

Exploiting Structure in the Bottleneck Assignment Problem[★]

Mitchell Khoo^{*} Tony A. Wood^{**} Chris Manzie^{**}
Iman Shames^{**}

^{*} *Department of Electrical and Electronic Engineering at the University of Melbourne, (e-mail: khoom1@student.unimelb.edu.au).*

^{**} *(e-mail: {wood.t,manziec,iman.shames}@unimelb.edu.au)*

Abstract: An assignment problem arises when there exists a set of tasks that must be allocated to a set of agents. The bottleneck assignment problem (BAP) has the objective of minimising the most costly allocation of a task to an agent. Under certain conditions the structure of the BAP can be exploited such that subgroups of tasks are assigned separately with lower complexity and then merged to form a combined assignment. In particular, we discuss merging the assignments from two separate BAPs and use the solution of the subproblems to bound the solution of the combined problem. We also provide conditions for cases where the solution of the subproblems produces an exact solution to the BAP over the combined problem. We then introduce a particular algorithm for solving the BAP that takes advantage of this insight. The methods are demonstrated in a numerical case study.

Keywords: Algorithms; Decision-making; Graph theory; Agents; Optimization problems.

1. INTRODUCTION

An assignment problem arises when multiple tasks are to be allocated to multiple agents. For example, situations where jobs are to be assigned to a group of workers or passengers positioned at different locations in a city are to be picked up by a fleet of cars. Tasks can be assigned based on many different criteria. See Gerkey and Mataric (2004), Burkard et al. (2009), and Pentico (2007) for reviews on the different objectives for assignment problems.

One particular objective is to assign tasks to agents such that the total cost of the assignment is minimised. This type of assignment problem is called the linear assignment problem (LAP). The Hungarian Method in Kuhn (1955) is a well-studied algorithm for solving the LAP. In Chopra et al. (2017), a distributed version of the Hungarian Method is presented; a distributed algorithm is one that does not rely on a centralised decision-maker for computation. In Bertsekas and Castañon (1991) and Zavlanos et al. (2008), so-called auction algorithms are presented to solve the LAP. A greedy algorithm is one where tasks are allocated to agents sequentially. Each allocation is made according to the lowest cost amongst the remaining choices. In Choi et al. (2009), the Consensus-Based Auction Algorithm (CBAA) is presented, which is a greedy algorithm used to obtain suboptimal solutions to the LAP with low computational cost compared to algorithms for solving the LAP exactly.

Another objective is to assign tasks to agents such that the costliest allocation is minimised, which corresponds to the bottleneck assignment problem (BAP). The BAP has application in time-critical problems. For example in

Shames et al. (2017), a set of decoys must travel to a set of positions such that the worst-case positioning time is minimised. In Garfinkel (1971), a threshold algorithm is presented, where a threshold is iteratively increased until it is possible to find an assignment containing only allocations of tasks to agents with costs smaller than the threshold. In Gabow and Tarjan (1988); Punnen and Nair (1994), the bound on the completion time of the threshold algorithm is reduced moving the threshold according to a binary search pattern. In Derigs and Zimmermann (1978), an algorithm is presented that iteratively solves the BAP over an increasing subset of agents and tasks. The subset size is increased until it contains all the agents and tasks. In Khoo et al. (2019), a distributed algorithm for solving the BAP is introduced. There are other variants of the BAP. Such variants include the scheduling problems in Carraraesi and Gallo (1984) and Aggarwal et al. (1986), which require assigning more than one task per agent. In fact, this can be regarded as an example of a time-extended assignment from the taxonomy in Gerkey and Mataric (2004).

In this paper, we focus on the BAP and restrict the scope to having each agent carry out at most one task and each task requiring at most one agent for completion. The contribution of this work is to investigate structure that can be exploited to solve the BAP efficiently. Consider partitioning the sets of agents and tasks, i.e., splitting the assignment problem into two smaller BAPs. We can use the two solutions of the subproblems for solving the combined BAP. Consider the following three ways to exploit the structure of the BAP. We relate each scenario to a ride-sharing application for illustration.

For the first scenario, assume the sets of agents and tasks were partitioned equitably, i.e., none of the subproblems has fewer tasks than agents. Merging the solutions of the

[★] The research is funded by Defence Science and Technology Group through research agreements MyIP: 7558 and MyIP: 7562.

two subproblems forms a valid but possibly suboptimal assignment in the combined problem. In fact, the cost of the merged assignment is an upper bound on the cost of the optimal BAP solution. In a ride-sharing application, two rival companies may assign their own vehicles to their own customers. However, they may find that pooling their resources allows a better service for all customers.

For the second scenario, we define a bottleneck cluster as a group of agents and tasks with small allocation costs amongst each other. When the two subproblems consist of two separate bottleneck clusters, we can determine conditions under which the solutions of the subproblems form an exact solution to the combined problem. Consider two cities each with their own sets of vehicles and customers. If the cities were geographically far apart, there is no benefit for vehicles in one city to serve customers in the other city.

The third scenario relates to the algorithm in Khoo et al. (2019). Knowing the solutions to the two subproblems leads to information about task-to-agents allocations that are particularly costly. We can eliminate suboptimal options when the algorithm is initialised to solve the combined problem. Assume a group of customers has been assigned to vehicles. Then, a new group of customers requests to be picked up. By only considering the idle vehicles for the new customers and not the previously assigned ones, the resulting assignment problem has lower complexity. The solution from the two subproblems can be used as a warm-start to solving the combined problem.

2. PRELIMINARIES

Given an arbitrary undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertex set \mathcal{V} and edge set \mathcal{E} , consider the following definitions found in Hopcroft and Karp (1973) and Khoo et al. (2019).

Definition 1. (Maximum Cardinality Matching). A matching \mathcal{M} in \mathcal{G} is a set of edges such that $\mathcal{M} \subseteq \mathcal{E}$ and no vertex $v \in \mathcal{V}$ is incident with more than one edge in \mathcal{M} . A maximum cardinality matching (MCM) is a matching \mathcal{M}_{max} in \mathcal{G} of maximum cardinality.

Let \mathcal{A}_b be a set of agents and \mathcal{B}_b be a set of tasks, where $\mathcal{A}_b \cap \mathcal{B}_b = \emptyset$. Consider an arbitrary complete bipartite graph $\mathcal{G}_b = (\mathcal{V}_b, \mathcal{E}_b)$ with vertex set $\mathcal{V}_b = \mathcal{A}_b \cup \mathcal{B}_b$ and edge set $\mathcal{E}_b = \{\{i, j\} | i \in \mathcal{A}_b, j \in \mathcal{B}_b\}$. Let $\mathcal{C}(\mathcal{G}_b)$ be the set of all MCMs of \mathcal{G}_b . Let $w : \mathcal{E}_b \mapsto \mathbb{R}$ map edges to real-valued weights. The BAP for graph \mathcal{G}_b is formulated as

$$BOT(\mathcal{G}_b) : \min_{\mathcal{M} \in \mathcal{C}(\mathcal{G}_b)} \max_{\{i, j\} \in \mathcal{M}} w(\{i, j\}). \quad (1)$$

Definition 2. (Bottleneck edge). A bottleneck edge of graph \mathcal{G}_b is any $e \in \arg \max_{\{i, j\} \in \mathcal{M}} w(\{i, j\})$, for any MCM \mathcal{M} that is a minimiser of $BOT(\mathcal{G}_b)$.

Definition 3. (Neighbours). The set of neighbours of vertex $v \in \mathcal{V}$ in \mathcal{G} is defined as $N_v = \{k | \{v, k\} \in \mathcal{E}\}$.

Note that given a vertex $v \in \mathcal{V}$, $\forall k \in N_v$, $v \in N_k$.

Definition 4. (Path). Let a sequence of distinct vertices $v_1, v_2, \dots, v_{l+1} \in \mathcal{V}$ be such that for $k = 1, 2, \dots, l$, $v_{k+1} \in N_{v_k}$. The set of edges $\mathcal{P} = \{\{v_k, v_{k+1}\}\}_{k=1, 2, \dots, l}$ is then said to be a path between v_1 and v_{l+1} , with length l .

Definition 5. (Alternating path). Given a matching \mathcal{M} and a path \mathcal{P} , \mathcal{P} is an alternating path relative to \mathcal{M} if and only if each vertex that is incident to an edge in \mathcal{P}

is incident with no more than one edge in $\mathcal{P} \cap \mathcal{M}$ and no more than one edge in $\mathcal{P} \setminus \mathcal{M}$.

Definition 6. (Free vertex). Given a matching \mathcal{M} , a vertex $v \in \mathcal{V}$ is free if and only if for all $w \in \mathcal{V}$, $\{v, w\} \notin \mathcal{M}$.

Definition 7. (Augmenting path). Given a matching \mathcal{M} and a path \mathcal{P} between vertices v and v' , \mathcal{P} is an augmenting path relative to \mathcal{M} if and only if \mathcal{P} is an alternating path relative to \mathcal{M} and v and v' are both free vertices.

Definition 8. (Alternating tree) Given a matching \mathcal{M} , \mathcal{G} is an alternating tree relative to \mathcal{M} if and only if \mathcal{G} is a tree and any path between the root vertex of \mathcal{G} and every other vertex in \mathcal{G} is an alternating path relative to \mathcal{M} .

3. PROBLEM FORMULATION

Let there be two sets of agents $\mathcal{A}_1 = \{a_1, a_2, \dots, a_{m_1}\}$ and $\mathcal{A}_2 = \{\alpha_1, \alpha_2, \dots, \alpha_{m_2}\}$ and two sets of tasks $\mathcal{B}_1 = \{b_1, b_2, \dots, b_{n_1}\}$ and $\mathcal{B}_2 = \{\beta_1, \beta_2, \dots, \beta_{n_2}\}$. Define the sets $\mathcal{A}_3 := \mathcal{A}_1 \cup \mathcal{A}_2$ and $\mathcal{B}_3 := \mathcal{B}_1 \cup \mathcal{B}_2$. Let $m_3 = m_1 + m_2$ and $n_3 = n_1 + n_2$ and assume $m_1 \geq n_1$ and $m_2 \geq n_2$. For $k = 1, 2, 3$, define $\mathcal{V}_k := \mathcal{A}_k \cup \mathcal{B}_k$, $\mathcal{E}_k := \{\{i, j\} | i \in \mathcal{A}_k, j \in \mathcal{B}_k\}$ and graph $\mathcal{G}_k := (\mathcal{V}_k, \mathcal{E}_k)$. Define $\mathcal{D}(\mathcal{G}_b) := \arg \min_{\mathcal{M} \in \mathcal{C}(\mathcal{G}_b)} \max_{\{i, j\} \in \mathcal{M}} w(\{i, j\})$, the set of solutions to $BOT(\mathcal{G}_b)$ for any bipartite graph \mathcal{G}_b .

Assumption 1. Assume we have $\mathcal{M}_1 \in \mathcal{D}(\mathcal{G}_1)$ and $e_1 \in \arg \max_{\{i, j\} \in \mathcal{M}_1} w(\{i, j\})$, i.e., an arbitrary solution to $BOT(\mathcal{G}_1)$ and a corresponding bottleneck edge of \mathcal{G}_1 .

Assumption 2. Assume we have $\mathcal{M}_2 \in \mathcal{D}(\mathcal{G}_2)$ and $e_2 \in \arg \max_{\{i, j\} \in \mathcal{M}_2} w(\{i, j\})$, i.e., an arbitrary solution to $BOT(\mathcal{G}_2)$ and a corresponding bottleneck edge of \mathcal{G}_2 .

Problem 1. Given Assumptions 1 and 2, find a solution to $BOT(\mathcal{G}_3)$, i.e., find some matching $\mathcal{M}_3 \in \mathcal{D}(\mathcal{G}_3)$.

In Section 4, we define structures of the BAP that can be exploited to solve Problem 1. Then in Section 5, we discuss a specific algorithm that allows us to exploit some structure of the BAP discussed in Section 4.

4. STRUCTURE OF THE BAP

In this section, we discuss structures of the BAP that can be exploited. We first introduce an upper bound on the weight of a bottleneck edge of \mathcal{G}_3 , in terms of the bottleneck edges \mathcal{G}_1 and \mathcal{G}_2 . Then, we introduce bottleneck clusters and provide conditions when the solution to Problem 1 is found by merging matchings \mathcal{M}_1 and \mathcal{M}_2 .

4.1 A Bound on the Optimal BAP Solution

Theorem 1. Under Assumptions 1 and 2, it holds that $\max\{w(e_1), w(e_2)\}$ is an upper bound on $w(e_3)$, where e_3 is a bottleneck edge of \mathcal{G}_3 .

Proof. By definition, $w(e_3) = \max_{\{i, j\} \in \mathcal{M}_3} w(\{i, j\}) \leq \max_{\{i, j\} \in \mathcal{M}} w(\{i, j\})$ for any arbitrary $\mathcal{M} \in \mathcal{C}_3$. Since $\tilde{\mathcal{M}} = \mathcal{M}_1 \cup \mathcal{M}_2 \in \mathcal{C}_3$, $w(e_3) \leq \max_{\{i, j\} \in \tilde{\mathcal{M}}} w(\{i, j\}) = \max\{w(e_1), w(e_2)\}$. \square

Given Assumptions 1 and 2, the set $\mathcal{M}_1 \cup \mathcal{M}_2$ is an MCM of \mathcal{G}_3 . This MCM is possibly suboptimal to $BOT(\mathcal{G}_3)$. However, this bound of the BAP allows us to make a decision before solving Problem 1 exactly. If the suboptimal

solution is sufficient in our application, there is no need to invest further resources to solve Problem 1 exactly.

4.2 Bottleneck Clusters

We now introduce the novel concept of a bottleneck cluster. In Khoo et al. (2019), conditions for determining if an edge is a bottleneck edge of a given graph are presented. We build on this result and discuss corresponding conditions under which $\mathcal{M}_3 = \mathcal{M}_1 \cup \mathcal{M}_2$ is an exact solution to $BOT(\mathcal{G}_3)$ when \mathcal{G}_1 and \mathcal{G}_2 are both bottleneck clusters.

Once again, consider an arbitrary complete bipartite graph $\mathcal{G}_b = (\mathcal{V}_b, \mathcal{E}_b)$. Given \mathcal{M} is an MCM of \mathcal{G}_b , we define $\phi(\mathcal{G}_b, \mathcal{M}) := \mathcal{M} \cup \{e \in \mathcal{E}_b \mid w(e) < \max_{e' \in \mathcal{M}} w(e')\}$, the union of \mathcal{M} and the set of edges that have weight strictly smaller than the largest edge in \mathcal{M} . With this tool we define a bottleneck cluster and a critical bottleneck edge.

Definition 9. (Bottleneck cluster). Let \mathcal{M}_b be a solution to $BOT(\mathcal{G}_b)$. Let $e = \{a_b, b_b\} \in \mathcal{M}_b$ be a bottleneck edge of \mathcal{G}_b . Graph \mathcal{G}_b is a bottleneck cluster relative to e if and only if for any vertex $v \in \mathcal{V}_b$, there exists an alternating path P between v and bottleneck task b_b such that $P \subseteq \phi(\mathcal{G}_b, \mathcal{M}_b)$.

Definition 10. (Critical bottleneck edge). Let \mathcal{M} be an MCM of graph \mathcal{G}_b . Edge e_c is a critical bottleneck edge of \mathcal{G}_b relative to \mathcal{M} if and only if $e_c \in \arg \max_{e \in \mathcal{M}} w(e)$ and $\phi(\mathcal{G}_b, \mathcal{M}) \setminus \{e_c\}$ does not contain an augmenting path relative to $\mathcal{M} \setminus \{e_c\}$.

Lemma 1 allows us to find a new MCM, which will have at least one less edge with weight $\max_{e \in \mathcal{M}} w(e)$ than \mathcal{M} .

Lemma 1. (Proof in Khoo et al. (2019)). Let \mathcal{M} be an arbitrary MCM of graph \mathcal{G}_b . Consider an edge $e \in \arg \max_{e \in \mathcal{M}} w(e)$. An augmenting path $P \subseteq \phi(\mathcal{G}_b, \mathcal{M}) \setminus \{e\}$ exists relative to $\mathcal{M} \setminus \{e\}$ if and only if there exists an MCM \mathcal{M}' of \mathcal{G}_b such that $\mathcal{M}' \subseteq \phi(\mathcal{G}_b, \mathcal{M}) \setminus \{e\}$.

Corollary 1. From Lemma 1, it follows that every critical bottleneck edge of \mathcal{G}_b is a bottleneck edge of \mathcal{G}_b .

An MCM that is a solution to $BOT(\mathcal{G}_b)$ may contain more than one critical bottleneck edge. The following proposition shows how a critical bottleneck edge forms a particular alternating path between the bottleneck agent and bottleneck task.

Assumption 3. Let \mathcal{M}_b be a solution to $BOT(\mathcal{G}_b)$ and let $e_c = \{a_c, b_c\} \in \mathcal{M}_b$ be a critical bottleneck edge of \mathcal{G}_b relative to \mathcal{M}_b .

Proposition 1. Consider Assumption 3. The length-one path $P = \{e_c\}$ is a unique alternating path in $\phi(\mathcal{G}_b, \mathcal{M}_b)$ relative to \mathcal{M}_b between a_c and b_c .

Proof. Path $P = \{e_c\}$ is trivially an alternating path relative to \mathcal{M}_b ; edge $e_c \in \mathcal{M}_b$, so $P \subseteq \phi(\mathcal{G}_b, \mathcal{M}_b)$. It remains to show that there does not exist another. Assume for contradiction that there exists an alternating path $P' \neq P$, $P' \subseteq \phi(\mathcal{G}_b, \mathcal{M}_b)$ relative to \mathcal{M}_b between a_c and b_c . It follows that $e_c \notin P'$ and therefore $P' \subseteq \phi(\mathcal{G}_b, \mathcal{M}_b) \setminus \{e_c\}$. Furthermore, P' is an augmenting path relative to $\mathcal{M}_b \setminus \{e_c\}$. By Definition 10, e_c is not a critical bottleneck edge of \mathcal{G}_b , which contradicts Assumption 3. \square

The following corollary describes the structure of a bottleneck cluster \mathcal{G}_b based on Proposition 1.

Corollary 2. Consider Assumption 3 and let \mathcal{G}_b be a bottleneck cluster with respect to the critical bottleneck edge e_c . We form two subgraphs of \mathcal{G}_b , denoted as $\mathcal{S}_\mu(\mathcal{G}_b) = (\mathcal{V}_\mu, \mathcal{E}_\mu)$ and $\mathcal{S}_\nu(\mathcal{G}_b) = (\mathcal{V}_\nu, \mathcal{E}_\nu)$. Let \mathcal{V}_μ contain the bottleneck agent a_c , and let \mathcal{V}_ν contain the bottleneck task b_c . Let $\mathcal{V}_b = \mathcal{V}_\mu \cup \mathcal{V}_\nu$ and $\mathcal{V}_\mu \cap \mathcal{V}_\nu = \emptyset$. By Definition 9, it must be possible to construct both $\mathcal{S}_\mu(\mathcal{G}_b)$ and $\mathcal{S}_\nu(\mathcal{G}_b)$ to be alternating trees such that $\mathcal{E}_\mu \cup \mathcal{E}_\nu \subseteq \phi(\mathcal{G}_b, \mathcal{M}_b)$. By Proposition 1, for all agents $a' \in \mathcal{V}_\mu \cap \mathcal{A}_b$ and for all tasks $b' \in \mathcal{V}_\nu \cap \mathcal{B}_b$, $\{a', b'\} \notin \phi(\mathcal{G}_b, \mathcal{M}_b)$.

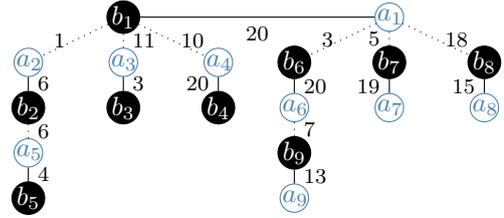


Fig. 1. A bottleneck cluster. Dotted lines represent edges not in matching \mathcal{M}_b , solid lines represent edges in \mathcal{M}_b . Shown here, a set of agents $\{a_1, a_2, \dots, a_9\}$ and a set of tasks $\{b_1, b_2, \dots, b_9\}$. Edge $\{a_1, b_1\}$ is a critical bottleneck edge so Corollary 2 applies.

Fig. 1 illustrates Corollary 2. The graph \mathcal{G}_b is represented by two alternating trees $\mathcal{S}_\nu(\mathcal{G}_b)$ and $\mathcal{S}_\mu(\mathcal{G}_b)$ with roots b_1 and a_1 respectively. The roots a_1 and b_1 are incident to the critical bottleneck edge e_c with weight $w(e_c) = 20$. All edges in both trees are elements of $\phi(\mathcal{G}_b, \mathcal{M}_b)$ as their weights are smaller than or equal to $w(e_c)$ and all edges not in the matching are strictly smaller than $w(e_c)$. Recall from Theorem 1 that $w(e_3) \leq \max\{w(e_1), w(e_2)\}$. Given Assumptions 1 and 2, the contrapositive of the following lemma provides conditions for the bottleneck edge of graph \mathcal{G}_3 to have equal weight to $\max\{w(e_1), w(e_2)\}$.

Lemma 2. Given Assumptions 1 and 2, assume both \mathcal{G}_1 and \mathcal{G}_2 are bottleneck clusters with respect to e_1 and e_2 respectively. Assume e_1 is a critical bottleneck edge of \mathcal{G}_1 relative to \mathcal{M}_1 and e_2 is a critical bottleneck edge of \mathcal{G}_2 relative to \mathcal{M}_2 . Let $w(e_1) \geq w(e_2)$. If $w(e_3) < w(e_1)$, then there exists vertices $i, j \in \mathcal{V}_2$ such that

- i. there exists an edge in \mathcal{E}_3 with weight less than $w(e_1)$ between agent i and a task b' in the vertex set of subgraph $\mathcal{S}_\nu(\mathcal{G}_1)$, and
- ii. there exists an edge in \mathcal{E}_3 with weight less than $w(e_1)$ between task j and an agent a' in the vertex set of subgraph $\mathcal{S}_\mu(\mathcal{G}_1)$, and
- iii. there exists an alternating path P between i and j containing only edges with weight less than $w(e_1)$, and $|P \cap \mathcal{M}_2| > |P \setminus \mathcal{M}_2|$.

Proof. Without loss of generality, let $e_1 = \{a_1, b_1\}$. By Proposition 1, e_1 is the only alternating path between a_1 and b_1 in $\phi(\mathcal{G}_1, \mathcal{M}_1)$. Assume there does not exist vertices i and j such that all i., ii., and iii. are true. Thus, e_1 is the only alternating path between a_1 and b_1 in $\phi(\mathcal{G}_3, \mathcal{M}_1 \cup \mathcal{M}_2)$. By Definition 10, e_1 is also a critical bottleneck edge of \mathcal{G}_3 since $e_1 \in \arg \max_{e \in \mathcal{M}_1 \cup \mathcal{M}_2} w(e)$ and there does not exist an augmenting path in $\phi(\mathcal{G}_3, \mathcal{M}_1 \cup \mathcal{M}_2) \setminus \{e_1\}$ relative to $(\mathcal{M}_1 \cup \mathcal{M}_2) \setminus \{e_1\}$. Thus, $w(e_3) = w(e_1)$. \square

In general, the converse of Lemma 2 does not hold unless we apply some additional assumptions. This leads to the following theorem.

Theorem 2. Given Assumptions 1 and 2, assume both \mathcal{G}_1 and \mathcal{G}_2 are bottleneck clusters with respect to e_1 and e_2 respectively. Assume e_1 is a critical bottleneck edge of \mathcal{G}_1 relative to \mathcal{M}_1 and e_2 is a critical bottleneck edge of \mathcal{G}_2 relative to \mathcal{M}_2 . Assume that $w(e_1) > w(e_2)$. Let $\arg \max_{\{i,j\} \in \mathcal{M}_1} w(\{i,j\})$ be a singleton. It holds that $w(e_3) < w(e_1)$ if and only if there exists vertices $i, j \in \mathcal{V}_2$ such that conditions i., ii., and iii. from Lemma 2 are true.

Proof. The necessary condition for $w(e_3) < w(e_1)$ holds from Lemma 2. We now prove the sufficient condition. Assume there exists vertices i and j such that all i., ii., and iii. are true. Then, aside from path $P = \{e_1 = \{a_1, b_1\}\}$, there exists another alternating path P' between a_1 and b_1 , which does not contain the edge e_1 . Namely, the alternating path P' constructed from the union of the alternating paths between a_1 and b' , b' and i , i and j , j and a' , and a' and b_1 . Thus, there exists an augmenting path $P' \subseteq \phi(\mathcal{G}_3, \mathcal{M}_1 \cup \mathcal{M}_2) \setminus \{e_1\}$ relative to $(\mathcal{M}_1 \cup \mathcal{M}_2) \setminus \{e_1\}$. From Lemma 1, there exists an MCM \mathcal{M}' of \mathcal{G}_3 such that $\mathcal{M}' \in \phi(\mathcal{G}_3, \mathcal{M}_1 \cup \mathcal{M}_2) \setminus \{e_1\}$. By the assumptions on e_1 , $\phi(\mathcal{G}_3, \mathcal{M}_1 \cup \mathcal{M}_2) \setminus \{e_1\}$ contains only edges with weights strictly smaller than $w(e_1)$. Thus, there exists an MCM of \mathcal{G}_3 with all edges having weight smaller than $w(e_1)$, i.e., $w(e_3)$ must be smaller than $w(e_1)$. \square

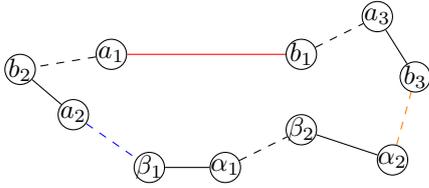


Fig. 2. An illustration of Theorem 2. We have $\mathcal{V}_1 = \{a_1, a_2, a_3\} \cup \{b_1, b_2, b_3\}$ and \mathcal{G}_1 is a bottleneck cluster with respect to $e_1 = \{a_1, b_1\}$. Edge e_1 is shown as a solid red line. We have $\mathcal{V}_2 = \{a_1, a_2\} \cup \{\beta_1, \beta_2\}$ and \mathcal{G}_2 is a bottleneck cluster with respect to $e_2 = \{a_2, \beta_2\}$. The length of each line corresponds to the weight of that edge. Solid lines show edges in $\mathcal{M}_1 \cup \mathcal{M}_2$. Dashed lines show edges in \mathcal{M}_3 and $w(e_3) < w(e_1)$.

Fig. 2 illustrates the sufficient condition of Theorem 2. The orange dashed line is an edge that satisfies the condition i. since $a_2 \in \mathcal{V}_2$ and there exists edge $\{\alpha_2, b_3\} \in \mathcal{V}_3$, where b_3 is a task in the vertex set of $\mathcal{S}_\nu(\mathcal{G}_1)$. The blue dashed line satisfies condition ii. since $\beta_1 \in \mathcal{V}_2$ and there exists edge $\{\beta_1, a_2\} \in \mathcal{V}_3$, where a_2 is an agent in the vertex set of $\mathcal{S}_\mu(\mathcal{G}_1)$. Condition iii. is satisfied since there is an alternating path $P = \{\{\alpha_1, \beta_1\}, \{\alpha_2, \beta_2\}, \{\alpha_1, \beta_2\}\}$ between β_1 and α_2 , and $|\{\{\alpha_1, \beta_1\}, \{\alpha_2, \beta_2\}\}| > |\{\{\alpha_1, \beta_2\}\}|$, i.e., P starts with a dashed line and ends with a dashed line. Corollary 3 follows from Theorem 1 and 2.

Corollary 3. If one or more of conditions i., ii., or iii. in Theorem 2 do not hold, then $\mathcal{M}_3 = \mathcal{M}_1 \cup \mathcal{M}_2$ is a solution to Problem 1.

5. ALGORITHM FOR SOLVING THE BAP

In this section, we discuss how the algorithm from Khoo et al. (2019) makes use of Assumptions 1 and 2 to solve

Problem 1. Let us refer to this algorithm as pruneBAP. Fig. 3 is an illustration of this algorithm.

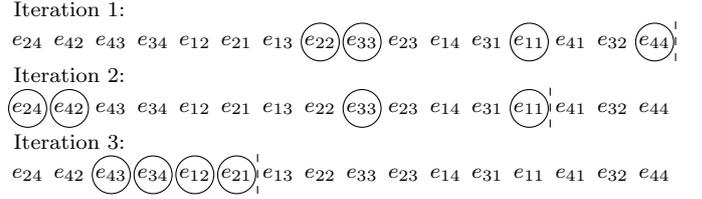


Fig. 3. A demonstration of pruneBAP with $\mathcal{A}_3 = \{a_1, a_2, a_3, a_4\}$ and $\mathcal{B}_3 = \{b_1, b_2, b_3, b_4\}$. Edges in \mathcal{E}_3 are arranged in order of ascending weight, where e_{pq} is the edge between agent a_p and task b_q . At iteration 1, the initial arbitrary MCM is denoted by the 4 circled edges. Edges to the right of the dashed lines have been pruned from \mathcal{E}_3 . Note, $w(e_{44}) \geq w(e_{11}) \geq w(e_{21})$, i.e., with each iteration the weight of the largest edge in the current MCM is non-increasing. The algorithm terminates when a matching of size 4 does not exist in the remaining edges to the left of the dashed line.

5.1 Warm-starting Versus Cold-starting pruneBAP

Solving Problem 1 by pruneBAP with an arbitrary MCM \mathcal{M}_0 at initialisation does not make use of Assumptions 1 and 2. We denote this as a cold-start to pruneBAP. Given Assumptions 1 and 2, consider the following. It holds that the set $\hat{\mathcal{M}} := \mathcal{M}_1 \cup \mathcal{M}_2$ is an MCM of the graph \mathcal{G}_3 . Without loss of generality, let $w(e_1) \geq w(e_2)$. Then, it also holds that e_1 is the largest edge in $\hat{\mathcal{M}}$. We use make use of $\hat{\mathcal{M}}$ to solve Problem 1 by choosing it as the initial MCM of pruneBAP. Edges in the set $\{e \in \mathcal{E}_3 | w(e) \geq w(e_1), e \notin \mathcal{M}_0\}$ are removed from \mathcal{E}_3 in the first iteration. We denote this as a warm-start to pruneBAP. Fig. 4 illustrates a warm-start to pruneBAP.

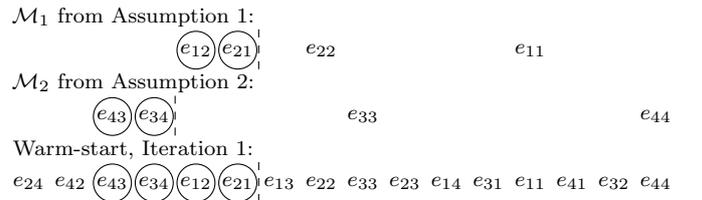


Fig. 4. A demonstration a warm-start. Here, $\mathcal{A}_1 = \{a_1, a_2\}$, $\mathcal{A}_2 = \{a_3, a_4\}$, $\mathcal{B}_1 = \{b_1, b_2\}$, and $\mathcal{B}_2 = \{b_3, b_4\}$. Then, \mathcal{A}_3 and \mathcal{B}_3 are the same as in Fig. 3. In this example, warm-starting pruneBAP allows the solution to $BOT(\mathcal{G}_3)$ to be found in 1 iteration.

Remark 1. Warm-starting is a heuristic, a warm-start does not guarantee fewer iterations for convergence to a solution to $BOT(\mathcal{G}_3)$. For a cold-start, we choose an arbitrary initial MCM \mathcal{M}_{cold} ; by chance this \mathcal{M}_{cold} could be the solution to $BOT(\mathcal{G}_3)$.

Given Assumptions 1 and 2, warm-starting is a way to make use of the available information to solve Problem 1.

6. CASE STUDIES

Consider agents and tasks represented by points in a vector space S with a distance function $D : S \times S \mapsto \mathbb{R}^+$.

For example, this could be a ride-sharing application, where agents are vehicles and their tasks are to pick up customers. Here we consider a 2-dimensional space $S = \mathbb{R}^2$ and Euclidean distance $D(x, y) = \|x - y\|_2$. Agents are to be assigned to move from their initial positions to target destinations based on the BAP with distance as weights.

6.1 Task Reassignment

Case Study 1. Let $\mathcal{A} = \{a_1, a_2, \dots, a_{m_3}\} \subset S$ be the initial locations of a set of agents. Let $\mathcal{B}_1 = \{b_1, b_2, \dots, b_{n_1}\} \subset S$ be the set of goal locations. Assume $m_3 > n_1$. We first solve $BOT((\mathcal{A} \cup \mathcal{B}_1, \mathcal{E}))$, where $\mathcal{E} = \{\{i, j\} | i \in \mathcal{A}, j \in \mathcal{B}_1\}$ to determine an assignment of tasks to agents that minimises the worst-case distance an agent must travel to reach a goal location. Without loss of generality, assume vehicles at positions $\mathcal{A}_1 = \{a_1, a_2, \dots, a_{n_1}\}$ are assigned to goals at \mathcal{B}_1 . Now assume that a second set of goal locations becomes available to agents. Let $\mathcal{B}_2 = \{\beta_1, \beta_2, \dots, \beta_{n_2}\} \subset S$ be the set of new goal locations. Assume that $m_3 \geq n_1 + n_2$. Let $\mathcal{A}_2 = \{a_{n_1+1}, a_{n_1+2}, \dots, a_{m_3}\}$ be the locations of the remaining unassigned agents. We now assign the new goals to the remaining agents, i.e., solve $BOT((\mathcal{A}_2 \cup \mathcal{B}_2, \mathcal{E}_2))$, where $\mathcal{E}_2 = \{\{i, j\} | i \in \mathcal{A}_2, j \in \mathcal{B}_2\}$.

By Theorem 1, the assignment obtained from solving $BOT((\mathcal{A} \cup \mathcal{B}_1, \mathcal{E}))$ and $BOT((\mathcal{A}_2 \cup \mathcal{B}_2, \mathcal{E}_2))$ in Case Study 1 is not necessarily the optimal solution to $BOT((\mathcal{A} \cup \mathcal{B}_1 \cup \mathcal{B}_2, \mathcal{E}_3))$, where $\mathcal{E}_3 = \{\{i, j\} | i \in \mathcal{A}, j \in \mathcal{B}_1 \cup \mathcal{B}_2\}$. Fig. 5 shows a numerical example of a case where the optimal assignment is of lower cost than the assignment used to warm-start pruneBAP. For this example, $m_3 = 40$, $n_1 = 20$ and $n_2 = 20$. The data was generated using continuous uniform distributions with range $[0, 100)$ for both coordinates x and y . Fig. 6 shows a plot of the average cost of the assignment used as warm-start to initialise pruneBAP and the average cost of the optimal assignment after pruneBAP has terminated. For all simulations, $m_3 = n_1 + n_2$. For each even value of m_3 , 100 simulations were generated. We observe that the cost of the assignment obtained from the subproblems is never greater than the cost of an optimal solution to $BOT(\mathcal{G}_3)$, in accordance with Theorem 1. In this case, the unstructured distribution of the locations results in all of the conditions in Theorem 2 being satisfied and we observe that $w(e_3) < \max\{w(e_1), w(e_2)\}$ as expected.

6.2 Clustering of Agents and Tasks

Case Study 2. Let $\mathcal{A}_1 = \{a_1, a_2, \dots, a_{m_1}\} \subset S$ and $\mathcal{A}_2 = \{a_1, a_2, \dots, a_{m_2}\} \subset S$ be the initial locations of two sets of agents. Let $\mathcal{B}_1 = \{b_1, b_2, \dots, b_{n_1}\} \subset S$ and $\mathcal{B}_2 = \{\beta_1, \beta_2, \dots, \beta_{n_2}\} \subset S$ be the sets of goal locations. Assume that $m_1 \geq n_1$ and $m_2 \geq n_2$, i.e., there are more agents than there are goals. Assume the set of locations \mathcal{A}_1 and \mathcal{B}_1 are separated geographically from \mathcal{A}_2 and \mathcal{B}_2 .

In Case Study 2, we illustrate an example where not all of the conditions i., ii., and iii. in Lemma 2 hold. Fig. 7 shows a numerical example where the initial assignment $\mathcal{M}_1 \cup \mathcal{M}_2$ used to warm-start pruneBAP is in fact the optimal assignment of $\mathcal{B}_1 \cup \mathcal{B}_2$ to $\mathcal{A}_1 \cup \mathcal{A}_2$. In this example, $m_1 = 20$, $m_2 = 20$, $n_1 = 20$ and $n_2 = 20$. The data in Fig. 7 was generated using independent normal distributions with a

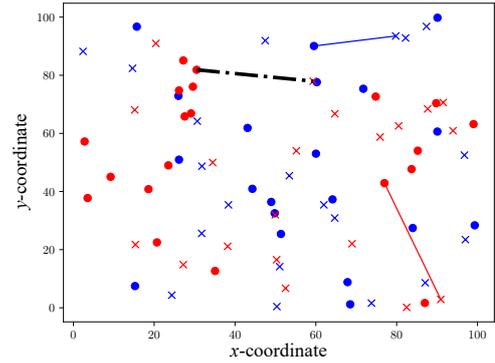


Fig. 5. Case Study 1: Sample configuration of agents and tasks. The locations in \mathcal{A}_1 , \mathcal{A}_2 , \mathcal{B}_1 , and \mathcal{B}_2 are represented by blue dots, red dots, blue crosses and red crosses respectively. The bottleneck edges from solving $BOT((\mathcal{A} \cup \mathcal{B}_1, \mathcal{E}))$, $BOT((\mathcal{A}_2 \cup \mathcal{B}_2, \mathcal{E}_2))$ and $BOT((\mathcal{A} \cup \mathcal{B}_1 \cup \mathcal{B}_2, \mathcal{E}_3))$ are shown by the solid blue, solid red and black dash-dotted lines respectively.

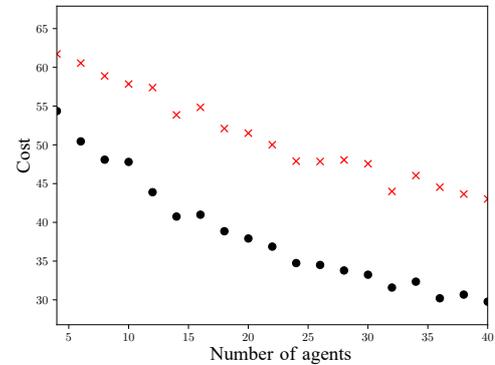


Fig. 6. Case Study 1: Cost of assignments. Red crosses show the average weight of the largest edge in the MCM used to warm-start pruneBAP, from 100 simulations. Black dots show the average weight of the bottleneck edge from solving $BOT((\mathcal{A} \cup \mathcal{B}_1 \cup \mathcal{B}_2, \mathcal{E}_3))$, from 100 simulations.

variance of 100 for each distribution. The distributions for sets \mathcal{A}_1 and \mathcal{B}_1 are centred at the point $(x, y) = (40, 60)$. The distributions for sets \mathcal{A}_2 and \mathcal{B}_2 are centred at the point $(x, y) = (60, 40)$. Fig. 8 shows the number of instances out of 100 simulations for which the behaviour in Fig. 7 is observed. That is, the instances where the bottleneck edges obtained from the subproblems directly results in an optimal solution to $BOT(\mathcal{G}_3)$, where \mathcal{G}_3 is defined as in Section 3. The number of agents equals the number of tasks for each simulation, i.e., $m_1 = n_1 = m_2 = n_2$. For each simulation, positions were generated using the same normal distribution as in Fig. 7. We now observe realisations where the cost of the assignment obtained from the subproblems is equal to the cost of an optimal solution to $BOT(\mathcal{G}_3)$. This illustrates that with this distribution of agents and tasks there are instances where there is structure such that the conditions in Theorem 2 do not all hold and $w(e_3) = \max\{w(e_1), w(e_2)\}$.

7. CONCLUSION

We discussed properties of pruneBAP that allow us to warm-start the algorithm given BAP solutions to divided

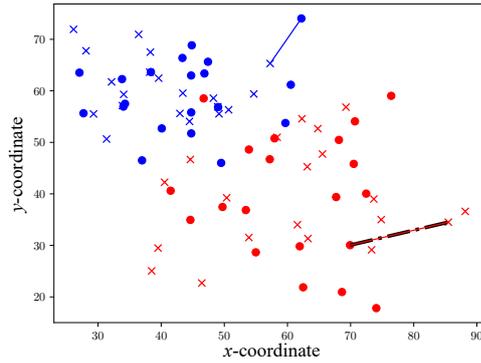


Fig. 7. Case Study 2: Sample configuration of agents and tasks. The positions of agents and tasks \mathcal{A}_1 , \mathcal{A}_2 , \mathcal{B}_1 , and \mathcal{B}_2 are represented by blue dots, red dots, blue crosses and red crosses respectively. The solid blue line shows the bottleneck edge from solving $BOT(\mathcal{G}_1)$. The solid red line shows the bottleneck edge from solving $BOT(\mathcal{G}_2)$. The black dash-dotted line shows the bottleneck edge from solving $BOT(\mathcal{G}_3)$.

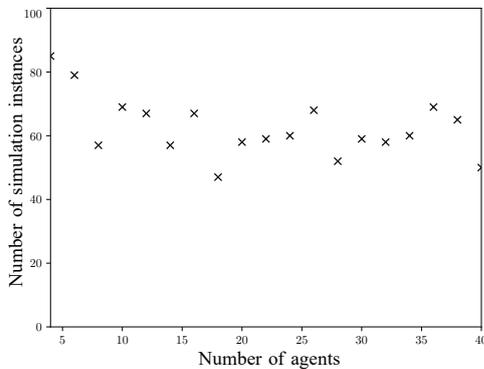


Fig. 8. Case Study 2: Empirical probability of solution to subproblems resulting in exact solution of the combined problem. Each cross shows the number of instances out of 100 simulations that the MCM $\mathcal{M}_3 = \mathcal{M}_1 \cup \mathcal{M}_2$ is an optimal solution to $BOT(\mathcal{G}_3)$. The number of agents represents $m_1 + m_2$.

sets of tasks and agents. The solutions based on the divided problems forms an MCM of the combined problem. The pruneBAP algorithm can be initialised with any MCM, and thus allows us to make use of the solutions based on the divided sets. We then have an upper bound on the BAP solution to the combined problem in terms of the bottleneck edges of the divided problems. We also introduced the novel concept of a bottleneck cluster relative to a bottleneck edge. This idea is inspired by the pruneBAP algorithm and the alternating tree that is obtained as a result of the algorithm. Using bottleneck clusters, we provided conditions such that the initial MCM used to warm-start pruneBAP is a solution to the BAP. From numerical simulations motivated by ride-sharing, we illustrate an example where the conditions hold if there exist clusters that are separated in space.

An interesting future direction is the investigation of methods to optimally partition agents and tasks. Another direction would be to investigate clustering properties for the LAP.

REFERENCES

- Aggarwal, V., Tikekar, V.A., and Hsu, L.F. (1986). Bottleneck assignment problems under categorization. *Computers and Operations Research*, 13, 11–26.
- Bertsekas, D.P. and Castañon, D.A. (1991). Parallel synchronous and asynchronous implementations of the auction algorithm. *Parallel Computing*, 17, 707–732.
- Burkard, R.E., Dell’Amico, M., and Martello, S. (2009). *Assignment problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia.
- Carrarsi, P. and Gallo, G. (1984). A multi-level bottleneck assignment approach to the bus drivers’ rostering problem. *European Journal of Operational Research*, 16, 163–173.
- Choi, H.L., Brenet, L., and How, J.P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25, 912–926.
- Chopra, S., Notarstefano, G., Rice, M., and Egerstedt, M. (2017). A distributed version of the hungarian method for multirobot assignment. *IEEE Transactions on Robotics*, 33, 932–947.
- Derigs, U. and Zimmermann, U. (1978). An augmenting path method for solving linear bottleneck assignment problems. *Computing*, 19, 285–295.
- Gabow, H.N. and Tarjan, R.E. (1988). Algorithms for two bottleneck optimization problems. *Journal of Algorithms*, 9, 411–417.
- Garfinkel, R.S. (1971). An improved algorithm for the bottleneck assignment problem. *Operations Research*, 19, 1747–1751.
- Gerkey, B.P. and Mataric, M.J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23, 939–954.
- Hopcroft, J.E. and Karp, R.M. (1973). An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2, 225–231.
- Khoo, M., Wood, T.A., Manzie, C., and Shames, I. (2019). Distributed algorithm for solving the bottleneck assignment problem. *IEEE 58th Conference on Decision and Control*, 1850–1855.
- Kuhn, H.W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2, 83–97.
- Pentico, D.W. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176, 774–793.
- Punnen, A.P. and Nair, K.P.K. (1994). Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem. *Discrete Applied Mathematics*, 55, 91–93.
- Shames, I., Dostovalova, A., Kim, J., and Hmam, H. (2017). Task allocation and motion control for threat-seeduction decoys. *IEEE 56th Annual Conference on Decision and Control*, 4509–4514.
- Zavlanos, M.M., Spesivtsev, L., and Pappas, G.J. (2008). A distributed auction algorithm for the assignment problem. *IEEE Conference on Decision and Control*, 1212–1217.