

# Uncertainty Intervals for Robust Bottleneck Assignment

Elad Michael\*, Tony A. Wood, Chris Manzie, and Iman Shames

**Abstract**—We examine the robustness of bottleneck assignment problems to perturbations in the assignment weights. We derive two algorithms that provide uncertainty bounds for robust assignment. We prove that the bottleneck assignment is guaranteed to be invariant to perturbations which lie within the provided bounds. We apply the method to an example of task assignment for a multi-agent system.

## I. INTRODUCTION

This paper focuses on quantifying the set of weight perturbations which maintain the optimality of the original assignment. Task assignment is a central part of autonomous multi-agent systems, assigning agents to tasks in such a way as to optimize some measure of the quality of the assignment. We model the problem with a graph, with the agents and tasks as vertices and the assignments as weighted edges. The optimization cost can be any of minimum weight, minimum total weight, balanced weight, or one of many other types of assignment problems. In many scenarios, especially in physical applications, the weight of assigning an agent to a task has some associated uncertainty. This could be due to a physical measurement error such as a radar covariance, a dynamic environment within which the decisions are being made, or simply decimal roundoff errors. The optimal assignment is robust to some set of perturbations, and this paper provides a parameterization of that set of allowable perturbations for the bottleneck assignment problem.

This problem is studied under various names, such as “sensitivity” [1], “stability” [2], or “post-optimality” [3]. We use *robustness* to refer to the range of variations in the problem parameters under which we can certify the optimal solution. An overview of modern linear sensitivity analysis and its application is covered in [4]. In [4], they differentiate between several types of sensitivity analysis, focused on different aspects of the optimal solution such as optimal cost, active constraints, and the slope of the optimal cost function. The assignment problem, due to its additional constraint structure, is best approached using what is defined as “Type 2 Sensitivity Analysis”. However, [4] focuses on perturbations of a single variable. We will use the term “robustness” to describe the effects of uncertainty in weight on all of the assignments.

In this paper we focus on the robustness of the bottleneck assignment problem (BAP). In a matching of agents to tasks, each with its associated weight, the BAP seeks to minimize the maximum weight within a matching. In [5] the BAP is used to assign multiple decoys to multiple incoming threats. The bottleneck property minimizes the worst case time it takes each decoy to seduce and divert the threat away from the asset. This ensures that all threats are dealt with in the minimum time. The BAP is used in other minimum time applications, such as [6] where they give the example of transporting perishable goods or military supplies reaching command posts during an emergency.

An algorithm with robustness guarantees is provided in [7] along with several complexity improvements on others’ work. However, this approach is used to find robust solutions, not to quantify the robustness of the optimal solution to the original problem. Robustness for the linear assignment problem to correlated perturbations in adjacent to a single node is examined in [1], reflecting an error in the state of an agent or a task. In [8], robustness/stability/sensitivity of the solution for the case where all perturbations are bounded by the same bound is established. This differs from the results of this paper where perturbation bounds are not uniform for all weights. Additionally, contrasting with [8], the algorithms for computing these bounds are presented.

Our contribution in this paper is finding individual perturbation bounds for each weight within which we certify the optimality of the original assignment solution. We allow all weights to vary independently and simultaneously, rather than restricting variations to a subset of the edges. In the presence of uncertainty, these bounds may provide sufficient robustness to certify that the assignment made was optimal. However, it may be that the assignments made are not robust enough for the user’s needs. In this case, the algorithms proposed here can only alert the user to that fact, they do not propose a more robust assignment solution.

The paper is organized as follows. Section 2 is devoted to background on the assignment problem and a formal definition of robustness. Section 3 will provide the lower bounds for the bottleneck assignment problem, and the algorithms to find them. Section 4 covers numerical examples and implementation. Section 5 concludes and remarks on further analysis.

\*All authors are with Department of Electrical and Electronic Engineering at the University of Melbourne  
eladm@student.unimelb.edu.au,  
{wood.t,manziec,iman.shames}@unimelb.edu.au

## II. PROBLEM FORMULATION

Define a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$  over which we will solve the assignment problem. Let the set of vertices be  $\mathcal{V} = \mathcal{V}_A \cup \mathcal{V}_T$  such that  $\mathcal{V}_A \cap \mathcal{V}_T = \emptyset$  and  $|\mathcal{V}_A| \leq |\mathcal{V}_T|$ , the edge set  $\mathcal{E} \subseteq \mathcal{V}_A \times \mathcal{V}_T$ , and edge weights  $\mathcal{W} = \{w_{ij} \in \mathbb{R} \mid (i, j) \in \mathcal{E}\}$ . We also define a set of binary decision variables  $\{\pi_{ij} \mid \pi_{ij} \in \{0, 1\}, (i, j) \in \mathcal{E}\}$  to indicate if the edge from vertex  $i$  to  $j$  was assigned. With this graph and these decision variables, the bottleneck assignment problem can be formulated as

$$\min_{\pi} \max_{(i,j) \in \mathcal{E}} \pi_{ij} w_{ij} \quad (1a)$$

$$\text{subject to } \sum_{i \in \mathcal{V}_A} \pi_{ij} = 1, \quad j \in \mathcal{V}_T \quad (1b)$$

$$\sum_{j \in \mathcal{V}_T} \pi_{ij} \leq 1, \quad i \in \mathcal{V}_A. \quad (1c)$$

If  $\pi_{ij} = 1$  then we say that vertex  $i$  is assigned to vertex  $j$ . In the rest of the paper the double indices associated with edge  $(i, j) \in \mathcal{E}$  will be represented as a single index. Constraints (1b) and (1c) ensure that every vertex is assigned to at most one other vertex. Two edges are called adjacent if they share a common vertex. A matching in  $\mathcal{G}$  is a set of pairwise non-adjacent edges. That is, no two edges share a common vertex. A *maximum matching*  $\mathcal{M}$  in  $\mathcal{G}$  is a matching with the largest number of edges. The bottleneck problem finds the maximum matching on the graph which minimizes the worst case edge weight. In this paper we focus on maximal cardinality bipartite matchings, where in at least one of the vertex sets  $\mathcal{V}_A$  all vertices are included in the matching. However, for brevity, we will refer to maximal cardinality bipartite matchings as maximal matchings.

The bottleneck assignment problem is well studied [9], and there are a many algorithms that solve it. The threshold algorithm, Algorithm 1 below, solves the bottleneck assignment problem [9], and provides intuition used later in the paper. The algorithm first sorts the edge set in descending according to their weights (line 1). If a maximum matching exists within the sorted edge set (line 3), then the edge with the greatest weight is removed (line 5). Checking for a maximum matching and removing the greatest weight edge is repeated until no maximum matching exists. The algorithm terminates with the last removed edge being the bottleneck edge. Algorithm 1 has complexity  $\mathcal{O}(T(N)N)$  where  $T(N)$  is the complexity for checking the existence of a maximum matching in the set of edges  $|\mathcal{E}| = N$ . The most common method is the Hopcroft-Karp Algorithm, which has complexity  $\mathcal{O}(|\mathcal{E}|\sqrt{|\mathcal{V}|})$  [9]. There are faster probabilistic methods, see [9] for a survey of methods. This gives Algorithm 1 complexity  $\mathcal{O}(N^2\sqrt{|\mathcal{V}|})$ . Note that Algorithm 1 is given here only for intuition and context, the methods described below pertain only to the robustness analysis of the optimal solution. The actual algorithm used

**Data:**  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$

**Result:** bottleneckAssignment

```

1  $\bar{\mathcal{E}} \leftarrow \mathcal{E}$ ;
2 while Maximum Matching  $\in \bar{\mathcal{E}}$  do
3    $e \leftarrow \operatorname{argmax}_{e \in \bar{\mathcal{E}}} \mathcal{W}$ ;
4    $\bar{\mathcal{E}} \leftarrow \bar{\mathcal{E}} \setminus \{e\}$ ;
5 end
6 return  $e \setminus \setminus$  The bottleneck edge

```

**Algorithm 1:** Threshold Algorithm for solving the BAP

to find the bottleneck is irrelevant to our results.

To formally define robustness related to the bottleneck assignment problem, we first define the perturbed graph.

**Definition 1.** Given a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ , define the perturbed graph  $\bar{\mathcal{G}} = (\mathcal{V}, \mathcal{E}, \bar{\mathcal{W}})$  such that  $\bar{\mathcal{W}} = \{w_e + \delta_e \mid e \in \mathcal{E}\}$ , where  $w_e \in \mathcal{W}$  is the weight of edge  $e$  from the original graph and  $\delta_e \in D$  is some perturbation from a set of scalar perturbations  $D \subseteq \mathbb{R}$ .

Define the mapping  $A(\mathcal{G})$  which takes a graph and returns the index of the bottleneck edge. If there are multiple equivalent bottleneck edges, then the mapping will return the set of them.

**Definition 2.** Let  $e^* \in A(\mathcal{G})$  be the bottleneck edge of the graph  $\mathcal{G}$ . If  $e^* \in A(\bar{\mathcal{G}})$ , for the perturbed weight graph in Definition 1, then the bottleneck assignment  $e^*$  is robust to the set of perturbations  $\{\delta_e \mid e \in \mathcal{E}\}$ .

With this definition of robustness, the problem can be stated as follows.

**Problem.** Find the set of allowable perturbations  $\Lambda$  such that the bottleneck assignment  $e^* \in \mathcal{G}$  is robust to the perturbation  $\lambda = \{\delta_e \mid e \in \mathcal{E}\}$  if and only if  $\lambda \in \Lambda$ .

**Remark.** The parametrization of the set  $\Lambda$  as defined in the problem statement has tangible repercussions for task assignment under uncertainty. For example, suppose the graph represents a task assignment in a physical multi-agent system. The agents and tasks are the vertices, and the edges are the possible assignments of agents to tasks, each with an associated weight. If the uncertainties associated with the assignment weights fall within the set of allowable perturbations  $\Lambda$ , then the bottleneck assignment is guaranteed to remain optimal, preventing any need for reassignment.

## III. ROBUSTNESS OF THE BOTTLENECK

To parametrize the set of allowable perturbations  $\Lambda$ , we first propose an optimization problem to find a subset  $\Lambda_{\Delta} = \{\delta_e \mid |\delta_e| \leq \Delta, e \in \mathcal{E}\} \subseteq \Lambda$

$$\max_{\Delta \in \mathbb{R}} \Delta \quad (2a)$$

$$\text{s.t. } A(\bar{\mathcal{G}}) = A(\mathcal{G}), \quad (2b)$$

$$\bar{\mathcal{W}} = \{w_e + \delta_e \mid w_e \in \mathcal{W}, |\delta_e| \leq \Delta, e \in \mathcal{E}\}, \quad (2c)$$

where we have continued to use the graph notation presented in Definition 1.

To find  $\Lambda_\Delta$ , we first define three bottleneck edges

$$e^- \in A(\mathcal{G}^-), \quad (3)$$

$$e^* \in A(\mathcal{G}), \quad (4)$$

$$e^+ \in A(\mathcal{G}^+). \quad (5)$$

These are the bottleneck edges of the original graph  $\mathcal{G}$  and of two subgraphs  $\mathcal{G}^-$  and  $\mathcal{G}^+$ . Define  $\mathcal{G}^-$  to be the graph constructed by removing the vertices, edges, and weights adjacent to and including  $e^*$  and  $\mathcal{G}^+$  to be constructed by removing only the edge and weight  $e^*, w_{e^*}$  from the original graph. The differences between the weights of these three bottleneck edges  $e^-, e^*$ , and  $e^+$  will bound the allowable perturbations, leading to the parameterization of  $\Lambda_\Delta$ .

**Lemma 1.** *Consider the indices  $e^-, e^*$ , and  $e^+$ , defined in (3) to (5). Then  $w_{e^-} \leq w_{e^*}$ , and if  $e^+$  exists then  $w_{e^*} \leq w_{e^+}$ .*

*Proof.* First we will prove that if  $e^+$  exists,  $w_{e^*} \leq w_{e^+}$ . By the definition of  $e^*$ , there is no maximum matching on the original graph  $\mathcal{G}$  composed of edges that all have weight less than  $w_{e^*}$ . Since  $w_{e^+}$  is an edge with the greatest weight in a maximum matching on the graph  $\mathcal{G}^+$ , which is missing the edge  $e^*$ ,  $w_{e^+}$  must be greater than or equal to  $w_{e^*}$ .

Next, to obtain a contradiction, assume  $w_{e^-}$  is greater than  $w_{e^*}$ . By the definition of  $e^*$ , there exists a maximum matching on  $\mathcal{G}$  with  $e^*$  the greatest weight edge in that matching. Let  $\pi^* \subseteq \mathcal{E}$  represent the edges in that matching. The set  $\pi^* \setminus \{e^*\} \subseteq \mathcal{E}^-$  and is a maximum matching on  $\mathcal{G}^-$ . However, the edge with the greatest weight in  $\pi^* \setminus \{e^*\}$  has weight less than or equal to  $w_{e^*}$ , which contradicts  $w_{e^-}$  being greater than  $w_{e^*}$ .  $\square$

Using the concept of the subgraphs  $\mathcal{G}^-$  and  $\mathcal{G}^+$ , we can bound  $\Delta$  from (2a), and construct  $\Lambda_\Delta$ .

**Theorem 1.** *Let  $w_{e^*}$  be the weight of the bottleneck edge  $e^*$  on the graph  $\mathcal{G}$ , and let  $w_{e^-}$ , and  $w_{e^+}$  be the bottleneck weights of the subgraphs  $\mathcal{G}^-$  and  $\mathcal{G}^+$ , respectively, as defined in (3) to (5). The bottleneck assignment is robust, in the sense of Definition 2, to perturbations in  $\Lambda_\Delta$  where*

$$0 \leq \Delta \leq \frac{1}{2} \min(w_{e^*} - w_{e^-}, w_{e^+} - w_{e^*}). \quad (6)$$

*In the case that the graph  $\mathcal{G}^+$  as defined in (5) does not have a maximal matching, the perturbations are bounded by*

$$0 \leq \Delta \leq \frac{1}{2} (w_{e^*} - w_{e^-}). \quad (7)$$

*Proof.* We will prove by contradiction that no edge  $x$  with weight  $w_x \neq w_{e^*}$  will be the bottleneck of the perturbed weight graph under some set of allowable perturbations.

Let  $\bar{w}_i = w_e + \delta_e$  be weight of edge  $e$  after perturbation. Note that  $e^*$  and  $e^-$  guarantee the existence of a maximum matching with greatest weight  $w_{e^*}$  and second greatest weight  $w_{e^-}$ . Let  $\pi^*$  be the set of edges corresponding to that maximum matching.

To obtain a contradiction, let the edge  $x \neq e^*$  be the bottleneck of the perturbed weight graph  $\bar{\mathcal{G}}$  as in Definition 1, with perturbed weight  $\bar{w}_x \neq \bar{w}_{e^*}$ .

We will first establish that  $\bar{w}_x$  must be strictly less than  $\bar{w}_{e^*}$ . By the definition of the bottleneck,  $\bar{w}_x$  is less than or equal to the greatest weight edge in every maximum matching on the perturbed weight graph. Therefore,  $\bar{w}_x$  is less than or equal to the greatest weight edge in the set  $\pi^*$ . By Lemma 1 and the bound on the perturbations,  $\bar{w}_{e^*}$  is the greatest weight edge in the matching  $\pi^*$ . Therefore,  $\bar{w}_x < \bar{w}_{e^*}$ . Additionally, because  $\bar{w}_x$  is the bottleneck edge, it must be the greatest weight edge in a maximum matching  $\pi^x$ , and thus each edge in  $\pi^x$  must have a perturbed weight less than  $\bar{w}_{e^*}$  and clearly  $e^* \notin \pi^x$ .

We will now establish that since  $\bar{w}_x < \bar{w}_{e^*}$ , it cannot be the bottleneck edge. Because of the bound on the perturbations, the only edges which can be perturbed to have weight less than  $\bar{w}_{e^*}$  are those edges which have unperturbed weight  $w_j < w_{e^+}$ . Therefore, every edge in the maximal matching  $\pi^x$  must have an unperturbed weight less than  $w_{e^+}$ . However, by the definition of  $e^+$ , there is no maximal matching composed of weights less than  $w_{e^+}$  that does not contain  $e^*$ . Therefore,  $\pi^x$  does not exist and  $\bar{w}_x$  cannot be the new bottleneck edge.  $\square$

**Example 1.** *Consider a scenario where  $|\mathcal{V}_A| = |\mathcal{V}_T| = 3$  and  $\mathcal{E} = \mathcal{V}_A \times \mathcal{V}_T$ . This could represent a physical scenario with a fully connected set of three agents and tasks. The weight set  $\mathcal{W}$  is represented by a matrix  $C$ , where  $w_{ij}$  is its  $ij$ -th element. This is a convenient representation and will be used for the rest of the paper. This matrix is termed the cost matrix and for this example is*

$$C = \begin{bmatrix} 3 & 2 & 1 \\ 4 & 5 & 6 \\ 9 & 8 & 7 \end{bmatrix}.$$

*Algorithm 1 can be applied to this graph. The sorted list of weights is clearly the integers from 9 to 1, and initially there are six maximum matchings in this graph. Removing the greatest weight edge leaves 4 maximum matchings in the graph. Removing the greatest weight edge again leaves us with the graph whose corresponding cost matrix is*

$$C = \begin{bmatrix} 3 & 2 & 1 \\ 4 & 5 & 6 \\ \infty & \infty & 7 \end{bmatrix}.$$

where  $\infty$  represents the absence of an edge.

Here if we were to remove the greatest weight edge there would be no maximum matchings available, so the edge with weight 7 is the bottleneck. We can now define the cost matrices of the two subgraphs  $\mathcal{G}^-$  and  $\mathcal{G}^+$  to be

$$C^- = \begin{bmatrix} 3 & 2 \\ 4 & 5 \end{bmatrix}, C^+ = \begin{bmatrix} 3 & 2 & 1 \\ 4 & 5 & 6 \\ 9 & 8 & \infty \end{bmatrix}.$$

Running Algorithm 1 on these two subgraphs, we find  $e^-$  and  $e^+$  to be the edges with weight 4 and 8 respectively. In accordance with Lemma 1,  $w_{e^-} = 4 \leq w_{e^*} = 7 \leq w_{e^+} = 8$ . Using (6), we find the  $\Delta$  value for this graph to be  $\frac{1}{2}$ . Therefore, the graph  $\mathcal{G}$  is robust to any set of weight perturbations such that no perturbation has magnitude greater than or equal to  $\frac{1}{2}$ .  $\square$

Equation (6) states that as long the perturbations are all within the interval  $[-\Delta, \Delta]$ , the optimal assignment remains unchanged. This results coincides with the result presented in [8]. However, by relaxing these constraints, allowing the intervals to be unique and asymmetric, we are able to expand this set of allowable perturbation intervals.

Note that in the proof for Theorem 1, the salient property of  $\Delta$  is that edges with weight  $w_j \geq w_{e^+}$  cannot be perturbed to have weight  $\bar{w}_j < \bar{w}_{e^*}$ , and edges with weight  $w_j \leq w_{e^-}$  cannot be perturbed to have weight  $\bar{w}_j > \bar{w}_{e^*}$ . However, this property can be maintained with larger and asymmetric bounds on the perturbations. Additionally, the edges which are between  $e^-, e^*, e^+$  are irrelevant to the method, they do not change the bottleneck edge regardless of their weight. Therefore, for edges between  $e^-$  and  $e^+$  other than  $e^*$ , we need not bound their perturbations at all. This leads us to Algorithm 2 below.

**Proposition 1.** *The bottleneck assignment is robust, in the sense of Definition 2, to the set of perturbations defined in Algorithm 2.*

*Proof.* Let  $e^-, e^*$ , and  $e^+$  be defined as in (3) to (5), and let  $\Delta^+$  and  $\Delta^-$  be defined by Algorithm 2. Let  $\bar{w}_j$  represent the perturbed weight of edge  $j$ . Define the sets of upper and lower bounds returned by Algorithm 3 as  $\mathcal{U}$  and  $\mathcal{L}$ . Let  $(\mathcal{L}_j, \mathcal{U}_j)$  represent the allowable perturbation for edge  $j$ , such that  $\bar{w}_j - w_j \in (\mathcal{L}_j, \mathcal{U}_j)$ .

If the bounds provided by Algorithm 2 guarantee  $\bar{w}_j \geq \bar{w}_{e^*}$  for any edge  $j$  with weight  $w_j \geq w_{e^+}$ , and  $\bar{w}_j \leq \bar{w}_{e^*}$  for any edge  $j$  with weight  $w_j \leq w_{e^-}$ , then the proof for Theorem 1 applies to the set of bounds from Algorithm 2.

For any edge  $j$  with weight  $w_j \geq w_{e^+}$ ,

$$\bar{w}_j > w_j + \mathcal{L}_j = w_j + (w_{e^*} + \Delta^+ - w_j) > \bar{w}_{e^*}.$$

For any edge  $j$  with weight  $w_j \leq w_{e^-}$ ,

$$\bar{w}_j < w_j + \mathcal{U}_j = w_j + (w_{e^*} - \Delta^- - w_j) < \bar{w}_{e^*}.$$

**Data:**  $(e^*, \mathcal{G})$

```

1  $e^- \leftarrow A(\mathcal{G}^-)$  ;
2  $e^+ \leftarrow A(\mathcal{G}^+)$  ;
3  $\mathcal{U} \setminus \setminus$  Upper bound on the allowable perturbations ;
4  $\mathcal{L} \setminus \setminus$  Lower bound on the allowable perturbations ;
5  $\Delta^+ \leftarrow \frac{w_{e^+} - w_{e^*}}{2}$  ;
6  $\Delta^- \leftarrow \frac{w_{e^*} - w_{e^-}}{2}$  ;
7 for  $w_j \in \mathcal{W}$  do
8   if  $w_j \geq w_{e^+}$  then
9      $\mathcal{L}[j] = (w_{e^*} + \Delta^+) - w_j$  ;
10  else if  $w_j \leq w_{e^-}$  then
11     $\mathcal{U}[j] = (w_{e^*} - \Delta^-) - w_j$  ;
12  else if  $j \neq e^*$  then
13     $\mathcal{L}[j] = -\infty$  ;
14     $\mathcal{U}[j] = \infty$  ;
15  else
16     $\mathcal{L}[j] = -\Delta$  ;
17     $\mathcal{U}[j] = \Delta$  ;
18  end
19 end
20 return  $\mathcal{U}, \mathcal{L}$  ;
```

**Algorithm 2:** Relaxed  $\Delta$  Algorithm for Robustness

Therefore, the perturbation intervals provided by Algorithm 2 have the same properties as those provided in Theorem 1, and the bottleneck assignment is robust to perturbations within the bounds provided by Algorithm 2.  $\square$

Consider the case where an edge with weight greater than the bottleneck is subject to a positive perturbation, or an edge with weight less than the bottleneck edge is subject to a negative perturbation. The bottleneck assignment is clearly robust to these types perturbations. This intuition provides additional insight into the set of allowable perturbations  $\Lambda$ .

**Proposition 2.** *Let  $w_x$  be the weight of an edge  $x \in \mathcal{E}$ ,  $\bar{w}_x$  be the perturbed weight of that edge, and  $e^* \in \mathcal{E}$  be the bottleneck as defined in (4). The bottleneck assignment is robust, in the sense of Definition 2, to any perturbations for which  $\bar{w}_x > \bar{w}_{e^*}$  iff  $w_x > w_{e^*}$ , and  $\bar{w}_x < \bar{w}_{e^*}$  iff  $w_x < w_{e^*}$  for all  $x \in \mathcal{E}$ .*

*Proof.* Let  $G$  be the set of all edges with weights greater than the bottleneck edge  $e^*$ , and  $L$  the set of all edges with weights less than the bottleneck edge  $e^*$ . By the definition of the bottleneck edge, there is guaranteed to be a maximum matching with all edges in the set  $\{L \cup e^*\}$ , and no maximum matching with all edges in the set  $L$ . By the definition of the set  $G$ , the maximum matching of edges from the set  $\{L \cup e^*\}$  will all have lower weights than any edges in the set  $G$ . Thus, every edge in the set  $G$  is excluded from being the bottleneck. Therefore, the ordering of the edges within the sets  $G$  and  $L$  has no impact on the bottleneck.  $\square$

**Remark.** *Proposition 2 demonstrates that the allowable perturbations on all edges with weight greater than the*

bottleneck are unbounded above, and the allowable perturbations on all edges with weights less than the bottleneck are unbounded below. In light of this, we will refer to the allowable perturbation sets as half space constraints, because they will only specify an upper or lower bound with the other bound being negative or positive infinity.

Finally, note that the method for constructing the set  $\Lambda_\Delta$  and the subsequent expansion of the allowable perturbation intervals has nearly the order computational complexity as the BAP, because it involves solving the bottleneck assignment problem on two subgraphs. One of these subgraphs,  $\mathcal{G}^+$ , is only one edge smaller than the original. However, by using only Proposition 2, we can construct a different set of allowable perturbation intervals, as shown in Algorithm 3.

**Data:**  $(e^*, \mathcal{W})$

```

1  $\mathcal{U} \setminus \setminus$  Upper bound on the allowable perturbations ;
2  $\mathcal{L} \setminus \setminus$  Lower bound on the allowable perturbations ;
3  $\Delta^+ \leftarrow \frac{1}{2} \min(|w_j - w_{e^*}|, \forall \{j \mid w_j \geq e^*\})$  ;
4  $\Delta^- \leftarrow \frac{1}{2} \min(|w_j - w_{e^*}|, \forall \{j \mid w_j \leq e^*\})$  ;
5 for  $w_j \in \mathcal{W}$  do
6   if  $w_j > w_{e^*}$  then
7      $\mathcal{L}[j] = (w_{e^*} + \Delta^+) - w_j$  ;
8   else if  $w_j < w_{e^*}$  then
9      $\mathcal{U}[j] = (w_{e^*} - \Delta^-) - w_j$  ;
10  else
11     $\mathcal{L}[j] = -\Delta^-$  ;
12     $\mathcal{U}[j] = \Delta^+$  ;
13  end
14 end
15 return  $\mathcal{U}, \mathcal{L}$  ;
```

**Algorithm 3:** Naive Algorithm for Robustness

**Proposition 3.** Algorithm 3 constructs a set of allowable perturbation intervals in  $\mathcal{O}(N)$  where  $N = |\mathcal{E}|$ .

*Proof.* Define  $w_x$  to be the weight of an edge  $x \in \mathcal{E}$ ,  $\bar{w}_x$  to be the perturbed weight of that edge, and  $e^* \in \mathcal{E}$  be the bottleneck edge as defined in (4). Define the sets of upper and lower bounds returned by Algorithm 3 as  $\mathcal{U}$  and  $\mathcal{L}$ . Let  $(\mathcal{L}_j, \mathcal{U}_j)$  represent the allowable perturbation for edge  $j$ , such that  $\bar{w}_j - w_j \in (\mathcal{L}_j, \mathcal{U}_j)$ .

Let  $G$  be the set of edges with weights greater than the bottleneck, and  $L$  the set of edges with weight less than the bottleneck. According to Proposition 2, if these sets are unchanged by the perturbation then the bottleneck assignment is robust to that perturbation. Let  $w_j > w_{e^*}$  be the weight of an edge in set  $G$ ,

$$\bar{w}_j > w_j + \mathcal{L}_j = w_j + (w_{e^*} + \Delta^+ - w_j) > \bar{w}_{e^*}.$$

Therefore,  $\bar{w}_j > \bar{w}_{e^*}$  for all elements of  $G$ , so  $G$  is unchanged by the perturbation. Let  $w_j < w_{e^*}$  be the weight of an edge in set  $L$ ,

$$\bar{w}_j < w_j + \mathcal{U}_j = w_j + (w_{e^*} - \Delta^- - w_j) < \bar{w}_{e^*}.$$

Therefore,  $\bar{w}_j < \bar{w}_{e^*}$  for all elements of  $L$ , so  $L$  is unchanged by the perturbation.  $G$  and  $L$  are unchanged by the perturbation, so the bottleneck assignment is robust to this set of perturbations.

Algorithm 3 constructs these bounds by performing one pass through the weight set to find the minimum distance weight to the bottleneck, and another pass through the edges to construct their upper and lower bounds. Therefore, it has complexity  $\mathcal{O}(|\mathcal{E}|)$ .  $\square$

#### IV. NUMERICAL EXAMPLE

First, to illustrate the various bounds, extensions, and applications, we will analyze a small example. Consider a fully connected multi-agent system with 3 agents and 4 tasks with cost matrix

$$C = \begin{bmatrix} 64.5 & 79.2 & 25.0 & 9.8 \\ 85.9 & 81.2 & 21.5 & 28.3 \\ 47.1 & 12.1 & 41.3 & 35.7 \end{bmatrix}.$$

The first step towards defining the set of allowable perturbations is to run the bottleneck assignment algorithm on the full graph to find  $e^*$ . The bottleneck edge  $e^*$  this example is  $(2, 3)$  with weight 21.5. The cost matrices of the two subgraphs  $\mathcal{G}^-$  and  $\mathcal{G}^+$  are

$$C^- = \begin{bmatrix} 64.5 & 79.2 & 9.8 \\ 47.1 & 12.1 & 35.7 \end{bmatrix},$$

$$C^+ = \begin{bmatrix} 64.5 & 79.2 & 25.0 & 9.8 \\ 85.9 & 81.2 & \infty & 28.3 \\ 47.1 & 12.1 & 41.3 & 35.7 \end{bmatrix}.$$

Running the bottleneck algorithm on the two subgraphs gives  $e^- = (3, 2)$  with weight  $w_{e^-} = 12.1$  and  $e^+ = (2, 4)$  with weight  $w_{e^+} = 28.3$ . To find the set of allowable perturbation intervals we first use Algorithm 2, because we have already identified  $e^-$ ,  $e^*$ , and  $e^+$ . The results are presented in Table I.

TABLE I  
ALLOWABLE PERTURBATIONS FROM ALGORITHM 2

	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>
A <sub>1</sub>	[-39.6, $\infty$ ]	[-54.3, $\infty$ ]	[- $\infty$ , $\infty$ ]	[- $\infty$ , 7.0]
A <sub>2</sub>	[-61.0, $\infty$ ]	[-56.3, $\infty$ ]	[-4.7, 3.4]	[-3.4, $\infty$ ]
A <sub>3</sub>	[-22.2, $\infty$ ]	[- $\infty$ , 4.7]	[-16.4, $\infty$ ]	[-10.8, $\infty$ ]

Next, we compare the allowable perturbation intervals from running Algorithm 3 over the same graph. The results are presented in Table II

Note that for some edges Algorithm 3 and Algorithm 2 provide the same bounds. However, the smallest intervals are larger using Algorithm 2. Some edges are also entirely unbounded using Algorithm 2.

In both algorithms edges above the bottleneck can be infinitely positively perturbed, and edges below the bottleneck can be likewise infinitely negatively perturbed.

TABLE II  
ALLOWABLE PERTURBATIONS FROM ALGORITHM 3

	T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>	T <sub>4</sub>
A <sub>1</sub>	[-41.25, ∞]	[-55.95, ∞]	[-1.75, ∞]	[-∞, 7.0]
A <sub>2</sub>	[-62.65, ∞]	[-57.95, ∞]	[-4.7, 1.75]	[-5.05, ∞]
A <sub>3</sub>	[-23.85, ∞]	[-∞, 4.7]	[-18.05, ∞]	[-12.45, ∞]

Therefore, to compare the algorithms, we propose examining the magnitude of allowable perturbation in the non-infinite direction. However, note that this method does not capture the number of unbounded edges from Algorithm 2.

Fig. 1 presents simulation results. The graphs generated have  $|\mathcal{V}_A| = |\mathcal{V}_T| = [3, 100]$  nodes and are fully connected  $\mathcal{E} = \mathcal{V}_A \times \mathcal{V}_T$ . The plots show the minimum magnitude allowable perturbation, or the “tightest” allowable perturbation bound. For each graph size from  $[3, 100]$  we randomly generated 1000 graphs with weights randomly sampled from  $[0, 100]$ .

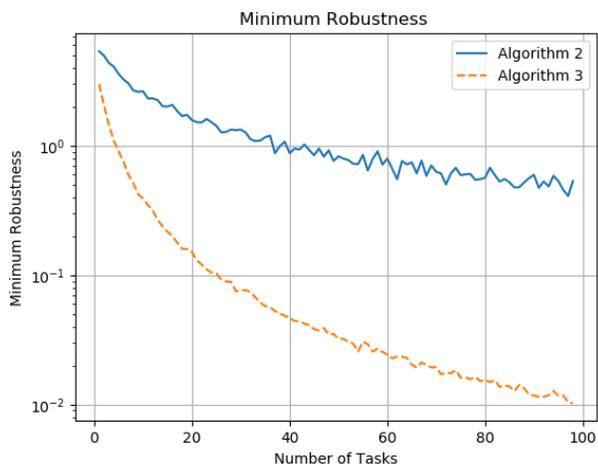


Fig. 1. Minimum size of allowable perturbations

Algorithm 2 provides a larger minimum size intervals as the problem size grows. The minimum robustness of Algorithm 3 is driven entirely by the density of the weights, and the larger the problems the more likely there are many weights near the bottleneck. However, Algorithm 2’s minimums also drop as the problem size grows larger, because its performance is tied to the density of the greatest weight edges in the maximum matchings, which will also increase with the number of tasks in the graph.

## V. CONCLUDING REMARKS

In this paper the robustness of the bottleneck assignment problem (BAP) to a set of perturbations is analyzed. Two methods of approximating the set of allowable perturbations are proposed, one with the same order computational complexity as the BAP, and the other is linear in the number

of edges in the graph. These approaches differ from the previous approaches in literature by allowing all weights to vary simultaneously and independently. Simulations showed that the higher order complexity bound provided better worst case robustness. The robustness of the graph can be used to guarantee the optimality of the bottleneck assignment if the errors in each weight fall within the set of allowable perturbations.

In future work, similar robustness for the linear assignment problem, the general assignment problem, and various other assignment problem formulations will be investigated. Another possible future problem would be examining the geometric implications of these robustness intervals, perhaps informing a method of deploying multi-agent system in a pattern likely to lead to robust solutions.

## REFERENCES

- [1] C.-J. Lin and U.-P. Wen, “Sensitivity analysis of objective function coefficients of the assignment problem,” *Asia-Pacific Journal of Operational Research*, vol. 24, no. 02, pp. 203–221, 2007.
- [2] B. Klaus, F. Payot, et al., “Paths to stability in the assignment problem,” *Cahier de recherches conomiques du DEEP*, no. 13.14, 2013.
- [3] G. Kindervater, A. Volgenant, G. de Leve, and V. van Gijlswijk, “On dual solutions of the linear assignment problem,” *European Journal of Operational Research*, vol. 19, no. 1, pp. 76 – 81, 1985. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0377221785903108>
- [4] B. Jansen, J. De Jong, C. Roos, and T. Terlaky, “Sensitivity analysis in linear programming: just be careful!” *European Journal of Operational Research*, vol. 101, no. 1, pp. 15–28, 1997.
- [5] I. Shames, A. Dostovalova, J. Kim, and H. Hmam, “Task allocation and motion control for threat-seduction decoys,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Dec 2017, pp. 4509–4514.
- [6] A. Ravindran and V. Ramaswami, “On the bottleneck assignment problem,” *Journal of Optimization Theory and Applications*, vol. 21, no. 4, pp. 451–458, Apr 1977. [Online]. Available: <https://doi.org/10.1007/BF00933089>
- [7] A. Volgenant and C. W. Duin, “Improved polynomial algorithms for robust bottleneck problems with interval data,” *Computers & Operations Research*, vol. 37, no. 5, pp. 909–915, 2010.
- [8] Y. N. Sotskov, V. Leontev, and E. N. Gordeev, “Some concepts of stability analysis in combinatorial optimization,” *Discrete Applied Mathematics*, vol. 58, pp. 169–190, 1995.
- [9] R. E. Burkard and E. Cela, “Linear assignment problems and extensions,” in *Handbook of combinatorial optimization*. Springer, 1999, pp. 75–149.