Traffic Control and Infrastructure Organization Using Reinforcement Learning

Daniel Kuknyo
Eötvös Loránd University, Budapest

June 28, 2023
Thesis Defense

Outline

Urban design in current days

Car dependency causes major problems

- Pollution
- Safety concerns
- Reduced accesibility for all
- More isolated communities
- Less business for services

Düsseldorf, Germany (1989)

Urban design in current days

Car dependency causes major problems

- Pollution
- Safety concerns
- Reduced accesibility for all
- More isolated communities
- Less business for services

Tendencies around the world

- Financial insolvency of car-dependent cities
- Some tending towards less car dependency
- Slow change of policies in EU

Düsseldorf, Germany (1989)

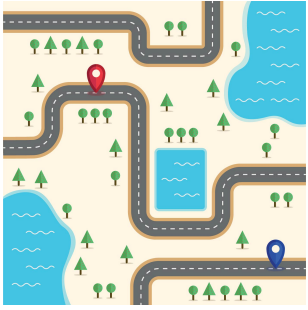
Düsseldorf, Germany (2019)

Practical uses of the research

- **Within-city**: Optimizing an existing piece of infrastructure
  - Keep buildings intact
  - Find optimal junction types
  - Fine-tune lane capacities

Practical uses of the research

- **Within-city**: Optimizing an existing piece of infrastructure
  - Keep buildings intact
  - Find optimal junction types
  - Fine-tune lane capacities
- **Green-field investment**: Only the junction locations are given. Everything else has to be found by the agent
  - Lanes can be built freely between any two junctions
  - Junction types can be set freely

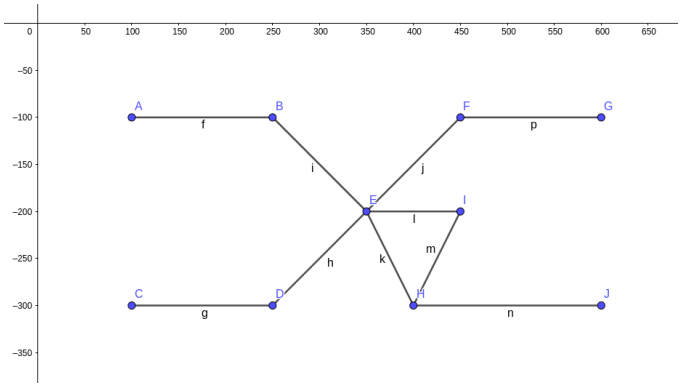


Practical uses of the research

- **Within-city**: Optimizing an existing piece of infrastructure
  - Keep buildings intact
  - Find optimal junction types
  - Fine-tune lane capacities
- **Green-field investment**: Only the junction locations are given. Everything else has to be found by the agent
  - Lanes can be built freely between any two junctions
  - Junction types can be set freely
- **Mixed (reparation mode)**: Optimize an existing piece of infrastructure freely
  - Lanes can be built between any two junctions
  - Junction types can be set freely

Constructing a city

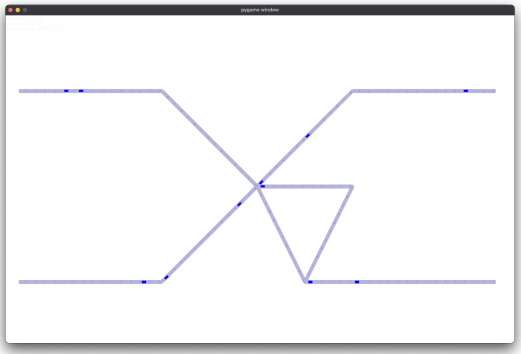
A city is read into the program using a construction protocol. The protocol stores the junctions and their relationships.



The design interface running in GeoGebra

Constructing a city

A city is read into the program using a construction protocol. The protocol stores the junctions and their relationships.



The same city loaded into the simulation

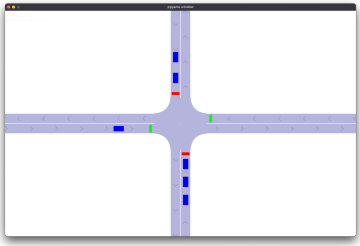
Running the simulation

The drivers behave according to the intelligent driver model. The model defines:

- Maximum speed: v_0
- Safe following distance: T
- Maximum acceleration: a
- Comfortable deceleration: b
- Acceleration exponent: δ
- Comfortable following distance: s_0

Intelligent driver model for vehicle α

$$\dot{v}_\alpha = a^{(\alpha)} \left[1 - \left(\frac{v_\alpha}{v_0^{(\alpha)}} \right)^{\delta} - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha} \right)^2 \right]$$

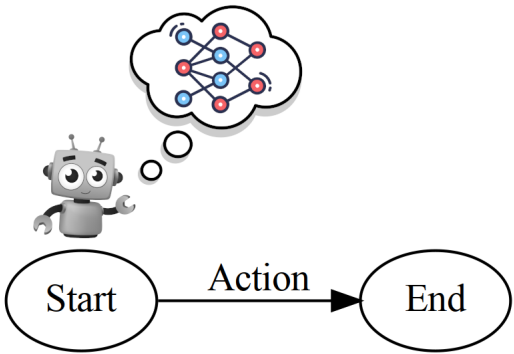


Vehicles in the simulation

The RL task

The task of the agent is to predict the ending node and action from its current position. Each iteration the ending node will become the current position. The choice of actions are:

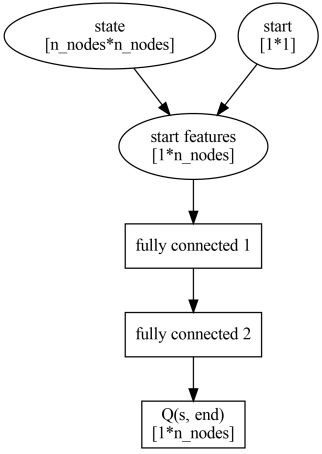
- Add a lane
- Remove a lane
- Build a right-hand intersection
- Build a roundabout
- Build a traffic light



One prediction pass for the agent

State-based architecture

Ending node prediction architecture

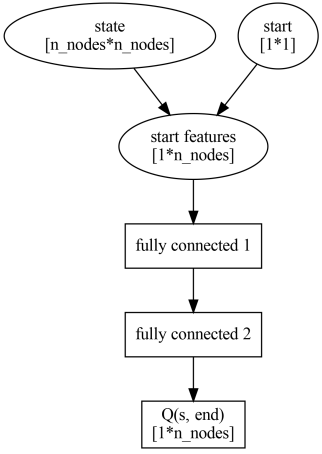


The state of the system

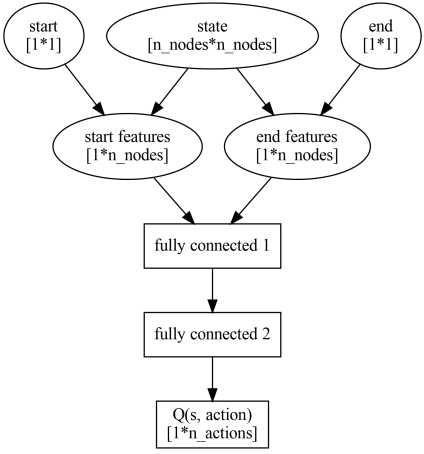
The graph of the city is represented by its adjacency matrix. For n nodes the descriptor matrix x is of size $n * n$. The diagonal elements $x(N_i, N_i)$ describe the self-connections e.g. the type of intersection that can be found at junction N_i in the city. All other connections $x(N_i, N_j)$, $i \neq j$ represent the number of lanes between nodes N_i, N_j.

State-based architecture

Ending node prediction architecture

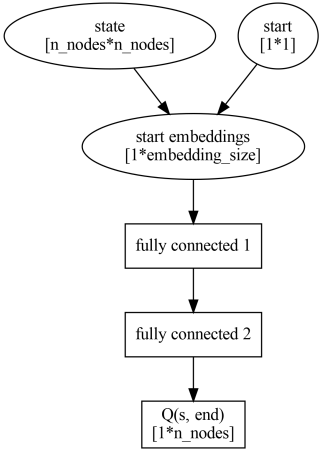


Action prediction architecture



State-embedding-based architecture

Ending node prediction architecture

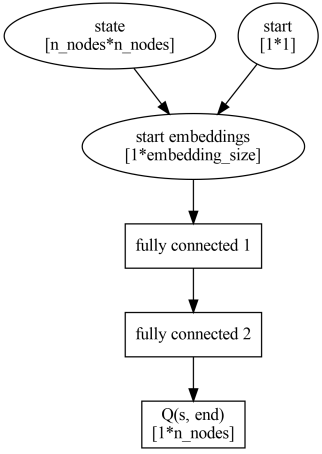


Embeddings with graph-structured data

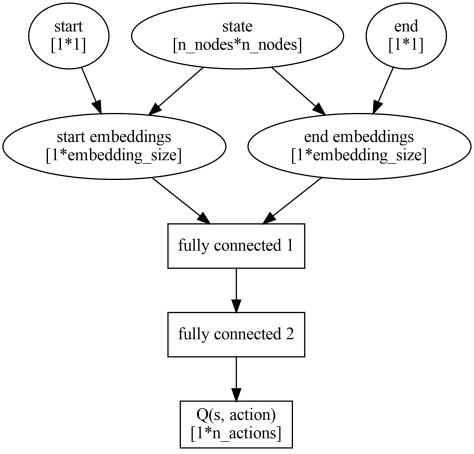
In graph neural networks (GNN) embeddings are used to capture the structure and relationships within the graph, allowing the network to perform tasks such as link prediction and node classification.

State-embedding-based architecture

Ending node prediction architecture

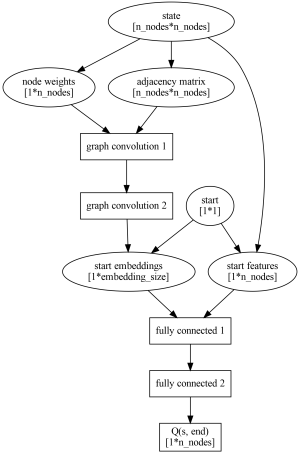


Action prediction architecture



Graph-convolutional architecture

Ending node prediction architecture



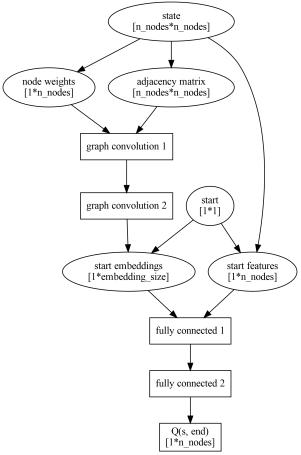
The propagation rule for GCN

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

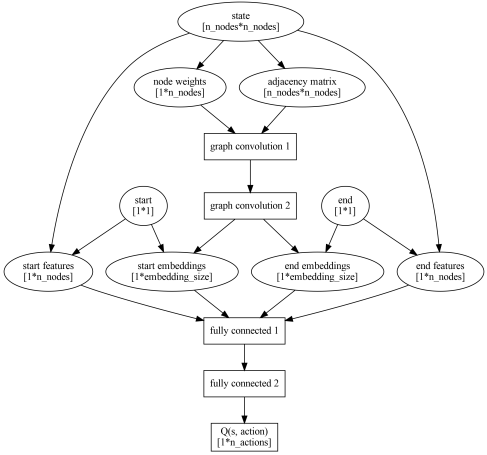
- $\sigma(.)$ is the activation function
- $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the degree matrix of the graph
- $\tilde{A} = A + I_N$ is the graph's adjacency matrix including self-connections
- $W^{(l)}$ are the trainable weights of the layers
- $H^{(l)}$ denotes the activations of the l-th layer of the network

Graph-convolutional architecture

Ending node prediction architecture

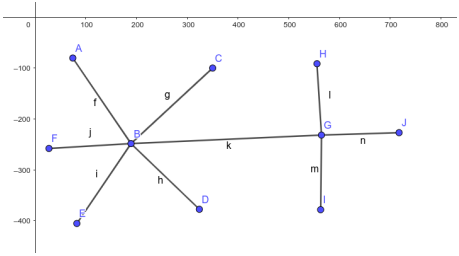


Action prediction architecture



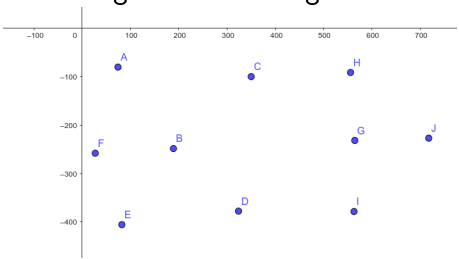
Testing environment

The mixed configuration



- The two intersections are bottlenecks
- All junctions are right-hand by default
- The roads have one lane in each direction

The green-field configuration



- Only the locations of the junctions are given
- The agent has to build everything from scratch

Episodic rewards for training

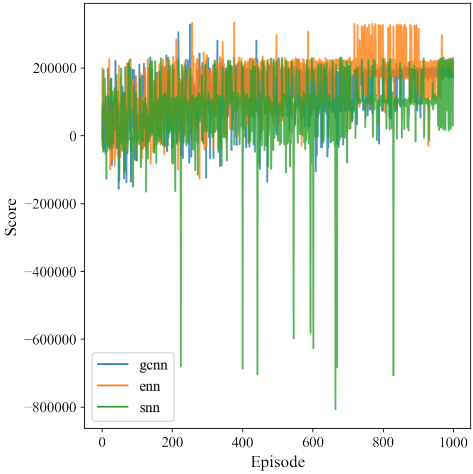
- The training was ran for 1000 episodes
- The performance was measured for all 3 models
- The unit of measurement is the cumulated reward over 15 building steps
- A city design at time step t is scored using the formula:

$$score^t_{city} = -cost^t_{infrastructure} + score^t_{human} + score^t_{vehicular}$$

- The reward signal is calculated as the difference in city scores:

$$r_t = \delta_{score_{city}} = -(score^{t-1}_{city} - score^t_{city})$$

Unsmoothed rewards (mixed mode)



Episodic rewards for training

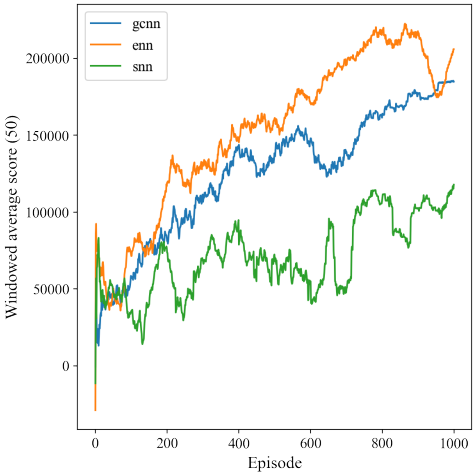
- The training was ran for 1000 episodes
- The performance was measured for all 3 models
- The unit of measurement is the cumulated reward over 15 building steps
- A city design at time step t is scored using the formula:

$$score^t_{city} = -cost^t_{infrastructure} + score^t_{human} + score^t_{vehicular}$$

- The reward signal is calculated as the difference in city scores:

$$r_t = \delta_{score_{city}} = -(score^{t-1}_{city} - score^t_{city})$$

Smoothed with 50-size window (mixed mode)



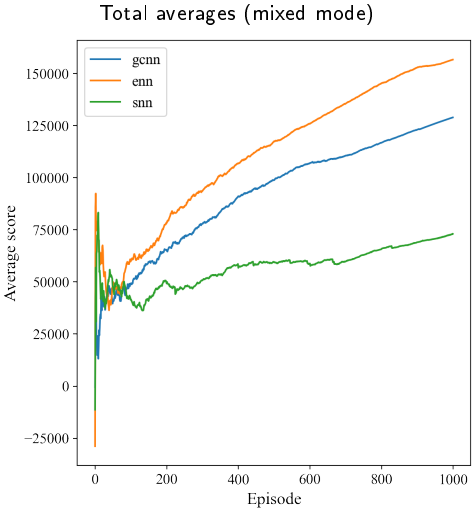
Episodic rewards for training

- The training was ran for 1000 episodes
- The performance was measured for all 3 models
- The unit of measurement is the cumulated reward over 15 building steps
- A city design at time step t is scored using the formula:

$$score^t_{city} = -cost^t_{infrastructure} + score^t_{human} + score^t_{vehicular}$$

- The reward signal is calculated as the difference in city scores:

$$r_t = \delta_{score_{city}} = -(score^{t-1}_{city} - score^t_{city})$$



Episodic rewards for training

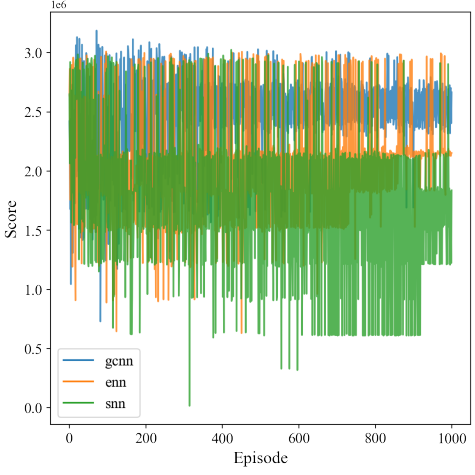
- The training was ran for 1000 episodes
- The performance was measured for all 3 models
- The unit of measurement is the cumulated reward over 15 building steps
- A city design at time step t is scored using the formula:

$$score^t_{city} = -cost^t_{infrastructure} + score^t_{human} + score^t_{vehicular}$$

- The reward signal is calculated as the difference in city scores:

$$r_t = \delta_{score_{city}} = -(score^{t-1}_{city} - score^t_{city})$$

Unsmoothed rewards (green-field mode)



Episodic rewards for training

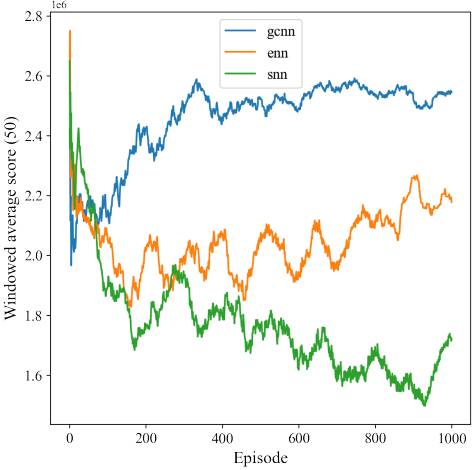
- The training was ran for 1000 episodes
- The performance was measured for all 3 models
- The unit of measurement is the cumulated reward over 15 building steps
- A city design at time step t is scored using the formula:

$$score^t_{city} = -cost^t_{infrastructure} + score^t_{human} + score^t_{vehicular}$$

- The reward signal is calculated as the difference in city scores:

$$r_t = \delta_{score_{city}} = -(score^{t-1}_{city} - score^t_{city})$$

Smoothed with 50-size window (green-field mode)



Episodic rewards for training

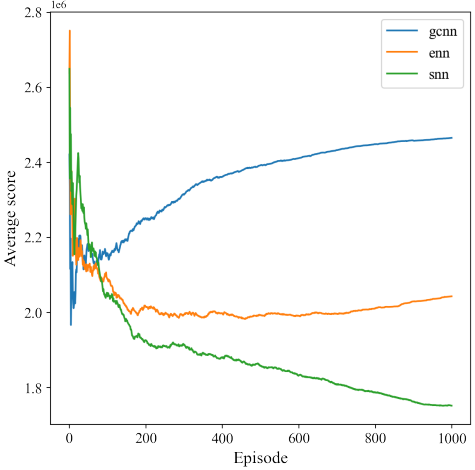
- The training was ran for 1000 episodes
- The performance was measured for all 3 models
- The unit of measurement is the cumulated reward over 15 building steps
- A city design at time step t is scored using the formula:

$$score^t_{city} = -cost^t_{infrastructure} + score^t_{human} + score^t_{vehicular}$$

- The reward signal is calculated as the difference in city scores:

$$r_t = \delta_{score_{city}} = -(score^{t-1}_{city} - score^t_{city})$$

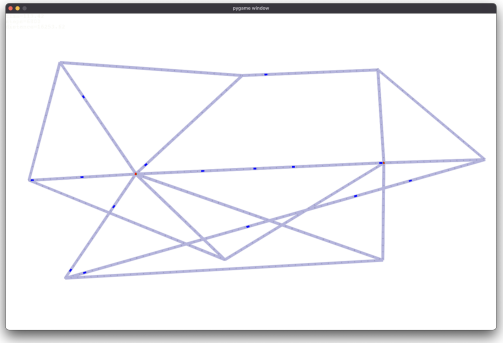
Total averages (green-field mode)



Cities built by the agent

After the learning episodes the agent was used to execute 15 building steps in the environment.

- Graph-convolutional model (mixed)

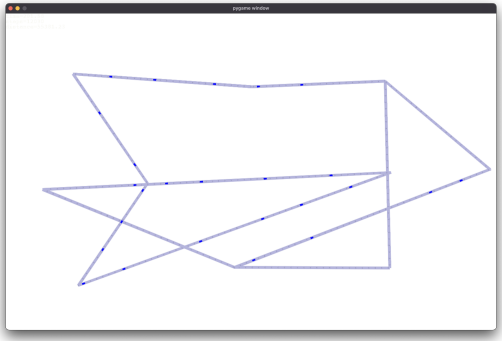


- Distant nodes were connected
- Traffic lights added on central junctions
- Fast traffic flow and humanly livable

Cities built by the agent

After the learning episodes the agent was used to execute 15 building steps in the environment.

- Graph-convolutional model (mixed)
- Graph-convolutional model (green-field)

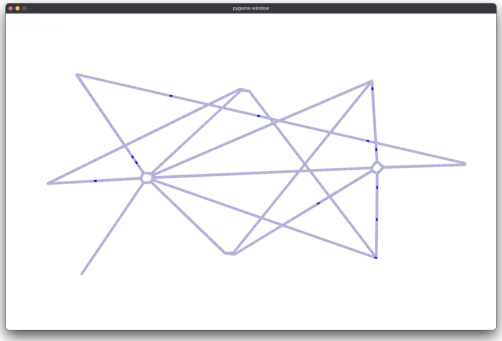


- Distant nodes were connected
- No traffic lights or roundabouts
- Road structure doesn't allow smooth flow

Cities built by the agent

After the learning episodes the agent was used to execute 15 building steps in the environment.

- Graph-convolutional model (mixed)
- Graph-convolutional model (green-field)
- Embedding-based model (mixed)

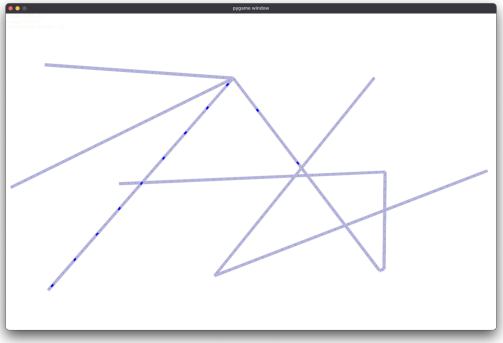


- Fully optimized for cars
- Roundabouts in central junctions
- Fast traffic flow but not human-friendly

Cities built by the agent

After the learning episodes the agent was used to execute 15 building steps in the environment.

- Graph-convolutional model (mixed)
- Graph-convolutional model (green-field)
- Embedding-based model (mixed)
- Embedding-based model (green-field)

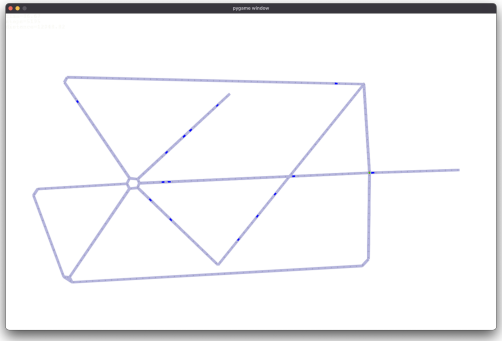


- The traffic system is not continuous
- No junction is left without a connection
- No roundabouts or traffic lights

Cities built by the agent

After the learning episodes the agent was used to execute 15 building steps in the environment.

- Graph-convolutional model (mixed)
- Graph-convolutional model (green-field)
- Embedding-based model (mixed)
- Embedding-based model (green-field)
- Adjacency matrix-based model (mixed)

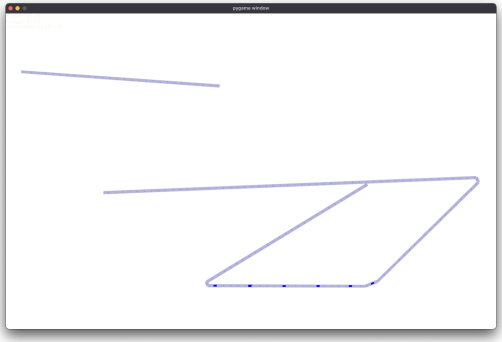


- Distant nodes were connected
- Roundabout and traffic light on junctions
- Fast traffic flow but suboptimal

Cities built by the agent

After the learning episodes the agent was used to execute 15 building steps in the environment.

- Graph-convolutional model (mixed)
- Graph-convolutional model (green-field)
- Embedding-based model (mixed)
- Embedding-based model (green-field)
- Adjacency matrix-based model (mixed)
- Adjacency matrix-model (green-field)



- Many nodes left without connection
- The traffic system is not continuous
- Unnecessary roundabouts added - raising costs

Conclusion

- The agents all learned to optimize the structure in one way or another
- The task is very noisy - it needs constraints to make it work in this environment
- The costs and rewards need thorough research to get the expected result
- A more realistic simulation environment and driver model would yield more accurate results, however it would also need a lot more computing power
- The models were very prone to overfitting: heavy regularization is required
- Single-network architectures couldn't learn to maximize long-term rewards

Thank you for your time and consideration!