Optimization of the Three-Dimensional Search allocation Game for UAVs

Florian Delavernhe *Université Bourgogne Europe, DRIVE UR 1859, 58000 Nevers, France*
florian.delavernhe@u-bourgogne.fr

***Abstract*— This paper addresses the optimization of evasive target search using unmanned aerial vehicles (UAVs) through a search allocation game framework. We introduce a novel three-dimensional UAV search model in which the UAV altitude is part of the decision-making and explicitly affects detection range, search efficiency, and visibility to the target, thereby influencing both search performance and target behavior. Unlike existing search allocation models, which often rely on simplified or unrealistic target assumptions, the proposed formulation captures complex target reactions induced by the UAVs visibility. This leads to a larger-scale and more realistic optimization problem that better reflects practical UAV-based search operations. We develop a mathematical formulation of the problem and propose a metaheuristic solution based on simulated annealing. Computational experiments demonstrate that the proposed approach can solve realistically sized instances within operationally relevant computation times, highlighting its potential for real-world UAV search applications.**

I. Introduction

The problem of optimizing the search for moving a target is an old problem [1], forming the basis of the field of search theory. It has been addressed in a wide range of applications [2], such as: (i) search-and-rescue missions for lost persons; (ii) early detection of forest fires; (iii) locating sunken ships; and (iv) the pursuit of escaping criminals or military enemies (such as submarines) actively evading searchers. The use of UAVs, equipped with cameras, for target search has become increasingly prominent in recent years, with many applications now considering fleets of UAVs as basic searchers. For instance in search-and-rescue missions [3], [4] or in search-and-capture (or search-and-destroy) missions [2], [5]. UAVs enable rapid deployment in infrastructure-less and hazardous areas, while their flying capacity can be fully exploited to cover wide search zones. However, the optimal planning of the UAVs' search is a complex task.

Because the search is imperfect, meaning that UAVs may fly over the target without guaranteeing its detection (e.g., due to obstacles between the camera and the target, or failures in image-based detection), most of the literature on target search optimization addresses the search allocation problem (SAP). SAP formulates the UAVs' search planning as the allocation of search resources, typically searching time, so as to maximize the detection probability [6], [7], [8]. In this framework, the search map is discretized into a 2D grid of cells, and an optimal distribution of search effort is sought across these cells depending on the moving target model and terrain characteristics. The longer a UAV remains in a given cell hosting the target, the higher the detection probability becomes. This may be due, for example, to capturing more images and allowing the UAV to more extensively explore the cell, thereby increasing detection chances. However, SAP generally assumes predictable target behavior, modeled as a set of possible trajectories with associated probabilities that are pre-computed.

A subset of the literature instead considers an evasive target, i.e., one that reacts to search efforts to avoid detection. In this case, SAP is combined with game theory to form the search allocation game (SAG) [9]. SAG also models target behavior using trajectories but assumes the target selects its trajectory in response to the search performed. The main weakness of SAG is that it assumes a perfect information game: the target is assumed to know all relevant details (e.g., number and characteristics of the UAVs), to observe every search effort, and to react optimally at all times. In the UAV context, however, this does not capture reality: a terrestrial target may initially lack information about the searchers (e.g., it may not even be aware of being pursued) but could gather new information later, for instance by seeing or hearing a UAV from afar, well before it is directly overhead and able to detect the target. This observation highlights the importance of UAV altitude as an additional decision variable which directly affects both search efficiency and the visibility of UAVs to the target. For example, flying at a higher altitude allows the UAV to cover a wider portion of the searched map, but with reduced efficiency (lower detection probability) and increased visibility making the UAV noticeable from farther away and enabling better evasion from the target.

In this work, we propose a new formulation of the search allocation game, explicitly tailored to UAV applications, with the introduction of 3D movements and an adaptive evasion. Our modeling contributions include the following novelties: (i) UAVs altitude is integrated into the decision process; (ii) higher altitudes allow drones to cover more cells, possibly with overlapping fields of view, but at reduced detection efficiency; and (iii) the target may follow either an SAP-type behavior (trajectory probabilities) or a SAG-type behavior (adversarial path selection), depending on its ability to observe the UAVs, which is determined by both UAVs altitude and position relatively to the target's position. Alongside this more complex yet realistic model, we propose a local search method, a simulated annealing, to produce high-quality solutions.

II. Related works

When considering an imperfect search, which is typical of UAV searches that rely on distant image capture, the search planning literature has focused on the optimal distribution of search resources, determining where and for how long to search for the target. As mentioned earlier, the operations research community has approached this through two popular target behavior modelizations: the search allocation problem (SAP) and the search allocation game (SAG). These problems differ in how they model target behavior.

The SAP relies on a classical model assuming that the target follows a predictable behavior computed in advance (e.g., precomputed movement probabilities or stationary targets). Such a model has been in place since the genesis of search theory [1] and is widely used in the literature such as for [4], [6], [7], [10], [11], [12], despite some of them implicitly focusing on a target that should be reactive, for instance, [7] with a camouflaging target. These many works widely study the modelization of realistic search plan but often overlook UAVs' constraints. For instance, in [6], [13], the problem addressed has each searcher assigned to a sub-zone for each time period, restricting the allocation of search resources to the assigned sub-zone. Such a formulation is unfit for UAV searchers, whose flying capacity makes them more flexible than being confined to arbitrary sub-zones. In contrast, [7], [10], [11] propose more generalized models where searchers' movements consume search resources. Typically, a time budget serves as the main constraint (e.g., a 24-hour search limit), and allocation must account for the time spent traveling between search locations. [10] models a continuous SAP, while [7] uses a discretized version allowing for linearization of the objective function. [11] considers multiple static targets for rescue missions. These models improve realism toward UAV applications but remain limited to non-reactive targets and do not incorporate 3D aspects, and more particularly its underlying problem: overlapping search positions. Specifically, in previous works, whatever the target's location, only one searcher position may detect it, which is not realistic for an UAV application as the sight range of an UAV depends on its altitude.

For the modeling of evading targets, i.e., those that respond to search efforts, the SAP extends to the SAG [9], which applies game theory to search allocation. In this formulation, the searcher and the target are opposed in a two-person zero-sum game: the target moves to minimize detection probability, while the searcher maximizes it. The optimal search is a solution of a max-min problem which maximizes detection probability in the worst-case scenario [9]. For instance of SAG works, [14] addresses energy-constrained targets, adding restrictions on target movement, while [15] studies the convergence of the precision of the discretization grid toward a continuous formulation. More recently, [16] examines multiple non-collaborating evading targets, and [8] proposes a multi-objective approach balancing short-term and long-term search efficiency. In this literature, the focus is set on target modelization rather than the searcher modelization, thus making it difficult to consider such works realistic for UAV searchers. In addition, these works adopt the standard SAG assumption of perfect information, with three key premises: (i) the target has complete knowledge of the searchers, including resources and detection capabilities; (ii) the target fully observes all searcher actions; and (iii) the target always reacts optimally. These assumptions make SAG further unrealistic for real UAV applications. Some works have attempted to address imperfect information, e.g., [17] and [18]. Yet, no work has relaxed all three assumptions simultaneously.

A hybrid approach for target modelization is proposed in [5], which considers an evading target model combining features of both SAP and SAG. Specifically, the authors address the search for an escaping criminal in a dense forest by a collaborative team of UAVs and humans. The target's movement is guided by precomputed probabilities but the presence probabilities are updated dynamically: searching a cell decreases the likelihood of the target being there in subsequent periods. Unlike SAP, the target reacts to the searchers but, unlike SAG, its evasion follows a human-specified strategy rather than purely adversarial optimization. However, it still assumes that the target has perfect and instantaneous knowledge of all searched cells, which still fails to capture the effect of an UAV altitude on its visibility and efficiency.

In summary, the literature has examined the UAV search allocation problem from two main perspectives: SAP and SAG. However, neither framework adequately addresses the case of a 3D search for an evasive target, which is a natural setting for UAV applications. SAP does not capture the full complexity of 3D search, particularly the overlapping fields of view, and it has been limited to non-reactive targets. Conversely, SAG accounts for evasive target behavior but relies on simplified search models that also overlook the 3D dimension and its impact on the searchers' visibility.

The present work aims to incorporate the three-dimensional nature of UAV searches into the search allocation literature, more specifically, with a complex target behavior. Our contributions are threefold: (i) a 3D decision framework that accounts for UAV altitude into the decision process, allowing higher flight position to cover wider zones at reduced efficiency, which implies the integration of overlapping region in the search allocation literature; (ii) a generalized target behavior model in which the target alternates between SAP- and SAG-like behaviors depending on its perception of search activities, more specifically, on the flight altitude, thereby modeling adaptive evasion that addresses the imperfect information game; and (iii) a simulated annealing method that produces efficient solutions to this complex problem.

III. 3D-Search allocation game

In the present section, we introduce the 3D search resource allocation game. It enhances both the search model (with 3D search and overlapping fields of view) and the target model (mixing SAG and SAP).

A. Time-space discretization

We consider the search for a single mobile target within a bounded search area and a finite time horizon. First, the area is discretized into a set $\mathcal{C}$ of disjoint cells where the target may be located. Each cell is spatially homogeneous, e.g., we have forest cells, mountain cells, city cells, etc. Second, the time horizon is discretized into a set $\mathcal{T}$ of n time periods. We assume that the target is static within each time period and moves only between periods. In other words, during period $t \in \mathcal{T}$, the target occupies exactly one cell $c \in \mathcal{C}$ and then moves to an adjacent cell for period $t+1$. The set of adjacent cells to c is denoted $\mathcal{C}(c)$, and we consider c to be adjacent to itself.

The search performed consists of distributing search resources across the cells of $\mathcal{C}$ for each time period in $\mathcal{T}$. The total search resource available at period $t \in \mathcal{T}$, is denoted $\Phi(t)$. The search budget can represent a search team consisting of a single UAV or multiple UAVs. Typically, a 24-hour search is discretized into 24 periods, with a search budget of one hour per period for a single UAV, or x hours per period if x UAVs are deployed. Unlike previous works, we introduce a third decision dimension: the flight altitudes of the UAVs. Specifically, the flight levels are discretized into a set $\mathcal{L}$. For each cell $c \in \mathcal{C}$, a searcher may fly at altitude $l \in \mathcal{L}$. This is illustrated in Figure 1, where three altitude levels over one cell are shown, with the intensity of the green color indicating the efficiency of detection for each UAV altitude. Higher altitude levels allow the UAV to cover more cells simultaneously but with reduced detection efficiency, as it will be discussed later. Note that, in a single time period, it is allowed to search at different altitudes in a same cell.

Apart from the overall capacity constraint $\Phi(t), \forall t \in T$, we do not impose any additional restrictions on the distribution of search resources, unlike some existing studies in the SAP literature (see Section II, notably with searchers' movements). This choice pertains from the fact that the present work is intended as a first step toward 3D modeling, with a primary focus on overlapping fields of view and adaptive target evasion. More complex models can be extended from the proposed modelization, but it is out of the scope of the present work.

B. Target model

To extend existing target modelizations, we consider a hybrid behavior that combines elements of both SAP and SAG. Specifically, if the target cannot see a searcher, it follows predefined movement probabilities to adjacent cells, as in SAP. However, if the target does see a searcher, it adopts a worst-case strategy, as in SAG. This yields a more nuanced behavior: unlike SAP, the target can react to ongoing search efforts, but unlike SAG, it does not possess perfect information throughout the entire time horizon, only partial information based on visible searchers.

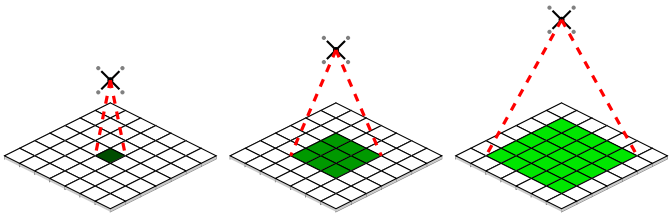
We assume that the target adopts a SAG-like behavior, i.e., a worst-case strategy for its next movement, whenever it lies within a searcher's field of view. In other words, the target's sight range is considered symmetric to that of the searchers. However, since the target is terrestrial and may benefit from cover (e.g., forests), unlike the flying UAVs, we assume that the target detects the UAVs perfectly, while the UAVs retain only imperfect detection of the target. For example, in Figure 1, whenever the target is located in a green-colored cell, it reacts optimally by selecting its next movement so as to minimize the detection probability over the time horizon, in line with the SAG literature. Note that, without loss of generality, our problem could be generalized to asymmetric sight range with introduction of additional notations which we avoid for the sake of clarity. A target in cell $c \in \mathcal{C}$ always moves to an adjacent cell in $\mathcal{C}(c)$ when following a SAG-like behavior.

Otherwise, if the searchers are out of sight, the target moves according to known probabilities $\pi(c, c'), \forall (c, c') \in \mathcal{C}^2$, where $\pi(c, c')$ denotes the probability of moving from c to c'. Finally, we consider as known the initial distribution $\eta(c), \forall c \in \mathcal{C}$, which represents the probability that the target starts the time horizon in cell c. With these elements, and for any given search plan, the probability of the target's presence in each cell and each time period can be computed.

Note that we adopt the Markovian target movements rather than trajectory-based movements, which also exist in the literature [8], [14], since evading targets lead to an exponential number of trajectories which is impractical.

C. Search model

Let us recall that we consider a distribution of the search resources for a team of UAVs, modeled as the sum of the search efforts of the UAVs in each cell and at each flight altitude. We consider an imperfect search: the presence of the target in the field of view of an UAV, i.e., in a covered cell, does not guarantee detection but only yields a detection probability. The longer an UAV searches, the higher the detection probability becomes. Accordingly, the decision variables, representing the search to be performed, are defined as $\varphi_{l,c,t} \in \mathbb{R}^+, \forall (l, c, t) \in \mathcal{L} \times \mathcal{C} \times \mathcal{T}$. The variable $\varphi_{l,c,t}$ represents the amount of resources (e.g., sum of time spent by the UAVs) allocated to cell c at flying altitude l during period t. The effect of searching in cell $c \in \mathcal{C}$, at altitude $l \in \mathcal{L}$, on another cell $c' \in \mathcal{C}$ depends both on the amount of resources allocated (i.e., $\varphi_{l,c,t}$) and on the efficiency of that search position on c'. The efficiency is modeled by a coefficient $\alpha(l, c, c') \in \mathbb{R}^+$, which equals 0 if c' is outside an UAV's field of view at position (l, c), and increases with the visibility of c' from position (l, c). The overlapping fields of view are modelized by these $\alpha(l, c, c')$. For instance, two positions, (l, c) and (l', c'), are overlapping on cell c'' if $\alpha(l, c, c'') > 0$ and $\alpha(l', c', c'') > 0$. Note that, without loss of generality, a search team composed of heterogeneous UAVs, equipped with different detecting capacities (e.g., different types of cameras) allowing for different efficiencies, could be considered by introducing additional indices for efficiency and resource budgets. However, this does not change the problem being addressed and only adds more notation.



(a) Altitude 1 (b) Altitude 2 (c) Altitude 3

Fig. 1: Illustration of the impact of different altitude levels on the UAV's field of view. Darker green corresponds to better efficiency.

If the target is located in cell $c \in \mathcal{C}$ at period $t \in T$, the detection probability, called P, is:

$$P(c,t) = 1 - e^{-\sum\limits_{\forall (l,c') \in \mathcal{L} \times \mathcal{C}} \alpha(l,c',c)\varphi_{l,c',t}} \quad (1)$$

with the exponent of the exponential being the sum of the amount of time spent on each position times the efficiency on c. Equation 1 defines the conditional detection probability, i.e., the probability of detecting the target in cell c, at period t, given that the target is actually located in c. The overall detection probability is then computed by combining this conditional probability with the target's presence probability, which itself depends on the target's non-detection during previous periods. This is because the mission terminates once the target is detected, hence there is no probability of detecting it in later periods if already detected. The complete objective function is modeled through recursive constraints in our mathematical formulation presented in Section III-D. For a more detailed description of this classical formulation, we refer the reader to previous works in the literature (e.g., [8], [19]) since we do not introduce any novelty in this aspect, other than doing a sum of decision variables in equation (1).

It is important to emphasize that the objective function is non-linear (as in SAP, but unlike SAG where it can be linearized without loss of generality). For better clarity in the remainder of this work, we instead use the complementary formulation, namely the probability of non-detection, which we aim to minimize.

D. The general mathematical model

In this section, we introduce the general mathematical model for the 3D search allocation game. The model uses the following types of decision variables:

- $\varphi_{l,c,t} \in \mathbb{R}^+, \forall (l,c,t) \in \mathcal{L} \times \mathcal{C} \times \mathcal{T}$: the amount of search resources allocated by the searchers at altitude level l, in cell c, during time period t.
- $\delta_{c,t} \in [0,1], \forall (c,t) \in \mathcal{C} \times \mathcal{T}$: the conditional non-detection probability from period t until the end of the time horizon, i.e., probability to detect the target during window $[t,n]$ assuming it is in cell c at period t.
- $x_{c,t} \in \{0,1\}, \forall (c,t) \in \mathcal{C} \times \mathcal{T}$: a binary indicator equal to 0 if the target in cell c at period t is able to see a searcher, and 1 otherwise. It represents the behavior of the target between SAG (0) and SAP (1). It is used to nullify irrelevant constraints when computing the conditional non-detection probabilities.

The equations from (3) to (10) are the objective function and the constraints:

- The objective function is the equation (2), i.e., the sum over all cells of the product between the probability that the target starts in each cell and the conditional non-detection probability over the time horizon. Note that only the $\delta_{c,1}, \forall c \in \mathcal{C}$ appear in the objective function, as they represent conditional non-detection probabilities over the entire time horizon $[1, |\mathcal{T}|]$. The remaining variables $\delta_{c,t}, \forall (c,t) \in \mathcal{C} \times (\mathcal{T}-1)$ are used solely to recursively compute their values.
- Constraints (3) are used to compute the conditional non-detection probability when the target cannot detect a searcher and thus follows a predicted movement behavior with known probabilities to move to neighboring cells. Note that there is no detection probability in that case since we assume symmetric sight ranges (a searcher can only detect the target if the target sees the searcher, as discussed earlier).
- Constraints (4) compute the conditional non-detection probability when the target is able to detect a searcher, adopting a SAG-like behavior [20]. In this case, the target's next move is in a way that minimizes the detection probability over the remaining time horizon.
- Constraints (5) compute the conditional non-detection probability for the last time period. Since the target does not move afterward, there is no distinction between detecting or not detecting a searcher.
- Constraints (6) ensure that the total allocation of search resources in each period does not exceed the available resources.
- Constraints (7) define the values of $x_{c,t}$ based on whether the target can see a searcher in cell c at time t. In our case, we assume a symmetric detection range, meaning that if the target is located in a cell covered by the searcher, the target is also able to see the searcher. Without loss of generality, the model can be slightly modified to handle a more generalized case with an asymmetric detection range, by simply adding the relevant φ terms in the numerator of the fraction of these constraints and by modifying the constraints (3) with the addition of an exponential function as in the constraints (4) (the very same exponential function also added as a product of the sum). We chose not to consider this more general case here, for the sake of clarity, as it would require introducing additional notation without bringing substantial novelty.

IV. Solving Method

A rapid inspection of the proposed Model 1 clearly indicates that exact solvers are impractical. More precisely, the problem is a mixed-integer non-linear program with a large number of constraints (up to $O(|\mathcal{C}|^2 \times |\mathcal{T}|)$ for a target with unrestrained movements) and a high number

$$\text{Minimize} \sum_{\forall c \in \mathcal{C}} \delta_{c,1}\eta(c) \tag{2}$$

$$\delta_{c,t} \geq -x_{c,t} + \sum_{\forall c' \in \mathcal{C}(c)} \pi(c,c')\delta_{c',t+1} \qquad \forall (c,t) \in \mathcal{C} \times \mathcal{T} - \{n\} \tag{3}$$

$$\delta_{c,t} \geq -(1-x_{c,t}) + exp\left(-\sum_{\forall (l,c'') \in \mathcal{L} \times \mathcal{C}} \alpha(l,c'',c)\varphi_{l,c'',t}\right)\delta_{c',t+1} \quad \forall (c,c',t) \in \mathcal{C} \times \mathcal{C}(c) \times \mathcal{T} - \{n\} \tag{4}$$

$$\delta_{c,n} \geq exp(-\sum_{\forall (l,c') \in \mathcal{L} \times \mathcal{C}} \alpha(l,c',c)\varphi_{l,c',n}) \qquad \forall c \in \mathcal{C} \tag{5}$$

$$\sum_{\forall c \in \mathcal{C}} \varphi_{c,t} \leq \Phi(t) \qquad \forall t \in \mathcal{T} \tag{6}$$

$$x_{c,t} \geq \frac{\sum_{\forall (l,c') \in \mathcal{L} \times \mathcal{C} | \alpha(l,c',c) > 0} \varphi_{l,c',t}}{\Phi(t)} \qquad \forall (c,t) \in \mathcal{C} \times \mathcal{T} - \{n\} \tag{7}$$

$$\varphi_{l,c,t} \in \mathbb{R}^+ \qquad \forall (l,c,t) \in \mathcal{L} \times \mathcal{C} \times \mathcal{T} \tag{8}$$

$$\delta_{c,t} \in \mathbb{R}^+ \qquad \forall (c,t) \in \mathcal{C} \times \mathcal{T} \tag{9}$$

$$x_{c,t} \in \{0,1\} \qquad \forall (c,t) \in \mathcal{C} \times \mathcal{T} - \{n\} \tag{10}$$

Fig. 1: Mathematical Model for the 3D Search Allocation Problem with a Target Exhibiting Mixed SAP–SAG Behavior

of decision variables ($O(|\mathcal{L}| \times |\mathcal{C}| \times |\mathcal{T}|)$ variables). We propose a metaheuristic approach to solve the problem. Our objective is not to develop the best possible algorithm, which would require a dedicated and extensive study of its own. Rather, the proposed method serves to demonstrate that, despite the additional complexity introduced in the target search problem, efficient solutions could be produced and resulting computational effort remains compatible with realistic operational time constraints (e.g., on the order of fifteen minutes, corresponding to the time required for UAVs to reach the search area).

In our preliminary experiments, population-based and classical local searches proved unsatisfactory. The difficulty arises from the high sensitivity of the target behavior: a small change in the allocation may induce a significant shift in the target's strategy. For example, adding resources to a given (level, cell) pair can trigger a SAG-like reaction, as the target sees a searcher, causing the target to adopt a completely different evasion and avoid areas already planned to be searched later in the time horizon. i.e., a small step can significantly reduce the overall detection probability and the impact of already-made decisions although it may be done in the right direction. Therefore, it is crucial to adopt a solving approach that allows temporary acceptance of degrading solutions and that avoids visiting the entirety close neighborhood of a solution, which motivated our choice of Simulated Annealing (SA).

SA is a classical metaheuristic that generates random solutions in the neighborhood of the current solution and accepts them as new current solutions either if they improve the objective function or with a probability based on the degradation and a temperature parameter, even if they are worse. The temperature decreases over time, allowing the early steps to explore the solution space widely, while later steps focus on exploiting the best solutions found. In our context, the initial steps are useful for selecting which (level, cell) pairs to search, whereas the later steps fine-tune the amount of resources allocated to the chosen pairs. SA is adapted to the specificities of our problem.

The overall procedure is summarized in Algorithm 1. To better recognize the algorithm parameters, we denoted them using a notation p_x for all parameters x.

Algorithm 1 Simulated Annealing for 3D Search allocation problem

```
solution ← GenerateRandomSolution()
temperature ← p_T
while stopping criterion not met do
    new_solution ← GenerateNeighbor(solution)
    if acceptationCriterion(new_solution, solution, temperature) then
        solution ← new_solution
    end if
    if restart criterion met then
        temperature ← p_T
    else
        temperature ← temperature × p_α
    end if
end while
```

Algorithm 2 Generate Neighbor for Simulated Annealing

```
for i = 1 to p_N do
    (level, cell, period) ← SelectRandomTupleWithResources()
    (level', cell', period) ← SelectRandomTuple()
    φ_{level,cell,period} ← φ_{level,cell,period} − p_S
    φ_{level',cell',period} ← φ_{level',cell',period} + p_S
end for
```

The random solution used as an initial basis is constructed by randomly allotting all resources; i.e., it is constructed such that for each time period in $\mathcal{T}$, $\Phi(t)$ resources are allocated. Note that it makes sure to allocate multiples of p_S, p_S denoting the step parameter of our algorithm.

The algorithm repeats p_N times (with p_N denoting the neighborhood size parameter) the process of moving resources between cells. It selects randomly a tuple (level, cell, period), where more than p_S resources are allocated, and transfers p_S resources to a randomly chosen tuple of the same period. Note that the same tuple may be selected,

Algorithm Parameter	Default value
p_R	3
p_L	5 000
p_T	1
p_α	0.99
p_S	0.1
p_N	4

TABLE I: Default values for our algorithm

meaning that some iterations do not modify the solution and thereby a neighborhood of size p_N implicitly includes all neighborhoods of smaller size. The neighbor generation procedure is detailed in Algorithm 2. The parameter p_α controls the temperature's decrease over time, while p_T defines the initial temperature value.

We also include a restart mechanism in our algorithm, which resets the temperature to its initial value in order to encourage exploration when the search becomes stuck in a solution. The restart criterion is triggered when no improvement has been achieved for p_L consecutive iterations (p_L being the parameter for loops without improvement). The stopping criterion is reached when the algorithm has restarted p_R times (restart parameter) and, in addition, no improvement has been achieved for the last p_L iterations since last restart.

Finally, the acceptance criterion for a new solution depends on the current solution and the temperature: a new solution is always accepted if it improves the objective function, otherwise it is accepted with a probability that decreases with the degradation of the solution and lower temperatures. It is accepted if (rand() $< e^{-\frac{f_1-f_2}{T}}$), with f_1 and f_2 being the objective function of new_solution and solution respectively, T the temperature and rand() a random generator between 0 and 1.

Recommended default values, obtained after preliminary experiments, are listed within Table I. These values were selected to allow for tractable computation times in our experiments, however they have to be adapted to the instance solved and computation time allowed. The experiment section can be use as a basis to give an overall idea on expected computation time with these default values.

V. Experimentation

The present section evaluates the efficiency of the proposed SA algorithm for solving the 3D search allocation game introduced in this work. Specifically, we investigate two aspects: computation time and solution quality. Since computation time naturally depends on the size of the instances considered, we vary the most relevant factors to assess scalability. Our experimentation addresses the following Research Questions (**RQs**):

- **RQ1**: How does the algorithm scale with the two dimensional discretization of the search space (i.e., the number of cells)?
- **RQ2**: How does it scale with the discretization of the third dimension (i.e., the number of altitude levels)?
- **RQ3**: How does it scale with time discretization (i.e., the number of time periods)?
- **RQ4**: What is the quality of the solutions produced?

The first three research questions focus on computation time and aim to demonstrate the scalability of the proposed solution method with respect to the problem parameters that most strongly affect complexity. The fourth research question concentrates exclusively on the quality of the obtained solutions.

For the experiments, we implemented our solution method in C++. Our experiments are conducted on an Intel Core i5-1135G7 with a clock speed of 2.40GHz, with 8 cores and 16 GB of RAM. All reported computation times are measured in term of CPU time. Default values listed in Table I are used for our algorithm.

A. Instance generator

To evaluate the performance of our method in terms of both computation time and solution quality, we run it on a set of varying instances designed to highlight performance differences. For this purpose, we developed an instance generator that creates random problem instances based on a set of input parameters. The generator first discretizes the search space into a grid of size $l \times L$, where l (number of columns) and L (number of rows) are input parameters. The altitude levels are defined as $\mathcal{L} = \{1, \ldots, m\}$. For each cell, the effect of altitude on coverage is computed as illustrated in Figure 1: each additional altitude level increases the coverage range by one cell in every direction from the initial position. In addition, each cell is assigned a random search efficiency value between 0 and 1. The efficiency of searching from position $(\text{level}, \text{cell})$ on covered cell c' is defined as this efficiency value divided by the corresponding altitude level. The time horizon is divided into n periods. The target always starts in the central area of the grid, defined by the intersection of the middle row and the middle column. If l (the number of columns) is even, there are two central columns, and similarly, if L (the number of rows) is even, there are two central rows, resulting in up to four possible starting cells. Random probabilities are assigned to starting cells. During the mission, if the target does not see a searcher, it moves with equal probability to one of its adjacent cells, including diagonally adjacent cells and its current cell. It has the same set of neighbor cells when following a SAG-like movement. Finally, the available search resource is normalized to 1 per time period.

Default values for our instance generator are summarized in Table II and the neighborhood of a cell, used for SAG and SAP (with equiprobable movements), is illustrated with Figure 2.

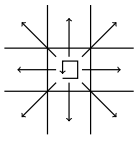


Fig. 2: Neighbor cells

Generator Parameter	Default value
m	3
$l \times L$	10×10
n	10

TABLE II: Default values of the instance generator

#periods	5	10	15	20
CPU time (s)	23.4411	73.3933	143.0043	231.3225

(a) CPU time results for different numbers of periods

$l \times L$	8×8	10×10	15×15	20×20
CPU time (s)	39.2572	73.3933	205.4665	424.5744

(b) CPU time results for different grid sizes

#altitude	2	3	4	5
CPU time (s)	26.4401	73.3933	123.2127	193.3706

(c) CPU time results for different numbers of altitude levels

TABLE III: CPU time results for different problem parameters

#periods	5	10	15	20
Improvement by Gurobi	7.17%	3.67%	4.37%	3.63%

(a) Improvement made by Gurobi for different numbers of periods

$l \times L$	8×8	10×10	15×15	20×20
Improvement by Gurobi	2.15%	3.67%	2.35%	3.85%

(b) Improvement made by Gurobi for different grid sizes

#altitude	2	3	4	5
Improvement by Gurobi	2.90%	3.67%	2.97%	3.41%

(c) Improvement made by Gurobi for different numbers of altitude levels

TABLE IV: Improvement made by Gurobi for different problem parameters

*B. **RQ1**, **RQ2** and **RQ3** : computation time scalability*

To address the scalability-related research questions, we evaluate our method on a wide range of instances with varying parameters. Specifically, we vary the number of periods $n \in \{5, 10, 15, 20\}$, generating thirty instances for each value while keeping all other parameters fixed at their default values. We then repeat the same procedure by varying the grid size $l \times L \in \{8 \times 8, 10 \times 10, 15 \times 15, 20 \times 20\}$ and the number of altitude levels $m \in \{3, 4, 5\}$. For each setting, we generate a benchmark of thirty random instances, again fixing the non-tested parameters to their default values. We report the average CPU time over all instances in Table IIIa, Table IIIb, and Table IIIc. Note that similar values are found for instances with parameters taking their default values (i.e., instances with $n = 10$, $l \times L = 10 \times 10$, and $m = 3$) as we do not need to re-generate instances and re-run the algorithm for Table IIIb and Table IIIc since results with the same generation parameters are available in Table IIIa.

These results highlight the impact that instance size characteristics, related to the space and time discretization precision (grid size, number of time periods, and number of altitude levels), have on the computation time of the solution method. As shown, all discretization precision parameters have significant effects on computation time, with a fast increase of computation time. However, the method scales well enough to remain under a few minutes of computation for all tested parameters, which is consistent with the targeted application's time constraints (a few minutes of computation that can be parallelized during UAV deployment setup). It demonstrates the feasibility of overcoming simplifications such as SAP and SAG, and adding a third dimension to the problem. Moreover, this experimentation also provides guidance on the appropriate discretization precision and method parameters relative to the available computation time.

*C. **RQ4** : Quality of the Solutions*

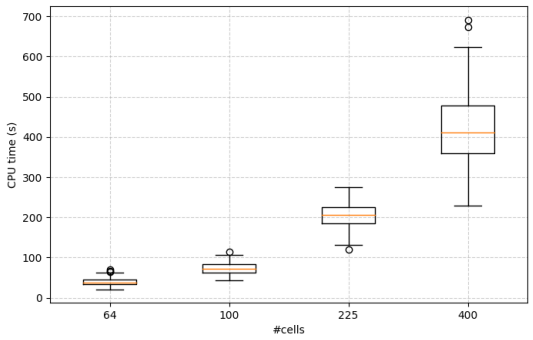
In this section, we evaluate the quality of the solutions by comparing them to reference solutions. Due to the complexity of the problem, MINLP with many constraints and variables, it is not realistic to expect exact solutions from exact solvers. Instead, we rely on Gurobi to compute the best possible solutions within a limited computation time. Specifically, Gurobi is used to solve Model 1 with a runtime limit of twenty minutes per instance, for each instance solved in the previous experiment. Note that Gurobi does no consider CPU-time limit, rather elapsed-time limit, thus meaning that actual CPU time is greater than twenty minutes, and was actually in average a little bit above one hour. To further improve the quality of these reference solutions, Gurobi is warm-started with our solutions. We report the average relative improvement in the objective function, computed as $1 - \frac{\text{Objective value found by Gurobi}}{\text{Objective value found by our solution}}$. The results are presented in Table IVa, Table IVb, and Table IVc. We emphasize that the present work does not aim to propose the best possible solution method for this problem, but rather to provide a first step toward the modeling of 3D search with complex target motion. In this context, the SA approach is used to demonstrate that efficient solutions can be obtained within realistic computation times.

This second experiment shows that, despite having more CPU time and being warm-started, a solver such as GUROBI brings only a small improvement to the solution of SA. Only marginal improvements are obtained despite the large difference in computation time. The worst improvement are observed for small numbers of periods, which shows the intricacy of the problem. In conclusion, these results show that our method is able to produce efficient solutions within a limited computation time. Although, more detailed analyses of the problem's complexity are required to fully understand which parameters positively influence the results.

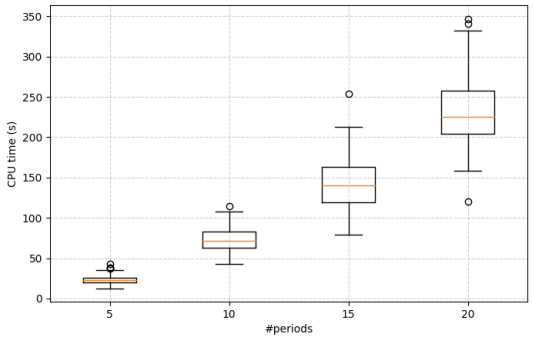
VI. Conclusion

In this work, we addressed a novel problem in the search theory literature by focusing on UAV applications of target search with the inclusion of 3D mobility for the searcher. This 3D mobility has two key implications: overlapping fields of view and adaptive target evasion. We proposed and formulated the 3D search allocation game with a multi-behaviors target. In the present work, we introduced the mathematical model in which the target alternates between SAP (non-reactive evasion) and SAG (perfect evasion) behaviors. We also propose a solution method specifically tailored to the characteristics of this problem, a simulated annealing method. Through experimentation, we demonstrate the potential of the proposed method.

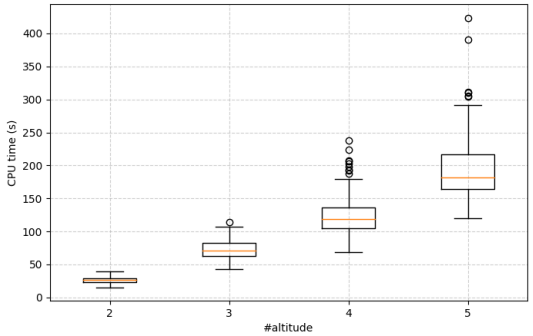
Though, this work focused on the 3D search modelization



(a) Results for varying #cells



(b) Results for varying #periods



(c) Results for varying #altitude levels

Fig. 3: Boxplots of the CPU time results of our simulated annealing method

in search theory, which led to a first step toward modeling complex evasion. Future research should consider even more complex evasion, with the target alternating between many behaviors, e.g., different SAP-like behaviors model through different prediction models. Moreover, more efficient solution methods can be proposed as the present focused on formulation a modelization thus proposing a straightforward method.

REFERENCES

[1] B. O. Koopman, "The theory of search. i. kinematic bases," *Operations research*, vol. 4, no. 3, pp. 324–346, 1956.

[2] L. D. Stone, J. O. Royset, A. R. Washburn, *et al.*, "Optimal search for moving targets," 2016.

[3] S. Waharte and N. Trigoni, "Supporting search and rescue operations with uavs," in *2010 International Conference on Emerging Security Technologies*. IEEE, 2010, pp. 142–147.

[4] Y.-J. Zheng, Y.-C. Du, W.-G. Sheng, and H.-F. Ling, "Collaborative human–uav search and rescue for missing tourists in nature reserves," *INFORMS Journal on Applied Analytics*, vol. 49, no. 5, pp. 371–383, 2019.

[5] Y.-J. Zheng, Y.-C. Du, H.-F. Ling, W.-G. Sheng, and S.-Y. Chen, "Evolutionary collaborative human-uav search for escaped criminals," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 217–231, 2019.

[6] H. A. Le Thi, D. M. Nguyen, and T. P. Dinh, "A based-dc programming approach for planning a multisensor multizone search for a moving target," in *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Springer, 2015, pp. 107–118.

[7] M. Lejeune, J. O. Royset, and W. Ma, "Multi-agent search for a moving and camouflaging target," *Naval Research Logistics (NRL)*, vol. 71, no. 4, pp. 532–552, 2024.

[8] F. Delavernhe, "Multi-objective search game: Long-term vs short-term," *Computers & Operations Research*, vol. 164, p. 106551, 2024.

[9] R. Hohzaki, "Search games: Literature and survey," *Journal of the Operations Research Society of Japan*, vol. 59, no. 1, pp. 1–34, 2016.

[10] F. Delavernhe, P. Jaillet, A. Rossi, and M. Sevaux, "Planning a multi-sensors search for a moving target considering traveling costs," *European Journal of Operational Research*, vol. 292, no. 2, pp. 469–482, 2021.

[11] V. Gonzalez and P. Jaillet, "Multi-drone rescue search in a large network," *European Journal of Operational Research*, vol. 324, no. 3, pp. 787–798, 2025.

[12] H. Vaillaud, C. Hanen, E. Hyon, and C. Enderli, "Target search with an allocation of search effort to overlapping cones of observation," in *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*. IEEE, 2023, pp. 801–811.

[13] C. Simonin, J.-P. Le Cadre, and F. Dambreville, "A hierarchical approach for planning a multisensor multizone search for a moving target," *Computers & Operations Research*, vol. 36, no. 7, pp. 2179–2192, 2009.

[14] R. Hohzaki, K. Iida, and T. Komiya, "Discrete search allocation game with energy constraints," *Journal of the Operations Research Society of Japan*, vol. 45, no. 1, pp. 93–108, 2002.

[15] R. Hohzaki, "Search allocation game," *European Journal of Operational Research*, vol. 172, no. 1, pp. 101–119, 2006.

[16] V. V. Gusev, "A multi-stage model of searching for two mobile objects on a graph," in *Frontiers in Games and Dynamic Games*. Springer, 2020, pp. 153–173.

[17] T. Matsuo and R. Hohzaki, "A search game with incomplete information about target's energy," *Scientiae Mathematicae Japonicae*, vol. 80, no. 1, pp. 57–76, 2017.

[18] R. Hohzaki, "A search game with incomplete information on detective capability of searcher," *Contributions to Game Theory and Management*, vol. 10, no. 0, pp. 129–142, 2017.

[19] B. O. Koopman, "The theory of search: Iii. the optimum distribution of searching effort," *Operations research*, vol. 5, no. 5, pp. 613–626, 1957.

[20] R. Hohzaki, "The search allocation game," *Wiely Encyclopedia of Operations Research and Management Science*, pp. 1–10, 2013.