MSc Artificial Intelligence
Master Thesis Proposal

Predicting plausible escape routes using reinforcement learning and graph representations

by
Rogier van der Weerd MSc
Student nr. 13454242

March 12, 2022

Thesis work (48EC)
November 2021 until June 2022

Supervisor:
D. Kuric MSc (UvA)
Mentors:
I.S. Van Droffelaar MSc (TU Delft)
T. Kastelein MSc (Dutch Police)

Examiner:
T.b.d.

Second reader:
T.b.d.

Universiteit van Amsterdam

Contents

1 Introduction **1**
1.1 Context, problem definition and starting point 1
1.2 Relevant definitions 1
1.2.1 Task and campaigns 2
1.2.2 Problem setting 2

2 Research Question **3**

3 Literature Review **4**
3.1 Search and Pursuit-Evasion (SPE) problems 4
3.2 (Deep) Reinfocement Learning in the SPE domain 4
3.3 Graph-based Reinforcement Learning 5

4 Methodology **6**
4.0 Preliminary work 6
4.1 Phase 1: Basline RL experiments to evaluate pursuit model performance 7
4.2 Phase 2: Develop generalization over graphs 8
4.3 Phase 3: Scale-up to real-world road networks 9
4.4 Phase 3+: Extend to sporadic position information 9

5 Planning **10**

A Example graphs & rollouts **11**

Bibliography **12**

Abstract

A search & pursuit strategy by police units (pursuers) is aimed at capturing a suspect that attempts to flee a crime scene (escaper). This problem can be modeled using graph traversal, where the graph represents a road network with intersections (vertices) and roads (edges). Many modeling choices are involved in representing this real-life problem in a meaningful and feasible way. Research at the Dutch Police has focused on finding pursuer strategies that maximize capture probabilities using various optimization techniques. However, the quality of these strategies is difficult to assess given the uncertainty of how an escaper will behave.

This thesis work will investigate whether Reinforcement Learning (RL) in combination with Graph Representation Learning can be applied to find escape policies that exploit weaknesses in these pursuer strategies. Ideally, escape policies can be learned for any (unknown) pursuer strategy and such escape policies generalize over different (unseen) graphs and different initial conditions (such as number of pursuers and their positions at the start of an episode). This represents a challenging task given that i) the environment will be non-Markovian with limited observability, ii) the problem is NP-hard with a combinatorial search space that explodes with increasing graph size and complexity.

If successful, the RL approach can be used to test the effectiveness and robustness of pursuit strategies and yield an objective performance metric of such strategies.

Chapter 1

Introduction

1.1 Context, problem definition and starting point

The Dutch Police are interested in decision support applications that can improve performance in search & pursuit scenarios of suspects fleeing a crime scene. In such scenarios, deployed police units are guided by operators in a control room that use real-time centralized information gathering and coordination. Decision making is based on experience and intuition by both dispatchers and field units. Effective response can be hindered by many factors, including lack of information, sub-optimal use of data, miscommunication, non-compliance of instructions, etc.

Prior work by Kastelein [Kas20] uses a graph-based representation of a road network within a bounded area around the crime-scene. She frames the optimization as a Integer Linear Programming (ILP) problem in order to output pursuer routings that maximize the number of potential escape routes that are intercepted within a certain number of time-steps. It assumes random behavior by the evader, possibly with directional preference. Her work was used to develop a prototype system called IMA that is ably to apply the optimization on the real Dutch road network, as one step towards bringing such information into the actual police operation. Van Droffelaar [vD21] is continuing this research with the purpose of improving the decision support method and identify timely, robust and explainable fugitive interceptions, given limited real-time data.

Testing performance and robustness of these search & pursuit optimization methods using an objective and reliable metric is crucial given the stakes in any real-life deployment. However, such performance measurement is difficult as escape behaviour is unpredictable and hard to model. Real-life data is hard to acquire and can only be observational, not interventional. One avenue worth investigating is the use of Reinforcement Learning (RL) to train an escape agent to based on experience in a simulated environment where the pursuer strategy is applied. Such a trained agent may expose weaknesses by learning to exploit these pursuer strategies. This information may be used to evaluate performance of pursuer strategies.

Note that RL could be used to develop pursuer strategies as well, potentially in a setting of adverserial learning. However, the primary goal of this work is to learn escape policies in order to assess the performance of pursuer strategies.

1.2 Relevant definitions

The domain of search, pursuit & evasion problems constitutes a large family of problems, with many variants depending on aspects such as the number and type of agents (mode of transport, capability[1]), their degrees of freedom in movement (the action space), definition of objectives, level of observability[2], level of cooperation, type of environment, etc. Many modeling decisions and assumptions will need to be made in order to define and evaluate escape/pursuer behaviors. As a starting point, two key aspects of problem can be defined.

1.2.1 Task and campaigns

In the context of search & pursuit optimization for the purpose of this study, an escape agent is located on a road network with a certain objective[3], and any number of pursuers are present on the network with the aim of capturing the escaper using coordination. We will distinguish:

- *Search campaigns*: the exact escaper position is unknown and there is no current visual identification of the escaper. There may be knowledge of recent escaper positions. The primary goal of pursuit units is to locate the escaper. Search campaigns can be:
  - Static: deployed pursuers assume strategic stationary positions with the objective of visual identification of an escaper
  - Dynamic: deployed pursuers navigate strategic surveillance routes with the objective of visual identification of an escaper
  - Hybrid[4]: deployment of a mix of passive and dynamic search agents
- *Pursuit campaigns*: current escaper location and direction is available through active visual contact by at least one pursuer. All pursuers coordinate with the objective of intercepting and stopping the escaper.

These two modes are different in nature: finding a target will require a different approach than stopping and capturing a target that is in sight. The primary interest of the Police lies in the effectiveness of search campaigns as the potential for improvement of this task is expected to be highest.

1.2.2 Problem setting

The search & pursuit problem can be most naturally represented on a graph that is representative of the road network. This representation can be literal, with nodes and edges representing physical intersections and road segments. The representation may also contain some level of abstraction, for instance by considering nodes to represent a general geographic area that can be observed when a pursuit agent is located anywhere in that area, and edges representing (coarsened) ways to move from one area to another.

A unique state of the graph is defined by the set of node positions of all agents at a specific time step, combined with their time histories (paths walked from the initial positions) and the static set of target nodes for the escaper. The events of visual identification or capture can be defined based on the state of the graph (i.e. escaper and at least one pursuer are located on the same node), or transitions between states (i.e. their walks have crossed on an undirected edge during a transition). Figure A.1 in Appendix A illustrates this basic setting.

[1]E.g. a helicopter or road camera can only observe, a police car can observe and intercept, etc.

[2]E.g., which information (location, direction and speed of travel, intentions) of other agents is known at which time-steps

[3]This might be simply not to get caught, possibly combined with a goal to reach a target location, or increase distance from the crime scene

[4]In practice, search campaign will be hybrid as agents will (rightfully) apply judgement to decide between holding a position and moving around. However, the distinction will be useful in modeling behaviors.

Chapter 2

Research Question

Given the problem definition described above, we will be considering a setting of an escape agent that traverses a directed graph. The graph contains a certain number of pursuit agents that are engaged in a search campaign using a strategy unknown to the escaper. The escaper has a primary objective of not being found and (potentially) a secondary objective of reaching a sub-set of nodes on the graph (considered a safe haven) with some level of urgency.

The main research question of this work is as follows:

> How can Reinforcement Learning (RL) be applied to find an optimal strategy to escape from pursuer adversaries on a graph, such that this strategy generalizes to i) any set of initial positions, ii) any number of pursuers, iii) any (unseen) graph, iv) any level of observability?

We will limit the work to involve pursuers that engage in search campaigns and assume their behavior to be a black box that can be interrogated. The escape agent will be trained with some budget (number of episodes) and its performance is measured on graphs and graph instances it has not observed before (zero-shot learning). Sub-questions include:

- How to model this problem such that it captures essential real-life behavior and constraints, while it remains feasible to perform optimization on it?
- How to solve the generalization problem in its multiple dimensions?
- If successful and generalizable escape strategies can be found, how can this knowledge be used define a performance metric of pursuer strategies?

Our hypothesis is that a combination of *Graph Representation Learning* (using graph neural networks) and Reinforcement Learning can be effective in this application. Recent advances in these techniques demonstrate promise in the ability to create latent representations of graph states and learn the abstract heuristics required to solve these type of combinatorial optimization problems on graphs.

Questions that are out of scope for this work include:

- How can RL be used to learn pursuer strategies, possibly in an adversarial setting?
- How to translate findings into a decision support model that can be deployed in actual police operations?
- How to extend this work to pursuers that engage in active pursuit?

Chapter 3

Literature Review

This work looks to bring to together three distinguishiable areas of research: i) multi-agent search and pursuit-evasion (SPE) problems, ii) reinforcement learning and iii) graph representation learning. The first two of these already have a range of combined applications. However, applying geometric deep learning to the SPE domain has not been extensively studied to date, although it is a natural extension of substantial work already done in applying GNNs to solve combinatorial / NP-hard graph-based problems.

3.1 Search and Pursuit-Evasion (SPE) problems

The starting point of this thesis work is formed by [Kas20], who translated the pursuit optimization problem in the scope of decision support in a control room of the Dutch Police. She applies Integer Linear Programming (ILP) in order to output pursuer routings that maximize the number of potential escape routes that are intercepted within a certain number of timesteps. [vD21] builds on this work with the aim of improving performance and explainability of the solutions, which will help in adoption by practitioners.

Research on search, pursuit and evasion problems goes back to early game theoretic formulations [ICfIM65] and formulations on graphs [Par78], where solutions under full information and alternating turns (dubbed *Cops and Robbers games*) are offered. [Als04],[CHI11] survey and categorize many variants of the problem and classical approaches to solving them such as the marking algorithm, probabilistic search and other methods developed in operations research. Small and specific formulations may have exact solutions, such as [WW95] that uses network flow techniques. Given the computational complexity, however, most existing solutions are heuristic in nature and few approximation algorithms have provable performance guarantees.

3.2 (Deep) Reinfocement Learning in the SPE domain

Many recent successes that combine deep learning with reinforcement learning techiques have shown their combined potential in intelligent decision making and control [BRW+19]. These techniques have also been introduced in the SPE domain with the main advantage they do not require differential game models: agents can learn optimal strategies solely by interacting with the environment. Most approaches aim to solve cooperative multi-agent tasks from the perspective of the pursuers with a range of methods, using (deep) Q-learning (DQN) [BKU15],[GEK17], various policy based methods such as Trust Region Policy Optimization (TRPO) [HAN+19], Deep Deterministic Policy Gradient DDPG [LWT+17], and extensions such as curriculum learning [dSNC+21]. [LWT+17] extends DDPG into a multi-agent policy gradient algorithm where decentralized agents learn a centralized critic based on the observations and actions of all agents.

Additional challenges are faced when addressing limited observability by the agents, typically described by Partially Observable Markov Decision Processes (POMDPs). There is well developed research into dealing with POMDPs in RL such as Deep Recurrent Q-Learning (DRQN) [HS15] (dealing with flickering screens in DQN for Atari games by jointly training convolutional and LSTM layers), [ZLPM17](incorporating past actions in state histories), [SSP+15] (adding attention layers to DRQN), and others [SYML18],[ZLPM17], mostly by introducing recurrent neural networks in the parametrized value or policy models. This idea was first shown to be feasible by [WFPS07], who used LSTMs [HS97] to map histories (memory states) to action probabilities in a Policy Gradient framework. Particular challenges arise when using RNNs in combination with experience replay, investigated by [KOQ+18],[SYML18]

[OAMAH15] extends the notion of decentralized POMDPs for multi-agent cooperation with asynchronous decision making, improving scalability by using belief space macro actions in planning. The challenge of generalizing policies such that they can be applied to different number of agents than what they were trained on was attempted by [XHG+19] using bi-directional LSTMs with DDPG.

3.3 Graph-based Reinforcement Learning

The field of Graph Representation Learning has gained importance given its ability to use inherent structure of information (encoded in the graph) in machine learning algorithms [Ham20]. This approach has seen breakthroughs in tasks such as node classification ([KW17] who introduced graph convolutions), relation/edge prediction (survey by [NMTG16]) and graph-level classification/regression/clustering (e.g. [GSR+17] who formulated message passing neural networks and applied it to molecular property prediction). [BHB+18] and [GNNSys21[1]] provide excellent overviews of the state of this field.

Particularly relevant for SPE problems are recent insights into combinatorial optimization and reasoning using graph neural networks and RL (surveyed in [CCK+21],[PCX21]). The SPE task can be framed as a routing problem, where next nodes need to be selected based on a graph state (which may include a node selection history). Early ideas involved *Pointer Networks* [VFJ15],[BPL+16],[VBO+20] where a recurrent network with is trained in a supervised manner to predict the sequence of visited cities in the Travelling Salesman Problem (TSP). [KDZ+17] approaches the same problem using *structure2vec* [DDS16], a structured data representation built by embedding latent variable models into feature spaces, and learning such feature spaces using discriminative information from graphs. [KvHW19] successfully applies *Graph Attention Networks* [VCC+18] to learn heuristics to solve routing problems on graphs.

It is to be expected that these methods can be applied to the SPE formulation studied in this thesis. This will require an appropriate definition of edge and node features and a network architecture that is able to deal with unknown and potentially stochastic behavior of the pursuer adversaries. Promising research [XLZ+20],[VYP+20] investigates how network architecture can be matched with the nature of the optimization problem (*algorithmic alignment*) in order to enable generalization. Lastly, any approach should be scalable to reasonably sized road networks (>1k node graphs). [MMD+19] introduces scalability techniques using a probabilistic greedy mechanism to predict the quality of a node, which may provide useful in our application.

[1]https://gnnsys.github.io/

Chapter 4

Methodology

In order to answer the research question, we will set up experiments that test the performance of the behavior of an escape agent, modeled by a parametrized policy $\pi_\theta(a|s)$. This policy assigns a probability to each available action a (choosing a next node to move to) for any graph state s. In order to find successful behaviors, we will be looking to define policy models that can learn to represent the graph state in a high-dimensional latent space and transform this representation to state value estimates or action probabilities. We will train these models using suitable Reinforcement Learning algorithms. This, in turn, will require simulation environments that are made up of different graph classes, populated by a varying number of pursuit agents that follow pre-defined search behavior. The performance of these trained policies will be evaluated and their generalization properties tested along the various dimensions mentioned in the research question. The various anticipated stages of this work are described next.

4.0 Preliminary work

Determining a modeling approach A first prerequisite will be a translation of our problem definition into a model that captures its essential aspects with the lowest level of complexity. This will require various design choices, depicted in Table 4.1. Some of these choices will have to be informed by the expertise of experienced agents and dispatchers at the Police, such as the question which information about pursuers might be visible to a sophisticated escaper.

		Design options	
Design choice		Basic	Potential extensions
World definition	Environment:	Reflexive Graph, node=intersection, directed edge=road, all weights=1	Expand to real road networks; Allow varying edge weights (distances)
	Size:	limited, $<$50 nodes	$\leq$10,000 nodes
	Complexity:	Max indegree ($\leq$6)	No limitation
	Speed of travel:	Normalized to 1 edge per timestep	Allow varying speeds
	World bounds:	Fixed, can't cross boundary	Bounding box moves along
Environment characteristics	State space:	All agents' positions and histories adjacency matrix if required	-
	Action space:	All nodes, masked by reachability	-
	Observation space:	Various levels of observability for E; Only own team positions for P	Sporadic information updates* for E/P
	Transitions:	Synchronous, deterministic	Noise (e.g. ignoring instructions)
	Termination:	Captured, Goal or T_{max} reached	-
	Reward function:	-1 per action, +10 goal, -10 captured	Some measure of safe passage
Pursuit scenario	# evaders E:	1	-
	# pursuit agents P:	[1..3]	Subject to optimizer limitations; Could be $>$20 in real life
	Goal of evader:	Reach some destination, uncaptured with some level of urgency	Longest time uncaptured (can hide)
	Definition of capture	E & some P occupy same node edge crossing allowed	E & P are within certain distance edge crossing means capture
Learning scenario	Who learns?	E	P or both P/E (adverserial learning)

* E.g., visibility of E by P when E passes a ANPR camera, or vice-versa when E uses spotters

Table 4.1: Overview of Reinforcement Learning design aspects (E=evader, P=pursuit agent(s))

Developing a simulation environment We will need to develop a customized Reinforcement Learning environment to conduct our experiments, following the broadly accepted standards of OpenAI's `gym`[1] framework. A particular challenge involves incorporating the existing optimization model and interface for calculating pursuer agent behavior. The work of [Kas20] and [vD21] has led to an IMA codebase (proprietary) containing various pursuit optimization models that can act in real-life scenarios on the Dutch road network. This optimization is computationally heavy, and some form of pre-calculation of pursuer actions will need to be developed in order to conduct feasible RL experiments.

To illustrate the basic escape task, consider Figure 4.1 which depicts set-ups with three basic graph layouts, useful for initial experiments. As time progresses, the pursuers will move to their designated observation positions (these are given by the IMA optimizer, assumed unknown to us). The primary goal of the escaper is to not get caught. The secondary goal is to reach a target area (depicted by the red line) with some level of urgency. At each time-step, all agents take an action (move to any reachable node) simultaneously.

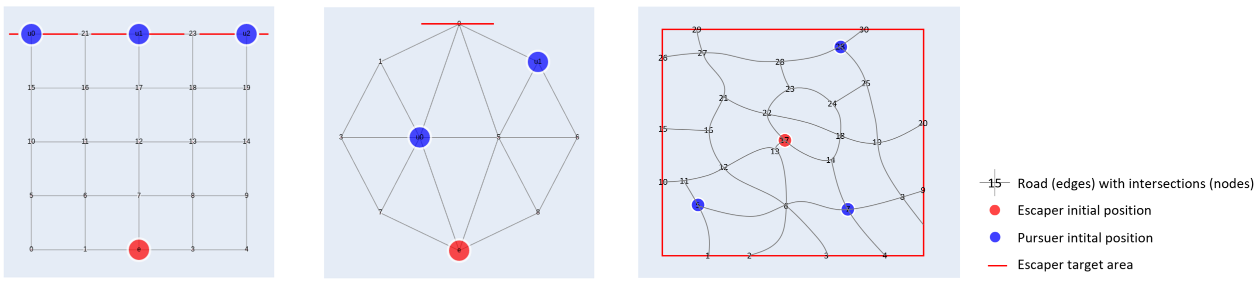


Figure 4.1: Examples of the basic escape task

Developing a Graph factory To test generalizability of learned policies to different (unseen) graph classes, it will be necessary to have access to different graph typologies and graph permutations within these typologies. A graph permutation can be created by removing a certain amount of edges from the base class. Each graph permutation can be spawned with different initial conditions: the starting position of all agents on the graph. Figure 4.3 illustrates graph permutations for a basic 3x3 Manhattan base class.

4.1 Phase 1: Basline RL experiments to evaluate pursuit model performance

The first challenge will be to demonstrate that RL can be used to find optimal escape behavior that generalizes over different initial positions of pursuers and evaders for a fixed graph. We will test the performance of selected RL algorithms and for different levels of observability. This experiment will start with small and simple environments (graphs) to design and test learning algorithms and increase size/complexity when successful. We won't be applying graph representation learning at this initial stage. State information will be encoded either as tuples of agents' node positions, or one-hot encodings as flattened vectors, depending on the algorithm used. This encoding will not contain any information on graph structure.

[1]https://gym.openai.com/

We will use four levels of observability of the environment by the escape agent: minimal observability of only E's own position at each time step with $\bar{o}_t = (e_t)$; minimal observability conditioned on visibility of all agents at $t = 0$: $\bar{o}_t = (e_t, e^0_0, u^0_0, u^1_0, ..., u^U_0)$; full observability at each time step: $\bar{o}_t = (e_t, u^0_t, u^1_t, ..., u^U_t)$; and full observability conditioned on visibility of all agents at $t = 0$: $\bar{o}_t = (e_t, u^0_t, u^1_t, ..., u^U_t, e^0_0, u^0_0, u^1_0, ..., u^U_0)$. For each of these, we will apply the following methods and compare performance on the full train set of graph instances:

- Apply basic heuristics for the escaper as low-marks of performance: random walk, shortest path to nearest target node, minimum in-degree path to nearest target node.
- Apply tabular Q-learning with unlimited size of the Q-value lookup tables to have a basic starting point
- Apply Deep Q-Learning (DQN) with Q-value function approximation using a neural network. We will test whether the escape agent can learn to generalize over initial conditions by using experience replay. We will attempt to extend this to Deep Recurrent Q-Learning (DRQN) by adding an LSTM module to improve performance in limited observability scenarios
- Apply policy-based RL using Proximal Policy Optimization (PPO), also in combination with an LSTM module (RNN-PPO).

4.2 Phase 2: Develop generalization over graphs

After having established baseline performance of well established RL methods, we will augment these methods by using graph embeddings that may enable generalization of escape policies over different graphs. This will require definitions of node features (and potentially edge features) as depicted in Figure 4.2. Our aim is to combine the learning of node and graph-level embeddings (encoder) with a transformation function (decoder) that can represent a value function or actor-critic functions when using policy-based methods. Our aim is to apply DQN and PPO as in phase 1, possibly with recurrent modules. If successful, this would yield and end-to-end training method to learn policies that can 'interpret' a graph state and learn the heuristics necessary to find best responses to arbitrary graph states that E may encounter on any (unseen) graph.

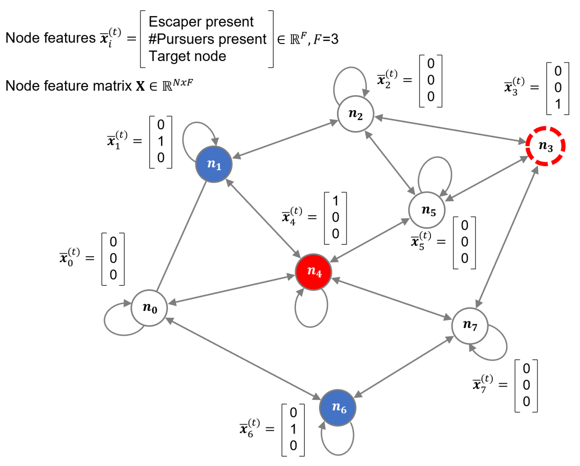

Node features $\bar{x}_i^{(t)}$ = [Escaper present, #Pursuers present, Target node] $\in \mathbb{R}^F, F=3$
Node feature matrix $\mathbf{X} \in \mathbb{R}^{NxF}$
n_0
n_1
n_2
n_3
n_4
n_5
n_6
n_7

Figure 4.2: Node features on a graph modeling a search-evasion scenario

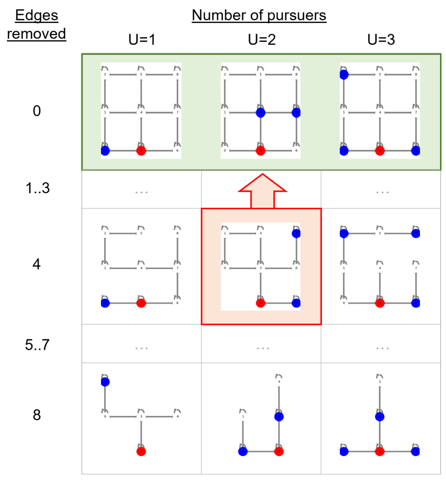


Figure 4.3: Example of graph permutations used to test generalization

To investigate generalization over graphs, experiments can be designed where we train our agent using a particular subset of graphs (e.g. the center segment indicated in Figure 4.3 and evaluate the performance of the learned policy on graphs that are outside the distribution of train graphs. For instance, we can test on the top segment indicated in Figure 4.3 to see if the agent can uphold performance there. Or, evaluate on instances with more pursuit agents present (to the right in Figure 4.3). We can even test on larger graph types to see if learned behavior on small graphs can be useful in larger graphs, perhaps because solving very localized evasion problems can used to find longer range solutions on large graphs.

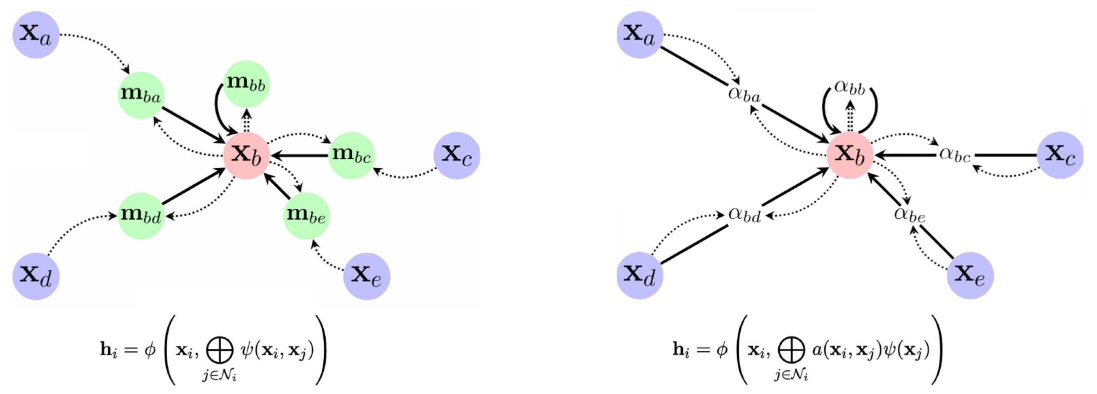


Figure 4.4: Generic propagation function formulations for message passing (left) and attentional (right) Graph Neural Networks (taken from [Vel21])

Our hypothesis is that graph embedding methods that have been successfully used for combinatorial optimization on graphs [PCX21] can be extended to our PSE scenarios. In particular, we will investigate the applicability of *structure2vec* representations and *Graph Attention Networks* [KvHW19]. These two methods can be interpreted as message passing and attentional Graph Neural Networks, respectively, following the generic formulation offered in [Vel21].

4.3 Phase 3: Scale-up to real-world road networks

Once feasibility of our approach is demonstrated in small experimental environments (graphs), we will attempt to apply it to larger real-world road networks. These can be extracted using the existing IMA framework. We expect various challenges associated with scalability. One of these may involve the lack of long-distance information propagation when GNNs are used with a limited number of layers (essentially: the amount of hops that are used to propagate information between nodes). Another challenge may arise from exponential computational complexity as the number of nodes increases. Strategies to deal with these challenges will need to be developed depending on how they manifest themselves.

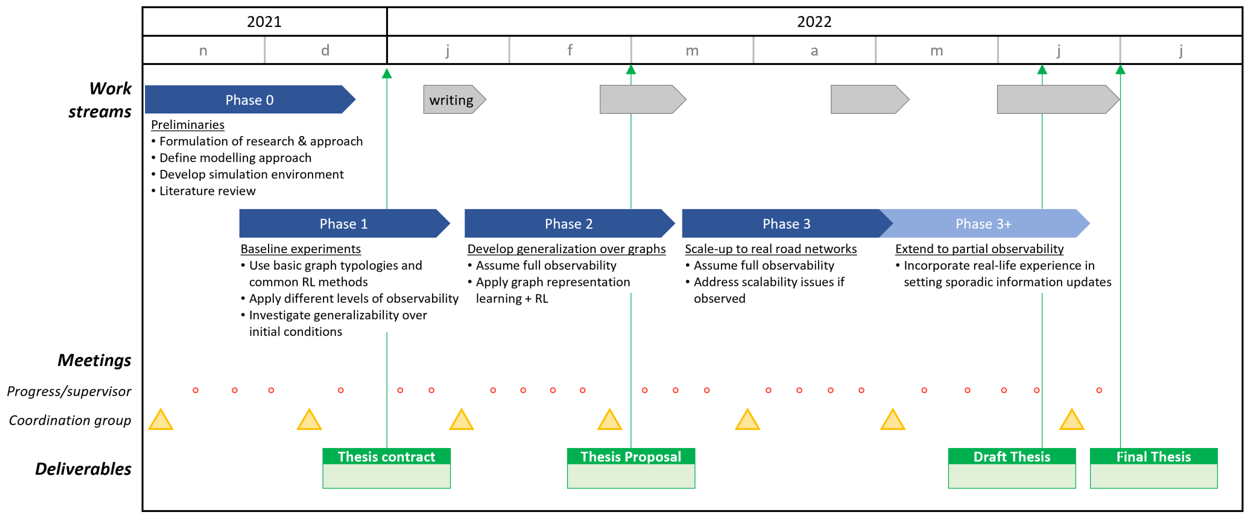
4.4 Phase 3+: Extend to sporadic position information

An aspirational extension of this work, when feasible, will be to allow for sporadic access to positional information of adversaries. This may go in two directions: an escaper may, with some frequency or probability, get passed information on the whereabouts of a subset of police units. Also, the pursuers may get sporadic updates through number plate recognition (ANPR) cameras or witness reports. By using state / observation masking during training and testing, sensitivity to various levels of access to information may be investigated. Whether this extension is feasible within the timeframe of the thesis project will depend on the progress made in the first three phases.

Chapter 5

Planning

A high-level outline of the different phases of research and their approximate timing is offered in Figure 5.1. A coordination group has been set up, consisting of the thesis supervisor and the mentors on behalf of the Dutch Police. Monthly coordination meetings are planned to discuss progress and adapt this research plan based on emerging findings.

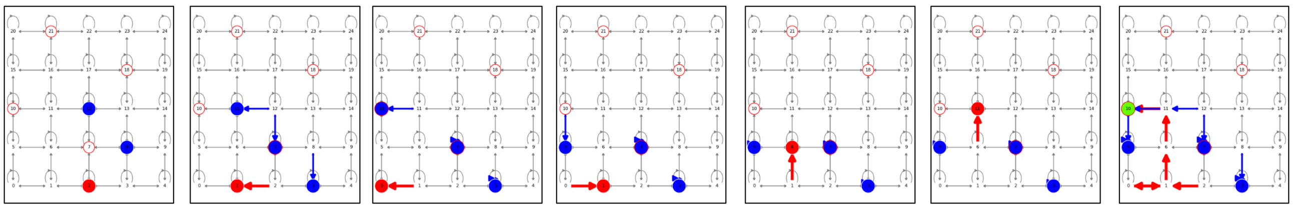


2021
2022
n
d
j
f
m
a
m
j
j
Work streams
Phase 0
writing
Preliminaries
• Formulation of research & approach
• Define modelling approach
• Develop simulation environment
• Literature review
Phase 1
Baseline experiments
• Use basic graph typologies and common RL methods
• Apply different levels of observability
• Investigate generalizability over initial conditions
Phase 2
Develop generalization over graphs
• Assume full observability
• Apply graph representation learning + RL
Phase 3
Scale-up to real road networks
• Assume full observability
• Address scalability issues if observed
Phase 3+
Extend to partial observability
• Incorporate real-life experience in setting sporadic information updates
Meetings
Progress/supervisor
Coordination group
Deliverables
Thesis contract
Thesis Proposal
Draft Thesis
Final Thesis

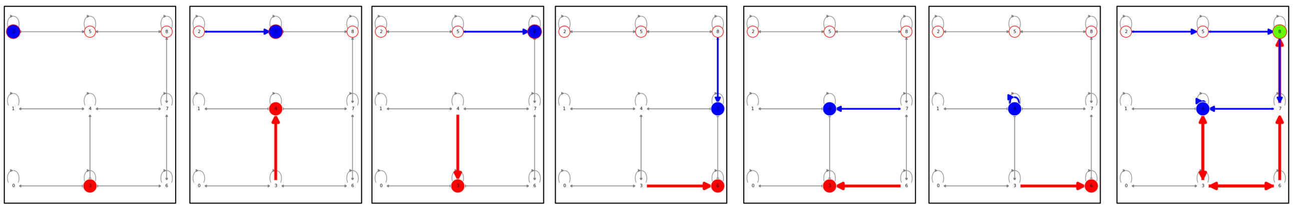
Figure 5.1: Thesis roadmap (subject to findings during execution

Appendix A

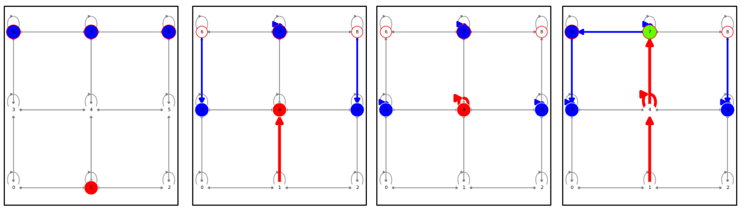
Example graphs & rollouts



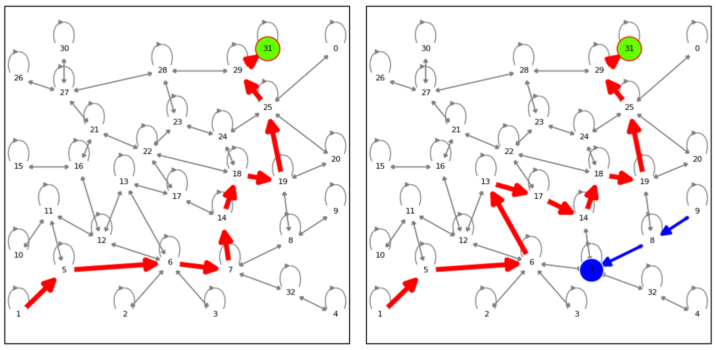
(a) 5x5 Manhattan graph, 4 random target nodes, U=3 static search units



(b) 3x3 Manhattan graph permutation with 3 edges removed, U=1



(c) 3x3 Manhattan graph, optimal solution requires memory (same observation, different actions)



(d) Example Metro graph, N=33 nodes, demonstrating an evasive action, one target node, U=1

Figure A.1: Illustration of basic graph types and rollouts. Passive search agent trajectories (in blue) are provided by a black box optimizer, escape routes (in red) are to be learned

Bibliography

[Als04] Brian Alspach. Searching and sweeping graphs: a brief survey. *Le matematiche*, 59(1, 2):5–37, 2004.

[BHB$^+$18] Peter Battaglia, Jessica Blake Chandler Hamrick, Victor Bapst, Alvaro Sanchez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andy Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Jayne Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018.

[BKU15] Ahmet Tunc Bilgin and Esra Kadioglu-Urtis. An approach to multi-agent pursuit evasion games using reinforcement learning. In *2015 International Conference on Advanced Robotics (ICAR)*, pages 164–169. IEEE, 2015.

[BPL$^+$16] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *CoRR*, abs/1611.09940, 2016.

[BRW$^+$19] Matthew Botvinick, Sam Ritter, Jane X. Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in Cognitive Sciences*, 23(5):408–422, 2019.

[CCK$^+$21] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4348–4355. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Survey Track.

[CHI11] Timothy H Chung, Geoffrey A Hollinger, and Volkan Isler. Search and pursuit-evasion in mobile robotics. *Autonomous robots*, 31(4):299–316, 2011.

[DDS16] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2702–2711, 2016.

[dSNC$^+$21] Cristino de Souza, Rhys Newbury, Akansel Cosgun, Pedro Castillo, Boris Vidolov, and Dana Kulić. Decentralized multi-agent pursuit using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):4552–4559, 2021.

[GEK17] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[GSR$^+$17] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1263–1272. JMLR.org, 2017.

[Ham20] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[HAN+19] Maximilian Hüttenrauch, Sosic Adrian, Gerhard Neumann, et al. Deep reinforcement learning for swarm systems. *Journal of Machine Learning Research*, 20(54):1–31, 2019.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HS15] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

[ICfIM65] R. Isaacs, Karreman Mathematics Research Collection, Society for Industrial, and Applied Mathematics. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization.* SIAM series in applied mathematics. John Wiley Sons, 1965.

[Kas20] Tamar Kastelein. Optimization of intercepting fugitives on the dutch highway. Master's thesis, Vrije Universiteit Amsterdam, School of Business and Economics, 2020.

[KDZ+17] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[KOQ+18] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

[KvHW19] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[KW17] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017.

[LWT+17] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[MMD+19] Sahil Manchanda, Akash Mittal, Anuj Dhawan, Sourav Medya, Sayan Ranu, and Ambuj Singh. Learning heuristics over large graphs via deep reinforcement learning. *arXiv preprint arXiv:1903.03332*, 2019.

[NMTG16] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[OAMAH15] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, and Jonathan P How. Decentralized control of partially observable markov decision processes using belief space macro-actions. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5962–5969. IEEE, 2015.

[Par78] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alavi and Don R. Lick, editors, *Theory and Applications of Graphs*, pages 426–441, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

[PCX21] Yun Peng, Byron Choi, and Jianliang Xu. Graph learning for combinatorial optimization: A survey of state-of-the-art. *Data Science and Engineering*, 6, 06 2021.

[SSP$^+$15] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, 2015.

[SYML18] Doo Re Song, Chuanyu Yang, Christopher McGreavy, and Zhibin Li. Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 311–318. IEEE, 2018.

[VBO$^+$20] Petar Veličković, Lars Buesing, Matthew Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. Pointer graph networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2232–2244. Curran Associates, Inc., 2020.

[VCC$^+$18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[vD21] I.S. van Droffelaar. Optimizing interception. Research Proposal, Faculty of Technology, Policy, Management, Delft University of Technology, 2021.

[Vel21] Petar Veličković. Theoretical foundations of graph neural networks, university of cambridge cst wednesday seminar 17 february 2021, https://petar-v.com/talks/gnn-wednesday.pdf, 2021.

[VFJ15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[VYP$^+$20] Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *International Conference on Learning Representations*, 2020.

[WFPS07] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. Solving deep memory pomdps with recurrent policy gradients. In *International conference on artificial neural networks*, pages 697–706. Springer, 2007.

[WW95] Alan Washburn and Kevin Wood. Two-person zero-sum games for network interdiction. *Operations research*, 43(2):243–251, 1995.

[XHG$^+$19] Lin Xu, Bin Hu, Zhihong Guan, Xinming Cheng, Tao Li, and Jiangwen Xiao. Multi-agent deep reinforcement learning for pursuit-evasion game scalability. In *Chinese Intelligent Systems Conference*, pages 658–669. Springer, 2019.

[XLZ$^+$20] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? In *International Conference on Learning Representations*, 2020.

[ZLPM17] Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On improving deep reinforcement learning for pomdps. *arXiv preprint arXiv:1704.07978*, 2017.