Robotica (2023), **41**, pp. 1568–1589
doi:10.1017/S0263574722001886

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Coordinated flight path planning for a fleet of missiles in high-risk areas

Luitpold Babel

Institut für Mathematik und Informatik, Fakultät Betriebswirtschaft, Universität der Bundeswehr München, 85579 Neubiberg, Germany
Email: luitpold.babel@unibw.de

Received: 1 July 2022; **Revised:** 12 November 2022; **Accepted:** 16 December 2022; **First published online:** 16 January 2023

Keywords: coordinated flight path planning, swarm attack, offline planning, online planning, maximum clique, branch and bound

Abstract

This paper addresses the flight path planning problem for multiple missiles engaging stationary targets in high-risk areas. Targets protected by air defence are preferably engaged by a fleet or swarm of missiles, not individual missiles. The concept of a swarm attack is that a large number of approaching missiles overwhelm air defence. The deployment of missiles is often part of a broader mission including further participants. Flight path planning is then an integral element of mission planning, requiring strict timing coordination of all members involved. The flight times of the missiles are dictated by the master planning. We present algorithms for offline planning and online re-planning of flight paths for a fleet of missiles with flight time constraints. The algorithms are based on an advanced bidirectional RRT* algorithm that generates risk-minimizing flight paths with predefined flight times. Online planning generates the flight paths of the fleet sequentially, maintaining a safety distance between the missiles to prevent mutual collision. Offline planning uses a global optimization approach to determine an optimal selection of flight paths from a large set of potential paths. The selection is performed by a branch and bound algorithm that determines optimal cliques in the path compatibility graph. The optimization is embedded in an iterative algorithm that allows to successively improve the mission success.

1. Introduction

The advent of unmanned aerial vehicles (UAVs) and their use in fleet formations has revolutionized aerial warfare. Unarmed UAVs are intended for surveillance, identification and reconnaissance, while armed aircraft, including unmanned combat aerial vehicles (UCAVs) and cruise missiles, are applied for target engagement. It can be assumed that the capabilities of a joint system go far beyond the sum of its parts. Operations can be performed more effectively through cooperation and coordination of the team members. This applies even more if the aircraft fleet is embedded in a joint mission with other participants (see e.g., refs. [1] and [2]).

The background to this work is the deployment of a fleet or a swarm of missiles or UCAVs against one or more stationary targets in a high-risk area. The missiles are released from a common platform or from different platforms and must arrive at the targets simultaneously. The concept behind a swarm attack is that a large number of missiles overextends the hostile air defence. Mission objectives are achieved by exploiting saturation effects. Accepting the loss of individual missiles, a sufficiently large number of aircraft will survive and engage the targets.

A successful mission requires spatial and temporal coordination of missiles to avoid mutual collision and collision with terrain and obstacles. The risk of the missiles must be kept low by avoiding all conceivable threats as far as possible. Other critical aspects include suitable assignments of missiles to targets and determining optimal attack directions. Finally, it should be noted that the missiles could be

part of more comprehensive operation including further participants. Hence, planning of the missiles could be an integral part of a mission planning system that dictates additional constraints such as arrival times for the missiles at the targets.

Over the past decades, numerous methods for path planning have been developed. This includes, among others, sampling-based methods such as probabilistic roadmaps and rapidly exploring random trees (RRTs), cell decomposition, Voronoi diagrams, the potential field method, mixed integer linear programming, network-based search algorithms such as the A*-algorithm and diverse variants and artificial intelligence-based techniques. These universal path finding algorithms are then adapted to the specific requirements of an aerial mission. For comprehensive surveys of the proposed techniques, see refs. [3] and [4].

Recently, the cooperation of a group of UAVs has attracted more and more attention which is reflected in a rapidly growing number of publications. The activities cover flight path planning with a wide area of different constraints and mission objectives. The former contains kinematic properties of the aircraft, environmental constraints such as obstacle and threat avoidance, allocation of tasks such as assigning aircraft to targets, and coordination constraints including collision avoidance and temporal synchronization of UAVs. The objective function is to minimize the total costs, where costs may be composed of flight time, path length, fuel consumption, risk of the aircraft and others.

Several surveys have been conducted on cooperative path planning of UAVs. The most recent ones include ref. [5] which provides a systematic review from the perspective of optimization and with focus on the analysis and classification of cooperative path planning problems. A review from the perspective of cooperation techniques is given in ref. [3]. The survey [6] pays special attention to flexible formation shape to achieve collision avoidance for multi-vehicle systems. For a review of artificial intelligence applied to path planning in UAV swarms, including reinforcement learning techniques, evolutionary computation, swarm intelligence-based methods and graph neural networks, see ref. [7].

The focus of this work is on the generation of flight paths for a fleet of missiles engaging one or several stationary targets in high-risk areas. The flight paths must be coordinated with respect to space and time and optimized for mission success. This includes to assign missiles to targets and to minimize the risk of the missiles from different types of threats. Another key issue is meeting specified flight times, which are dictated by a master mission plan that integrates the fleet into a joint operation. Flight paths must be planned offline prior to launch of the missiles, but also online during flight to be able to react to dynamically changing environments. To the best of our knowledge, no studies have been published dealing with risk minimization of a fleet of missiles under flight time constraints in strongly protected areas.

In a previous work, see ref. [8], we studied the target assignment and UAV path planning problem with flight time constraints in scenarios with no-fly areas. The problem has been addressed by constructing a sophisticated network and applying graph search algorithms to find shortest paths in the network. After assigning aircraft to targets, flight paths are prolonged to meet the desired flight times by automatic insertion of waypoints. The main drawback is that the method is not designed to minimize the risk of the aircraft. It generates flight paths in the horizontal plane. However, threat avoidance is based to a large extent on the exploitation of terrain masking effects and requires optimization of flight paths in the three-dimensional space. Moreover, the algorithm is too time-consuming to be applicable for online planning.

The approach presented in this paper uses the RRT* Fit Algorithm (see ref. [9]) which generates flight paths with predefined flight time in scenarios with no-fly areas and moving obstacles. The algorithm is inspired by the classical RRT* Algorithm and uses three-dimensional Dubins paths to take into account flight properties of the aircraft. Flight time constraints are realized by concatenating two growing RRTs by Dubins paths with varying radii. Here, the algorithm is further extended to find flight paths with predefined flight time and minimum risk in scenarios with known and potential unknown threats.

The modified RRT* Fit Algorithm is the basis for planning and coordinating flight paths for a fleet of missiles. The intention is to develop algorithms which can be applied for offline planning and online re-planning. Offline planning requires comprehensive information about the environment, including

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

obstacles, no-fly areas and threats. It is performed prior to starting the mission and is not subject to high time pressure. Online planning uses real-time environmental data which are obtained from on-board sensors or external sources. It is performed during the mission and must provide solutions very quickly. Online planning can rely on results obtained from offline planning and must be able to re-plan flight paths during the flight.

The proposed method for online planning is based on sequential optimization of flight paths. It adopts the assignment of missiles to targets obtained from offline planning and applies the modified RRT* Fit Algorithm to re-plan flight paths. The flight paths are generated one after the other, considering the flight paths of previously planned missiles as moving obstacles for the actual missile. It turns out that the algorithm provides very satisfying results within very short computation times, however, without the claim of optimality.

Offline planning pursues the concept of global optimization. The key is the conversion into a combinatorial optimization problem, which consists of finding a maximum weight clique in the path compatibility graph. This graph represents a large number of different flight paths for the missiles, with the information which of the paths are compatible with each other. An optimal clique provides an appropriate assignment of missiles to targets with optimized attack directions, along with optimized coordinated flight paths for the missiles. The optimization technique is embedded in an iterative algorithm that allows to successively improve the flight paths. The algorithm produces high-quality solutions with short lead times.

The remainder of this paper is structured as follows. In Section 2, the flight path algorithm that generates flight paths with predefined flight times is revisited and extended to an algorithm minimizing the risk of the aircraft. The algorithms for coordinated flight path planning are discussed in Section 3. We sketch a framework for path planning, embedded in an integrated mission planning system, and present the algorithms for online and offline planning. For offline planning, we develop a branch and bound algorithm that finds a clique with maximal weight in the path compatibility graph. Section 4 includes computational results demonstrating the capability of the algorithms. We conclude in Section 5 with a summary and perspectives on future research.

2. Flight path planning with flight time constraints

2.1. RRT fit algorithm*

The classical flight path optimization problem is to find a flight path for an air vehicle from a start point to a destination point taking into account the flight capabilities of the vehicle and avoiding obstacles and no-fly areas. The aim is to minimize the flight path length, the flight time, the fuel consumption or some other objective. While the classical flight path optimization problem has been extensively investigated, less is known about finding paths of predefined length or flight time.

However, compliance with flight time constraints is critical for many missions, especially for cooperating units such as fleets of missiles pursuing a common goal. Requirements to flight path planning of the units could be to arrive at the targets simultaneously, within a limited time window, or in a predefined temporal sequence to exploit saturation effects of air defence. In this section, we revisit and extend an algorithm (see ref. [9]) that generates flight paths with predefined flight times in dynamic scenarios with static and moving obstacles.

The idea of the RRT algorithm (see ref. [10]) is to build a space-filling tree by drawing random samples from the search space and connecting them to the nearest node in the tree. The algorithm was further developed to the RRT* algorithm (see ref. [11]) by optimizing the connections of the tree. A near neighbour search allows to select the best predecessor node for the new node before inserting it into the tree. A rewiring operation restructures the tree by removing and adding edges from the tree. The purpose of both procedures is to shorten the distances from the nodes to the root of the tree.

The RRT* algorithm connects nodes of the tree by straight lines. We consider the kinematic constraints of the missile by modelling it as a Dubins vehicle and replace straight lines by three-dimensional

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

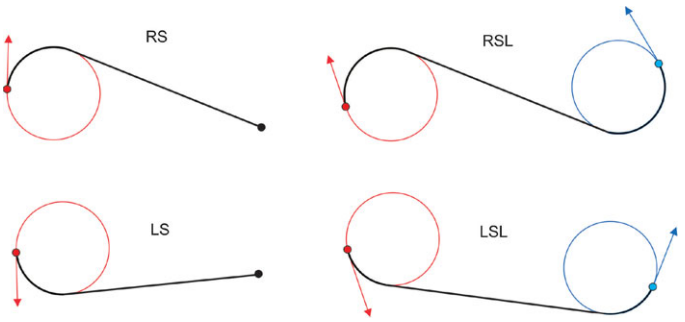

Figure 1. *Dubins paths of type CS and CSC.*

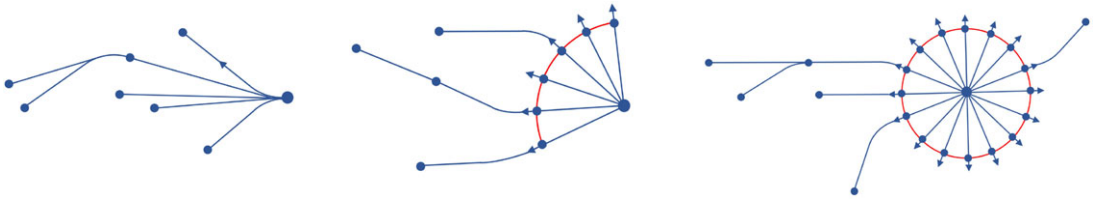

Figure 2. *Root of T-tree with fixed approach direction or with approach sectors.*

Dubins paths. The obtained paths in the tree consist of concatenations of circle segments and straight lines. The radius of the circle segments corresponds to the turning radius of the missile, and the inclination angle of the path segments is limited by the maximum angle of climb and descent.

Each node of the tree is assigned a direction corresponding to the heading of the missile. The root node is assigned the release direction. The heading of a new node added to the tree results from the shortest connection between the predecessor node and the new node. The corresponding paths are of type CS, meaning that a circle segment (right or left bent) is followed by a straight line segment. Figure 1 on the left shows two examples of Dubins paths between a node with prescribed heading (the predecessor node) and a node with free heading (the new node).

Taking up an idea of ref. [12], we construct two RRTs rooted in the start and target point, respectively, thus realizing a bidirectional search. The *S*-tree is rooted in the start point, and the *T*-tree is rooted in the target point (see Fig. 3, left). In view of the application discussed later in this paper, we modify the *T*-tree by allowing an approach sector instead of a fixed approach direction. The *T*-tree is initialized by a star-shaped sub-tree which contains the possible directions (see Fig. 2). The tree is extended by attaching path segments to the ends of the branches.

The key idea of the RRT* Fit Algorithm is to concatenate the trees in such a way that the required flight path length (or flight time) is obtained. Two nodes of different trees with prescribed headings are connected by a Dubins paths of type CSC (see Fig. 1, right) consisting of a circle segment, a straight line, and a second circle segment. The length of the Dubins path can be varied by changing the radius of the circles. Using a bisection method to speed up the computation, the radius is modified until a flight path with the required length is obtained. Figure 3 shows on the right side three of many connecting Dubins paths as examples.

The algorithm avoids collisions of the missile with terrain and with static obstacles, but also with moving objects, by including only path segments in the trees guaranteeing a safe flight. This applies both to objects with known and unknown route. In the latter case, the algorithm uses a forecast model predicting the motion of the object and recalculates the flight path if a route change endangers the missile.

Avoiding collision with a moving object, for instance with another missile, means maintaining a specified safety distance. A collision is indicated when the distance between the missiles falls below the safety margin. Figure 4 shows two flight paths with timestamps which are crossing. However, the missiles

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

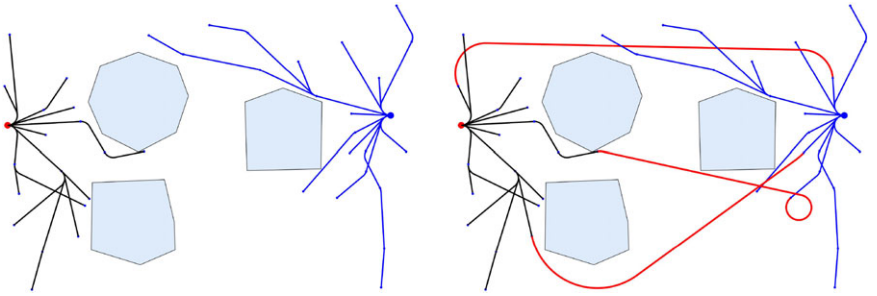

Figure 3. *S-tree and T-tree connected by Dubins paths of type CSC.*

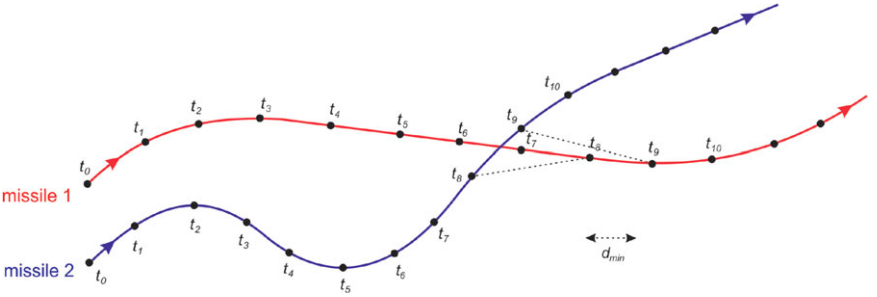

Figure 4. *Flight paths keeping safety distance.*

do not endanger each other, since the safety distance $d_{\min}$ is preserved throughout the flight. Before adding a new flight path segment to the tree, the algorithm checks whether the segment collides with a route of a moving object. If this is the case, the flight path segment cannot be used and is discarded.

The algorithm terminates as soon as a flight path with the required flight time has been found. For more details, we refer to ref. [9].

2.2. Minimum risk flight path with predefined flight time

A missile or an UCAV operating in a hostile environment may be subject to known and unknown threats. The first includes anti-aircraft artillery (AAA) and surface-to-air missile (SAM) systems with known position and radar range. In addition, there is also a potential risk from unpredictable pop-up threats. This may include hidden air defence sites, non-localized man portable air defence systems (MANPADS) and others. Considering threats leads to the even more challenging problem of finding flight paths with smallest possible risk.

Minimizing the risk of a missile corresponds to maximizing its survivability. A realistic estimate of the probability of a missile reaching a target is difficult to achieve and requires advanced threat models. Our model for known threats includes the radar cross section of the missile as a function of the aspect angle. It also considers characteristic properties of the radar site, atmospheric conditions, and the line of sight between radar site and missile to assess the detection probability of the missile. In addition to detection, the model also includes interception of the missile by the defender.

The model for unknown threats takes into account the degree of exposure of the missile with respect to terrain, as well as reconnaissance information about potential hostile forces. Flight path planning aims to expose the missile to the least possible extent by generating low-level flight paths that exploit terrain features for masking. Relevant information about the visibility of points in the space is preprocessed and stored in maps, enabling fast planning.

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

Using the outlined models, the RRT* Fit Algorithm is extended to generate flight paths with predefined flight time and lowest possible risk. Flight paths are evaluated according to the survival probability of the missile. The missile survives on its flight to the target if it survives both known and unknown threats. The missile gets lost if it is detected and intercepted. Hence,

$$p_{survive} = \tilde{p}_{survive} \cdot \hat{p}_{survive} = (1 - \tilde{p}_{detect} \cdot \tilde{p}_{intercept}) \cdot (1 - \hat{p}_{detect} \cdot \hat{p}_{intercept}) \tag{1}$$

where $\tilde{p}$ and $\hat{p}$ denote the respective probabilities with respect to known and unknown threats.

The detection probability $\tilde{p}_{detect}$ takes into account whether the missile is within radar range and is illuminated by the radar for a sufficient period of time. This means that no obstacles and terrain must obscure the missile. It is checked whether the aspect angle of the missile results in a radar cross section that returns a sufficiently intense echo to the radar site, and whether the reception power of the radar is sufficiently high.

The interception probabilities $\tilde{p}_{intercept}$ and $\hat{p}_{intercept}$ indicate whether AAA or SAMs are capable of intercepting the missile, either by a direct hit or blast, depending on the warhead. The detection probability $\hat{p}_{detect}$ depends on the visibility of the missile along its flight path, as well as the probability that the missile is within the range of unknown threats. A detailed description of the threat models and the calculation of the probabilities is omitted, as this would go far beyond the scope of this paper.

As stated before, the RRT* Fit Algorithm generates two RRTs by adding flight path segments in the form of Dubins paths. Instead of considering only length, each flight path segment is also associated with costs including the survival probability of the missile along the path segment. Since the survivability along a flight path is computed multiplicatively from the survivabilities along the individual flight path segments, that is,

$$p_{survive} = \prod_{S \in P} p_{survive}(S) \tag{2}$$

for all flight path segments S of the flight path P, the probabilities are transformed by taking the negative logarithm:

$$c(S) = -\log p_{survive}(S) \tag{3}$$

Then maximizing the product of the probabilities is equivalent to minimizing the sum of the costs $c(S)$ along the path P. This follows from the transformation:

$$\log p_{survive} = \log \prod_{S \in P} p_{survive}(S) = \sum_{S \in P} \log p_{survive}(S) \tag{4}$$

and the fact that maximizing a function $f(x)$ is equivalent to minimizing the negative function $-f(x)$.

The near neighbour search and the tree rewiring operation of the RRT* Fit Algorithm are modified by choosing flight path segments providing minimal costs instead of minimal lengths from the nodes to the root of the tree. The concatenation of the trees by Dubins paths of type CSC remains unchanged. The algorithm iteratively improves the solution by generating more and more flight paths of predefined length (or flight time) and selecting the path with the highest survivability. The algorithm terminates when the maximal calculation time has elapsed or the desired survivability is reached.

3. Coordinated flight path planning

3.1. Integrated mission planning

The deployment of a fleet or a swarm of missiles is often embedded in a more complex mission that includes further airborne and land-based weapon systems cooperating in a network-centric operation. The missile flight path planning is then an integral component of a mission planning that requires strict timing coordination of all participants involved. The flight times of the missiles are dictated by the master planning.

A saturation attack is a tactical method in which the attacking side attempts to overwhelm the air defence system by launching a large number of missiles. If the missiles approach in a very short period

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

of time, the system does not have enough time to react to every single missile. On the other hand, if the number of missiles is large enough, the system runs out of weapons and is not able to neutralize the large number of incoming threats. The strategy is to accept the loss of individual missiles, while others survive and reach the targets.

The missiles are released from a common or multiple land-, air- or sea-based platforms with the objective of engaging a single or multiple targets. The task of flight path planning is to find an optimal assignment of missiles to targets and to generate coordinated flight paths in compliance with the predefined flight times. Planning must prevent mutual collision of the missiles, collision of the missiles with terrain and collisions with obstacles. No-fly areas must not be passed, and threat areas should be avoided to the extent possible. To take advantage of saturation effects, the missiles must reach the targets simultaneously or successively within a small time window. The overall goal is to maximize mission success.

While offline planning has the claim to process all kind of available information about the scenario and the environment, online planning must deal with the situation that part of the information is not available in advance. This might include unforeseen obstructions detected by external sensors or by sensors of the missiles, or new threats popping up unexpectedly. Even a change in the time of arrival at the target or a change of the destination during the mission could occur due to operational reasons.

Offline planning is conducted prior to the launch of the mission and is not subject to strong limitations of computer runtime. The time horizon ranges from a few minutes up to several hours, depending on the urgency of the operation. In contrast, online planning takes place during the flight and must be extremely quick to immediately react to changing circumstances and unexpected events.

In the following, we present algorithms for offline planning with lead times in the order of tens of seconds to several minutes, depending on the complexity of the scenario and the desired quality of the solutions, and for online planning with computation times in the order of fractions of a second up to very few seconds. The algorithm for offline planning is based on a global optimization approach and ensures high-quality solutions. The algorithm for online planning uses a sequential optimization procedure that provides flight paths with slightly decreasing quality.

The algorithms for offline and online flight path planning can be integrated in a higher-level mission planning system. A possible framework could be as follows:

(1) Mission planning coordinates and synchronizes joint operations. The created mission plan defines the temporal restrictions for all parties involved in the mission, including the flight times for the missiles to engage the targets.

(2) Offline planning is performed with short lead time before launching the mission, taking into account all available information. The global optimization approach provides an optimal assignment of missiles to targets and the optimized coordinated flight paths with optimized attack directions.

(3) Online planning is performed during the mission whenever new information is obtained that requires re-planning of one or more flight paths. The sequential optimization method provides modified coordinated flight paths with appropriate attack directions.

While the algorithm for offline planning is able to simultaneously optimize target assignment and flight paths, online planning uses the assignment obtained from offline planning or a modified assignment obtained from an external source.

3.2. Online flight path planning by sequential optimization

The online planning algorithm is based on the minimum risk flight path algorithm presented in Section 2. It takes into account no-fly areas and threats and allows to consider predefined flight times for the individual missiles. The generated flight paths are coordinated with respect to space and time. The focus is on keeping the computation time of the algorithm small enough to be able to perform a re-planning of flight paths during the mission.

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

The idea of the algorithm is to generate the flight paths sequentially, one after the other, avoiding collisions with the missiles of the already planned paths. The assignment of missiles to targets is predetermined. Each missile has a specific flight time to its target, which is defined by the specified arrival time. If the flight times are not identical, the planning is done according to increasing flight times, that is, starting with the missile with the smallest flight time and ending with the missile with the largest time. The motivation is that paths with long flight times are usually more flexible in the sense that it is easier to avoid collisions with previously planned flight paths.

Each single flight path is computed using the minimum risk RRT* Fit Algorithm. It is crucial that the algorithm allows to consider moving obstacles. In sequential planning of the flight paths, the previously planned missiles can be regarded as moving obstacles for the actual missile. The approach is sketched in the algorithmic scheme below. It is formulated for the case that all flight paths have to be re-planned. The scheme applies accordingly if only one or a few flight paths are re-planned.

Online flight path algorithm

Input: start nodes $v_{start,i}$ and target nodes $v_{target,i}$ with flight times T_i, $i = 1, 2, \ldots, m$

Output: paths P_i between $v_{start,i}$ and $v_{target,i}$, $i = 1, 2, \ldots, m$

1. order the m missiles according to increasing flight times
2. such that $T_1 \leq T_2 \leq \ldots \leq T_m$
3. *for* $i = 1, 2, \ldots, m$ *do*
4. generate a flight path P_i for missile i from $v_{start,i}$ to $v_{target,i}$
5. with flight time T_i and smallest possible risk
6. avoiding collision with the flight paths $P_1, \ldots, P_{i-1}$
7. using algorithm RRT* Fit
8. *return* $P_1, \ldots, P_m$

The algorithm generates a set of collision-free flight paths. The flight path for missile i leads from start node $v_{start,i}$ to target node $v_{target,i}$. The location of the start node is the actual position of the missile. The direction assigned to the start node is the actual heading of the missile. If the approach direction of the target node is not specified, an optimal direction is provided by the RRT* Fit Algorithm (see Section 2.1). The flight time T_i is the remaining time for missile i until the planned arrival.

Sequential optimization corresponds to a greedy strategy that makes the locally optimal choice at each stage. A greedy algorithm generally does not produce an optimal solution but can provide locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time. The proposed algorithm maximizes the arrival probability of the missile for each path but does not take into account the paths that still need to be optimized. Flight paths of initially planned missiles might block flight paths of further missiles or prevent the existence of low-risk paths. Typical for a greedy strategy, the algorithm proves to be extremely fast but can provide suboptimal solutions.

The remainder of this section deals with an algorithm that overcomes the shortcomings of sequential optimization and optimizes flight paths globally. In contrast to the previous approach, the new algorithm will significantly improve the quality of solutions, however, at the cost of increased computation time.

3.3. Offline flight path planning by global optimization

While online planning requires the targets to be assigned to the missiles, the algorithm for offline planning is able to optimize the assignment. On the other hand, with minor adjustments, it can also process predefined assignments as input. The assignment and the arrival probabilities of the missiles at the targets decide on the success of the mission. In the context of this work, a target is considered successfully engaged if at least one missile reaches the target. The mission success is then expressed as the number of hit targets.

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

Assume first that m missiles are engaging a single target and let $p_1, p_2, \ldots, p_m$ denote the arrival probabilities of the missiles at the target. Then,

$$(1-p_1)\cdot(1-p_2)\cdot\ldots\cdot(1-p_m) \tag{5}$$

is the probability to *not* hit the target (all missiles miss the target). The probability to hit the target is

$$p_{hit} = 1 - \prod_{i=1}^{m}(1-p_i) \tag{6}$$

(at least one missile hits the target). The objective function of the optimization is to maximize the hit probability, that is,

$$p_{hit} \rightarrow \max \tag{7}$$

Suppose next that m missiles are engaging t targets and let $p_{hit,1}, p_{hit,2}, \ldots, p_{hit,t}$, denote the hit probabilities of the targets. Then,

$$E_{hit} = p_{hit,1} + p_{hit,2} + \ldots + p_{hit,t} = \sum_{j=1}^{t} p_{hit,j} \tag{8}$$

is the expected number of hit targets. The objective function of the optimization is to maximize the expectation, that is,

$$E_{hit} \rightarrow \max \tag{9}$$

The basic idea of global optimization for coordinated flight path planning can be summarized as follows:

(1) Generate, for each missile, a large number of diverse flight paths to all targets

(2) Find an optimal selection of mutually collision-free flight paths

The first step is to generate not just one, but a significant number of different flight paths. For each missile there should be as many alternatives as possible, with different targets and different attack directions. The flight paths must comply with the specified flight times and be collision-free with obstacles and terrain. Generation is not limited to low-risk paths but could also include flight paths with higher risk.

The second step is to find an optimal selection of flight paths, one path for each missile. The flight paths must be mutually collision-free which is ensured by keeping a predefined safety distance between the missiles. That is, the flight paths may be spatially close or even cross each other, but there must be an appropriate distance between the missiles following the paths. The flight paths are selected in such a way that the mission success is maximal. The optimal selection of flight paths provides, as a by-product, an optimal assignment of missiles to targets and optimal attack directions of the targets.

The selection of flight paths is realized by converting the problem into the combinatorial optimization problem of finding a maximum weight clique in a special graph. In this graph, hereafter referred to as the *path compatibility graph*, each flight path is represented by a vertex. Two vertices of the graph are connected by an edge if the associated flight paths are compatible, that is, the missiles following the flight paths do not collide and keep a given safety distance. The weights of the vertices correspond to the arrival probabilities of the missiles for the associated flight paths.

More formally, let m denote the number of missiles and n_i the number of flight paths associated with missile i, with $i = 1, 2, \ldots, m$. Each flight path is represented by a vertex $v \in V$, where the vertex set V corresponds to the set of all flight paths. Let V_i denote the vertex set with all flight paths belonging to missile i. Then, $V_1, V_2, \ldots, V_m$ is a partition of the vertex set, that is,

$$V = \bigcup_{i=1}^{m} V_i, \quad V_i \cap V_j = \emptyset, \quad i,j = 1,2,\ldots,m, \quad i \neq j \tag{10}$$

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

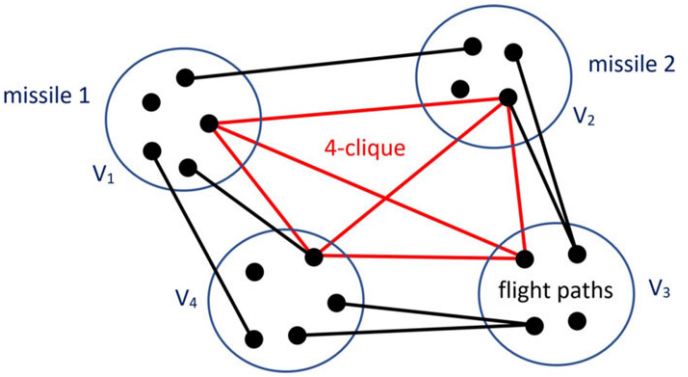

Figure 5. *Clique in path compatibility graph.*

The number of vertices is

$$|V| = n_1 + n_2 + \ldots + n_m = n \tag{11}$$

Two vertices u and v are connected by an edge, if the corresponding flight paths are compatible. We write $(u, v) \in E$ with E denoting the set of edges of the graph. The graph $G = (V, E)$ contains edges between different partitions only. The vertices within each partition V_i are not connected by edges and form a so-called *independent set*. The path compatibility graph G is an *m-partite graph*, that is, a graph with a partition of the vertices into m independent sets.

A *clique* is a set of vertices which are pairwise connected by an edge. A *k-clique* is a clique containing k vertices (see also the example in Fig. 5). Obviously, m-partite graphs do not contain cliques with more than m vertices. Each m-clique in the path compatibility graph corresponds to a collection of flight paths, one path for each missile, which are pairwise compatible.

Each vertex $v \in V$ is assigned, as a weight, the arrival probability of the missile. The classical maximum weight clique problem is to find a clique with maximal weight, where the weight of a clique is the sum of the weights of its vertices. In our application, a different objective function is applied. The weight of an m-clique corresponds to the mission success implied by the associated flight paths. To summarize, the optimization task is to find the m-clique in the path compatibility graph with maximal mission success.

3.4. Branch and bound algorithm for the clique problem

Finding a maximum clique in an arbitrary graph is an NP-hard problem (see ref. [13]), which means that it is very unlikely that an algorithm exists solving the problem in polynomial time. Despite of the intractability of the problem, algorithms have been developed which are capable to solve problems with up to several hundreds of vertices within a reasonable time (for a review see ref. [14]). The objective function of these algorithms is the number of vertices or the sum of the weights of the vertices in the clique. Unfortunately, since our application requires a different objective function, these algorithms are not suitable for our purpose.

The basis of our clique finding algorithm is an enumeration algorithm which returns a complete list of all m-cliques in an m-partite graph (see ref. [15]). Since we do not need all m-cliques, but the m-clique with optimal weight (mission success) only, the algorithm is tailored and accelerated by using upper bounds to eliminate non-optimal m-cliques. The adapted branch and bound algorithm truncates branches of the search tree, improving both memory requirements and running time of the enumeration algorithm. The algorithm computes the m-clique with optimal weight, that is, a set of m mutually collision-free flight paths with maximal mission success.

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

The core of the algorithm is the recursive function OptClique which is presented below. The parameter l denotes the depth of the recursion, whereas a is the number of the start partition.

Branch and bound algorithm

Input: partition of the vertex set V into independent sets $V_1, V_2, \ldots, V_m$

Output: vertex set S_{opt} of an optimal clique

Initialization:

$E := \emptyset$, opt_weight $:= 0$
for $b = 1, 2, \ldots, m$ *do*
  $N_b := V_b$
  $U_b := \emptyset$
  $S(b) := 0$
  for all $v \in V_b$ *do*
    partition$(v) := b$

Call of the function:

$a := 1, l := 1$
OptClique(l, a)

Function:

```
OptClique(l, a)
1  for all v_i ∈ N_a do
2     new_empty_partitions := 0
3     for b = 1, 2, ..., m do
4        if new_empty_partitions ≤ 1 then
5           for all v_j ∈ N_b do
6              if not yet done then check compatibility of v_i and v_j
7                 if v_i and v_j are compatible then E := E ∪ (v_i, v_j)
8              if (v_i, v_j) ∉ E then
9                 N_b := N_b \ {v_j}
10                U_l := U_l ∪ {v_j}
11                if N_b = ∅ then
12                   new_empty_partitions := new_empty_partitions +1
13    if new_empty_partitions ≤ 1 then
14       S(a) := v_i
15       if l = m then
16          compute weight(S)
17          if weight(S) > opt_weight then
18             opt_weight := weight(S)
19             S_opt := S
20       else
21          compute upper bound up_bound for weight of maximum clique
22          if up_bound ≥ opt_weight then
23             c := index of smallest non-empty set N_b
24                OptClique(l + 1, c)
25          S(a) := 0
26       for all v_j ∈ U_l do
27          b := partition(v_j)
28          N_b := N_b ∪ {v_j}
29          U_l := U_l \ {v_j}
30 Output S_opt
```

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

The main idea of the function is to generate, in each step, a partial solution $S \subseteq V$. A partial solution is a clique consisting of less than m vertices. The size of the clique corresponds to the level of the recursion. If the recursion level l is equal to m, then the solution is complete, that is, S is an m-clique.

The task of the instructions in lines 2–12 is to update the sets of compatible vertices for a vertex v_i of the start partition N_a and to store the erased vertices. The set of compatible vertices N_b contains all vertices from the partition V_b which are connected to all vertices of the partial solution S (line 9). The set U_l contains all vertices which are erased from the sets of compatible vertices in level l of the function (line 10). Note that the set N_a becomes empty, since it does not contain vertices which are compatible with v_i. If a further set N_b becomes empty, then the number of empty sets exceeds 1 (new_empty_partitions > 1), which means that the partial solution cannot be extended.

If the partial solution is expandable (line 13), then the vertex v_i becomes a vertex of the partial solution (line 14). If the recursion level is equal to m (line 15), then an m-clique has been found. If the recursion level is smaller than m (line 20) then, under a certain *bounding condition* (see below), the function is called (line 24) and processes the next level with the extended partial solution. That is, the algorithm performs a *branching step*. The decision to select the smallest set N_b for branching (line 23) is motivated by the idea to keep the number of iterated calls of the function as small as possible.

The instructions in line 25 and in lines 27–29 reverse the decisions made in the previous step. The vertex v_i is deleted from the partial solution S (line 25), the sets of compatible vertices N_b are refilled with the erased vertices (line 28), and the set of erased vertices U_l is reduced by the same vertices (line 29).

The original algorithm for enumerating all m-cliques is accelerated and modified to a branch and bound algorithm that provides the optimal m-clique only, with the following additions:

(1) The enumeration algorithm repeatedly checks whether two vertices of the graph are connected by an edge or not (line 8). An edge exists if the flight paths associated with the vertices are compatible. The straightforward idea to construct the graph prior to starting the algorithm is very time-consuming since, for all pairs of flight paths, it must be verified whether the missiles following the paths keep the predefined safety distance. Instead, compatibility is checked during the algorithm when needed (lines 6–7). This saves a lot of effort as not all pairs have to be examined. Hence, the graph is not part of the input but is constructed during the algorithm.

(2) The enumeration algorithm provides and saves all m-cliques of the graph. The branch and bound algorithm calculates the mission success of the flight paths belonging to the actual m-clique, compares it with the best solution found so far, and overwrites the best solution if an improvement could be achieved (lines 16–19).

(3) The enumeration algorithm performs a systematic enumeration of all solutions by means of state space search. Repeated branching (line 24) generates a *search tree* with the full set of solutions at the root. The branches of the tree correspond to subsets of the solution set, while the leaves of the tree represent the individual solutions. The adapted algorithm supplements branching by a bounding technique in order to prune the search tree. By computing upper bounds for the weight of the solutions in branches of the tree, some branches of the complete search tree are cut off that cannot produce a better solution (lines 21–22). The bounding technique significantly speeds up the calculations.

The computation of upper bounds requires a closer look at the search tree. Each node of the search tree is associated with two index sets I and J, a clique S and vertex sets $N_j, j \in J$:

- root (level 0 of the tree): $I = \emptyset$ and $J = \{1, 2, \ldots, m\}$;$S = \emptyset$, vertex sets $N_j = V_j, j \in J$
- interior nodes (level k of the tree): $I \subset \{1, 2, \ldots, m\}$, $|I| = k$, and $J = \{1, 2, \ldots, m\} \backslash I$; k-clique S with vertices $v_i \in V_i$, $i \in I$ and vertex sets $N_j, j \in J$ with $N_j = \{w \in V_j \mid (v, w) \in E \text{ for all } v \in S\}$
- leaves (level m of the tree): $I = \{1, 2, \ldots, m\}$ and $J = \emptyset$;m-clique S with vertices $v_i \in V_i$, $i \in I$

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

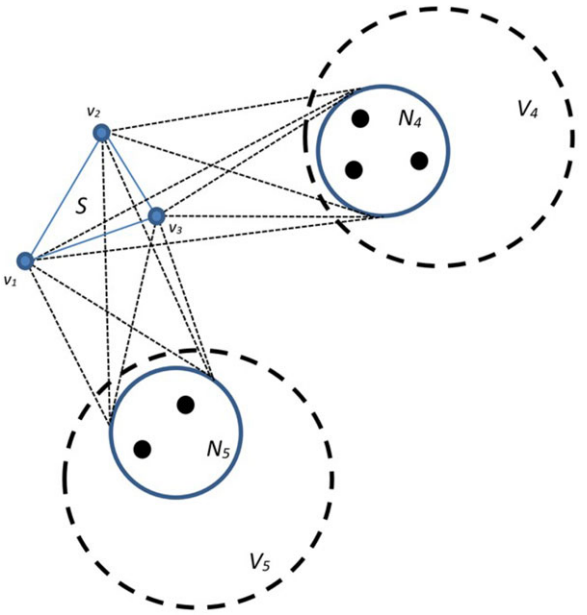

Figure 6. *Data associated with interior node of search tree (I).*

A node at level k represents a partial solution consisting of a k-clique S and vertex sets N_j that contain all vertices from V_j which are connected to all vertices from the clique S (see also the example in Fig. 6). The partial solution corresponds to a set of flight paths for k missiles which are pairwise compatible. For the remaining $m - k$ missiles, no flight paths have been selected so far. The set N_j corresponds to the set of all flight paths of missile j which are compatible with all flight paths of the partial solution.

Let p_i denote the arrival probability for the flight path represented by vertex $v_i \in S$ and let

$$p_j^* = \max\{p_j \mid v_j \in N_j\} \tag{12}$$

be the largest arrival probability for all flight paths belonging to N_j. In case of a single target

$$p_{hit}^* = 1 - \prod_{i \in I} (1 - p_i) \cdot \prod_{j \in J} (1 - p_j^*) \tag{13}$$

is an upper bound for the hit probability p_{hit} and hence for the mission success at the node (see also Section 3.3). For each missile without a selected flight path, the bound uses the maximal possible arrival probability.

For multiple targets, let $I_q \subseteq I$ contain the indices of the vertices from the clique S with flight paths assigned to target q and $N_{j,q}$ the vertices from N_j belonging to flight paths with target q (see the example in Fig. 7 for two targets). Then,

$$p_{j,q}^* = \max\{p_j \mid v_j \in N_{j,q}\} \tag{14}$$

is the largest arrival probability for all flight paths from N_j with target q and

$$p_{hit,q}^* = 1 - \prod_{i \in I_q} (1 - p_i) \cdot \prod_{j \in J} (1 - p_{j,q}^*) \tag{15}$$

is an upper bound for the hit probability of target q. The sum of the upper bounds for all t targets

$$p_{hit,1}^* + p_{hit,2}^* + \ldots + p_{hit,t}^* \tag{16}$$

is an upper bound for the mission success (see also Section 3.3).

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

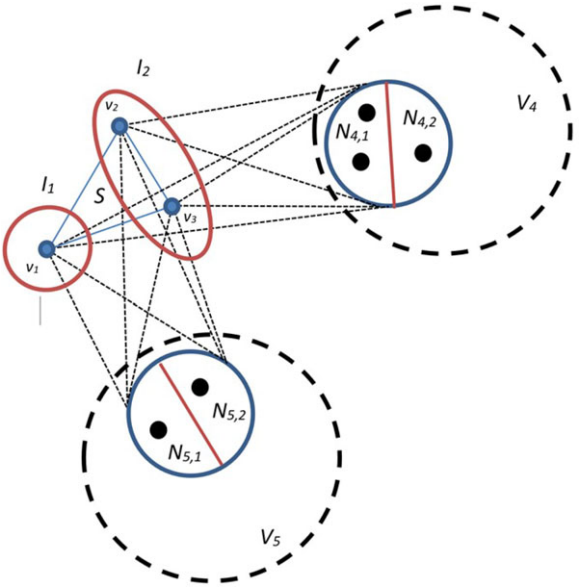

Figure 7. *Data associated with interior node of search tree (II).*

The upper bounds have been achieved by the strategy of relaxation. In a maximization problem, a relaxation is a related problem obtained by removing constraints such that the optimal value is at least equal to the optimal value of the original problem. Here, compatibility between flight paths of missiles without selected flight paths (from the set J) is not considered. Moreover, in the case of multiple targets, the assignment of these missiles to specific targets is neglected.

3.5. Iterative planning algorithm

The goal of this section is to develop an iterative algorithm for coordinated flight path planning of a fleet of missiles. The algorithm shall provide a first rough solution very quickly and shall allow successive improvements with increasing computer runtime. It shall be able to produce a high-quality solution if enough time is invested. Termination of the algorithm shall be possible at any time, albeit with a suboptimal solution.

The algorithm is composed of the previously discussed modules. It uses the technique to generate flight paths with predefined flight time and minimum risk (see Section 2.2), the construction of path compatibility graphs representing large sets of flight paths (see Section 3.3), and the branch and bound algorithm to find an m-clique with maximal weight in an m-partite graph (see Section 3.4). The algorithm takes up the idea of global optimization.

The key point is to generate several RRTs with roots in the release and target points, respectively. The trees are growing simultaneously and are connected with each other to produce flight paths of predefined lengths for the missiles to the different targets. The number of flight paths is steadily increasing. In parallel, the path compatibility graph is constructed which is expanding with the number of flight paths. In this growing graph, the m-cliques with maximal mission success are determined that include the newly inserted vertices. The procedure is embedded in an endless loop, which can be terminated according to various criteria such as maximum computation time or required mission success.

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

A detailed description of the algorithm is stated below.

Offline flight path algorithm

Input: s release points with release directions
m missiles associated to release points with predefined flight times
t target points with sectors of possible attack directions

Output: assignment of missiles to targets
attack directions of the targets
compatible flight paths for the missiles
mission success

Initialization Phase:

1 *for* $i = 1, 2, \ldots, s$ *do*
2 initialize S-tree S_i with root located at release point i and associated direction
3 *for* $j = 1, 2, \ldots, t$ *do*
4 initialize T-tree T_j with root located at target point j and associated attack sector
5 *for* $i = 1, 2, \ldots, s$ *do*
6 *for all* missiles associated to release point i *do*
7 *for* $j = 1, 2, \ldots, t$ *do*
8 generate flight paths with flight time of missile by connecting root
9 of S-tree S_i with all nodes of T-tree T_j (using Dubins paths of type CSC)
10 construct path compatibility graph associated to flight paths
11 find and save m-clique with maximal mission success in path compatibility graph

Extension Phase:

12 *while* (true) *do*
13 *for* $i = 1, 2, \ldots, s$ *do*
14 extend S-tree S_i by a node (using Dubins path of type CS)
15 *for all* missiles associated to release point i *do*
16 *for* $j = 1, 2, \ldots, t$ *do*
17 generate flight paths with flight time of missile by connecting new node
18 of S-tree S_i with all nodes of T-tree T_j (using Dubins paths of type CSC)
19 extend path compatibility graph by vertices associated to new flight paths
20 find m-clique with maximal mission success containing new vertex
21 *if* better solution has been found *then* save best solution
22 *for* $j = 1, 2, \ldots, t$ *do*
23 extend T-tree T_j by a node (using Dubins path of type CS)
24 *for* $i = 1, 2, \ldots, s$ *do*
25 *for all* missiles associated to release point i *do*
26 generate flight paths with flight time of missile by connecting new node
27 of T-tree T_j with all nodes of S-tree S_i (using Dubins paths of type CSC)
28 extend path compatibility graph by vertices associated to new flight paths
29 find m-clique with maximal mission success containing new vertex
30 *if* better solution has been found *then* save best solution
31 *if* termination criterion is fulfilled *then*
32 Stop and output current best solution

While the initialized S-trees associated with the start points (line 2) consist of the root node only, the T-trees belonging to the targets (line 4) consist of the root node and its neighbours covering the sector of possible attack directions (see Section 2.1). In the initialization phase, the algorithm generates, for each

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

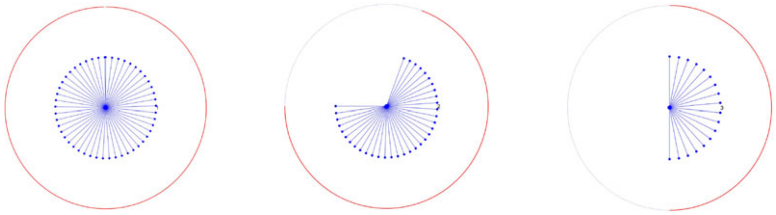

Figure 8. *Initialized T-trees with different attack sectors.*

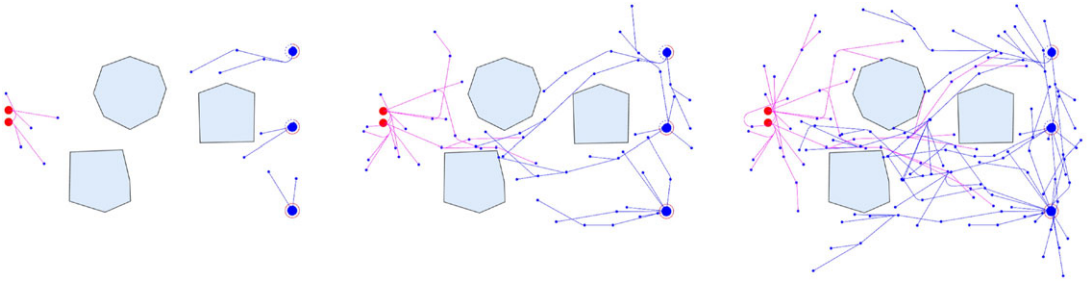

Figure 9. *Growing S-trees and T-trees.*

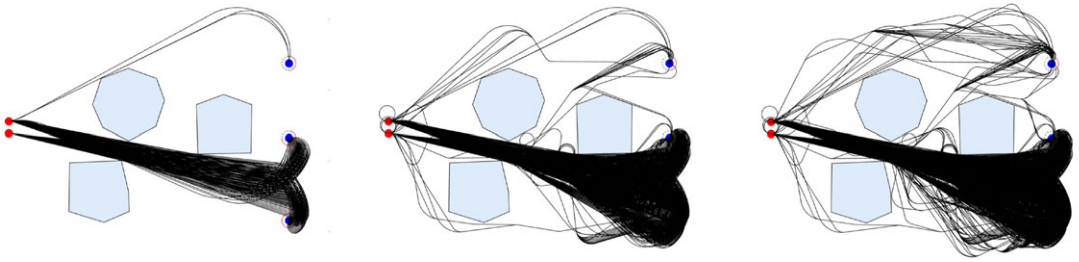

Figure 10. *Increasing number of flight paths.*

missile, flight paths to all targets (lines 8–9). The same occurs in the extension phase each time a tree is extended by a new node (lines 17–18 and 26–27). The construction of the path compatibility graph (lines 10–11) and the extension of the graph (lines 19–20 and 28–29) occurs within the clique finding function (see Section 3.4).

To illustrate how the algorithm proceeds, we consider a scenario with three no-fly areas, two release points and three targets. The fleet consists of eight missiles, where four missiles are released from each release point in a temporal distance of some seconds. Although the scenario does not contain any known threats, the missiles are exposed to potential unknown threats (see Section 2.2). Figure 8 shows the three initialized T-trees with the associated sectors of possible attack directions (drawn in red colour). The T-tree on the left comprises all directions, the tree in the middle a sector from 180° to 70°, and the tree on the right a sector from 270° to 90°.

Figure 9 shows the horizontal view of the two growing S-trees rooted in the release points of the missiles (with red colour) and the three growing T-trees rooted in the target points (with blue colour) at different stages of the algorithm. Figure 10 illustrates the increasing number of flight paths for the missiles to all targets with the predefined flight times.

Figure 11 shows the best solution at different stages of the algorithm, consisting of collision-free flight paths for the eight missiles with the current maximum mission success. As can be seen, the assignment of the missiles to the targets changes, as well as the attack directions. What the figure does not show is that, from phase to phase, the missiles are less visible from the terrain and less exposed to unknown

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

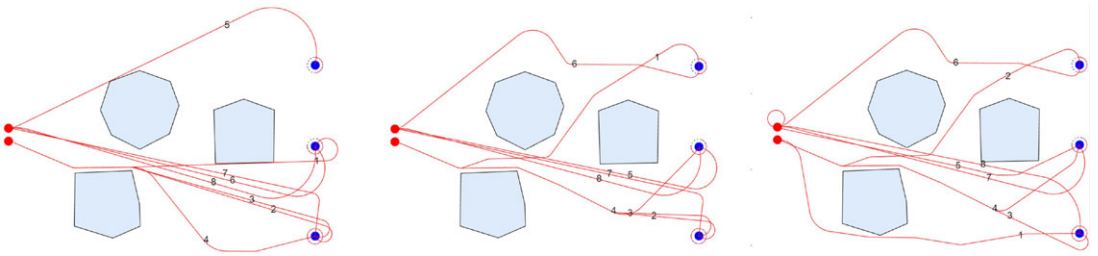

Figure 11. *Solutions with increasing mission success.*

threats. Exploiting terrain masking effects reduces the risk of the missiles and increases the mission success.

The presented algorithm generates several new flight paths in each iteration. Hence, the path compatibility graph grows constantly with the number of iterations. With long runtime of the algorithm, this could lead to memory space problems. Moreover, finding optimal cliques in an ever expanding graph is getting more and more costly.

Memory and runtime problems are resolved by discarding selected flight paths and by restricting the size of the path compatibility graph. For each missile, a long but limited list of flight paths is maintained. The compatibility graph contains a subset of the flight paths and is dynamically adapted by deleting unpromising and including new promising flight paths. The selection is depending on the risk of the flight paths, which is given in the form of the arrival probabilities. However, since an optimal set of coordinated flight paths may not contain only low-risk flight paths, some randomly selected flight paths with higher risk are also included.

4. Computational results

4.1. Test cases for offline planning

This section presents computational results for a number of selected scenarios. The algorithms have been implemented and tested in Matlab R2022a on a standard PC running Windows 10 with Intel Core i7-6600U 2.6 GHz CPU and 16 GB RAM.

The fleet is assumed to consist of identical missiles. The nominal velocity of the missiles is 100 m/s, and the minimum turning radius is 1000 m. The angles of climb and descent are limited to 4° and 5°, respectively. The safety distance between the missiles is set at 500 m. The safety margin for flight above terrain is 50 m. Each scenario contains no-fly areas and threat areas. No-fly areas have polygonal shape (concave or non-concave), and threat areas are plotted as circular areas. Release points of the missiles are drawn as red dots and targets as blue dots.

The first test case contains two no-fly areas and three threat areas in a scenario of size 30 km × 15 km, see Fig. 12. Two targets located within a threat area are engaged by four missiles, with two missiles released from each of two platforms. The release is in direction east. One of the targets has a restricted attack sector, indicated by the red circle segment, and can only be approached from the west. As in all test cases, the missiles must reach the targets simultaneously. The flight times of the missiles launched first from each start point are 340 s. The other missiles are released at a time interval of 10 s with flight times of 330 s.

Figure 13 shows the results of offline planning for computer runtime increasing from 10 s, 20 s and 30 s to 50 s. The flight paths of the four missiles are plotted on radar visibility maps which show, for each point within the range of the radar, the minimal altitude above ground with a line of sight to the radar. The parts of the flight paths marked in green are visible from the radar sites. The other parts are hidden by the terrain. The test case demonstrates that the assignment of missiles to targets as well as

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

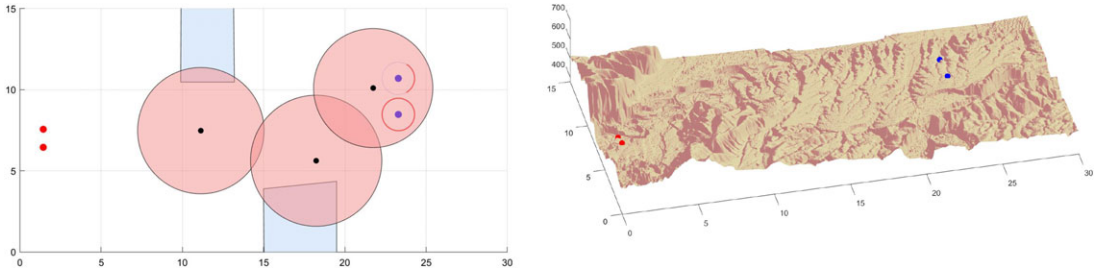

Figure 12. *Scenario with two release platforms and two targets.*

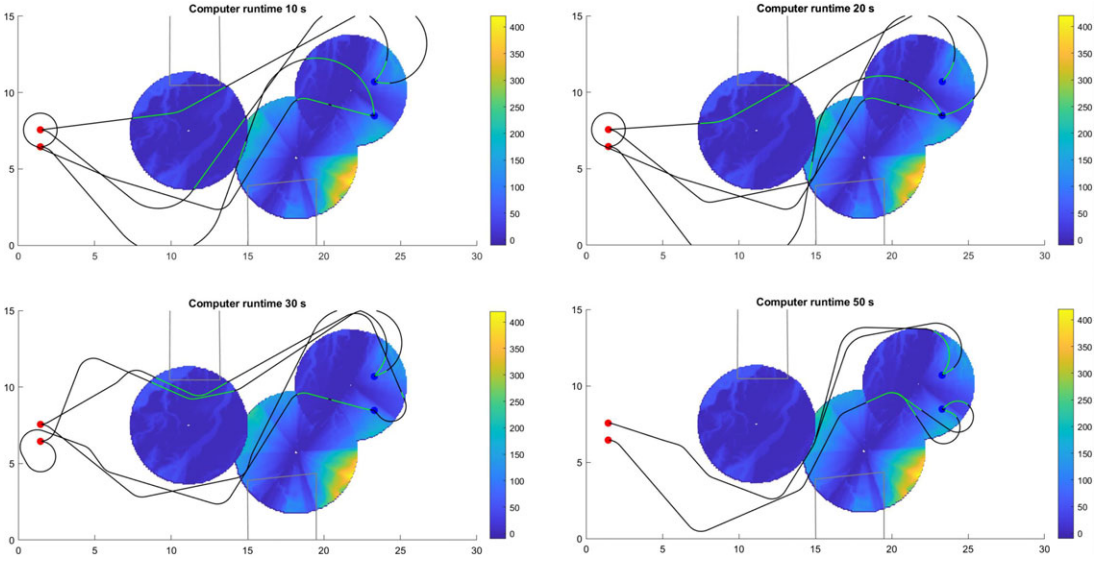

Figure 13. *Solutions with increasing mission success.*

the attack directions are changing during the iterative optimization process. The mission success, that is, the expected number of hit targets, increases from 0.41, 0.78 and 0.94 to 0.98.

In the next test case, five targets are engaged by a fleet consisting of ten missiles. The scenario is of size 40 km × 35 km with two no-fly areas and two threat areas (see Fig. 14). The missiles are released from five platforms, with each platform including two missiles. The flight times are 350 s and 320 s for the missiles of each platform. The release of the missiles takes place in direction east. The attack directions of the targets are restricted to sectors. The figure shows the results of the offline planning algorithm for computation times of 20 s, 120 s and 240 s. The mission success is increasing from 3.14 to 3.32 and 3.68.

4.2. Test cases for online planning

The next scenario of size 50 km × 35 km contains two threat areas located between two no-fly areas (see Fig. 15). Four missiles are released from the same platform in direction east to engage a common target. The flight time of the last launched missile is 500 s. The other missiles are launched beforehand at intervals of 5 s with flight times 505 s, 510 s and 515 s. The target can be attacked from all directions. The figure shows the results of two runs of the online planning algorithm with a computation time of 1 s each. Note that due to the random nature of the algorithm, different runs will provide different flight paths. Accordingly, the mission success of 0.93 and 0.94 differs slightly.

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

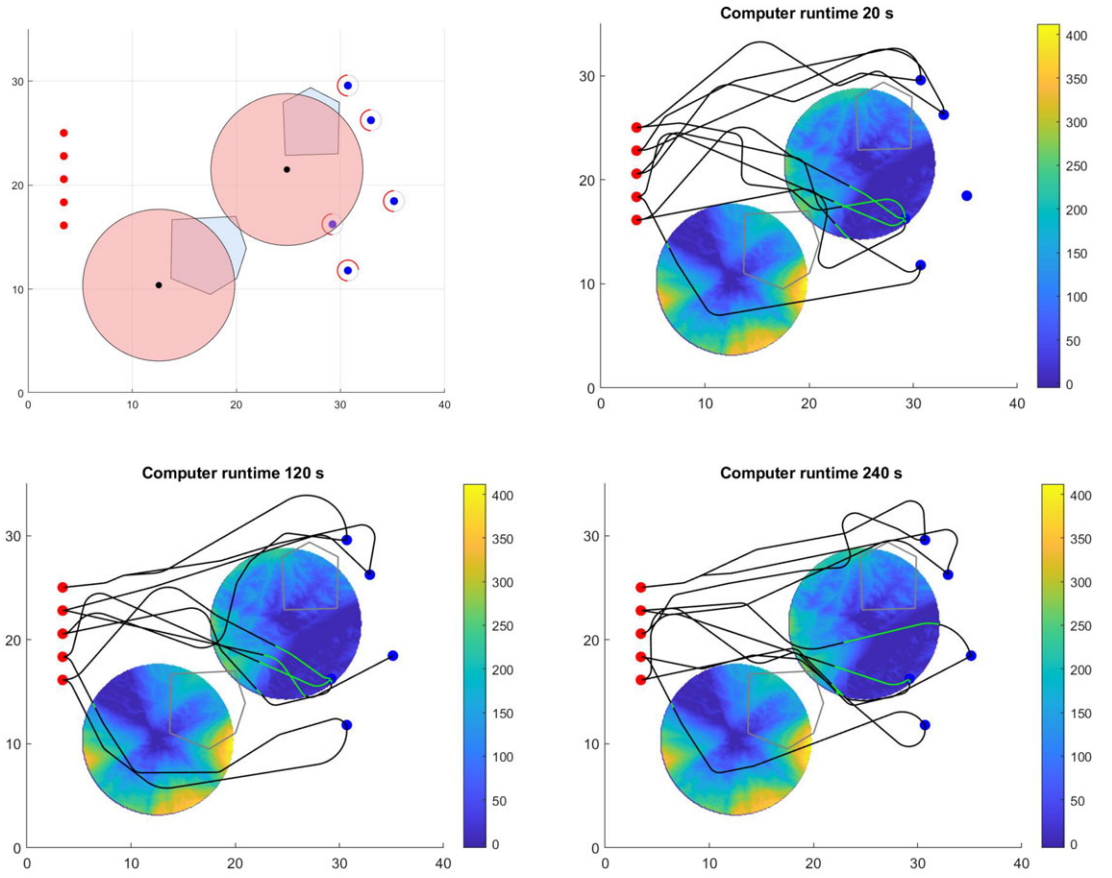

Figure 14. *Solutions in scenario with five release platforms and five targets.*

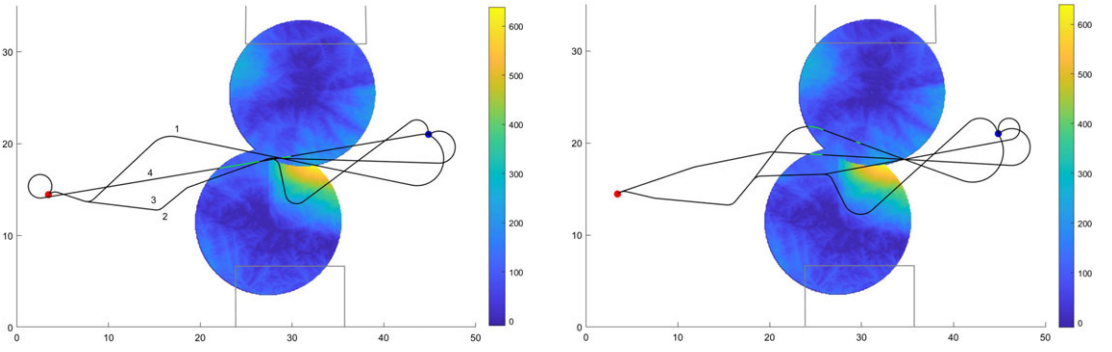

Figure 15. *Solutions in scenario with single release platform and single target.*

The plots in Fig. 16 show the mutual distance between the missiles along the flight paths for the first run of the algorithm. The vertical flight paths over terrain with the predefined safety margin of 50 m above ground, exemplary for the first two missiles, are presented in Fig. 17. In this test case, the missiles are launched from land-based platforms and approach the targets in a dive attack.

Performing 50 runs of the algorithm with the same computer runtime of 1 s yields a mission success in the range of 0.88 to 0.97 with an empirical standard deviation of 0.028. While the scatter in online

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

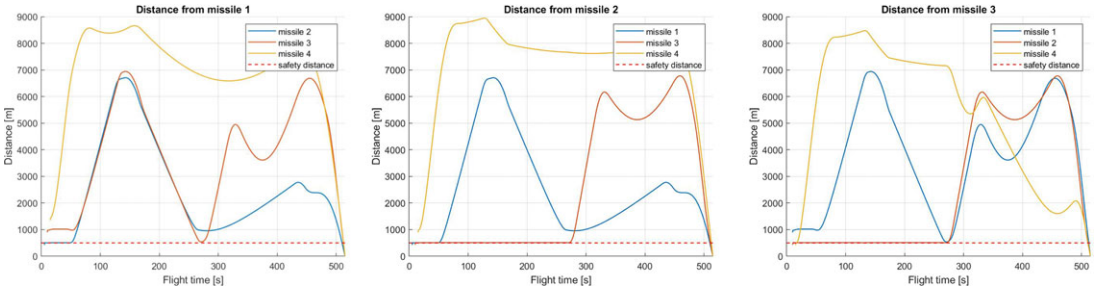

Figure 16. *Distance between missiles.*

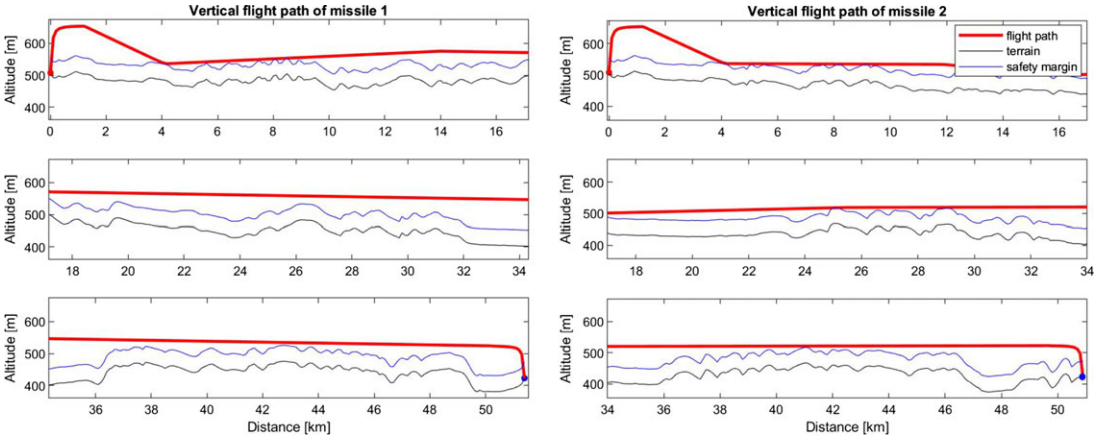

Figure 17. *Vertical flight paths over terrain.*

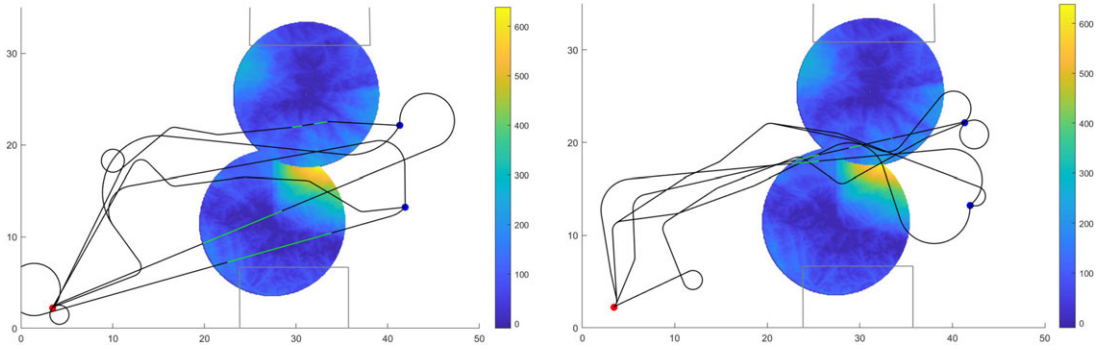

Figure 18. *Solutions for online and offline planning algorithms.*

planning is generally significant, computational experiments indicate that the results provided by the offline planning algorithm are stable and the scatter is negligible.

The purpose of the last test case is to compare the results of the online and offline planning algorithms. The scenario is the same as before, containing two no-fly areas and two threat areas (see Fig. 18). Now six missiles are released from the same platform to engage two targets. The release direction is north east. The approach directions of the targets are unspecified. The flight time of the first launched missile is 600 s. The five remaining missiles are released in a salvo one after the other, with an interval of 5 s, with correspondingly shortened flight times.

https://doi.org/10.1017/S0263574722001886 Published online by Cambridge University Press

The figure shows on the left side the flight paths obtained by the online planning algorithm using sequential optimization. The assignment of missiles to targets is predefined and part of the input of the algorithm. With a computation time of 2 s for all six flight paths, the mission success is 1.46. The flight paths on the right side result from offline planning with global optimization. The algorithm also optimizes the assignment of missiles to targets. The flight paths are less exposed to threats, increasing the mission success to 1.77. However, this comes at the cost of a much higher computation time of 60 s.

In online planning, the calculation time for the individual flight paths is usually set in advance. Different calculation times clearly lead to different quality of results. In the previous test case, reducing the total calculation time to 1 s deteriorates the mission success to 1.02. On the other hand, in response to unexpected events, it is usually not necessary to re-plan all the flight paths of the fleet since only a few are affected. Reasonable results for individual flight paths can be obtained within fractions of a second, high-quality results within less than a second. Finally, it should be noted that the runtime of the algorithms strongly depends on the complexity of the scenario and the level of detail of the threat model. Simplifying the sophisticated threat model used by our algorithms can significantly reduce computation time.

5. Conclusions

UAVs play an important role in achieving information superiority, conducting precision strikes, and supporting rapid combat operations. In particular, the deployment of fleets or swarms of UAVs or missiles can lead to superiority in warfare. Since combat operations usually take place in highly protected areas, large numbers of aircraft are used to overwhelm hostile air defence and to exploit saturation effects. Major aspects of planning a mission of a fleet of aircraft against multiple targets include target assignment and flight path generation.

The objective of this work was to develop algorithms for offline and online flight path planning of a fleet of missiles against stationary targets in scenarios with no-fly areas and threats. The flight times of the missiles are determined by a high-level planner who creates an overall mission plan. Mission success is maximized by finding an appropriate assignment of missiles to targets and generating flight paths with low risk. While offline planning uses a global optimization approach to determine an optimal selection of compatible flight paths from a huge set of possible paths, online planning is based on local optimization that generates flight paths sequentially.

Offline planning is realized by an iterative algorithm capable of constantly improving mission success, with short lead times of no more than a few minutes. Online planning allows flight paths to be re-planned within fractions of a second up to very few seconds, with the quality of the solutions depending on the complexity of the scenario and the time available. The capability of the two algorithms was demonstrated using typical test cases.

Future work will focus on coordinated flight path planning in conjunction with aircraft communications. In military operations, aircraft often need to communicate with each other or with other participants in a joint mission. For instance, information perceived by sensors of an aircraft, such as obstacles or emerging threats, must be shared with other members of the fleet. Another example is the synchronization of the tasks of the individual aircraft. However, communications and power resources are often limited, and communications between different aircraft can be unreliable. This imposes additional constraints that significantly affect flight path planning.

Author contributions. The study was conducted and the article was written by Luitpold Babel.

Financial support. This research received no specific grant from any funding agency, commercial or not-for-profit sectors.

Conflicts of interest. The author declares no conflicts of interest exist.

Ethical approval. Not applicable.

References

[1] G. Skorobogatov, C. Barrado and E. Salami, "Multiple UAV systems: A survey," *Unman. Syst*. **8**(2), 149–169 (2020).
[2] S. Chung, A. Paranjape, P. Dames, S. Shen and V. Kumar, "A survey on aerial swarm robotics," *IEEE Trans. Robot.* **34**(4), 837–855 (2018).
[3] S. Aggarwal and N. Kumar, "Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges," *Comput. Commun.* **149**, 270–299 (2020).
[4] A. Vashisth, R. S. Batth and R. Ward, "Existing Path Planning Techniques in Unmanned Aerial Vehicles (UAVs): A Systematic Review," **In:** *International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)* (2021) pp. 366–372.
[5] H. Zhang, B. Xin, L. Dou, J. Chen and K. Hirota, "A review of cooperative path planning of an unmanned aerial vehicle group," *Front. Inform. Technol. Electron. Eng*. **21**(12), 1671–1694 (2020).
[6] Y. Liu and R. Bucknall, "A survey of formation control and motion planning of multiple unmanned vehicles," *Robotica* **36**(7), 1019–1047 (2018).
[7] A. Puente-Castro, D. Rivero, A. Pazos and E. Fernandez-Blanco, "A review of artificial intelligence applied to path planning in UAV swarms," *Neural Comput. Appl*. **34**(1), 153–170 (2022).
[8] L. Babel, "Coordinated target assignment and UAV path planning with timing constraints," *J. Intell. Robot. Syst*. **94**(3-4), 857–869 (2019).
[9] L. Babel, "Online flight path planning with flight time constraints for fixed-wing UAVs in dynamic environments," *Int. J. Intell. Unman. Syst*. **10**(4), 416–433 (2022).
[10] S. M. LaValle, Rapidly-exploring random trees: A new tool for path planning, Technical Report, Computer Science Department, Iowa State University, TR 98-11 (1998).
[11] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.* **30**(7), 846–894 (2011).
[12] J. J. Kuffner and S. M. LaValle, "RRT-connect: An Efficient Approach to Single-Query Path Planning," **In:** *Proceedings 2000 IEEE International Conference on Robotics and Automation* (2000).
[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, New York, 1979).
[14] Q. Wu and J. K. Hao, "A review on algorithms for maximum clique problems," *Eur. J. Oper. Res.* **242**(3), 693–709 (2015).
[15] T. Grünert, S. Irnich, H. J. Zimmermann, M. Schneider and B. Wulfhorst, "Cliques in k-partite graphs and their application in textile engineering," *Comput. Oper. Res*. **29**(1), 13–31 (2002).

Cite this article: L. Babel (2023). "Coordinated flight path planning for a fleet of missiles in high-risk areas", Robotica **41**, 1568–1589. https://doi.org/10.1017/S0263574722001886