Seeing the System: A World Model for SDG&E Scheduling
DSC Capstone Quarter 2 Project Report

Zhenghao Gong z3gong@ucsd.edu
Bingyan Liu bil010@ucsd.edu
Stephanie Wang yuw168@ucsd.edu
Weijie Zhang wez042@ucsd.edu

Phi Nguyen pnguyen@sdge.com
Fatemeh Aarabi FAarabi@sdge.com
Mike Hilden MHilden@sdge.com

Abstract

Modern utility operations rely on complex scheduling systems that must coordinate thousands of tasks, engineers, geographic districts, and operational constraints. However, existing scheduling pipelines often emphasize short-term feasibility rather than systemic understanding, leading to limited visibility into long-term risk patterns such as overdue tasks and resource imbalance. This project proposes a world-model–driven analytical framework for San Diego Gas & Electric (SDG&E) scheduling that aims to "see the system" rather than only optimize isolated outputs. We implement a constraint-aware scheduling program that first generates multiple feasible candidate schedules under operational and resource limitations. These candidates are subsequently evaluated through a world-model–based predictive framework that estimates overdue risk, duration deviation, and schedule-level performance indicators. To enable this evaluation, we construct a heterogeneous graph representation of each daily schedule using the SDG&E CLICK dataset, integrating tasks, crews, technicians, task types, districts, and temporal relationships into a unified relational structure. Empirical analysis shows that modeling scheduling as a connected system enhances interpretability, exposes hidden congestion patterns, and enables proactive rather than reactive planning. Beyond predictive accuracy, the proposed framework functions as a practical decision-support simulator, offering transparency, scalability, and strategic insight for real-world utility scheduling. This work demonstrates how graph-based world modeling can transform operational scheduling into a system-level intelligence process that jointly optimizes feasibility, risk awareness, and long-term efficiency.

Website: https://georgong.github.io/DSC180B/
Code: https://github.com/georgong/WorkOrder-World-Model

1 Introduction

San Diego Gas & Electric (SDG&E, hereafter referred to as SDG&E) provides essential energy infrastructure for millions of customers throughout San Diego and Southern Orange County. As a company dedicated to clean and reliable service, SDG&E continually seeks to enhance its operational performance to meet community expectations for safety, efficiency, and sustainability. A central challenge in achieving these goals lies in the complexity of utility field operations, which require large-scale coordination of maintenance, inspection, and repair tasks distributed across time, geography, and workforce resources.

Daily scheduling decisions must account for thousands of tasks, technician availability, crew composition, travel distances, and operational priorities. The interactions among these factors make scheduling not merely a logistical exercise but a system-level challenge in which local assignment decisions can produce cascading global effects. Inefficient schedules may lead to overdue accumulation, regional congestion, and uneven workload distribution, ultimately affecting both operational cost and long-term service reliability. Existing industrial scheduling pipelines often emphasize feasibility and rule-based optimization, ensuring that a schedule satisfies constraints without necessarily revealing its broader structural implications. While such approaches can produce executable plans, they typically lack mechanisms for understanding how scheduling choices influence delay propagation, resource bottlenecks, or long-term performance stability. As a result, planners may be able to generate schedules, but they have limited visibility into systemic risk and limited tools for comparing alternative plans beyond surface-level metrics.

To address this limitation, this project introduces a graph-based world model framework that treats scheduling as an interconnected system rather than a collection of independent tasks. Using the SDG&E CLICK dataset, we represent daily schedules as heterogeneous graphs that encode relationships among tasks, crews, technicians, districts, and temporal dependencies. This system-level representation enables the model to capture structural patterns that directly impact utility operations, such as an imbalanced workload or a delay propagation. For example, workload imbalance emerges when certain engineers are assigned disproportionately high volumes of work while others remain underutilized, which can reduce workforce efficiency. Delay propagation reflects how small scheduling disruptions can cascade through downstream assignments, causing subsequent tasks to become overdue. By explicitly modeling these operational dependencies, the proposed framework provides scheudlers with greater visibility into how scheduling decisions influence system-wide performance, enabling earlier identification of bottlenecks and more proactive operational planning. Building upon this representation, we develop an end-to-end pipeline designed to evaluate the performance of existing and historical schedules. Through the pipeline, operational scheduling data from SDG&E is first transformed from relational tables into a heterogeneous graph. This graph representation is then used to train a predictive world model that predicts system-level metrics.

The primary objective of this work is to construct an analytical framework that provides interpretable insights into scheduling performance. Instead of optimizing schedules directly, the system evaluates how well a given schedule performs by predicting risk indicators such as overdue likelihood and completion-time deviation. Through empirical analysis, we demonstrate that modeling scheduling as a connected structure improves interpretability, exposes hidden inefficiencies, and offers a scalable foundation for intelligent utility scheduling systems.

2 Methods

Our project develops a world model that can simulate how a full day of SDG&E schedules behaves under different conditions. The model is trained on real field-operation data provided by SDG&E, which contains detailed records of tasks, crews, technicians, assignments, and sequencing relationships. Rather than predicting task outcomes in isolation, the framework represents an entire daily schedule as a connected structure. This system level perspective enables the model to capture interactions that emerge through shared resources, temporal dependencies, and geographic constraints patterns that are not observable through independent task modeling.

To construct this representation, we transform the raw operational tables into a heterogeneous graph and analyze its structural and statistical properties to ensure that relational dependencies are meaningfully encoded. To evaluate how different graph learning model capture these relationships, we experiment with multiple Graph Neural Network architectures, including GraphSAGE, RGCN, and HGT. These models differ in how they aggregate neighborhood information and incorporate relation types, allowing us to compare the effectiveness of simple neighborhood aggregation versus relation-aware message passing in heterogeneous scheduling data. In addition to predictive performance, we analyze the learned node embeddings produced by the models. Specifically, we apply dimensionality reduction using t-SNE to the final hidden-layer embeddings of the model to visualize the latent representation space and examine whether the models capture meaningful structural patterns in the constructed heterogeneous graph.

2.1 Data Description & Preprocessing

A central part of this project relies on operational data provided by SDG&E. The dataset captures detailed information about field tasks, engineers, assignments, geographic organization, and work timing. Each table describes a different part of daily operations, such as when tasks were scheduled and completed, which engineer was assigned to which work, and how work is organized by district and department. After merging these tables, we have nearly ten million rows of historical records, giving us a broad analysis of how field operations unfold across many real conditions. This scale and diversity are critical to allow the model to learn scheduling dynamics such as delay propagation, workload congestion, and uneven resource utilization.

We load raw SDG&E data from CSV tables using a YAML configuration that specifies the columns to read and the intended data type for each field. The loader supports both singlefile tables and shared files, concatenates shards vertically, standardizes missing values to nulls, and enforces consistent schemas by adding any missing columns. After ingestion, we perform schema-driven outlier filtering, where categorical columns are filtered to a set of values, numeric columns are constrained to lower and upper bounds, and datetime columns are restricted to valid time ranges.

After data preprocessing, we generate node features using a schema-driven pipeline implemented in Polars. Numeric variables are cast to floating-point format. Categorical variables are compressed using a top-K frequency strategy ($K = 30$), with infrequent categories grouped into an "OTHER" bucket before one-hot encoding. Temporal features such as start time and due date are decomposed into year, month, day, hour, and weekday components, with additional cyclic sine–cosine encodings to capture periodic effects. For tasks and assignments tables, we also derive operational metrics such as scheduled duration, lead time, slack time, and completion time. Raw timestamp columns are removed after transformation.

2.2 Heterogeneous Graph Construction

We convert the operational records into a heterogeneous graph with 7 node types and 20 edge relations:

- **Task nodes** (5.4M nodes): feature dimension 72.
- **Assignment nodes** (5.7M nodes): feature dimension 42.
- **Engineer nodes** (10,941 nodes): feature dimension 46.
- **District nodes** (316 nodes): feature dimension 5.
- **Department nodes** (8 nodes): feature dimension 1.
- **Task statuses nodes** (29 nodes): feature dimension 1.
- **Task types nodes** (3096 nodes): feature dimension 9.

Edges encode operational relationships such as task–assignment, assignment–engineer, and task–district connections, etc.

For each node type, the row-level features are computed from cleaned tables and then aggregated using mean pooling by entity key to produce node-level feature tensors. For assignment nodes, the completion time is separated as the supervised prediction target to prevent data leakage.

Relational edges are constructed through schema-defined join operations between tables. For each pair of linked entities, we join on shared keys (e.g., task ID, assignment ID, engineer ID), map raw identifiers to node indices, and generate directed "edge_index" tensors. In addition to direct relations, we construct higher-order connections via metapath projection (e.g., engineer $\rightarrow$ assignment $\rightarrow$ task $\rightarrow$ task type) using sparse matrix multiplication. This approach enables the model to capture indirect but operationally meaningful relationships.

To encode contextual similarity, we further introduce intra-type edges based on shared temporal or categorical traits, such as same weekday, month, or category. Depending on configuration, nodes within the same group are connected using pairwise sampling, random k-nearest grouping, or context-node construction, with limits on group size and edge counts. The resulting graph captures structural, temporal, and categorical dependencies while remaining computationally tractable.

2.3 Model Architecture

To evaluate the effectiveness of incorporating relational structure into prediction, we implemented two non-graph baseline models and three graph-based models of increasing expressive power: LightGBM, MLP, GraphSAGE, RGCN, and HGT. These models allow us to compare piecewise tabular learning against structured representation learning on the same dataset.

2.3.1 Baseline Models

We consider two non-graph baselines: a **Multi-Layer Perceptron (MLP)** and **LightGBM**. Both models treat each target sample independently and cannot directly operate on graph data. Therefore, the local neighborhood of each target node is transformed into a fixed-length feature vector by flattening relational information into aggregated statistics.

The MLP baseline represents a neural approach for tabular prediction, while LightGBM serves as a strong tree-based baseline widely used for structured data. These models allow us to evaluate how much performance gain comes specifically from modeling relational structure with graph neural networks.

2.3.2 RGCN

RGCN extends graph convolution to heterogeneous graphs by incorporating relation-specific transformations[3]. Each edge type is modeled with its own weight matrix:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathscr{R}} \sum_{j\in\mathscr{N}i^r} \frac{1}{ci,r} W_r^{(l)} h_j^{(l)}\right)$$

where ($\mathscr{R}$) represents the set of relation types. This design allows the model to distinguish semantically different interactions (e.g., assignment-to-task versus geographic association), enabling richer modeling of structured operational dependencies.

RGCN is particularly suited for datasets where relationships carry explicit meaning defined by domain schema.

2.3.3 GraphSAGE

GraphSAGE introduces explicit neighborhood aggregation through message passing[1]. Instead of flattening relational data, it learns node embeddings by sampling and aggregating features from neighboring nodes:

$$h_i^{(l+1)} = \sigma\left(W^{(l)} \cdot \mathrm{AGG}\left(h_i^{(l)} \cup h_j^{(l)}, j \in \mathcal{N}(i)\right)\right)$$

We employ mean aggregation with stochastic neighbor sampling to ensure scalability. GraphSAGE shares parameters across all edges and does not distinguish relation types, making it computationally efficient while still capturing local structural context. This model provides an inductive framework capable of generalizing to unseen nodes.

2.3.4 HGT

HGT further enhances representational capacity by introducing type-aware attention mechanisms[2]. Unlike uniform aggregation, HGT dynamically weighs messages based on node type, edge type, and learned attention scores:

$$\mathrm{Attention}_{(s,t,r)} = \mathrm{softmax}\left(\frac{(Q_s W_r^Q)(K_t W_r^K)^T}{\sqrt{d}}\right)$$

HGT is designed to capture higher-order dependencies and global structural signals, making it well-suited for modeling organizational and geographic interactions that influence task outcomes.

2.4 Model Training

The graph models were implemented using the PyTorch Geometric library, which provides scalable tools for constructing and training Graph Neural Networks on heterogeneous graphs. All experiments were run on an Apple M4 Pro. During training, Weights & Biases (WandB) was used to track model performance, record training metrics, and manage experiment configurations. This logging framework allowed us to monitor model performance behavior and compare different model architectures throughout the development process.

Training was performed using the Adam optimizer with a learning rate of 2×10^{-3} and weight decay of 1×10^{-4}. A dropout rate of 0.1 was applied during training to improve model generalization and reduce overfitting. To scale training to large graphs, we adopted a neighborhood sampling strategy where five neighbors were sampled per layer during message passing. Mini-batch training was used with a batch size of 2048 nodes, and models were trained for 2800 optimization steps.

To evaluate model performance, we used 2-fold cross-validation to assess generalization across subsets of the data. The random seed was fixed to ensure reproducibility of training results and data splits. The full set of training hyperparameters used in the experiments is summarized in Table 1.

Table 1: Model training configuration.

Category	**Value**
Dropout	0.1
Optimizer	Adam
Learning Rate	2×10^{-3}
Weight Decay	1×10^{-4}
Training Steps	2800
Neighbor Sampling	5 neighbors per layer
Batch Size	2048
Evaluation	2-fold cross-validation
Random Seed	0

2.5 Application Design

Our prediction model is designed as an analytical evaluation layer that operates alongside SDG&E's existing scheduling platform. Rather than replacing current scheduling systems, the model evaluates the performance of completed or historical schedules by reconstructing their heterogeneous graph representations and estimating operational risk indicators. Given a schedule, the system predicts assignment-level risk scores and aggregates these results into system-level metrics such as delay risk scores, workload imbalance across engineers, and risk distribution across districts and departments.

In addition to predictive modeling, we developed a web application that integrates the trained graph-based model with a dashboard interface. The dashboard enables users to explore the relational structure of schedules through graph visualization and inspect derived performance metrics. By analyzing historical schedules, schedulers can identify bottlenecks, uneven resource utilization, and patterns of delay propagation that may not be visible through traditional tabular-based dashboard reporting.

3 Results

In this section, we present the empirical evaluation of our proposed graph based framework for scheduling analysis. The goal of the experiments is to examine whether modeling scheduling data as a relational graph can improve prediction performance compared with traditional tabular approaches. We first compare several graph neural network architectures to understand how different relational learning mechanisms perform on the constructed heterogeneous graph. We then evaluate the best performing graph model against strong tabular baselines to assess the benefits of incorporating structural information from the scheduling system.

3.1 Graph Model Comparison

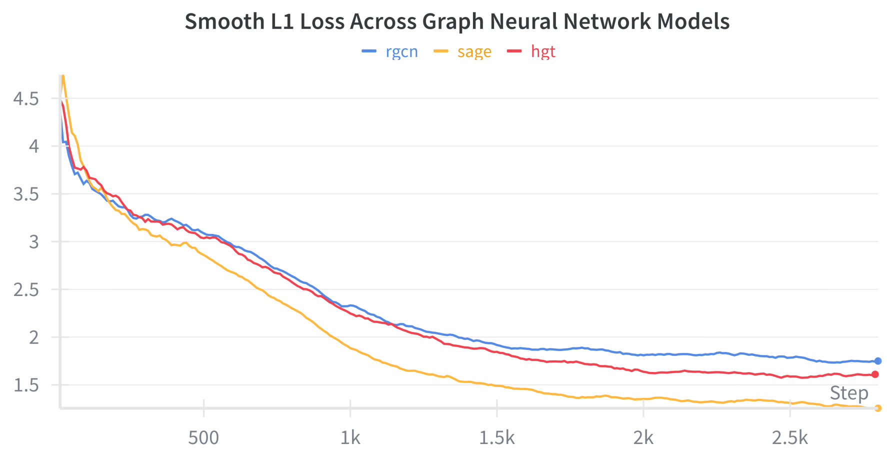


Figure 1: Training convergence comparison of GNN architectures under Smooth L1 loss.

Table 2: Model comparison using Smooth L1 loss.

Model	**Step**	**Smooth L1**
RGCN	2800	1.749
HGT	2800	1.610
MLP	2800	1.584
GraphSAGE	2800	**1.255**

We first compare several graph neural network architectures on the constructed heterograph, including RGCN, HGT, and GraphSAGE. All models are trained under the same training configuration summarized in Table 1. Figure 1 shows the training convergence under the Smooth L1 loss. All graph models exhibit stable convergence, while GraphSAGE consistently achieves lower loss values compared to the other architectures.

As summarized in Table 2, GraphSAGE obtains the best final performance with a Smooth L1 loss of 1.255, outperforming RGCN (1.749) and HGT (1.610). This result suggests that the neighborhood aggregation mechanism of GraphSAGE effectively captures local relational patterns in the task graph.

Based on this comparison, GraphSAGE is selected as the representative graph model for the subsequent experiments.

3.2 Graph Models vs. Tabular Models

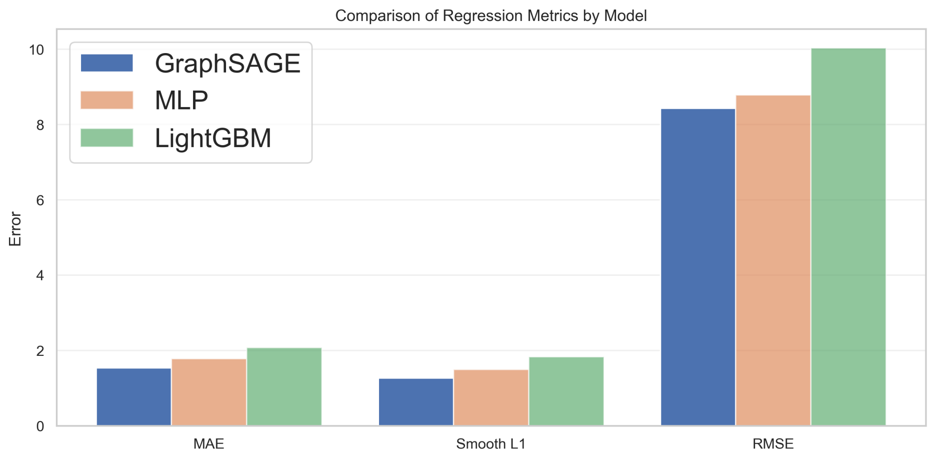


Figure 2: Comparasion of Graph Model: GraphSAGE and Tabular Model: MLP and LightGBM

To evaluate the effectiveness of graph-based modeling, we compare GraphSAGE with two strong tabular baselines: a Multi-Layer Perceptron (MLP) and LightGBM. Figure 2 summarizes the regression performance of the three models across MAE, Smooth L1 loss, and RMSE.

As shown in Figure 2, GraphSAGE consistently achieves the lowest error across all evaluation metrics. GraphSAGE obtains an MAE of 1.5366, compared to 1.7866 for MLP and 2.0771 for LightGBM. For Smooth L1 loss, GraphSAGE achieves 1.2719, outperforming MLP (1.4974) and LightGBM (1.8309). Similarly, GraphSAGE achieves the lowest RMSE of 8.4262, compared to 8.7856 for MLP and 10.0344 for LightGBM.

These results indicate that the graph-based approach provides consistently better prediction performance than tabular baselines.

3.3 Prediction versus ground truth for GraphSAGE, MLP, and LightGBM.

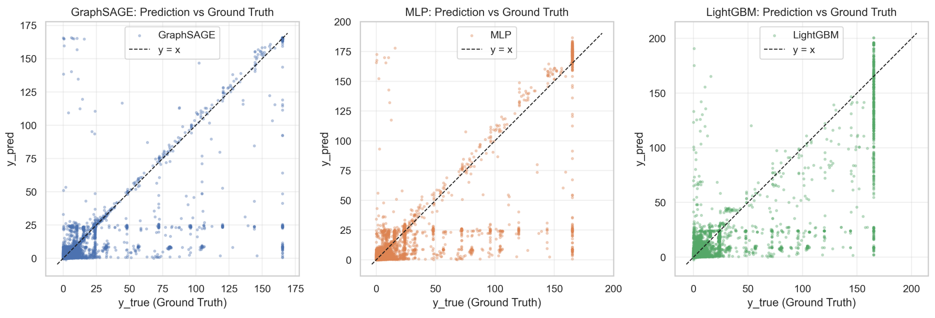


Figure 3: Prediction vs. ground truth comparison for GraphSAGE, MLP, and LightGBM.

The dashed line represents perfect prediction ($y = x$). GraphSAGE predictions are concentrated around the diagonal, indicating a closer alignment with the true completion times. In contrast, the tabular models (MLP and LightGBM) exhibit noticeable horizontal band patterns, where many samples are predicted with similar values despite different ground truth labels. This behavior reflects the piecewise nature of tabular models, which tend to produce discretized predictions based on limited feature interactions. By incorporating relational information through message passing, the graph model captures dependencies among neighboring tasks and produces smoother and more accurate predictions.

3.4 Hard-case Analysis

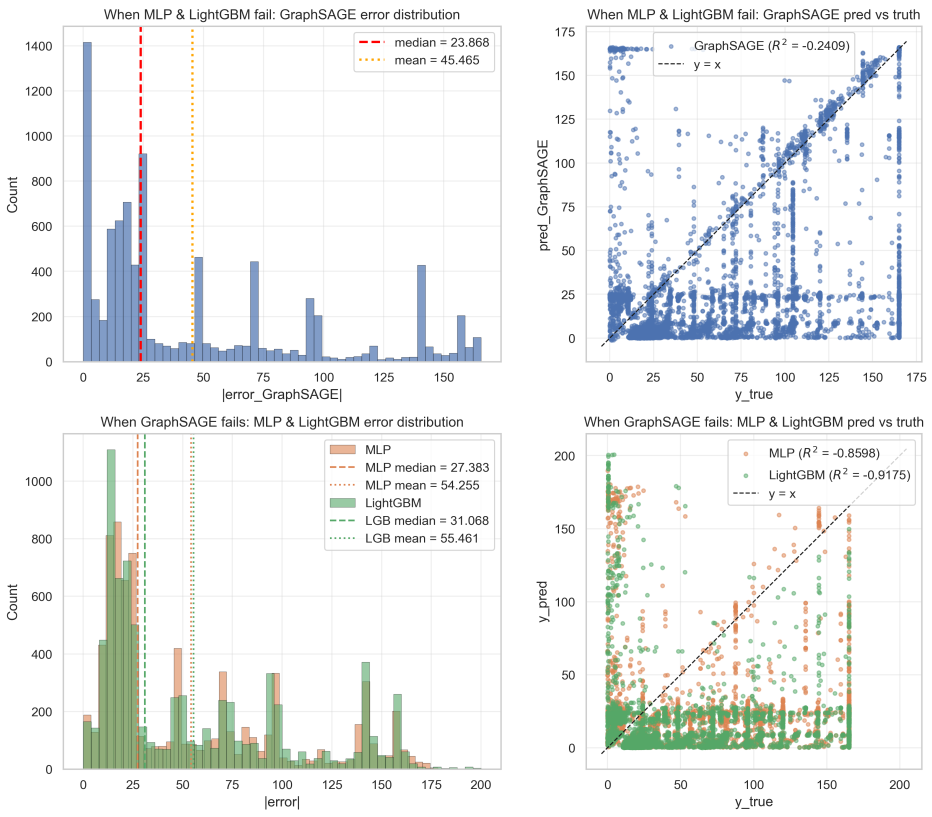


Figure 4: Hard-case analysis using an MAE threshold of 10.

While the overall metrics show that GraphSAGE achieves lower prediction errors than tabular models, we further investigate the underlying reason for this performance difference. In particular, we analyze cases where tabular models produce large prediction errors.

To identify difficult samples, we define a hard-case subset using a threshold on the Mean Absolute Error (MAE). Specifically, we select samples where both MLP and LightGBM produce an absolute error greater than 10. This allows us to focus on cases where tabular models fail to make accurate predictions.

Figure 4 presents the error distribution and prediction patterns on these hard cases. The top row shows the performance of GraphSAGE on samples where both MLP and LightGBM fail, while the bottom row shows the performance of the tabular models on samples where GraphSAGE fails.

As shown in Figure 4, GraphSAGE still produces meaningful predictions for many of these difficult samples, as indicated by the visible diagonal structure in the prediction scatter plot. In contrast, tabular models tend to collapse toward low prediction values, leading to large errors when the true completion time is high.

4 Discussion

In this section, we interpret the experimental findings and discuss what they reveal about the structure and dynamics of the scheduling system. While the results presented in the previous section demonstrate that graph based models achieve stronger predictive performance than traditional tabular approaches, the numerical metrics alone do not fully explain why this improvement occurs. To better understand the underlying reasons, we examine the behavior of different models and analyze the patterns shown in the figures and tables. By connecting these observations with the design of our heterogeneous graph representation, we aim to explain how relational information within the scheduling system contributes to improved prediction performance and provides additional insights into operational scheduling patterns.

4.1 Model Analysis

Our results show that representing scheduling data as a connected system leads to more effective prediction than treating each assignment as an isolated sample. Across all experiments, the graph based models, especially GraphSAGE, demonstrate stronger performance than the tabular baselines. This suggests that the relational structure in the scheduling system contains useful information that cannot be fully captured by standard tabular learning methods. Since our graph includes links among tasks, assignments, engineers, districts, and task related attributes, the model is able to incorporate operational context when making predictions, rather than relying only on local features of a single record. This is consistent with our original motivation of modeling SDG&E scheduling as an interconnected system instead of a set of independent observations.

Figure 1 and Table 2 together provide the clearest evidence for the advantage of GraphSAGE. In Figure 1, all three graph neural network architectures show stable convergence during training, which indicates that the constructed heterograph is suitable for graph based learning and that the optimization process is well behaved. However, the convergence patterns are not equally strong across models. GraphSAGE reaches a lower loss level than both RGCN and HGT, and Table 2 confirms this difference quantitatively, with a final Smooth L1 loss of 1.255 compared with 1.610 for HGT and 1.749 for RGCN. This result suggests that, for our scheduling graph, simple neighborhood aggregation is more effective than more complex relation specific or attention based designs. One possible reason is that the most useful signal in this dataset comes from local structural context, such as nearby assignments, shared engineers, and related task attributes. GraphSAGE is able to aggregate this information efficiently, while the more expressive heterogeneous models may introduce extra complexity that does not translate into better performance on this particular prediction task.

The comparison in Figure 2 further strengthens this interpretation by showing that GraphSAGE outperforms both MLP and LightGBM across every evaluation metric. GraphSAGE achieves the lowest MAE, Smooth L1 loss, and RMSE, indicating that its advantage is not limited to one particular error measure. This consistency is important because it suggests that the gain is robust and not the result of a single favorable metric. MLP performs better than LightGBM, which implies that even a flexible neural baseline can benefit from nonlinear feature learning, but both tabular methods still fall behind the graph model. The most likely explanation is that tabular models only use flattened or aggregated relational information, which inevitably removes part of the structure present in the original scheduling system. In contrast, GraphSAGE learns directly from the graph and can preserve how entities are connected. In practical terms, this means that the completion time of an assignment is influenced not only by its own attributes, but also by surrounding operational context such as who is assigned, what other tasks are nearby in the graph, and how workload is distributed across related entities.

Figure 3 further explains the metric differences through the prediction–ground truth relationship. While horizontal band patterns appear in the outputs of all models, they are noticeably weaker for GraphSAGE. These bands indicate that the model tends to produce similar predictions for many different inputs, a behavior commonly associated with regression models that partially collapse toward the mean of the target distribution. Compared with the tabular baselines, GraphSAGE exhibits fewer and less pronounced bands, and its predictions follow the diagonal trend more closely. This suggests that the graph model is less prone to mean-biased predictions and is better able to capture meaningful variation in completion times across different tasks. As a result, GraphSAGE preserves the overall structure of the target distribution more effectively, leading to improved predictive performance across both common and less frequent cases.

As shown in Figure 4, the analysis of hard cases further reveals the advantage of graph models on samples that are particularly difficult for tabular approaches. Rather than only comparing average performance, this analysis focuses on hard cases defined as samples where both MLP and LightGBM have an absolute error greater than 10. On this subset, GraphSAGE still retains a visible diagonal pattern in the prediction scatter plot, indicating that it continues to make informative predictions even when the tabular models fail. The error distribution also shows that GraphSAGE handles many of these difficult samples with lower error. In contrast, when GraphSAGE fails, the tabular models tend to produce predictions concentrated at relatively low values, especially when the true completion time is high. This suggests that the tabular baselines are more likely to collapse toward the center or lower end of the distribution when they encounter unusual or complex scheduling situations. A likely explanation is that hard cases depend more heavily on relational context, such as congestion around certain engineers, task interactions, or structural bottlenecks across the scheduling graph. Because GraphSAGE explicitly incorporates neighborhood information, it is better equipped to recover these patterns. This finding is particularly important in real operational settings, where difficult cases are often the most relevant to scheduling risk and delay management.

Taken together, the figures and tables suggest that the main benefit of the proposed framework is not simply a small improvement in average accuracy, but a stronger ability to model the system level dependencies that shape scheduling outcomes. The success of GraphSAGE indicates that many prediction errors in this domain are not purely feature level problems, but are instead linked to missing relational context. This also helps explain why the graph based approach is useful for the broader goal of this project. If the model can better capture structural patterns such as workload imbalance, delay propagation, and contextual similarity across operational entities, then it can provide more informative signals for downstream analysis and decision support. Overall, the results support our hypothesis that scheduling performance is best understood as an emergent property of the full operational system, not just the sum of isolated task level predictions.

4.2 Application

To demonstrate the practical use of our graph model, we developed a web application that integrates the data processing pipeline, model prediction, and visualization tools into a unified workflow. Our application begins with a data ingestion interface where users upload operational scheduling tables, including assignments, tasks, engineers, districts, department, task type, and task status datasets in CSV or JSON format. After validating and standardizing the input schema, the backend pipeline constructs a heterogeneous graph that captures relationships among scheduling entities. This graph representation is then used by our trained Graph Neural Network to model the relational dependencies within the scheduling system and predict task completion time and associated risk indicators.

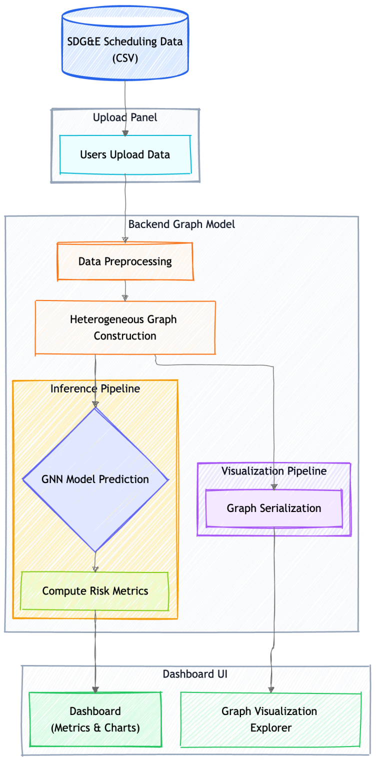
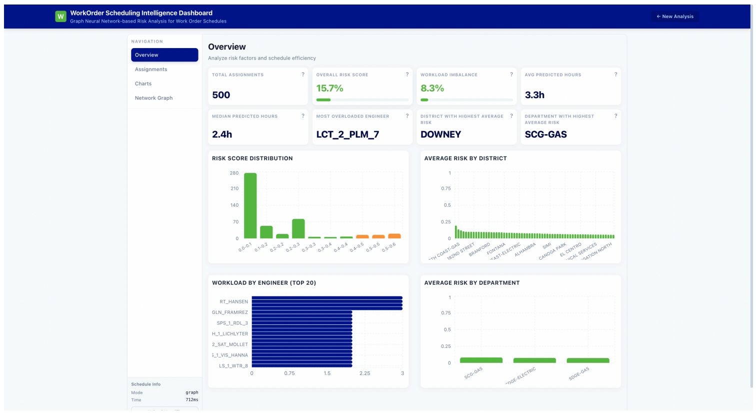


Figure 5: Workflow for web application, from data upload and graph construction to GNN inference and dashboard visualization.

The application presents model prediction outputs through an interactive dashboard that summarizes key operational metrics and charts. These metrics include average predicted completion hours, average workload imbalance scores, and the most overloaded engineers and districts, with charts highlighting risk score distribution, risk by district and department, and workload across engineers. By translating model predictions into interpretable indicators, the dashboard provides a high-level overview of scheduling performance and efficiency and help users detect potential bottlenecks or scheduling insights that may not be immediately visible from raw scheduling tables.

In addition to the dashboard, our application includes a graph exploration interface that visualizes the heterogeneous graph. Nodes represent entities such as tasks, assignments, engineers, districts, and departments, while edges capture their operational relationships, such as indicating which assignments were assigned to which engineer in what districts. Users can interactively inspect nodes, view predicted values, and explore structural connections within the scheduling system. This visualization provides an intuitive way to understand relational dependencies and offers insights into how scheduling patterns propagate across different components of the operational workflow.


WorkOrder Scheduling Intelligence Dashboard
Graph Neural Network-based Risk Analysis for Work Order Schedules
← New Analysis
NAVIGATION
Overview
Assignments
Charts
Network Graph
RISK SCORE DISTRIBUTION
AVERAGE RISK BY DISTRICT
WORKLOAD BY ENGINEER (TOP 20)
AVERAGE RISK BY DEPARTMENT

Overview

Analyze risk factors and schedule efficiency

TOTAL ASSIGNMENTS 500

OVERALL RISK SCORE 15.7%

WORKLOAD IMBALANCE 8.3%

AVG PREDICTED HOURS 3.3h

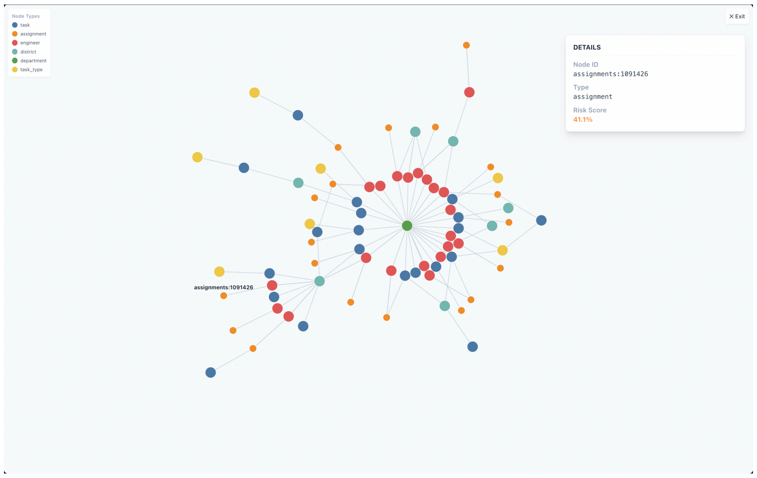
MEDIAN PREDICTED HOURS 2.4h

MOST OVERLOADED ENGINEER LCT_2_PLM_7

DISTRICT WITH HIGHEST AVERAGE RISK DOWNEY

DEPARTMENT WITH HIGHEST AVERAGE RISK SCG-GAS

Figure 6: Interactive dashboard summarizing scheduling risk metrics, workload imbalance, and district-level performance.



DETAILS

Node ID
assignments:1091426

Type
assignment

Risk Score
41.1%

Figure 7: Interactive visualization of the heterogeneous graph with node relationships and predicted risk values.

5 Conclusion

The results indicate that scheduling outcomes are influenced not only by individual task characteristics, but also by relational structure. This highlights the value of graph-based analysis as a practical direction for supporting utility planning, improving resource allocation, and reducing operational inefficiencies. The framework also demonstrates potential as a foundation for decision-support tools that can help utility companies make more informed scheduling decisions and improve overall operational performance.

5.1 Limitations

The current graph structure has limited representation of engineer-to-engineer relationships. Because of data size and distribution constraints, direct records of collaboration between engineers are sparse, making it difficult to fully model team-level interactions. As a result, some operational dependencies related to crew coordination may not yet be fully captured. In addition, the graph mainly captures static historical relationships and does not fully represent dynamic scheduling changes such as delays or reassignment. Some important operational factors, including weather conditions and emergency priority changes, are also not included in the current dataset.

5.2 Future Work

There are several directions for extending this work. First, the current framework focuses on evaluating historical schedules; future work could explore integrating the predictive model into an automated scheduling system for real-time schedule evaluation, in order to dynamically adjust schedule generation. Second, the graph representation could be expanded to incorporate additional contextual information, such as travel time between job sites, real-time operational constraints, or environmental factors that may influence task completion. Finally, further research could investigate more advanced graph learning approaches and larger-scale datasets to improve predictive accuracy and robustness. These extensions would help move the system from an analytical evaluation tool toward a more comprehensive decision-support platform for utility scheduling operations.

References

[1] William L. Hamilton, Rex Ying, and Jure Leskovec. “Inductive Representation Learning on Large Graphs”. In: 2018. arXiv: 1706.02216 [cs.SI]. URL: https://arxiv.org/abs/1706.02216.

[2] Ziniu Hu et al. “Heterogeneous Graph Transformer”. In: 2020. arXiv: 2003.01332 [cs.LG]. URL: https://arxiv.org/abs/2003.01332.

[3] Michael Schlichtkrull et al. “Modeling Relational Data with Graph Convolutional Networks”. In: 2017. arXiv: 1703.06103 [stat.ML]. URL: https://arxiv.org/abs/1703.06103.