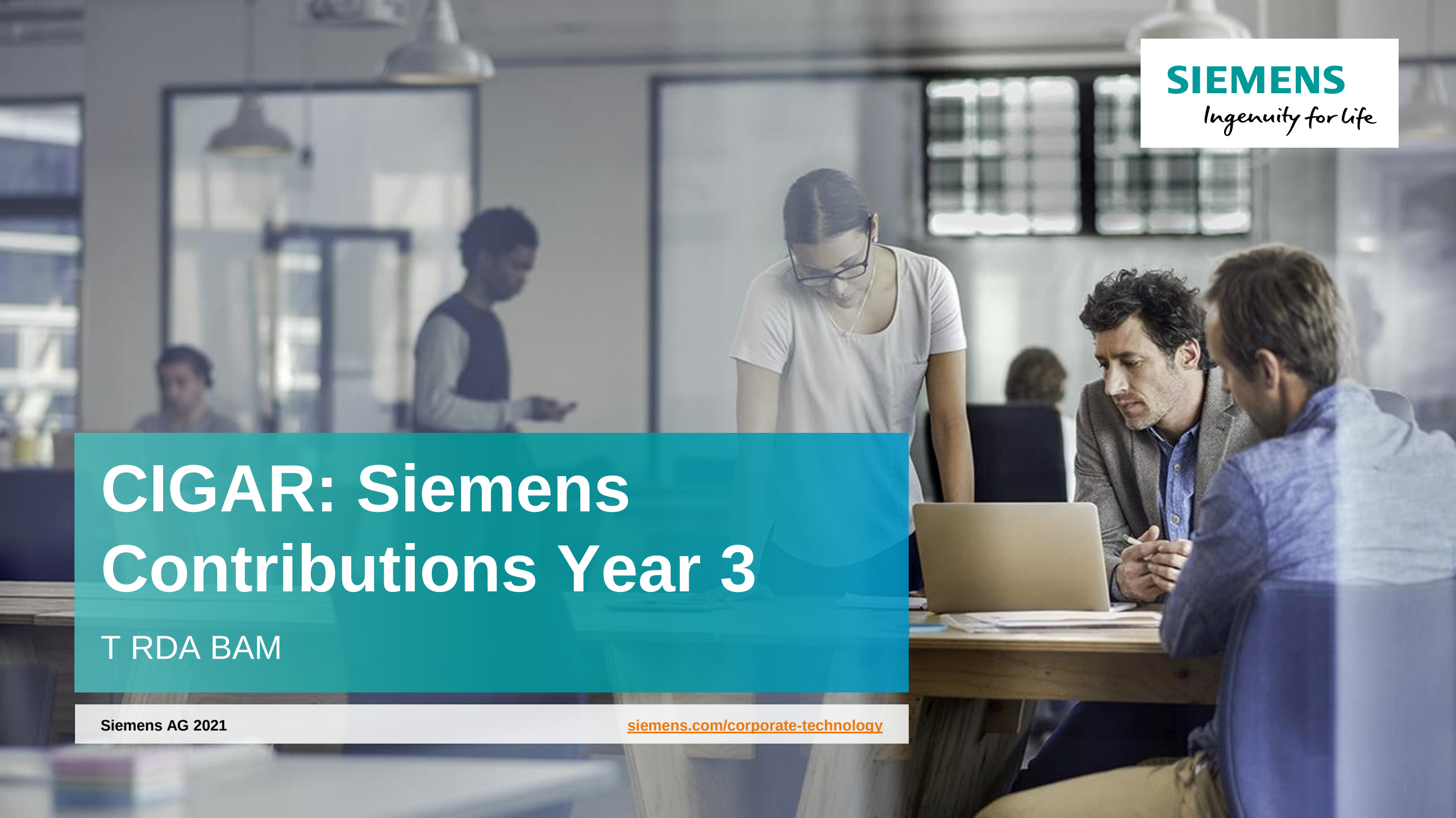


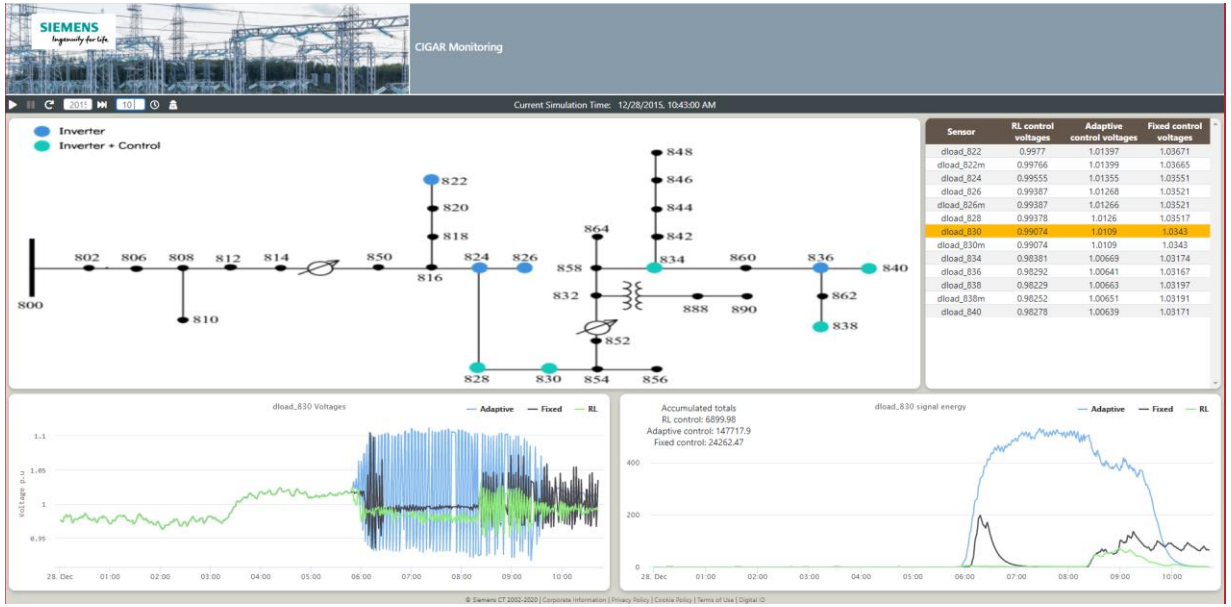
CIGAR: Siemens Contributions Year 3

T RDA BAM

Background

- Siemens Contributions to CIGAR:
  - Year 1: RL algorithm down-selection and initial implementation
  - Year 2: Support LBNL in development of PyCIGAR and implementation of RL algorithms
  - Year 3: Exploration of advanced RL approaches, esp. Graph Convolutional Networks (GCN) for RL
    - Discussed in the following slides

- Independent Technology-to-market Effort:
  - Development of a CIGAR Demonstrator for internal presentation to Siemens Business Units and customers

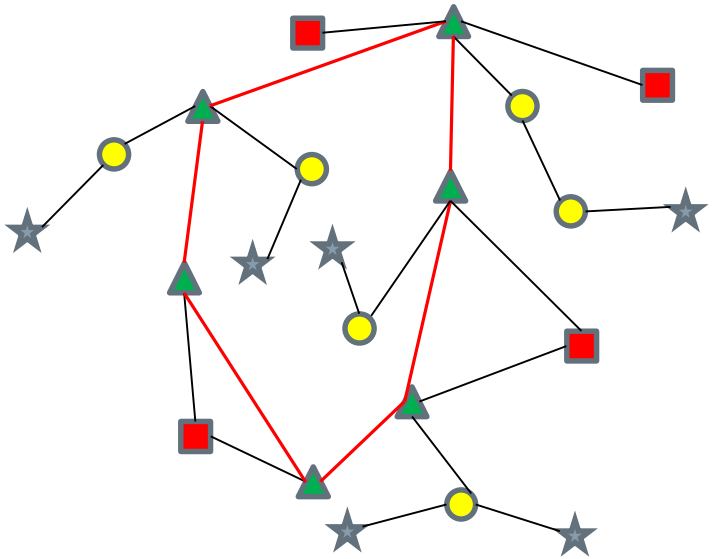


Motivation

SIEMENS
Ingenuity for life

Using Graph Convolutional Networks and Reinforcement Learning: control policies in complex systems

- The system is a combination of interconnected sub-systems, represented as a graph (network)
- Nodes – sub-systems to be controlled with a set of features (observations), which may be unique for a given node/node type
- Edges/Arcs - connections between nodes (potentially with their own associated feature sets)
- Changing topology of the graph
  - New or deleted nodes/edges both during training/actual control
  - The connections can be physical/logical/temporal/functional



Process control that addresses the following challenges:

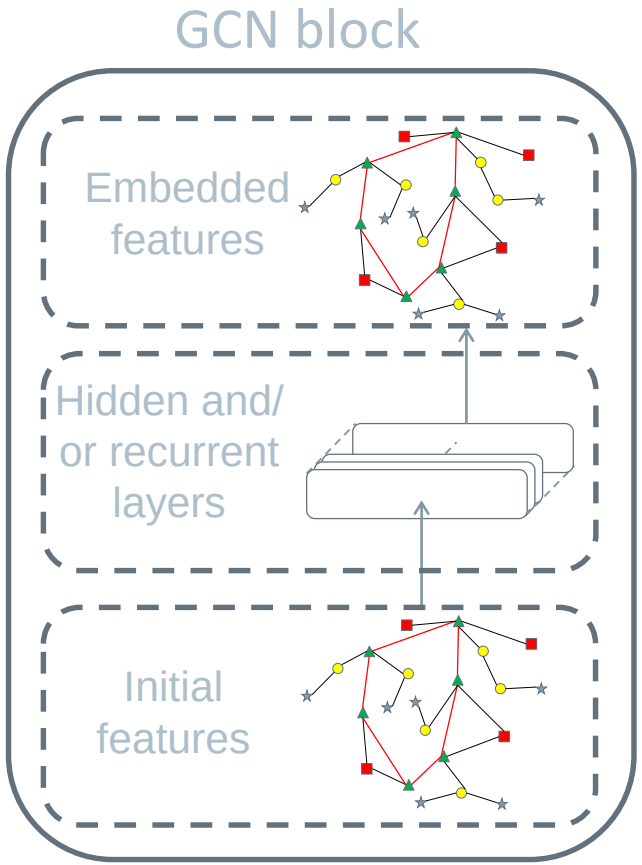
- **Balance between global/local control**
- **Changing topology**
- **Computational scalability**
- **Transfer learning from one network to another**

Existing Applications

SIEMENS
Ingenuity for life

- Applications of GCNs and RL:
  - molecular graph generation [3]
  - **autonomous driving** [4]
  - combinatorial optimization [5, 6]
  - **traffic signal control** [7]
  - **multi-agent cooperation** [8, 9]
  - ***Optimal power grid control***

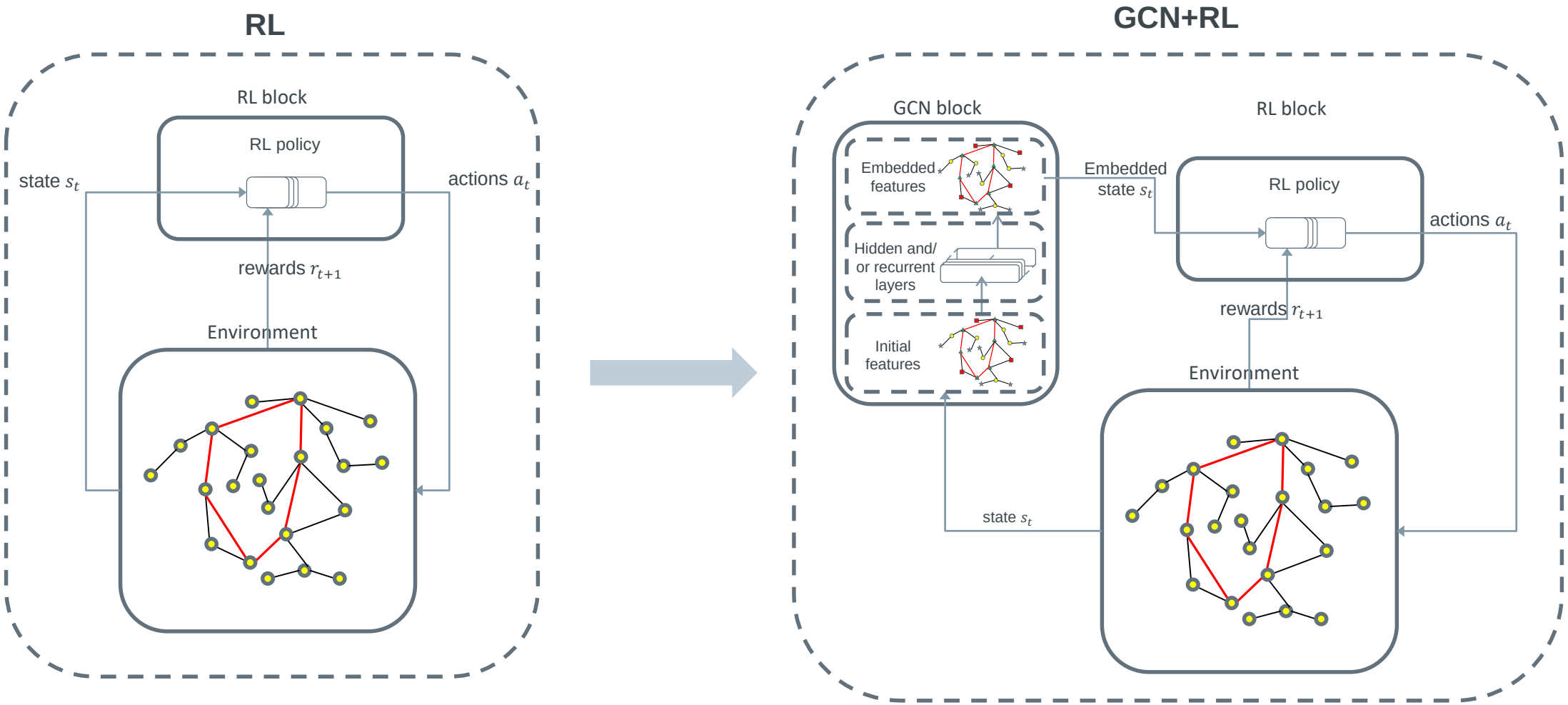
Graph Convolutional Networks



- An adaptation of Artificial Neural Networks to graphs
- Simplest architecture: a sequence of aggregation (convolution) and fully connected layers
- Final output is features for each node in the embedded space
- Weights can be leaned in several setups: autoencoder, node's feature prediction, edge/node prediction
- Graphs can be both static or dynamic

GCN+RL: simplest case

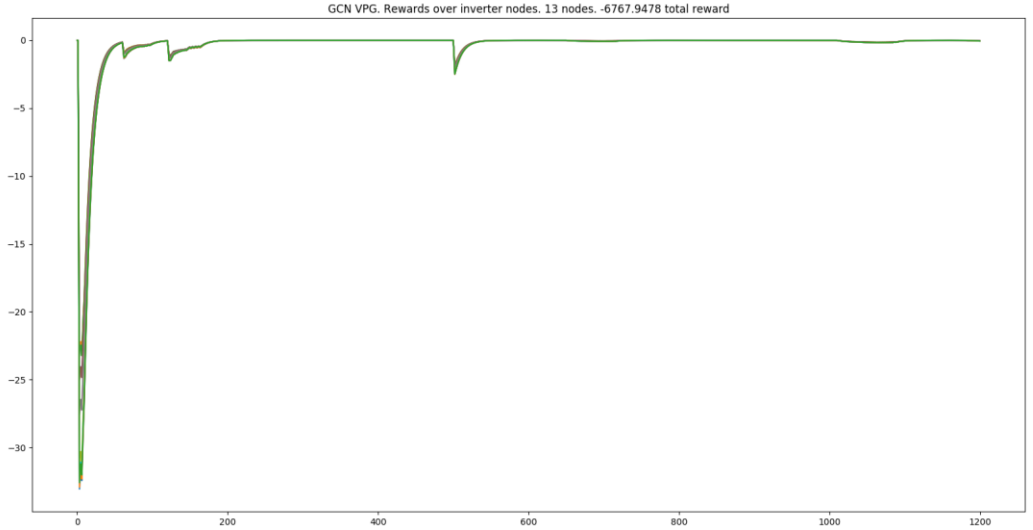
- Simply add several GCN layers before RL layers, the rest is the same as in “conventional” RL

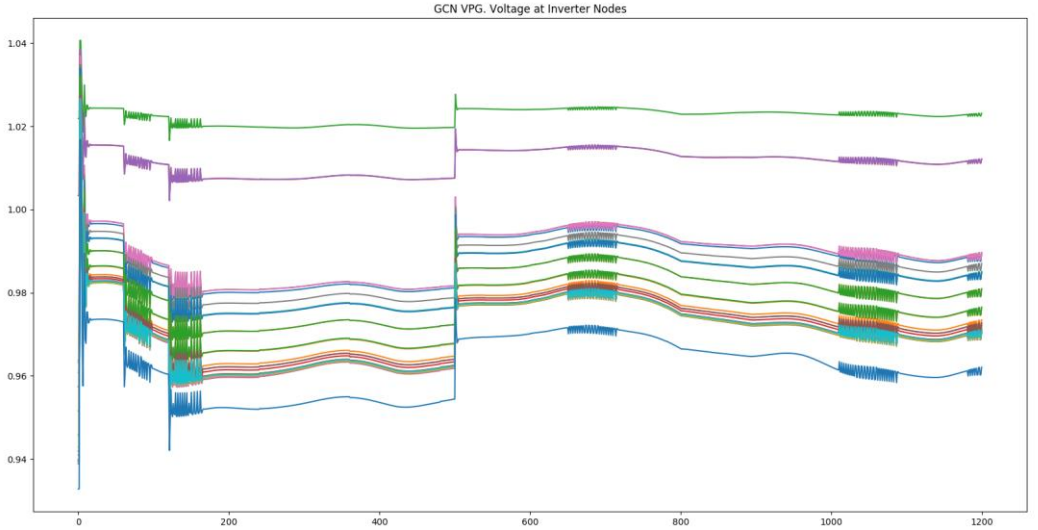


GCN+VPG on seconds with hack

SIEMENS
Ingenuity for life

```
'start_time’: 42900, 'end_time': 44100,'hack_start': 500,
inverter_nodes_ids=tuple(range(5, 18)),
ai_nodes_ids=(4, 5, 6, 8, 10, 11, 12),
attack_settings=(1.0, 1.001, 1.001, 1.01),
percent_hacked=np.array((0.5, .5, 0.5, 0, .5, .5, .5, 0, .5, 0, 0, .5, 0.5)
```

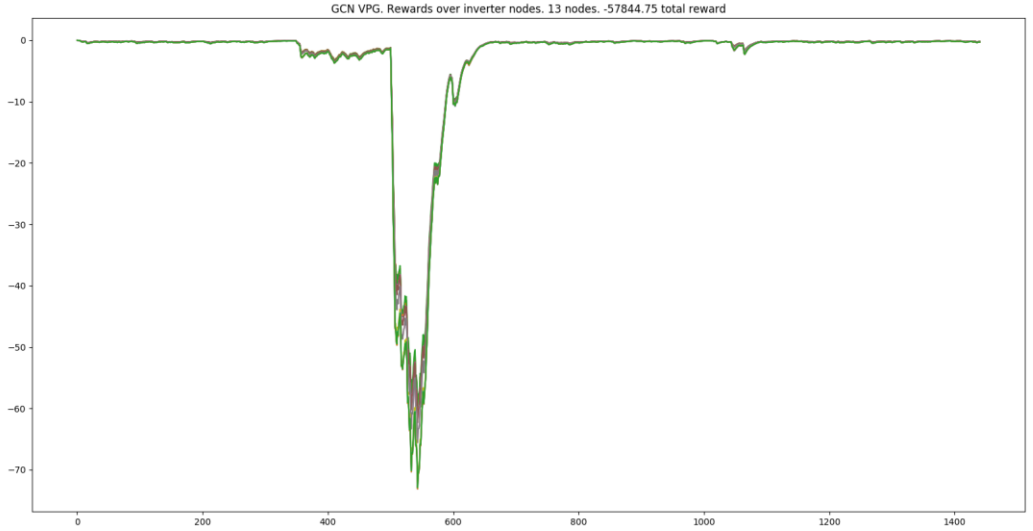


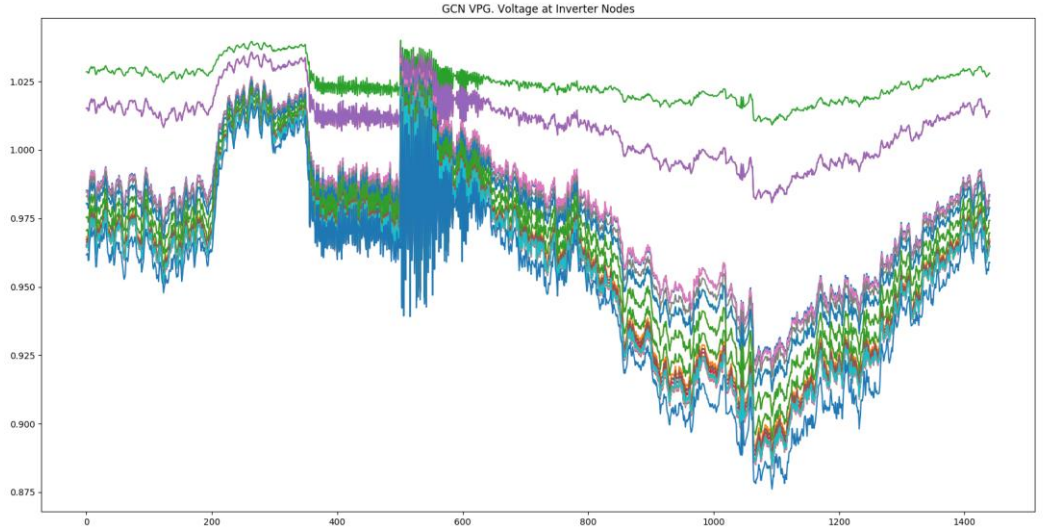


GCN+VPG on minutes with hack

SIEMENS
Ingenuity for life

```
'start_time’: 0, 'end_time': 1440,'hack_start': 500,
inverter_nodes_ids=tuple(range(5, 18)),
ai_nodes_ids=(4, 5, 6, 8, 10, 11, 12),
attack_settings=(1.0, 1.001, 1.001, 1.01),
percent_hacked=np.array((0.5, .5, 0.5, 0, .5, .5, .5, 0, .5, 0, 0, .5, 0.5)
```





GCN+RL vs RL vs Fixed vs Adaptive control

SIEMENS
Ingenuity for life

- All GCN+RL and RL algorithms significantly outperformed Adaptive and Fixed Control
- PPO outperformed all
- GCN+VPG outperformed VPG (and DDPG): next best
- GCN+PPO was difficult to tune:
  - it requires much longer time to converge

Algorithm	Rewards On Seconds	Rewards On Minutes
GCN+VPG	-1331.02	-57844.75
VPG	-2204.63	-61044.66
GCN+PPO	-1938.02	-76310.87
PPO	**-1093.40**	**-41014.69**
DDPG	-1377.09	-61508.12
Fixed Control	-37946.086	-289604.8
Adaptive Control	-29518.09	-115710.36

GCN+RL Plan

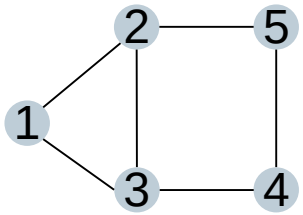
SIEMENS
Ingenuity for life

1. Given a power grid, add several GCN layers to existing RL architectures, learn from the scratch GCN layers and policies simultaneously.
2. Explore additional parameter options, adjacency matrix definitions, etc.
3. Given a power grid, pretrain GCN layers on simulated data (Fixed/Adaptive Inverters) using autoencoder or future node feature prediction (recurrent layers), learn RL layers and possibly continue learning the GCN layers with much smaller learning rate
4. Include grouping of nodes according to graph topology or functional/correlation analysis. Having an independent RL agent for each group, repeat experiments 1 and 2, where GCN layers will be shared across all RL groups
5. Repeat experiments 2 and 3 when GCN layers are pretrained across several power grids: same node features but varying grid topologies. Assess transferability of trained architectures from one power grid to another

References

1. R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. (Second Edition)., MIT Press, 2018.
2. J. Zhou, G. Cui, Z. Zhang, Z. Liu, L. Wang, C. Li and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434,* 2018.
3. J. You, B. Liu, Z. Ying, V. Pande and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Advances in neural information processing systems*, 2018, pp. 6410-6421.
4. M. Huegle, G. Kalweit, M. Werling and J. Boedecker, "Dynamic Interaction-Aware Scene Understanding for Reinforcement," *arXiv preprint arXiv:1909.13582,* 2019.
5. Z. Li, Q. Chen and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Advances in Neural Information Processing Systems*, 2018, pp. 539-548.
6. A. Mittal, A. Dhawan, S. Manchanda, S. Medya, S. Ranu and A. Singh, "Learning heuristics over large graphs via deep reinforcement learning," *arXiv preprint arXiv:1903.03332,* 2019.
7. T. Nishi, K. Otaki, K. Hayakawa and T. Yoshimura, "Traffic Signal Control Based on Reinforcement Learning with Graph Convolutional Neural Nets," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, 2018.
8. J. Jiang, C. Dun, T. Huang and Z. Lu, "Graph Convolutional Reinforcement Learning," in *International Conference on Learning Representations*, 2020.
9. A. Khan, E. Tolstaya, A. Ribeiro and V. Kumar, "Graph Policy Gradients for Large Scale Robot Control," *arXiv preprint arXiv:1907.03822,* 2019.
10. P. Goyal, S. R. Chhetri and A. Canedo, "dyngraph2vec: Capturing network dynamics using dynamic graph representation learning," *Knowledge-Based Systems,* vol. 187, p. 104816, 2020.

Graph Convolutional Layers



adjacency matrix

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Normalized

$$\hat{A} = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 0 & 1/4 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 1/3 & 0 & 1/3 & 1/3 \end{bmatrix}$$

Normalized with identity

$$\hat{A} = \begin{bmatrix} 1 & 1/2 & 1/2 & 0 & 0 \\ 1/3 & 1 & 1/3 & 0 & 1/3 \\ 1/3 & 1/3 & 1 & 1/3 & 0 \\ 0 & 0 & 1/2 & 1 & 1/2 \\ 0 & 1/2 & 0 & 1/2 & 1 \end{bmatrix}$$

Symmetric Normalized

$$\hat{A} = D^{-0.5}(A + I)D^{-0.5}$$

- Given a graph $G = (V, E)$ with n nodes and a feature matrix $X \in \mathbb{R}^{n \times d}$, for each node we want to aggregate information available in its neighborhood
- If A is adjacency matrix, $\hat{A}$ can be:
  - Normalized adjacency matrix
  - Symmetric normalized adjacency matrix
  - Normalized adjacency matrix with identity preserving

- $H^{(0)} = X$
- From level k to level $k + 1$:

$$H^{(k+1)} = q^{(k)}\left(\hat{A}H^{(k)}W^{(k)} + bias^{(k)}\right)$$

- $W^{(k)}$ and $bias^{(k)}$ are trainable.

GCN+VPG Tuning

SIEMENS
Ingenuity for life

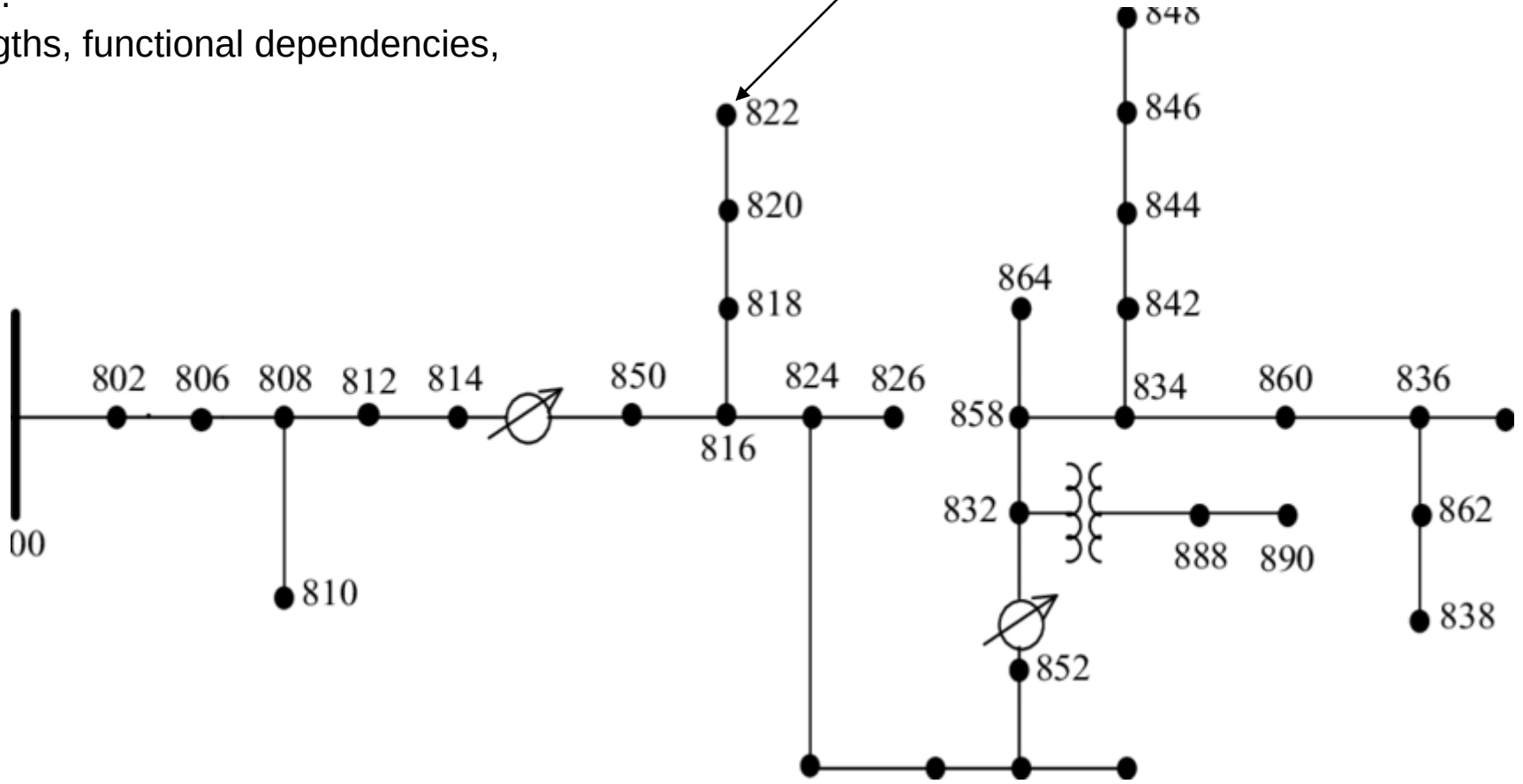
```
tune_gcn_vpg_args = {
    'gcn_layers': [1, 2],
    'neighbors': [3, 4],
    'adjmat_normalization': ["symmetric_normalized_matrix",
                             "normalized_matrix",
                             "normalized_with_loops"],
    'buffer_capacity': 4 * 1440,
    'gamma': 0.99,
    'lam': 0.95,
    'lr': [0.05, 0.005],
    'ann': [0, 1],
    'optim_epoch': 10,
    'batch_size': 7 * 36,
    'epsilon': 0.2,
    'trace_length': None,
    'ent_coef': 0.0,
    'vf_coef': [0.1, 0.01],
    'clip_value': [0, 1],
    'action_distribution': 'Dirichlet',
    'output_activation': 'relu',
    'train': 1,
    'num_update': 100,
    'update_freq': 4,
    'horizon': 1440,
    'save_iter': 5
}
```

GCN-specific

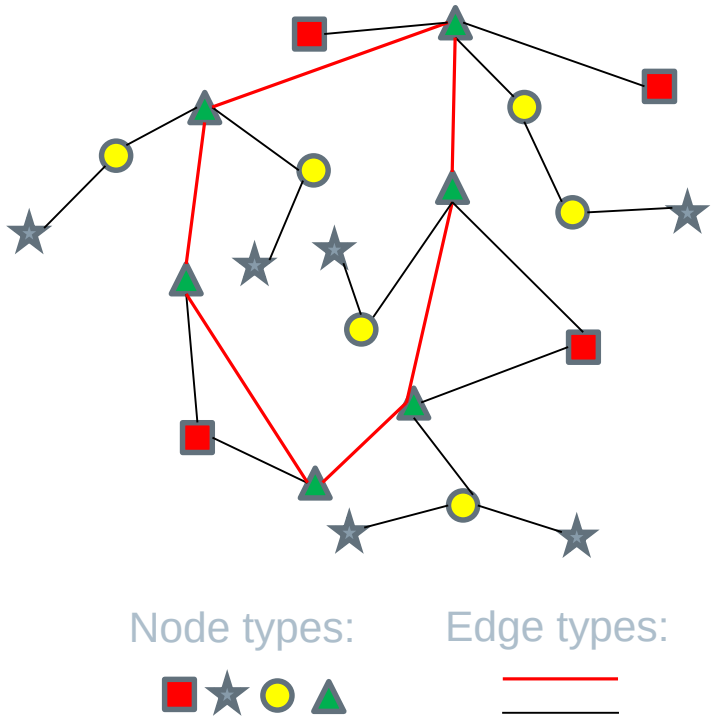
Remarks

- Constructing an adjacency matrix
  - If hyperparameter ‘neighbors’ is set to k, when each load node is connected with k its neighbors in the power grid in the hop distance sense.
  - No respect to the line lengths, functional dependencies, etc.

- Node 822 is connected with nodes 820 and 818 if $k = 2$

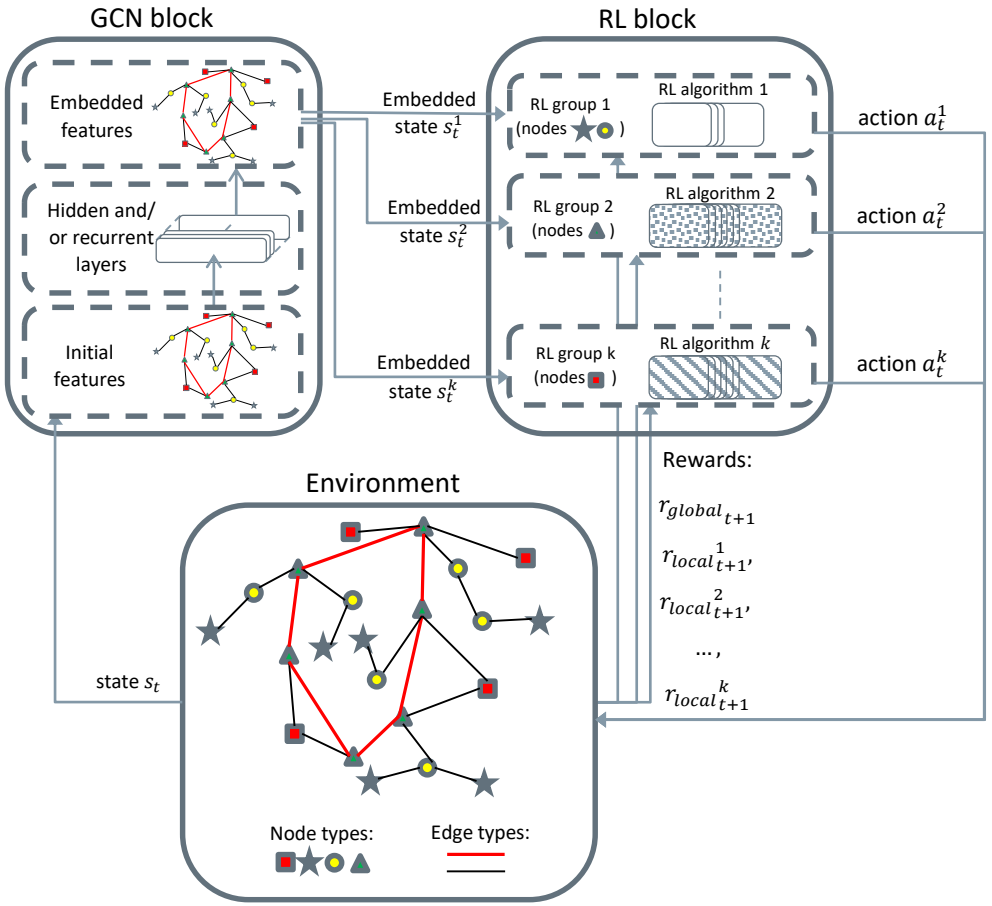


Heterogeneous Agent Groups



- Divide nodes into groups and have a separate control policy per group
- Grouping strategies:
  - According to node types
  - Domain-driven: a domain expert suggests grouping strategy
  - Topology-driven, e.g., hub nodes fall in one group, and the nodes on the periphery fall in the other
  - Data-driven: nodes are divided into groups according to their similarity with some clustering approach
  - Function-driven: nodes' function in the graph may change over time based on what they are connected with.

GCRL with Heterogeneous Agent Groups



- Nodes and edges may be of different types and have their associated feature sets, which are observed in the environment at time moment t and constitute state s_t of the system
- The underlying graph G_t is naturally a part of s_t as it depicts the topology at time moment t.
- The embedded feature set is split into embedded states s_t^i and forwarded to RL group policy i
- Reward r_t^i may contain both local reward $r_{local}{}_{t+1}^i$ (specific to the node group) and global reward $r_{global}{}_{t+1}$ of the system.
- During the learning process, triplets $\left(s_t^i, a_t^i, r_{t+1}^i\right)$ will be used to update RL policy parameters as in conventional RL, and further update corresponding parameters in the GCN layers, which will further tailor the sharable layers to the system control task at hands.

GCRL with Heterogeneous Agent Groups: Advantages

SIEMENS
Ingenuity for life

- Sharable knowledge of the network across policies is encoded in the GCN layers
- Specific control in Group Policies is generated by the RL models
- Increased scalability learning the Group Policies separately and backpropagating the RL policy information to the GCN layers
- Adaptivity to changing conditions (changing topology, new/dropped nodes and links) is learned via aggregation and/or recurrent layers that analyze temporal transitions and thus capture varying network dynamics.
- Adaptive/Fixed Clustering of nodes into groups based on similarity, domain knowledge or differences in action space. Furthermore, as the embeddings capture the node and edge temporal evolution, clustering can be done based on the functional properties of the nodes in the graph.