Problem Statement

Developing a Graph Neural Network (GNN)-based anomaly detection framework for adaptive robotic navigation, focusing on identifying adversarial perturbations (e.g., GPS spoofing, sensor tampering) and operational faults (e.g., wheel slippage, motor overheating) in real-time within dynamic environments. Example scenarios include warehouse robots misled by adversarial floor markings or delivery drones encountering spoofed geolocation data during urban flights.

Hypotheses

1. GNN-based anomaly detection methods will achieve higher precision and recall in identifying adversarial inputs (e.g., image or LIDAR-based perturbations that mislead path planning) and irregular trajectories (e.g., unexpected detours or oscillations in path) compared to traditional autoencoders and statistical baselines.
2. Incorporating GNNs into adaptive robot controllers will improve situational awareness and robustness, such as faster re-routing or task reallocation, when anomalies are detected in multi-agent warehouse robots or autonomous service units.

Research Questions

1. To what extent can GNNs detect real-time anomalies in robotic agents navigating dynamic, uncertain environments—such as an autonomous vehicle detecting inconsistent pedestrian movement or a cleaning robot encountering an unplanned obstacle?
2. How does the integration of GNN-based anomaly detection affect trustworthiness (e.g., operator confidence score, intervention rate) and failure recovery time in digital twin simulations of complex systems like hospital delivery robots or manufacturing co-bots?
3. What are the trade-offs in computational cost, model latency, and deployment feasibility when applying GNNs for online anomaly detection in edge-deployed robots (e.g., JetBot, TurtleBot3) in time-sensitive environments?

Literature Review 1

Title: Graph Anomaly Detection With Graph Neural Networks: Current Status and Challenges

Authors: Hwan Kim, Byung Suk Lee, Won-yong Shin and Sungsu Lim

Publication Year: 2022

Key Ideas

The focus the paper went into is that anomalies in graph-structured data are hard to detect using traditional ML because of complex node and edge relationships. It discusses how Graph Neural Networks are a new approach that can learn from both the features and structure of the graph. It categorizes the anomalies into static graphs, which are graphs that don't change over time, and dynamic graph anomalies that come from evolving graphs. GNNs are able to detect different types of anomalies like node-level, edge-level, and graph-level.

Contributions

The paper provided a detailed review of GNN-based anomaly detection methods that group GNN frameworks by anomaly types like static vs. dynamic and node vs. edge. It discussed current methods like attention mechanisms (GAT) that focus on key connections, self-supervised learning to reduce dependency on labeled data, and using reconstruction loss and embedding distance to spot anomalies.

Limitations

While GNNs are considered useful, there is limited research on them, which makes outputs harder to interpret. There is also a data imbalance, making anomalies rare when training models. For real-world use, there are few approaches that work well for few-shot or zero-shot detection. There is also limited discussion on how GNNs perform in noisy or adversarial environments, like when a robot is under attack.

Extend/Improvement

For my paper, I would explore explainable GNNs while also finding methods for few-shot development to use in real-world environments. I will use a digital twin as I integrate with dynamic environments to expand the data.

Literature Review 2

Title: Challenges and Opportunities in Deep Reinforcement Learning With Graph Neural Networks: A Comprehensive Review of Algorithms and Applications

Authors: Sai Munikoti, Deepesh Agarwal, Laya Das, Mahantesh Halappanavar and Balasubramaniam Natarajan

Publication Year: 2023

Key Ideas

The paper focused on a hybrid framework using Deep Reinforcement Learning and Graph Neural Networks that can potentially help solve tasks that involve sequential decision-making in graph-structured environments. It discusses the advantages of their relationships, with GNN modeling relationships between agents and nodes and DRL handling long-term policy learning under uncertainty. This will allow robots to learn from relational state spaces, adapt to dynamic environments, and handle real-world problems such as traffic routing.

Contributions

The paper categorizes hybrid DRL-GNN approaches into two main types. The symbiotic framework focuses on how DRL helps with GNN structure, like adversarial node injection using Q-learning. The application-specific approach is where the hybrid can be applied to real-world solutions like COVID-19 contact tracing. It also highlighted that this hybrid approach outperformed traditional models and provided a detailed analysis of model goals.

Limitations

The limitations focus on the transferability of learned models to new dynamic environments, which is still new. The explainability is limited, so the research on this hybrid model is not studied as much. As well, the computational cost is expensive to deploy in real-time robotic systems.

Extend/Improvement

For the project, I plan to explore the DRL-GNN models for anomaly detection. I'll focus on designing a more efficient hybrid model pipeline for real-time use on edge devices in cybersecurity settings and improving adaptation to dynamic environments.

Proposed Approach

My project focuses on detecting and mitigating adversarial anomalies in robotic agents while navigating post-disaster environments. The main idea is to use graph based learning to analyze relationships between agents, zones, tasks, and anomalies. By combining synthetic graph generation, GNN-based anomaly detection and quantum optimization helps support intelligent trust aware decision making.

Models

- Graph Neural Networks (GNNs)
    - Learning patterns in agent-task-zone-anomaly relationships
- Reinforcement learning (Q-Learning)
    - Understanding agent behavior under constraints (FrozenLake simulation)
- Quantum Optimization (QUBO/ Qiskit)
    - Task assignment under adversarial or uncertain conditions

Libraries/Tools

- NetworX
  - Graph generation and visualization
- Neo4j
  - Visual and query-based exploration of knowledge graph
- Qiskit and D0cplex
  - Quantum optimization and QUBO modeling
- Gymnasium
  - Reinforcement learning simulation

Cyber Security Context

- Trust modeling
  - Evaluating id an agents behavior is anomalous based on sensor or GPS spoofing
- Incident Predictions
  - Use GNNs to detect propagating threats or anomalies in agent-task relationships
- Secure Task Allocation
  - Using quantum optimization to assign robots to critical tasks in high-risk zones and resource constraints

Workflow

Synthetic Graph Generator

↓

Nodes: Agents, Tasks, Zones, Sensors, Anomalies

↓

GNN Anomaly Detection

↓

Reinforcement Learning (Q-Learning)

↓

QUBO Formulation via Qiskit

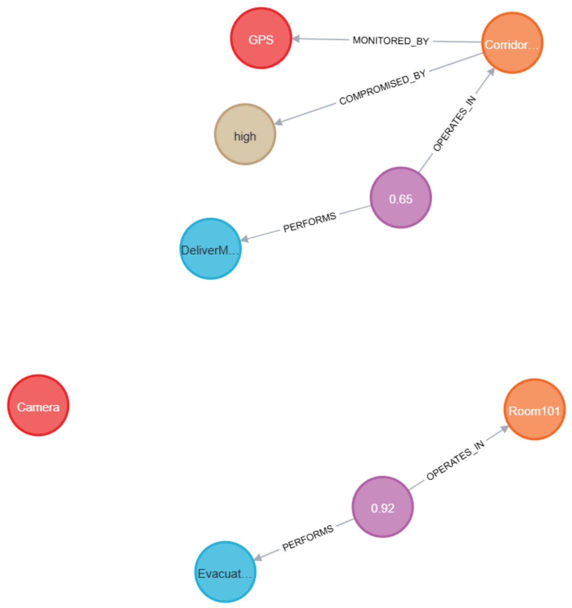
↓

Quantum Optimization

Preliminary Results

Graph Modeling

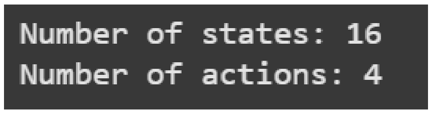
- Generated a synthetic hospital rescue scenario graph with 9,800 nodes and 6,800 edges using NetworkX
- The knowledge graph presented agent-task-zone-anomaly relationships modeled in Neo4j



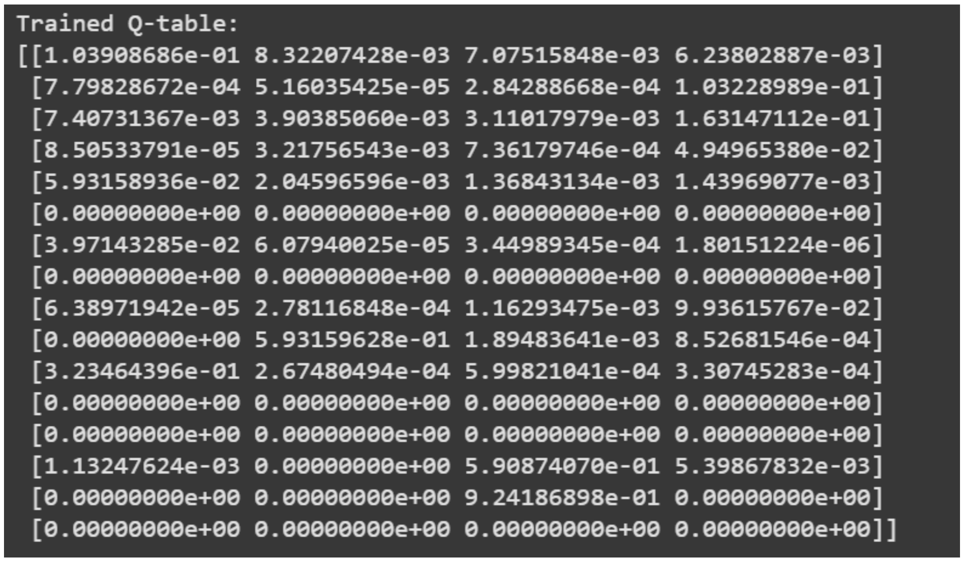
- The challenge is visual clutter from node density, so it was better to extract subgraphs

Reinforcement Learning with Gymnasium

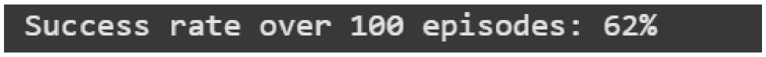
- I trained a Q-learning agent in FrizwnLake-v1 using Gymnasium
- The output of the trained Q-table had a success rate of ~62% over 100 episodes



```
Number of states: 16
Number of actions: 4
```



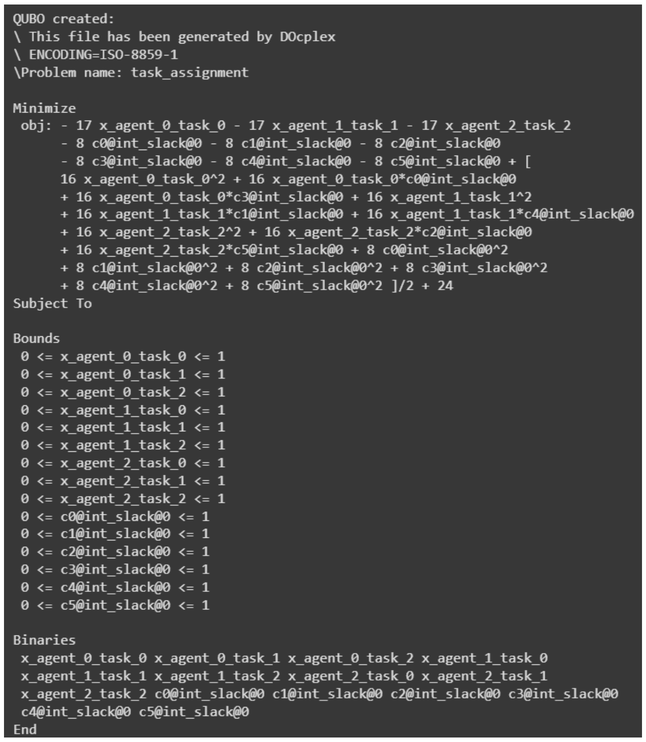
```
Trained Q-table:
[[1.03908686e-01 8.32207428e-03 7.07515848e-03 6.23802887e-03]
 [7.79828672e-04 5.16035425e-05 2.84288668e-04 1.03228989e-01]
 [7.40731367e-03 3.90385060e-03 3.11017979e-03 1.63147112e-01]
 [8.50533791e-05 3.21756543e-03 7.36179746e-04 4.94965380e-02]
 [5.93158936e-02 2.04596596e-03 1.36843134e-03 1.43969077e-03]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [3.97143285e-02 6.07940025e-05 3.44989345e-04 1.80151224e-06]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [6.38971942e-05 2.78116848e-04 1.16293475e-03 9.93615767e-02]
 [0.00000000e+00 5.93159628e-01 1.89483641e-03 8.52681546e-04]
 [3.23464396e-01 2.67480494e-04 5.99821041e-04 3.30745283e-04]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]
 [1.13247624e-03 0.00000000e+00 5.90874070e-01 5.39867832e-03]
 [0.00000000e+00 0.00000000e+00 9.24186898e-01 0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00]]
```



```
Success rate over 100 episodes: 62%
```

- The challenge I had is balancing exploration and exploitation in a simple but theoretical environment

Quantum Optimization

- I extracted a subgraph with 3 agents and 3 tasks and encoded task assignment into QUBO format using Qi Kit and D0c Plex
- The QUBO outputted a model with binary variable and slack ters enforcing onto to one tasks 

```
QUBO created:
\ This file has been generated by DOcplex
\ ENCODING=ISO-8859-1
\Problem name: task_assignment

Minimize
 obj: - 17 x_agent_0_task_0 - 17 x_agent_1_task_1 - 17 x_agent_2_task_2
      - 8 c0@int_slack@0 - 8 c1@int_slack@0 - 8 c2@int_slack@0
      - 8 c3@int_slack@0 - 8 c4@int_slack@0 - 8 c5@int_slack@0 + [
      16 x_agent_0_task_0^2 + 16 x_agent_0_task_0*c0@int_slack@0
      + 16 x_agent_0_task_0*c3@int_slack@0 + 16 x_agent_1_task_1^2
      + 16 x_agent_1_task_1*c1@int_slack@0 + 16 x_agent_1_task_1*c4@int_slack@0
      + 16 x_agent_2_task_2^2 + 16 x_agent_2_task_2*c2@int_slack@0
      + 16 x_agent_2_task_2*c5@int_slack@0 + 8 c0@int_slack@0^2
      + 8 c1@int_slack@0^2 + 8 c2@int_slack@0^2 + 8 c3@int_slack@0^2
      + 8 c4@int_slack@0^2 + 8 c5@int_slack@0^2 ]/2 + 24
Subject To

Bounds
 0 <= x_agent_0_task_0 <= 1
 0 <= x_agent_0_task_1 <= 1
 0 <= x_agent_0_task_2 <= 1
 0 <= x_agent_1_task_0 <= 1
 0 <= x_agent_1_task_1 <= 1
 0 <= x_agent_1_task_2 <= 1
 0 <= x_agent_2_task_0 <= 1
 0 <= x_agent_2_task_1 <= 1
 0 <= x_agent_2_task_2 <= 1
 0 <= c0@int_slack@0 <= 1
 0 <= c1@int_slack@0 <= 1
 0 <= c2@int_slack@0 <= 1
 0 <= c3@int_slack@0 <= 1
 0 <= c4@int_slack@0 <= 1
 0 <= c5@int_slack@0 <= 1

Binaries
 x_agent_0_task_0 x_agent_0_task_1 x_agent_0_task_2 x_agent_1_task_0
 x_agent_1_task_1 x_agent_1_task_2 x_agent_2_task_0 x_agent_2_task_1
 x_agent_2_task_2 c0@int_slack@0 c1@int_slack@0 c2@int_slack@0 c3@int_slack@0
 c4@int_slack@0 c5@int_slack@0
End
```

- The challenge I face is I could execute QAOA fully due to compute limitations, but was able to formulate a complete and valid output for simulation

Experimental Design and Expectations

Inputs

- Synthetic knowledge graph
  - Generated using NetworkX and Neo4j with 9,800 nodes and 6,800 edges representing agents, zones, tasks, sensors, and anomalies
- Subgraphs
  - Smaller task specific subgraphs for optimization and simulation
- Sensor Logs
  - Simulated GPS spoofing, sensor failure, or latency drift
- Adversarial inputs

    - Injected anomalous behavior in graph for testing

Evaluation Metrics

- Graph learning tasks
    - Focused on accuracy, precision, recall, and F1-score for node classification
        - Anomaly vs. normal
- Quantum
    - Priority/trust score from task-agent assignment
    - Number of assignment violations
- RL tasks (FrozenLake)
    - Success rate of successful episode
    - Steps per episode, indicating policies efficiency

Baselines

- Assign agents to tasks manually based on priority or zone proximity
- Use linear programming or brute force to solve task assignment
- Using FrozenLake Q-table as a basic decision making benchmark
- Using GrpahSAGE, GAT, or simple MLP for anomaly classification

Expected Results

- QUBO is expected to produce optimal task agent mappings under constraints like limited trust or bandwidth
- GNN models should outperform simple threshold based system under adversarial inputs
- RL behavior is expected to increase success rate over time, agents learning how to avoid unsafe zones

Limitations

- Large-scale GNNs might require GPU acceleration
- Quantum solver may be limited by available qubits and precision

Reproduc a SOTA Model

Model: **DOMINANT** – Deep Anomaly Detection on Attributed Networks
Source: *Graph Anomaly Detection With GNNs: Current Status and Challenges*

Implementation Details

- Uses graph autoencoder framework, combining node features and structure using GCN layers
- Used dataset from a synthetic graph generated from task 1 with labeled anomalies
- Encoder uses 2 GCN layers, with the decoder reconstructing adjacency and attributes

Initial Evaluation

- The model successfully assignment higher reconstruction loss to anomalous nodes
- Only evaluated a small subgraph (~300 nodes)
- Results were subjective but showed correct behavior, with anomalous zones flagged based on spoofed sensor values

Comparison

- Original DOMINANT model showed high AUC scores on real world datasets like Cora
- My model on synthetic graph were menial but the score pattern was consistent with expectations
- Adaption validated that even simplified GNN autoencoders can detect synthetic cyber anomalies

References

Kim, Hwan, Byung Suk Lee, Won-yong Shin, and Sungsu Lim. "Graph Anomaly Detection with Graph Neural Networks: Current Status and Challenges." IEEE Access, vol. 10, 2022, pp. 111820–29, https://doi.org/10.1109/access.2022.3211306.

Munikoti, Sai, Deepesh Agarwal, Laya Das, Mahantesh Halappanavar, and Balasubramaniam Natarajan. "Challenges and Opportunities in Deep Reinforcement Learning with Graph Neural Networks: A Comprehensive Review of Algorithms and Applications." IEEE Transactions on Neural Networks and Learning Systems, Institute of Electrical and Electronics Engineers, Jan. 2023, pp. 1–21, https://doi.org/10.1109/tnnls.2023.3283523.