Graph Neural Networks: Learning Representations of Robot Team Coordination Problems

Matthew Gombolay and Zheyuan Wang

International Conference on Autonomous Agents and Multi-Agent Systems 2022

May 8, 2022
Outline

Multi-Robot Coordination Problems

• Introduction to Graph Neural Networks

• Solving Multi-Robot Coordination with Graph Neural Networks

• Ongoing and Future Work
Resource Optimization

- Diverse application domains:
  - Airplane manufacturing
  - Patient appointment scheduling in healthcare
  - Logistics in e-commerce
  - Airline and crew scheduling

- Coordination in multi-robot systems
  - Team communication
  - Task planning
    - Our focus: task allocation and scheduling
    - Motion planning

- Plane maintenance scheduling
  - Prevent / reduce the adverse effect of aircraft failures
  - Maximize aircraft availability
  - Picking the right time for repair / maintenance

- Coordinate a finite number of resources to accomplish a set of tasks as efficiently as possible.
Multi-Robot Systems Increasingly Adopted

- Research in multi-robot systems: multi-robot communication, team formation and control, path planning, task scheduling and routing
- Real-life applications: manufacturing, warehouse automation, pick up and delivery, surveillance at regular intervals, space exploration, and search and rescue, etc.
Focus on Assembly Manufacturing

• Who: assignment of tasks to robots
• When: schedule of tasks for each robot
• Key challenges
  • Temporal constraints on task completions and relations
  • Spatial/resource constraints create inter-coupled, disjunctive constraints
Taxonomy of Coordination Problems

- Task Allocation and Scheduling
  - The Gerkey’s taxonomy\(^1\)
  - The iTax taxonomy\(^2\)

Visual representation of the Gerkey’s taxonomy\(^1\)

Various types of tasks that can be performed by agents in iTax\(^2\)

---

Scheduling Class: [XD] ST-SR-TA

• Class
  • single-task robots (ST)
  • single-robot tasks (SR)
  • time-extended assignment (TA)
  • cross-schedule dependencies [XD] under the iTax taxonomy

• Two assumptions
  • Robot-task proficiency known beforehand
  • Travel time much smaller than task duration
# Real Robotics is Hard

<table>
<thead>
<tr>
<th>Ordering of Tasks</th>
<th>Task Allocation</th>
<th>Timing of Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{i,j} \in {0,1}, \forall \tau_i, \tau_j \in \tau$</td>
<td>$A_{a,i} \in {0,1}, \forall \tau_i \in \tau, a \in A$</td>
<td>$s_i, f_i \geq 0, \forall \tau_i \in \tau$</td>
</tr>
</tbody>
</table>

**Minimize Makespan**

$$\min z = \left( \max_{i,j} (f_j - s_i) \right) + g(x, A, s, f, \tau)$$

**1 Agent per Task**

$$\sum_{a \in A} A_{a,i} = 1, \forall \tau_i \in \tau$$

**Wait/Precedence constraints**

$$s_i - f_j \geq W_{i,j}, \forall W_{i,j} \in W$$

$$x_{i,j} = 1, \forall W_{i,j} \in W$$

**Deadline Constraints**

$$s_i - f_j \leq D_{i,j}, \forall D_{i,j} \in D$$

**Agent capabilities**

$$C_{i,a} \leq f_i - s_i + M(1 - A_{a,i}), \forall \tau_i \in \tau$$

**Agents execute tasks in series**

$$s_j - f_i \geq -M(1 - x_{i,j}) - M(2 - A_{a,i} - A_{a,j}), \forall \tau_i, \tau_j \in \tau$$

$$s_i - f_j \geq -Mx_{i,j} - M(2 - A_{a,i} - A_{a,j}), \forall \tau_i, \tau_j \in \tau$$

**Physical Constraints**

$$s_j - f_i \geq -M(1 - x_{i,j}), \forall \tau_i, \tau_j \in \tau_R$$

$$s_i - f_j \geq -Mx_{i,j}, \forall \tau_i, \tau_j \in \tau_R$$
Problem Description

Ordering of Tasks

\[ x_{i,j} \in \{0,1\}, \forall \tau_i, \tau_j \in \tau \]

Task Allocation

\[
\begin{align*}
\min z &= \max_{a \in A} \left( \sum_{\tau_i} C_{\tau_i}^a \times A_{\tau_i}^a \right) \\
1 &= \sum_{\tau_i} A_{\tau_i}^a, \forall a
\end{align*}
\]

Timing of Events

\[ s_i, f_i \geq 0, \forall \tau_i \in \tau \]

Minimize Makespan

\[(x, A, s, f, \tau)\]

1 Agent per Task

Wait/Precedence constraints

\[ x_{i,j} = 1 \ \forall W_{i,j} \in W \]

Deadline Constraints

\[ s_i - f_j \leq D_{i,j}, \forall D_{i,j} \in D \]

Agent capabilities

\[ C_i \leq f_i - s_i + M(1 - A_{a,i}), \forall \tau_i \in \tau \]

Agents execute tasks in series

\[ d_{\tau_i} - C_{\tau_i}^a \geq C_{\tau_j} \]

\[ \forall \tau_i, \tau_j \in \tau_{active} \ s.t. \]

\[ A_{\tau_i}^a = A_{\tau_j}^a \forall a, (\tau_i, \tau_j) \in \tau_R \]

Physical Constraints

Novel Sequencing Test
**Tercio: Robot Team Coordination**

**Task Allocation**

\[
\min z = \max_{a \in A} \left( \sum_{\tau_i} C_{\tau_i}^a \times A_{\tau_i}^a \right)
\]

\[
1 = \sum_{\tau_i} A_{\tau_i}^a, \forall a
\]

**Novel Sequencing Test**

\[
d_{\tau_i} - C_{\tau_i}^a \geq C_{\tau_j}
\]

\[
\forall \tau_i, \tau_j \in \tau_{active} \text{ s.t.}
\]

\[
A_{\tau_i}^a = A_{\tau_j}^a \forall a, (\tau_i, \tau_j) \in \tau_R
\]

**Schedule Satisfied?**

Real Robotics is Hard

Work begins on the fuselage according to the nominal schedule.

2017 Silver Vendor
Extracting Domain Expertise is the Key
Extracting Domain Expertise is the Key

Problems:

1) Manually extracting this knowledge is labor-intensive and often gives an inaccurate representation of the decision-making process

2) Application-specific nuances makes it impossible to scale expertise of the researcher
Need to Scale up Extraction of Domain Expertise

• Heuristic approaches
  • Lightweight and effective
  • Domain experts needed

• Exact solvers
  • Optimal schedule
  • Computationally expensive
    \(\rightarrow\) needs heuristic warm-start

• Meta-heuristics
  • Informed search
  • Randomly-generated solutions not feasible
    \(\rightarrow\) Need heuristic warm-start

• Learning
  • Could address these limitations
  • Only been applied on simple problems
Need to Scale up Extraction of Domain Expertise
Three Critical Challenges

1. How do we learn the right features of scheduling problems?

2. How do we learn a policy with application-specific reasoning?

3. How do we learn problem size-agnostic reasoning for scalability?

Graph Neural Networks!!!
Outline

Multi-Robot Coordination Problems

Introduction to Graph Neural Networks
  • Solving Multi-Robot Coordination with Graph Neural Networks
  • Ongoing and Future Work
Convolutional Neural Networks

- Extracting high level representations from the data automatically
- Outstanding performance on various vision-based tasks

VGG-16 Architecture, Simonyan et al. 2015
Convolutional Neural Networks (CNN)

- Key components:
  - Local connections
  - Shared weights
  - Pooling
  - Multi-layer structure

- CNN is effective for representation learning in Euclidean space, so any problem with a fixed matrix representation can benefit from it.

- Can we use it for representation learning in graph-structured data?

- A lot of real-world problems consist of data in the form of a graph.
Convolutional Neural Networks (CNN)

- Key components:
  - Local connections
  - Shared weights
  - Pooling
  - Multi-layer structure

- Can we use CNN for representation learning on graphs?
  - Nodes and pixels are different
  - Relations/Edges have directions 😞
Build Model using Computational Modules

• There have been a significant development in extending deep neural networks to non-Euclidean data, such as graphs and manifolds.

• Prior work sought to formulate appropriate model architectures for learning on graphs, which gave birth to the Graph Neural Networks (GNNs) family.
  • E.g., Scarselli et al., 2009; Bruna et al., 2013; Defferrard et al., 2016; Kipf & Welling, 2017; Hamilton et al., 2017; Velickovic et al., 2018
  • Propagation Module is used to propagate information between nodes so that the aggregated information could capture both feature and topological information
  • Sampling Module is used for large-scale graphs
  • Pooling Module is used to obtain high-level graph embeddings
Graph Neural Networks

• Graph neural network (GNN) generalizes deep learning to capture structural information in the data by modeling a set of node entities together with their relationships (edges).

• Broad application range: molecules, social networks, knowledge graphs and recommender systems (Zitnik et al., 2018; Schlichtkrull et al., 2018; Hamilton et al., 2018; Ying et al., 2018), or in general any datasets that have structural information.
Example: classification of paper research area.

- **DNN**: use paper content features

- **GNN**: use paper content features + citation relations between papers

---

**Structural Information**
Specify Graph Type and Scale

• Graphs with complex types could provide more information on nodes and their connections.
Find Graph Structure

• In **structural scenarios**, the graph structure is **explicit** in the applications, such as applications on molecules, social networks, knowledge graphs, etc.

![A model of an aspartame molecule](image1)

Introduction to Question Answering over Knowledge Graphs

Find Graph Structure

• In **non-structural scenarios**, graphs are implicit so that we have to first build the graph from the task

  • Ex, building a fully-connected “word” graph for text or building a scene graph for an image. After we get the graph, the later design process attempts to find an optimal GNN model on this specific graph.


Specify Graph Type and Scale

• **Homogeneous/Heterogeneous Graphs** Nodes and edges in homogeneous graphs have same types, while nodes and edges have different types in heterogeneous graphs. Types for nodes and edges play important roles in heterogeneous graphs and should be further considered.
Specify Graph Type and Scale

• **Static/Dynamic Graphs** When input features or the topology of the graph vary with time, the graph is regarded as a dynamic graph. The time information should be carefully considered in dynamic graphs.
Specify Graph Type and Scale

• No clear classification criterion for “small” and “large” graphs. The criterion is still changing with the development of computation devices (e.g. the speed and memory of GPUs).

• When the adjacency matrix or the graph Laplacian of a graph (the space complexity is $O(n^2)$) cannot be stored and processed by the device, then we regard the graph as a large-scale graph and then some sampling methods should be considered.

**Graph Laplacian** \[ L = D - A \]

D: degree matrix
A: adjacency matrix
Specify Graph Type and Scale

• No clear classification criterion for “small” and “large” graphs. The criterion is still changing with the development of computation devices (e.g. the speed and memory of GPUs).

• When the adjacency matrix or the graph Laplacian of a graph (the space complexity is $O(n^2)$) cannot be stored and processed by the device, then we regard the graph as a large-scale graph and then some sampling methods should be considered.

Consider an $L$-layer graph convolutional network with hidden state size $H$ running on an $N$-node graph. Storing the intermediate hidden states requires $O(NLH)$ memory, easily exceeding one GPU’s capacity with large $N$ (millions or even billions of nodes or edges).

Minibatch training with neighborhood sampling
An Overview of Computational Modules

GNN from A Message Passing Perspective

• Recent studies (Gilmer et al., 2017; Battaglia et al., 2018) manage to unify different GNN variants into the message passing paradigm.

• Every neighbor node talks to each other, by passing messages.
GNN from A Message Passing Perspective

• (Vanilla) Graph Convolution Operator

\[ h_i^{l+1} = \sigma \left( \sum_{j \in N_i} \frac{1}{c_{ij}} h_j^l W_R^l \right) \]

• Invariant of input orders of nodes
• Propagation by graph structure
• Graph-based reasoning

1. Send message
2. Receive message
3. Aggregate message
GNN from A Message Passing Perspective

• The message passing paradigm defines the edge-wise and node-wise computation of $G(V; E)$

Edge-wise: $m_{e}^{t+1} = \phi(x_v^t, x_u^t, w_e^t), (u, e, v) \in E$

message function used on each edge to generate a message by combining the edge feature with the features of its incident nodes

t: layer

src, edge, dst
GNN from A Message Passing Perspective

• The message passing paradigm defines the edge-wise and node-wise computation of $G(V; E)$

Edge-wise: $m_e^{t+1} = \emptyset(x_v^t, x_u^t, w_e^t), (u, e, v) \in E$

Node-wise: $x_v^{t+1} = \phi(x_v^t, \rho(\{m_e^{t+1}: (u, e, v) \in E\}))$

**update function** used on each node to update the node feature

**reduce function** used on each node to aggregate incoming messages
Graph Learning Tasks

- **Node-level tasks**
  - Node classification tries to categorize nodes into several classes.
  - Node regression predicts a continuous value for each node.
  - Node clustering aims to partition the nodes into several disjoint groups, where similar nodes should be in the same group.
Graph Learning Tasks

• **Edge-level tasks**
  - Edge classification and link prediction, which require the model to classify edge types or predict whether there is an edge existing between two given nodes.

Predict if there is a potential missing link between members

Should we recommend a connection between Henry and Terry?

How about Henry and Bill?
Graph Learning Tasks

• **Graph-level tasks**
  - Graph classification, graph regression, and graph matching, all of which need the model to learn graph representations.

---

Generative Models of Graphs

• Having generative models of graphs can be useful
  • Infer graphs from input (map building, relation extraction, etc.)
  • Graph completion (knowledge graph completion, social networks)
  • Generate new structures (drug discovery)

• Deep Generative Models of Graphs (DGMG) generates a graph by following a state machine. At each time step, you either:
  • Add a new node to the graph
  • Select two existing nodes and add an edge between them

Generative Models of Graphs

• Generation of a graph ⇒ A sequence of graph generating decisions.

• DGMG trains the model with behavior cloning.
  • Assume for each graph there exists a sequence of oracle actions $a_1, \ldots, a_T$ that generates it.

$$p(a_1, \ldots, a_T) = p(a_1)p(a_2|a_1) \cdots p(a_T|a_1, \ldots, a_{T-1})$$

• MLE loss

Graph Learning Tasks

• **Supervised setting** provides labeled data for training.

• **Semi-supervised setting** gives a small amount of labeled nodes and a large amount of unlabeled nodes for training.
  - In test phase, the transductive setting requires the model to predict the labels of the given unlabeled nodes, while the inductive setting provides new unlabeled nodes from the same distribution to infer.
  - Most node and edge classification tasks are semi-supervised. Most recently, a mixed transductive-inductive scheme is undertaken by Wang and Leskovec (2020) and Rossi et al. (2018), craving a new path towards the mixed setting.

• **Unsupervised setting** only offers unlabeled data for the model to find patterns. Node clustering is a typical unsupervised learning task.

• **Reinforcement learning setting** designs GNN-based policies to learn from interaction with the environment.
Survey Papers

  • gives a formal definition of early graph neural network approaches

  • provides a thorough review of geometric deep learning


Python Packages

• Deep Graph Library
  • Framework Agnostic (currently supporting PyTorch, MXNet and TensorFlow)
  • Large Collection of Model Zoos
    https://github.com/dmlc/dgl/tree/master/examples

• PyG (PyTorch Geometric)
  • PyTorch-Native

• Spektral
  • Uses Keras API & TensorFlow 2

More on DGL in the Coding Session
Outline

- Multi-Robot Coordination Problems

Introduction to Graph Neural Networks

Solving Multi-Robot Coordination with Graph Neural Networks

- Ongoing and Future Work
Solving Multi-Robot Coordination with Graph Neural Networks

- Simple Temporal Networks
- Constructing a GNN
- MDP Training Setup
- Imitation via Q-learning
- STN Simplification Trick
- Attention
- Heterogeneous Graph
Simple Temporal Networks

• Ubiquitous in modeling multi-robot task allocation and scheduling problems\(^1\).

• A STN is a directed edge-weighted graph \(<V, E>^2\).
  • Each node, \(V\), represent an event-related time point
  • Each edge, \(E\): \(i \rightarrow j\), is labeled by a weight \(a_{ij}\), representing the linear inequality \(X_j - X_i \leq a_{ij}\)

\[
6 \leq X_j - X_i \leq 10
\]


Representing Scheduling Problems

Example simple temporal network (STN) with 3 tasks

• Deadline constraint on its finish time, $f_1$, of task 1:
  $$f_1 \leq 7$$

• There is a wait constraint requiring the start of task 3 to be at least 2 minutes after finish of task 2:
  $$f_2 - s_3 \leq -2$$

• Challenge: Disjunctive Temporal Problem
  
  • Which order should the tree tasks be performed to minimize cost? $\Rightarrow$ NP Hard
Hard to Know: When to Do Nothing

Example #1:
• Tasks:
  • Task 1: 5 min
  • Task 2: 2 min
  • Task 3: 3 min
• Wait:
  • Task 2 must start at least 3 minutes after Task 1 finishes
• Deadline:
  • Task 2 must finish no more than 10 minutes after Task 1 starts

Example #2:
• Same as before except
  • Task 3: 4 min
Applying Graph Convolutions

Example for Node $f_2$

- Assume
  - $h_i = [1_{\text{start node}}, \text{duration}]^T \in \mathbb{R}^m, \forall i \in V$
  - $W_{\text{node}} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \in \mathbb{R}^{m \times m}$; $W_{\text{edge}} = \begin{bmatrix} -1 \\ -2 \end{bmatrix} \in \mathbb{R}^m$
  - $c_{ij} = |N(i)|, \forall i, j$

$$h_i' = \sigma \left( \sum_{j \in N_i} \frac{1}{c_{ij}} (W_{\text{node}} h_j + W_{\text{edge}} w_{ji}) \right)$$

(Vanilla)
Scheduling-based operator
Applying Graph Convolutions

Example for Node $f_2$

- Assume
  - $h_i = [1_{\text{start node}}, \text{duration}]^T \in \mathbb{R}^m, \forall i \in V$
  - $W_{\text{node}} = \begin{bmatrix} \frac{1}{3} & 2 \frac{1}{4} \\ 1 & 4 \end{bmatrix} \in \mathbb{R}^{m \times m}$; $W_{\text{edge}} = \begin{bmatrix} -1 \\ -2 \end{bmatrix} \in \mathbb{R}^m$
  - $c_{ij} = |N(i)|, \forall i, j$

$$h'_i = \sigma \left( \sum_{j \in N(i)} \frac{1}{c_{ij}} \right) \left( W_{\text{node}} h_j + W_{\text{edge}} w_{ji} \right)$$

$$h'_{f_2} = \sigma \left( \frac{1}{c_{s_3 f_2}} \left( W_{\text{node}} h_{s_3} + W_{\text{edge}} w_{s_3 f_2} \right) + \frac{1}{c_{s_2 f_2}} \left( W_{\text{node}} h_{s_2} + W_{\text{edge}} w_{s_2 f_2} \right) + \frac{1}{c_{f_0 f_2}} \left( W_{\text{node}} h_{f_0} + W_{\text{edge}} w_{f_0 f_2} \right) \right)$$

$$= \sigma \left( \frac{1}{3} \left[ \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \right] \left[ \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right] \right) + \frac{1}{3} \left( \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} \right) \left[ \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right] + \frac{1}{3} \left( \begin{bmatrix} 2 \\ 4 \end{bmatrix} \right) \left[ \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right]$$

$$= \sigma \left( \begin{bmatrix} \frac{1}{3} & 2 \\ 3 & 4 \end{bmatrix} \left[ \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right] \right) + \frac{1}{3} \left( \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix} \right) \left[ \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right] + \frac{1}{3} \left( \begin{bmatrix} 2 \\ 4 \end{bmatrix} \right) \left[ \begin{bmatrix} 1 \\ -2 \end{bmatrix} \right]$$

$$h'_{f_2} = \begin{bmatrix} 4.6 \\ 10 \end{bmatrix}$$

// ReLU activation fn()
Applying Graph Convolutions

Example for Node $f_2$

- Assume
  - $h_i = [\text{start node}, \text{duration}]^T \in \mathbb{R}^m, \forall i \in V$
  - $W_{\text{node}} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \in \mathbb{R}^{m \times m}$; $W_{\text{edge}} = \begin{bmatrix} -1 \\ -2 \end{bmatrix} \in \mathbb{R}^m$
  - $c_{ij} = |N(i)|, \forall i, j$

\[
h'_i = \sigma \left( \sum_{j \in N(i)} \frac{1}{c_{ij}} (W_{\text{node}} h_j + W_{\text{edge}} w_{ji}) \right)
\]
Multiple Convolutional Layers

Example for Node $f_2$

- Assume
  - $h_i^{(0)} = [1_{\text{start node}}, \text{duration}]^T \in \mathbb{R}^m$, $\forall i \in V$
  - $W_{\text{node}} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \in \mathbb{R}^{m \times m}$; $W_{\text{edge}} = \begin{bmatrix} -1 \\ -2 \end{bmatrix} \in \mathbb{R}^m$
  - $c_{ij} = |N(i)|$, $\forall i, j$

$$h_i^{(l+1)} = \sigma \left( \sum_{j \in N(i)} \frac{1}{c_{ij}} (W_{\text{node}} h_j^{(l)} + W_{\text{edge}} w_{ji}) \right)$$

- Can apply same operations multiple times to make sure enough information gets passed around to all the nodes.
  - Worst case requires $O(|V|)$ rounds.
- Rule of thumb: three rounds.
Computing “Graph” and “Task” Embeddings

Task Embeddings:

\[ h' \equiv [h'_1 \ldots h'_i \ldots h'_{|V|}] \]

• For our example...

\[ h' = [h_1 \ldots h_{f_2} \ldots h_{|V|}] \]

\[ = [h_1 \ldots \frac{4.6}{10} \ldots h_{|V|}] \]

Graph Embeddings:

• \[ H \equiv \frac{1}{|V|} \sum_{i=1}^{|V|} h'_i \]

• Averaging loses information...

  \[ \Rightarrow \text{We’ll have a better approach later (heterogeneous graphs)!} \]
Ranking Choices with Embeddings

- Should Task $\tau_1$, $\tau_2$, or $\tau_3$ come first?
- One approach: Connect regular NN output layer to make prediction.
  \[ \pi_\theta(\tau_i) = \frac{e^{z_i}}{\sum_{j=1}^{\mid\tau\mid} e^{z_j}} \]
  \[ z_i = \theta \left[ H; h'_{s_i}; h'_{f_i} \right] \]
  \[ \theta \in \mathbb{R}^{q \times (3m)} \]
- Training to predict the “correct” choice, $\pi_\theta(\tau_i^*)$
  \[ \Theta^* = \left\{ \theta^*, W_{node}^*, W_{edge}^* \right\} = \arg\min_{\Theta} -\log \pi_\theta(\tau_i^*) \]
Scheduling as a Markov Decision Process

• Markov decision process (MDP) – \( \langle S, A, T, R, \gamma \rangle \)

• **State, \( s \)**
  - STN
  - Unallocated tasks
  - Assignments of robots to tasks

• **Action, \( a = \langle \tau_i, r_j \rangle \)** – Adding unscheduled task, \( \tau_i \), to partial schedule for \( r_j \).

• **Transition** – Add edge in STN to enforce consequences of action
  - For action \( a_{t-1} = \langle \tau_i, r_j \rangle \) followed by \( a_t = \langle \tau_p, r_q \rangle \), do...
    - If \( r_j = r_q \), then add constraint such that \( s_p \geq f_i \)

• **Reward** – Value of scheduling action in a given state

• \( \gamma \) – Discount factor
Computing the Reward

- Example policy gradient-based approach
  - \( R(s, a) = (\text{shortest path from } f_0 \text{ to } s_0) \times \begin{cases} -1 & \text{if not finished} \\ -10 & \text{otherwise} \end{cases} \)
  - \( \hat{Q}(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} R_{t'} \)

**Policy, \( \pi_\theta \), assigns...**
- \( a_1 = \text{robot } r_1 \) to task \( \tau_1 \)
- \( a_2 = \text{robot } r_2 \) to task \( \tau_3 \)
- ...
- \( a_5 = \text{robot } r_1 \) to task \( \tau_2 \)

**Reward:**
- \( R_1 = -5 \)
- \( R_2 = -5 \)
- ...
- \( R_5 = -9 \times 10 = -90 \)

**Q-value Estimate:**
- \( \hat{Q}(s_1, a_1) = -5 + \gamma(-5) + \gamma^4(-90) = -100 \)
- \( \hat{Q}(s_2, a_2) = -5 + \gamma(-90) = -95 \)
- ...
- \( \hat{Q}(s_5, a_5) = -90 \)

Update policy: \( \Delta \theta = \sum_t^{\infty} \hat{Q}(s_t, a_t) \nabla \log \pi_\theta(s_t, a_t) \)

**Transitions:** For action \( a_{t-1} =< \tau_i, r_j > \) followed by \( a_t =< \tau_p, r_q >, \) if \( r_j = r_q \), then add constraint such that \( s_p \geq f_i \).

**Schedule State, \( S \)**

**Partial Schedule**

**Unscheduled Tasks**

---

How can we do this with a GNN?
Augmenting Graph for Multiple Robots

Multi-robot Formulation
• Assume
  • $h_i = [1_{\text{start node}}, \text{duration}, z_i^T] \in \mathbb{R}^{m+r}, \forall i \in V$
  • $W_{\text{node}} \in \mathbb{R}^{(m+r) \times (m+r)} ; W_{\text{edge}} \in \mathbb{R}^{(m+r)}$
  • $c_{ij} = |N(i)|, \forall i, j$

$$ z_i^T = \begin{cases} 0 & \text{if no robot assigned} \\ e_r & \text{otherwise} \end{cases} $$

1-hot encoding of $r$
Augmenting Graph for Multiple Robots

Multi-robot Formulation

- Assume
  - \( h_i = [1_{\text{start node}}, \text{duration}, z^T_r] \in \mathbb{R}^{m+r}, \forall i \in V \)
  - \( W_{\text{node}} \in \mathbb{R}^{(m+r) \times (m+r)} ; W_{\text{edge}} \in \mathbb{R}^{(m+r)} \)
  - \( c_{ij} = |N(i)|, \forall i, j \)

Example for Node \( f_1 \)

\[
W_{\text{node}} = \begin{bmatrix}
1 & 2 & 5 & 6 \\
3 & 4 & 7 & 8 \\
5 & 6 & 1 & 2 \\
7 & 8 & 3 & 4
\end{bmatrix} ;
W_{\text{edge}} = \begin{bmatrix}
1 \\
2 \\
3 \\
4
\end{bmatrix}
\]

Node Features

- \( h_{s_1} = [1,5, [1,0]]^T \)
- \( h_{s_2} = [1,4, [1,0]]^T \)
- \( h_{s_0} = [1,0, [0,0]]^T \)
- \( h_{s_1} = [0,0, [0,0]]^T \)

\[
h_i' = \sigma \left( \sum_{j \in N(i)} \frac{1}{c_{ij}} (W_{\text{node}} h_j + W_{\text{edge}} w_{ji}) \right)
\]

\[
h_i' = \sigma \left( \frac{1}{c_{ij}} \left[(W_{\text{node}} h_{s_1} + W_{\text{edge}} w_{s_1 f_1}) + (W_{\text{node}} h_{s_2} + 0) + (W_{\text{node}} h_{s_0} + 0) + 0 \right] \right)
\]

\[
h_i' = \begin{bmatrix}
9 & \frac{17}{4} & \frac{21}{2} & 29 \frac{3}{4}
\end{bmatrix}^T
\]
Augmenting Graph for Multiple Robots

Multi-robot Formulation
- Assume
  - $h_i = [\text{start node}, \text{duration}, z_r^T]^T \in \mathbb{R}^{m+r}, \forall i \in V$
  - $W_{\text{node}} \in \mathbb{R}^{(m+r) \times (m+r)}$; $W_{\text{edge}} \in \mathbb{R}^{(m+r)}$
  - $c_{ij} = |N(i)|, \forall i, j$

Example for Node $f_1$
- $W_{\text{node}} = \begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \\ 5 & 6 & 1 & 2 \\ 7 & 8 & 3 & 4 \end{bmatrix}$; $W_{\text{edge}} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$

Node Features
- $h_{s_1} = [1,5, [1,0]^T$
- $h_{s_2} = [1,4, [1,0]^T$
- $h_{s_0} = [1,0, [0,0]^T$
- $h_{s_1} = [0,0, [0,0]^T$

$$h'_i = \sigma \left( \sum_{j \in N(i)} \frac{1}{c_{ij}} (W_{\text{node}}h_j + W_{\text{edge}}w_{ji}) \right)$$

$$h'_i = \sigma \left( \frac{1}{c_{ij}} \left[ (W_{\text{node}}h_{s_1} + W_{\text{edge}}w_{s_1f_1}) + (W_{\text{node}}h_{s_2} + 0) + (W_{\text{node}}h_{s_0} + 0) + 0 \right] \right)$$

$$h'_i = \begin{bmatrix} 9 & \frac{1}{4} & 21 & \frac{1}{2} & 29 & \frac{3}{4} \end{bmatrix}^T$$
For each $\tau_i \in \tau_{\text{available}}$, compute $\left[H; h_{s_i}'; h_{f_i}'\right]_t$ and $f_\theta$.

Pick $\hat{\tau}_i \sim \pi_\theta(\tau_i)$.

Add constraint and propagate (e.g., APSP).

Set $t \leftarrow 0$

$\tau_{\text{available}} = \text{Set of unallocated tasks at time } t$

If $\tau_{\text{available}} = \emptyset$ yes

If $r_{\text{available}} = \emptyset$ yes

Pick $r_j \in r_{\text{available}}$ and assign to $\hat{\tau}_i$.

If all tasks allocated yes

Return schedule

Set $t \leftarrow t + 1$ no
Schedule Generation Nuances

• How do you pick the robot?
  • Expensive to iterate over all actions as task-robot pairs, \( a = (\tau, r) \)
  • Policy could output probability distribution over robots (i.e., \( \pi_\alpha(r|s) \))...
    • Conditioned on one robot, then output probability distribution over actions (i.e., \( \pi_\beta(\tau|s,r) \))
    • Resulting Policy: \( \Pi(a|s) = \pi_\beta(\tau|s,r)\pi_\alpha(r|s) \)

• How do you know what tasks can be selected?
  • Deeper dive on Simple Temporal Networks outside of scope
Imitation Learning with Expert Demonstrations

Why not learn directly via RL?

- Large search space
- Most infeasible
- Ineffective learning

Why imitation learning (IL)?

- Expert schedules (e.g., Gurobi solutions on small-scale problems).
- Scalability of GNNs
I’m searching for novel options

Einstellung Effect

Bilalic et al. (2008)
Imitation Learning with “Expert” Demonstrations

• If experts are available and solutions are high enough quality, can use their schedules.
  • Could learn from small- or large-scale schedules

• Else, we can learn from an exact method
  • Optimal but relegated to smaller-scale examples
    \( \rightarrow \) GNNs can learn representations on small schedules and extrapolate how to solve larger scheduling problems
Imitation Learning with Expert Demonstrations

Why not learn directly via RL?

- Large search space
- Most infeasible
- Ineffective learning

Why imitation learning (IL)?

- Expert schedules (e.g., Gurobi solutions on small-scale problems).
- Scalability of GNNs
**Imitation Learning with Expert Demonstrations**

Why not learn directly via RL?

- Large search space
- Most infeasible
- Ineffective learning

Why imitation learning (IL)?

- Expert schedules (e.g., Gurobi solutions on small-scale problems).
- Scalability of GNNs

At least two (RL-based) approaches:

- Policy gradients: $\Delta \Theta = \sum_{t}^{\infty} \hat{Q}(s_t, a_t) \nabla_{\Theta} \log \pi_{\Theta}(s_t, a_t)$

- Q-functions: $\Delta \Theta = -\sum_{t}^{\infty} \left( \hat{Q}(s_t, a_t) - Q_{\Theta}(s_t, a_t) \right) \nabla_{\Theta} Q_{\Theta}(s_t, a_t)$
At least two (RL-based) approaches:

• Policy gradients: $\Delta \Theta = \sum_{t}^{\infty} \hat{Q}(s_t, a_t) \nabla \log \pi_{\Theta}(s_t, a_t)$

• Q-functions: $\Delta \Theta = -\sum_{t}^{\infty} \left( \hat{Q}(s_t, a_t) - Q_{\Theta}(s_t, a_t) \right) \nabla Q_{\Theta}(s_t, a_t)$

\[ \hat{Q}(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} R_{t'} \]

Rule of thumb:

• For deterministic problems, learning a Q-function seems to be more efficient
  • Can learn from the quality of the resulting schedule

• For stochastic problems, learning a policy seems to help mitigate variance/stochasticity
  • On-policy could be safer w.r.t. constraints
Imitation Learning with Expert Demonstrations

• Expert loss

\[ L_{\text{expert}} = \mathbb{E} \left[ (\hat{Q}(s_t, a_t) - Q_\Theta(s_t, a_t))^2 \right] \]

\[ \hat{Q}(s_t, a_t) = \sum_{t'=t}^{\infty} \gamma^{t'-t} R_{t'} \]

• Counterfactual (alternate action) loss

  • Rather than computing the value of counterfactuals, assume to be worse by \( q_0 \)

\[ L_{\text{alt}} = \mathbb{E} \left[ \sum_{a \neq a_t} (Q_\Theta(s_t, a) - \min([Q_\Theta(s_t, a), \hat{Q}(s_t, a) - q_0]))^2 \right] \]

• Full loss:

\[ L_{\text{sup}} = L_{\text{exp}} + \lambda_1 L_{\text{alt}} + \lambda_2 \| \Theta \|_2 \]

Recall: \( Q_\Theta(s_t, a_t) \rightarrow Q_\Theta \left( [H_t, h'_s, h'_f] \right) \) where \( a_t = \langle t_i, r_j \rangle \)
Reducing Complexity

- In worst case, current approach requires $O(|\tau|)$ calls to an all-pairs shortest path algorithm (APSP)
  - One call per scheduling action
  - Johnson’s APSP is $O(|V|^2 \log V)$, which is fast, but it adds up.
- After each call, we need to pass graph through our GNN.
  - Each layer requires $O(|V|^2)$ operations
- Can we do better?
  - Under the homogeneous robot team setting, task duration is deterministic without knowing the actual assignment.
  - We develop a novel simplification trick to reduce the model complexity.
STN Simplification Trick

• 1. Run Johnson's all-pairs-shortest-paths algorithm to find the minimum distance graph
STN Simplification Trick

• 1. Run all-pairs-shortest-paths algorithm to find the minimum distance graph
STN Simplification Trick

1. Run all-pairs-shortest-paths algorithm to find the minimum distance graph
2. Remove the finish nodes \( f_i \) (except \( f_0 \)) from the distance graph
STN Simplification Trick

1. Run all-pairs-shortest-paths algorithm to find the minimum distance graph
2. Remove the finish nodes $f_i$ (except $f_0$) from the distance graph
3. Use the resulted graph for the graph neural network computation
STN Simplification Trick

1. Run all-pairs-shortest-paths algorithm to find the minimum distance graph
2. Remove the finish nodes $f_i$ (except $f_0$) from the distance graph
3. Use the resulted graph for the graph neural network computation

The simplified STN, using only half the nodes, still reserves all the necessary temporal constraints.

Each task can be represented by its start time node (as task nodes) with task duration now serving as its node feature.
Disjunctive Temporal Problem with Trick

1. Run all-pairs-shortest-paths algorithm to find the minimum distance graph
2. Remove the finish nodes $f_i$ (except $f_0$) from the distance graph
3. Use the resulted graph for the graph neural network computation
Graph Attention Networks

- Graph convolution operator
  \[ h_i^{l+1} = \sigma \left( \sum_{j \in N_i} c_{ij} h_j^l W_R^l \right) \]

Normalization is structure treats neighbors equally, which may hurt generalizability

→ A better solution: use attention [1]!

\[ h_i^{l+1} = \sigma \left( \sum_{j \in N_i} \alpha_{ij} h_j^l W_R^l \right) \]

\[ \alpha_{ij} = \frac{e^{\sigma'(a[h_i^l W_R^l \| h_j^l W_R^l])}}{\sum_{k \in N_i} e^{\sigma'(a[h_i^l W_R^l \| h_k^l W_R^l])}} \]

ReLU:
\[ \sigma(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{o/w} \end{cases} \]

Leaky ReLU:
\[ \sigma'(x) = \begin{cases} x & \text{if } x > 0 \\ bx & \text{o/w} \end{cases} \]

[1] P. Velickovic et al., Graph Attention Networks, ICLR, 2018

\textbf{Concatenation operator: ||}
Attention Equations

\[ h'_i = \sigma \left( \sum_{j \in N(i)} \alpha_{ij} (W_{node} h_j + W_{edge} w_{ji}) \right) \]

\[ \alpha_{ij} = \frac{e^{\sigma'(a[W_{node} h_i \| W_{node} h_j \| W_{edge} w_{ji}])}}{\sum_{k \in N(i)} e^{\sigma'(a[W_{node} h_i \| W_{node} h_k \| W_{edge} w_{ki}])}} \]
Results for “RoboGNN”

Heterogeneous Graph Neural Networks

- Heterogeneous graphs (Heterographs) are graphs that contain different types of nodes and edges / relations.

- Can learn per-edge-type message passing and per-node-type feature reduction mechanisms.

- Better interpretability and model expressiveness
  - Graph mining tasks (Zhang et al. KDD 2019, Wang et al. WWW 2019, Fu et al. The Web Conference 2020)
  - Malicious Account Detection (Liu et al. CIKM 2018)
ScheduleNet

• Extend STN formulation into heterogeneous graph
  • Robot, location nodes to encode location/resource constraints

We propose a novel heterogeneous graph attention network, ScheduleNet, that learns per-edge-type message passing and per-node-type feature reduction mechanisms on this graph.
Taxonomy of Coordination Problems

• Task Allocation and Scheduling
  • The Gerkey’s taxonomy¹
  • The iTax taxonomy²

ScheduleNet with Homogeneous Robots

Making every component of the problem a node

- Extend the simplified STN with robot, location/resource nodes

Towards a fully convolutional computing framework

- State summary node
- Q-value nodes.

Attention mechanism similar to RoboGNN
Heterogeneous Graph Attention Layer

1. Per-edge-type aggregation result computed first
2. Merged to obtain the destination node’s output feature

$$h'_i = \sigma \left( \sum_{j \in N_{temporal}(i)} \alpha_{ij}^{temporal} (W_{temporal}h_j + W_{tempEdge}w_{ji}) \right)$$

$$\alpha_{ij}^{temporal} = \text{softmax}_j \left( \sigma' (a_{tmp, [W_{temporal}h_i \| W_{temporal}h_j \| W_{tempEdge}w_{ji}]) \right)$$

$W_{edgeType}$: per-edge-type learnable weights
Heterogeneous Graph Attention Layer

1. Per-edge-type aggregation result computed first
2. Merged to obtain the destination node’s output feature

Task \( h_i' = \sigma \left( \sum_{j \in N_{\text{temporal}}(i)} \alpha_{ij}^{\text{temporal}} (W_{\text{temporal}} h_j + W_{\text{tempEdge}} w_{ji}) \right) \)

Robot \( h_i' = \sigma \left( \sum_{j \in N_{\text{assignedTo}}(i)} \alpha_{ij}^{\text{assignedTo}} W_{\text{assignedTo}} h_j + \sum_{k \in N_{\text{comm}}(i)} \alpha_{ij}^{\text{comm}} W_{\text{comm}} h_k \right) \)

Location \( h_i' = \sigma \left( \sum_{j \in N_{\text{locatedIn}}(i)} \alpha_{ij}^{\text{locatedIn}} W_{\text{locatedIn}} h_j + \sum_{k \in N_{\text{near}}(i)} \alpha_{ij}^{\text{near}} W_{\text{near}} h_k \right) \)

State \( h_s' = \sigma \left( \sum_{j \in N_{\text{taskIn}}(i)} \alpha_{ij}^{\text{taskIn}} W_{\text{taskIn}} h_j + \sum_{k \in N_{\text{robotIn}}(i)} \alpha_{ij}^{\text{robotIn}} W_{\text{robotIn}} h_k + \sum_{m \in N_{\text{locIn}}(i)} \alpha_{ij}^{\text{locIn}} W_{\text{locIn}} h_m + W_{\text{stateIn}} h_s \right) \)

\( \alpha_{ij}^{\text{edgeName}} = \text{softmax}_j \left( \sigma'( \alpha_{\text{edgeName}} [W_{\text{dstType}} h_i || W_{\text{edgeName}} h_j]) \right) \)

*Depends on the destination node type:* equal to \( W_{\text{comm}} \) \( W_{\text{near}} \) and \( W_{\text{stateIn}} \) when the destination node type is robot, location and state, respectively.
Heterogeneous Graph Attention Layer

1. Per-edge-type aggregation result computed first
2. Merged to obtain the destination node’s output feature

\[ h'_i = \sigma \left( \sum_{j \in N_{\text{temporal}}(i)} \alpha_{ij}^{\text{temporal}} (W_{\text{temporal}}h_j + W_{\text{tempEdge}}w_{ji}) \right) \]

\[ h'_i = \sigma \left( \sum_{j \in N_{\text{assignedTo}}(i)} \alpha_{ij}^{\text{assignedTo}} W_{\text{assignedTo}}h_j + \sum_{k \in N_{\text{comm}}(l)} \alpha_{ik}^{\text{comm}} W_{\text{comm}}h_k \right) \]

\[ h'_i = \sigma \left( \sum_{j \in N_{\text{locatedIn}}(l)} \alpha_{ij}^{\text{locatedIn}} W_{\text{locatedIn}}h_j + \sum_{k \in N_{\text{near}}(l)} \alpha_{ik}^{\text{near}} W_{\text{near}}h_k \right) \]

\[ h'_i = \sigma \left( \sum_{j \in N_{\text{taskIn}}(l)} \alpha_{ij}^{\text{taskIn}} W_{\text{taskIn}}h_j + \sum_{k \in N_{\text{robotIn}}(l)} \alpha_{ik}^{\text{robotIn}} W_{\text{robotIn}}h_k + \sum_{m \in N_{\text{locIn}}(l)} \alpha_{im}^{\text{locIn}} W_{\text{locIn}}h_m + W_{\text{stateIn}}h_s \right) \]

\[ Q \text{ value } h'_q = \sigma (W_{\text{taskTo}}h_t + W_{\text{robotTo}}h_r + W_{\text{stateTo}}h_s + W_{\text{valueTo}}h_q) \]

\[ \mathbf{w}_{\text{edgeType}}: \text{per-edge-type learnable weights} \]
Results

• Dataset (homogeneous robots)
  • 2-robot teams, 5-robot teams & 10-robot teams.
    • Small (16-20 tasks)
    • Medium (40-50 tasks)
    • Large (80-100 tasks)
    • Ex-large (160-200 tasks).

• Metrics
  • Percentage of problems solved within optimality ratio
    • “Solved” at ratio, $r$, if $\frac{z_{\text{algorithm}}}{z_{\text{optimal}}} \leq r$.

• Benchmarks
  • EDF, Tercio, and Gurobi
  • Homogeneous GNN$^1$
    • An early version of the RoboGNN scheduler, which also hard-codes the robot information instead of using robot-specific features, and without the simplification trick.
Results on 10-Robot Teams

• For ex-large problems, ScheduleNet managed to find substantially more feasible schedules than Gurobi.

Running Time Statistics

- Only feasible solutions were counted for each method.
- With increasing size...
  - ScheduleNet solution quality consistent with comparable increase in computation time.

Application-Specific Objective Function

• Minimizing the weighted sum of the completion times.

\[ z = \sum_i c_i f_i, \quad c \sim U[1, 10] \]

• Highest Cost Tardiness First (HCTF) priority heuristic.

ScheduleNet started to outperform HCTF when ratio became larger, resulting in a better overall performance.

Workers are Homogeneous

\[ C_i \leq f_i - s_i, \forall \tau_i \in \tau \]
Workers are Homogeneous Heterogeneous

\[ C_i \leq f_i - s_i, \forall \tau_i \in \tau \]

\[ C_{i,a} \leq f_i - s_i + M(1 - A_{a,i}), \forall \tau_i \in \tau \]
ScheduleNet with Heterogeneous Robots

- The duration of a task depends on the robot which is assigned to the task.
- For scheduled tasks, duration is given by assigned robot.
- For unscheduled tasks, can only use a relaxed set of bounds (min and max duration among all robots)

\[ \text{dur}_{\text{min}} \leq f_i - s_i \leq \text{dur}_{\text{max}}, \forall \tau_i \in \{\text{unscheduled}\} \]

- To preserve all need information:
  - Extend the set of task node features to include multiple descriptive statistics.
  - Add two new edge types between nodes of unscheduled tasks and robot nodes.
ScheduleNet with Heterogeneous Robots

- Handling 2D proximity constraints
- Only connect locations that fall within the minimum allowed safety distance to represent the proximity constraints
  - Instead of connecting location nodes with each other
- The edge type (location -> near -> location), although shares the same name as in homogeneous robot case, now encodes the 2D proximity constraints.
Heterogeneous Graph Attention Layer

- The feature update formula for scheduled tasks remains the same
  - Newly-added edges only affect unscheduled tasks
- For unscheduled tasks
  \[
  h_i' = \sigma \left( \sum_{j \in N_{\text{temporal}}(i)} \alpha_{ij} W_{\text{temporal}} h_j + W_{\text{tempEdge}} \alpha_{ij} \text{useTime} h_j + \sum_{k \in N_{\text{useTime}}(i)} \alpha_{ik} W_{\text{useTime}} h_k + W_{\text{useTimeEdge}} \alpha_{ik} \text{edge} \right)
  \]
- Attention coefficients
  \[
  \alpha_{ij} \text{useTime} = \text{softmax}_k \left( \sigma \left( \tilde{a}_{ij}^T \text{useTime} \left[ W_{\text{temp}} \tilde{h}_i | W_{\text{useTime}} \tilde{h}_k | W_{\text{useTimeEdge}} \text{edge} \right] \right) \right)
  \]

- The feature update equation of robot nodes also changes
  \[
  h_i' = \sigma \left( \sum_{j \in N_{\text{assignedTo}}(i)} \alpha_{ij} W_{\text{assignedTo}} h_j + \sum_{k \in N_{\text{comm}}(i)} \alpha_{ik} W_{\text{comm}} h_k + \sum_{m \in N_{\text{takeTime}}(i)} \alpha_{im} W_{\text{takeTime}} h_m + W_{\text{takeEdge}} \alpha_{im} \text{edge} \right)
  \]
- Attention coefficients
  \[
  \alpha_{im} \text{takeTime} = \text{softmax}_m \left( \sigma \left( \tilde{a}_{im}^T \text{takeTime} \left[ W_{\text{comm}} \tilde{h}_i | W_{\text{takeTime}} \tilde{h}_m | W_{\text{takeEdge}} \text{edge} \right] \right) \right)
  \]
- No changes for location nodes, state and value nodes
  \[
  N_{\text{near}}(i) \text{ now only considers neighbor locations within safety distance instead of all locations}
  \]
Results on 10-Robot Teams

For extra-large problems, EDF, Tercio and Gurobi all failed to find any feasible solutions, whereas ScheduleNet was able to find schedules for up to 24% of the problems.

Running Time Statistics

- Computation times in heterogeneous case increased for all three methods.

- For ScheduleNet, similar time change patterns with respect to increasing problem scales can be observed as the homogeneous robot case.

ScheduleNet in Robotarium

- A five-robot team in a simulated environment for airplane fuselage construction
  - Left: homogeneous robot case in 1D space
  - Right: heterogeneous robot case in 2D space


### ScheduleNet Takeaways

| Outperforming prior work | • Optimality  
| • Feasibility |
|--------------------------|-------------------|
| Better scalability       | • Up to 200 tasks, 10 robots, 10 locations |
| A more flexible framework. | • Adapt to changing objectives |
| An order of magnitude speedup vs. an exact method. | • At least 10x vs Gurobi |
Outline

• Multi-Robot Coordination Problems

• Introduction to Graph Neural Networks

Solving Multi-Robot Coordination with Graph Neural Networks

• Ongoing and Future Work
Outline

• Multi-Robot Coordination Problems

• Introduction to Graph Neural Networks

• Solving Multi-Robot Coordination with Graph Neural Networks

Ongoing and Future Work
Predictive Aircraft Maintenance

• Aircraft maintenance scheduling
  – Prevent / reduce the adverse effect of aircraft failures
  – Maximize aircraft availability
  – Picking the right time for repair / maintenance
Heterogeneous Graph Model for Policy Learning

- We develop novel heterogeneous graph attention networks to enable end-to-end policy learning from observations.
- We implement different Imitation Learning & RL algorithms and tested their performance in the scheduling environment.

- Imitation learning from hybrid heuristic
- RL, Q-learning-based
- RL, policy gradient-based
  - REINFORCE
  - REINFORCE with time-based baselines [1]
  - Advantage Actor Critic (A2C)
  - A2C with Generalized Advantage Estimation [2]

HetGPO Learns to Outperform Prior State-of-the-Art

### Profit-based Objective

<table>
<thead>
<tr>
<th>Methods</th>
<th>Small (M2 (%))</th>
<th>Medium (M2 (%))</th>
<th>Large (M2 (%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.522 ± 0.025</td>
<td>0.532 ± 0.021</td>
<td>0.533 ± 0.016</td>
</tr>
<tr>
<td>Corrective</td>
<td>0.539 ± 0.023</td>
<td>0.547 ± 0.016</td>
<td>0.546 ± 0.016</td>
</tr>
<tr>
<td>Condition-based</td>
<td>0.656 ± 0.050</td>
<td>0.661 ± 0.041</td>
<td>0.648 ± 0.051</td>
</tr>
<tr>
<td>Periodic</td>
<td>0.599 ± 0.051</td>
<td>0.598 ± 0.047</td>
<td>0.587 ± 0.048</td>
</tr>
<tr>
<td>Model-based</td>
<td>0.669 ± 0.052</td>
<td>0.671 ± 0.044</td>
<td>0.658 ± 0.054</td>
</tr>
<tr>
<td>DeepRM</td>
<td>0.533 ± 0.015</td>
<td>0.538 ± 0.011</td>
<td>0.539 ± 0.013</td>
</tr>
<tr>
<td>Decima</td>
<td>0.651 ± 0.021</td>
<td>0.660 ± 0.017</td>
<td>0.663 ± 0.014</td>
</tr>
<tr>
<td>HetGPO-Single</td>
<td>0.680 ± 0.012</td>
<td>0.676 ± 0.011</td>
<td>0.666 ± 0.011</td>
</tr>
<tr>
<td>HetGPO-Skip</td>
<td>0.695 ± 0.010</td>
<td>0.697 ± 0.009</td>
<td>0.695 ± 0.008</td>
</tr>
<tr>
<td>HetGPO-Full</td>
<td>0.693 ± 0.011</td>
<td>0.694 ± 0.009</td>
<td>0.693 ± 0.008</td>
</tr>
</tbody>
</table>

### Revenue-based Objective

1% improvement for revenue metric results in $657,800,000 revenue increase. We estimate HTGPO-Full would achieve $9,270,000,000 improvement in revenue over Corrective scheduling.

GNN in Multi-Agent Systems

• Straightforward modeling of teams as graphs
  • Agents as nodes
  • Interaction/Relation as edges

• The message passing computation of Graph Convolutional Layer aligns naturally with communication in multi-agent systems
  • Decentralized execution as each node operates on local information
  • Scalability in terms of nodes
Wildfire Fighting with Composite Teams

Three aspects of the coordination problem in Heterogeneous Teams

I. How to explore? How to prioritize/deprioritize different regions?

II. When to communicate sensed information?

III. Whom to communicate to with the information?

IV. How to stay “in-touch” (e.g., within communication ranges) while sensing or manipulating?

V. How to choose from received tasks?
HetNet uses heterogeneous graph-attention networks (HetGAT) and a class-specific encoder-decoder comm. channel in a CTDE paradigm to learn binarized comm. for coordinating composite teams.
Predator agents need to find a stationary hidden prey and move to its location as fast as possible. Agents only see their own grid and can move $\downarrow, \uparrow, \rightarrow, \leftarrow$.

N Predator agents need to find a stationary prey and move to its location. Capture agents (blind) must do the same and also take the capture_prey action when on a prey.

Perception agents must explore an environment to discover hidden fires. Action agents (blind) must extinguish found fires. Fire propagates and grows over time.

We benchmark HetNet against similar CTDE on-policy methods: CommNet, IC3Net, TarMAC, and MAGIC.
Empirical Evaluation: Experimental Results (Training & Testing)

Summary

1- HetNet significantly outperforms all baselines in both training and testing.

2- HetNet’s advantage is much more significant when the heterogeneous coordination dynamics to solve the game are more complex (i.e., FC).

3- Overall, the heterogeneous communication model learned by HetNet outperforms baselines by achieving 8.1% to 434.7% performance improvement.

---

Summary

1- HetNet model achieves 200x reduction in the required communication bandwidth per round of communication over baselines while also setting a new SOTA in team performance.

Challenges & Open Questions

• Multi-Robot Coordination
  • Production environments in the real world
    • Machine breakdowns
    • Unexpected releases of high priority jobs
    • Uncertainty in the processing times
    • ....
  • Learning the `pickRobot()` function inside GNN
    \[
    \pi(u|x) = \pi(\tau, r|x) = \pi(r|x) \cdot \pi(\tau|r, x)
    \]
• Human co-workers

• Graph Neural Network Modeling
  • Graph complexity
  • Batching of heterogeneous graphs
  • Distributed training of large-scale problems

\(\Delta, \nu, u > \delta, \nu, \Delta > \tau, \nu, \Delta > \nu, \Delta > \nu, \Delta > \nu, \Delta > \nu, \Delta >\)
Thanks for your attention!

Q&A

What’s Next: coding tutorial on working with ScheduleNet