A Identical parallel machine scheduling problem (IPMS) with makespan minimization objective

A.1 Formulation

IPMS is a problem defined in continuous state/continuous time space. In MRRC, zero processing time of a task was assumed and only the travel times mattered. While there is no travel time concept in IPMS, IPMS has 'processing time' and 'setup time'. Once service of a task i begins, it requires a deterministic duration of time τ_i for a machine to complete - we call this the processing time. Machines are all identical, which means processing time of each tasks among machines are all the same. Processing times of each tasks are all different. Before a machine can start processing a task, it is required to first setup for the task. In this paper, we discuss IPMS with 'sequence-dependent setup times'. In this case, a machine must conduct a setup prior to serving each task. The duration of this setup depends on the current task i and the task j that was previously served on that machine - we call this the setup time. The completion time for each task is thus the sum of the setup time and processing time. Under this setting, we solve the IPMS problem for make-span minimization as discussed in Kurz et al. (2001). That is, we seek to minimize the total time spent from the start time to the completion of the last task. IPMS problem's sequential decision making problem formulation resembles that of MRRC with continuous-time and continuous-space. That is, every time there is a finished task, we make assignment decision for a free machine. We call this times as 'decision epochs' and express them as an ordered set $(t_1, t_2, \dots, t_k, \dots)$. Abusing this notation slightly, we use $(\cdot)_{t_k} = (\cdot)_k$. This problem can be cast as a Markov Decision Problem (MDP) whose state, action, and reward are defined as follows:

A.2 State

The state s_k at epoch k is represented as $(g_k, \mathcal{D}_k)$ where a graph $g_k = ((\mathcal{M}, \mathcal{T}_k), (\mathcal{E}_k^{\mathcal{T}\mathcal{T}}, \mathcal{E}_k^{\mathcal{M}\mathcal{T}}))$ and associated feature set $\mathcal{D}_k = (\mathcal{D}_k^{\mathcal{M}}, \mathcal{D}_k^{\mathcal{T}}, \mathcal{D}_k^{\mathcal{T}\mathcal{T}}, \mathcal{D}_k^{\mathcal{M}\mathcal{T}})$. The elements of graph g_k are defined as: (See Figure 1):

- $\mathcal{M} = \{1, ..., M\}$ is the index set of all machines. The index i and j will be used to specifically denote machines in the manuscript.
- $\mathcal{T}_k = \{1, ..., N\}$ is the index set of all remaining unserved tasks at decision epoch k. The index p and q will be used to specifically denote tasks in the manuscript.
- $\mathcal{E}_k^{\mathcal{T}\mathcal{T}} = \{\epsilon_{pq}^{\mathcal{T}\mathcal{T}} | p \in \mathcal{T}_k, q \in \mathcal{T}_k\}$ is the set of all directed edges from a task in $\mathcal{T}_k$ to any other task in $\mathcal{T}_k$. Abusing notation slightly, we consider each edge as a random variable. The task-to-task edge $\epsilon_{pq}^{\mathcal{T}\mathcal{T}} = 1$ indicates the event that a machine that has just completed task p subsequently completes task q. We call the probability $p(\epsilon_{pq}^{\mathcal{T}\mathcal{T}} = 1) \in [0, 1]$ the presence probability of the edge $\epsilon_{pq}^{\mathcal{T}\mathcal{T}}$.
- $\mathcal{E}_k^{\mathcal{M}\mathcal{T}} = \{\epsilon_{ip}^{\mathcal{M}\mathcal{T}} | i \in \mathcal{M}, p \in \mathcal{T}_k\}$ is the set of all directed edges from a machine in $\mathcal{R}$ to any other tasks in $\mathcal{T}_k$. Abusing notation similarly, we say the machine-to-task edge $\epsilon_{ip}^{\mathcal{M}\mathcal{T}} = 1$ indicates the event that robot i is assigned to the task p. This edge is defined deterministically depending on the joint assignment action. If machine i is assigned to task p, then $p(\epsilon_{ip}^{\mathcal{M}\mathcal{T}}) = 1$, otherwise 0.

The element of feature set $\mathcal{D}_k$ associated with the graph g_k is defined as:

- $\mathcal{D}_k^{\mathcal{M}} = \{d_i^{\mathcal{M}} | i \in \mathcal{M}\}$ is the set of node features for the robot nodes in $\mathcal{M}$ at epoch k. In IPMS, $d_i^{\mathcal{R}}$ is defined as the task processing status of robot i at epoch k (epoch index k is omitted).
- $\mathcal{D}_k^{\mathcal{T}} = \{d_p^{\mathcal{T}} | p \in \mathcal{T}_k\}$ is the set of node features for the task nodes in $\mathcal{T}_k$ at epoch k. In IPMS, $d_p^{\mathcal{T}}$ is not used.
- $\mathcal{D}_k^{\mathcal{T}\mathcal{T}} = \{d_{pq}^{\mathcal{T}\mathcal{T}} | p \in \mathcal{T}_k, q \in \mathcal{T}_k\}$ is the set of task-task edge features at epoch k. $d_{pq}^{\mathcal{T}\mathcal{T}}$ denotes the duration for a machine that has just completed task p to subsequently compete task q. We call this duration *task completion time.* In IPMS, a task completion time is sum of processing time and setup time.
- $\mathcal{D}_k^{\mathcal{M}\mathcal{T}} = \{d_{ip}^{\mathcal{M}\mathcal{T}} | i \in \mathcal{M}, p \in \mathcal{T}_k\}$ is the set of machine-task edge features at epoch k. $d_{ip}^{\mathcal{M}\mathcal{T}}$ denotes the task completion time for robot i to reach task p.

Table 4: IPMS test results for makespan minimization with deterministic task completion time (our algorithm / best Google OR tool result)

Makespan minimization		**# Machines**			
		3	5	7	10
# Tasks	50	106.7%	117.0%	119.8%	116.7%
	75	105.2%	109.6%	113.9%	111.3%
	100	100.7%	111.0%	109.1%	109.0%

A.3 Action

An action a_k, a joint assignment at epoch k, is defined as a maximal bipartite matching of the complete bipartite graph $(\mathcal{M}, \mathcal{T}_k, \mathcal{E}_k^{\mathcal{MT}})$ composed of the machine nodes $\mathcal{M}$, the remaining task nodes $\mathcal{T}_k$, and the fully connected edges between them $\mathcal{E}_k^{\mathcal{MT}}$. That is, given the current state $s_k = (g_k, \mathcal{D}_k)$, a_k is a subset of $\mathcal{E}_k^{\mathcal{MT}}$ satisfying (1) no two machines can be assigned to the same tasks, and (ii) a machine may only remain without assignment when the number of machines exceeds the number of remaining tasks. If $\epsilon_{ip}^{\mathcal{RT}} \in a_k$, it means that machine i is assigned with task p at epoch k. For example, Figure 1 shows the case where $a_k = (\epsilon_{1,1}^{\mathcal{RT}}, \epsilon_{2,3}^{\mathcal{RT}})$. (Note, we equivalently may say this as $\epsilon_{1,1}^{\mathcal{RT}} = \epsilon_{2,3}^{\mathcal{RT}} = 1$ and 0 otherwise.) In IPMS, only free machines are newly assigned.

A.4 State transition

When the joint assignment a_k is executed given the current state $s_k = (g_k, \mathcal{D}_k)$, the next state $s_{k+1} = (g_{k+1}, \mathcal{D}_{k+1})$ will be determined. The details differ depending on the problem.
Graph update when the decision epoch corresponds to the point when task p is completed, the corresponding task node will be removed in the updated task nodes as $\mathcal{T}_{k+1} = \mathcal{T}_k/\{p\}$, and the task-task edges and machine-task edges, $\mathcal{E}_{k+1}^{\mathcal{TT}}$ and $\mathcal{E}_{k+1}^{\mathcal{RT}}$, will be accordingly updated.
Feature update At decision epoch $k+1$, $\mathcal{D}_{k+1} = (\mathcal{D}_{k+1}^{\mathcal{R}}, \mathcal{D}_{k+1}^{\mathcal{T}}, \mathcal{D}_{k+1}^{\mathcal{TT}}, \mathcal{D}_{k+1}^{\mathcal{RT}})$ is determined. The machine-task edge features $\mathcal{D}_{k+1}^{\mathcal{MT}}$ will be updated according to $\mathcal{D}_{k+1}^{\mathcal{M}}$ as well. How these features are updated is determined by the problem specifications (environment).

A.5 Reward and objective

Define an assignment policy ϕ as a function that maps a state s_k to action a_k. Denote $T\left(s_k, a_k, s_{k+1}\right)$ as the time difference between epoch k and $k+1$ according to (s_k, a_k, s_{k+1}). Given s_0 initial state, an IPMS problem with makespan minimization objective can be expressed as a problem of finding an optimal assignment policy ϕ^* such that

$$\phi^* = \underset{\phi}{\operatorname{argmin}} \, \mathbb{E}_\phi \left[\sum_{k=0}^{\infty} T\left(s_k, a_k, s_{k+1}\right) | s_0 \right].$$

A.6 Experiments

For IPMS, we test it with continuous time, continuous state environment. While there have been many learning-based methods proposed for (single) robot scheduling problems, to the best our knowledge our method is the first learning method to claim scalable performance among machine-scheduling problems. Hence, in this case, we focus on showing comparable performance for large problems, instead of attempting to show the superiority of our method compared with heuristics specifically designed for IPMS (actually no heuristic was specifically designed to solve our exact problem (makespan minimization, sequence-dependent setup with no restriction on setup times))

For each task, processing times is determined using uniform [16, 64]. For every (task i, task j) ordered pair, a unique setup time is determined using uniform [0, 32]. As illustrated in Appendix A, we want to minimize make-span. As a benchmark for IPMS, we use Google OR-Tools library Google (2012). This library provides metaheuristics such as Greedy Descent, Guided Local Search, Simulated Annealing, Tabu Search. We compare our algorithm's result with the heuristic with the best result for each experiment. We consider cases with $3, 5, 7, 10$ machines and $50, 75, 100$ jobs.

The results are provided in Appendix Table 4. Makespan obtained by our method divided by the makespan obtained in the baseline is provided. Although our method has limitations in problems with a small number of tasks, it shows comparable performance to a large number of tasks and shows its value as the first learning-based machine scheduling method that achieves scalable performance.

B minimax multiple traveling salesman problem (minimax mTSP)

B.1 Formulation

Minimax multiple traveling salesman problem (minimax mTSP) is almost the same problem as IPMS problem defined in Appendix A. As in IPMS, we seek to minimize the total time spent from the start time to the completion of the last task. (Remark: minimax mTSP problem's original objective is to minimize the longest tour of all the salesmen. As the traveling speed of all the salesmen are identical, this objective is equivalent to makespan minimization objective of IPMS problem.) One difference in minimax mTSP from IPMS problem is that it's task completion time is just traveling time from a task to another task. Another difference is the existence of a 'depot' in minimax mTSP problem; every salesman start from the depot at time 0 and must return to the depot in the end.

IPMS problem's sequential decision making problem formulation is the same as that of IPMS. That is, every time there is a finished task, we make assignment decision for a free machine. We call this times as 'decision epochs' and express them as an ordered set $(t_1, t_2, \ldots, t_k, \ldots)$. Abusing this notation slightly, we use $(\cdot)_{t_k} = (\cdot)_k$. This problem can be cast as a Markov Decision Problem (MDP) whose state, action, and reward are almost exactly the same as that of IPMS except that we call a machines of IPMS a salesman.

B.2 Estimating state-action value function

In minimax mTSP, we don't delete the tasks that were previously served.

As in MRRC, two hierarchical layers of *random structure2vec* is used to infer the Q-function. Recall that in MRRC, the only input of the first random structure2vec was each task's robot assignment information. In minimax mTSP, we add three more information as input: each task's distance from the depot, each task's coordinate, and whether the task has been served by then.

The second structure2vec is the same as that of MRRC.

Given the output vectors of second structure2vec, we separately sum the vectors for the tasks that are not yet served and vectors for the tasks that are yet served. Given two separately summed output vectors, we concatenate the two resulting vectors and estimate the Q-function.

B.3 Experiments

Achieving multiple-TSP performance similar to the single-TSP result of famous Dai et al. (2017) (90%-92% optimal) qualifies itself a separate conference or journal paper. In this paper, we briefly introduce the reader the capability of our proposed method to solve minimax mTSP in a similar level of performance to Dai et al. (2017). In this experiment, we encoded relational information among nodes using TrXL-I (Parisoto et al., 2020) before computing presence probability. After encoding relational information among nodes, we used this information as node features for our model and computed presence probability with the edge features.

Dataset. We used the standard minmax mTSP dataset and state-of-art optimal solution baselines for them provided in minmaxTSPlib (2021). Each problem in the dataset is named after a city, where task locations in each city are originated from real world locations. For example, Berlin52 problem means that the task locations are originated from 52 real locations of the city of Berlin in Germany. minmaxTSPlib (2021) provides the state-of-art solution (found by integer linear programming model solved by CPLEX or reported by others) that minimizes the longest length tour.

Table 5 shows proposed algorithm's performance compared with the result provided in minmaxTSPlib (2021).

Table 5 compares the outcome of our method and the state-of-art solution provided by minmaxTSPlib (2021) and Google OR-ToolGoogle (2012). The state-of-art solution. Our proposed method's

Table 5: Test results on minmaxTSPlib (2021)

Problem name	# agents	CPLEX	OR-Tool	Ours
eli51	**M=2**	222.7	243.4	233.4
	M=3	159.6	170.1	171.9
	M=5	124.0	127.5	131.7
	M=7	112.1	112.1	114.8
berlin52	**M=2**	4110.2	4665.5	4313.9
	M=3	3244.4	3311.3	3243.5
	M=5	2441.4	2482.6	2638.5
	M=7	2440.9	2440.9	2474.1
eli76	**M=2**	280.9	318.0	298.8
	M=3	197.3	212.4	215.6
	M=5	150.3	143.4	158.4
	M=7	139.62	128.3	140.8
rat99	**M=2**	733.8	762.2	728.7
	M=3	592.6	552.1	587.2
	M=5	502.9	473.7	469.3
	M=7	473.1	442.5	443.9
	Average ratio	1	1.0180	1.0268

solution achieves in average 2.68% sub-optimality, which is the first to achieve a comparable result to Google OR-Tool (1.80%). We can see that the optimal solution has within 5% less cost than our proposed method's solution, which is much better than the sub-optimality of Dai et al. (2017) for single-traveling salesman problem.

C Proof of Theorem 1.

We first define necessary definitions for our proof. Given a random PGM $\{\mathcal{G}_{\mathcal{X}}, \mathcal{P}\}$, a PGM is chosen among $\mathcal{G}_{\mathcal{X}}$, the set of all possible PGMs on $\mathcal{X}$. The set of semi-cliques is denoted as $\mathfrak{C}_{\mathcal{X}}$. As discussed in the main text, if we are given $\mathcal{P}$ then we can easily calculate the presence probability p_m of semi-clique $\mathcal{D}_m$ as $p_m = \sum_{G \in \mathcal{G}_{\mathcal{X}}} \mathcal{P}(G) 1_{\mathcal{D}_m \in G}$.

For each semi-clique $\mathcal{D}^i$ in $\mathfrak{C}_{\mathcal{X}}$, define a binary random variable V^i: $\mathcal{F} \mapsto \{0, 1\}$ with value 0 for the factorization that does not include semi-clique $\mathcal{D}^i$ and value 1 for the factorization that include semi-clique $\mathcal{D}^i$. Let V be a random vector $V = \left(V^1, V^2, \ldots, V^{|\mathfrak{C}_{\mathcal{X}}|}\right)$. Then we can express $P(X_1, \ldots, X_n | V) \propto \prod_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|} \left[\phi^i\left(\mathcal{D}^i\right)\right]^{V^i}$. We denote $\left[\phi^i\left(\mathcal{D}^i\right)\right]^{V^i}$ as $\psi(\mathcal{D}^i)$.

Now we prove Theorem 1.

In mean-field inference, we want to find a distribution $Q\left(X_1, \ldots, X_n\right) = \prod_{i=1}^{n} Q_i(X_i)$ such that the cross-entropy between it and a target distribution is minimized. Following the notation in Koller & Friedman (2009), the mean field inference problem can written as the following optimization problem.

$$\begin{aligned} \min_{Q} \quad & \mathbb{D}\left(\prod_i Q_i \,|P\left(X_1, \ldots, X_n | V\right)\right) \\ \text{s.t.} \quad & \sum_{x_i} Q_i\left(x_i\right) = 1 \quad \forall i \end{aligned}$$

Here $\mathbb{D}\left(\prod_i Q_i \mid P\left(X_1, \ldots, X_n | V\right)\right)$ can be expressed as $\mathbb{D}\left(\prod_i Q_i \mid P\left(X_1, \ldots, X_n | V\right)\right) = \mathbb{E}_Q\left[\ln\left(\prod_i Q_i\right)\right] - \mathbb{E}_Q\left[\ln\left(P\left(X_1, \ldots, X_n | V\right)\right)\right]$.

Note that

$$
\begin{aligned}
\mathbb{E}_Q\left[\ln\left(P\left(X_1,\ldots,X_n|V\right)\right)\right] &= \mathbb{E}_Q\left[\ln\left(\frac{1}{z}\Pi_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|}\psi^i\left(\mathcal{D}^i,V\right)\right)\right] \\
&= \mathbb{E}_Q\left[\ln\left(\frac{1}{z}\prod_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|}\psi^i\left(\mathcal{D}^i,V\right)\right)\right] \\
&= \mathbb{E}_Q\left[\sum_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|}V^i\ln\left(\phi^i\left(\mathcal{D}^i\right)\right)\right] - \mathbb{E}_Q[\ln(Z)] \\
&= \sum_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|}\mathbb{E}_Q\left[V^i\ln\left(\phi^i\left(\mathcal{D}^i\right)\right)\right] - \mathbb{E}_Q[\ln(Z)] \\
&= \sum_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|}\mathbb{E}_{V^i}\left[\mathbb{E}_Q\left[V^i\ln\left(\phi^i\left(\mathcal{D}^i\right)\right)|V^i\right]\right] - \mathbb{E}_Q[\ln(Z)] \\
&= \sum_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|}P\left(V^i=1\right)\left[\mathbb{E}_Q\left[\ln\left(\phi^i\left(\mathcal{D}^i\right)\right)\right]\right] - \mathbb{E}_Q[\ln(Z)] \\
&= \sum_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|}p_i\left[\mathbb{E}_Q\left[\ln\left(\phi^i\left(\mathcal{D}^i\right)\right)\right]\right] - \mathbb{E}_Q[\ln(Z)].
\end{aligned}
$$

Hence, the above optimization problem can be written as

$$
\begin{aligned}
\max_{Q} \quad & \mathbb{E}_Q\left[\sum_{i=1}^{|\mathfrak{C}_{\mathcal{X}}|}p_i\ln\left(\phi^i\left(\mathcal{D}^i\right)\right)\right] + \mathbb{E}_Q\sum_{i=1}^{n}\left(\ln Q_i\right) \\
\text{s.t.} \quad & \sum_{x_i}Q_i\left(x_i\right)=1 \quad \forall i
\end{aligned}
\tag{A.1}
$$

In Koller & Friedman (2009), the fixed point equation is derived by solving an analogous equation to (A.1) without the presence of the p_i. Theorem 1 follows by proceeding as in Koller & Friedman (2009) with straightforward accounting for p_i.

D Hilbert space embedding of distributions.

We start from the motivation of Hilbert space embedding of distributions, with a particular focus on mean-field inference application. As discussed in section 3, mean-field inference methods try to search over the space of distributions, looking for the best surrogate distribution. While exact optimal solution search is a open, difficult optimization problem, at least we know that the optimal distribution must satisfy a fixed point equation we saw in section 3 Koller & Friedman (2009). While this is only a necessary condition and does not bring us an optimal solution, in practice distributions that satisfies such condition works as a nice approximate solution. Nevertheless, finding a distribution that satisfies the fixed equation of distribution involves an intractable equation of integrals.

A Hilbert space embedding of distributions transforms this kind of optimization problems over distributions into optimization problems over a vector space. Suppose that a random vector X is associated with a joint distribution F. Then for a function ϕ on the range of X, we can define a mapping towards a Hilbert space defined as $\mu_X := \mathbb{E}_X[\phi(X)] = \int \phi(x)dF(x)$. This kind of operation was first introduced in Smola et al. (2007). According to Sriperumbudur et al. (2008), there exist some ϕ that makes this operation an injective operation. Therefore, when we map the entire fixed point iteration on the distribution space to the Hilbert space, we don't lose any mathematical structure.

E Proof of Lemma 1.

Since we assume semi-cliques are only between two random variables, we can denote $\mathfrak{C}_{\mathcal{X}} = \{\mathcal{D}^{ij}\}$ and presence probabilities as $\{p_{ij}\}$ where i, j are node indexes. Denote the set of nodes as $\mathcal{V}$.

From here, we follow the approach of Dai et al. (2016) and assume that the joint distribution of random variables can be written as

$$p\left(\{H_k\}, \{X_k\}\right) \propto \prod_{k \in \mathcal{V}} \psi^i\left(H_k | X_k\right) \prod_{k,i \in \mathcal{V}} \psi^i\left(H_k | H_i\right).$$

Expanding the fixed-point equation for the mean field inference from Theorem 1, we obtain:

$$\begin{aligned}
&Q_k\left(h_k\right) = \\
&\frac{1}{Z_k} \exp\left\{\sum_{\psi^i : H_k \in \mathcal{D}^i} \mathbb{E}_{\left(\mathcal{D}^i - \{H_k\}\right) \sim Q}\left[\ln \psi^i\left(H_k = h_k | \mathcal{D}^i\right)\right]\right\} \\
&= \frac{1}{Z_k} \exp\{ln \phi\left(H_k = h_k | x_k\right) + \\
&\sum_{i \in \mathcal{V}} \int_{\mathcal{H}} p_{ki} Q_i\left(h_i\right) \ln \phi\left(H_k = h_k | H_i\right) dh_i\}.
\end{aligned}$$

This fixed-point equation for $Q_k\left(h_k\right)$ is a function of $\{Q_j\left(h_j\right)\}_{j \neq k}$ such that

$$Q_k\left(h_k\right) = f\left(h_k, x_k, \{p_{kj} Q_j\left(h_j\right)\}_{j \neq k}\right).$$

As in Dai et al. (2016), this equation can be expressed as a Hilbert space embedding of the form

$$\tilde{\mu}_k = \tilde{\mathcal{T}} \circ \left(x_k, \{p_{kj} \tilde{\mu}_j\}_{j \neq i}\right),$$

where $\tilde{\mu}_k$ indicates a vector that encodes $Q_k\left(h_k\right)$. In this paper, we use the nonlinear mapping $\tilde{\mathcal{T}}$ (based on a neural network form) suggested in Dai et al. (2016):

$$\tilde{\mu_k} = \sigma\left(W_1 x_k + W_2 \sum_{j \neq k} p_{kj} \tilde{\mu}_j\right)$$

F Presence probability inference method used for MRRC

In this experiment, we encoded relational information among nodes using TrXL-I style Multi-Head Self-Attention structure Parisotto et al. (2020) to compute presence probability. To compute a better presence probability (which is closer to the optimal solution), it is essential to consider relational information among nodes. Node features are stacked and then passed to Multi-Head Self-Attention, which encodes the relational information between nodes. After encoding relational information among nodes, we used these relational node features as the original node features for our GNN model and computed presence probability with these relational node features and the edge features. The rest of the part for estimating the Q-function is identical to the MRRC problem.

G Complete algorithm of section 5.2 with task completion time as a random variable

We combine random sampling and inference procedure suggested in section and Figure 2. Denote the set of task with a robot assigned to it as $\mathcal{T}^A$. Denote a task in $\mathcal{T}^A$ as t_i and the robot assigned to t_i as r_{t_i}. The corresponding edge in $\mathcal{E}^{RT}$ for this assignment is $\epsilon_{r_{t_i} t_i}$. The key idea is to use samples of $\epsilon_{r_{t_i} t_i}$ to generate N number of sampled $Q(s,a)$ value and average them to get the estimate of $E(Q(s,a))$. First, for $l = 1 \dots N$ we conduct the following procedure. For each task t_i in $\mathcal{T}^A$, we sample one data $e^l_{r_{t_i} t_i}$. Using those samples and $\{p_{ij}\}$, we follow the whole procedure illustrated in section D to get $Q(s,a)^l$. Second, we get the average of $\{Q(s,a)^l\}_{l=1}^{l=N}$ to get the estimate of $E(Q(s,a))$, $\frac{1}{N}\sum_{l=1}^{l=N} Q(s,a)^l$.

The complete algorithm of section D with task completion time as a random variable is given as below.

```
$age_i =$ age of node $i$
*The set of nodes for assigned tasks* $\equiv \mathcal{T}_A$
*Initialize* $\{\tilde{\mu}_i^{(0)}\}, \{\gamma_i^{(0)}\}$
 for $l = 1$ to $N$:
     for $t_i \in \mathcal{T}$:
         if $t_i \in \mathcal{T}^A$ do:
             sample $e^l_{r_{t_i} t_i}$ from $\epsilon_{r_{t_i} t_i}$
             $x_i = e^l_{r_{t_i} t_i}$
         else: $x_i = 0$
     for $t = 1$ to $T_1$ do
         for $i \in \mathcal{V}$ do
             $l_i = \sum_{j \in \mathcal{V}} p_{ji} \tilde{\mu}_j^{(t-1)}$
             $\tilde{\mu}_i^{(t)} = relu\left(W_3 l_i + W_4 x_i\right)$
     $\widetilde{\mu}_l =$ Concatenate $\left(\tilde{\mu}_i^{(T_1)}, age_i\right)$
     for $t = 1$ to $T_2$ do
         for $i \in \mathcal{V}$ do
             $l_i = \sum_{j \in \mathcal{V}} p_{ji} \gamma_j^{(t-1)}$
             $\gamma_j^{(t)} = relu\left(W_5 l_i + W_6 \tilde{\mu}_i\right)$
     $Q_l = W_7 \sum_{i \in \mathcal{V}} \gamma_i^{(T)}$
$Q_{avg} = \frac{1}{N} \sum_{l=1}^{N} Q_l$
```

H Proof of Lemma 2

Statement: *Denote result of OTAP using true Q-functions* $\{Q^{(n)}\}$ *as* $\mathcal{M}^{(N)} = \{m^{(1)} \dots m^{(N)}\}$*. If Q-function approximation method has order transferability, then* $\mathcal{M}^{(N)} = \mathcal{M}_\theta^{(N)}$ *holds.*
Proof. Recall that we say Q-function approximation method has order transferability if $\text{argmax}_{a_{t_k}} Q^n(s_{t_k}, a_{t_k}) = \text{argmax}_{a_{t_k}} Q_\theta^n(s_{t_k}, a_{t_k})$. We prove by induction.
Base case: For $n = 0$, $\mathcal{M}^{(0)} = \phi = \mathcal{M}_\theta^{(0)}$.
For $n > 0$, suppose that $\mathcal{M}^{(n)} = \mathcal{M}_\theta^{(n)}$ holds, i.e. $m^{(j)} = m_\theta^{(j)}$ for $1 \leq j \leq n$. Then according to $n + 1^{th}$ step OTEP operation,
$m^{(n+1)} = \text{argmax}_m Q^{n+1}\left(s_{t_k}, \mathcal{M}^{(n)} \cup \{m\}\right)$
$=\text{argmax}_m Q_\theta^{n+1}\left(s_{t_k}, \mathcal{M}^{(n)} \cup \{m\}\right)$ ($\because$ Order transferability assumption)
$= \text{argmax}_m Q_\theta^{n+1}\left(s_{t_k}, \mathcal{M}_\theta^{(n)} \cup \{m\}\right)$ ($\because$ induction argument)
$= m_\theta^{(n+1)}$.
Therefore, $\mathcal{M}^{(n+1)} = \mathcal{M}^{(n)} \cup \{m^{(n+1)}\} = \mathcal{M}_\theta^{(n)} \cup \{m_\theta^{(n+1)}\} = \mathcal{M}_\theta^{(n+1)}$.

I Statement and Proof of Lemma 3.

Denote the space of all possible policies as Π. For $\pi \in \Pi$, let the vector d_t^π denote the distribution of states at arbitrary time t assuming that we have been following policy π from time 0. We call a policy $\mu \in \Pi$ is exploratory with respect to Π if $\exists C < \infty$ such that $\forall \pi \in \Pi$, $\frac{d_t^\pi(s)}{\mu(s)} \leq C$ holds $\forall s \in \mathcal{S}$ and $\forall t \geq 0$. The assumption that such exploratory policy μ exists is called *Concentrability* assumption. Recall that in Auction-fitted Q-iteration (AFQI) we want to find θ that empirically minimizes $E_{(s_k, a_k, r_k, s_{k+1}) \sim D}\left[Q_\theta(s_k, a_k) - \left[r(s_k, a_k) + \gamma Q_\theta\left(s_{k+1}, \pi_{Q_\theta}(s_{k+1})\right)\right]\right]$ where D is a dataset. Denote the set of possible Q-functions as $\mathcal{F} \subset \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$. In our case, $\mathcal{F} = \{Q_\theta\}$. If we denote an operation $\mathcal{T}$ on the $\mathcal{F}$ such that $\mathcal{T} f(s, a) =: E_{(s,a,r,s') \sim D}\left[\left[r + \gamma f\left(s, \pi_f(s')\right)\right]\right]$, our problem can be restated as finding $f^* = \text{argmin}_{f \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}}(f - \mathcal{T} f)$. We say that this problem is *realizable* if $f^* \in \mathcal{F}$. We say that $\mathcal{F}$ is *closed under Bellman update* if $\forall f \in \mathcal{F}, \mathcal{T} f \in \mathcal{F}$. For details of above assumptions, see Agarwal et al. (2019).

We now formally state Lemma 3.

Lemma 3 Kang & Kumar (2021). Under the assumption that concentrability, realizability and closure under Bellman update holds, the policy we achieve by AFQI is assured to have performance at least $1 - 1/e$ compared with the optimal policy.

J Decentralized algorithm

In this section, we show that we can modify the auction procedure in OTAP at each timestep as a special case of Han-Lim Choi et al. (2009)'s sequential greedy algorithm for solving MRTA problem. This enables us to conclude, without further discussion, that 1) assignment consensus is guaranteed among robots even under frequent communication packet loss and 2) centralized algorithm's performance bound is inherited. The Decentralized algorithm is almost the same as the centralized version. Bolded sentences indicate what is different in decentralized version of suggested algorithm.

Initial message choice phase. In the n^{th} bidding phase, initially *all robots know* $\mathcal{M}_\theta^{(n-1)}$, the ordered set of $n-1$ robot-task edges in $\mathcal{E}_{t_k}^{RT}$ determined by the previous $n-1$ iterations. An unassigned robot i ignores all others unassigned and calculates $Q_\theta^n(s_{t_k}, \mathcal{M}_\theta^{(n-1)} \cup \{\epsilon_{ip}^{RT}\})$ for each unassigned task p as if those k robots (robot i together with all robots assigned tasks in the previous $n-1$ iterations) only exist in the future and will serve all remaining tasks. (Here, $\epsilon_{ip}^{RT} \in \mathcal{E}_{t_k}^{RT}$ is the edge corresponding to assigning robot i to task p at decision epoch t_k.) If task ℓ has the highest value, robot i chooses $\{\epsilon_{i\ell}^{RT}, Q_\theta^n(s_t, \mathcal{M}_\theta^{(n-1)} \cup \{\epsilon_{i\ell}^{RT}\})\}$ as the initial message to be sent to others. (Note that, since the number of ignored robots varies at each iteration, transferability of Q-function inference is crucial)

Consensus phase. In the n^{th} consensus phase, robot i keeps sending message to neighbouring robots within its one-hop communication range. At first, agent i keeps sending its initial message chosen in above phase to neighbors. Then every time a robot i receives a message $\{\epsilon_{jm}^{RT}, Q_\theta^n(s_t, \mathcal{M}_\theta^{(n-1)} \cup \{\epsilon_{jm}^{RT}\})\}$ of robot j, it compares $Q_\theta^n(s_t, \mathcal{M}_\theta^{(n-1)} \cup \{\epsilon_{jm}^{RT}\})$ with $Q_\theta^n(s_t, \mathcal{M}_\theta^{(n-1)} \cup \{\epsilon_{i\ell}^{RT}\})$. If the former is larger, robot i sets $\{\epsilon_{jm}^{RT}, Q_\theta^n(s_t, \mathcal{M}_\theta^{(n-1)} \cup \{\epsilon_{jm}^{RT}\})\}$ as the new message to be sent to others. Agent i keeps sending its new message to neighbors until it hears all robot's initial messages. Denote the message of i right after it hears all robot's initial message as $m_\theta^{(n)}$ (Note that the initial message of i includes robot i's assignment information). Then agent i updates the assignment set as $\mathcal{M}_\theta^{(n)} = \mathcal{M}_\theta^{(n-1)} \cup m_\theta^{(n)}$.

K Proof of Theorem 2

Statement: *Denote* $N = \max(|\mathcal{R}|, |T_t|)$.

Suppose that Q-function approximation method has order transferability. Denote $\mathcal{M}_\theta^{(N)}$ *as the result of OTAP using* $\{Q_\theta^n\}$ *and* $\mathcal{M}^*$ *as* $\operatorname{argmax}_{a_{t_k}} Q(s_{t_k}, a_{t_k})$. *If 1) the marginal value of adding one robot is positive, i.e.* $Q^{|\mathcal{M}|+1}(s_{t_k}, \mathcal{M} \cup \{m\}) - Q^{|\mathcal{M}|}(s_{t_k}, \mathcal{M}) \geq 0$ *for all* $\mathcal{M} \subset \mathcal{E}_t^{RT}$ *and 2) the marginal value of adding one robot diminishes as the robot number increases, i.e.,* $Q^{|\mathcal{M}|+1}(s_{t_k}, \mathcal{M} \cup \{m\}) - Q^{|\mathcal{M}|}(s_{t_k}, \mathcal{M}) \leq Q^{|\mathcal{N}|+1}(s_{t_k}, \mathcal{N} \cup \{m\}) - Q^{|\mathcal{N}|}(s_{t_k}, \mathcal{N})$ *for* $\mathcal{N} \subset \mathcal{M} \subset \mathcal{E}_t^{RT}$, *for all* $m \in \mathcal{E}_t^{RT}$, *then the result of OTAP is at least better than* $1 - 1/e$ *of optimal assignment, i.e.,* $Q_\theta^N(s_{t_k}, \mathcal{M}_\theta^{(N)}) \geq Q^{|\mathcal{M}^*|}(s_{t_k}, \mathcal{M}^*)(1 - 1/e)$.

Proof. From the assumption 1) that the marginal value of adding one robot is nonnegative, without loss of generality, we can consider $\mathcal{M}^*$ with $|\mathcal{M}^*| = N$ in the further proof procedure. Denote $\mathcal{M}^* = \{m^{(1)*}, m^{(2)*}, \ldots, m^{(n)*}\}$ and denote $\mathcal{M}_\theta^{(N)} = \{m_\theta^{(1)}, m_\theta^{(2)}, \ldots, m_\theta^{(N)}\}$.
For notation simplicity, define $\Delta(m \mid \mathcal{M}) =: Q^{|\mathcal{M} \cup \{m\}|}(s_t, \mathcal{M} \cup \{m\}) - Q^{|\mathcal{M}|}(s_t, \mathcal{M})$.

Then the optimal value $OPT = Q^N(s_{t_k}, \mathcal{M}^*) \leq Q^{|\mathcal{M}_\theta^{(n)} \cup \mathcal{M}^*|}(s_{t_k}, \mathcal{M}_\theta^{(n)} \cup \mathcal{M}^*)$
$= Q^n(s_{t_k}, \mathcal{M}_\theta^{(n)}) + \sum_{j=1}^N \Delta(m^{(j)*} \mid \mathcal{M}_\theta^{(n)} \cup \{m^{(1)*}, \cdots, m^{(j-1)*}\})$
$\leq Q^n(s_{t_k}, \mathcal{M}_\theta^{(n)}) + \sum_{j=1}^N \Delta(m^{(j)*} \mid \mathcal{M}_\theta^{(n)})$($\because$ condition 2 - decreasing marginal value condition)
$\leq Q^n(s_{t_k}, \mathcal{M}_\theta^{(n)}) + \sum_{j=1}^N \Delta(m_\theta^{(n+1)} \mid \mathcal{M}_\theta^{(n)})$
($\because$ OTAP chooses $m_\theta^{(n+1)} = \operatorname{argmax}_m Q_\theta^{n+1}\left(s_t, \mathcal{M}_\theta^{(n)} \cup \{m\}\right)$ and
$\operatorname{argmax}_m Q_\theta^{n+1}\left(s_t, \mathcal{M}_\theta^{(n)} \cup \{m\}\right) = \operatorname{argmax}_m Q^n\left(s_t, \mathcal{M}_\theta^{(n)} \cup \{m\}\right)$ from *Lemma 2*)
$= Q^n(s_{t_k}, \mathcal{M}_\theta^{(n)}) + N\Delta(m_\theta^{(n+1)} \mid \mathcal{M}_\theta^{(n)})$.
Therefore, $\Delta(m_\theta^{(n+1)} \mid \mathcal{M}_\theta^{((n))}) \geq \frac{1}{N}(OPT - Q^n(s_{t_k}, \mathcal{M}_\theta^{(n)})$.
Note that $OPT - Q^n(s_{t_k}, \mathcal{M}_\theta^{(n)})$ denotes current iteration $(= n^{th})$ outcome $\mathcal{M}_\theta^{(n)}$'s size of sub-optimality compared to OPT. Denote $OPT - Q^n(s_{t_k}, \mathcal{M}_\theta^{(n)}) =: \beta_n$. Then since $Q^0(s_{t_k}, \phi) = 0$, $\beta_0 = OPT$. Therefore, we have $\Delta(m_\theta^{(n+1)} \mid \mathcal{M}_\theta^{((n))}) \geq \frac{1}{N}\beta_n$.
Also, note that $\Delta(m_\theta^{(n+1)} \mid \mathcal{M}_\theta^{(n)}) = Q^{n+1}(s_t, \mathcal{M}_\theta^{(n)} \cup \{m_\theta^{(n+1)}\}) - Q^n(s_t, \mathcal{M}_\theta^{(n)})$
$= Q^{n+1}(s_t, \mathcal{M}_\theta^{(n+1)}) - Q^n(s_t, \mathcal{M}_\theta^{(n)}) = (OPT - Q^n(s_t, \mathcal{M}_\theta^{(n)}) - (OPT - Q^{n+1}(s_t, \mathcal{M}_\theta^{(n+1)}))$
$= \beta_n - \beta_{n+1}$.
Therefore, $\beta_n - \beta_{n+1} \geq \frac{1}{N}\beta_n$, i.e., $\beta_{n+1} \leq \beta_n\left(1 - \frac{1}{N}\right)$.
This implies $OPT - Q^N(s_{t_k}, \mathcal{M}_\theta^{(N)}) = \beta_N \leq \beta_0(1 - \frac{1}{N})^N = OPT(1 - \frac{1}{N})^N$ and thus we get $Q^N(s_{t_k}, \mathcal{M}_\theta^{(N)}) = OPT(1 - (1 - \frac{1}{N})^N) \sim OPT(1 - \frac{1}{e})$ as $N \to \infty$.

L Scalability analysis

Computational complexity. MRRC can be formulated as a semi-MDP (SMDP) based multi-robot planning problem (e.g., Omidshafiei et al. (2017)). This problem's complexity with R robots and T tasks and maximum H time horizon is $O((R!/T!(R-T)!)^H)$. For example, Omidshafiei et al. (2017) state that a problem with only 13 task completion times ('TMA nodes' in their language) possessed a policy space with cardinality $5.622 * 10^{17}$. In our proposed method, this complexity is addressed by a combination of two complexities: computational complexity and training complexity. For computational complexity of joint assignment decision at each timestep, it is $O(|R||T|^3) = O((1) \times (2) \times (3) \times (4) + (5))$ where $(1) - (5)$ are as follows.

(1) # of Q-function computation required in one time-step = $O(|R||T|)$: Shown in section 4.2

(2) # of mean-field inference in one Q-function computation = 2 (constant): Two embedding steps (Distance embedding, Value embedding) each needs one mean-field inference procedure

(3) # of structure2vec propagation operation in one mean-field inference= $O(|T|^2)$: There is one structure2vec operation from a task to another task and therefore the total number of operations is $|T| \times (|T|-1)$.

(4) # of neural net computation for each structure2vec propagation operation=C (constant): This is only dependent on the hyperparameter size of neural network and does not increase as number of robots or tasks.

(5) # of neural net computation for inference of random PGM=$O(|T|^2)$ As an offline stage, we infer the semi-clique presence probability for every possible directed edge, i.e. from a task to another task using algorithm introduced in Appendix F. This algorithm complexity is $O(|T| \times (|T|-1)) = O(|T|^2)$.