TU Delft

Delft University of Technology

Simulation–optimization configurations for real-time decision-making in fugitive interception

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

DOI
10.1016/j.simpat.2024.102923

Publication date
2024

Document Version
Final published version

Published in
Simulation Modelling Practice and Theory

Citation (APA)
van Droffelaar, I. S., Kwakkel, J. H., Mense, J. P., & Verbraeck, A. (2024). Simulation–optimization configurations for real-time decision-making in fugitive interception. *Simulation Modelling Practice and Theory*, *133*, Article 102923. https://doi.org/10.1016/j.simpat.2024.102923

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

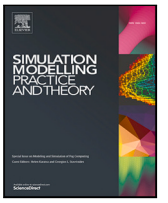
This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Simulation Modelling Practice and Theory 133 (2024) 102923

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat



Simulation–optimization configurations for real-time decision-making in fugitive interception

Irene S. van Droffelaar [a,*], Jan H. Kwakkel [a], Jelte P. Mense [b], Alexander Verbraeck [a]

[a] Delft University of Technology, Faculty of Technology, Policy, and Management, Jaffalaan 5, Delft, 2628 BX, The Netherlands
[b] Utrecht University, Police AI Lab, Princetonplein 5, Utrecht, 3584 CC, The Netherlands

ARTICLE INFO

Dataset link: Software&Code (Original data)

Keywords:
Simulation–optimization
Fugitive interception
Search problem
Real-time optimization
Emergency service optimization

ABSTRACT

Simulation–optimization models are well-suited for real-time decision-support to the control room for search and interception of fugitives by Police on a road network, due to their ability to encode complex behavior while still optimizing the interception.

The typical simulation–optimization configuration is simulation model optimization, where the simulation model describes the system to be optimized, and the optimizer attempts to find the combination of decision variables that maximizes the interception probability. However, the repeated evaluation of the simulation model leads to high computation time, thus rendering it inadequate for time-constrained decision contexts. To support police interception operations in real-time, timely calculation of the solution is essential. Sequential simulation–optimization, where the simulation model, with its rich behavior, constructs (part of) the constraints of an optimization problem, could decrease the computation time.

We compare the computation time for two configurations of simulation–optimization (typical simulation model optimization and sequential simulation–optimization) for various problem instances of the fugitive interception problem. We show that sequential simulation–optimization reduces the computation time of large instances of the fugitive interception case study ten-fold. This result illustrates the potential of sequential simulation–optimization to mitigate the expensive optimization of simulation models.

1. Introduction

Search and interception of fugitives by the police on a road network is a challenging task due to the complexity of the network, the unknown whereabouts of the fugitive and uncertainty about the routes that the fugitive takes, the stressful decision-making context [1], and time pressure [2]. Both stress and time pressure have an adverse effect on the amount of information that can be processed and, therefore, on the quality of the decision-making process [2]. Information technology, supported by modeling and simulation to depict the complex and stochastic decision space, can mitigate these effects by suggesting interception positions for police units. Related search and interception routing problems are solved using a variety of approaches. For example: general graph search [3], continuous space search using mobile robotics [4], missile interception [5], and search and rescue [6], or more specifically, finding the lost MH370 [7]. Simulation–optimization models, in particular, seem suitable to solve the fugitive interception problem due to their ability to encode complex behavior and solve for good interception routes.

To support police interception operations in real-time, timely calculation of the solution is essential [8]. Given the complexity of the problem, caused by a large number of edges in a road network, the uncertainty in the behavior of the fugitive, and the

* Corresponding author.
E-mail address: i.s.vandroffelaar@tudelft.nl (I.S. van Droffelaar).

https://doi.org/10.1016/j.simpat.2024.102923
Received 11 December 2023; Received in revised form 4 March 2024; Accepted 8 March 2024
Available online 11 March 2024
1569-190X/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

degrees of freedom of the police units, solving a typical simulation–optimization configuration in real-time is infeasible. Typically, the computation time of simulation–optimization is improved by increasing computation power and algorithm efficiency. For larger networks, however, the computation time of the fugitive interception problem would still be too large for using simulation–optimization in a real-world context where a solution is needed in less than a minute. A promising alternative is to combine simulation and optimization differently than in classical simulation–optimization, such that the number of times the simulation model has to run is drastically reduced [9].

A different way of combining simulation and optimization is sequential simulation–optimization, where the simulation constructs (part of) the constraints of an optimization problem. This paper provides an extension to the taxonomy of simulation–optimization configurations, presents and researches sequential simulation–optimization, and provides a quantitative analysis of the real-time performance of classical simulation–optimization compared to sequential simulation–optimization. We apply the comparison to a fugitive interception problem to two case studies: a 2D grid and a city road network. Thus, we show the potential of sequential simulation–optimization to mitigate the expensive optimization of simulation models.

Section 2 outlines the background literature on simulation–optimization. Section 3 describes the methods used in the paper, including a description of the case study, the models, and the optimization algorithms. The subsequent sections detail the obtained results, specifically on a grid network (Section 4.1) and a real-world road network (Section 4.2). Possible threats to the validity of the results are discussed in Section 5, and we share our conclusions in Section 6.

2. Simulation–optimization

Two paradigms of prescriptive analytics are simulation and optimization. Simulation answers ‘what-if’ questions about a system: what is the system response given a set of values for the decision variables? In contrast, optimization aims to answer ‘how-to’ questions: how to maximize or minimize the system response by choosing the optimal values for the decision variables? [10,11] Simulation–optimization combines the two. Despite broad applicability, simulation–optimization is still less popular than pure simulation or optimization studies [12].

The terms ‘optimization via simulation’ [13,14], or ‘simulation for optimization’ [15] are often conflated with the term simulation–optimization. They describe the application of simulation methods for solving optimization problems — not the combination of simulation models and optimization models. Well-known examples are simulated annealing and ant colony optimization.

This section describes the related literature in real-time simulation–optimization and simulation–optimization configurations. In the latter subsection, we dive deeper into the configurations and provide a synthesis of concepts that add to the broader understanding of simulation–optimization.

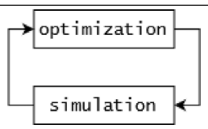
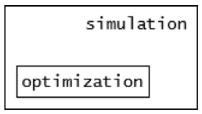
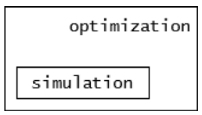
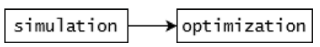
2.1. Real-time simulation–optimization

To be useful for real-time decision making, the timely calculation of the optimal solution is essential. In classical simulation model optimization, the simulation model is evaluated for each set of input parameter values that constitute a candidate solution determined by the optimizer. In this setup, two factors determine the computation time: (1) the number of function evaluations (*i.e.,* a single run of the simulation model) needed by the optimizer to find the optimal solution and (2) the computation time per function evaluation. The first is dependent on the efficiency of the optimizer. Since discrete simulation (DEVS) typically yields a rugged fitness landscape, the computation time of the optimization is high [16]. Improvement of optimization algorithms for simulation–optimization is an active field of research with extensive literature [17]. Review articles over time are provided by, among others, [17–21]. The popularity of Digital Twins to support decision-making is increasing the need for approaches for timely simulation–optimization, and, consequently, for research done in the field [22]. Recently, the focus has shifted towards finite time performance rather than asymptotic performance, where the optimizer has reached convergence, as exemplified in [23–25]. This development is relevant for real-time decision making, because the focus shifts to obtaining timely ‘as-good-as-feasible’ solutions rather than ‘as-good-as-possible’ solutions in as much time as the computation budget allows. For example, De Armas et al. [26] solve the uncapacitated facility location problem for telecommunications in real time using a tailor-made simulation-based metaheuristic. However, even with improving and tailoring the optimization approach, thousands of simulation runs must be completed in the optimization search for larger problems. For complex simulation models, this leads to infeasibly high computation times for real-time simulation. The second factor, computation time per function evaluation - running the simulation model - is often not reducible due to the complexity inherent to the problem. In some cases, a solution is to develop surrogate models - or metamodels - that describe the input–output relations of the simulation model and are computationally cheaper to evaluate [27]. However, this method requires an initial time investment to fit the surrogate model to the complex and stochastic fitness landscape of the simulation model, and information is lost in the process.

2.2. Simulation–optimization configurations

The term ‘simulation–optimization’ is ambiguous, with most articles describing the optimization of a simulation model (the top configuration in Table 1). Yet, simulation and optimization can be coupled in various configurations. Figueira & Almada-Lobo [9] attempt to reduce the ambiguity by presenting a taxonomy for simulation–optimization. They distinguish four dimensions: simulation

Table 1
A tabular overview of the four simulation–optimization configurations.

	Configuration	Goal	Type of output	Iterations
(a)		Optimize a simulation model	Optimal solution	Optimization: > 1 Simulation: > 1
(b)		Implement part(s) of a simulation model as optimization	Model output	Optimization: ≥ 1 Simulation: 1
(c)		Implement part(s) of an optimization model as simulation	Optimal solution	Optimization: 1 Simulation: ≥ 1
(d)		Construct constraints of an optimization model through simulation	Optimal solution	Optimization: 1 Simulation: 1

purpose, hierarchical structure, search method, and search scheme. Table 1 summarizes the hierarchical structure dimension, which this paper examines further.

We examine the connections between simulation and optimization in each of the configurations. An optimization model consists of constraints, an objective function, and an optimizer [28]. The objective function is a function of model variables that should be maximized or minimized; constraints define feasibility by imposing limitations on model variables and parameters; the optimizer, or solver, is the algorithm that finds the optimal solution. Likewise, following the Discrete Event System Specification modeling framework [29], a simulation model consists of an experimental frame, a model, and a simulator. The experimental frame provides the input arguments: the conditions under which the system is experimented with; the model describes the logic of the simulation; the simulator executes the model. In Fig. 1, we use these frameworks [28,29] to specify the simulation and optimization components in the simulation optimization taxonomy [9]. In each configuration, the model is assumed to be fixed. Changes to the model are passed through the experimental frame as changes to parameters or decision variables. Changes to the model itself would introduce structural uncertainty, which is outside the scope of this paper.

The following paragraphs describe the four configurations in more detail and provide examples. For each configuration, we discuss the purpose of the simulation and optimization components, which components are endogenous/exogenous, how stochasticity in the simulation model is managed, and an illustrative example from the literature.

Configuration (a): Simulation Model Optimization The simulation model describes the system to be optimized, and the optimizer attempts to find the combination of decision variables X that minimizes or maximizes the objective value $f(X)$ where f is the transformation of input to output by the simulation model and $Y = f(X)$ are the model output variables [13]. In each iteration, the optimization and the simulation are both executed [9]. For the simulation, this means that the model is run to completion. If the simulation model is stochastic, the simulation model's output is a summary statistic over multiple replications. For optimization, the optimizer receives the simulation model's output for the previous set of decision variables and determines a new set of decision variables to evaluate. The decision variables are passed to the simulator through the experimental frame.

[30] describe a real-world example using an extensive simulation model of transportation behavior in the central Ohio region. The optimal locations of electric vehicle charging stations are determined to maximize the service rate. Many more examples are described in the literature, with applications in, among others, transport, logistics, and health care [17]. Extensions of simulation model optimization are, for example, robust optimization and model calibration, where additional functions are applied to the outcomes of the simulation model. In robust optimization, the objective function is based on the robustness of the output given a set of values for the decision variables. Robustness is often defined as the variance of the output for a set of simulation runs with different input parameter values, but there is a broader array of robustness metrics available (see [31]). Simulation model calibration minimizes the distance between the output of the simulation model and the associated observed value [32].

Configuration (b): Optimization Model as Part of a Simulation The optimization model describes a process within the simulation model, representing something that is also optimized in the system under study. In each iteration of the simulation, one or more complete optimization runs are performed [9]. The simulator calls the optimization model, which calculates the optimal solution given the current states of the simulation, the constraints, and the objective function. The optimizer's output is endogenous and used in the remainder of the simulation run. Compared to configuration A, the objective function becomes endogenous. The experimental frame is exogenous and is determined by the model user. If the simulation model is stochastic, this entire process should be executed for multiple replications.

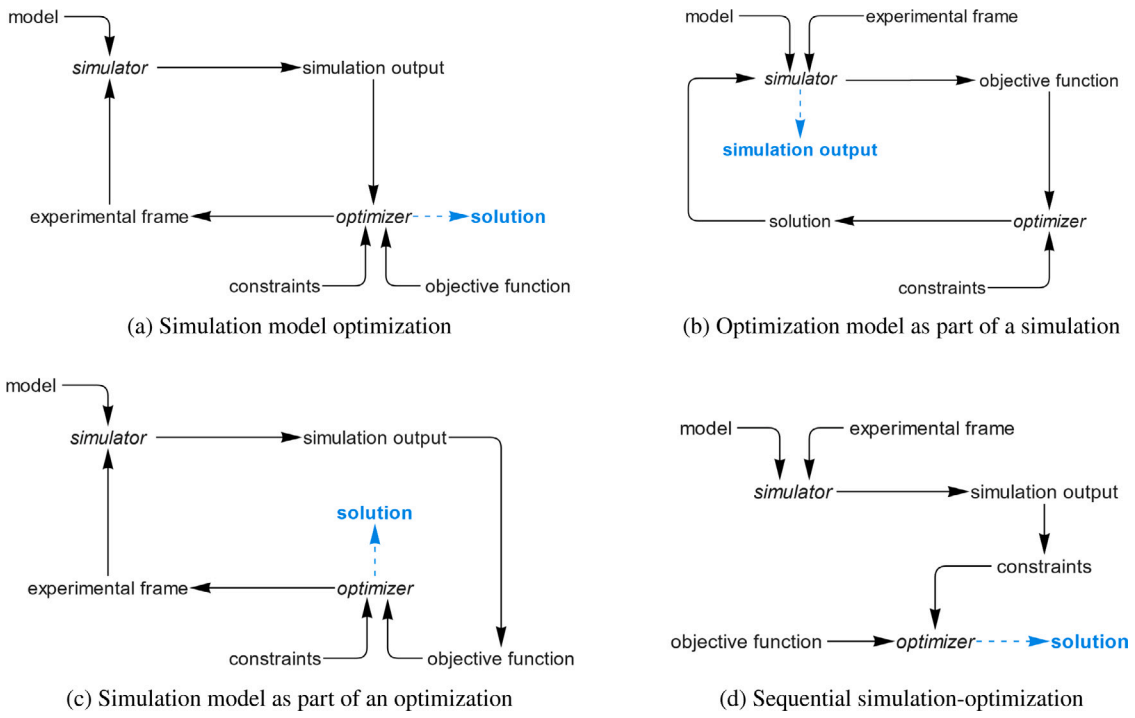


Fig. 1. Configurations of combining simulation and optimization. The blue dashed arrow indicates the entity that delivers the final output to the model user.

An illustrative example is described by [33], where ambulance operations are examined. An optimization model dynamically determines the re-locations of ambulances to anticipate demand each time step of the simulation model. Many agent-based models also fall into this category: each agent optimizes its own behavior, and we observe the emergent behavior on the model level.

Configuration (c): Simulation Model as Part of an Optimization The simulation model describes a process within an optimization model, which cannot or should not be characterized by constraints in the optimization model [34]. Similar to configuration B, the objective function is endogenous. In contrast, the simulation output is endogenous, and the simulation–optimization output is the optimizer's solution. The optimizer provides the input for the simulation model. The simulation output contributes to a part of the objective function. The simulation typically introduces stochasticity in the optimization. Many replications should be run to obtain output with a small confidence interval since classical optimization methods cannot handle stochasticity. In each iteration of the optimization model, one or multiple simulation runs are completed [9].

An example is using a simulation model as a more realistic representation of a queue, in contrast to a simple but unrealistic, first-in-first-out (or other simple optimization-based) queuing system. [34] provides the example of a resource allocation model, where a simulation model of the inventory represents the stochasticity in lead time and inventory.

Configuration (d): Sequential Simulation-Optimization The simulation and optimization modules run sequentially, where the simulation experiment runs only once [9]. If the simulation model is stochastic, this simulation experiment should consist of multiple replications to obtain the simulation output. Table 1 and Fig. 1(d) depict simulation ⟶, but the simulation and optimization components could be connected in either order. In simulation ⟶ optimization, the simulation output forms (a part of) the constraint set of the optimization model. The constraints to the optimization problem are endogenous. Similar to configurations (a) and (c), the final output is the optimal solution. In contrast, both the simulation and optimization components are only run once, suggesting a major potential improvement in efficiency. In optimization ⟶ simulation, the optimization model provides a configuration or schedule to be used in the simulation. The final output is the simulation output, similar to configuration (b).

An illustrative example of sequential simulation–optimization is provided by [35], where simulation is used to construct scenario trees that are subsequently used in a financial portfolio optimization problem.

3. Method

In this section, we first describe the case study, independent of the specific implementations. Second, we explain the implementation of simulation model optimization (Configuration (a) from Table 1). Next, we present the sequential simulation–optimization formalization (Configuration (d) from Table 1). Finally, we discuss the solution approaches and methods for comparison.

Table 2
Notation of parameters and decision variables. Simulation model optimization and sequential simulation–optimization require different forms of $\pi_{u(,v)}$ and $\phi_{r(,v),t}$. For completeness, both are included in this table.

Decision variables	
$z_r \in \{0,1\}$	binary variable for the interception of route r
$\pi_u \in V$	target vertex of police unit u
$\pi_{u,v} \in \{0,1\}$	binary parameter for the target vertex v of police unit u
Parameters	
$V = \{v\}$	set of vertices
$R = \{r\}$	set of fugitive routes
$U = \{u\}$	set of police units
$T = \{t\}$	ordered index set of time steps
$\phi_{r,t} \in V$	vertex of fugitive route r at time t
$\phi_{r,v,t} = \{0,1\}$	binary parameter for the presence of fugitive route r at vertex v at time step t
$\tau_{u,t} \in V$	vertex of police unit u at time t
$\tau_{u,v,t} = \{0,1\}$	binary parameter for the reachability of vertex v by police unit u at time t

3.1. Case study: fugitive interception

We use the positioning of police units to maximize the probability of intercepting a fleeing fugitive on a road network as a case to examine the effect of simulation–optimization configurations on the computation time. The problem is modeled from the time of the incident until the fugitive is either intercepted or has escaped. Police units have no knowledge of the fugitive's whereabouts, so they have to move to a vertex in the network where the probability of intercepting is highest, *e.g.*, a chokepoint in the network where many routes pass through.

3.1.1. Related optimization problems

Search problems, and more specifically, interception problems, describe related problems. A search problem related to the case study addressed in this paper optimizes the routes of searchers to maximize the probability of finding a target or to minimize the time to detect a target [3]. However, optimizing an action for each time step quickly becomes untractable with larger networks and longer time horizons.

Primarily applied in the field of robotics, these problems are often modeled in continuous spaces or on grids [4]. In theoretical mathematical exploration, problems are solved for different graph topologies, such as grids, circular graphs, trees, and random graphs. On various graph topologies, these problems are proven to be pseudo-P to strongly NP-complete, depending on the specific problem formulation and properties of the graph [36]. Due to the complexity of these optimization problems, the problem instances that are studied are typically very small. To our knowledge, there are no applications in real-world street graphs.

Problems with both stationary and moving targets have been addressed. Stationary targets may be placed randomly or adversarially to evade capture. Similarly, moving targets may be adversarial or non-reactive. Non-reactive moving targets are generally modeled as random walks [4]. Extensive efforts have been made to analyze and model the behavior of lost persons [6,37,38]. However, these behaviorally rich models have not been integrated with search optimization problems, and models of criminal routing behavior have not been published.

Another related area of research describes *flow interception problems*, a special type of Facility Location Problem. Developed by Hodgson [39] and Berman et al. [40], the original model aims to maximize the flow intercepted by a certain number of facilities. For example, these models are used to maximize the number of consumers who encounter at least one facility on their path. Gendreau et al. [41] extended the FIP to include a gain coefficient a_{rv} for each vertex v belonging to route r instead of implicitly relating the gain to the flow values. Tanaka & Kurita [42] adapt the FIP to handle probabilistic interception and reward early interception of travelers. The generic Flow Interception Problem is NP-hard, meaning it cannot be solved in polynomial time [40].

3.1.2. Modeling choices

We choose to model the problem as a variation on the Flow Interception Problem, since it is expected to be a more tractable problem to solve in real time. Instead of optimizing a position for each police unit for each time step, we optimize the target position for each police unit.

We simulate the routing behavior of the fugitive on a graph as a random walk starting at the location of the incident. At each intersection, the fugitive chooses the next vertex to travel to, which is a stochastic process where each neighboring vertex has equal probability. The fugitive does not turn around unless the vertex only has one neighboring vertex (i.e., a dead end). Each instance of this simulation generates an element r in R, consisting of $|R|$ fugitive routes. Table 2 specifies the model variables and parameters.

$$\phi_{r,t+1} = Uniform(Neighbors(\phi_{r,t})) \qquad \forall r \in R \quad \forall t \in T \tag{1}$$

The travel time between two vertices i and j is determined by the length of the edge ($length_{i,j}$) and the maximum allowed speed on that edge ($Vmax_{i,j}$).

$$travel\ time_{i,j} = \frac{length_{i,j}}{Vmax_{i,j}} \tag{2}$$

We optimize the positioning of the police units ($\pi_{u,v}$) to jointly maximize the number of intercepted fugitive routes (z_r). The police units drive the shortest route from their position at the time of the incident to their respective target vertex. The police intercept a fugitive route r if a police unit has arrived at its target vertex (π_u), and the fugitive and the police unit are at the same vertex at the same time. The police units stay at their target vertex. Therefore, if a fugitive's route crosses that vertex at a later time step, it also results in an interception.

3.1.3. Model description: simulation model optimization

The implementation of simulation model optimization consists of a simulation model ($f(\pi_u; \phi_{r,t}, \tau_{u,t})$) that describes the movement of the fugitive, the movement of the police units, and the interception process given the starting and target nodes of the police units ($\tau_{u,t}$, π_u) and the fugitive routes ($\phi_{r,t}$). The simulation model outputs the number of intercepted fugitive routes (z_r). To account for stochasticity in the behavior of the fugitive, the model contains 500 instances of the fugitive model entity. These escape routes are the same for each function evaluation. The target vertices of the police units are optimized to maximize the number of intercepted fugitive routes (Eq. (4)).

$$\text{Maximize:} \quad Z = \sum_{r \in R} z_r \tag{3}$$

$$\text{Subject to:} \quad z_r = f(\pi_u; \phi_{r,t}, \tau_{u,t}) \tag{4}$$

The simulation model is implemented in pyDSOL, a Python implementation of the Distributed Simulation Object Library (DSOL) simulation library [43].[1,2]

3.1.4. Model description: sequential simulation–optimization

The implementation of sequential simulation model optimization consists of a simulation model that describes the movement of the fugitive and a separate optimization model that determines the routing of the police units. The simulation model is run 500 times to generate an ensemble of plausible fugitive routes and to construct $\phi_{r,v,t}$. The optimization problem is formulated as a Flow Interception Problem [39,40] with a time constraint on interception, determined by the initial position and speed of the police units.

Analogous to the simulation model formalization in the previous section, the decision variables of the optimization problem are the police unit positions $\pi_{u,v}$ and the intercepted routes z_r. Given the decision variables and parameters outlined in Table 2, the optimization problem is defined as follows:

$$\text{Maximize: } Z = \sum_{r \in R} z_r \tag{5}$$

$$\text{Subject to: } \sum_{v \in V} \pi_{u,v} == 1 \qquad \forall u \in U \tag{6}$$

$$z_r = \min\left(1, \sum_{u \in U} \sum_{t \in T} \sum_{v \in V} \phi_{r,v,t} \cdot \pi_{u,v} \cdot \tau_{u,v,t}\right) \qquad \forall r \in R \tag{7}$$

The objective function of the optimization (5) describes the maximization of the number of intercepted routes at vertices $\pi_{u,v}$. Furthermore, $\tau_{u,v,t}$ is a given for any starting point of a police unit and can be pre-loaded. Constraint (6) ensures that only one position is chosen for each police unit. Constraint (7) ensures that a route is intercepted if a police unit is placed at any vertex on a route r and the police unit can reach vertex v at time t. If a route contains more than one police unit, it will be counted in the objective function once since z_r is a binary variable. If no vertex in route r contains a police unit, the variable z_r equals 0 (i.e., not intercepted). z_r equals 1 (i.e., is intercepted) if a police unit is present on at least one vertex in route r, [44].

3.2. Solution approaches

3.2.1. Exact optimization

Exact optimization methods guarantee to find an optimal solution. However, the Flow Interception Problem is NP-complete [36], meaning they cannot be solved in polynomial time. Only small-scale instances can be solved using exact methods, threatening the real-time applicability of exact optimization methods. Regardless, many different commercial and open-source exact solvers apply (a mixture of) approaches that exploit common characteristics of these classes of problems to find the optimal solution efficiently. We choose the Coin-OR branch-and-cut open-source solver [45] for its applicability to Mixed-Integer Problems and open-source availability.

We use the true optimum found by the exact solver to assess the convergence of the metaheuristic optimization algorithm.

[1] pyDSOL core: https://github.com/averbraeck/pydsol-core
[2] pyDSOL model: https://github.com/imvs95/pydsol-model

Table 3
Default settings of the Borg MOEA [52]. For the PM rate and UM rate, L is the number of decision variables.

Parameter	Value	Parameter	Value
PM rate	1/L	PCX nr. of parents	10
PM distribution index	20	PCX nr. of offspring	2
SBX rate	1	PCX eta	0.1
SBX distribution index	15	PCX Zeta	0.1
DE crossover rate	0.1	UNDX nr. of parents	10
DE step size	0.5	UNDX nr. of offspring	2
UM rate	1/L	UNDX eta	0.1
SPX nr. of parents	10	UNDX zeta	0.1
SPX nr. of offspring	2	Population size	100
SPX epsilon	0.3	Offspring size	200
		Logging frequency	200 nfe

3.2.2. Metaheuristic optimization

Heuristics are problem-specific solution methods that exploit the properties of the problem to reach a solution efficiently but do not guarantee an optimal solution. Therefore, they are effective for the problem they were designed for while being inefficient for others [46]. Conversely, a metaheuristic is a generic algorithm that can be applied to any optimization problem. While convergence metrics can be used to track the algorithm's progress, optimality cannot be guaranteed. Abdel-Basset et al. [47] distinguish metaphor-based metaheuristics (for example, based on biology (*e.g.*, evolutionary algorithms) or physics (*e.g.*, simulated annealing)) and non-metaphor-based metaheuristics (*e.g.*, Tabu Search and Variable Neighborhood Search). Genetic algorithms – a subset of evolutionary algorithms – generally perform well on combinatorial optimization problems with complex interactions between decision variables [48–50]. State-of-the-art examples of evolutionary algorithms for multi-objective optimization problems are ϵ-NSGA-II [51] and Borg [52].

We choose a simple genetic algorithm supplemented with the auto-adaptive framework from Borg, which co-evolves the probabilities of the evolutionary operators used for population adaptation based on their relative success in finding fitter offspring. This means that the algorithm optimizes the probability of each operator being used during the optimization, speeding up convergence by leveraging each operator when performing best. The operators used are (1) Simulated Binary Crossover (SBX), (2) Differential Evolution (DE), (3) Parent-Centric Crossover (PCX), (4) Simplex Crossover (SPX), (5) Unimodal Normal Distribution Crossover (UNDX), and (6) Uniform Mutation (UM) applied with probability [52]. At the initialization of the algorithm, each operator has equal probability. We use the default settings for Borg, as presented in Table 3. Further research should systematically compare various suitable optimization algorithms, for example, using a testbed like [53].

The solutions of the metaheuristic are scaled to the solution found by the exact optimization approach. Therefore, a scaled score of 1 means that the metaheuristic has found the best possible solution, not that the solution intercepts all fugitive routes.

3.3. Search space representation

Following [54], three search space representation measures are implemented to speed up convergence: a linear index representation of the search space and consecutive filtering and sorting of the possible values for the decision variables.

The standard way to represent the set of possible target vertices for the police units is the *binary representation*, where each combination of police unit u and vertex v is a binary variable $\pi_{u,v}$. Bode et al. [54] signal that evolutionary algorithms have difficulty traversing the search space due to the large set of possible combinations and strong interdependency of decision variables. Therefore, Bode et al. [54] propose the *linear index representation*, where the decision variables are linear indices that point to the target vertex for each of the police units. Therefore, the number of decision variables only depends on the number of police units to be positioned and not also on the number of possible target vertices.

Secondly, Bode et al. [54] suggest sorting the indices of the linear index representation so that proximity in the search space is more related to proximity in the objective space. Therefore, we sort the possible target vertices for the police units on their proximity to the starting vertex of the fugitive.

Lastly, we reduce the search space size by filtering the vertices that cannot be reached within the planning horizon by the fugitive or the respective police unit. These vertices do not contribute to increasing the objective value and do not need to be considered. This filtering considerably decreases the set of possible values for each decision variable and, therefore, the number of permutations, especially for instances with a high number of police units (Table 4).

3.4. Design of experiments

We examine the effect of simulation–optimization configurations on the computation time for varying problem sizes. Specifically, we vary two parameters: the number of vertices in the graph and the number of police units to be positioned and record the computation time. The number of police units determines the number of decision variables and is, therefore, expected to have an effect on the computation time. Preliminary experiments demonstrated that the computation time is especially sensitive to the number of vertices in the network [55].

Table 4
Effect of search space reduction on the possible values for the decision variables and the number of permutations, averaged over 50 seeds. Without applying the search space reduction techniques, the number of permutations is $|V|^{|U|}$, where $|V|$ is the number of vertices in the network and $|U|$ is the number of police units.

Problem size		Avg. reduction per decision variable	Nr. of permutations (% of unfiltered)
$\|U\|$	$\|V\|$		
1	900	86.3%	1.23e2 (13.7%)
5	900	81.1%	1.52e11 (0.026%)
10	900	85.6%	9.37e20 (2.69e−7%)
5	100	36.5%	1.08e9 (10.8%)
5	2500	90.0%	2.56e12 (0.0026%)

Table 5
Parameter ranges for the experiments. The length of the planning horizon (L) depends on the network type.

Parameter	Range	Default value
Number of units to be positioned ($\|U\|$)	1–10	5
Number of vertices in network ($\|V\|$)	100–2500	900
Length of planning horizon (L)	n.a.	$5 + (0.5 * \sqrt{\|V\|})$; radius (m)/5
Number of routes ($\|R\|$)	n.a.	500
Number of starting positions	10	n.a.
Number of random seeds for metaheuristic	5	n.a.

In each experiment, one fugitive and $|U|$ police units are placed on random vertices. We sample 500 fugitive routes. The fugitive is intercepted if it occupies the same vertex at the same time as a police unit, as defined in Eqs. (4) and (7). We optimize the target vertex of each of the police units to maximize the number of intercepted fugitive routes. We evaluate the performance for 10 different combinations of starting locations of police units and the fugitive.

Table 5 details the parameter ranges used in the experiments. The number of vertices in the network and the number of police units to be positioned are chosen to be realistic for the application. The network size determines the length of the planning horizon, meaning the maximum time in the model. It is chosen so that it is possible to just traverse the network within L minutes. Specifically, L is $5 + (0.5 * \sqrt{|V|})$ for the 2D grid and the radius of the road network in meters, divided by 5 for the city road network. The number of fugitive routes simulated in the model is chosen to be sufficient to cover the network. We optimize 10 different combinations of starting locations of police units and the fugitive to account for the varying complexity between combinations of starting locations. Some instances of the problem may be much easier to solve due to a convenient starting location. By sampling 10 different combinations, we control for this variance. To control for the stochastic processes in the optimization algorithm, we run each experiment for 5 seeds. All experiments are conducted on the same machine, the DelftBlue supercomputer [56] on a dedicated node to prevent interference. This cluster offers an Intel XEON E5-6248R 24C 3.0 GHz CPU with 48 cores and 192 GB memory.

3.5. Road networks

The topology of the road network dictates the patterns in the fugitive routes and the relative reachability of parts of the network. To account for the effect of the topology, the experiments are performed on two networks: a 2D 'Manhattan' grid with an edge travel time of 1 min and an extract of the road network of Rotterdam, a typical European city. A grid network is suggested by [57] to improve cross-study comparison of methods and algorithms. The city road network is extracted from OpenStreetMap[3] using Boeing's Python library OSMnx [58]. We vary the radius (in meters) from a central point in the city, resulting in varying-sized networks. For the 2D grid, we vary the diameter, yielding networks of varying sizes.

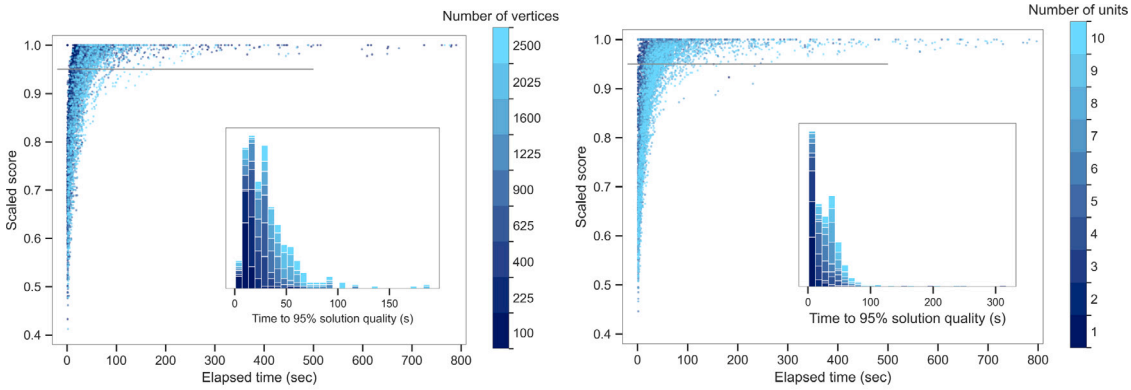
Examples of each network and a combination of starting locations are presented in Fig. 2.

We use the best possible solution obtained by the exact optimization algorithm to assess the convergence of the metaheuristic optimization algorithm for the equidistant grid case. The MIP optimization problem is indexed on time, which works nicely for

[3] OpenStreetMap: https://www.openstreetmap.org

(a) A 100-vertices Manhattan grid (b) A 689-vertices Rotterdam road network

Fig. 2. Test networks, where the starting position and sampled routes of the fugitive are indicated in orange, and the starting positions of the police units are indicated in blue.



(a) Varied number of vertices (b) Varied number of police units

Fig. 3. Convergence of the metaheuristic on simulation model optimization for varying problem size. Each dot indicates an improvement found by the algorithm. The insets portray a histogram of the time to 95% solution quality.

an equidistant graph such as the 2D grid in the first set of experiments. Since the vertices in a real-world road network are not equidistant, time discretization has implications for the accuracy of the optimization model. For a city road network, a very small time step of 1 s is needed to avoid discretization errors and obtain accurate results to assess the quality of the metaheuristic. This leads to high computation times that exceed the time constraint for real-time decision making. Therefore, the best-found solution across seeds is used as the reference set for the city road network.

4. Results

4.1. Casus 1: grid test graph

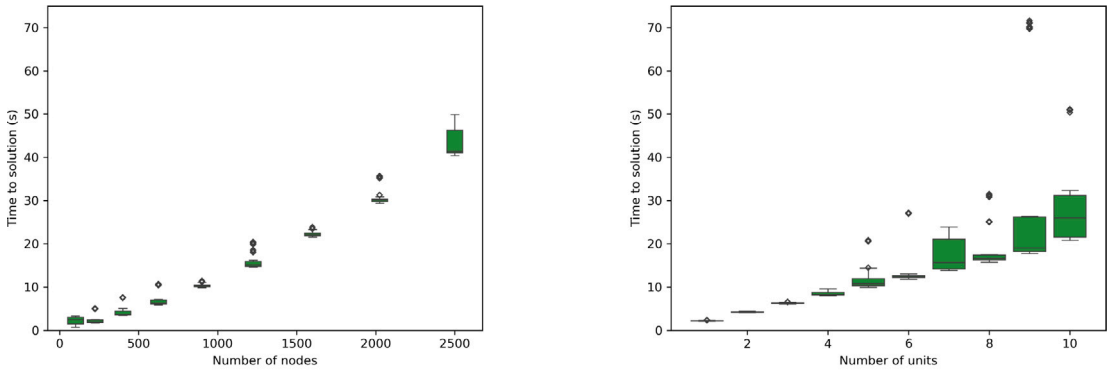
We perform the first set of experiments on a 2D 'Manhattan-like grid. The following paragraphs describe the results for simulation model optimization and sequential simulation–optimization individually, followed by a comparison of the approaches.

4.1.1. Simulation model optimization

We examine the effect of increasing problem size on the convergence of the Borg algorithm using the simulation model optimization configuration. We see that the convergence speed decreases with increasing problem size (Fig. 3). With an increasing number of police units and number of vertices, the density of the improvements shifts downwards (to lower quality of results) and to the right (to longer computation time). The inset histograms demonstrate that the time to 95% of the maximum attainable solution quality increases with increasing problem complexity. The majority of optimization instances reach this quality within 100 s, with outliers for large networks up to 200 s and outliers for a large number of police units up to 320 s.

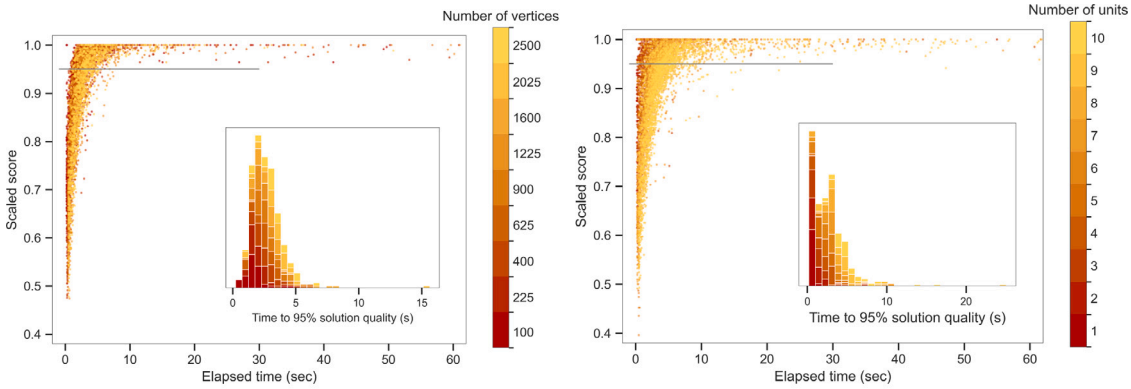
4.1.2. Sequential simulation–optimization

Exact solution approach. Using the exact algorithm COIN-OR Branch-and-Cut (CBC), the time to solution increases with increasing problem size (Fig. 4). Given a fixed network size of 900 vertices, the computation time appears to increase linearly with an increasing



(a) Varied number of vertices (b) Varied number of police units

Fig. 4. Computation time of exact solver CBC on sequential simulation–optimization for varying problem size.



(a) Varied number of vertices (b) Varied number of police units

Fig. 5. Convergence of the metaheuristic on sequential simulation–optimization for varying problem size. Each dot indicates an improvement found by the algorithm. The insets portray a histogram of the time to 95% solution quality.

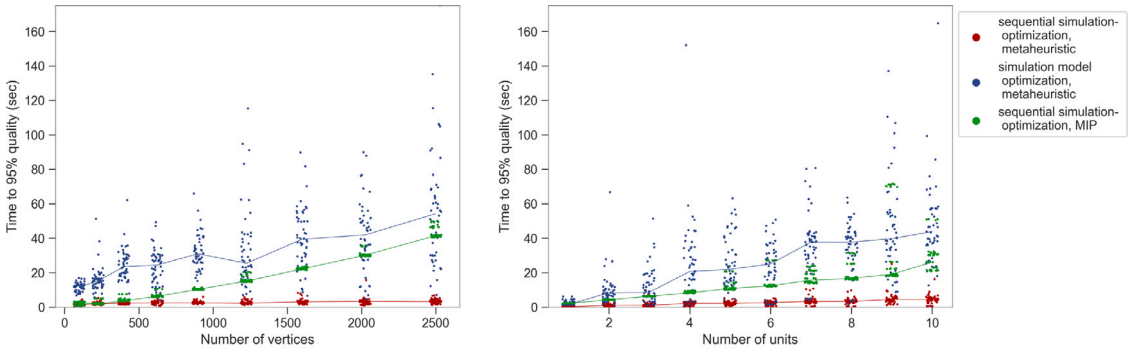
number of police units (Fig. 4(b)). This is because the model treats each police unit independently: adding another police unit adds another decision variable with the same number of options. The interaction effects between police unit interceptions are not explicitly modeled but rather are reflected by the expected number of intercepted routes. With an increasing network size, the computation time increases faster than linearly (Fig. 4(a)). This can be explained by the number of possible combinations between police units and target vertices increasing factorially. With 5 police units, the time to find the optimal solution increases from less than a minute for small network instances with 100 vertices to over 13 min for larger network instances with 2500 vertices. A typical city road network in the Netherlands consists of 2000–4000 vertices. A computation time of over 10 min is unacceptable for real-world application. Moreover, the variance of the computation time increases with increasing problem size, though the predictability of the computation time is crucial for real-time decision-support.

Metaheuristic solution approach. We examine the effect of increasing problem size on the convergence of the metaheuristic algorithm using the sequential simulation–optimization configuration. The solution quality is scaled to the optimal solution found by the exact solver CBC.

Using the metaheuristic algorithm Borg, we see that the convergence speed decreases with increasing problem instance size (Fig. 5). With an increasing number of police units, the density of the improvements shifts downwards (to lower quality of results) and to the right (to longer computation time). This effect is also visible, though to a lesser extent, with an increasing number of vertices. The inset histograms demonstrate that the time to 95% of the maximum attainable solution quality increases with increasing problem complexity. The majority of cases reach this quality within 10 s, with outliers for large networks up to 16 s and outliers for a large number of police units up to 25 s.

4.1.3. Comparison

To compare the computation time of the two configurations and two solution approaches, we calculate the elapsed time at which each problem instance reached a scaled score of at least 0.95. The scaled score is the quality of each solution divided by the best-found solution for the problem instance by the exact MIP solver.



(a) Varied number of vertices (b) Varied number of police units

Fig. 6. Comparison of the computation time for a varying problem size for three approaches: simulation model optimization (solved using a metaheuristic) and sequential simulation–optimization (solved with an exact optimization algorithm and a metaheuristic). The lines indicate the median time to 95% solution quality for each approach.

Figs. 6(a) and 6(b) show that sequential simulation optimization, especially when solved using a metaheuristic approach, outperforms simulation model optimization. The computation time of simulation model optimization is spread out: for example, for $|V| = 2500$, the time to 95% solution quality varies from 15 s to 140 s. For application in a real-world control room, a near ten-fold variance in the time to solution is unacceptable.

The difference in (variance of) computation time between simulation model optimization and sequential simulation–optimization, both solved using a metaheuristic, is mainly caused by the different computation time per function evaluation. The number of function evaluations to convergence is similar for both simulation–optimization configurations. Each function evaluation, a full-fletched simulation model containing both fugitive and police entities is run to completion. This is obviously more time consuming than sequential simulation–optimization, where, after preprocessing, each function evaluation involves comparing matrices to count interceptions.

Some problem instances with $|U| = 9$ or $|U| = 10$ pose difficulty for the CBC MIP solver. Most likely, the corresponding starting configurations do not allow for easy exploitation of the problem structure by CBC, and force the algorithm to evaluate many candidate solutions. These problem instances do not cause a similar effect on the convergence speed of the metaheuristic.

4.2. Casus 2: real-world road network

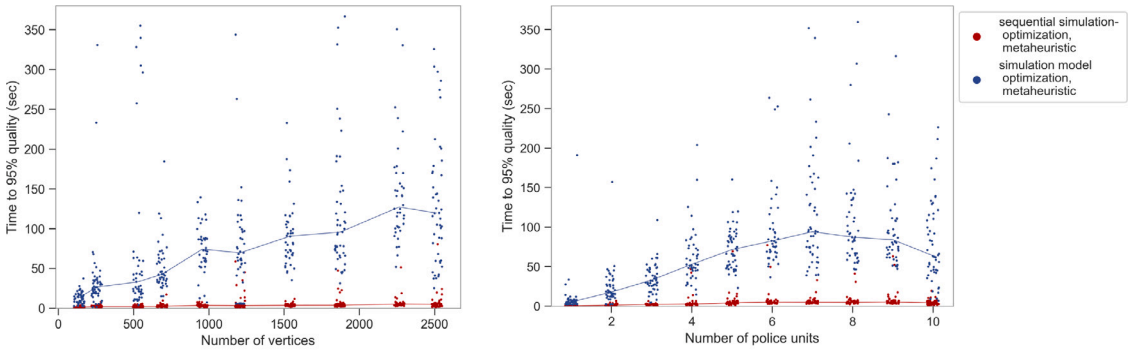
The same set of experiments on the road network of Rotterdam, the Netherlands, demonstrate the generalizability of the results to different graph topologies. Due to the discretization error in the optimization on city road networks, we do not consider the exact optimization approach. Fig. 7 shows that the computation time of simulation model optimization further increases compared to the experiments on a simple grid due to additional computational overhead. The reliability of both solution approaches is lower due to the larger influence of the starting positions of the fugitive and police units on the number of feasible permutations.

With an increasing number of police units to be positioned, the median computation time increases gradually until $|U| = 7$, after which it decreases. Evidently, it is relatively easy to find high-quality interception strategies using many police units on a real-world road network compared to a grid. There are relatively fewer good interception positions, which decreases the solution space, leading to faster convergence.

4.3. Discussion

The experiments show that sequential simulation–optimization significantly decreases the computation time compared to simulation model optimization of the same problem. Sequential simulation–optimization lends itself well to problems where the external, uncontrollable factors can be separated from the controllable factors. Simulation–optimization is often approached as the 'bolting on' of an optimization engine on an existing simulation model [25]. This approach is not easily transformed into a sequential simulation–optimization model. Firstly, the problem at hand should be suitable for implementation in the particular configuration. Modeling complexity in the controllable factors is more difficult compared to a simulation model optimization approach, as this complexity has to be described by constraints. In the considered fugitive interception problem, the external factor is the fugitive's behavior, and the controllable factor is the behavior of the police units. Hence, introducing more complex behavior of the police units is more difficult when using sequential simulation–optimization compared to simulation model optimization. Second, sequential simulation–optimization requires a specific formulation of the simulation model that yields the constraints for the optimization. This model formulation differs from the typical simulation model used to answer 'what-if'-type questions about the system under study.

The interception problem described in this paper is an example of a class of problems where an optimal intervention has to be determined independent of the uncertainty in the system. Therefore, controllable and uncontrollable components are separable into



(a) Varied number of vertices (b) Varied number of police units

Fig. 7. Comparison of the computation time for a varying problem size for simulation model optimization and sequential simulation–optimization. The lines indicate the median time to 95% solution quality for each approach.

optimization and simulation, respectively. Another example of this class of problems is the control of an autonomous vehicle. The best control action has to be determined, meaning that the car follows its intended route and avoids crashes, while the behavior of the vehicles around it is unknown. Similar to the fugitive interception problem discussed in this paper, the control action must be available quickly — within a second or even less. To accomplish this using sequential simulation–optimization, the possible trajectories of the nearby road users are simulated. A robust optimization yields the control action [59]. Sequential simulation–optimization is a promising way to quickly generate a large ensemble of plausible routes, incorporate these in an optimization problem, and find the optimal control action for the autonomous vehicle.

5. Threats to validity

The first threat to validity concerns the experimental setup. The computational experiments are based on a default configuration with 900 vertices (a 30×30 square grid or an 8 km^2 patch of the Rotterdam road network) and 500 predicted escape routes for the fugitive. In a typical fugitive route-choice simulation, approximately 500 routes are adequate to describe the plausible escape routes of the fugitives if only considering highways. If minor roads are included, the additional intersections cause this number to increase. A network size of approximately 1000 vertices is required to adequately describe the main network around a typical urban incident location. If the network is extended to minor roads, this number increases rapidly, analogous to the number of predicted escape routes required. In practice, the road network grows with the desired length of the planning horizon. The number of vertices that describe the road network can be reduced through graph coarsening [60]. However, the extent to which graph coarsening can be applied is limited: interception loses its real-world meaning if the network used in the model is too coarse. Therefore, the rapid increase in computation time with increasing vertices is troublesome, as the timely calculation of relevant solutions is threatened.

Secondly, the tested approaches may be more or less suitable for further reduction of the computation time. There are two main approaches to reducing the computation time of both simulation–optimization configurations: (1) increasing the computation power, for example, through further parallelization with High-Performance Computing; (2) improving the optimization algorithm. The former reduces the computation time per function evaluation. Increasing the computation power improves the absolute computation time, but the computation time scales the same way with increasing problem size as presented in this paper. The same holds for replacing Python with a computationally more efficient language, such as C. Furthermore, the improvement from parallelization is dependent on the specific optimization algorithm used. For many algorithms, for example, those based on hill-climbing, the optimization algorithm depends on the simulation model's output for its next proposed set of values for the decision variables. In this case, the improvement from parallelization is limited. In this paper, the applied algorithm, the input space, and the solution space are the same for both compared simulation–optimization approaches. Therefore, each of the discussed approaches to further reduce the computation time is expected to affect the computation time of both approaches similarly.

6. Conclusion

Simulation–optimization can be used to support real-time decision making for fugitive interception. To be useful for real-time decision making, timely calculation of the optimal solution is essential. Besides increasing computation power and algorithm efficiency, the configuration in which simulation and optimization are combined can reduce the computation time of simulation–optimization of large problems.

This paper examined the scaling of computation time with increasing problem size for two configurations of simulation–optimization: (1) sequential simulation–optimization: the output of a simulation model describes (part of) the constraints of an optimization model; (2) (common) simulation model optimization: a simulation model evaluates values for the decision variables proposed by an optimization algorithm to find the values that maximize or minimize the outcome(s) of interest, which are calculated through the simulation model. Our analysis using the fugitive interception example showed that

1. Sequential simulation–optimization vastly outperforms simulation model optimization in terms of computation time, especially for large problem instances;
2. Metaheuristic solution approaches reach a high quality of solutions in a fraction of the computation time of exact optimization algorithms.
3. Experiments on a real-world city road network demonstrate that these findings hold for various graph topologies

These results are generalizable to a class of problems where an optimal intervention has to be determined independent of the uncertainty in the system. In other words, separating controllable and uncontrollable components into optimization and simulation, respectively, leads to a significant reduction in the computation time.

Data availability

Link to code added at the 'Attach File' step.

Software&Code (Original data) (Github)

References

[1] G. Phillips-Wren, M. Adya, Decision making under stress: the role of information overload, time pressure, complexity, and uncertainty, J. Decis. Syst. 29 (2020) 213–225.
[2] G. Skinner, B. Parrey, A literature review on effects of time pressure on decision making in a cyber security context, J. Phys. Conf. Ser. 1195 (2019) 012014.
[3] B. Alspach, Searching and sweeping graphs: A brief survey, Le Mat. 59 (2004) 5–37.
[4] T.H. Chung, G.A. Hollinger, V. Isler, Search and pursuit-evasion in mobile robotics: A survey, Auton. Robots 31 (4) (2011) 299–316.
[5] M. Raap, M. Preuß, S. Meyer-Nieberg, Moving target search optimization – A literature review, Comput. Oper. Res. 105 (2019) 132–140.
[6] E. Sava, C. Twardy, R. Koester, M. Sonwalkar, Evaluating lost person behavior models, Trans. GIS 20 (2016) 38–53.
[7] S. Ivić, B. Crnković, H. Arbabi, S. Loire, P. Clary, I. Mezić, Search strategy in a complex and dynamic environment: the MH370 case, Sci. Rep. 10 (2020) 19640.
[8] B. van Dijk, J.B. Terpstra, P. Hulshof, Heterdaadkracht in Twee Haagse Pilotgebieden (in Dutch), Politie & Wetenschap, Amsterdam, 2013.
[9] G. Figueira, B. Almada-Lobo, Hybrid simulation–optimization methods: A taxonomy and discussion, Simul. Model. Pract. Theory 46 (2014) 118–134.
[10] M.C. Fu, Overview of the handbook, in: Michael C. Fu (Ed.), Handbook of Simulation Optimization, Springer, New York, NY, 2015, pp. 1–8.
[11] R.E. Shannon, Introduction to the art and science of simulation, in: D.J. Medeiros, E.F. Watson, J.S. Carson, M.S. Manivannan (Eds.), Proceedings of the 1998 Winter Simulation Conference, IEEE, Washington, DC, USA, 1998, pp. 7–14.
[12] R.D. Tordecilla, A.A. Juan, J.R. Montoya-Torres, C.L. Quintero-Araujo, J. Panadero, Simulation-optimization methods for designing and assessing resilient supply chain networks under uncertainty scenarios: A review, Simul. Model. Pract. Theory 106 (2021) 102166.
[13] M.C. Fu, Optimization via simulation: A review, Ann. Oper. Res. 53 (1) (1994) 199–247.
[14] L.J. Hong, B.L. Nelson, J. Xu, Discrete optimization via simulation, in: M.C Fu (Ed.), Handbook of Simulation Optimization, Springer, New York, NY, 2015, pp. 9–44.
[15] M.C. Fu, Optimization for simulation: Theory vs. Practice, INFORMS J. Comput. 14 (3) (2002) 192–215.
[16] H.R. Maier, S. Razavi, Z. Kapelan, L.S. Matott, J. Kasprzyk, B.A. Tolson, Introductory overview: Optimization using evolutionary algorithms and other metaheuristics, Environ. Model. Softw. 114 (2019) 195–213.
[17] S. Amaran, N.V. Sahinidis, B. Sharda, S.J. Bury, Simulation optimization: A review of algorithms and applications, Ann. Oper. Res. 240 (1) (2016) 351–380.
[18] Y. Carson andA.Anu Maria, Simulation optimization: methods and applications, in: S. Andradóttir, K.J. Healy, D.H. Withers, B.L. Nelson (Eds.), Proceedings of the 1997 Winter Simulation Conference, ACM Press, New York, New York, USA, 1997, pp. 118–126.
[19] S. Andradóttir, A review of simulation optimization techniques, in: D.J. Medeiros, E.F. Watson, J.S. Carson, M.S. Manivannan (Eds.), Proceedings of the 1998 Winter Simulation Conference, IEEE, Washington, Washington DC, 1998, pp. 151–158.
[20] M.C. Fu, F.W. Glover, J. April, Simulation optimization: A review, new developments, and applications, in: F.B. Armstrong, J.A. Joines, N. Steiger, M.E. Kuhl (Eds.), Proceedings of the 2005 Winter Simulation Conference, IEEE, Orlando, FL, USA, 2005, pp. 83–95.
[21] W.T. Garrison, M.D. Petty, A comparison of simulation optimization algorithm performance, in: Proceedings of SoutheastCon, IEEE, Huntsville, AL, USA, 2019, pp. 1–7,
[22] A. Sharma, E. Kosasih, J. Zhang, A. Brintrup, A. Calinescu, Digital twins: State of the art theory and practice, challenges, and open research questions, J. Ind. Inf. Integr. 30 (2022) 100383.
[23] N.A. Dong, D.J. Eckman, X. Zhao, S.G. Henderson, M. Poloczek, Empirically comparing the finite-time performance of simulation-optimization algorithms, in: W.K.V. Chan, A. D'Ambrogio, G. Zacharewicz, N. Mustafee, G. Wainer, E. Page (Eds.), Proceedings of the 2017 Winter Simulation Conference, IEEE, Las Vegas, NV, USA, 2017, pp. 2206–2217.
[24] S. Ghadimi, G. Lan, Stochastic approximation methods and their finite-time convergence properties, in: M.C. Fu (Ed.), Handbook of Simulation Optimization, Springer, New York, NY, 2015, pp. 179–206.
[25] S.G. Henderson, Reflections on simulation optimization, in: S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, M. Loper (Eds.), Proceedings of the 2021 Winter Simulation Conference, IEEE, Piscataway, NJ, USA, 2021, pp. 1–15.
[26] J. de Armas, A.A. Juan, J.M. Marquès, J.P. Pedroso, Solving the deterministic and stochastic uncapacitated facility location problem: from a heuristic to a simheuristic, J. Oper. Res. Soc. 68 (10) (2017) 1161–1176.
[27] J. Kleijnen, R. Sargent, A methodology for fitting and validating metamodels in simulation, European J. Oper. Res. 120 (2000) 14–29.
[28] J. Nocedal, S. Wright, Numerical Optimization, second ed., Springer, New York, NY, USA, 2006.
[29] B. Zeigler, H. Prähofer, T.G. Kim, Theory of Modeling and Simulation, second ed., Academic Press, San Diego, CA, USA, 2000, pp. 27–41.
[30] X. Xi, R. Sioshansi, V. Marano, Simulation–optimization model for location of a public electric vehicle charging infrastructure, Transp. Res. Part D: Transp. Environ. 22 (2013) 60–69.
[31] C. McPhail, H.R. Maier, J.H. Kwakkel, M. Giuliani, A. Castelletti, S. Westra, Robustness metrics: How are they calculated, when should they be used and why do they give different results? Earth's Future 6 (2) (2018) 169–191.
[32] M.R. Wigan, The fitting, calibration, and validation of simulation models, Simulation 18 (1972) 188–192.
[33] T.C. van Barneveld, S. Bhulai, R.D. Van Der Mei, The effect of ambulance relocations on the performance of ambulance service providers, European J. Oper. Res. 252 (1) (2016) 257–269.

[34] F. Azadivar, Simulation optimization methodologies, in: Phillip A. Farrington, Harriet Black Nembhard, David T. Sturrock, Gerald W. Evans (Eds.), Proceedings of the 1999 Winter Simulation Conference, IEEE, New York, New York, USA, 1999, pp. 93–100.
[35] N. Gülpınar, B. Rustem, R. Settergren, Simulation and optimization approaches to scenario tree generation, J. Econom. Dynam. Control 28 (2004) 1291–1315.
[36] R. Borie, C. Tovey, S. Koenig, Algorithms and complexity results for graph-based pursuit evasion, Auton. Robots 31 (2011) 317–332.
[37] A. Hashimoto, L. Heintzman, R. Koester, N. Abaid, An agent-based model reveals lost person behavior based on data from wilderness search and rescue, Sci. Rep. 12 (2022) 5873.
[38] R.J. Koester, Lost Person Behavior: A Search and Rescue Guide on Where to Look for Land, Air, and Water, dbS Productions, Charlottesville, VA, 2008.
[39] M.J. Hodgson, A flow-capturing location-allocation model, Geogr. Anal. 22 (1990) 270–279.
[40] O. Berman, R.C. Larson, N. Fouska, Optimal location of discretionary service facilities, Transp. Sci. 26 (3) (1992) 201–211.
[41] M. Gendreau, G. Laporte, I. Parent, Heuristics for the location of inspection stations on a network, Nav. Res. Logist. 47 (4) (2000) 287–303.
[42] K. Tanaka, O. Kurita, The probabilistic minisum flow interception problem: minimizing the expected travel distance until intercept under probabilistic interception, Geogr. Anal. 52 (2) (2020) 211–230.
[43] P.H.M. Jacobs, The DSOL Simulation Suite: Enabling Multi-Formalism Simulation in a Distributed Context (Ph.D. thesis), Delft University of Technology, 2005, URL http://resolver.tudelft.nl/uuid:4c5586e2-85a8-4e02-9b50-7c6311ed1278.
[44] M. Boccia, A. Sforza, C. Sterle, Flow intercepting facility location: problems, models and heuristics, J. Math. Model. Algorithms 8 (1) (2009) 35–79.
[45] T. Ralphs, CBC: COIN-OR branch-and-cut solver, 2022, Version 2.10.8.
[46] F. Rothlauf, Design of modern heuristics: principles and application, in: F. Rothlauf (Ed.), Natural Computing Series, Springer, Berlin, Heidelberg, 2011, pp. 45–102.
[47] M. Abdel-Basset, L. Abdel-Fatah, A.K. Sangaiah, Chapter 10 - metaheuristic algorithms: A comprehensive review, in: A.K. Sangaiah, M. Sheng, Z. Zhang (Eds.), Intelligent Data-Centric Systems, Academic Press, 2018, pp. 185–231.
[48] H. Mühlenbein, M. Gorges-Schleuter, O. Krämer, Evolution algorithms in combinatorial optimization, Parallel Comput. 7 (1) (1988) 65–85.
[49] A. Tolk, Simulation-based optimization: Implications of complex adaptive systems and deep uncertainty, Information 13 (10) (2022) 469.
[50] E.A. Torres, S. Khuri, Applying evolutionary algorithms to combinatorial optimization problems, in: Proceedings of the International Conference on Computational Science, ICCS '01, Springer-Verlag, Berlin, Heidelberg, 2001, pp. 689–700.
[51] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[52] D. Hadka, P.M. Reed, Borg: an auto-adaptive many-objective evolutionary computing framework, Evol. Comput. 21 (2) (2013) 231–259.
[53] D.J. Eckman, S.G. Henderson, S. Shashaani, Diagnostic tools for evaluating and comparing simulation-optimization algorithms, INFORMS J. Comput. 35 (2) (2023) 350–367.
[54] F. Bode, P.M. Reed, S. Reuschen, W. Nowak, Search space representation and reduction methods to enhance multiobjective water supply monitoring design, Water Resour. Res. 55 (2019) 2257–2278.
[55] I.S. van Droffelaar, J.H. Kwakkel, J.P. Mense, A. Verbraeck, Simulation-optimization configurations for fugitive interception, in: B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, P. Lendermann (Eds.), Proceedings of the 2022 Winter Simulation Conference, IEEE, Singapore, 2022, URL https://informs-sim.org/wsc22papers/pos133.pdf.
[56] Delft High Performance Computing Centre (DHPC), Delftblue supercomputer (phase 1), 2022, https://www.tudelft.nl/dhpc/ark/44463/DelftBluePhase1.
[57] A. Rydzewski, P. Czarnul, Recent advances in traffic optimisation: systematic literature review of modern models, methods and algorithms, IET Intell. Transp. Syst. 14 (13) (2020) 1740–1758.
[58] G. Boeing, OSMNX: New methods for acquiring, constructing, analyzing, and visualizing complex street networks, Comput. Environ. Urban Syst. 65 (2017) 126–139.
[59] M. Zanon, J. Frasch, M. Vukov, S. Sager, M. Diehl, Model predictive control of autonomous vehicles, in: H. Waschl, I. Kolmanovsky, M. Steinbuch, L. del Re (Eds.), Optimization and Optimal Control in Automotive Systems, Springer, Cham, Switzerland, 2014.
[60] P. Krishnakumari, O. Cats, H. van Lint, Heuristic coarsening for generating multiscale transport networks, IEEE Trans. Intell. Transp. Syst. 21 (6) (2020) 2240–2253.