'] < w_{b'} \leq w_b$, i.e. that the updated weight was less than the bottleneck edge weight of $B_e$, and remained so after being updated, and therefore does not affect the bottleneck edge weight. □
Appendix G. Proof of Theorem 2

First, to obtain a contradiction, we assume there exists an array of intervals $\Lambda' \in L_0 \cap L_A$ which agrees with $\Lambda^*$ on the first $k$ minimal magnitude bounds, i.e.,

$$\rho_i(\Lambda') = \rho_i(\Lambda^*), \quad \forall \ i \in \{1, \ldots, k\},$$

but has a strictly greater $k+1$ minimal magnitude bound value, i.e. $\rho_{k+1}(\Lambda') < \rho_{k+1}(\Lambda^*)$. We show that $\Lambda'$ contains a perturbation $P$ such that $\Pi \not\in \text{BAP}(W + P)$, which is a contradiction by the definition of $L_A$. By Assumption 3, the bottleneck edges at each iteration of the Algorithm have been unique, so for $B_k = \{\lambda_1^*, \lambda_2^*, \ldots\}$ the bounds which have been set by the $k$-th iteration of Algorithm 3 we have $\lambda_i^* = \lambda_i^*, \ \forall \ \lambda_i^* \in B_k$. The bound $\rho_{k+1}(\Lambda^*)$ is determined by a bottleneck assignment $\Pi' = \text{BAP}(B_e)$ for some $e \in \Pi$, with $e \not\in \Pi'$. For each edge $e' \in \Pi'$, we have that whichever of $\underline{\lambda}_e, \overline{\lambda}_e$ have been undetermined by the $k+1$-th iteration of Algorithm 3 will be greater than or equal to $\rho_{k+1}(\Lambda^*)$, by Lemma 5. However, from the definition of each $B_e[e']$, there must then exist a perturbation $P \in \Lambda'$ such that $\Pi$ contains all edges with weight strictly less than $\Pi$. We therefore have that $\Pi \not\in \text{BAP}(W + P)$ for some $P \in \Lambda'$, and thus $\Lambda' \not\in L_A$, which is a contradiction.

We have shown there exists no lexicographically larger array of intervals $\Lambda' \in L_A$, however we have not yet shown that $\Lambda^* \in L_A$. In order to show this, we utilise the previously developed theory of exclusive sets. Algorithm 3 terminates when $\min_{e \in \Pi} B_e[b_e] = \infty$. An edge weight $e'$ of $B_e$ is defined to be $\infty$ only if $e' = e$, or $e' - \underline{\lambda}_e \geq w_e + \overline{\lambda}_e$. Therefore, if $\min_{e \in \Pi} B_e[b_e] = \infty$, we precisely have that there exists an exclusive set of edges $S_e$ such that $\Lambda^* \in L_e$, for $L_e$ defined in (10). However, with $\Lambda^* \in L_e, \ \forall \ e \in \Pi$, we have that for every perturbation $P \in \Lambda^*$ there exists no assignment with all edges strictly less than any $e \in \Pi$, and therefore $\Pi \in \text{BAP}(W + P), \ \forall \ P \in \Lambda^*$ and $\Lambda^* \in L_A$. \hfill \Box

References

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job shop scheduling. Management science 34, 391–401.

Arslan, G., Marden, J.R., Shamma, J.S., 2007. Autonomous vehicle-target
assignment: A game-theoretical formulation. Journal of Dynamic Systems, Measurement, and Control 129, 584–596.

Burkard, R.E., Rendl, F., 1991. Lexicographic bottleneck problems. Operations Research Letters 10, 303–308.

Carraresi, P., Gallo, G., 1984. A multi-level bottleneck assignment approach to the bus drivers’ rostering problem. European Journal of Operational Research 16, 163–173.

Fu, Y., Sun, J., Lai, K., Leung, J.W., 2015. A robust optimization solution to bottleneck generalized assignment problem under uncertainty. Annals of Operations Research 233, 123–133.

Harchol-Balter, M., Crovella, M.E., Murta, C.D., 1999. On choosing a task assignment policy for a distributed server system. Journal of Parallel and Distributed Computing 59, 204–228.

Jansen, B., De Jong, J., Roos, C., Terlaky, T., 1997. Sensitivity analysis in linear programming: just be careful! European Journal of Operational Research 101, 15–28.

Kasperski, A., Zieliński, P., 2013. Bottleneck combinatorial optimization problems with uncertain costs and the owa criterion. Operations Research Letters 41, 639–643.

Koltai, T., Terlaky, T., 2000. The difference between the managerial and mathematical interpretation of sensitivity analysis results in linear programming. International Journal of Production Economics 65, 257–274.

Lin, C.J., Wen, U.P., 2003. Sensitivity analysis of the optimal assignment. European Journal of Operational Research 149, 35–46.

Lin, C.J., Wen, U.P., 2007. Sensitivity analysis of objective function coefficients of the assignment problem. Asia-Pacific Journal of Operational Research 24, 203–221.

Michael, E., Wood, T.A., Manzie, C., Shames, I., 2019. Uncertainty intervals for robust bottleneck assignment, in: 2019 18th European Control Conference (ECC), IEEE. pp. 4204–4209.
Nam, C., Shell, D.A., 2015. When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE. pp. 1249–1254.

Pentico, D.W., 2007. Assignment problems: A golden anniversary survey. European Journal of Operational Research 176, 774–793.

Pferschy, U., 1997. Solution methods and computational investigations for the linear bottleneck assignment problem. Computing 59, 237–258.

Punnen, A.P., Nair, K., 1994. Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem. Discrete Applied Mathematics 55, 91–93.

Ramaswamy, R., Orlin, J.B., Chakravarti, N., 2005. Sensitivity analysis for shortest path problems and maximum capacity path problems in undirected graphs. Mathematical Programming 102, 355–369.

Shames, I., Dostovalova, A., Kim, J., Hnam, H., 2017. Task allocation and motion control for threat-seduction decoys, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pp. 4509–4514. doi:10.1109/CDC.2017.8264324.

Sotskov, Y.N., Leontev, V., Gordeev, E.N., 1995. Some concepts of stability analysis in combinatorial optimization. Discrete Applied Mathematics 58, 169–190.

Volgenant, A., 2006. An addendum on sensitivity analysis of the optimal assignment. European Journal of Operational Research 169, 338–339.

Volgenant, A., Duin, C.W., 2010. Improved polynomial algorithms for robust bottleneck problems with interval data. Computers & Operations Research 37, 909–915.

Wood, T.A., Khoo, M., Michael, E., Manzie, C., Shames, I., 2020. Collision avoidance based on robust lexicographic task assignment. IEEE Robotics and Automation Letters 5, 5693–5700.