Embedding a random graph via GNN: Extended mean-field inference theory and RL applications to NP-Hard multi-robot/machine scheduling

Hyunwook Kang[† 1] **Aydar Mynbay**[2] **James R. Morrison**[* 3] **Jinkyoo Park**[* 3]

Abstract

We illustrate that developing a theory of 'how to embed a random graph using GNN' is the key to achieving the first near-optimal learning-based scheduling algorithm for an NP-hard multi-robot scheduling problem for tasks with time-varying rewards. We focus on a problem referred to as a Multi-Robot Reward Collection (MRRC) problem, of which immediate applications are ridesharing and pickup-and-delivery problems. We 1) observe that states in our robot scheduling problems can be represented as an extension of probabilistic graphical models (PGMs), which we refer to as random PGMs, and 2) develop a mean-field inference method for random PGMs. We then prove that a simple heuristic for applying deep graph encoder for random graph embedding is theoretically justified. We illustrate how a two-step hierarchical inference induces precise Q-function estimation. We empirically demonstrate that our method achieves near-optimality (plus transferability and scalability, machine scheduling (IPMS) applications in the appendix section). Arxiv preprint: https://arxiv.org/abs/1905.12204.

1. Introduction

We consider a set of robots and seek to serve a set of spatially distributed tasks. A reward is given for serving each task promptly, resulting in a decaying reward collection problem. We focus on 'scheduling' problems possessing constraints such as 'no possibility of two robots assigned to a task at once'. Such problems prevail in various operation research problems, e.g., dispatching multiple vehicles to deliver customers in a city or scheduling multiple machines in a factory. Impossibility results in asynchronous communication[1] [Fischer et al. (1985)] make these problems inherently centralized. This study is the first to explore the possibility of near-optimally solving such larger scale multi-robot, multi-task NP-hard scheduling problems with time-dependent rewards using a learning-based algorithm.

Proposed methods and contributions. We first extend the probabilistic graphical model (PGM)-based mean-field inference theory in (Dai et al., 2016) for random PGM. Then we show that a naively-looking two-step heuristic of "First, approximate each edge's presence probability" and "Then, apply a typical deep graph encoder with probabilistic adjustments" is justified from the subsequent theoretical results. We call structure2vec ((Dai et al., 2016) combined with this heuristic as 'random structure2vec'. We then empirically demonstrate the effectiveness of this method. After observing that each state of robot scheduling problem can be represented as a 'random' PGM, we use random structure2vec to design a transferable reinforcement learning method that is first to learn a near-optimal NP-hard multi-robot/machine scheduling (97% optimal for a multi-robot scheduling problem called multi-robot reward collection (MRRC) with deterministic environment with linearly-varying rewards where true optimal baselines are available).

2. Multi-robot scheduling problem formulation

We formulate a multi-robot reward collection problem (MRRC) as a discrete-time, discrete-state sequential decision-making problem. We assume that we are given "time required for an assigned robot to complete a task" which we call "task completion time". A task completion time may be deterministic, or a random variable[2]. These durations are time-invariant. At each time-step, we assign robots to remaining tasks. We cast MRRC problem as a sequential decision-making problem as follows.

[1]First author. Department of Electrical & Computer Engineering, Texas A&M University, College Station, Texas, USA [2]PUBG, Seoul, South Korea [3]Dept. of Industrial & Systems Engineering, KAIST, Daejeon, South Korea. Correspondence to: James R. Morrison <james.morrison@kaist.edu>, Jinkyoo Park <jinkyoo.park@kaist.ac.kr>.

Graph Representation Learning and Beyond (GRL+) workshop of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

[1]Due to this limitation, multi-agent (decentralized) methods are rarely used in industries (e.g., factories)

[2]Our method requires only samples of the random variable; its distribution is not required.

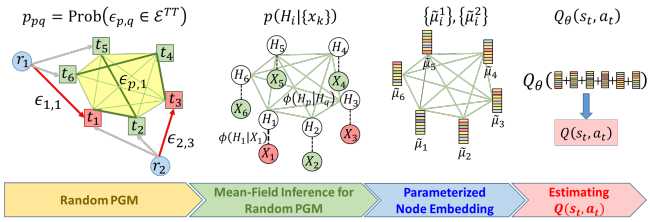

Figure 1. Representing a State into a random PGM

State. The state s_t at time step t is represented as $(\mathcal{G}_t, \alpha_t)$. $\mathcal{G}_t$ is a directed graph $(\mathcal{R}_t \cup \mathcal{T}_t, \mathcal{E}_t)$ where $\mathcal{R}_t$ is the set of all robots at time step t, $\mathcal{T}_t$ is the set of all remaining unserved tasks at time step t, and $\mathcal{E}_t = \mathcal{E}_t^{RT} \cup \mathcal{E}_t^{TT}$ is the set of directed edges. A directed edge $\epsilon_{i,p}^{RT} \in \mathcal{E}_t^{RT}$ is assigned with a random variable denoting the task completion time for robot i to finish task p A directed edge $\epsilon_{p,q}^{TT} \in \mathcal{E}_t^{TT}$ is assigned with a weight (random variable) denoting the task completion time (a duration) for a robot that just competed task p to complete task q. $\alpha_t = \{\eta_t^p \in \mathbb{R} | p \in \mathcal{T}_t\}$ is a set of ages where η_t^p indicates the age of task p at time-step t. We denote the set of possible states as $\mathcal{S}$. See Figure 1.

Joint assignment. A joint assignment of robots to tasks at current state $s_t = (\mathcal{G}_t, \alpha_t)$, denoted as a_t, should satisfy two constraints: (i) no two robots can be assigned to the same task, and (ii) a robot may only remain without assignment when the number of robots exceeds the number of remaining tasks. Thus, a joint assignment a_t is a maximal bipartite matching in the bipartite graph $(\mathcal{R}_t \cup \mathcal{T}_t, \mathcal{E}_t^{RT})$. A policy π is defined as $\pi : \mathcal{S} \mapsto \mathcal{A}$.

Transition function and reward. In the hierarchical control literature, our 'assignment' is termed a 'macro-action'. In pursuit of the macro-action, robots may make multiple sequential micro-actions to service task. (We allow reassignment prior to arrival.) The transition probability associated with 'macro-action' is derived from transition probability associated with 'micro-action' [Omidshafiei et al. (2017)]. In MRRC, each task has an arbitrarily determined initial age. At each time-step, the age of each task increases by one. When a task is served, a reward is given according to a predetermined reward function that computes reward according to the task's age at the time of service. Note that the state and assignment information s_t, a_t and s_{t+1} are thus sufficient to determine the reward at time step $t+1$. As such we denote the reward function as $R(s_t, a_t, s_{t+1}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$.

Objective. Given an initial state $s_0 \in \mathcal{S}$, the MRRC seeks to maximize the sum of expected rewards through time by the selection of an assignment policy π^* satisfying $\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_{\pi,P'} \left[\sum_{t=0}^{\infty} R\left(s_t, \pi(s_t), s_{t+1}\right) | s_0\right]$.

MRRC is a general class of problems with numerous applications. For example, by considering 'movement from location A to location B' as a task to which a robot or vehicle is assigned, ride-sharing problems or package delivery problems are naturally modeled as an MRRC problem.

3. Scheduling by Inferencing Random PGM

3.1. Backgrounds on probabilistic graphical model (PGM)

PGM. Given random variables $\{X_k\}$, suppose that the joint distribution factors as $P(X_1, \ldots, X_n) = \frac{1}{Z} \prod_i \phi_i(\mathcal{D}_i)$ where $\phi_i(\mathcal{D}_i)$ denotes a marginal distribution or conditional distribution with a set of random variables $\mathcal{D}_i$; Z is a normalizing constant. Then $\{X_k\}$ is called a probabilistic graphical model (PGM). In a PGM, $\mathcal{D}_i$ is called a clique and $\phi_i(\mathcal{D}_i)$ is called a clique potential for $\mathcal{D}_i$. When we write simply ϕ_i, suppressing $\mathcal{D}_i$, $\mathcal{D}_i$ is called the scope of ϕ_i.

Mean-field inference with PGM. A popular use of PGM is PGM-based mean-field inference. In mean-field inference, we find a surrogate distribution $q(X_1, \ldots, X_n) = \prod_i q_i(x_i)$ with smallest Kullback-Leibler distance to the original joint distribution $P(X_1, \ldots, X_n)$. This surrogate distribution is used to conduct the inference. Koller & Friedman (2009) shows that when we are given a PGM, the $q_i(x_i)$ can be obtained by a fixed point equation. Despite the usefulness of this approach, we are not often directly given the probabilistic graphical model.

***Structure2vec*.** Dai et al. (2016) suggests that for graph-like structured data the edges of PGM (i.e. probabilistic relationships) can be assumed heuristically, and this info may be enough for an inference method called *structure2vec*. There are two types of random variables: $\{Y_k\}$ serving as input to the inference problem (e.g., atomic number of atom k in Dai et al. (2016)) and $\{H_k\}$ serving as latent random variables where H_k corresponds to Y_k. Y_k and H_k are connected with an edge. The edges among $\{H_k\}$ are heuristically assumed from the graph structure of data (e.g. molecular bindings in Dai et al. (2016)). In PGM, the joint probability distribution on random variables $(\{H_k\}, \{Y_k\})$ can be factored as $P(\{H_k\}, \{Y_k\}) \propto \prod_{k \in \mathcal{V}} \phi(H_k, Y_k) \prod_{k,i \in \mathcal{V}} \phi(H_k, H_i)$, where $\mathcal{V}$ denotes the set of vertex indexes and ϕ is a clique potential. Dai et al. (2016) has proposed *structure2vec* to embed the posterior marginal $p(\{h_k\}|\{y_k\})$ of a feature mapping $\phi(H_i)$ of a hidden variable H_i as $\mu_i = \int_{\mathcal{H}} \phi(H_i) p(h_i|y_i) dh_i$, to generate the parameterized fixed point equation for $\tilde{\mu}_k$ as $\tilde{\mu}_k = \sigma(W_1 y_k + W_2 \sum_{j \neq k} \tilde{\mu}_j)$. Here σ denotes Relu function and W denotes parameters of neural networks (see Dai et al. (2016)).

3.2. Inference with random PGM

Random PGM. Denote the set of all random variables in the inference problem as $\mathcal{X} = \{X_i\}$. Suppose that the set of possible PGMs on $\mathcal{X}$, denoted as $\mathcal{G}_{\mathcal{X}}$, is prior knowledge

(in many robot scheduling problems, this is the set of all possible Bayesian networks - see Appendix 3). A random PGM on $\mathcal{X}$ is then defined as $\{\mathcal{G}_{\mathcal{X}}, \mathcal{P}\}$ where $\mathcal{P} : \mathcal{G}_{\mathcal{X}} \mapsto [0, 1]$ is the probability of realizing an element of $\mathcal{G}_{\mathcal{X}}$. Note that the inference of $\mathcal{P}$ will be difficult. To avoid this task, we start by defining 'semi-cliques'. Suppose that we are given the set of all possible cliques on $\mathcal{X}$ as $\mathfrak{C}_{\mathcal{X}}$. Only a few probabilistic relationships in $\mathfrak{C}_{\mathcal{X}}$ are actually realized as an element of PGM and become real cliques. We call the elements $\mathcal{D}_m \in \mathfrak{C}_{\mathcal{X}}$ as 'semi-cliques'. Note that if we are given $\mathcal{P}$ then we can easily calculate the presence probability p_m of semi-clique $\mathcal{D}_m$ as $p_m = \sum_{G \in \mathcal{G}_{\mathcal{X}}} \mathcal{P}(G) 1_{\mathcal{D}_m \in G}$.

Mean-field inference with random PGM. The following theorem extends mean-field inference with PGM Koller & Friedman (2009) to mean-field inference with random PGM. It shows that we only need infer the presence probability of each semi-clique in the random PGM.

Theorem 1. Random PGM based mean field inference. Suppose we are given a random PGM on $\mathcal{X} = \{X_k\}$. Also, assume that we know presence probability $\{p_m\}$ for all semi-cliques $\mathcal{D}_m \in \mathfrak{C}_{\mathcal{X}}$. The latent variable distribution $\{q_k(x_k)\}$ in mean-field inference is locally optimal only if $q_k(x_k) = \frac{1}{Z_k} \exp\left\{\sum_{m: X_k \in \mathcal{D}_m} p_m \mathbb{E}_{(\mathcal{D}_m - \{X_k\}) \sim q} \left[\ln \phi_m(\mathcal{D}_m, x_k)\right]\right\}$ where Z_k is a normalizing constant and ϕ_m is the clique potential for clique m. (For the proof see Appendix 4.)

Random *structure2vec*. From *Theorem 1*, we can develop a *random structure2vec* corresponding to a random PGM with $(\{H_k\}, \{Y_k\})$. That is, we can combine (1) the fixed point equation of the mean field approximation for $q_k(h_k)$ (*Theorem 1*) and (2) the injective embedding for $\mu_i = \int_{\mathcal{H}} \phi(H_i) p(h_i|y_i) dh_i$ to come up with parameterized fixed point equation for $\tilde{\mu}_k$ (see Figure 1). As in Dai et al. (2016), we restrict our discussion to the case where there are semi-cliques between two random variables. In this case, the notation we use for $\mathcal{D}_m$ and p_m is $\mathcal{D}_{ij}$ and p_{ij}.

Lemma 1. Structure2vec for random PGM. Given a random PGM on $\mathcal{X} = (\{H_k\}, \{Y_k\})$. Assume that we know the presence probabilities $\{p_{ij}\}$ for all pairwise semi-cliques $\mathcal{D}_{ij} \in \mathfrak{C}_{\mathcal{X}}$. The fixed point equation in Theorem 1 for posterior marginal $P(\{H_k\}|\{y_k\})$ can be embedded in a nonlinear function mapping to generate the fixed point equation $\tilde{\mu}_k = \sigma\left(W_1 y_k + W_2 \sum_{j \neq k} p_{kj} \tilde{\mu}_j\right)$.

The proof of *Lemma 1* can be found in Appendix 5. The implication of *Lemma 1* is *only a simple inference task is required: inferring the presence probability of each semi-clique*. See Appendix 6 for the algorithm. We formalize this observation as a separate corollary.

Corollary 1. For MRRC, the random PGM representation for Lemma 1 is $((\mathcal{T}_t, \mathcal{E}_t^{TT}), \{p_{ij}\})$ where $\{p_{ij}\}$ denotes the probability of a robot choosing task i after serving task j.

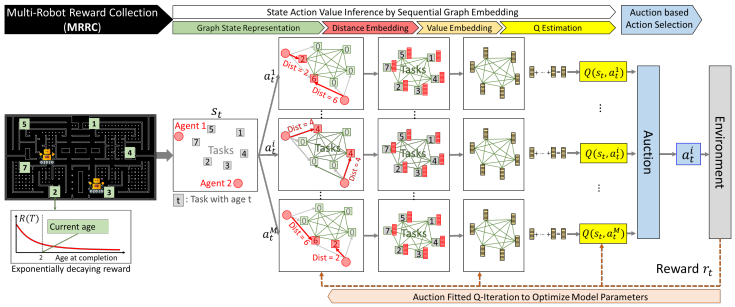

Figure 2. Illustration of overall pipeline of our method

4. Scheduling MRRC with *Random structure2vec*

We discuss how *random structure2vec* in Section 3.2 can be used to estimate $Q(s_t, a_t)$. For assignment choice and exploration method that enables scalability, see Appendix.

4.1. Inferring $Q(s_t, a_t)$

First, we discuss the two-step sequential and hierarchical state-embedding strategy using *random structure2vec*.For brevity, we illustrate the inference procedure where task completion time is deterministic. (For random task completion times, see Appendix 7.)

Step 1. Distance Embedding. Intuitively the output vectors $\{\tilde{\mu}_k^1\}$ of *structure2vec embed local graph information around that vector node* Dai et al. (2016). We here focus on embedding information on robot locations around a task node and thus infer each task's 'relative graphical distance' from the robots around it. For the input of the first use of *structure2vec* ($\{x_k\}$ in Lemma 1), we only use robot assignment information (if k is an assigned task, we set the value of x_k to 'task completion time of assignment' (a duration); if k is not an assigned task:, we set $x_k = 0$). This procedure is illustrated in Figure 2.

Step 2. Value Embedding. The second step is designed to infer 'How much value is likely in the local graph around each task'. Recall that the output vectors of the first step $\{\tilde{\mu}_k^1\}$ carry information about the relative graphical distance from all robots to each task. For each task k, we concatenate the age η_t^k the corresponding vector $\tilde{\mu}_k^1$ and use the resulting graph as the input ($\{x_k\}$ in Lemma 1) to a second *structure2vec*. See Figure 2. The resulting output vectors $\{\tilde{\mu}_k^2\}$ provide sufficient information about 'How much value is likely in the local graph around each task'.

Step 3. Computing $Q(s_t, a_t)$. To infer $Q(s_t, a_t)$, we aggregate the embedding vectors for all nodes as $\tilde{\mu}^2 = \sum_k \tilde{\mu}_k^2$ to obtain one global vector $\tilde{\mu}^2$ to embed the 'value affinity' of the global graph. We then use a neural network to map $\tilde{\mu}^2$ into $Q(s_t, a_t)$. The intuition behind transferability and detailed algorithm of the above procedure (with random task

Table 1. Performance test (50 trials of training for each cases)

Rew.	Env.	Base.	Testing size : Robot (R) / Task (T)						
			2R/20T	3R/20T	3R/30T	5R/30T	5R/40T	8R/40T	8R/50T
Linear	Determ.	%Opt	98.31 (4.23)	97.50 (4.71)	97.80 (5.14)	95.35 (5.28)	96.99 (5.42)	96.11 (4.56)	96.85 (3.40)
		%Ekisi	99.86	97.50	118.33	110.42	105.14	104.63	120.16
		%SGA	137.3	120.6	129.7	110.4	123.0	119.9	119.8
	Stocha.	%SGA	130.9	115.7	122.8	115.6	122.3	113.3	115.9
Exp.	Determ.	%SGA	111.5	118.1	118.0	110.9	118.7	111.2	112.6
	Stocha.	%SGA	110.8	117.4	119.7	111.9	120.0	110.4	112.4

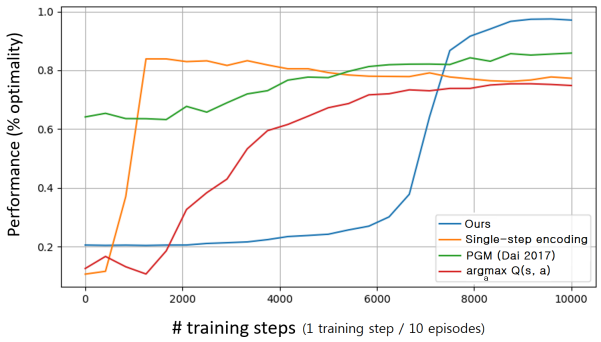

Figure 3. Ablation study, with 0) Ours, 1) Ours without sequntial embedding, 2) Ours without random PGM, 3) Ours without TAP

completion times) is provided in Appendix 7 and 9.

The key idea is that this over/under-estimation does not matter in Q-learning van Hasselt et al. (2015) as long as the *order* of Q-function value among actions are the same.

5. Experiment

Target MRRC problem. We test our algorithm using the modified maze (see Figure 2) generator of Neller et al. (2010) (code provided in Appendix 10) and compare with the baselines. (For the tests overcoming inherent artificialness, inscalability, discreteness of the simulator, see appendix 2) To generate the task completion times of a maze, Dijkstra's algorithm and dynamic programming were used for deterministic and stochastic environments, respectively. In the deterministic environment, robots always succeed in their movement. In the stochastic environment, a robot makes its intended move with a certain probability. (Cells with a dot: success with 55%, every other direction with 15% each. Cells without a dot: 70% and 10%, respectively.) We consider two reward rules: linearly decaying rewards $f(age) = \max\{200 - age, 0\}$ and nonlinearly decaying rewards $f(age) = \lambda^{age}$ with $\lambda = 0.99$, where age is the task age when served. Initial age of tasks were uniformly distributed in the interval $[0, 100]$.

Baselines. We consider three baselines:
• *Optimum*: We have this baseline only when the environment is deterministic and the reward is linear. The exact solution of it was computed using Gurobi Optimization (2019) with a 60-min time limit.
• *Ekici et al*: For deterministic environments and linear rewards, an up-to-date, fast heuristic for MRRC is available Ekici & Retharekar (2013). It claims around 93% optimality for 50 tasks and 4 robots.
• *Sequential Greedy Algorithm (SGA)*: To our knowledge, there is no literature addressing MRRC with stochastic environments or exponential rewards. Instead, we construct an indirect baseline using a general-purpose multi-robot task allocation algorithm called SGA Han-Lim Choi et al. (2009). We will provide our performance divided by SGA performance as %SGA. We will see that the %SGA in the deterministic linear-reward case is maintained for other cases.

The performance is expressed as % of ratio ρ = (rewards collected by the proposed method/reward collected by the baseline).

Performance test. We tested the performance under four environments: deterministic/linear rewards, deterministic/nonlinear rewards, stochastic/linear rewards, stochastic/nonlinear rewards. See Table 1. For linear/deterministic rewards, our method achieves near-optimality with 3% fewer rewards than *optimal* on average. The standard deviation for ρ is provided in parentheses. For others, we see that the %SGA ratio for linear/deterministic is well maintained in stochastic or nonlinear environments.

Ablation study. There are three components in our proposed method: 1) Q-function inference with random PGM, 2) a careful encoding of information using two-layers of *random structure2vec*, and 3) transferability-enabled policy (TAP) (see appendix). We have removed each component from the full model and compared with full model. We only considered deterministic/linearly decaying rewards where the optimal solution is available for comparison. Figure 3 shows the results. While the full method requires more training steps, only the full method achieves near-optimum.

For the transferability and scalability tests, see appendix.

6. Concluding Remarks

We proposed the first near-optimal learning-based algorithm for solving NP-hard multi-robot/machine scheduling problems (MRRC here and IPMS in the appendix). As discussed in Section 2, ride-sharing and pickup delivery problems can naturally be formulated as MRRC problems. We created the concept of random PGM to represent scheduling problems with random task completion times. We developed mean-field inference theory for random PGM. We prove that the popular GNN-based mean-field inference method can be extended to achieve the mean-field inference method we call *random structure2vec*. This method enabled a precise Q-function estimation for multi-robot scheduling problems.

References

Dai, H., Dai, B., and Song, L. Discriminative Embeddings of Latent Variable Models for Structured Data. 48:1–23, 2016. doi: 1603.05629.

Ekici, A. and Retharekar, A. Multiple agents maximum collection problem with time dependent rewards. *Computers and Industrial Engineering*, 64(4):1009–1018, 2013. ISSN 03608352. doi: 10.1016/j.cie.2013.01.010. URL http://dx.doi.org/10.1016/j.cie.2013.01.010.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, apr 1985. ISSN 0004-5411. doi: 10.1145/3149.214121. URL http://dl.acm.org/doi/10.1145/3149.214121.

Gurobi Optimization, L. Gurobi optimizer reference manual, 2019. URL http://www.gurobi.com.

Han-Lim Choi, Brunet, L., and How, J. Consensus-Based Decentralized Auctions for Robust Task Allocation. *IEEE Transactions on Robotics*, 25(4):912–926, aug 2009. ISSN 1552-3098. doi: 10.1109/TRO.2009.2022423.

Koller, D. and Friedman, N. *Probabilistic graphical models : principles and techniques, page 449-453*. The MIT Press, 1st edition, 2009. ISBN 9780262013192.

Neller, T., DeNero, J., Klein, D., Koenig, S., Yeoh, W., Zheng, X., Daniel, K., Nash, A., Dodds, Z., Carenini, G., Poole, D., and Brooks, C. Model AI Assignments. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pp. 1919–1921, 2010.

Omidshafiei, S., Agha–Mohammadi, A., Amato, C., Liu, S., How, J. P., and Vian, J. Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *The International Journal of Robotics Research*, 36(2):231–258, 2017. doi: 10.1177/0278364917692864.

van Hasselt, H., Guez, A., and Silver, D. Deep Reinforcement Learning with Double Q-learning. 2015.

Appendix of - "Embedding a random graph via GNN: Extended mean-field inference theory and RL applications to NP-Hard multi-robot/machine scheduling"

Hyunwook Kang [1] **Aydar Mynbay** [2] **James R. Morrison** [3] **Jinkyoo Park** [3]

1. MRRC with continuous state/continuous time space formulation, or with setup time and processing time

In continuous state/continuous time space formulation, the initial location and ending location of robots and tasks are arbitrary on $\mathbb{R}^2$. At every moment at least a robot finishes a task, we make assignment decision for a free robot(s). We call this moments as 'decision epochs' and express them as an ordered set $(t_1, t_2, \dots, t_k, \dots)$. Abusing this notation slightly, we use $(\cdot)_{t_k} = (\cdot)_k$.

Task completion time can consist of three components: travel time, setup time and processing time. While a robot in the travel phase or setup phase may be reassigned to other tasks, we can't reassign a robot in the processing phase. Under these assumptions, at each decision epoch robot r_i is given a set of tasks it can assign itself: if it is in the traveling phase or setup phase, it can be assigned to any tasks or not assigned; if it is in the processing phase, it must be reassigned to its unfinished task. This problem can be cast as a Markov Decision Problem (MDP) whose state, action, and reward are defined as follows:

State. The state s_k at decision epoch k is represented as $(\mathcal{G}_k, \alpha_k)$. $\mathcal{G}_k$ is a directed graph $(\mathcal{R}_k \cup \mathcal{T}_k, \mathcal{E}_k)$ where $\mathcal{R}_k$ is the set of all robots at decision epoch k, $\mathcal{T}_k$ is the set of all remaining unserved tasks at decision epoch k, and $\mathcal{E}_k = \mathcal{E}_k^{RT} \cup \mathcal{E}_k^{TT}$ is the set of directed edges. A directed edge $\epsilon_{i,p}^{RT} \in \mathcal{E}_k^{RT}$ is assigned with a weight (random variable) denoting the task completion time (a duration) for robot i to finish task p (note this time subsumes the information about the robot's present location). A directed edge $\epsilon_{p,q}^{TT} \in \mathcal{E}_k^{TT}$ is assigned with a weight (random variable) denoting the task completion time (a duration) for a robot that just competed task p to complete task q. $\alpha_k = \{\eta_k^p \in \mathbb{R} | p \in \mathcal{T}_k\}$ is a set of ages where η_k^p indicates the age of task p at decision epoch k. We denote the set of possible states as $\mathcal{S}$.

Action. A joint assignment of robots to tasks at current state $s_k = (\mathcal{G}_k, \alpha_k)$, denoted as a_k, should satisfy two constraints: (i) no two robots can be assigned to the same task, and (ii) a robot may only remain without assignment when the number of robots exceeds the number of remaining tasks. Thus, a joint assignment a_k is a maximal bipartite matching in the bipartite graph $(\mathcal{R}_k \cup \mathcal{T}_k, \mathcal{E}_k^{RT})$. For example, robot i in $\mathcal{R}_k$ is matched with task p in $\mathcal{T}_k$ if we assign robot i to task p at decision epoch k. We denote the set of all possible joint assignments as $\mathcal{A}$. A policy π is defined as $\pi : \mathcal{S} \mapsto \mathcal{A}$.

Reward. In MRRC, Each task has an arbitrarily determined initial age. At each decision epoch, the age of each task increases by the time it passed during the epoch. When a task is serviced, a reward is determined only by its age when serviced. Denote this reward rule as $R(k)$. One can easily see that whether a task is served at epoch k is completely determined by s_k, a_k and s_{k+1}. Therefore, we can denote the reward we get with s_k, a_k and s_{k+1} as $R(s_k, a_k, s_{k+1})$.

Objective. We can now define an assignment policy ϕ as a function that maps a state s_k to action a_k. Given s_0 initial state, an MRRC problem can be expressed as a problem of finding an optimal assignment policy ϕ^* such that

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

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

2.1. Formulation

IPMS is a problem defined in continuous state/continuous time space. 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,

Table 1. 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%

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:

State. Defined the same as MRRC with continuous state/time space, except that the robot is now task, and task completion time is the sum of (setup time + processing time).

Action. Defined the same as MRRC with continuous state/time space.

Reward. Let's denote the time between decision epoch k and decision epoch $k+1$ as $T_k = t_k - t_{k-1}$. One can easily see that T_k is completely determined by s_k, a_k and s_{k+1}. Therefore, we can denote the reward we get with s_k, a_k and s_{k+1} as $T(s_k, a_k, s_{k+1})$.

Objective. We can now define an assignment policy ϕ as a function that maps a state s_k to action a_k. 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

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

2.2. 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 2.1, 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 1. 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.

3. Bayesian Network representation

Here we analytically show that robot scheduling problem randomly induces a random Bayesian Network from state s_t. Given starting state s_t and action a_t, a person can repeat a random experiment of "sequential decision making using policy ϕ". In this random experiment, we can define events 'How robots serve all remaining tasks in which sequence'. We call such an event a 'scenario'. For example, suppose that at time-step t we are given robots $\{A, B\}$, tasks $\{1, 2, 3, 4, 5\}$, and policy ϕ. One possible scenario S^* can be {robot A serves task $3 \to 1 \to 2$ and robot B serves task $5 \to 4$}. Define random variable X_k a task characteristic, e.g. 'The time when task k is serviced'. The question is, 'Given a scenario S^*, what is the relationship among random variables $\{X_k\}$'? Recall that in our sequential decision making formulation we are given all the 'task completion time' information in the s_t description. Note that, task completion time is only dependent on the previous task and assigned task. In our example above, under scenario S^* 'when task 2 is served' is only dependent on 'when task 1 is served'. That is, $P(X_2|X_1, X_3, S^*) = P(X_2|X_1, S^*)$. This relationship is called 'conditional independence'. Given a scenario S^*, every relationship among $\{X_i|S^*\}$ can be expressed us-

ing this kind of relationship among random variables. A graph with this special relationship is called 'Bayesian Network' [Koller & Friedman (2009)], a probabilistic graphical model.

4. 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 $Q(X_1, \ldots, X_n) = \prod_{i=1}^n Q_i(X_i)$ 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

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

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

5. 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 infer-

ence from Theorem 1, we obtain:

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

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):

$$\tilde{\mu_k} = \sigma\left(W_1x_k + W_2\sum_{j\neq k}p_{kj}\tilde{\mu}_j\right)$$

6. Simple presence probability inference method used for MRRC

Denote ages of task i, j as age_i, age_j. Note that if we generate M samples of ϵ_{ij} as $\{e_{ij}^k\}_{k=1}^M$, then $\frac{1}{M}\sum_{k=1}^M f(e_{ij}^k, age_i, age_j)$ is an unbiased and consistent estimator of $E[f(\epsilon_{ij}, age_i, age_j)]$. The corresponding neural network-based inference is as follows: for each sample k, for each task i and task j, we form a vector of u_{ij}^k = (e_{ij}^k, age_i, age_j) and compute g_{ij} = $\sum_{k=1}^M\frac{1}{M}W_1(relu(W_2u_{ij}^k)$. We obtain $\{p_{ij}\}$ from $\{g_{ij}\}$ using softmax.

The pseducode implementation is as follows: In lines 1 and 2, the likelihood of the existence of a directed edge from each node m to node n is computed by calculating $W_1\left(relu\left(W_2u_{mn}^k\right)\right)$ and averaging over the M samples. In lines 3 and 4, we use the soft-max function to obtain $p_{m,n}$.

1 For $m, n\in\mathcal{V}$ do
2 $\quad g_{mn} = \frac{1}{M}\sum_{k=1}^M W_1\left(relu\left(W_2u_{mn}^k\right)\right)$
3 For $m, n\in\mathcal{V}$ do
4 $\quad p_{m,n} = \frac{e^{g_{mn}/\tau}}{\sum_{j\in v}e^{g_{mn}/\tau}}$.

Table 2. Transferability test (50 trials of training for each cases, linear & deterministic env.)

	Testing size : Robot (R) / Task (T)						
Training size (Robot(R)/Task(T))	2R/20T	3R/20T	3R/30T	5R/30T	5R/40T	8R/40T	8R/50T
2R/20T	98.31	93.61	97.31	92.16	92.83	90.94	93.44
3R/20T	95.98	97.50	96.11	93.64	91.75	91.60	92.77
3R/30T	94.16	96.17	97.80	94.79	93.19	93.14	93.28
5R/30T	97.83	94.89	96.43	95.35	93.28	92.63	92.40
5R/40T	97.39	94.69	95.22	93.15	96.99	94.96	93.65
8R/40T	95.44	94.43	93.48	93.93	96.41	96.11	95.24
8R/50T	95.69	96.68	97.35	94.02	94.50	94.86	96.85

7. Complete algorithm of section 4.1 with task completion time as a random variable

We combine random sampling and inference procedure suggested in section 4.1 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 $l = 1\ldots N$ we conduct the following procedure. For each task t_i in $\mathcal{T}^A$, we sample one data $e_{r_{t_i}t_i}^l$. Using those samples and $\{p_{ij}\}$, we follow the whole procedure illustrated in section 4.1 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 4.1 with task completion time as a random variable is given as below.

1 age_i = age of node i
2 *The set of nodes for assigned tasks* $\equiv\mathcal{T}_A$
3 *Initialize* $\{\mu_i^{(0)}\}, \{\gamma_i^{(0)}\}$
4 for $l = 1$ to N:
5 $\quad$ for $t_i\in\mathcal{T}$:
5 $\quad\quad$ if $t_i\in\mathcal{T}^{\mathcal{A}}$ do:
6 $\quad\quad\quad$ sample $e_{r_{t_i}t_i}^l$ from $\epsilon_{r_{t_i}t_i}$
7 $\quad\quad\quad$ $x_i = e_{r_{t_i}t_i}^l$
9 $\quad\quad$ else: $x_i = 0$
10 $\quad$ for $t = 1$ to T_1 do
11 $\quad\quad$ for $i\in\mathcal{V}$ do
12 $\quad\quad\quad$ $l_i = \sum_{j\in\mathcal{V}}p_{ji}\mu_j^{(t-1)}$
13 $\quad\quad\quad$ $\mu_i^{(t)} = relu\left(W_3l_i + W_4x_i\right)$
14 $\quad$ $\tilde{\mu}_l =$ Concatenate $\left(\mu_i^{(T_1)}, age_i\right)$
15 $\quad$ for $t = 1$ to T_2 do
16 $\quad\quad$ for $i\in\mathcal{V}$ do
17 $\quad\quad\quad$ $l_i = \sum_{j\in\mathcal{V}}p_{ji}\gamma_j^{(t-1)}$
18 $\quad\quad\quad$ $\gamma_j^{(t)} = relu\left(W_5l_i + W_6\tilde{\mu}_i\right)$
19 $\quad$ $Q_l = W_7\sum_{i\in\mathcal{V}}\gamma_i^{(T)}$
20 $Q_{avg} = \frac{1}{N}\sum_{l=1}^N Q_l$

Table 3. Training complexity (mean of 20 trials of training, linear & deterministic env.)

Linear & Deterministic	**Testing size : Robot (R) / Task (T)**						
	2R/20T	3R/20T	3R/30T	5R/30T	5R/40T	8R/40T	8R/50T
Performance with full training	98.31	97.50	97.80	95.35	96.99	96.11	96.85
# Training for 93 optimality	19261.2	61034.0	99032.7	48675.3	48217.5	45360.0	47244.2

8. Transferability

8.1. Intuition behind transferability of inference steps in section 4.1

Why are the inference steps introduced in section 4.1 transferable? For the first step, it is trivial; the inference problem is a scale-free task. In the second step, the 'value affinity' embedding will be underestimated or overestimated according to the ratio (number of robots/number of tasks) in a local graph.: underestimated if the ratio in the training environment is smaller than the ratio in the testing environment; overestimated otherwise.

8.2. Experiments to check transferability

Suppose that we trained an algorithm with problems of 3 robots/30 tasks. We can claim transferability of this algorithm if it achieves similar performance compared to the algorithm trained with problems of 8 robots/50 tasks when both are tested for problems with 8 robots/50 tasks. Table 2 shows comprehensive transferability test results. The rows indicate training conditions, while the columns indicate testing conditions. The results in the diagonal cells in red (cells with the same training size and testing size) serve as baselines (direct testing). The results in the off-diagonal show the results for the transferability testing, and demonstrate how the algorithms trained with different problem size perform well on test problems. We can see that lower-direction transfer tests (trained with larger size problem and tested with smaller size problems) show only a small loss in performance. For upper-direction transfer tests (trained with smaller size problems and tested with larger size problem), the performance loss was up 4 percent.

9. Methods for scalability

9.1. Transferability-enabled assignment choice

In *Bidding phase.*, every robot r_i in $\mathcal{R}_t$ evaluate $Q_t(s_t, a_t)$ over all the feasible action $\epsilon_{iq} \in \mathcal{E}_t^{RT}$ while assuming there are no unsigned robots and proposes (bids) the best action ϵ_{iq^*} selecting t_q^* in $\mathcal{T}_t$ with its corresponding $Q_t(s_t, \epsilon_{iq^*})$ values. In *Consensus phase.*, the auctioneer finds the bid with the best value, and select one pair of robot r_i^* and task t_q^*, say $\epsilon_{i^*q^*}$. Then, the algorithm update the current state as $\mathcal{G}_t^{'} = (\mathcal{R}_t/r_{i^*} \cup \mathcal{T}_t/t_{q^*}, \mathcal{E}_t^{RT}/\epsilon_{i^*q^*})$ and proceeds these steps again from the bidding phase.

Bidding phase. In the kth bidding phase, initially *all robots know* the $k-1$ robot-task matchings determined in the previous $k-1$ iterations. Denote this matching as $\mathcal{M}_{k-1}$, a bipartite subgraph of $(\mathcal{R}_t \cup \mathcal{T}_t, \mathcal{E}_t^{RT})$. Unassigned robot i, ignores all others unassigned, and calculates $Q(s(t), \mathcal{M}_{k-1} \cup \{\epsilon_{ip}^{RT}\})$ for each unassigned task p as if those k robots only exist in the future and serve all remaining tasks (here, $\epsilon_{ip}^{RT} \in \mathcal{E}_t^{RT}$ is the edge corresponding to assigning robot i to task j). If task ℓ has the highest value, robot i bids $\{\epsilon_{i\ell}^{RT}, Q(s_t, \mathcal{M}_{k-1} \cup \{\epsilon_{i\ell}^{RT}\})\}$ to the centralized auctioneer. Since the number of ignored robots varies at each iteration, transferability of Q-function inference is crucial.

Consensus phase. At kth consensus phase, the centralized auctioneer finds the bid with the best bid value, say $\{\epsilon_{i^*p^*}^{RT}$, bid value with $\epsilon_{i^*p^*}^{RT}\}$. (Here i^* and p^* denote the best robot task pair.) The centralized auctioneer then updates everyone's $\mathcal{M}_{k-1}$ as $\mathcal{M}_k = \mathcal{M}_{k-1} \cup \{\epsilon_{i^*p^*}^{RT}\}$.

These two phases iterate until $\mathcal{M}_k$ becomes a maximal bipartite matching of $(\mathcal{R}_t \cup \mathcal{T}_t, \mathcal{E}_t^{RT})$. This matching is then chosen as the joint assignment a_t^* at time step t. One can easily verify that the computational complexity of computing π_{Q_θ} is $\mathrm{O}(|L_R|\,|L_T|)$, which is only polynomial. In the experiments, we show empirically that TAP achieves near-optimal performance for MRRC.

9.2. Auction-fitted Q-iteration framework

Auction-fitted Q-iteration.
We incorporate TAP into a fitted Q-iteration framework to find θ that empirically minimizes $E_{\pi_{Q_\theta}, s_{k+1} \sim P'}\left[Q_\theta(s_k, a_k) - [r(s_k, a_k) + \gamma Q_\theta(s_{k+1}, \pi_{Q_\theta}(s_{k+1}))]\right]$. We call it auction-fitted Q-iteration.

Bidding phase. In a bidding phase, all robots initially know the matching determined in previous iterations (the partial solution for the given scheduling problem). We denote this matching as $\mathcal{Y}$, a bipartite subgraph of $((\mathcal{R}_t \cup \mathcal{T}_t), \mathcal{E}_t^{RT})$. When making a bid, a robot r_i ignores all other unassigned robots. For example, suppose robot r_i considers t_j for bidding. For r_i, $\mathcal{Y} \cup \epsilon_{ij}$ is a proper action (according to definition in section 2) for the scheduling problem that excludes currently unassigned robots. That is, robot r_i thus can compute $Q(s_t, \mathcal{Y} \cup \epsilon_{r_i t_j})$ for all unassigned task t_j while ignoring other unassigned robots. If task t^* is with

the highest value, robot r_i bids $\{\epsilon_{r_i t^*}, Q(s_t, \mathcal{Y} \cup \epsilon_{r_i t^*})\}$ to auctioneer. Since number of robots ignored by r_i is different at each iteration, transferability of Q-function inference plays key role.

Consensus phase. In the consensus phase, the auctioneer finds the bid with the best value, say $\{\epsilon^*$, bid value with $\epsilon^*\}$. Then auctioneer updates everyone's $\mathcal{Y}$ as $\mathcal{Y} \cup \{\epsilon^*\}$.

Exploration. How can we conduct exploration in the auction-fitted Q-iteration framework? Unfortunately, we can't use ϵ-greedy method since 1) an arbitrary random deviation in a joint assignment often induces a catastrophic failure Maffioli (1986), 2) joint assignment space, complex and combinatorial, is hard to explore efficiently with such arbitrary random exploration policy. In learning the parameters θ for $Q_\theta(s_k, a_k)$, we use the exploration strategy that perturbs the parameters θ randomly to actively explore the joint assignment space with TAP. Note that θ denotes all neural network parameters used in the *random structure2vec* iterations and the read-out function mapping the node embedding $\tilde{\mu}_k$ to $Q(s_t, a_t)$. While this method was originally developed for policy-gradient based methods Plappert et al. (2017), exploration in parameter space is also particularly useful in our auction-fitted Q-iteration as well.

9.3. Computational complexity analysis

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 6. This algorithm complexity is $O(|T| \times (|T| - 1)) = O(|T|^2)$.

While we don't explicitly compute the training complexity of our method, we show below that empirical evidence (Table 2) shows that training requirement does not necessarily increases and thus complexity is at most sub-polynomial.

9.4. Experiments testing training complexity.

Training efficiency is required to obtain scalability. To quantify this we measured the training time required to achieve 93% optimality. As before, we consider a deterministic environment with linear rewards and compare with the exact optimum. See Table 3. There, training time does not necessarily increase with problem size, suggesting the proposed algorithm is scalable.

10. Code for the experiment

For the entire codes used for experiments, please go to the following Google drive link for the codes.

References

Dai, H., Dai, B., and Song, L. Discriminative Embeddings of Latent Variable Models for Structured Data. 48:1–23, 2016. doi: 1603.05629.

Google. Google OR-Tools, 2012. URL https://developers.google.com/optimization/.

Koller, D. and Friedman, N. *Probabilistic graphical models : principles and techniques, page 449-453*. The MIT Press, 1st edition, 2009. ISBN 9780262013192.

Kurz, M. E., Askin, R. G., Kurzy, M. E., and Askiny, R. G. Heuristic scheduling of parallel machines with sequence-dependent set-up times. *International Journal of Production Research*, 39(16):3747–3769, 2001. ISSN 0020-7543. doi: 10.1080/00207540110064938.

Maffioli, F. Randomized algorithms in combinatorial optimization: A survey. *Discrete Applied Mathematics*, 14(2):157 – 170, 1986. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(86)90058-2. URL http://www.sciencedirect.com/science/article/pii/0166218X86900582.

Omidshafiei, S., Agha–Mohammadi, A., Amato, C., Liu, S., How, J. P., and Vian, J. Decentralized control of multi-robot partially observable Markov decision processes us-

ing belief space macro-actions. *The International Journal of Robotics Research*, 36(2):231–258, 2017. doi: 10.1177/0278364917692864.

Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R. Y., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M. Parameter Space Noise for Exploration. pp. 1–18, 2017.