Graph coarsening for fugitive interception

van Droffelaar, Irene S.; Kwakkel, Jan H.; Mense, Jelte P.; Verbraeck, Alexander

DOI
10.1007/s41109-024-00689-1

Publication date
2025

Document Version
Final published version

Published in
Applied Network Science

Applied Network Science

RESEARCH Open Access

Graph coarsening for fugitive interception

Irene S. van Droffelaar[1*], Jan H. Kwakkel[1], Jelte P. Mense[2] and Alexander Verbraeck[1]

*Correspondence:
i.s.vandroffelaar@tudelft.nl

[1] Policy Analysis, Delft University of Technology, Jaffalaan 5, 2628 BX Delft, The Netherlands
[2] Model-Driven Decisions Lab, Delft University of Technology, Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Abstract

The police control room determines where to send available police units to intercept a fleeing fugitive. Models can support the police with decision-making for fugitive interception. The police have, at most, a few minutes to determine an interception strategy. Therefore, a timely calculation of the interception positions is essential to support police interception operations. The number of nodes in the network, each being a crossing where routes of the fleeing suspect can split, greatly contributes to the computation time. Graph coarsening is a promising approach to reduce the complexity of the network, and therefore the computation time. We compare four graph coarsening algorithms on five road networks and assess their impact on computation time and solution quality for the fugitive interception problem. Based on the comparison, we propose and test a new method specifically for fugitive interception. This method, Search Space Representation, improves the quality of the best solutions obtained by the optimization algorithm with up to 12%, improves the reliability of the optimization to find high-quality solutions, and decreases the number of function evaluations required to obtain high-quality solutions to 5000–10,000 depending on the size and complexity of the road network, which is feasible for real-time decision-making. Search Space Representation can be applied to reduce the computation time of other network-based optimization problems.

Keywords: Graph coarsening, Network topology, Fugitive interception, Search problem, Real-time optimization

Introduction

Fugitive interception is a challenging task, requiring police to decide in at most a few minutes on the optimal positions of police units to intercept a fleeing suspect. The fleeing fugitive moves from the incident location to escape interception, e.g., by crossing the border or reaching the highway. Police units do not know the fugitive's whereabouts, so they have to move to a vertex in the network where the probability of interception is highest, e.g., a chokepoint in the network where many routes pass through. The goal of the so-called 'fugitive interception problem' is to position the police units in such a way that they maximize the number of intercepted escape routes. Possible escape routes are simulated using a generative model of escape behavior (van Droffelaar et al. 2024). Note that the fugitive interception problem is not about chasing a fleeing fugitive; it concerns *intercepting* a fleeing fugitive, who is taking an unknown escape route from a known crime location. Models can support police decision-making for this problem, but the

timely calculation of optimal interception positions is challenging due to the complexity of the problem. A major contributor to the overall computation time is the size of the underlying road network, particularly the number of nodes in the network (van Droffelaar et al. 2024).

Graph coarsening, a technique to reduce the size of a graph while preserving essential structural properties, offers a promising approach to reducing the computation time of the fugitive interception problem. Graph coarsening is also referred to as contraction hierarchies (Geisberger et al. 2008), graph reduction (Loukas 2019), edge contraction (Asano and Hirata 1983), and graph sparsification (Peleg and Schäffer 1989). Graph coarsening algorithms have successfully been applied to various graph-based optimization problems where reducing the number of nodes significantly improves computation time, such as routing optimization (Sanders and Schultes 2012), the Traveling Salesman Problem (Walshaw 2004) and graph partitioning (Chevalier and Safro 2009).

Graph coarsening algorithms vary depending on the application, as the importance of the nodes and links is very case-specific. For example, coarsening for transport modeling often focuses on preserving the shortest paths (Sanders and Schultes 2012), while coarsening for graph partitioning aims to minimize the number of edges (Chevalier and Safro 2009; Safro et al. 2015). Pung et al. (2022) propose an algorithm that coarsens road networks using characteristics most prominent in the United States – grids and cul-de-sacs.

We distinguish two approaches to graph coarsening with different trade-offs between solution quality and computation time for fugitive interception: pre-processing and on-the-fly coarsening. The first approach coarsens networks in advance and saves them for later application. Hence, the computation time of the coarsening algorithm does not affect the computation time of any subsequent optimizations. However, the coarsening algorithms cannot take any incident-specific information into account, like the starting positions of police units, the starting position of the fugitive, or their plausible escape routes. Therefore, pre-processed coarsening risks removing nodes of high importance to any specific fugitive interception problem and, therefore, decreasing the solution quality. In contrast, on-the-fly coarsening approaches can take all relevant incident characteristics into account, likely leading to a higher solution quality. However, the computation time of the coarsening is critical in this case and might offset any gains in computation time for the optimization.

This paper compares four graph coarsening techniques on both computation time and solution quality for fugitive interception. We measure the solution quality by running the optimization algorithm for 100 000 function evaluations across 10 seeds and take the best-found solution. To measure the computation time, we consider (1) the number of function evaluations required by the optimization algorithm to find a solution, and (2) the computation time per function evaluation. We evaluate these methods across five different types of road networks. The evaluation studies both pre-processing and incorporating on-the-fly graph reconstruction into the optimization process.

The contribution of this research is two-fold: (1) we compare the effectiveness of existing graph coarsening algorithms for a new application, and (2) we propose an approach incorporating on-the-fly graph reconstruction into the Search Space Representation in the optimization process. This allows for more flexibility, capable of handling different fugitive profiles and network structures.

Section "Optimization problem"describes the case study used throughout the paper, fugitive interception optimization. Section "Method" describes the experimental setup. Section "Coarsening algorithms: results and discussion" details each of the chosen algorithms, their implementation for fugitive interception, and the results of the experiments. Based on the results, Sect. "Proposed method: search space representation" proposes and tests an on-the-fly coarsening approach for fugitive interception. Section "Discussion" discusses the possible threats to the validity and implications of the research, and, lastly, we share our conclusions in Sect. "Conclusion".

Optimization problem

Background

The fugitive interception problem aims to find the best positions for police units to maximize the probability of intercepting a fleeing fugitive on a road network. The problem is modeled from the start of the escape from the incident until the fugitive is either intercepted or escaped. Since the police do not know the fugitive's exact location, they have to position themselves at points in the network where there is a high chance of interception, such as chokepoints where many escape routes intersect.

Related optimization problems in literature are search, and, more specifically, interception problems. Alspach (2004) optimizes the routes of searchers to either maximize the probability or minimize the time to find a target. However, determining an optimal action for each time step becomes computationally infeasible when the size of the network and the length of the time horizon increase.

Pursuit-evasion games, where both the routing of the searcher (or pursuer) and the target (or evader) are optimized, are primarily used in robotics (Chung et al. 2011). Pursuit-evasion problems are solved for different graph topologies, such as grids, circular graphs, trees, and random graphs. Depending on the network topology, these problems are proven to be pseudo-P to strongly NP-complete (Borie et al. 2011). Due to the computational complexity, the problem instances that are studied are typically very small.

Formalization of the optimization problem

We formulate the optimization problem as a variant of the Flow Interception Problem (Hodgson 1990; Berman et al. 1992), meaning the objective is to position each police unit to maximize the number of intercepted escape routes. A route is considered intercepted if (1) it passes a police unit's target position and (2) the police unit can reach that position before the fugitive does.

The decision variables of the optimization problem are the target nodes of the police units ($\pi_{u,v}$) and the intercepted routes (z_r) (Table 1). The optimization problem is formalized in Eqs. 1-3. A route is considered intercepted ($z_r = 1$) if, for a given route (r), the fugitive is at the same place (v) at the same time (t) as the position of a police unit ($\pi_{u,v}$), and that position is within reach at that time for that particular police unit ($\tau_{u,v,t}$). The positions of the police units are optimized to maximize the number of intercepted escape routes. Routes are only intercepted at target positions, not at intermediate points along the route. The minimization function in Eq. 3 ensures that each route can only be intercepted once and contribute to the objective function.

Table 1 Notation of parameters and decision variables

Decision variables	
$z_r \in \{0, 1\}$	Route r is intercepted
$\pi_{u,v} \in \{0, 1\}$	Node v is the target node of police unit u
Parameters	
$V = \{v\}$	Set of nodes
$R = \{r\}$	Set of fugitive routes
$U = \{u\}$	Set of police units
$S = \{s\}$	Set of sensors
$T = \{t\}$	Ordered index set of time steps
t_{max}	Maximum time step; length of planning horizon
$\phi_{r,v,t} = \{0, 1\}$	Fugitive route r is present at node v at time step t
$\tau_{u,v,t} = \{0, 1\}$	Node v is reachable by police unit u at time t

$$\text{Maximize: } Z = \sum_{r \in R} z_r \tag{1}$$

$$\text{Subject to: } \sum_{v \in V} \pi_{u,v} = 1 \quad \forall u \in U \tag{2}$$

$$z_r = \min\left(1, \sum_{u \in U} \sum_{t \in T} \sum_{v \in V} \phi_{r,v,t} \cdot \pi_{u,v} \cdot \tau_{u,v,t}\right) \quad \forall r \in R \tag{3}$$

Simulation of the fugitive escape routes

The optimization depends on the simulated escape routes of the fugitive. The routes are modeled as the shortest paths from the incident location to the escape nodes of the network. To generate a diverse set of plausible routes, 2% noise is added to the routes, meaning that the fugitive makes a wrong turn at 2% of the intersections, after which a new shortest path is recalculated from their current position (van Droffelaar et al. 2024). This approach produces a distribution of routes around the optimal paths. Simulating the fugitive escape routes through this method takes a few seconds, depending on the road network, the starting position of the fugitive, and the locations of the escape nodes (Winterswijk: 2.4 s, Utrecht: 9.9 s, Manhattan: 1.4 s, Main roads: 1.1 s, Rotterdam: 9.2 s). The simulation of fugitive routes should be further optimized and parallelized before implementation in a control room. In future work, this model could be replaced with a more detailed behavioral model.

Solution approach

The optimization problem is NP-hard, meaning that exactly solving real-world cases could take years (Boccia et al. 2009). Therefore, we use a genetic algorithm supplemented with the auto-adaptive framework Borg, which co-evolves the probabilities of the evolutionary operators used for population adaptation based on their success in generating better solutions (Hadka and Reed 2013). van Droffelaar et al. (2024) compare this metaheuristic optimization algorithm to the exact optimization algorithm CBC (Ralphs

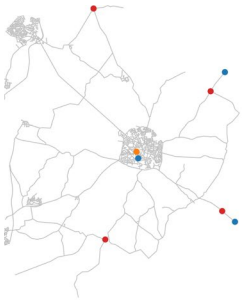
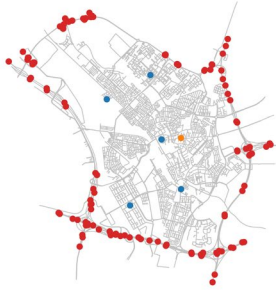
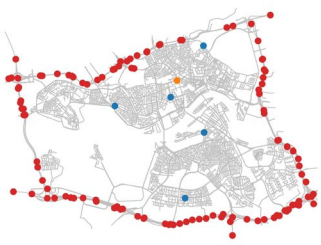
2022) and show that the metaheuristic finds near-optimal solutions in a fraction of the computation time. Problem instances on networks with 2 500 nodes are solved in 5–10% of the computation time, and the computation time increases less rapidly with increasing network size. To speed up convergence, the nodes are sorted on their proximity to the starting position of the fugitive. Thus, the proximity of solutions in the search space is more related to proximity in the objective space. To further ensure solution quality, the optimization algorithm is run for ten random seeds for 100 000 function evaluations. All experiments are performed on the Delft Blue supercomputer, with an Intel XEON E5-6248R 24C 3.0 GHz CPU with 48 cores and 192 GB memory (Delft High Performance Computing Centre 2022).

Method

Case study networks

To reduce the dependency of the experiments on the topology of the road network, we evaluate the coarsening approaches on five distinct road networks (see Table 2 for an overview). First, Winterswijk, the Netherlands, represents a rural area. Sparse roads lead from the town to the border with Germany in the north, east, and south. Second, Manhattan, New York, United States of America, represents a grid layout city with traffic lights and cameras at most intersections. Third, Utrecht, The Netherlands,

Table 2 Case study road networks used in this study. Escape nodes are marked in red. The starting positions are displayed in blue (police units) and orange (fugitive)

Winterswijk	Manhattan	Utrecht
Rural area Incident: city center Escape: cross the border Size: 1926 nodes	Modern, grid city Incident: Union Square Escape: get off the peninsula Size: 2533 nodes	Typical European city Incident: city center Escape: reach the highway Size: 4557 nodes
		
Main road network	**Rotterdam**	
Typical highway network Incident: Amsterdam Escape: network edges Size: 3241 nodes	Modern European city Incident: city center Escape: reach the highway Size: 7108 nodes	
		

represents a typical European city with a historical center surrounded by more modern neighborhoods. Fourth, the main road network around Amsterdam, The Netherlands, consists of highways, and primary and secondary roads around the city. This network is vastly different from city road networks and represents an application where a fugitive flees at high speeds over a larger distance. Fifth, Rotterdam, The Netherlands, represents a large modern European city divided by a large river. The starting position of the fugitive is a central location in each road network, and the police start locations are the local police stations in the respective areas.

The networks are obtained from OpenStreetMap via the OSMnx Python library (Boeing 2017, 2024). We use the built-in `simplify_graph` functionality to remove nodes that do not represent intersections, as well as dead ends. The OpenStreetMap raw data consists of sets of straight-line segments: curved roads contain intermediate nodes to represent their geometry. For fugitive interception, these nodes do not add any value but do increase the number of nodes considerably. For example, the unsimplified Winterswijk network consists of 9540 nodes, whereas simplifying the network reduces the number of nodes to 1926.

Evaluation method

Each coarsening algorithm is evaluated on each of the road networks using the framework depicted in Fig. 1. The evaluation method consists of four steps:

1. The fugitive escape routes are simulated on the uncoarsened road network G using the method described in Sect. .
2. The road network G is coarsened using the algorithm under evaluation, resulting in the coarsened network G_c. Each coarsening algorithm has its own tuning parameters, which are varied to obtain different extents of coarsening for each algorithm.
3. The police interception positions are optimized based on the coarsened network G_c, meaning that only nodes that remain in G_c are considered possible interception positions. The number of routes intercepted by a combination of interception positions – the quality of the candidate solution – is calculated using the set of routes generated in step 1.
4. The police interception positions optimized based on G_c are then evaluated on the original graph G. Discrepancies in the number of intercepted escape routes arise when either the path between the police start position and the calculated intercep-

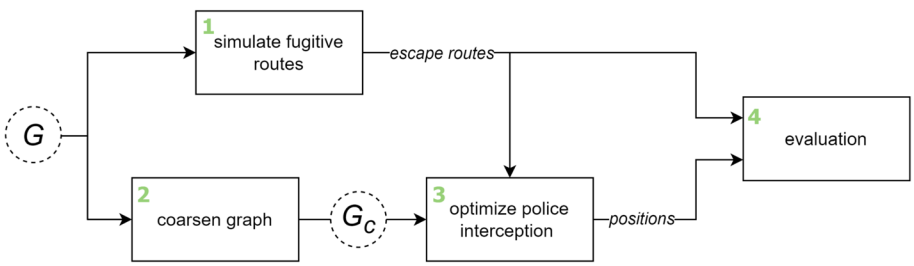


Fig. 1 Schematic representation of the evaluation method used in this study

tion position is longer in G than in G_c, or when the path does not exist in G_c or in G. We collect three metrics from each parametrization of each coarsening algorithm:

- The *solution quality*, which is the percentage of escape routes intercepted by the calculated police interception positions. Note that the metaheuristic solution approach does not guarantee finding the exact optimal solution. Therefore, we take the best-found solution after 100 000 function evaluations across 10 seeds to account for variations in convergence between seeds. The solution quality is scaled to the best-found solution quality on the uncoarsened graph G to obtain the degradation of the solution quality caused by the coarsening of the graph.
- The *convergence*, which is the number of function evaluations at which the optimization algorithm obtains 95% of its solution quality. Effectively, this is the number of function evaluations at which the search stalls. To account for variation between seeds, we take the minimum number of function evaluations at which a seed reaches 95% of its best-found solution quality. This statistic is collected for every seed of each parametrization of each coarsening algorithm.
- The *time per function evaluation*, which, combined with convergence, indicates how much graph coarsening reduces the computation time. For these experiments, we rerun a subset of the coarsening algorithm parametrizations on a dedicated node of the supercomputer to prevent interference with other jobs.

Coarsening algorithms: results and discussion

We compare the effectiveness of four graph coarsening algorithms for fugitive interception: three preprocessing algorithms and one on-the-fly approach. The three preprocessed graph coarsening algorithms are selected for their diverse approaches to graph coarsening and the availability of open-source implementations (preferably in Python). The on-the-fly coarsening method was developed as part of this research. The algorithms considered are:

1. *Pruning*: a first step to simplifying the network, by removing dead ends and cul-de-sacs (Pung et al. 2022).
2. *Node consolidation*: a generic, application-agnostic graph coarsening approach that merges nodes that are topologically close together to simplify complex intersections and clusters in the network (Boeing 2024).
3. *Heuristic coarsening*: a transport-specific coarsening algorithm that preserves key properties of the network, such as connectivity and shortest paths (Krishnakumari et al. 2020).
4. *On-the-fly coarsening*: a case-specific coarsening algorithm that filters the network before optimization, retaining only the nodes and edges that are relevant to the specific interception case.

The following paragraphs outline the four graph coarsening algorithms, first discussing them in general, second detailing the implementation for the fugitive interception case

used throughout this paper, and third, presenting the results. Lastly, the computation time and obtained solution quality of the algorithms are compared.

Pruning

Background

A logical first step to simplify the road network is pruning dead ends and cul-de-sacs (Pung et al. 2022). Dead ends and cul-de-sacs are not expected to be nodes with a high probability of intercepting a fleeing suspect. Therefore, pruning these nodes should not degrade the solution quality by much.

Implementation

This study uses a simple recursive algorithm that removes nodes with only one incoming or outgoing edge. Self-loops, which are edges that connect a node to itself, are also removed. The recursion ensures that any new dead ends or self-loops created during the pruning process are also removed.

Results and discussion

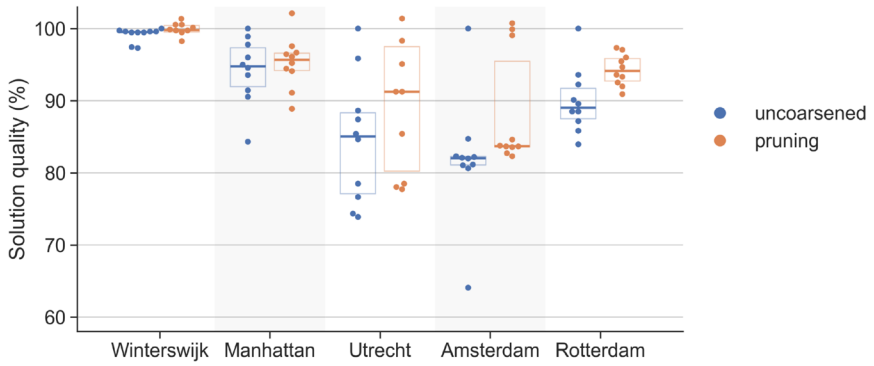
Depending on the road network, the number of nodes in the network is reduced by 2.7% to 29.1%, depending on the road network (Table 3). This is a considerable reduction, especially because the removal of these nodes does not affect the solution quality. Figure 2a shows that, relative to the uncoarsened graph, pruning obtains the same best-found solution quality across seeds. Notably, the mean obtained solution quality increases for all road networks. Note that for some seeds, the solution quality exceeds 100%, indicating that the metaheuristic solution approach identifies solutions better than the best-known solutions for the uncoarsened network. This occurs because metaheuristics do not guarantee finding the exact optimal solution. The reduction in the number of nodes available for interception decreases the size of the solution space, which speeds up convergence and reduces the likelihood of getting stuck in local optima.

Additionally, the variation between seeds either stays similar or decreases. This is important for the predictability of computation time for the decision-maker. A large variation in solution quality across seeds indicates that a decision-maker should run with many seeds in parallel and aggregate the results.

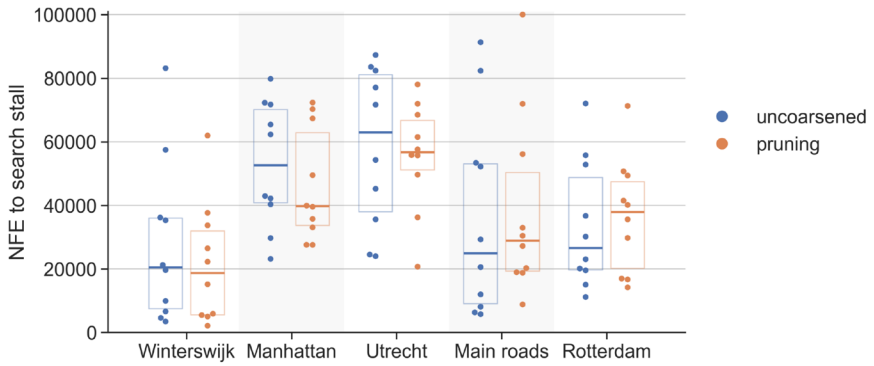
Figure 2b shows that the convergence of the optimization algorithm is largely dependent on the seed and the initial sample of solutions. On average, pruning leads to slower

Table 3 Node reduction by pruning for the five case study road networks, relative to the uncoarsened networks

City	Node reduction (%)
Winterswijk	29.1
Manhattan	2.7
Utrecht	15.1
Main roads	5.9
Rotterdam	13.0



(a) Impact of pruning on the solution quality. The results are scaled to the best-found solution quality across seeds for the uncoarsened network.



(b) Impact of pruning on the convergence. The NFE to search stall is the NFE at which that seed reached 95% of its best-found solution quality.

Fig. 2 Results of the pruning experiments with ten seeds per city. Each dot represents the results of one random seed. The overlaid boxes represent the first quartile, median, and third quartile

convergence in most road networks. This, however, is a distortion of the plot due to the much higher obtained solution quality (Fig. 2a). The earliest NFE at which the optimization algorithm finds a solution with the quality of the uncoarsened network solution is lower after pruning.

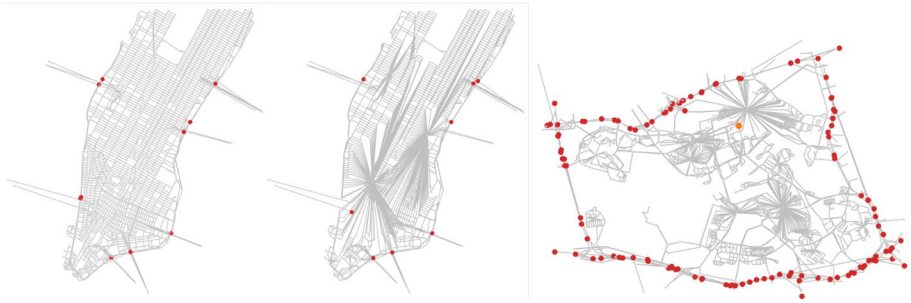
Node consolidation

Background

Node consolidation is a graph simplification method that merges nodes that are topologically close together (Boeing 2024). Real-world road networks often have complex intersections that, when converted to a graph, result in a group of nodes. For instance, a roundabout is represented by four or eight nodes, depending on its specific layout. For transport planning – and for fugitive interception – we can consider these multiple nodes as one.

Implementation

This study uses an OSMnx's node consolidation algorithm (Boeing 2024). The algorithm's tuning parameter, 'tolerance' (measured in meters), defines the buffer radius around each node. Overlapping node buffers are merged into a single node at the center



(a) Manhattan (10m). (b) Manhattan (45m). (c) Rotterdam (50m).

Fig. 3 Road networks after node consolidation with different tolerance settings (in brackets)

Table 4 Node reduction by node consolidation for the five case study road networks, relative to the uncoarsened networks, for a tolerance value of 5, 25 and 50

	Node reduction (%)		
Tolerance	**5**	**25**	**50**
Winterswijk	24.1	58.2	89.9
Manhattan	3.4	25.0	93.7
Utrecht	17.5	55.0	90.2
Main roads	14.3	52.7	61.2
Rotterdam	10.0	53.6	89.9

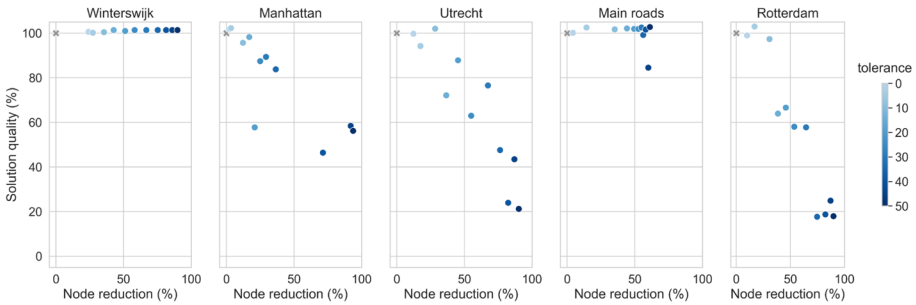
of the buffer area. In this study, we vary the tolerance from 1 to 50 ms. A low tolerance simplifies complex intersections into single nodes (Fig. 3a). A higher threshold collapses more of the network, but keeps the main roads intact (Fig. 3c). A grid network, like Manhattan, collapses after surpassing a tolerance threshold of the distance between the blocks (Fig. 3b).

The node consolidation algorithm retains information about which original nodes were consolidated into each new node. After consolidation, the starting positions of the police, the fugitive, and the escape nodes are mapped to the corresponding consolidated nodes. By handling this in post-processing, rather than exempting certain nodes during consolidation, the coarsened network is flexible to any incident location.

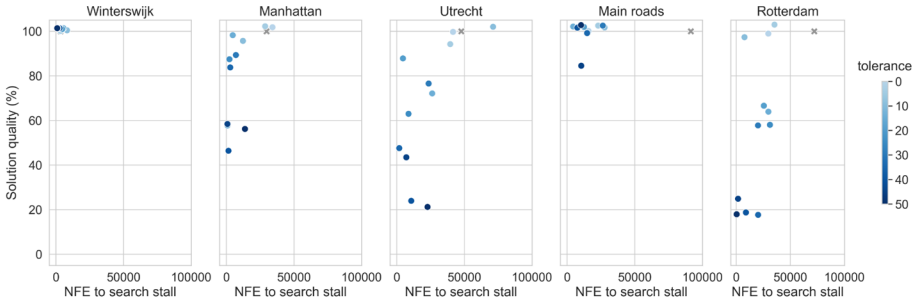
Results and discussion

Table 4 show that varying the 'tolerance' parameter produces coarsened networks with varying numbers of nodes. The exact impact of tolerance differs across networks. When the tolerance is set to its maximum of 50 ms, some networks are reduced to as little as 7% of their original size.

To maintain clarity, Fig. 4 only shows the best solutions across seeds. The solution quality varies a lot between seeds, which improves with coarsening but still makes it difficult to observe clear trends. For all networks except Winterswijk, there are some outliers due to the optimization algorithm's sensitivity to the random seed, even across 10 seeds.



(a) Impact of node consolidation on the solution quality. The results are scaled to the best-found solution quality across seeds for the uncoarsened network.



(b) Impact of node consolidation on the convergence. The NFE to search stall is the NFE at which that seed reached 95% of its best-found solution quality.

Fig. 4 Results of the node consolidation experiments, showing only the best result across seeds for each setting of the tolerance parameter. The grey crosses indicate the best result for the uncoarsened graph

The impact of network size reduction on solution quality varies between networks (Fig. 4a). For Winterswijk and the main road network, node consolidation has little to no effect on solution quality. The key interception positions and the paths from the starting positions are preserved. In Manhattan, we see a jump in node reduction once the tolerance exceeds the spacing between streets, which results in a large drop in solution quality. For Rotterdam, we see two of these drops: the first consolidations barely affect the solution quality, which then drops to approximately 60%, and later to 20%. For Utrecht, we see a more gradual but similarly dramatic decline in solution quality.

Figure 4b shows that reducing the number of nodes considerably decreases the number of function evaluations to search stall. The results for Winterswijk and the main road network are particularly interesting as the convergence speed is improved considerably without a loss in solution quality.

Heuristic coarsening

Background

Krishnakumari et al. (2020) propose a heuristic coarsening algorithm tailored to applications in transportation. The algorithm preserves key properties such as graph connectivity, shortest paths, and trip length distribution, making it a promising coarsening algorithm for fugitive interception.

The algorithm consists of four coarsening steps, repeated until either the maximum number of iterations is reached or the network cannot be coarsened further using the current settings.

1. Assign weights to the links in the graph. The weights can be based on properties relevant to the application, such as link length, road type, or speed.
2. Rank the nodes for removal. Instead of selecting nodes randomly, as is common in other coarsening algorithms, nodes are ranked deterministically for removal. This ensures reproducibility and reduces the computational time required for coarsening.
3. Contract and prune nodes. To avoid an excessive increase in the average node degree in the coarsened network (Geisberger et al. 2008), the algorithm applies a contraction criterion. Only nodes meeting this criterion are contracted. The tuning parameter 'threshold' (ρ) determines how strictly the criterion is followed, with higher values resulting in greater node reduction. Another parameter, pruning, removes dead ends, self-loops, and disconnected components.
4. Update the link weights. After contracting and pruning, link weights are recalculated for the coarsened graph, and steps 2–4 are repeated until the stopping criterion is reached.

Implementation

This study applies the heuristic coarsening algorithm from Krishnakumari et al. (2020), originally implemented in Matlab, which we have re-implemented in Python.[1]

We conducted two sets of experiments. In the first set, we used the default settings, contracting nodes based on road type. These road types are crowdsourced in OpenStreetMap and range from 'pedestrian path' and 'bus lane' to 'motorway'. Nodes that serve as a connection between different types of roads (e.g., a highway and a residential street) are not contracted, assuming these nodes are important for the network's connectivity (Krishnakumari et al. 2020). In the second set of experiments, nodes are contracted based on their betweenness centrality. Nodes with a high betweenness centrality are preserved, assuming that these nodes are important for network connectivity and for fugitive escape routes.

The coarsening can be pre-processed, so the computation time of the coarsening algorithm does not affect the real-time performance of the decision support system. If the police starting positions, fugitive starting position, or escape nodes are removed during coarsening, the shortest paths from these positions to the coarsened network are added back to the network afterward. This post-processing step makes it possible to handle any incident location.

Both variants of the algorithm are tested with the same parameter settings as in Krishnakumari et al. (2020). We run the algorithm both for a single iteration and until completion, with thresholds set to either the minimum (0) or maximum (1000). We experiment with and without pruning.

Results and discussion

When coarsening the network based on road type, the node reduction is relatively limited (Table 5), but the solution quality declines quickly. Even for Winterswijk and the main road network, we see a large degradation of solution quality, while other coarsening

[1] The Python implementation can be found at https://github.com/irene-sophia/HeuristicCoarsening

Table 5 Node reduction by heuristic coarsening (type) for the five case study road networks, relative to the uncoarsened networks

	Node reduction (%)			
Pruning	**0**	**0**	**1**	**1**
Iterations	**1**	**Max**	**1**	**Max**
Threshold	**0**	**1000**	**0**	**1000**
Winterswijk	1.1	3.1	30.0	36.2
Manhattan	4.2	20.8	10.9	32.3
Utrecht	5.9	15.3	25.8	42.0
Main roads	38.6	55.7	47.5	74.5
Rotterdam	7.0	18.3	25.2	43.8

Table 6 Node reduction by heuristic coarsening (betweenness) for the five case study road networks, relative to the uncoarsened networks

	Node reduction (%)			
Pruning	**0**	**0**	**1**	**1**
Iterations	**1**	**Max**	**1**	**Max**
Threshold	**0**	**1000**	**0**	**1000**
Winterswijk	72.9	73.7	79.6	82.7
Manhattan	64.4	71.8	65.5	75.9
Utrecht	35.9	45.7	44.8	62.8
Main roads	54.4	71.6	56.8	83.6
Rotterdam	0.1	18.3	11.5	43.8

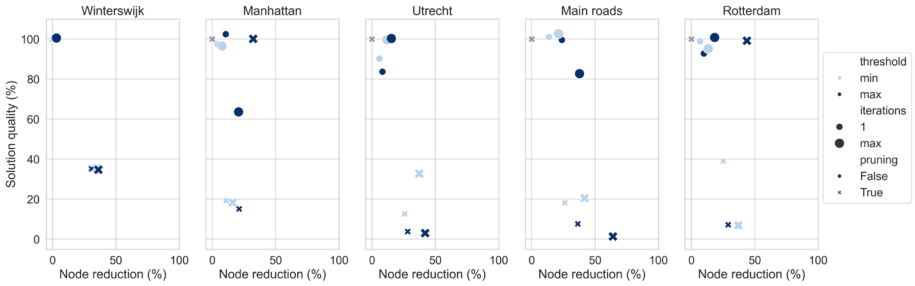
algorithms had less problems with these networks. Pruning, in particular, causes a drastic decline in solution quality to 5–40 % of the original quality (Fig. 5a). Again, the node reduction does considerably speed up the convergence, while obtaining poor solutions (Fig. 5b).

Using betweenness centrality improves the heuristic coarsening algorithm for fugitive interception (Fig. 6). The solution quality for Utrecht, Rotterdam, and the main road network is affected little by the coarsening. For Winterswijk, the size of the road network is only reduced very little without pruning (Table 6). However, enabling pruning causes a sharp decline in solution quality. Similarly, pruning leads to very poor solution quality for Manhattan. Additional analysis shows that, while the original interception positions are largely preserved, the paths from the police starting positions to the interception positions are not. The node reduction does considerably speed up the convergence (Fig. 6b).

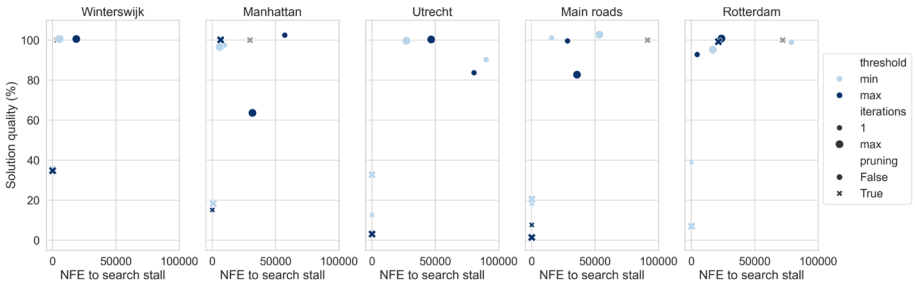
On-the-fly coarsening

Background

Instead of generic coarsening algorithms that allow for pre-processing, it is also possible to construct the road network on the fly for each optimization run. No important

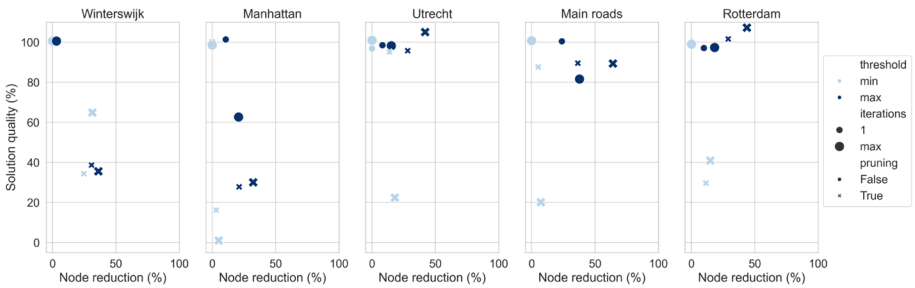


(a) Impact of heuristic coarsening on the solution quality. The results are scaled to the best-found solution quality across seeds for the uncoarsened network.

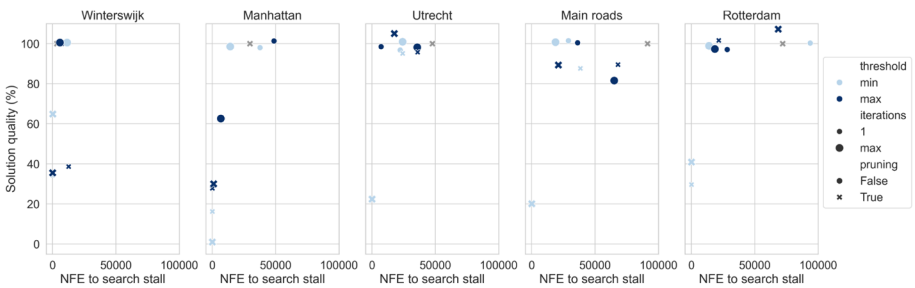


(b) Impact of heuristic coarsening on the convergence. The NFE to search stall is the NFE at which that seed reached 95% of its best-found solution quality.

Fig. 5 Results of the heuristic coarsening (road type) experiments, showing only the best result across seeds for each parameter setting. The grey crosses indicate the best result for the uncoarsened graph



(a) Impact of heuristic coarsening on the solution quality. The results are scaled to the best-found solution quality across seeds for the uncoarsened network.



(b) Impact of heuristic coarsening on the convergence. The NFE to search stall is the NFE at which that seed reached 95% of its best-found solution quality.

Fig. 6 Results of the heuristic coarsening (betweenness centrality) experiments, showing only the best result across seeds for each parameter setting. The grey crosses indicate the best result for the uncoarsened graph

Table 7 Node reduction by on-the-fly network construction for the five case study road networks, relative to the uncoarsened networks

City	Node reduction (%)
Winterswijk	70.8
Manhattan	45.0
Utrecht	52.3
Main roads	54.2
Rotterdam	52.9

nodes or paths are lost, only the unimportant parts of the network for each specific case are removed. The on-the-fly coarsening method was developed as part of this paper. To our knowledge, no research exists that implements this network representation, though it could be a promising approach, especially for Flow Interception Problems.

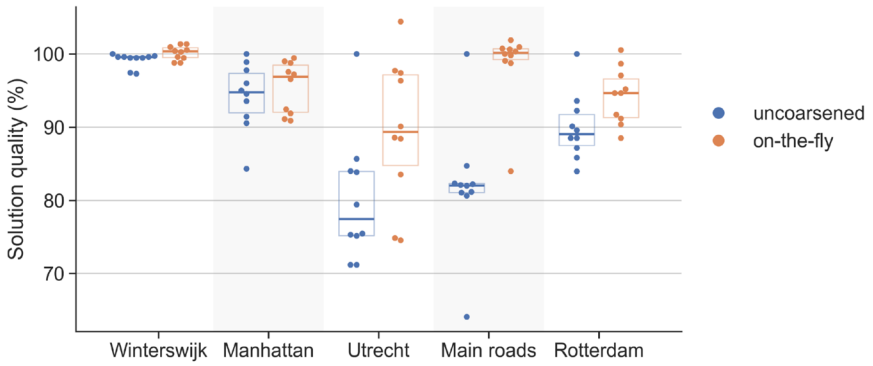
Implementation

A new network is created from the simulated escape routes and the shortest paths from the police starting positions to any node on these escape routes. The network reconstruction takes a few seconds, depending on the number of nodes in the network (Winterswijk: 0.22 s, Utrecht: 3.74 s, Manhattan: 1.12 s, Main roads: 0.84 s, Rotterdam: 5.89 s).

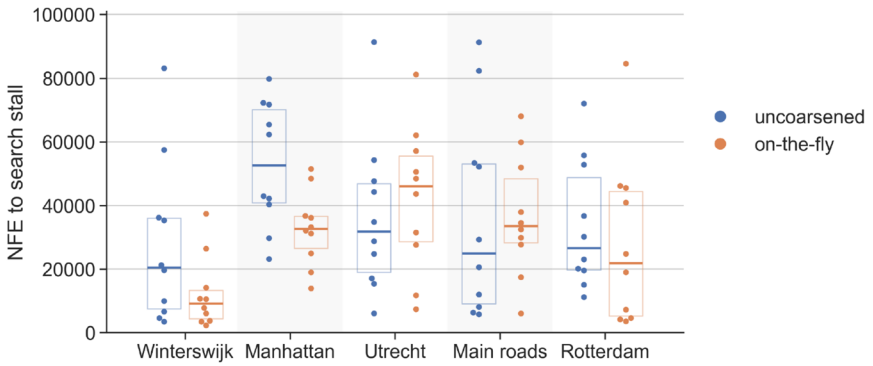
Results and discussion

Table 7 shows that removing unimportant parts of the network results in a considerable reduction in the number of nodes, ranging from 45.0 to 70.8%. This reduction improves the average solution quality across all networks (Fig. 7a). By removing nodes that do not lie on escape routes or police paths, many possible combinations of police interception positions that do not lie on any fugitive route (with a solution quality of 0) are removed.

The number of function evaluations to search stall is lower for Winterswijk, Manhattan, and Rotterdam – networks with a smaller increase in solution quality from on-the-fly network reconstruction. In these cases, the node reduction primarily speeds up the search. On the other hand, for Utrecht and Manhattan, where the network reconstruction leads to a more substantial improvement in solution quality, the optimization requires more function evaluations. However, even in these cases, the optimization algorithm reaches a solution quality equivalent to the uncoarsened network earlier in the process. Additionally, the earliest NFE at which the search stalls (across multiple seeds) is lower, indicating a more efficient search despite the greater number of evaluations required for significant improvements. However, the minimum NFE at which the optimization algorithm finds a solution with the same quality as the uncoarsened network is lower. Additionally, across seeds, the earliest point at which the search stalls is also reached sooner, indicating that the network reconstruction not only improves solution quality but also speeds up convergence.



(a) Impact of on-the-fly network construction on the solution quality. The results are scaled to the best-found solution quality across seeds for the uncoarsened network.



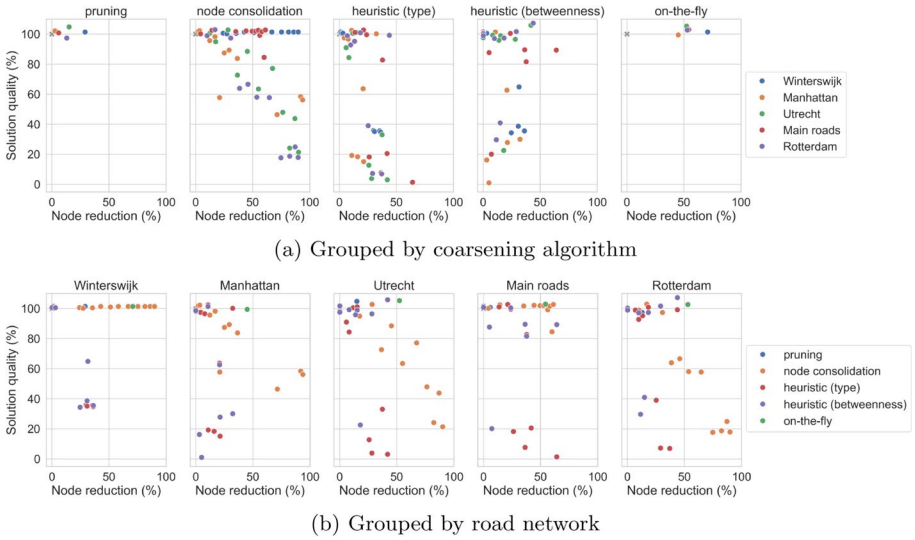
(b) Impact of on-the-fly network construction on the convergence. The NFE to search stall is the NFE at which that seed reached 95% of its best-found solution quality.

Fig. 7 Results of the on-the-fly network construction experiments

Comparison

Figure 8 combines the results, grouped by approach and by city. Appendix presents a comparison of the results in tabular form. Evidently, pruning and, to a larger extent, on-the-fly network reconstruction reduce the size of the network while achieving the same or higher solution quality. The effectiveness of the other coarsening algorithms varies by network. Node consolidation and heuristic coarsening (based on betweenness centrality) perform well for Winterswijk and the main road network. Node consolidation shows a more gradual decline in solution quality, whereas heuristic coarsening – particularly when using the road type in the algorithm – shows a larger degradation, even at low node reduction. This is surprising since heuristic coarsening was specifically designed for transportation networks.

Figure 9 shows the results of the timing experiments. The relationship between the number of nodes and the time per function evaluation generally follows a power-law trend. On-the-fly network reconstruction is an exception since many low-quality solutions are removed from the set of possible interception positions. For every function evaluation, the optimizer first checks whether a path exists between the police starting position and the candidate interception position. If that path exists, the length of the shortest path is calculated. That second step obviously adds to the computation time.



(a) Grouped by coarsening algorithm

(b) Grouped by road network

Fig. 8 Comparison of coarsening algorithms. The solution quality is scaled to the best-found solution quality across seeds for the uncoarsened network

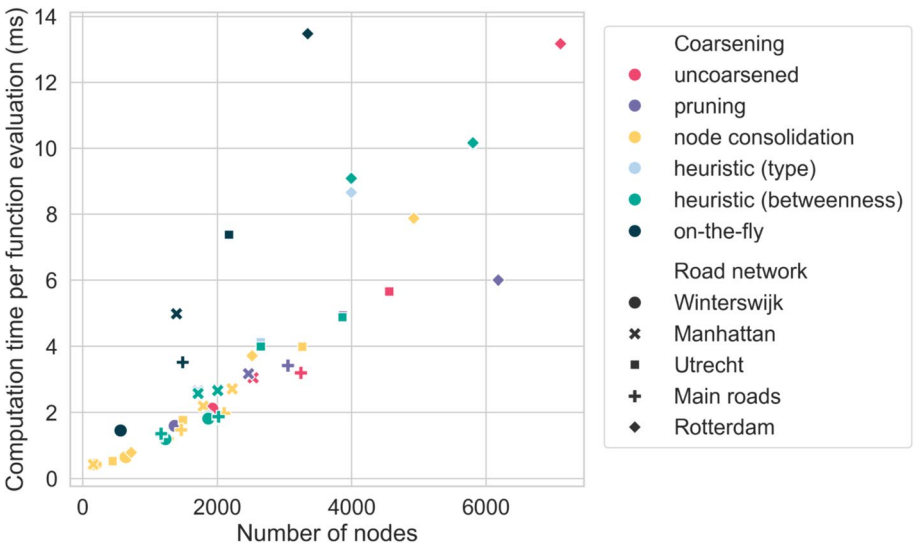
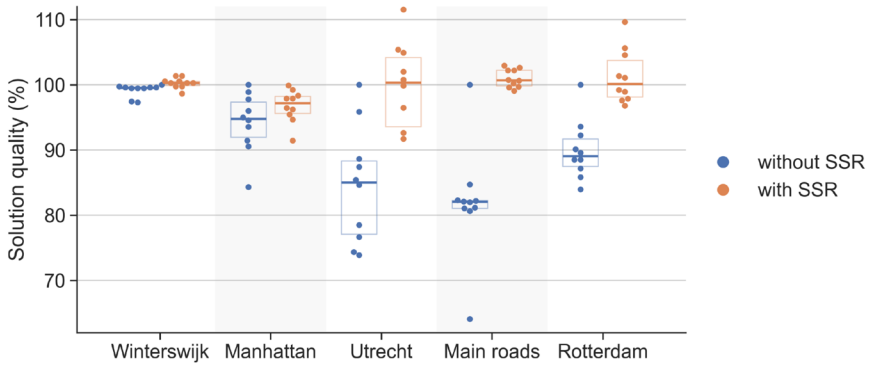


Fig. 9 The computation time per function evaluation across different graph coarsening approaches and road networks

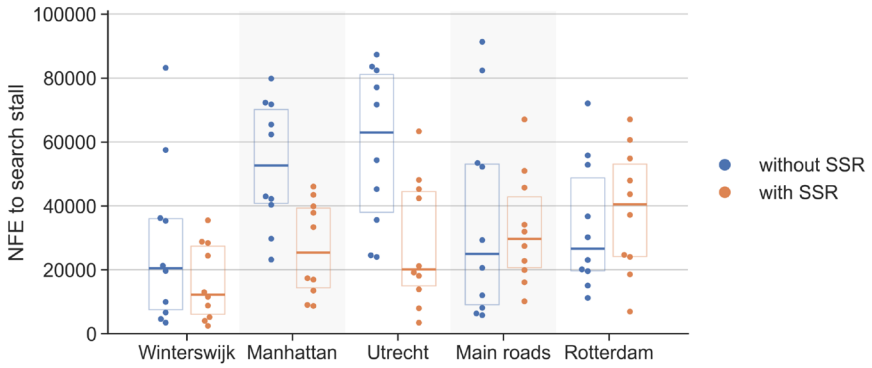
In other words, a network with many infeasible solutions shows a shorter computation time, though the solution quality is poor. The experiments presented in Fig. 7 have shown that on-the-fly network reconstruction significantly reduces the number of function evaluations to convergence.

Proposed method: search space representation

The experiments show that pruning and on-the-fly network reconstruction effectively reduce network size, while preserving solution quality. This, in turn, reduces the NFE to search stall, and the time per function evaluation. In other words, the computation time is reduced in two ways.



(a) Impact of the proposed method (SSR) on the solution quality. The results are scaled to the best-found solution quality across seeds for the uncoarsened network.



(b) Impact of the proposed method (SSR) on the convergence. The NFE to search stall is the NFE at which that seed reached 95% of its best-found solution quality.

Fig. 10 Results of the proposed method

Drawing inspiration from Bode et al. (2019), we incorporate on-the-fly network construction in the representation of the search space, filtering out low-quality solutions. The remaining solutions are the combinations of police unit positions that a) are located on at least one escape route and b) can be reached by the respective police unit. intro-duces variability in the number of potential positions for different police units, depending on their initial locations. Compared to on-the-fly network reconstruction, the proposed Search Space Representation further reduces the size of the optimization problem by removing unreachable positions.

Figure 10 shows that this approach is generally effective in achieving a higher-quality solution using fewer function evaluations. For all case study road networks, the best-found solution quality increased, up to 12%. Especially for Utrecht, the Main roads and Rotterdam, both the average and best solution quality across seeds increased dramatically (Fig. 10a). Increasing the average obtained solution quality improves the reliability of the optimization for the decision-maker. Since there are no possible solutions that have a solution quality of 0 (not intercepting any escape routes), the algorithm gets stuck less often in local optima and therefore converges to a solution with a higher quality. For Winterswijk, Manhattan and Utrecht, these high-quality solutions are also found in fewer function evaluations. For Rotterdam, the average number of function evaluations

until the search stalls is higher, but the seed with the quickest convergence converges at a lower number of function evaluations than without the Search Space Representation (SSR). For the Main roads, the slower convergence is due to the significantly higher obtained solution quality: the earliest NFE at which the optimization algorithm finds a solution with the quality of the solution without the SSR is lower (Fig. 10b). Across networks, the number of function evaluations required to obtain high-quality solutions is reduced to 3 000 -10 000 depending on the size and complexity of the road network. Considering that the time per function evaluation is 2–13 ms depending on the size of the network, this number of function evaluations is feasible for real-time decision-making.

Filtering the search space does not add to the overall computation time. In the optimization, the nodes are sorted on their proximity to the fugitive starting positions to speed up convergence (also in the uncoarsened case, Sect.). While the filtering step takes time, this is compensated because the number of nodes to be sorted is shorter. Depending on the network, this means that introducing the filtering step adds up to 0.3 s (Rotterdam) to reducing the computation time by 0.3 s (Utrecht), or does not impact the computation time (Winterswijk, Manhattan, Main roads).

Discussion

The effectiveness of graph coarsening algorithms for fugitive interception is dependent on the topology of the road network. Networks with dominant interception positions, like a highway network or Winterswijk, are relatively easily coarsened without degrading the solution quality. For other networks it proves difficult to find a general coarsening algorithm that both reduces the size of the network (and thus the computation time), while preserving the solution quality, i.e., the interception positions with a high probability of interception.

This research used a shortest-path model with noise to generate fugitive escape routes. Alternative models of fugitive behavior, such as avoiding busy roads, could affect the effectiveness of the coarsening algorithms that use preprocessing. The choice of escape nodes is also crucial; in this research, they are set at network boundaries like highway on-ramps or border crossings. If escape nodes were instead located at places like parking garages, the likely interception points would change, which could affect the effectiveness of the coarsening algorithms to different extents. In contrast, on-the-fly network reconstruction maintains the solution quality regardless of the fugitive escape routes. The solution quality is not affected by different models of fugitive behavior, but the reduction in computation time decreases when the number of nodes visited by the fugitive increases.

The data quality of open-source road networks influences experiments with graph coarsening. Since OpenStreetMap data is crowd-sourced, errors occur in network topology and attributes, such as road classification or speed limits. For example, in our research, we found a roundabout where one section was labeled as 'unclassified', while the rest was labeled as a 'residential road'. The heuristic coarsening algorithm that relies on the road classification to determine which nodes to contract, therefore produces incorrect results. Another example we encountered was a highway on-ramp that was mistakenly not connected to the main highway in the data. Such an error affects both

the generation of escape routes and the suggested police paths to interception positions. In this case, the generated escape routes falsely suggested that the fugitive would not use the on-ramp. While we have corrected these mistakes, other errors likely persist in the data.

The Search Space Representation approach in this paper can be applied to other network-based optimization problems. In cases like search and rescue, where the location and paths of a lost person are uncertain (Koester 2008), filtering out non-essential parts of the network and focusing on likely routes can significantly reduce computational complexity. For other network-based optimization problems, such as large-scale route planning, this method can help speed up computation and improve solution quality. Using the detailed network is only critical at the beginning (departure from the warehouse) and the end (delivery point). Preserving the full detailed graph is essential at these locations to ensure accurate routing, while coarsening the network in between could significantly speed up the optimization.

Conclusion

This paper compares four graph coarsening techniques for fugitive interception across five road networks. Pruning – the removal of dead ends and self-loops – seems to always be effective: it removes 2.7% to 29.1% of nodes (depending on the network), but these nodes are likely not relevant for fugitive interception. Other preprocessed graph coarsening algorithms can significantly reduce the number of nodes in the networks, but cause the solution quality to deteriorate significantly. Important interception positions and paths for the police units are often not preserved for these algorithms. In contrast, on-the-fly network reconstruction, where a new network is created from the escape routes and the shortest paths from the police starting positions to any node on these escape routes, improves the optimization. By removing poor-quality solutions, the optimization algorithm converges more quickly and results in higher-quality solutions.

Based on these results, we propose an approach incorporating on-the-fly graph reconstruction into the Search Space Representation in the optimization process. This allows for more flexibility, capable of handling different fugitive profiles and network structures. Search space representation improves the quality of the best solutions obtained by the optimization algorithm with up to 12%. Notably, the reliability of the optimization to find high-quality solutions is increased: the average obtained solution quality across seed increases by up to 24%. Meanwhile, the number of function evaluations required to obtain high-quality solutions is reduced to 5 000 -10 000 depending on the size and complexity of the road network, which is feasible for real-time decision-making.

The Search Space Representation approach in this paper can be applied to other network-based optimization problems, specifically search and rescue, and more generally to large-scale route planning.

Tabular comparison of results

Tables 8 , 9, 10, 11 and 12 present the results of the evaluated graph coarsening algorithms for the five road networks. The results for node consolidation are presented using a tolerance value of 30 ms, as this value results in a balanced trade-off between

Table 8 Results of the coarsening algorithms for fugitive interception in Winterswijk

		Min	Med	Mean	Max
G	Z (% of max)	0.72 (97.3%)	0.74 (99.5%)	0.73 (99.2%)	0.74 (100%)
	NFE	3443	20433	27752	83163
Pruning	$Z(G_c)/Z(G)$	0.98	1.00	1.00	1.01
	$NFE(G)/NFE(G_c)$	0.62	5.43	6.27	18.00
Node consolidation	$Z(G_c)/Z(G)$	0.69	0.74	0.78	1.01
	$NFE(G)/NFE(G_c)$	0.41	2.88	3.04	6.22
Heuristic - type	$Z(G_c)/Z(G)$	0.34	0.34	1.00	0.35
	$NFE(G)/NFE(G_c)$	0.03	0.76	2.69	9.12
Heuristic - betweenness	$Z(G_c)/Z(G)$	0.34	0.34	0.34	0.36
	$NFE(G)/NFE(G_c)$	0.03	1.69	2.71	13.30
On-the-fly	$Z(G_c)/Z(G)$	0.99	1.00	1.00	1.01
	$NFE(G)/NFE(G_c)$	0.66	2.66	3.55	10.86

Table 9 Results of the coarsening algorithms for fugitive interception in Manhattan

		Min	Med	Mean	Max
G	Z (% of max)	0.76 (84.3%)	0.85 (94.8%)	0.85 (94.2%)	0.90 (100%)
	NFE	23159	52635	52994	79822
Pruning	$Z(G_c)/Z(G)$	0.89	0.96	0.95	1.02
	$NFE(G)/NFE(G_c)$	1.19	1.72	2.00	3.12
Node consolidation	$Z(G_c)/Z(G)$	0.12	0.45	0.45	0.89
	$NFE(G)/NFE(G_c)$	0.31	1.09	1.73	4.15
Heuristic - type	$Z(G_c)/Z(G)$	0.01	0.42	0.48	1.00
	$NFE(G)/NFE(G_c)$	0.01	0.35	0.54	1.67
Heuristic - betweenness	$Z(G_c)/Z(G)$	0.01	0.01	0.05	0.30
	$NFE(G)/NFE(G_c)$	0.05	0.13	0.14	0.33
On-the-fly	$Z(G_c)/Z(G)$	0.91	0.97	0.96	0.99
	$NFE(G)/NFE(G_c)$	0.60	1.41	1.41	2.22

Table 10 Results of the coarsening algorithms for fugitive interception in Utrecht

		Min	Med	Mean	Max
G	Z (% of max)	0.48 (73.9%)	0.55 (85.0%)	0.55 (84.5%)	0.65 (100%)
	NFE	24000	62978	58561	87309
Pruning	$Z(G_c)/Z(G)$	0.78	0.91	0.90	1.05
	$NFE(G)/NFE(G_c)$	0.86	2.36	2.32	3.25
Node consolidation	$Z(G_c)/Z(G)$	0.05	0.50	0.45	0.77
	$NFE(G)/NFE(G_c)$	0.66	1.78	2.13	4.14
Heuristic - type	$Z(G_c)/Z(G)$	0.00	0.00	0.01	0.03
	$NFE(G)/NFE(G_c)$	0.00	0.00	0.00	0.00
Heuristic - betweenness	$Z(G_c)/Z(G)$	0.80	1.00	0.99	1.06
	$NFE(G)/NFE(G_c)$	0.25	1.30	1.38	2.89
On-the-fly	$Z(G_c)/Z(G)$	0.75	0.90	0.90	1.05
	$NFE(G)/NFE(G_c)$	0.31	1.92	1.76	3.38

Table 11 Results of the coarsening algorithms for fugitive interception in the main road network around Amsterdam

		Min	Med	Mean	Max
G	Z (% of max)	0.61 (64.1%)	0.78 (82.0%)	0.78 (82.0%)	0.96 (100%)
	NFE	5769	24928	36126	91322
Pruning	$Z(G_c)/Z(G)$	0.82	0.84	0.88	1.01
	$NFE(G)/NFE(G_c)$	1.53	5.00	6.68	17.34
Node consolidation	$Z(G_c)/Z(G)$	0.83	0.99	0.95	1.03
	$NFE(G)/NFE(G_c)$	0.30	8.01	7.23	15.50
Heuristic - type	$Z(G_c)/Z(G)$	0.00	0.00	0.00	0.01
	$NFE(G)/NFE(G_c)$	0.02	0.02	0.02	0.02
Heuristic - betweenness	$Z(G_c)/Z(G)$	0.00	0.02	0.34	0.89
	$NFE(G)/NFE(G_c)$	0.02	0.02	2.38	9.35
On-the-fly	$Z(G_c)/Z(G)$	0.84	1.01	0.99	1.03
	$NFE(G)/NFE(G_c)$	1.05	5.80	6.34	11.80

Table 12 Results of the coarsening algorithms for fugitive interception in Rotterdam

		Min	Med	Mean	Max
G	Z (% of max)	0.31 (84.0%)	0.33 (89.0%)	0.34 (90.0%)	0.37 (100%)
	NFE	11169	26599	33641	72047
Pruning	$Z(G_c)/Z(G)$	0.91	0.94	0.94	0.97
	$NFE(G)/NFE(G_c)$	1.27	3.39	3.28	6.38
Node consolidation	$Z(G_c)/Z(G)$	0.06	0.17	0.20	0.58
	$NFE(G)/NFE(G_c)$	0.90	2.84	3.26	6.70
Heuristic - type	$Z(G_c)/Z(G)$	0.84	0.93	0.92	0.99
	$NFE(G)/NFE(G_c)$	1.05	1.69	2.36	7.80
Heuristic - betweenness	$Z(G_c)/Z(G)$	0.89	0.93	0.94	1.07
	$NFE(G)/NFE(G_c)$	0.75	1.62	1.98	6.13
On-the-fly	$Z(G_c)/Z(G)$	0.89	0.96	0.96	1.03
	$NFE(G)/NFE(G_c)$	0.32	1.96	2.51	7.57

node reduction and maintaining solution quality. For the heuristic coarsening algorithms, the maximum coarsening settings are applied, with pruning set to 1, iterations to the maximum, and the threshold also at its maximum value. For each algorithm, we report the minimum, median, mean, and maximum values across seeds. The first row in each table presents the results for the uncoarsened graph: Z is the fraction of intercepted routes and NFE is the number of function evaluations to search stall (reaching 95% of the solution quality). For each coarsening algorithm, the results are scaled to the best found for the uncoarsened graph: the maximum value for Z and the minimum value for NFE. A Z value close to or above 1 indicates good solution quality, while an NFE value below 1 indicates faster convergence.

Author contributions
ID, JK, JM, and AV designed the research; ID performed the research, implemented computer experiments, analyzed data, and wrote the original draft of the paper; JK, JM, and AV were involved in the supervision, and revised and corrected the paper. The authors read and approved the final manuscript.

Funding
This research is fully funded by the National Police Artificial Intelligence Lab of the Netherlands.

Availability of data and materials
The datasets generated and/or analyzed during the current study are available in the ***CoarseningFIP*** GitHub repository: https://github.com/irene-sophia/CoarseningFIP.

Declarations

Competing interests
The authors declare that they have no conflict of interest.

Received: 24 October 2024 Accepted: 30 December 2024
Published online: 14 January 2025

References

Asano T, Hirata T (1983) Edge-contraction problems. J Comput Syst Sci 26(2):197–208. https://doi.org/10.1016/0022-0000(83)90012-0
Alspach B (2004) Searching and sweeping graphs: a brief survey. Le Mat 59:5–37
Berman O, Larson RC, Fouska N (1992) Optimal location of discretionary service facilities. Transp Sci 26(3):201–211. https://doi.org/10.1287/trsc.26.3.201
Boeing G (2017) Osmnx: new methods for acquiring, constructing, analyzing, and visualizing complex street networks. Comput Environ Urban Syst 65:126–139. https://doi.org/10.1016/j.compenvurbsys.2017.05.004
Boeing G (2024) Modeling and analyzing urban networks and amenities with OSMnx. https://geoffboeing.com/publications/osmnx-paper/
Bode F, Reed PM, Reuschen S, Nowak W (2019) Search space representation and reduction methods to enhance multiobjective water supply monitoring design. Water Resour Res 55:2257–2278. https://doi.org/10.1029/2018WR023133
Boccia M, Sforza A, Sterle C (2009) Flow intercepting facility location: problems, models and heuristics. J Math Model Algorithms 8(1):35–79. https://doi.org/10.1007/s10852-008-9098-5
Borie R, Tovey C, Koenig S (2011) Algorithms and complexity results for graph-based pursuit evasion. Auton Robot 31:317–332. https://doi.org/10.1007/s10514-011-9255-y
Chung TH, Hollinger GA, Isler V (2011) Search and pursuit-evasion in mobile robotics: a survey. Auton Robot 31(4):299–316. https://doi.org/10.1007/s10514-011-9241-4
Chevalier C, Safro I (2009) Comparison of coarsening schemes for multilevel graph partitioning. Learning and intelligent optimization. Springer, Berlin, Heidelberg, pp 191–205. https://doi.org/10.1007/978-3-642-11169-3_14
Delft High Performance Computing Centre (2022) DelftBlue supercomputer (phase 1). https://www.tudelft.nl/dhpc/ark/delftbluephase1
Geisberger R, Sanders P, Schultes D, Delling D (2008) Contraction hierarchies: faster and simpler hierarchical routing in road networks. In: International workshop on experimental and efficient algorithms, pp 319–333. https://doi.org/10.1007/978-3-540-68552-4_24
Hodgson MJ (1990) A flow-capturing location-allocation model. Geogr Anal 22:270–279. https://doi.org/10.1111/j.1538-4632.1990.tb00210.x
Hadka D, Reed P (2013) Borg: an auto-adaptive many-objective evolutionary computing framework. Evol Comput 21(2):231–259. https://doi.org/10.1162/EVCO_a_00075
Krishnakumari P, Cats O, Lint H (2020) Heuristic coarsening for generating multiscale transport networks. IEEE Trans Intell Transp Syst 21(6):2240–2253. https://doi.org/10.1109/TITS.2019.2912430
Koester RJ (2008) Lost person behavior: a search and rescue guide on where to look for land, air, and water. DbS Productions, Charlottesville, VA, United States
Loukas A (2019) Graph reduction with spectral and cut guarantees. J Mach Learn Res 20(116):1–42
Pung J, D'Souza RM, Ghosal D, Zhang M (2022) A road network simplification algorithm that preserves topological properties. Appl Netw Sci 7(1):79. https://doi.org/10.1007/s41109-022-00521-8
Peleg D, Schäffer AA (1989) Graph spanners. J Graph Theory 13(1):99–116. https://doi.org/10.1002/jgt.3190130114
Ralphs T (2022) CBC: COIN-OR branch-and-cut solver. Version 2.10.8. https://github.com/coin-or/Cbc
Sanders P, Schultes D (2012) Engineering highway hierarchies. J Exp Algorithmics 17:1. https://doi.org/10.1145/2133803.2330080
Safro I, Sanders P, Schulz C (2015) Advanced coarsening schemes for graph partitioning. J Exp Algorithmics 19:1–24. https://doi.org/10.1145/2670338
van Droffelaar IS, Kwakkel JH, Mense JP, Verbraeck A (2024) The effect of models of fugitive behavior on police interception strategies. In: Advances in social simulation. ESSA 2024. Springer proceedings in complexity. Springer, Switzerland
van Droffelaar IS, Kwakkel JH, Mense JP, Verbraeck A (2024) Simulation-optimization configurations for real-time decision-making in fugitive interception. Simul Model Pract Theory. https://doi.org/10.1016/j.simpat.2024.102923
Walshaw C (2004) A multilevel approach to the travelling salesman problem. Oper Res 50:862. https://doi.org/10.1287/opre.50.5.862.373

Publisher's Note