Cyclic-Routing of Unmanned Aerial Vehicles☆

Nir Drucker[a], Hsi-Ming Ho[b], Joël Ouaknine[c,b], Michal Penn[a], Ofer Strichman[a]

[a] *Industrial Engineering and Management, Technion, Haifa, Israel*
[b] *Department of Computer Science, University of Oxford, Oxford, UK*
[c] *Max Planck Institute for Software Systems, Saarland Informatics Campus, Germany*

Abstract

Various missions carried out by Unmanned Aerial Vehicles (UAVs) are concerned with permanent monitoring of a predefined set of ground targets under *relative deadline* constraints, i.e., the targets have to be revisited 'indefinitely' and there is an upper bound on the time between two consecutive successful scans of each target. A *solution* to the problem is a set of routes—one for each UAV—that jointly satisfy these constraints. Our goal is to find a solution with the least number of UAVs. We show that the decision version of the problem (given k, is there a solution with k UAVs?) is PSPACE-complete. On the practical side, we propose a portfolio approach that combines the strengths of constraint solving and model checking. We present an empirical evaluation of the different solution methods on several hundred randomly generated instances.

Keywords:
motion planning, computational complexity, model checking

1. Introduction

Unmanned Aerial Vehicles (UAVs) have many uses, ranging from civilian to military operations. Like other autonomous systems, they are particularly well-suited to 'dull, dirty and/or dangerous' missions [2]. A common scenario in such missions is that a set of targets have to be visited (or, more aptly, *scanned*) by a limited number of UAVs. This has given rise to a large body of research on *route planning* for UAVs. Depending on the specific application at hand, routes of UAVs may be subject to various complex constraints, e.g., related to kinematics or fuel (see, among others, [3, 4, 5, 6]). In this work, we focus primarily on timing constraints and consider the *Cyclic-Routing UAV* problem (CR-UAV) [7]: a *recurrent* UAV route-planning problem in which each target must be scanned not only once but repeatedly, i.e., at intervals of prescribed maximal duration (*relative deadline*). With the minimal *flight time* between each (ordered) pair of the targets and the minimal *scanning time* needed for each target also given as inputs, a *solution* is a set of (infinite) routes that together satisfy these timing constraints. The goal is to find a solution with the least possible number of UAVs. As an example, consider a number of sensitive areas that have to be monitored permanently to prevent intrusion. An obvious solution is to simply deploy one UAV to stay at each location. To save costs, however, one seeks to minimise the number of UAVs used while ensuring that the time between any two consecutive successful scans of each target, plus the reaction time of security forces, is not longer than the time it takes for a successful intrusion. Similar applications include inspection of bridges and dams for damage, surveillance of oil and gas pipelines for spillage, monitoring of forests for fire, etc.

☆Part of this work appeared in the *Proceedings of the 18th International Conference on Foundations of Software Science and Computation Structures* (2015), and in the *Proceedings of the 13th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming* (2016). Joël Ouaknine was supported by ERC grant AVS-ISS (648701), and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) — Projektnummer 389792660 — TRR 248.

Email addresses: nirdru@gmail.com (Nir Drucker), hsimho@gmail.com (Hsi-Ming Ho), joel@mpi-sws.org (Joël Ouaknine), mpenn@ie.technion.ac.il (Michal Penn), ofers@ie.technion.ac.il (Ofer Strichman)

We give a formal definition of the CR-UAV problem in Section 2. While the problem is formulated in a very general manner, we make some useful observations to simplify the analysis and modelling in later sections. In particular, we can focus on solutions that are *periodic* and *integer-timed*. The complexity of the CR-UAV problem is studied in Section 3. We first show that the period of a solution can be exponential in the size of the inputs, refuting an incorrect claim in the literature [8]. Then, extending this idea, we show that the problem is indeed PSPACE-hard by reducing from the PERIODIC SAT problem, known to be PSPACE-complete [9]. We derive lower and upper bounds on the number k of required UAVs in Section 4 and detail a reduction from the decision version of the CR-UAV problem to the model-checking problem for Linear Temporal Logic (LTL) [10] on symbolic transition systems in Section 5. The CR-UAV problem can then be solved by a binary search on k, invoking a model checker in each step. In Section 6, we propose a simple approach to the CR-UAV problem: use a satisfiability-modulo-theories (SMT) solver [11] to find solutions and run, in parallel, a dedicated algorithm to prove that no solution exists. We also exhibit a method to reduce the searched state-space, which simplifies both tasks, based on a *simulation relation*. Experimental results are reported in Section 7 where we compare the efficiency of the approaches proposed in Sections 5 and 6. We close the article with some discussion and outlook in Section 8.

This paper is a revised and extended version of two conference papers [12, 13]. This new version contains a more precise formulation of the problem, a reduction to model checking, and new experimental results.

2. Definitions

2.1. Scenario and problem inputs

Let there be a number of targets, i.e., areas to be scanned, and a flock of UAVs, which we treat as free-moving points. Each target is associated with a ***scanning time***: the least amount of time a UAV must stay at the target to complete a successful scan. Moreover, each target has a ***relative deadline***: an upper bound requirement on the time between two consecutive successful scans. We make several assumptions:

- When more than one UAV are involved, each UAV flies at a different altitude. This allows us to ignore the issue of intersecting routes that might otherwise lead to collisions.
- All UAVs are identical, i.e., their top speeds are the same. It follows that we can associate with each ordered pair of targets a ***flight time***: the least amount of time needed for a UAV to fly from *some* point within the first target to *some* point within the other.
- Scans of targets are not split between UAVs.
- The scanning time of each target is not less than the least amount of time needed for a UAV to fly between *any* two points within the target. Thus, we can ignore the actual shape of each target and imagine it as a circle with scanning time as the 'diameter'.

In the rest of this article we fix the set of n targets to be $V = \{0, \ldots, n-1\}$ and let $v_i = i$, $v_1 = 1$, …, $v_{n-1} = n-1$. In this way, we can refer to the elements of V not only as targets, but also as unique indices. The inputs are as follows:

1. An array ST of size n such that for every $v \in V$, $ST[v]$ is the scanning time of v.
2. An array RD of size n such that for every $v \in V$, $RD[v]$ is the relative deadline of v.
3. A two-dimensional array FT of size $n \times n$ such that for every pair $v, v' \in V$, $FT[v, v']$ is the flight time from v to v'. In particular, $FT[v, v'] = 0$ if and only if $v = v'$.

Time units are chosen so that all the scanning times, relative deadlines and flight times are non-negative integers. We further assume that $FT[v, v'] + ST[v'] + FT[v', v''] \geq FT[v, v'']$ for all $v, v', v'' \in V$—as we will show later on, we can transform the input problem into an equivalent one without scanning times, from which it then follows that we are working with a metric. Without loss of generality, we further require that (i) $ST[v]$ is even for all $v \in V$ (as we will divide $ST[v]$ by 2 in the said transformation), (ii) $RD[v] > 0$ for all $v \in V$.[1]

[1] If $RD[v] = 0$ one may simply assign to each such v a dedicated UAV; it can be shown that this strategy does not affect the least number of UAVs required.

2.2. Routes and solutions

Now let $\mathcal{I} = \langle ST, RD, FT \rangle$ be an ***instance*** of the problem. A ***route*** of $\mathcal{I}$ is a function r from $\mathbb{R}_{>0}$ (the positive reals) to $V_\epsilon = V \cup \{\epsilon\}$ such that:

- For all $t, t' \in \mathbb{R}_{>0}$, $t < t'$ and $v, v' \in V$ such that $r(t) = v$ and $r(t') = v'$, we have $t' - t \geq FT[v, v']$.

Intuitively, a route records the trajectory of a UAV, which may be at some target or 'in transit' (represented by the special symbol ϵ) at any given time. For a positive integer P, we denote by $r\restriction_{(0,P]}$ the restriction of r to the right-closed interval $(0, P]$. If $r = (r\restriction_{(0,P]})^\omega$, i.e., r is the infinite concatenation of $r\restriction_{(0,P]}$, we say that r is ***periodic***, P is a ***period*** of r, and $r\restriction_{(0,P]}$ is a ***cycle*** of r. To capture successful scans, we define a function $secured_r$ from $\mathbb{R}_{>0}$ to V_ϵ for each route r: $secured_r(t) = v$ if (i) $ST[v] = 0$ and $r(t) = v$ or (ii) $ST[v] > 0$ and $r(t') = v$ for all $t' \in (t - ST[v], t)$; otherwise $secured_r(t) = \epsilon$. Intuitively, $secured_r(t) = v$ for $v \in V$ only when the UAV stays 'long enough' at v. A ***solution*** to $\mathcal{I}$ with k UAVs ($k \geq 1$) is an indexed set S of k routes that satisfies the following conditions:

- For all $v \in V$, we have $secured_r(t) = v$ for some $r \in S$ and $t \in (0, RD[v]]$.
- For all $v \in V$, $t, t' \in \mathbb{R}_{>0}$ with $t' - t = RD[v]$, we have (i) $secured_r(t'') = v$ for some $r \in S$ and $t'' \in (t, t')$, or (ii) $secured_r(t) = v$ for some $r \in S$ and $secured_{r'}(t') = v$ for some $r' \in S$.

The first condition requires that the *first* successful scan of each target v to be done, at the very latest, by its relative deadline $RD[v]$.[2] The second condition ensures that in any time interval of length $RD[v]$, a successful scan must happen within the inteval or at its endpoints. We say $S = \{r_0, \ldots, r_{k-1}\}$ has a period P if $r_i = (r_i\restriction_{(0,P]})^\omega$ for all $i \in \{0, \ldots, k-1\}$; in this case, $S' = \{r_0\restriction_{(0,P]}, \ldots, r_{k-1}\restriction_{(0,P]}\}$ is a cycle of S. In particular, if each r_i ($i \in \{0, \ldots, k-1\}$) has a period P_i, then the least common multiple of P_1, …, P_{k-1} is a period of $S = \{r_0, \ldots, r_{k-1}\}$.

Example 1. Consider $\mathcal{I} = \langle ST, RD, FT \rangle$ where

$$ST = \begin{bmatrix} 0 & 4 & 0 \end{bmatrix} \qquad RD = \begin{bmatrix} 10 & 5 & 10 \end{bmatrix} \qquad FT = \begin{bmatrix} 0 & 1 & 6 \\ 1 & 0 & 3 \\ 6 & 3 & 0 \end{bmatrix},$$

as illustrated in Fig. 1. Two solutions S^a (with a period 10) and S^b (with a period 18) to $\mathcal{I}$ are depicted in Figs. 2 and 3 where the horizontal axes are time t, the vertical axes are the indices of the targets, and the multiples of the period P are indicated with dashed lines. The set of routes $S^c = \{r_0^c, r_1^c\}$ (in Fig. 4) is not a solution to $\mathcal{I}$ since the relative deadlines of v_0 and v_2 are clearly violated. The function r_1^d (in Fig. 5) is not a route of $\mathcal{I}$ since $r_1^d(4) = v_0$, $r_1^d(8) = v_2$, but $8 - 4 < FT[v_0, v_2] = 6$.

2.3. Objective

We now formally define the CR-UAV problem and the associated decision problem.[3]

Definition 1 (The CR-UAV problem)**.** Given $\mathcal{I} = \langle ST, RD, FT \rangle$ as described above, output a solution with the least possible number of UAVs.

Definition 2 (The decision version of the CR-UAV problem)**.** Given $\mathcal{I} = \langle ST, RD, FT \rangle$ as described above and a positive integer k, is there a solution with k UAVs?

[2] Note in passing that we postulate that all UAVs use a global clock which starts at time 0.

[3] We assume that all numerical constants are encoded in binary, but our results hold irrespective of this assumption.

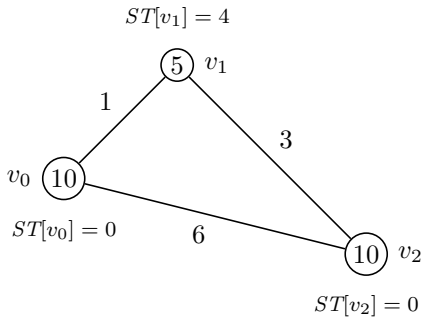

Figure 1: $\mathcal{I}$ in Example 1. The flight times are labelled on the edges (FT is symmetric in this example) and the relative deadlines are labelled on the targets; the scanning times are labelled adjacent to the targets.

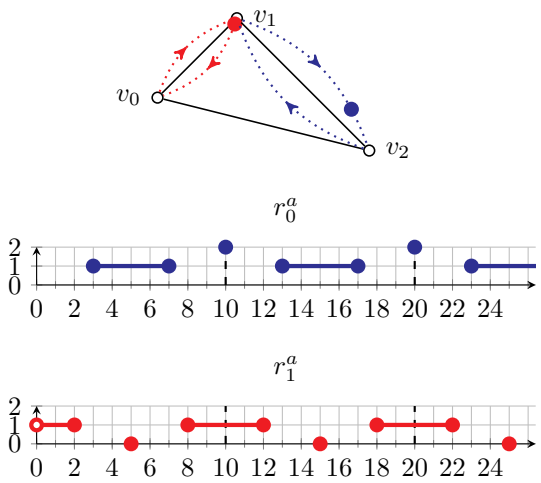

Figure 2: A $k = 2$ solution $S^a = \{r_0^a, r_1^a\}$ to $\mathcal{I}$ in Example 1.

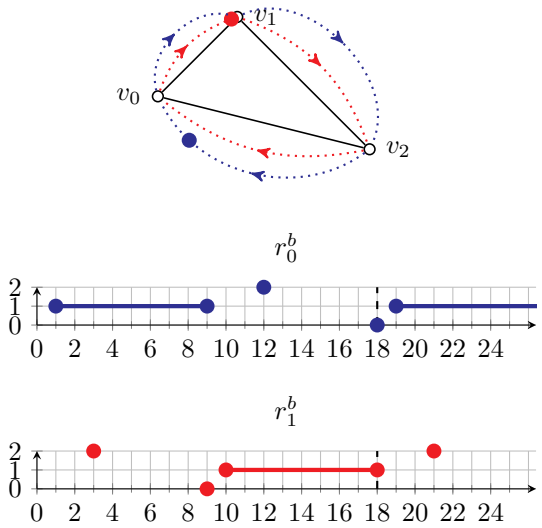

Figure 3: A $k = 2$ solution $S^b = \{r_0^b, r_1^b\}$ to $\mathcal{I}$ in Example 1.

2.4. Related work

As we will see in Section 3, (the decision version of) the CR-UAV problem is PSPACE-complete, and hence NP-complete problems cannot be more than weakly related to this problem. We nevertheless briefly mention here two problems that have interesting connections to CR-UAV.

The first of those is the *generalised windows scheduling* problem [14]. The input to this problem is a sequence of n pairs of positive integers $\langle (w_1, l_1), (w_2, l_2), \ldots, (w_n, l_n) \rangle$. Each such pair corresponds to a job that has to be scheduled, such that the processing length of job i is l_i slots and the window between two consecutive beginnings of executions of job i is at most w_i slots. The goal is to repeatedly and non-preemptively schedule all the jobs on the fewest possible machines under these constraints. This problem is a special case of CR-UAV, where flight times between targets are uniform. In that case we can simply add the flight time to the scanning time at the target and consider the move between targets as instantaneous (i.e., no flight time). Now the scanning time at target i is l_i and the relative deadline is w_i, which is exactly the generalised windows scheduling problem. Interestingly, [14] mentions that this problem is known to be NP-hard, but neither there or anywhere else did we find a result showing the exact complexity class of this problem.

Various versions of multiprocessor scheduling, starting from Liu and Layland's pivotal work [15], have certain characteristics that remind of CR-UAV. Generally the problem is to schedule tasks on one or more

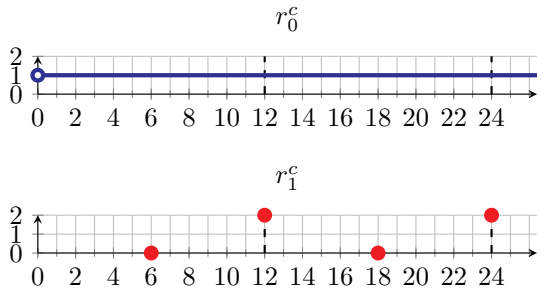

Figure 4: $S^c = \{r_0^c, r_1^c\}$ is not a solution to $\mathcal{I}$ in Example 1 as v_0 and v_2 miss their relative deadlines.

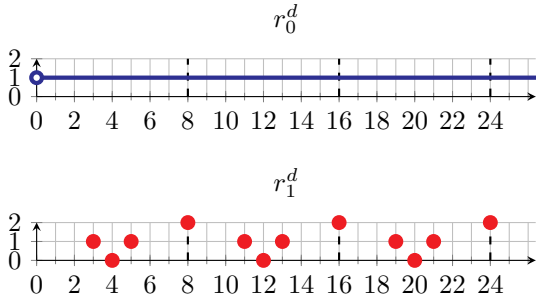

Figure 5: r_1^d is not a route of $\mathcal{I}$ in Example 1 as $r_1^d(4) = v_0$, $r_1^d(8) = v_2$ but $FT[v_0, v_2] = 6$.

processors periodically, with exact relative deadlines, in contrast to CR-UAV where the relative deadlines are only upper bounds (i.e., we can visit the target earlier and reset the clock). Also, the flight time in CR-UAV cannot simply be reduced to be the processing time in [15] because the processing time only depends on the current task whereas in CR-UAV the flight time depends on both the current and the next task.

Additional related work is mentioned in Section 8.

2.5. Some simplifying observations

We now describe some observations regarding inputs and solutions to the problem. As we will soon see, these observations greatly simplify the analysis and modelling in later sections.

Eliminating scanning times. Recall that each target v is an *area*, and we therefore treat it as a circle with 'diameter' $ST[v]$. This complication, however, can be avoided by 'shrinking' each target to a point. Formally, for $\mathcal{I} = \langle ST, RD, FT \rangle$, define $\mathit{shrink}(\mathcal{I}) = \langle ST', RD, FT' \rangle$ where $ST'[v] = 0$, $FT'[v, v] = 0$ for all $v \in V$, and $FT'[v, v'] = \frac{1}{2}ST[v] + FT[v, v'] + \frac{1}{2}ST[v']$ for all distinct $v, v' \in V$.

Proposition 1. *For any $\mathcal{I}$ and $k \geq 1$, $\mathcal{I}$ has a solution with k UAVs iff $\mathit{shrink}(\mathcal{I})$ has a solution with k UAVs.*

Proof. Consider a solution S to $\mathcal{I} = \langle ST, RD, FT \rangle$. We define, for each $r \in S$, a corresponding function r': $r'(t) = v$ ($v \in V$) if $\mathit{secured}_r(t + \frac{1}{2}ST[v]) = v$ and $r'(t) = \epsilon$ otherwise. If $r'(t) = v$, $r'(t') = v'$ for some $t, t' \in \mathbb{R}_{>0}$, $t < t'$ and distinct $v, v' \in V$, we have $\mathit{secured}_r(t + \frac{1}{2}ST[v]) = v$ and $\mathit{secured}_r(t' + \frac{1}{2}ST[v']) = v'$, which in turn give $r(t'') = v$ for all $t'' \in (t - \frac{1}{2}ST[v], t + \frac{1}{2}ST[v])$ and $r(t''') = v'$ for all $t''' \in (t' - \frac{1}{2}ST[v'], t' + \frac{1}{2}ST[v'])$ (assuming $ST[v], ST[v'] > 0$). Now since $t''' - t''$ can be arbitrarily close to $t' - \frac{1}{2}ST[v'] - (t + \frac{1}{2}ST[v])$, yet for any such t'' and t''' it must be the case that $t''' - t'' \geq FT[v, v']$, we have $t' - t \geq \frac{1}{2}ST[v] + FT[v, v'] + \frac{1}{2}ST[v']$. It follows that r' is a route of $\mathit{shrink}(\mathcal{I})$, and one can show that $\{r'_i \mid r_i \in S,\ 0 \leq i < |S|\}$ is a solution to $\mathit{shrink}(\mathcal{I})$ with a simple proof by contradiction. Conversely, given a solution S' to $\mathit{shrink}(\mathcal{I})$, we define a function $f_{r'}$ from $\mathbb{R}_{>0}$ to V_ϵ for each $r' \in S'$: $f_{r'}(t) = v$ if $r'(t - \frac{1}{2}ST[v]) = v$ and $f_{r'}(t) = \epsilon$ otherwise. Now define $r''(t) = v$ ($v \in V$) if (i) $ST[v] = 0$ and $f_{r'}(t) = v$, or (ii) $ST[v] > 0$ and there exists $t' > t$ such that $f_{r'}(t') = v$ and $t' - t < ST[v]$, and $r''(t) = \epsilon$ otherwise. Finally let $r(t) = r''(t + \frac{1}{2}\max_{v \in V}\{ST[v]\})$ (the purpose of the extra shifting is to ensure that the first successful scans to the targets are completed in time). Similarly, one can verify that r is a route of $\mathcal{I}$ and $\{r_i \mid r'_i \in S',\ 0 \leq i < |S'|\}$ is a solution to $\mathcal{I}$. □

Example 2. Consider $\mathcal{I}$ in Example 1. We have $\mathit{shrink}(\mathcal{I}) = \langle ST', RD, FT' \rangle$ where

$$ST = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \qquad RD = \begin{bmatrix} 10 & 5 & 10 \end{bmatrix} \qquad FT = \begin{bmatrix} 0 & 3 & 6 \\ 3 & 0 & 5 \\ 6 & 5 & 0 \end{bmatrix},$$

as illustrated in Fig. 6. Two solutions to $\mathit{shrink}(\mathcal{I})$, S^e (corresponding to S^a in Fig. 2) and S^f (corresponding to S^b in Fig. 3), are depicted in Figs. 7 and 8, respectively.

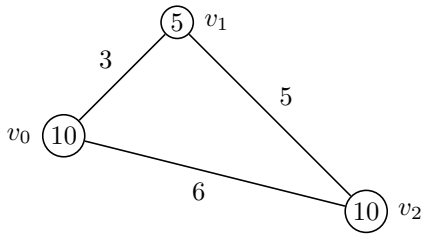

Figure 6: *shrink*($\mathcal{I}$) in Example 2.

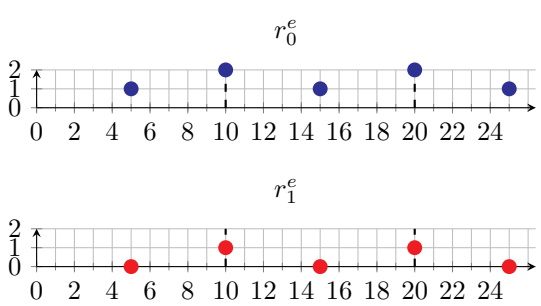

Figure 7: A solution $S^e = \{r_0^e, r_1^e\}$ to *shrink*($\mathcal{I}$) in Example 2.

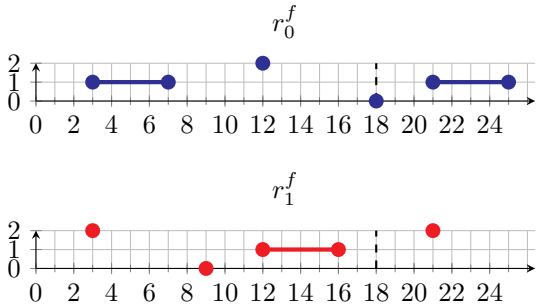

Figure 8: A solution $S^f = \{r_0^f, r_1^f\}$ to *shrink*($\mathcal{I}$) in Example 2.

Thanks to Proposition 1 we can (without loss of generality) disregard ST altogether and from now on consider only $\mathcal{I} = \langle RD, FT \rangle$ where $FT[v, v'] + FT[v', v''] \geq FT[v, v'']$ for all distinct v, v', v''. In the sequel, we call such an $\mathcal{I}$ an ***instance*** of the CR-UAV problem.

Digitisation of solutions. We say a route r is ***integer-timed*** iff the UAVs only reach/leave the targets at integer time points (i.e., for each non-negative integer N, either (i) $r(t') = v$ for all $t' \in (N, N+1)$ and some $v \in V$, or (ii) $r(t') = \epsilon$ for all $t' \in (N, N+1)$). We say a route r is ***instantaneous*** iff the UAV never stays at a target in r (i.e., $r(t) \neq \epsilon$ implies that t is a point of discontinuity of r).[4] These notions extend to solutions in the expected way. The following proposition is reminiscent of a standard digitisation result in the theory of timed automata [16].

Proposition 2. *For any $\mathcal{I}$ and $k \geq 1$, a solution to $\mathcal{I}$ with k UAVs can be transformed into an instantaneous integer-timed solution to $\mathcal{I}$ with k UAVs.*

Proof. Consider a solution S to $\mathcal{I} = \langle RD, FT \rangle$. We define, for each $r \in S$, a corresponding function r^d: $r^d(t) = v$ ($v \in V$) if t is a positive integer and (i) there exists $t' \geq t$ such that $t' - t < \frac{1}{2}$ and $r(t') = v$, or (ii) there exists $t'' \leq t$ such that $t - t'' \leq \frac{1}{2}$ and $r(t'') = v$, and $r^d(t) = \epsilon$ otherwise (note that r^d is well-defined as we have $FT[v, v'] \geq 1$ for all distinct $v, v' \in V$). Obviously, all the routes in $\{r_i^d \mid r_i \in S,\ 0 \leq i < |S|\}$ are instantaneous and integer-timed. Now suppose to the contrary that $\{r_i^d \mid r_i \in S,\ 0 \leq i < |S|\}$ is not a solution to $\mathcal{I}$; in other words, there are $t, t' \in \mathbb{R}_{>0}$ such that (i) $t' - t = RD[v] \geq 1$ for some $v \in V$, (ii) $r_i^d(t'') \neq v$ for all i, $0 \leq i < |S|$ and $t'' \in (t, t')$, and (iii) if $r_i^d(t) = v$ for some i, $0 \leq i < |S|$ then $r_i^d(t') \neq v$ for all i, $0 \leq i < |S|$, and vice versa. Consider the following cases:

- Both t and t' are integers: if $r_i^d(t) \neq v$ and $r_i^d(t') \neq v$ for all i, $0 \leq i < |S|$ then obviously $r_i(t'') \neq v$ for all i, $0 \leq i < |S|$ and $t'' \in [t, t']$. If, say, $r_i^d(t) = v$, we must have $r_i(t'') \neq v$ for all i, $0 \leq i < |S|$ and $t'' \in [t + \frac{1}{2}, t' + \frac{1}{2})$. In both subcases, S cannot be a solution to $\mathcal{I}$, which is a contradiction.

[4] Note that, however, we allow UAVs to take self-loops.

- Both t and t' are not integers: as $r_i^d(\lceil t \rceil) \neq v$ and $r_i^d(\lfloor t' \rfloor) \neq v$ for all i, $0 \leq i < |S|$, we have $r_i(t'') \neq v$ for all i, $0 \leq i < |S|$ and $t'' \in [\lceil t \rceil - \frac{1}{2}, \lfloor t' \rfloor + \frac{1}{2})$. Since $t' - t = \lfloor t' \rfloor - \lceil t \rceil + 1$, S cannot be a solution to $\mathcal{I}$, which is a contradiction. □

Example 3. Consider $\mathcal{J} = \langle RD, FT \rangle$ where

$$RD = \begin{bmatrix} 3 & 3 & 3 & 3 \end{bmatrix} \qquad FT = \begin{bmatrix} 0 & 2 & 3 & 2 \\ 2 & 0 & 2 & 3 \\ 3 & 2 & 0 & 2 \\ 2 & 3 & 2 & 0 \end{bmatrix},$$

as illustrated in Fig. 9. A solution S^g (in which the UAVs reach/leave the targets at non-integer points) and its 'digitised' counterpart S^h (in which the UAVs reach/leave the targets only at integer points) are depicted in Figs. 10 and 11, respectively.

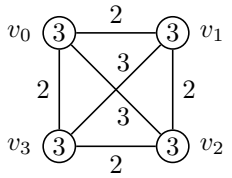

Figure 9: $\mathcal{J}$ in Example 3.

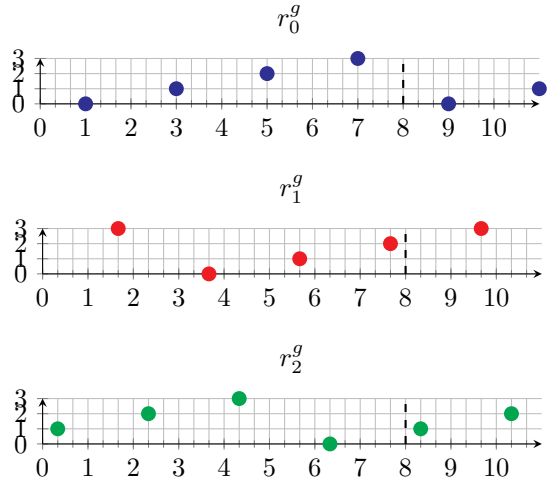

Figure 10: A solution $S^g = \{r_0^g, r_1^g, r_2^g\}$ to $\mathcal{J}$ in Example 3.

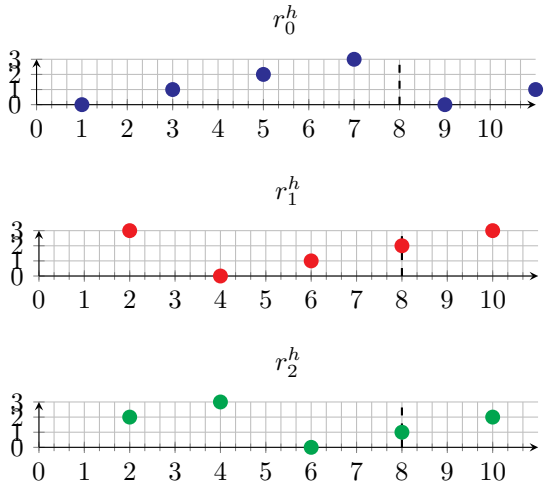

Figure 11: An integer-timed solution $S^h = \{r_0^h, r_1^h, r_2^h\}$ to $\mathcal{J}$ in Example 3.

In light of Proposition 2, from now on we let $FT[v, v] = 1$ (instead of 0) for all $v \in V$.

Punctuality of solutions. We say a route r is ***punctual*** iff for all $t, t' \in \mathbb{R}_{>0}$, $t < t'$, and $v, v' \in V$ such that $r(t) = v$, $r(t') = v'$, and $r(t'') = \epsilon$ for all $t'' \in (t, t')$, either (i) $v \neq v'$ and $t' - t = FT[v, v']$, or (ii) $v = v'$ and $t' - t = 1$. We now show that it suffices to consider punctual solutions (i.e., all routes are punctual).

Proposition 3. *For any $\mathcal{I}$ and $k \geq 1$, an instantaneous integer-timed solution to $\mathcal{I}$ with k UAVs can be transformed into a punctual instantaneous integer-timed solution to $\mathcal{I}$ with k UAVs.*

Proof. Consider an instantaneous integer-timed solution S to $\mathcal{I} = \langle RD, FT \rangle$. For each $r \in S$, let r' be a route such that

- $r'(t'') = v'$ ($v' \in V$) if t'' is a positive integer and either (i) $r(t'') = v'$, or (ii) $t'' \in (t, t')$, $r(t) = v$, $r(t') = v'$, $r(t''') = \epsilon$ for all $t''' \in (t, t')$, and $t'' - t \geq FT[v, v']$ for some positive integers t, t' and $v \in V$;
- $r'(t'') = \epsilon$ otherwise.

Intuitively, arriving earlier at targets and staying (whenever possible) cannot invalidate a solution. Thus, it is clear that $\{r'_i \mid r_i \in S,\ 0 \leq i < |S|\}$ is a punctual instantaneous integer-timed solution to $\mathcal{I}$. □

Periodicity of solutions. Our final observation, stated in the proposition below, enables us to consider only periodic solutions without sacrificing correctness.

Proposition 4. *For any $\mathcal{I}$ and $k \geq 1$, a punctual instantaneous integer-timed solution to $\mathcal{I}$ with k UAVs can be transformed into a periodic punctual instantaneous integer-timed solution to $\mathcal{I}$ with k UAVs.*

Proof. Given $\mathcal{I}$ and $k \geq 1$, let $\mathsf{prefix}_{\mathcal{I},k}$ be the set of k-tuples $\langle f_0, \ldots, f_{k-1} \rangle$ where there is a positive integer T such that

- each f_i is of the form $r\restriction_{(0,T]}$ where r is a punctual instantaneous integer-timed route of $\mathcal{I}$,
- $r_i(T) = v$ for some $i \in \{0, \ldots, k-1\}$ and $v \in V$, and
- for each $v \in V$, $r_i(T') = v$ for some $T' \leq T$ and $i \in \{0, \ldots, k-1\}$.

Now define an equivalence relation $\sim$ on $\mathsf{prefix}_{\mathcal{I},k}$: $\langle f_0, \ldots, f_{k-1} \rangle \sim \langle g_0, \ldots, g_{k-1} \rangle$ iff for each k-tuple $\langle r_0, \ldots, r_{k-1} \rangle$ of punctual instantaneous integer-timed routes of $\mathcal{I}$, $\{f_0 r_0, \ldots, f_{k-1} r_{k-1}\}$ is a solution to $\mathcal{I}$ whenever $\{g_0 r_0, \ldots, g_{k-1} r_{k-1}\}$ is a solution to $\mathcal{I}$. We will see in Section 5 that $\sim$ partitions $\mathsf{prefix}_{\mathcal{I},k}$ into finitely many equivalence classes. Therefore, if S is a punctual instantaneous integer-timed solution to $\mathcal{I}$ with k UAVs, there must be two distinct 'prefixes' of S that are $\sim$-equivalent. The proposition follows by repeating the in-between part *ad infinitum*. □

Corollary 1. *For any $\mathcal{I}$ and $k \geq 1$, a solution to $\mathcal{I}$ with k UAVs can be transformed into a periodic punctual instantaneous integer-timed solution to $\mathcal{I}$ with k UAVs.*

Following Corollary 1, we will only consider periodic punctual instantaneous integer-timed solutions in the rest of this article.

3. Complexity

We will see in Section 5 that the decision version of the CR-UAV problem can be solved in PSPACE via a rather straightforward reduction to LTL model-checking on symbolic transition systems [17]. On the other hand, it is not immediate whether the problem can be solved in NP. It is claimed in [8] that in the single-UAV case (i.e., $k = 1$), it suffices to consider solutions with periods bounded by the largest relative deadline; if constants are encoded in unary, this would immediately imply NP-membership of the problem in the single-UAV case. However, that claim is unfortunately wrong, because of the following proposition.

Proposition 5. *There is a family of* CR-UAV *instances $\{\mathcal{I}_n\}_{n>0}$ such that the shortest possible period of a single-UAV solution to $\mathcal{I}_n$ is exponential in the size of $\mathcal{I}_n$.*[5]

[5] This example was suggested by Daniel Bundala.

Proof. (*Sketch.*) See Fig. 12 for an illustration of $\mathcal{I}_3$. The i-th 'diamond' ($i \geq 1$, in top-down order) has p_n branches where p_n is the n-th prime number. Each unlabelled edge has flight time set to 1, and each missing edge has flight time set to the 'shortest distance' between the two relevant targets. It can be shown that a solution must be an infinite repetition of either (i) from v_{top} through all the diamonds to v_{bot}, to v_{mid} and to v_{top} again, or (ii) from v_{bot} through all the diamonds to v_{top}, to v_{mid} and to v_{bot} again; without loss of generality we assume (i). In this case, in each diamond one must go straight down, and only the edges shown in the figure can be used. It follows that the shortest period of a solution to $\mathcal{I}_n$ is bounded below by $\prod_{i=1}^{n} p_i = \Omega(e^n)$. On the other hand, the number of targets and the largest relative deadline in $\mathcal{I}_n$ are both $O(n^2 \ln n)$. □

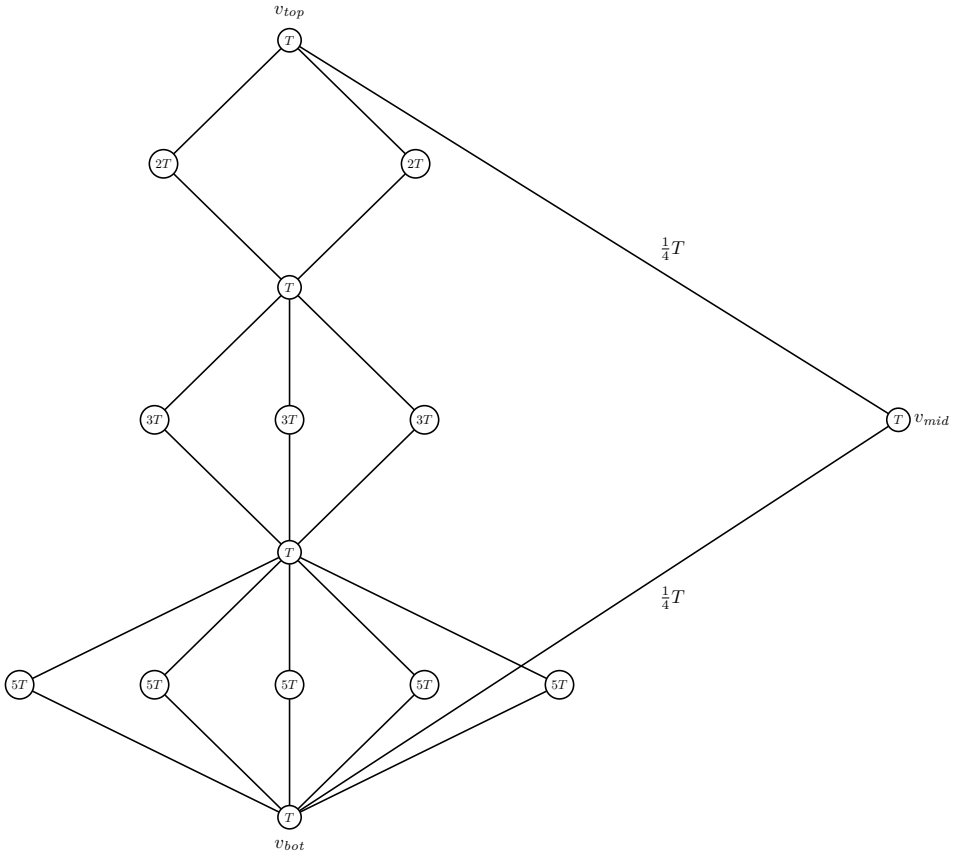

Figure 12: The CR-UAV instance $\mathcal{I}_3$ in Proposition 5 ($T = 12$).

In what follows, we show that the decision version of the CR-UAV problem is indeed PSPACE-hard even in the single-UAV case. The proof is accomplished by reduction from the PERIODIC SAT problem, known to be PSPACE-complete [9].

3.1. The PERIODIC SAT *problem*

The PERIODIC SAT problem is one of the many PSPACE-complete problems introduced in [9]. In the rest of this section, let X be a finite set of (boolean) variables and let X^j be the set of variables obtained from X by adding a superscript j to each variable.

Definition 3 (The PERIODIC SAT problem [9])**.** Consider a CNF formula $\varphi(0)$ over $X^0 \cup X^1$. Let $\varphi(j)$ be the formula obtained from $\varphi(0)$ by replacing all variables $x_i^0 \in X^0$ by x_i^j and all variables $x_i^1 \in X^1$ by x_i^{j+1}. Is there an assignment to $\bigcup_{j\geq 0} X^j$ such that $\bigwedge_{j\geq 0} \varphi(j)$ is satisfied?

Note that there are at most $2^{|X|}$ possible assignments to $|X|$. It follows that if there is a satisfying assignment to $\bigwedge_{j\geq 0} \varphi(j)$, it can be turned into a *periodic* satisfying assignment, i.e., there is a positive integer $N \leq 2^{|X|}$ such that for all non-negative integers j_1 and j_2, the truth values of X^{j_1} and X^{j_2} are identical if $j_1 \equiv j_2 \pmod N$.

Example 4. Let $X = \{x_1, x_2\}$ and

$$\varphi(0) = (x_1^0 \vee x_1^1) \wedge (x_2^0 \vee x_2^1) \wedge (\neg x_1^0 \vee \neg x_1^1) \wedge (\neg x_2^0 \vee \neg x_2^1)\,.$$

Then

$$\varphi(1) = (x_1^1 \vee x_1^2) \wedge (x_2^1 \vee x_2^2) \wedge (\neg x_1^1 \vee \neg x_1^2) \wedge (\neg x_2^1 \vee \neg x_2^2)\,,$$
$$\varphi(2) = (x_1^2 \vee x_1^3) \wedge (x_2^2 \vee x_2^3) \wedge (\neg x_1^2 \vee \neg x_1^3) \wedge (\neg x_2^2 \vee \neg x_2^3)\,,$$

and so on. The infinite conjunction $\bigwedge_{j\geq 0} \varphi(j)$ is satisfied by the following periodic assignment to variables $\bigcup_{j\geq 0} X^j$ (with $N = 2$):

$$(x_1^i, x_2^i) \mapsto \begin{cases} (\textbf{true}, \textbf{true}) & \text{if } i \text{ is even} \\ (\textbf{false}, \textbf{false}) & \text{otherwise.} \end{cases}$$

3.2. The construction

Consider a CNF formula $\varphi(0) = c_1 \wedge \cdots \wedge c_h$ over $X^0 = \{x_1^0, \ldots, x_m^0\}$ and $X^1 = \{x_1^1, \ldots, x_m^1\}$. Without loss of generality, we assume that each clause c_j of $\varphi(0)$ is non-trivial (i.e., c_j does not contain both positive and negative occurrences of a variable) and $m > 2$, $h > 0$. We construct an instance $\mathcal{I} = \langle RD, FT\rangle$ of the CR-UAV problem such that $\bigwedge_{j\geq 0} \varphi(j)$ is satisfiable iff $\mathcal{I}$ has a solution with a single UAV. The reduction can be done in polynomial time (the largest constant in $\mathcal{I}$ has magnitude $O(m^2 h)$ and $|V| = O(mh)$); in particular FT is symmetric, i.e., it forms a *metric* on V. The general idea of the reduction is inspired by the textbook reduction from 3SAT to HAMILTONIAN PATH [18]: assignments to variables correspond to the ways in which targets are visited, and a clause is satisfied if one of its laterals is 'hit'. More precisely, we construct ***variable gadgets*** that can be traversed in two 'directions' (corresponding to assignments **true** and **false** to variables). A ***clause target*** is visited if the corresponding clause is satisfied by the assignment. Crucially, we use ***consistency gadgets***, in which we set the relative deadlines of the targets carefully to ensure that the directions of traversals of the variable gadgets for X^1 (corresponding to a particular assignment to variables) in a given iteration is consistent with the directions of traversals of the variable gadgets for X^0 in the next iteration.

We now describe and explain each part of $\mathcal{I}$ in detail. The reader, however, is advised to glance ahead to Fig. 17 (on page 14) to get an impression of $\mathcal{I}$ as a whole. In what follows, let $l = 24h + 34$ and

$$T = 2\Big(m\big(2(3m+1)l + l\big) + m\big(2(3m+2)l + l\big) + l + 2h\Big)\,.$$

These expressions may look horrendous at this point, but they will become clear as we proceed; roughly speaking, l is the time needed to traverse a 'row' in Fig. 13, and T is the time needed to complete a round-trip in Fig. 17.

Variable gadgets. For each variable x_i^0, we construct a corresponding *variable gadget* with the following targets (see Fig. 13):

- Three targets on the left side ($LS_i = \{v_i^{t,L}, v_i^{m,L}, v_i^{b,L}\}$)
- Three targets on the right side ($RS_i = \{v_i^{t,R}, v_i^{m,R}, v_i^{b,R}\}$)

- A '*clause box*' ($CB_i^j = \{v_i^{a,j}, v_i^{b,j}, v_i^{c,j}, v_i^{d,j}, v_i^{e,j}, v_i^{f,j}\}$) for each $j \in \{1, \ldots, h\}$
- A '*separator box*' ($SB_i^j = \{v_i^{\bar{a},j}, v_i^{\bar{b},j}, v_i^{\bar{c},j}, v_i^{\bar{d},j}, v_i^{\bar{e},j}, v_i^{\bar{f},j}\}$) for each $j \in \{0, \ldots, h\}$
- A target at the top (v_{top} if $i = 0$, v_{i-1} otherwise)
- A target at the bottom (v_i).

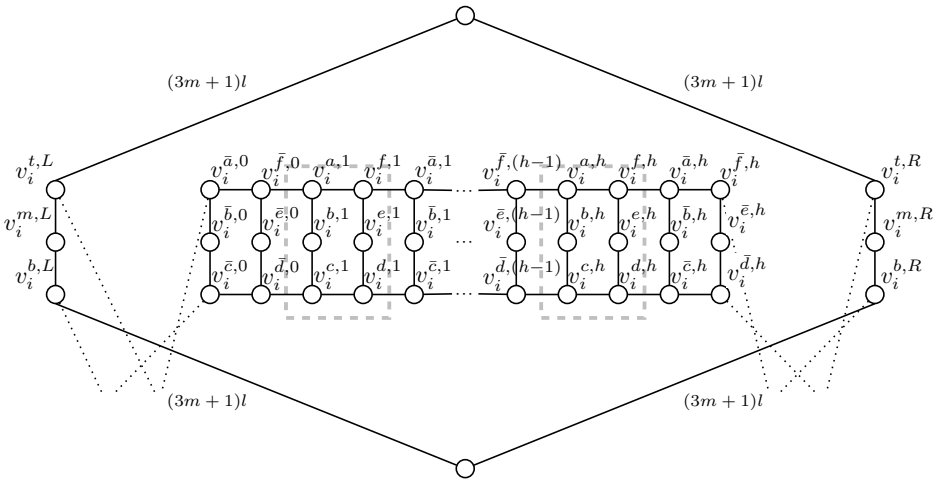

Figure 13: The variable gadget for x_i^0. Dotted lines indicate the connections to the i-th consistency gadgets LCG_i and RCG_i (not shown in this figure).

The clause boxes for $j \in \{1, \ldots, h\}$ are aligned horizontally in the figure. A separator box is laid between each adjacent pair of clause boxes and at both ends. This row of boxes ($Row_i = \bigcup_{j\in\{1,\ldots,h\}} CB_i^j \cup \bigcup_{j\in\{0,\ldots,h\}} SB_i^j$) is then put between LS_i and RS_i. The relative deadlines of all targets $v \in LS_i \cup RS_i \cup Row_i$ are set to $T+l+2h$. The targets are connected as indicated by solid lines. The four 'long' edges in the figure have their flight times set to $(3m+1)l$ while all other edges have flight times set to 2, e.g., $FT[v_{top}, v_1^{t,L}] = (3m+1)l$ and $FT[v_1^{b,1}, v_1^{c,1}] = 2$. There is an exception though: $FT[v_m^{b,L}, v_m]$ and $FT[v_m^{b,R}, v_m]$ (in the variable gadget for x_m^0) are set to $(3m+2)l$. The variable gadgets for variables x_i^1 are constructed almost identically. The three targets on the left and right side are now LS_{i+m} and RS_{i+m}. The set of targets in the row is now $Row_{i+m} = \bigcup_{j\in\{1,\ldots,h\}} CB_{i+m}^j \cup \bigcup_{j\in\{0,\ldots,h\}} SB_{i+m}^j$. The target at the top is v_{i+m-1} and the target at the bottom is v_{i+m} ($i \neq m$) or v_{bot} ($i = m$). The relative deadlines of targets in $LS_{i+m} \cup RS_{i+m} \cup Row_{i+m}$ are set to $T + l + 2h$, and the flight times of the edges are set as before, except that all the long edges now have flight times set to $(3m+2)l$. Now consider the following ordering of variables:

$$x_1^0, x_2^0, \ldots, x_m^0, x_1^1, x_2^1, \ldots, x_m^1 .$$

Observe that the variable gadgets for two 'neighbouring' variables (with respect to this ordering) have a target in common. Specifically, the set of the shared targets is $S = \{v_1, \ldots, v_{2m-1}\}$. We set the relative deadlines of all targets in S to $T + 2h$ and the relative deadlines of v_{top} and v_{bot} to T.

Clause targets. For each clause c_j in $\varphi(0)$, there is a *clause target* v^{c_j} with relative deadline set to $\frac{3}{2}T$. If x_i^0 occurs in c_j as a literal, we connect the j-th clause box in the variable gadget for x_i^0 to v^{c_j} as shown in Fig. 14 and set the flight times of these new edges to 2 (e.g., $FT[v^{c_j}, v_i^{c,j}] = FT[v^{c_j}, v_i^{d,j}] = 2$). If instead $\neg x_i^0$ occurs in c_j, then v^{c_j} is connected to $v_i^{a,j}$ and $v_i^{f,j}$ (with the flight times of the new edges set to 2). Likewise, the variable gadget for x_i^1 may be connected to v^{c_j} via $\{v_{i+m}^{c,j}, v_{i+m}^{d,j}\}$ (if x_i^1 occurs in c_j) or $\{v_{i+m}^{a,j}, v_{i+m}^{f,j}\}$ (if $\neg x_i^1$ occurs in c_j).

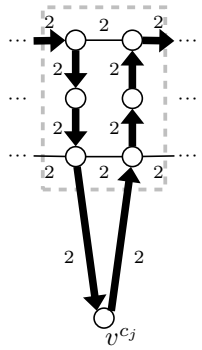

Figure 14: The variable occurs positively in c_j.

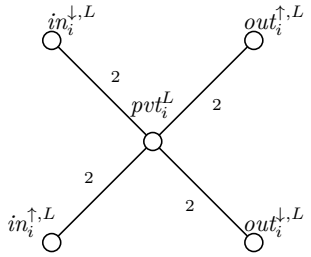

Figure 15: A consistency gadget LCG_i.

Consistency gadgets. For each $i \in \{1, \ldots, m\}$, we construct two *consistency gadgets* LCG_i (see Fig. 15) and RCG_i. In LCG_i, the target at the centre (pvt_i^L) has relative deadline set to $\frac{1}{2}T + m\big(2(3m+2)l + l\big) - (2i-1)l + 4h$. The other four targets ($in_i^{\downarrow,L}$, $out_i^{\uparrow,L}$, $in_i^{\uparrow,L}$ and $out_i^{\downarrow,L}$) have relative deadlines set to $\frac{3}{2}T$. The flight time from pvt_i^L to any of the other four targets is set to 2. RCG_i is identical except that the subscripts on the targets change from L to R. The i-th consistency gadgets LCG_i and RCG_i are connected to the variable gadgets for x_i^0 and x_i^1 as in Fig. 16. The targets $in_i^{\downarrow,L}, out_i^{\uparrow,L}, in_i^{\downarrow,R}, out_i^{\uparrow,R}$ are connected to certain targets in the variable gadget for x_i^0—this allows pvt_i^L and pvt_i^R to be traversed 'from above'. Similarly, the edges connected to $in_i^{\uparrow,L}, out_i^{\downarrow,L}, in_i^{\uparrow,L}, out_i^{\downarrow,L}$ allow pvt_i^L and pvt_i^R to be traversed 'from below'. Formally, $FT[v, v'] = 2$ if

- $v = in_i^{\downarrow,L}, v' \in \{v_i^{b,L}, v_i^{\bar{c},0}\}$ or $v = in_i^{\downarrow,R}, v' \in \{v_i^{\bar{f},h}, v_i^{b,R}\}$
- $v = out_i^{\uparrow,L}, v' \in \{v_i^{t,L}, v_i^{\bar{a},0}\}$ or $v = out_i^{\uparrow,R}, v' \in \{v_i^{\bar{d},h}, v_i^{t,R}\}$
- $v = in_i^{\uparrow,L}, v' \in \{v_{(i+m)}^{b,L}, v_{(i+m)}^{\bar{c},0}\}$ or $v = in_i^{\uparrow,R}, v' \in \{v_{(i+m)}^{\bar{f},h}, v_{(i+m)}^{b,R}\}$
- $v = out_i^{\downarrow,L}, v' \in \{v_{(i+m)}^{t,L}, v_{(i+m)}^{\bar{a},0}\}$ or $v = out_i^{\downarrow,R}, v' \in \{v_{(i+m)}^{\bar{d},h}, v_{(i+m)}^{t,R}\}$.

Finally, there is a target v_{mid} with $RD(v_{mid}) = T$ connected to v_{bot} and v_{top} with two edges, each of which has flight time set to $\frac{1}{4}T$. The flight times of all the missing edges are set to the 'shortest distance' between the two relevant targets; we will simply assume these edges are never taken. This completes the construction of $\mathcal{I}$. An example with $m = 3$ is given in Fig. 17, where targets in S (shared by two variable gadgets) are depicted as solid circles.

Proposition 6. *$\bigwedge_{j\geq 0} \varphi(j)$ is satisfiable iff $\mathcal{I}$ has a solution with a single UAV.*

3.3. Correctness of the reduction

The rest of this section is devoted to the proof of Proposition 6—in particular, we show that if $\mathcal{I}$ admits a solution with a single UAV, then each of the clause targets must be visited exactly once, and the directions in which the variable gadgets are traversed give the satisfying assignment to $\bigwedge_{j\geq 0} \varphi(j)$. Since we work with a single UAV, a solution may be conveniently denoted by an infinite sequence $s \in V^\omega$. The ***duration*** of $u \in V^\star$, which we denote by $dur(u)$, is defined to be the sum of the flight times along u. We first prove the forward direction. Given a satisfying assignment to $\bigwedge_{j\geq 0} \varphi(j)$, we construct a solution s as follows: s starts from v_{top} and goes through the variable gadgets for $x_1^0, x_2^0, \ldots, x_m^0, x_1^1, x_2^1, \ldots, x_m^1$ in order, eventually reaching v_{bot}. Each variable gadget is traversed according to the truth value assigned to its corresponding variable. In such a traversal, both pvt_i^L and pvt_i^R are visited once (see the thick arrows in Fig. 16 for the situation when x_i^0 is assigned **true** and x_i^1 is assigned **false**). Along the way from v_{top} to v_{bot}, s detours at certain times and 'hits' each clause target exactly once as illustrated by the thick arrows in Fig. 14 (this

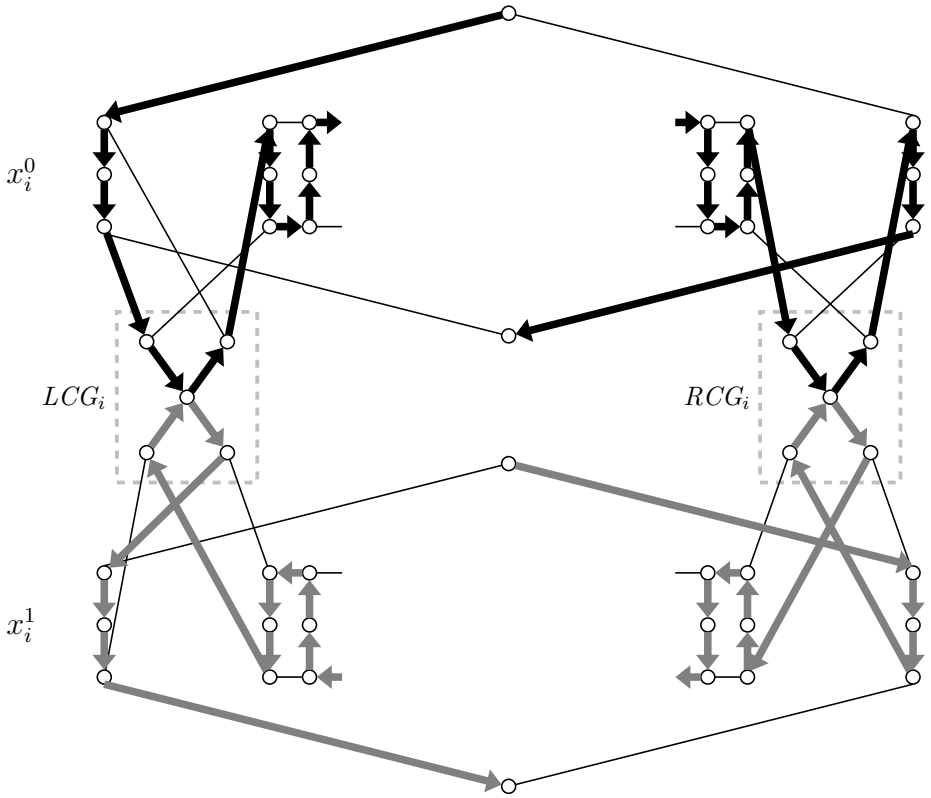

Figure 16: Connecting the variable gadgets for x_i^0 and x_i^1 to LCG_i and RCG_i. Parts of the intended path when x_i^0 is assigned to **true** and x_i^1 is assigned to **false** are indicated by the thick arrows: the UAV first follows the black thick arrows (thereby traversing the variable gadget for x_i^0 'left-to-right'), then it traverses through some other variable gadgets, and then it follows the grey thick arrows (thereby traversing the variable gadget for x_i^1 'right-to-left').

can be done as $\varphi(0)$ is satisfied by the assignment). Then s goes back to v_{top} through v_{mid} and starts over again, this time following the truth values assigned to variables in $X^1 \cup X^2$, and so on. One can verify that s is a solution to $\mathcal{I}$.

Now consider the other direction. Let

$$s = (v_{mid} s_1 v_{mid} \dots v_{mid} s_p)^\omega$$

be a solution to $\mathcal{I}$ where each *segment* s_j, $j \in \{1, \dots, p\}$ is a finite subpath visiting only targets in $V \setminus \{v_{mid}\}$. We further assume that s satisfies the first case of the following proposition (this is sound as s can be 'reversed' while remaining a valid solution). Let $s_{j-1} = s_p$ if $j = 1$ and $s_{j+1} = s_1$ if $j = p$.

Proposition 7. *In $s = (v_{mid} s_1 v_{mid} \dots v_{mid} s_p)^\omega$, either of the following holds:*

- *Each s_j, $j \in \{1, \dots, p\}$ starts with v_{top} and ends with v_{bot}*
- *Each s_j, $j \in \{1, \dots, p\}$ starts with v_{bot} and ends with v_{top}.*

To prove this proposition, we first establish several simple lemmas and propositions.

Lemma 1. *Each segment s_j must start with and end with v_{top} or v_{bot}.*

Lemma 2. *The time needed from v_{top} or v_{bot} to any other target is at least $(3m + 1)l$.*

Lemma 3. *The time needed from v_{mid} to any other target is at least $\frac{1}{4}T$.*

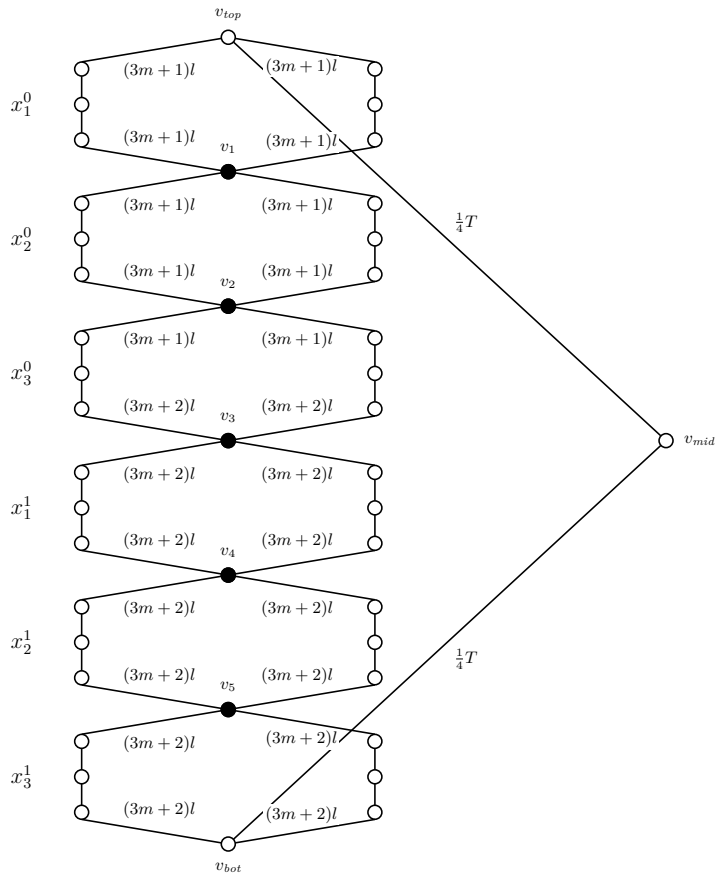

Figure 17: An example with $m = 3$. Solid circles denote the shared targets $S = \{v_1, \ldots, v_5\}$.

Lemma 4. *Each segment s_j must contain more than one target.*

Proof. By Lemma 1, without loss of generality let $s_j = v_{bot}$, a single target. It is easy to see that s_{j-1} must end with v_{top} and s_{j+1} must start with v_{top}, otherwise the relative deadline of v_{top} will be violated. Now consider v_1 (with $RD[v_1] = T + 2h$). By Lemma 2 and the fact that $dur(v_{top}v_{mid}v_{bot}v_{mid}v_{top}) = T$, the relative deadline of v_1 is violated for sure even if s visits v_1 immediately after v_{top}. This is a contradiction. □

Proposition 8. *For each segment s_j, $0 < dur(s_j) \leq \frac{1}{2}T$.*

Proof. By Lemma 4 we have $dur(s_j) > 0$. For the upper bound, note that $dur(v_{mid}s_jv_{mid}) = \frac{1}{2}T + dur(s_j)$ and $RD[v_{mid}] = T$. □

Proposition 9. *Each segment s_j contains all targets in $V \setminus \{v_{mid}\}$ with relative deadlines less or equal than $T + l + 2h$.*

Proof. Let $v \in V \setminus \{v_{mid}\}$ be a target missing in s_j with $RD[v] \leq T + l + 2h$. By Lemmas 1, 2 and 4, $dur(s_j) \geq 2(3m+1)l > l + l > l + 2h$. We have $dur(v_{mid}s_jv_{mid}) = \frac{1}{2}T + dur(s_j) > \frac{1}{2}T + l + 2h$. By Lemma 3, $dur(vv_{mid}s_jv_{mid}v)$ must be greater than $T + l + 2h$ for any $v \in V \setminus \{v_{mid}\}$, which is a contradiction. □

By Lemma 1 and Proposition 9, we first derive a (crude) lower bound on $dur(s_j)$. The sum of the minimum times needed to enter *and* leave every $v \in S$ and the minimum times needed to enter *or* leave the two ends of s_j (when both are v_{top}) gives

$$dur(s_j) \geq (m-1)\big(2(3m+1)l\big) + m\big(2(3m+2)l\big) + 2(3m+1)l\,. \tag{1}$$

Proposition 10. *v_{top}, v_{bot} and each $v \in S$ appears exactly once in each segment s_j.*

Proof. Without loss of generality, assume one of these targets appears more than once in s_j. By a similar argument as above, we derive that $dur(s_j)$ is at least $(m-1)\big(2(3m+1)l\big) + m\big(2(3m+2)l\big) + 2(3m+1)l + 2(3m+1)l > \frac{1}{2}T$. This contradicts Proposition 8. □

Proof (of Proposition 7). By Lemma 1 and Proposition 10, s_j must start and end with different targets from $\{v_{top}, v_{bot}\}$. Therefore, we can revise our lower bound in Eq.(1) and obtain

$$dur(s_j) \geq (m-1)\big(2(3m+1)l\big) + m\big(2(3m+2)l\big) + (3m+1)l + (3m+2)l\,. \tag{2}$$

Now without loss of generality let s_j ends with v_{top} and s_{j+1} starts with v_{top}. By Eq.(2), $dur(s_j) + dur(s_{j+1}) \geq 2\Big((m-1)\big(2(3m+1)l\big)+m\big(2(3m+2)l\big)+(3m+1)l+(3m+2)l\Big) > \frac{1}{2}T$, and hence $dur(s_j v_{mid} s_{j+1}) > T$. By Lemma 1 and Proposition 10, v_{bot} can only appear at both ends of $s_j v_{mid} s_{j+1}$, hence its relative deadline must be violated. This is a contradiction. □

We now argue that s 'witnesses' a satisfying assignment to $\bigwedge_{j\geq 0}\varphi(j)$.

Lemma 5. *In each segment s_j, each target in $\bigcup_{i\in\{1,\dots,m\}}\{pvt_i^L, pvt_i^R\}$ appears twice whereas other targets in $V \setminus \{v_{mid}\}$ appear once.*

To prove this lemma, we first refine our lower bound in Eq.(2) by taking into account other targets in variable gadgets and consistency gadgets with relative deadline less or equal to $T+l+2h$ (by Proposition 9). As many of these targets are adjacent, we only accumulate the minimum times needed to enter them:

- h clause boxes and $h+1$ separator boxes in each of the $2m$ rows: $2 \cdot 6 \cdot (2h+1) \cdot 2m$.
- pvt_i^L and pvt_i^R, $i \in \{1,\dots,m\}$: $2 \cdot 2 \cdot m$.
- LS_i and RS_i, $i \in \{1,\dots,2m\}$: Note that in the calculation of Eq.(2), one of the two targets connected to v_{top} and one of the four targets connected to each $v \in S$ 'have already been entered' and cannot be included in the current calculation. This gives $2 \cdot 6 \cdot 2m - 2 - 2 \cdot (2m-1) = 20m$.

In total, we now have

$$dur(s_j) \geq \frac{1}{2}T - 20m - 2h\,. \tag{3}$$

Proposition 11. *Each segment s_j contains all targets with relative deadlines equal to $\frac{3}{2}T$, i.e., clause targets and targets in $\bigcup_{i\in\{1,\dots,m\}}\big((LCG_i \setminus \{pvt_i^L\}) \cup (RCG_i \setminus \{pvt_i^R\})\big)$.*

Proof. Assume that there is such a target v not appearing in s_j. By Eq.(3), we have $dur(v_{bot} v_{mid} s_j v_{mid} v_{top}) \geq \frac{3}{2}T - 20m - 2h$. By Lemma 2, the relative deadline of v must be violated as $dur(v v_{bot} v_{mid} s_j v_{mid} v_{top} v) \geq \frac{3}{2}T - 20m - 2h + 2(3m+1)l > \frac{3}{2}T$. This is a contradiction. □

Based on the previous proposition, we can further refine our lower bound on the duration of a segment. The minimum times needed to enter

- clause targets v^{c_j}, $j \in \{1,\dots,h\}$
- targets in $\bigcup_{i\in\{1,\dots,m\}}\big((LCG_i \setminus \{pvt_i^L\}) \cup (RCG_i \setminus \{pvt_i^R\})\big)$

can now be included in the calculation. We have

$$dur(s_j) \geq \frac{1}{2}T - 4h\,. \tag{4}$$

Proposition 12. *In each segment s_j, each target in $\bigcup_{i\in\{1,\dots,m\}}\{pvt_i^L, pvt_i^R\}$ appears more than once.*

Proof. Let there be such a target v appearing only once in a segment. By Lemma 2, there are two occurrences of v in s separated by at least $\frac{1}{2} \cdot \left(\frac{1}{2}T + (\frac{1}{2}T - 4h) + \frac{1}{2}T\right) + (3m+1)l$. This exceeds all possible values of $RD[v]$. □

Proof (of Lemma 5). By Proposition 12, we assume that each target in $\bigcup_{i\in\{1,\ldots,m\}}\{pvt_i^L, pvt_i^R\}$ appears twice in a segment. Counting each such target once again gives an extra time of $4h$. The sum of this with Eq.(4) matches the upper bound in Proposition 8. Any more visit to a target in $V \setminus \{v_{mid}, v_{top}, v_{bot}, v_1, \ldots, v_{2m-1}\}$ will immediately contradict Proposition 8. □

Based on Lemma 5, we show that s cannot 'jump' between variable gadgets via clause targets. It follows that the traversal of each Row_i must be done in a single pass.

Proposition 13. *In each segment s_j, if v^{c_k} is entered from a clause box (in some variable gadget), the edge that immediately follows must go back to the same clause box.*

Proof. Consider a 3×3 'box' formed by a separator box and (the left- or right-) half of a clause box. Note that except for the four targets at the corners, no target in this 3×3 box is connected to the rest of the graph. Recall from [19] that if each target in this 3×3 box is to be visited only once (as enforced by Lemma 5), it must be traversed in the patterns illustrated in Figs. 18 and 19.

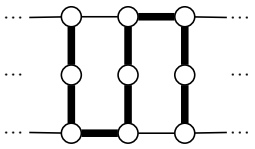

Figure 18: Pattern '⊔⊓'.

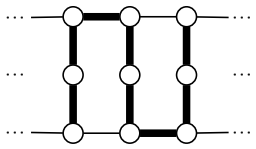

Figure 19: Pattern '⊓⊔'.

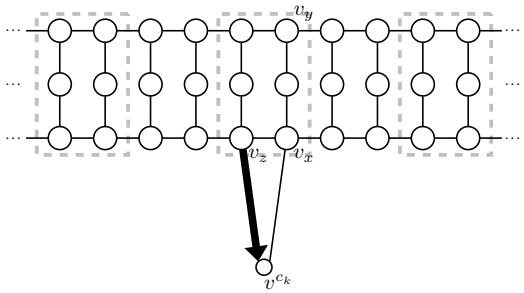

Figure 20: x_i^0 occurs positively in c_k.

Now consider for example the situation in Fig. 20 where s_j goes from v_z to v^{c_k}. The 3×3 box with v_z at its lower-right must be traversed in Pattern '⊔⊓' (as otherwise v_z will be visited twice). Suppose to the contrary that s_j does not visit v_x immediately after v^{c_k}. As v_x cannot be entered or left via v_z and v^{c_k}, the 3×3 box with v_x at its lower-left must also be traversed in Pattern '⊔⊓'. However, there is then no way to enter or leave v_y. This is a contradiction. □

Note that in Fig. 20, the three clause boxes (framed by dashed lines) are all traversed in Pattern '⊓' or they are all traversed in Pattern '⊔'. More generally, we have the following proposition.

Proposition 14. *In each segment s_j, clause boxes in a given variable gadget are all traversed in Pattern '⊓' or they are all traversed in Pattern '⊔' (with possible detours via clause targets).*

Write $v \to v'$ for the edge from v to v' and $v \rightsquigarrow v'$ for a finite path that starts with v and ends with v'. By Lemma 5, each segment s_j can be written as $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$ where $b_1, \ldots, b_{2m-1}$ is a permutation of $1, \ldots, 2m-1$. We show that each subpath $v \rightsquigarrow v'$ of s_j with distinct $v, v' \in S \cup \{v_{top}, v_{bot}\}$ and no $v'' \in S \cup \{v_{top}, v_{bot}\}$ in between must be of a very restricted form. For convenience, we call such a subpath $v \rightsquigarrow v'$ a *fragment*.

Proposition 15. *In each segment $s_j = v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$, a fragment $v \rightsquigarrow v'$ visits pvt_i^L and pvt_i^R (once for each) for some $i \in \{1, \ldots, m\}$. Moreover, each fragment $v \rightsquigarrow v'$ in $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_m}$ visits a different set $\{pvt_i^L, pvt_i^R\}$. The same holds for $v_{b_m} \rightsquigarrow v_{b_{m+1}} \rightsquigarrow \cdots \rightsquigarrow v_{bot}$.*

Proof. From Fig. 17, it is clear that $dur(v \rightsquigarrow v') \geq 2(3m+1)l$, and hence $dur(v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots v_{b_m}) \geq m\big(2(3m+1)l\big)$. Let there be a target $v \in \bigcup_{i \in \{1,\ldots,m\}} \{pvt_i^L, pvt_i^R\}$ missing in $v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots v_{b_m}$. Since the time needed from v_{b_m} to v is greater than $(3m+1)l$, even if s_j visits v as soon as possible after v_{b_m}, the duration from v_{bot} in s_{j-1} to v in s_j will still be greater than $\frac{1}{2}T + m\big(2(3m+1)l\big) + (3m+1)l > RD[v]$, which is a contradiction. Therefore, all targets in $\bigcup_{i \in \{1,\ldots,m\}} \{pvt_i^L, pvt_i^R\}$ must appear in the subpath from v_{top} to v_{b_m}. The same holds for the subpath from v_{b_m} to v_{bot} by similar arguments. Now note that by Proposition 13 and Fig. 16, a fragment $v \rightsquigarrow v'$ may visit, among $\bigcup_{i \in \{1,\ldots,m\}} \{pvt_i^L, pvt_i^R\}$, at most two targets—$\{pvt_i^L, pvt_i^R\}$ for some $i \in \{1, \ldots, m\}$. The proposition then follows from Lemma 5. □

Proposition 16. *In each segment s_j, a fragment $v \rightsquigarrow v'$ visits all targets in either Row_i or Row_{i+m} for some $i \in \{1, \ldots, m\}$ but not a single target in $\bigcup_{j \in \{1,\ldots,m\}}^{j \neq i} (Row_j \cup Row_{j+m})$.*

Now consider a fragment $v \rightsquigarrow v'$ that visits pvt_i^L and pvt_i^R (by Proposition 15). By Lemma 5, $v \rightsquigarrow v'$ must also visit exactly two targets other than pvt_i^L in LCG_i and exactly two targets other than pvt_i^R in RCG_i (once for each). By the argument above, $v \rightsquigarrow v'$ must contain, in order, the following subpaths (together with some obvious choices of edges connecting these subpaths):

(i). A long edge, e.g., $v_i \to v_i^{b,R}$.
(ii). A 'side', e.g., $v_i^{b,R} \to v_i^{m,R} \to v_i^{t,R}$.
(iii). A subpath consisting of a *pvt* target and two other targets in the relevant consistency gadget, e.g., $out_i^{\uparrow,R} \to pvt_i^R \to in_i^{\downarrow,R}$.
(iv). A traversal of a row with detours.
(v). A subpath consisting of a *pvt* target and two other targets in the relevant consistency gadget.
(vi). A side.
(vii). A long edge.

The following proposition is then immediate. In particular, the exact value of $dur(v \rightsquigarrow v')$ is decided by:

- flight times of the long edges taken in (i) and (vii)
- detours to clause targets in (iv).

Proposition 17. *In each segment s_j, the following holds for all fragments $v \rightsquigarrow v'$:*

$$2(3m+1)l + l \leq dur(v \rightsquigarrow v') \leq 2(3m+2)l + l + 2h.$$

Proposition 18. *The order the sets $\{pvt_i^L, pvt_i^R\}$ are visited (regardless of which target in the set is first visited) in the first m fragments of each segment s_j is identical to the order they are visited in the last m fragments of s_{j-1}.*

Proof. By Proposition 17, if this does not hold then there must be a *pvt* target having two occurrences in s separated by more than $\frac{1}{2}T + m\big(2(3m+1)l + l\big) + 2(3m+1)l$. This is a contradiction. □

For each segment s_j, we denote by $first(s_j)$ the 'first half' of s_j, i.e., the subpath of s_j that consists of the first m fragments of s_j and by $second(s_j)$ the 'second half' of s_j. Write $\exists(v \rightsquigarrow v') \subseteq u$ if u has a subpath of the form $v \rightsquigarrow v'$.

Proposition 19. *In each segment $s_j = v_{top} \rightsquigarrow v_{b_1} \rightsquigarrow \cdots \rightsquigarrow v_{b_{2m-1}} \rightsquigarrow v_{bot}$, we have $b_i = i$ for all $i \in \{1, \ldots, 2m-1\}$.*

Proof. First note that by construction and Proposition 15, $\{pvt_m^L, pvt_m^R\}$ must be the last set of *pvt* targets visited in $second(s_{j-1})$. By Proposition 18, it must also be the last set of *pvt* targets visited in $first(s_j)$. Now assume that a long edge of flight time $(3m+2)l$ is taken before pvt_m^L and pvt_m^R are visited in $first(s_j)$. Consider the following cases:

- $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq second(s_{j-1})$ and $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq first(s_j)$: Note that the last edge taken in s_{j-1} is a long edge of flight time $(3m+2)l$, and hence there are two occurrences of pvt_m^L in s separated by at least $\frac{1}{2}T + m\big(2(3m+1)l + l\big) + 2l > \frac{1}{2}T + m\big(2(3m+1)l + l\big) + l + 4h = RD[pvt_m^L]$.
- $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq second(s_{j-1})$ and $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq first(s_j)$: The same argument shows that pvt_m^R must miss its relative deadline.
- $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq second(s_{j-1})$ and $\exists(pvt_m^L \rightsquigarrow pvt_m^R) \subseteq first(s_j)$: The same argument shows that both pvt_m^L and pvt_m^R must miss their relative deadlines.
- $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq second(s_{j-1})$ and $\exists(pvt_m^R \rightsquigarrow pvt_m^L) \subseteq first(s_j)$: The same argument shows that both pvt_m^L and pvt_m^R must miss their relative deadlines.

We therefore conclude that in $first(s_j)$, all long edges taken before pvt_m^L and pvt_m^R are visited must have flight time equal to $(3m+1)l$. Furthermore, all such long edges must be traversed 'downwards' (by Lemma 5). It follows that $b_i = i$ for $i \in \{1, \ldots, m-1\}$. By Proposition 18, Lemma 5 and $m > 2$, we easily derive that $b_m = m$ and then $b_i = i$ for $i \in \{m+1, \ldots, 2m-1\}$. □

By Proposition 19, the long edges in each variable gadget must be traversed in the ways shown in Figs. 21 and 22.

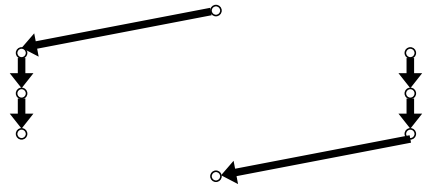

Figure 21: The variable is assigned to **true**.

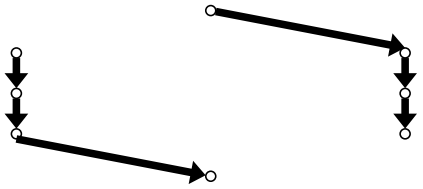

Figure 22: The variable is assigned to **false**.

Proposition 20. *For each segment s_j, the ways in which the long edges are traversed in the last m fragments of s_j are consistent with the ways in which the long edges are traversed in the first m fragments of s_{j+1}.*

Proof. Without loss of generality, consider the case that $\exists(pvt_i^L \rightsquigarrow pvt_i^R) \subseteq second(s_j)$ and $\exists(pvt_i^R \rightsquigarrow pvt_i^L) \subseteq first(s_{j+1})$. By Proposition 19, these two occurrences of pvt_i^L in s are separated by, at least, the sum of $\frac{1}{2}T + m\big(2(3m+2)l + l\big) - (2i-1)l$ and the duration of the actual subpath $pvt_i^R \rightsquigarrow pvt_i^L$ in $first(s_{j+1})$. It is clear that pvt_i^L must miss its relative deadline. □

Proposition 21. *In each segment s_j, if a variable gadget is traversed as in Fig. 21 (Fig. 22), then all of its clause boxes are traversed in Pattern '⊔' (Pattern '⊓').*

Proof (of Proposition 6). Consider a segment s_j. As each clause target is visited once in s_j (by Lemma 5), the ways in which the long edges are traversed in all fragments $v \rightsquigarrow v'$ of s_j (i.e., as in Fig. 21 or Fig. 22) can be seen as a satisfying assignment to $\varphi(0)$ (by construction and Proposition 21). By the same argument, the ways in which the long edges are traversed in all fragments of s_{j+1} can be seen as a satisfying assignment to $\varphi(1)$. Now by Proposition 20, the assignment to variables X^1 is consistent in both segments. The same argument can again be applied to s_{j+2} which itself can be seen as a satisfying assignment to $\varphi(2)$ where the assignment to variables X^2 is consistent with s_{j+1}, and so on. It follows that s witnesses a (periodic) satisfying assignment to $\bigwedge_{j \geq 0} \varphi(j)$. □

The main result of this section follows immediately from PSPACE-membership of the problem (Proposition 24).

Theorem 1. *The decision version of the* CR-UAV *problem is* PSPACE*-complete.*[6]

4. Lower and upper bounds on the number of UAVs

Given a CR-UAV instance $\mathcal{I} = \langle RD, FT \rangle$, the least number $k_{\mathcal{I}}$ of UAVs required in a solution to $\mathcal{I}$ can be computed by solving the decision version of the CR-UAV problem for increasing values of $k \in \{1, \ldots, |V|\}$. In this section, we improve this naïve algorithm by giving better lower and upper bounds on $k_{\mathcal{I}}$.

4.1. A lower bound on the number of UAVs

In what follows, let

$$FT_{min}(v) = \min_{\substack{v' \in V \\ v' \neq v}} \{FT[v, v']\}\,.$$

In words, $FT_{min}(v)$ denotes the minimal FT on any outgoing edge of v. We say a target $v \in V$ is ***isolated*** iff $RD[v] \leq FT_{min}(v)$.

Example 5. Consider the CR-UAV instance $\mathcal{I}$ depicted in Fig. 23. The only isolated target is v_0. A solution with three UAVs appears in Fig. 24. We will soon show that this number matches the lower bound.

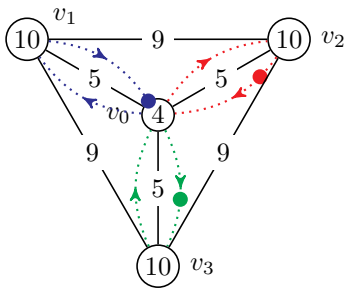

Figure 23: $\mathcal{I}$ in Example 5.

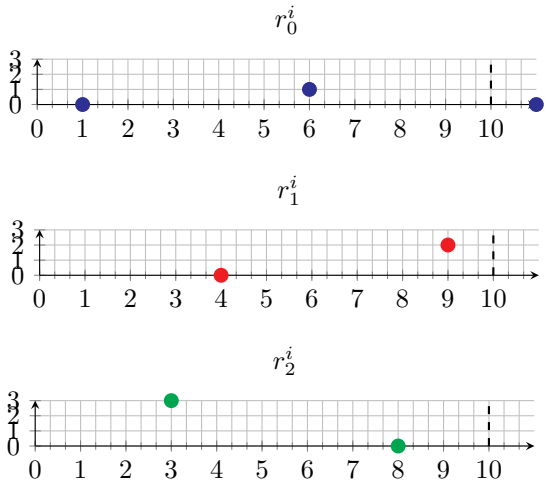

Figure 24: A solution $S^i = \{r_0^i, r_1^i, r_2^i\}$ to $\mathcal{I}$ in Example 5.

Proposition 22. *For $\mathcal{I} = \langle RD, FT \rangle$, a lower bound on $k_{\mathcal{I}}$ is given by*

$$|W| + \left\lceil \sum_{v \in (V \setminus W)} \frac{FT_{min}(v)}{RD[v]} \right\rceil \leq k_{\mathcal{I}}$$

where $W \subseteq V$ is the set of isolated targets.

[6] Our result holds irrespective of whether the numbers are encoded in unary or binary.

Proof. Let S be a periodic instantaneous integer-timed solution to $\mathcal{I}$ with a period P. Let $T_{sl}(v) \leq P$ be the total time S spent at a target v on self-loops in $(0, P]$. The number of UAV entries to v during $(0, P]$ must be at least

$$\left\lceil \frac{P - T_{sl}(v)}{RD[v]} \right\rceil ,$$

and hence the total flight time dedicated to v must be at least

$$\left\lceil \frac{P - T_{sl}(v)}{RD[v]} \cdot FT_{min}(v) \right\rceil + T_{sl}(v) \,. \tag{5}$$

The overall flight time is given by aggregating Eq.(5) over V:

$$\sum_{v \in V} \left(\left\lceil \frac{P - T_{sl}(v)}{RD[v]} \cdot FT_{min}(v) \right\rceil + T_{sl}(v) \right) .$$

This term must not exceed the total flight time of all UAVs during T, which is given by $P \cdot k_{\mathcal{I}}$:

$$\sum_{v \in V} \left(\left\lceil \frac{P - T_{sl}(v)}{RD[v]} \cdot FT_{min}(v) \right\rceil + T_{sl}(v) \right) \leq P \cdot k_{\mathcal{I}} \,.$$

We now separate the elements in the sum on the left according to whether $v \in W$:

$$\begin{aligned} &\sum_{v \in W} \left(\left\lceil \frac{P - T_{sl}(v)}{RD[v]} \cdot FT_{min}(v) \right\rceil + T_{sl}(v) \right) + \\ &\sum_{v \in (V \setminus W)} \left(\left\lceil \frac{P - T_{sl}(v)}{RD[v]} \cdot FT_{min}(v) \right\rceil + T_{sl}(v) \right) \leq P \cdot k_{\mathcal{I}} \,. \end{aligned} \tag{6}$$

Let us focus on the first summation: since this expression is monotone in $T_{sl}(v)$ (when the other variables are fixed) and $0 \leq T_{sl}(v) \leq P$, its value is in the range

$$\left[\sum_{v \in W} P, \; \sum_{v \in W} \left\lceil \frac{P}{RD[v]} \right\rceil \cdot FT_{min}(v) \right]$$

(recall that for each $v \in W$ we have $RD[v] \leq FT_{min}(v)$). Thus, the first sum in Eq.(6) can be lowered to $P \cdot |W|$, which gives us

$$P \cdot |W| + \sum_{v \in (V \setminus W)} \left(\left\lceil \frac{P - T_{sl}(v)}{RD[v]} \cdot FT_{min}(v) \right\rceil + T_{sl}(v) \right) \leq P \cdot k_{\mathcal{I}} \,. \tag{7}$$

Furthermore, the second summation is larger than

$$\sum_{v \in (V \setminus W)} \left(\frac{P - T_{sl}(v)}{RD[v]} \cdot FT_{min}(v) + T_{sl}(v) \right)$$

(by removing the ceiling operator), which can be rewritten into

$$\sum_{v \in (V \setminus W)} \left(\frac{P \cdot FT_{min}(v)}{RD[v]} + T_{sl}(v) \cdot \left(1 - \frac{FT_{min}(v)}{RD[v]} \right) \right) . \tag{8}$$

Note that by the definition of isolated targets, for every $v \in (V \setminus W)$ it holds that $\frac{FT_{min}(v)}{RD[v]} \leq 1$, which implies that the right operand is positive and consequently Eq.(8) is larger than

$$\sum_{v \in (V \setminus W)} \frac{P \cdot FT_{min}(v)}{RD[v]} \,.$$

Hence, based on Eq.(7) we have that

$$P \cdot |W| + \sum_{v \in (V \setminus W)} \frac{P \cdot FT_{min}(v)}{RD[v]} \leq P \cdot k_{\mathcal{I}} \,.$$

Dividing by P and rounding up concludes the proof. □

The bound is tight as witnessed by $\mathcal{I}$ in Example 5: by the proposition above, a lower bound on $k_{\mathcal{I}}$ is given by $1 + \lceil (\frac{5}{10} + \frac{5}{10} + \frac{5}{10}) \rceil = 3$; this matches the actual value of $k_{\mathcal{I}}$ (a solution to $\mathcal{I}$ with three UAVs—the least possible—is given in Fig. 24).

Remark 1. Proposition 22 may tempt the reader to think that for each $\mathcal{I}$ with some isolated targets, there must be a solution with $k_{\mathcal{I}}$ UAVs where a UAV is dedicated to each isolated target. This, however, is not true. To see this, observe that for $\mathcal{I}$ in Example 5 ($k_{\mathcal{I}} = 3$), dedicating a UAV to v_0 would force us to dedicate a UAV for each of the other three targets, hence using four UAVs altogether.

4.2. An upper bound on the number of UAVs

Proposition 23. *For $\mathcal{I} = \langle RD, FT \rangle$, an upper bound on $k_{\mathcal{I}}$ is given by*

$$k_{\mathcal{I}} \leq \min\{|V|, \left\lceil \frac{TSP(\mathcal{I})}{RD_{min}} \right\rceil\} \tag{9}$$

where $TSP(\mathcal{I})$ is the length of the tour when $\mathcal{I}$ is regarded as an instance of the Travelling Salesman Problem and RD_{min} is the minimal RD in $\mathcal{I}$.

Proof. If $|V| < \left\lceil \frac{TSP(\mathcal{I})}{RD_{min}} \right\rceil$, dedicate a UAV to each target; otherwise let all the UAVs follow the TSP tour and be evenly spaced (see Fig. 10 for an example). In both cases one obtains a solution to $\mathcal{I}$. □

While an upper bound which in itself may take exponential time to solve seems not very useful, we note that (i) in our application domain there is often a relatively small number of targets to begin with, and (ii) the famous result by Christofides [20] gives us a tour within a factor of 1.5 of the optimal tour in polynomial time, as long as the problem is defined over a metric. On the other hand, the bound is tight as witnessed by $\mathcal{J}$ in Example 3: by the proposition above, an upper bound on $k_{\mathcal{J}}$ is given by $\min\{4, \lceil \frac{8}{3} \rceil\} = 3$; this matches the actual value of $k_{\mathcal{J}}$.

5. Solving the cr-uav problem via a reduction to LTL model checking

In this section, we give a reduction from the decision version of the CR-UAV problem to LTL model checking on symbolic transition systems. For a CR-UAV instance $\mathcal{I}$ and a positive integer k (assuming $k < n$), we construct a symbolic transition system $\mathcal{A}_{\mathcal{I},k}$ and an LTL formula φ such that a solution to $\mathcal{I}$ with k UAVs corresponds directly to a *loop* in $\mathcal{A}_{\mathcal{I},k}$ that violates φ, i.e., $\mathcal{I}$ has a solution with k UAVs iff there exists such a loop in $\mathcal{A}_{\mathcal{I},k}$.[7] In practice, we can use a model checker to find a *lasso* (i.e., an infinite path of the form su^ω where s, u are non-empty sequences of states) in $\mathcal{A}_{\mathcal{I},k}$ that violates φ, from which we can easily extract such a loop (u^ω) and obtain the corresponding solution by observing the states. The following proposition follows from [17]. In particular, (the optimisation version of) the CR-UAV problem can be solved by a binary search on k (between the lower and upper bounds given in the previous section).

Proposition 24. *The decision version of the* CR-UAV *problem is in* PSPACE.

[7] The state space of $\mathcal{A}_{\mathcal{I},k}$ is exponential (in the size of $\mathcal{I}$) yet it is described symbolically, i.e., as Boolean formulas over state variables, which keeps the representation polynomial.

5.1. Symbolic transition systems

We first make explicit what we mean by *symbolic transition systems*. For notational simplicity, we will use Boolean variables in the definitions below; it is straightforward to extend these definitions to accommodate bounded-width integer variables and arithmetic operations on them (they can be broken up into individual bits by model checkers).

Definition 4 (Symbolic transition systems)**.** A (finite-state) *symbolic transition system* $\mathcal{T}$ is a tuple $\langle X, \mathit{Init}, R\rangle$ where

- X is a finite set of Boolean variables
- *Init* (the *initial predicate*) is a Boolean formula over X
- R (the *transition relation*) is a Boolean formula over $X \cup X'$ where $X' = \{x' \mid x \in X\}$.

A *state* q of $\mathcal{T}$ is a valuation of X, i.e., $q : X \mapsto \{\mathbf{true}, \mathbf{false}\}$. We write q' for the valuation of X' such that $q'(x') = q(x)$ for each $x \in X$. A *transition* from q_0 to q_1 (written $q_0 \to q_1$) is in $\mathcal{T}$ iff $R[q_0/X, q_1'/X']$ (the Boolean formula obtained from R by substituting all $x \in X$ with $q_0(x)$ and all $x' \in X'$ with $q_1'(x')$) holds. An *infinite path* in $\mathcal{T}$ is an infinite sequence $q_0 \to q_1 \to \cdots$ of transitions in $\mathcal{T}$ such that $\mathit{Init}(q_0)$ holds.

5.2. The symbolic transition system $\mathcal{A}_{\mathcal{I},k}$

Let $\mathcal{I} = \langle RD, FT\rangle$ be the given CR-UAV instance and k be the presumed number of UAVs. We now detail the construction of $\mathcal{A}_{\mathcal{I},k}$. The essence of the construction is to record the information associated with each prefix in $\mathsf{prefix}_{\mathcal{I},k}$ (defined in the proof of Proposition 4) in the state variables.

State variables. Each state q of $\mathcal{A}_{\mathcal{I},k}$ is a tuple

$$\langle TC, UC, \mathit{latest}, \mathit{stop}\rangle :$$

- TC is an array of size n such that for each $v \in V$, $TC[v]$ (the 'target clock' for v) is the time elapsed since the last scan of v ($0 \le TC[v] < RD[v]$).
- UC is an array of size k such that for each $i \in \{0, \ldots, k-1\}$, $UC[i]$ (the 'UAV clock' for the i-th UAV) is the time elapsed since the i-th UAV scanned some target ($0 \le UC[i] < FT_{max}$ where FT_{max} is the maximal flight time in FT).
- *latest* is an array of size k such that for each $i \in \{0, \ldots, k-1\}$, $\mathit{latest}[i] \in V$ is the last target scanned by the i-th UAV.
- *stop* is a Boolean variable that is set when the UAVs cease to proceed.

The initial predicate is $\neg \mathit{stop}$.

Transitions. The transition

$$\langle TC_0, UC_0, \mathit{latest}_0, \mathbf{false}\rangle \to \langle TC_1, UC_1, \mathit{latest}_1, \mathbf{false}\rangle$$

is in $\mathcal{A}_{\mathcal{I},k}$ iff there is a mapping from $\{0, \ldots, k-1\}$ to V (which we represent as an array '*heading*') and the state variables satisfy the following (note in particular that $TC_1[v] < RD[v]$, $UC_1[i] < FT_{max}$):

- $\mathit{step} = \min_{i \in \{0,\ldots,k-1\}} \{FT[\mathit{latest}_0[i], \mathit{heading}[i]] - UC_0[i]\} > 0$.
- For all $v \in V$, $TC_1[v] = \begin{cases} 0 & \text{if } \mathit{heading}[i] = v,\ UC_0[i] + \mathit{step} = FT[\mathit{latest}_0[i], v] \\ & \text{for some } i \in \{0, \ldots, k-1\} \text{ and } TC_0[v] + \mathit{step} \le RD[v] \\ TC_0[v] + \mathit{step} & \text{otherwise.} \end{cases}$

- For all $i \in \{0, \ldots, k-1\}$, $UC_1[i] = \begin{cases} 0 & \text{if } UC_0[i] + step = FT[latest_0[i], heading[i]] \\ UC_0[i] + step & \text{otherwise.} \end{cases}$
- For all $i \in \{0, \ldots, k-1\}$, $latest_1[i] = \begin{cases} heading[i] & \text{if } UC_0[i] + step = FT[latest_0[i], heading[i]] \\ latest_0[i] & \text{otherwise.} \end{cases}$

Intuitively, $heading[i]$ is the presumed next target of the i-th UAV and $step$ is the least time required for some UAV to reach its presumed target. In addition to these 'normal' transitions, we also have

$$\langle TC_0, UC_0, latest_0, stop \rangle \to \langle TC_0, UC_0, latest_0, \mathbf{true} \rangle$$

for both values of $stop$; in other words, $\mathcal{A}_{\mathcal{I},k}$ can go into a 'stop' state at any point and loop there forever.

Example 6. Consider the CR-UAV instance $\mathcal{L} = \langle RD, FT \rangle$ where

$$RD = \begin{bmatrix} 10 & 5 & 10 \end{bmatrix} \qquad FT = \begin{bmatrix} 1 & 3 & 6 \\ 3 & 1 & 5 \\ 6 & 5 & 1 \end{bmatrix},$$

and $k = 2$. A transition in $\mathcal{A}_{\mathcal{L},2}$ is illustrated in Fig. 25 where the two states are labelled with $\langle TC, UC, latest \rangle$ (the values of $stop$ are assumed to be **false** and hence omitted for clarity) and the transition itself is labelled with the corresponding $\langle heading, step \rangle$. In this case, UAV 0 is heading towards target 1, UAV 1 is heading towards target 2; the value of $step$ is then $\min\{FT[0,1] - 0, FT[0,2] - 0\} = 3$. Since $FT[0,2] - 0 > 3$, UAV 1 will not reach target 2; in fact, since $heading$ is not part of the states, only the choices of the UAVs that actually reach their headed targets (in this case, UAV 0) are relevant. In the destination state, TC becomes $[3, 0, 9]$ as only target 1 is actually visited; UC becomes $[0, 3]$ and $latest$ becomes $[1, 0]$ as it is UAV 0 that visits target 1. Note that since the relative deadline of v_2 is bound to expire afterwards (no matter how the UAVs are moved), the only outgoing transition from $\langle [3, 0, 9], [0, 3], [1, 0] \rangle$ goes to a state where $stop$ is **true** (not depicted in the figure).

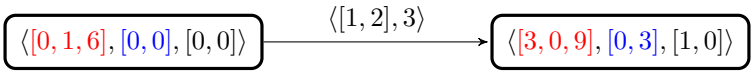

Figure 25: A transition in $\mathcal{A}_{\mathcal{L},2}$ in Example 6.

5.3. The LTL *specification* φ

It is clear that a solution to $\mathcal{I}$ with k UAVs can be mapped to an infinite path in $\mathcal{A}_{\mathcal{I},k}$ where $stop$ is always **false**. On the other hand, if $\mathcal{I}$ has no solution with k UAVs, one of the relative deadlines must be violated no matter how the UAVs are moved. As we enforce that $TC[v] < RD[v]$ for each $v \in V$ and $step > 0$ in the definition of $\mathcal{A}_{\mathcal{I},k}$, any infinite path must go into a stop state and stuck there at some point. We therefore let

$$\varphi = \mathsf{F}\, stop$$

(F is the LTL operator for 'in the future') which asserts the latter—the infinite path contains a state where $stop$ is **true**.[8]

Example 7. Consider the CR-UAV instance $\mathcal{J}$ in Example 3 (on page 7) and $k = 3$. A loop in $\mathcal{A}_{\mathcal{J},3}$ (where $stop$ is always **false**) is illustrated in Fig. 26. From this loop in $\mathcal{A}_{\mathcal{J},3}$, one can easily reconstruct the solution S^h (in Fig. 11) to $\mathcal{J}$ with three UAVs (note that the i-th UAV reaches a target whenever $UC[i] = 0$); the CR-UAV instance $\mathcal{J}$ and the corresponding configurations of UAVs are also depicted in the figure.

[8] We omit the full definition of LTL; interested readers are referred to, for instance [21], for details.

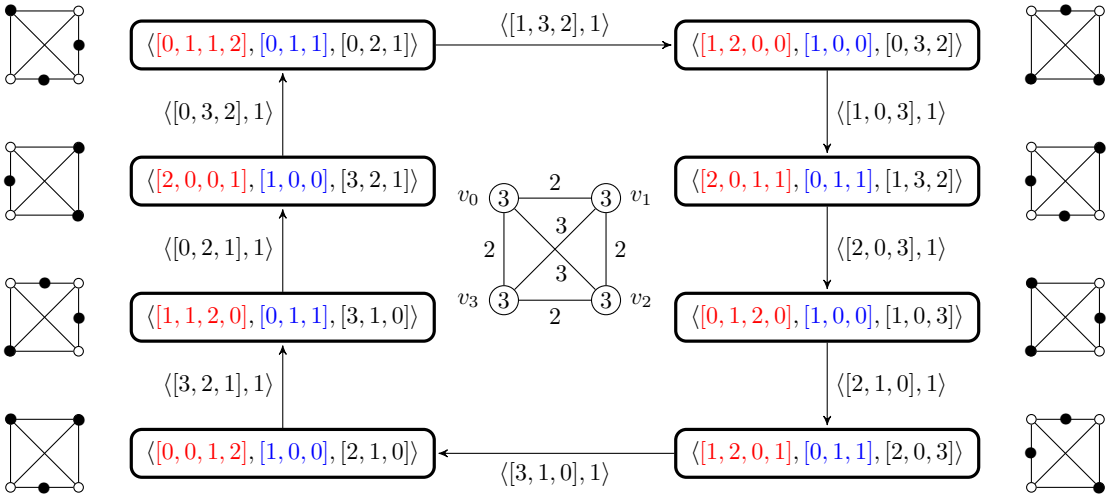

Figure 26: A loop in $\mathcal{A}_{\mathcal{J},3}$ in Example 7. The states are labelled with $\langle TC, UC, latest\rangle$ (the values of *stop* are assumed to be **false**) and the transitions are labelled with $\langle heading, step\rangle$. For example, in the top-left state, UAV 0 is at v_0 ($UC[0] = 0$ and $latest[0] = 0$) whereas UAV 1 has left v_2 for 1 time unit ($UC[1] = 1$ and $latest[1] = 2$).

6. An optimisation based on a simulation relation on states

As we mentioned earlier, the results in the last two sections can be combined with an LTL model checker to obtain a *complete* procedure for the CR-UAV problem. However, it turns out that a 'bounded model-checking' approach [22] is usually much faster in *finding solutions*: unroll the transition relation of $\mathcal{A}_{\mathcal{I},k}$ to a bounded depth d and use an SMT solver to check whether there is a loop for increasing values of d. This usually works very well since d tends to be small; in other words, an exponentially-sized d (cf. Proposition 5) is rarely needed in practice. Still, unless such a large d (which is almost always infeasible for current SMT solvers) is used, this approach cannot *prove the absence of solutions*: bounded model checking is logically *incomplete*. These observations suggest a simple approach to solving the CR-UAV problem faster: run a bounded model checker to search for a solution, and in parallel run an LTL model checker, which, recall, is complete and hence is able to prove the absence of solutions. One of the two tasks must eventually return an answer, by which time we can terminate both of them. This type of architecture is commonplace when performing model checking.

We continue by showing that the performance of both tasks can be improved by a *simulation relation*, which effectively reduces the state space to be considered. In Section 6.2 we will describe the bounded model-checking approach, and in Section 6.3 we will describe an explicit model-checking algorithm for solving our problem. In fact since we assume the parallel architecture mentioned above, we optimised our explicit model checker such that it is unable to find solutions, but can prove their absence faster.

6.1. Reducing the state space via a simulation relation

We first give a formal definition of simulation relations. Intuitively, we write $q_1 \trianglelefteq q_0$ if q_1 is capable of imitating all transitions from q_0 (q_1 is more 'promising' [23]).

Definition 5 (Simulation relation)**.** A preorder $\trianglelefteq$ over the set of states of $\mathcal{T}$ (denoted by $Q_{\mathcal{T}}$) is a *simulation relation* on $\mathcal{T} = \langle X, Init, R\rangle$ if for all $q_0, q_1, q_2 \in Q_{\mathcal{T}}$ such that $q_1 \trianglelefteq q_0$ and $R(q_0, q_2')$ holds, there exists $q_3 \in Q_{\mathcal{T}}$ such that $R(q_1, q_3')$ holds and $q_3 \trianglelefteq q_2$.

Now, for our purpose, let us consider a slightly modified model $\mathcal{A}^{np}_{\mathcal{I},k}$ ('np' stands for 'non-punctual') with no stop states. Each transition $\langle TC_0, UC_0, latest_0\rangle \to \langle TC_1, UC_1, latest_1\rangle$ in $\mathcal{A}^{np}_{\mathcal{I},k}$ can be seen as labelled with a triple $\langle turn, heading, step\rangle$ where $turn \in \{0, \ldots, k-1\}$, $heading$ is a mapping from $\{0, \ldots, k-1\}$ to V, and (RD_{min} is the minimal relative deadline in RD):

- $\max\{FT[latest_0[turn], heading[turn]] - UC_0[turn], 0\} \leq step \leq RD_{min}$.[9]
- For all $v \in V$, $TC_1[v] = \begin{cases} 0 & \text{if } heading[turn] = v \text{ and } TC_0[v] + step \leq RD[v] \\ TC_0[v] + step & \text{if } heading[turn] \neq v \text{ and } TC_0[v] + step \leq RD[v]. \end{cases}$
- For all $i \in \{0, \ldots, k-1\}$, $UC_1[i] = \begin{cases} 0 & \text{if } turn = i \\ UC_0[i] + step & \text{if } turn \neq i \text{ and } UC_0[i] + step \leq FT_{max} \\ FT_{max} & \text{if } turn \neq i \text{ and } UC_0[i] + step > FT_{max}. \end{cases}$
- For all $i \in \{0, \ldots, k-1\}$, $latest_1[i] = \begin{cases} heading[i] & \text{if } turn = i \\ latest_0[i] & \text{otherwise.} \end{cases}$

Observe that if two states have same $latest[i]$ for all $i \in \{0, \ldots, k-1\}$, then the state with smaller $TC[v]$ for all $v \in V$ and larger $UC[i]$ for all $i \in \{0, \ldots, k-1\}$ is more promising than the other. This leads to a simulation relation $\preceq$ on the states of $\mathcal{A}^{np}_{\mathcal{I},k}$.

Example 8. Consider the CR-UAV instance $\mathcal{L}$ in Example 6. The following diagram shows two states of $\mathcal{A}^{np}_{\mathcal{L},2}$, between which $\preceq$ holds, and below them their destination states. Note that $\preceq$ also hold between the two destination states.

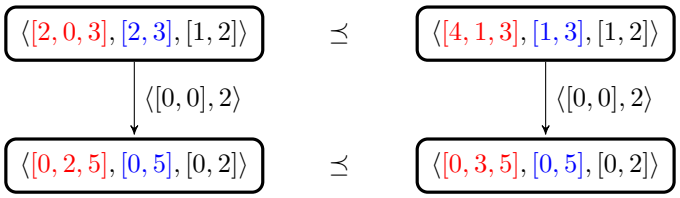

Figure 27: A pair of transitions in $\mathcal{A}_{\mathcal{L},2}$ in Example 6. The values of *stop* are omitted (all of which are **false**).

Formally, we define $\preceq$ as follows.

Definition 6. For states $q_0 = \langle TC_0, UC_0, latest_0 \rangle$ and $q_1 = \langle TC_1, UC_1, latest_1 \rangle$ of $\mathcal{A}^{np}_{\mathcal{I},k}$, let $q_1 \preceq q_0$ iff all of the following holds:

- For each $v \in V$, $TC_1[v] \leq TC_0[v]$.
- For each $i \in \{0, \ldots, k-1\}$, $UC_1[i] \geq UC_0[i]$.
- For each $i \in \{0, \ldots, k-1\}$, $latest_1[i] = latest_0[i]$.

6.2. Finding solutions with bounded model checking

A loop of length d in $\mathcal{T} = \langle X, Init, R \rangle$ is encoded by

$$\psi^d = Init(q_0) \wedge R(q_0, q'_1) \wedge R(q_1, q'_2) \ldots R(q_{d-2}, q'_{d-1}) \wedge R(q_{d-1}, q'_0)\,. \tag{10}$$

In other words, there exists a loop of length d in $\mathcal{T}$ iff ψ^d is satisfiable. In practice, since we will use an SMT solver to check the satisfiability of ψ^d, it is written directly in terms of bounded-width integer variables. We now show that this encoding can be improved with a simulation relation on $\mathcal{T}$.

[9] Note that *step* can be larger than the flight time required.

Proposition 25. *Let $\trianglelefteq$ be a simulation relation on $\mathcal{T} = \langle X, Init, R\rangle$. Let*

$$\psi^d_{sim} = Init(q_0) \wedge R(q_0, q'_1) \wedge R(q_1, q'_2) \dots R(q_{d-2}, q'_{d-1}) \wedge R(q_{d-1}, q'_d) \wedge q_d \trianglelefteq q_0 \,.^{10} \tag{11}$$

Then we have the following:

- *If ψ^d is satisfiable then ψ^d_{sim} is satisfiable.*
- *If ψ^d_{sim} is satisfiable then $\psi^{d'}$ is satisfiable for some $d' \geq 1$.*

Proof. The first item is trivial (take $q_d = q_0$). For the other, by definition, there exists $q_{d+1} \in Q_\mathcal{T}$ such that $R(q_d, q'_{d+1})$ and $q_{d+1} \trianglelefteq q_1$. Continuing this reasoning, we obtain an infinite path in $\mathcal{T}$. Since $Q_\mathcal{T}$ is finite, there must be a loop in this path. □

6.3. Proving the absence of solutions with explicit model checking

Let $\mathcal{T} = \langle X, Init, R\rangle$. The set of successors of $Q \subseteq Q_\mathcal{T}$ is denoted by $\mathsf{post}(Q) = \{q' \in Q_\mathcal{T} \mid \exists q \in Q_\mathcal{T}.\, R(q, q')\}$. Let $\mathsf{post}^0(Q) = Q$ and $\mathsf{post}^i(Q) = \mathsf{post}\big(\mathsf{post}^{i-1}(Q)\big)$ for $i \geq 1$. We denote by $\mathsf{post}^+(Q)$ the set $\bigcup_{i\geq 1} \mathsf{post}^i(Q)$. The question of whether there is an infinite path in $\mathcal{T}$ can be decided by an algorithm that computes the following sequence (where $\mathsf{Init} = \{q \in Q_\mathcal{T} \mid Init(q)\}$):

Sequence 0

- $\mathsf{FF}(0) = \mathsf{Init}$
- $\mathsf{FF}(i) = \mathsf{post}^+\big(\mathsf{FF}(i-1)\big)$ for all $i \geq 1$.

More specifically, the algorithm computes this sequence until a *fixed point* is reached, i.e., $\mathsf{FF}^* = \mathsf{FF}(i) = \mathsf{FF}(i+1)$ for some $i \geq 0$, and check if it is empty. This algorithm can be improved by exploiting a simulation relation $\trianglelefteq$ on $\mathcal{T}$. The idea is simple: for any set of states used in the algorithm, we can maintain its most promising states instead of the set itself. Denote the set of minimal elements of $Q \subseteq Q_\mathcal{T}$ by $\mathsf{Min}(\trianglelefteq, Q) = \{q \in Q \mid \forall q' \in Q.\, (q' \trianglelefteq q \implies q \trianglelefteq q')\}$. We write $Q_0 \sqsubseteq Q_1$ if $\forall q \in Q_0.\, \big(\exists q' \in Q_1.\, (q' \trianglelefteq q)\big)$ and $Q_0 \approx Q_1$ if $Q_0 \sqsubseteq Q_1$ and $Q_1 \sqsubseteq Q_0$. We can compute the following sequence instead:[11]

Sequence 1

- $\widehat{\mathsf{FF}}(0) = \mathsf{Min}\big(\trianglelefteq, \mathsf{Init}\big)$
- $\widehat{\mathsf{FF}}(i) = \mathsf{Min}\Big(\trianglelefteq, \mathsf{post}^+\big(\widehat{\mathsf{FF}}(i-1)\big)\Big)$ for all $i \geq 1$.

In particular, $\mathsf{Min}\Big(\trianglelefteq, \mathsf{post}^+\big(\widehat{\mathsf{FF}}(i-1)\big)\Big)$ can be computed as the fixed point of the following sequence:

- $\widehat{\mathsf{F}}(0) = \mathsf{Min}\Big(\trianglelefteq, \mathsf{post}\big(\widehat{\mathsf{FF}}(i-1)\big)\Big)$
- $\widehat{\mathsf{F}}(j) = \mathsf{Min}\Big(\trianglelefteq, \mathsf{post}\big(\widehat{\mathsf{F}}(j-1)\big) \cup \widehat{\mathsf{F}}(j-1)\Big)$ for all $j \geq 1$.

[10] We postulate that $q_1 \trianglelefteq q_0$ can be written as constraints on (variables comprising) q_1 and q_0; this is true for $\mathcal{A}_{\mathcal{I},k}$ and $\preceq$.
[11] In [23], this sequence is called *the forward repeated reachability sequence of promising states*.

The algorithm computes Sequence 1 until $\widehat{\mathsf{FF}}^* = \widehat{\mathsf{FF}}(i) \approx \widehat{\mathsf{FF}}(i+1)$ for some $i \geq 0$.

Theorem 2 ([23])**.** FF^* *is non-empty iff* $\widehat{\mathsf{FF}}^*$ *is non-empty.*

The following proposition allows us to avoid the costly nested fixed point computation if $\mathsf{Init} = Q_{\mathcal{T}}$.

Proposition 26. $\mathsf{post}^{i+1}(Q_{\mathcal{T}}) \sqsubseteq \mathsf{post}^{i}(Q_{\mathcal{T}})$ *for all* $i \geq 0$.

It follows from the transitivity of $\trianglelefteq$ that we can compute the following (further simplified) sequence:

Sequence 2

- $\overline{\mathsf{FF}}(0) = \mathsf{Min}(\trianglelefteq, \mathsf{Init})$
- $\overline{\mathsf{FF}}(i) = \mathsf{Min}\Big(\trianglelefteq, \mathsf{post}\big(\overline{\mathsf{FF}}(i-1)\big)\Big)$ for all $i \geq 1$.

Proposition 27. *If* $\mathsf{Init} = Q_{\mathcal{T}}$*, then* $\overline{\mathsf{FF}}(i) = \widehat{\mathsf{FF}}(i)$ *for all* $i \geq 0$.

Finally, as the SMT solver is responsible for finding solutions, here we only need a *semi-algorithm* that terminates when there is no solution. It follows that we can simply suppress the (expensive) termination check and stop when $\overline{\mathsf{FF}}(i)$ becomes empty.

Proposition 28. *If* $\mathsf{Init} = Q_{\mathcal{T}}$*, then* FF^* *is empty iff* $\overline{\mathsf{FF}}(i)$ *is empty for some* $i \geq 0$.

7. Experimental results

In this section, we check the relative effectiveness of the approaches discussed thus-far. We compare the approach discussed in Section 6 and the approach based on LTL model-checking (Section 5). For finding solutions, we use the SMT solver Z3 [11] (in particular, we use the theory of quantifier-free linear integer arithmetic) in an incremental fashion, i.e., we test the satisfiability of ψ^d_{sim} (defined in Proposition 25) with increasing values of d. For proving the absence of solutions, we implemented the semi-algorithm described in the last section (which computes $\overline{\mathsf{FF}}(i)$ until it becomes empty) in C++ using the AaPAL library [24]. All experiments were conducted on an IBM x3550 server (Intel Xeon X5680 @ 3.33GHz (12C/24T) + 64GB RAM) with a timeout of 1200 seconds. The benchmarks are available from [1].

7.1. Experimental setup

Test cases. The set of test cases consists of 500 randomly generated pairs of $\mathcal{I}, k$ (where 217 of them have no solution). The following parameters are chosen at random when generating them:

- *type* = {'Euclidean', 'Spherical', 'Random'}
- $n \in \{4, 5, 6, 7, 8\}$
- $k \in \{1, 2, 3\}$.

For the 'Euclidean' type, we generate points in a 10000×10000 grid and calculate the Euclidean distances between them. For the 'Spherical' type, we generate latitude and longitude values and compute the distances between them, assuming they are points on the Earth, with the haversine formula [25]. These distances are then scaled down and filled into FT to make $FT_{max} = 20$. For the 'Random' type, we generate the values in FT (except '1's on the diagonal) from $\{1, \ldots, 20\}$. Finally, for all these types of instances, we generate the values in RD from $\{1, \ldots, 3FT_{max}\}$.

Tools and modelling. We compared the following tools:

- NuSMV [26] is a BDD-based symbolic (LTL) model checker. For each CR-UAV instance $\mathcal{I}$ and presumed number of UAVs k, we model $\mathcal{A}_{\mathcal{I},k}$ in its modelling language and check it against the LTL specification φ in Section 5.3.
- ABC [27] is a SAT-based model checker that won the liveness track of the Hardware Model-Checking Competition 2014 (HWMCC'14) [28]. For our purpose, we translate our NuSMV models into AIGER circuits [29] with a customised version of SMVFlatten [30] and use ABC to check them.
- IImc [31] is a SAT-based model checker that won the liveness track of HWMCC'13. We use it in the same way as ABC.

7.2. Results

The execution time of 217 test cases with no solution are illustrated in Figure 28. From the figure, it is clear that both ABC and IImc are significantly faster than NuSMV. In particular, IImc performs better than ABC on these instances. Our semi-algorithm, labelled with 'sim' in the figure, is much more efficient than these three model checkers. Moreover, IImc and ABC use three to four solver engines that are ran on different CPU cores, whereas sim uses a single core.

The execution time of 283 test cases with solutions are illustrated in Figure 29. It is clear that Z3 is much faster than the other model checkers.

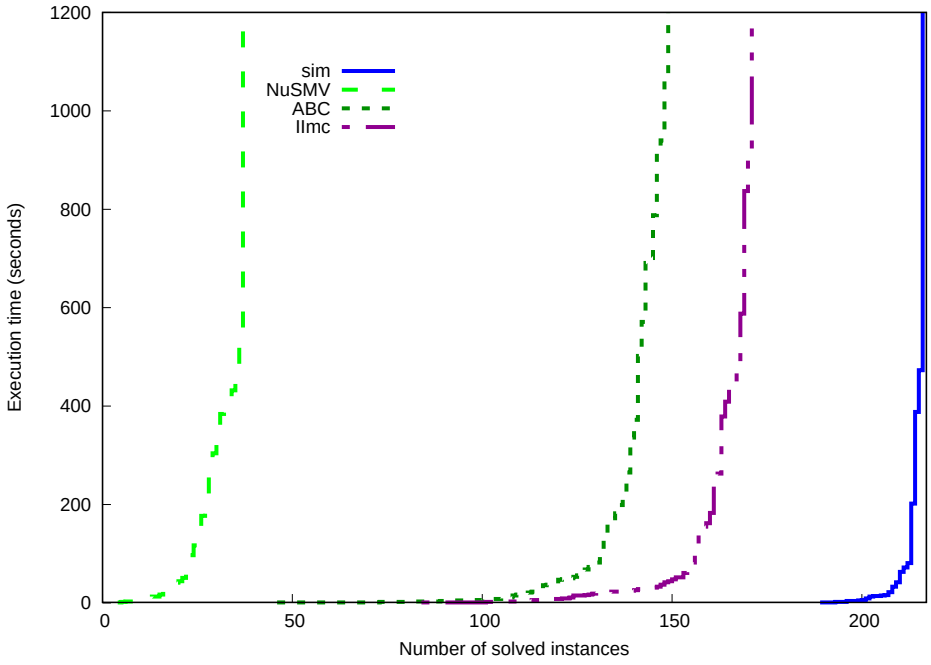

Figure 28: Comparison with model checkers for the 217 'no solution' instances. Our semi-algorithm is labelled with 'sim'.

8. Summary and future work

We defined the CR-UAV problem, proved that it is PSPACE-complete even in the single-UAV case (contrary to a claim in the literature [8]), suggested a reduction to model-checking, and presented experimental results with various algorithms, including a portfolio which runs in parallel a bounded model checker for detecting solutions and an explicit-state search that attempts to prove their absence.

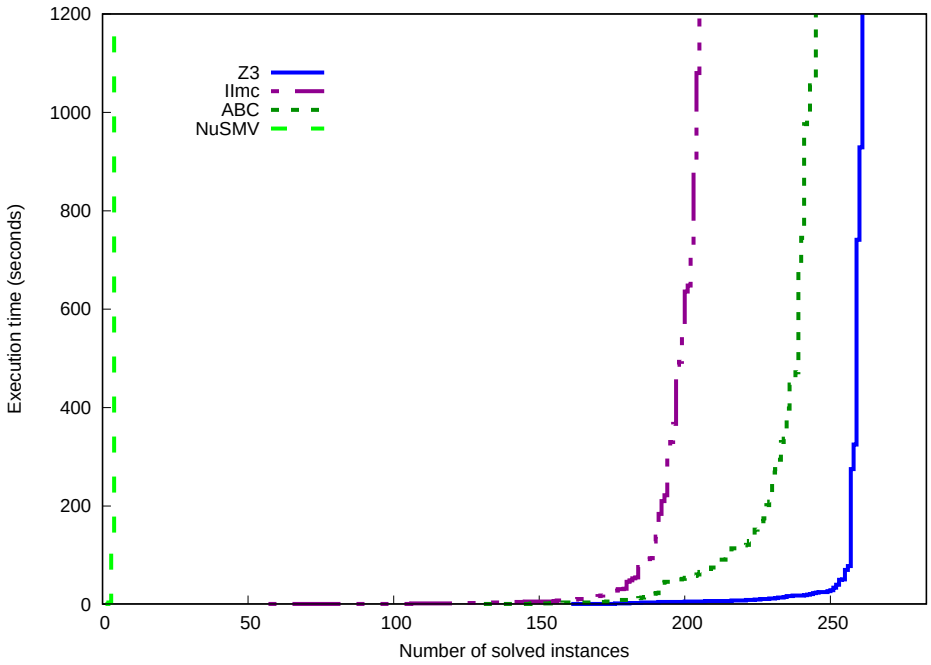

Figure 29: Comparison with model checkers for the 283 instances for which there is a solution.

There are multiple ways in which this research can be extended. On the algorithmic side, one may consider methods to accelerate the search by reducing the magnitude of the constants. It is trivial that they can all be divided (except the '1's on the diagonal of FT) by their greatest common divisor without affecting the solution. But what is more interesting is that when all constants are coprime, one may adopt the standard notion of *weakening* and *strengthening* specifications [16]. For example, if there is no solution to the instance obtained by dividing all the constants by a number, rounding up all the relative deadlines and rounding down all the flight times, then one can conclude that there is no solution to the original instance. One may also apply *symmetry reduction* [32] for further state-space reduction. Intuitively, since all UAVs are identical, it is not necessary to maintain an ordering between them.

On the complexity front, there are many unanswered questions. Our PSPACE-hardness proof crucially depends on the freedom to set flight times and relative deadlines. In [33], it is claimed that the *Euclidean* version of the problem, i.e., in which targets can be realised as points in a two-dimensional plane (with discretised distances between points), is NP-complete (with a single UAV). In the view of our result, we would like to investigate whether this claim is indeed true. It is conceivable that the techniques used in the well-known NP-hardness proof of EUCLIDEAN TSP [19] might be useful in this regard, but we were unfortunately unable to leverage them in the case at hand.

As we mentioned earlier, introducing *periodicity* into many NP-complete problems renders them PSPACE-complete [9]. The CR-UAV problem, on the other hand, can be seen as a recurrent variant of the decision version of the *Travelling Salesman Problem with Time Windows* (TSPTW) with only upper bounds (or *TSP with Deadlines* [34]). Its PSPACE-hardness hence stems from *recurrency*: the decision version of the (non-recurrent) TSPTW problem is NP-complete [35]. The main difference between the two notions is that, in the former, a problem instance consists of a number of parts that explicitly correspond to neighbouring periods (e.g., in the case of PERIODIC SAT, the input is a formula with two parts that correspond to two copies of variables), whereas this is not the case in the latter. Our reduction reveals a connection between these two types of problem specifications. In addition to the generalised windows scheduling problem we mentioned earlier (and a more restricted variant, the *generalised pinwheel scheduling* problem [36]), there are also crucial problems in other domains that are of a recurrent nature, e.g., the *message ferrying* prob-

lem [37]. Most of these problems are only known to be NP-hard. It would be interesting to investigate whether our reduction can be adapted to establish PSPACE-hardness of these problems. Another possible way to extend our complexity result is to do a more refined complexity analysis of the CR-UAV problem. For example, could it be *fixed-parameter tractable* (FPT) in some parameters? Is there a *polynomial-time approximation scheme* (PTAS) under certain conditions (see [38] for some related results)? We leave these questions for future work.

[1] The CR-UAV problem home-page, `http://ie.technion.ac.il/~ofers/cruav/`.

[2] Unmanned air vehicle systems association, `http://www.uavs.org/`.

[3] M. Alighanbari, Y. Kuwata, J. P. How, Coordination and control of multiple uavs with timing constraints and loitering, in: Proceedings of ACC 2003, Vol. 6, IEEE Press, 2003, pp. 5311–5316.

[4] K. Sundar, S. Rathinam, Route planning algorithms for unmanned aerial vehicles with refueling constraints, in: Proceedings of ACC 2012, IEEE Press, 2012, pp. 3266–3271.

[5] G. Yang, V. Kapila, Optimal path planning for unmanned air vehicles with kinematic and tactical constraints, in: Proceedings of CDC 2002, Vol. 2, IEEE Press, 2002, pp. 1301–1306.

[6] A. Richards, J. P. How, Aircraft trajectory planning with collision avoidance using mixed integer linear programming, in: Proceedings of ACC 2002, Vol. 3, IEEE Press, 2002, pp. 1936–1941.

[7] N. Drucker, M. Penn, O. Strichman, Cyclic routing of unmanned air vehicles, Tech. rep., Technion, Industrial Engineering and Management, iE/IS-2014-12. Available from [1]. (2014).

[8] N. Basilico, N. Gatti, F. Amigoni, Developing a deterministic patrolling strategy for security agents, in: Proceedings of WI-IAT 2009, IEEE Computer Society Press, 2009, pp. 565–572.

[9] J. B. Orlin, The complexity of dynamic languages and dynamic optimization problems, in: Proceedings of STOC 1981, ACM Press, 1981, pp. 218–227.

[10] A. Pnueli, The temporal logic of programs, in: Proceedings of FOCS 1977, IEEE Computer Society Press, 1977, pp. 46–57.

[11] L. de Moura, N. Bjørner, Z3: An efficient SMT solver, in: Proceedings of TACAS 2008, Vol. 4963 of LNCS, Springer-Verlag, 2008, pp. 337–340.
URL `http://dx.doi.org/10.1007/978-3-540-78800-3_24`

[12] H.-M. Ho, J. Ouaknine, The CR-UAV problem is pspace-complete, in: Proceedings of FoSSaCS 2015, Vol. 9034 of LNCS, Springer, 2015, pp. 328–342.

[13] N. Drucker, M. Penn, O. Strichman, Cyclic routing of unmanned aerial vehicles, in: Proceedings of CPAIOR 2016, Vol. 9676 of LNCS, Springer, 2016, pp. 125–141.

[14] A. Bar-Noy, R. E. Ladner, T. Tamir, T. VanDeGrift, Windows scheduling of arbitrary-length jobs on multiple machines, J. Scheduling 15 (2) (2012) 141–155.

[15] C. L. Liu, J. W. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment, J. ACM 20 (1) (1973) 46–61.

[16] T. A. Henzinger, Z. Manna, A. Pnueli, What good are digital clocks?, in: Proceedings of ICALP 1992, Vol. 623 of LNCS, Springer, 1992, pp. 545–558.

[17] K. L. McMillan, Symbolic model checking, Kluwer, 1993.

[18] M. Sipser, Introduction to the Theory of Computation, Cengage Learning, 2012.

[19] C. H. Papadimitriou, The Euclidean traveling salesman problem is NP-complete, Theoretical Computer Science 4 (3) (1977) 237–244.

[20] N. Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Tech. Rep. 388, CMU (1976).

[21] C. Baier, J.-P. Katoen, Principles of model checking, MIT Press, 2008.

[22] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, Y. Zhu, Bounded model checking, Advances in Computers 58 (2003) 117–148.

[23] L. Doyen, J.-F. Raskin, Antichains algorithms for finite automata, in: Proceedings of TACAS 2010, Vol. 6015 of LNCS, Springer, 2010, pp. 2–22.

[24] A. Bohy, Antichain based algorithms for the synthesis of reactive systems, Ph.D. thesis, University of Mons (2014).

[25] R. W. Sinnott, Virtues of the Haversine, Sky and Telescope 68 (1984) 158.

[26] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, NuSMV2: An opensource tool for symbolic model checking, in: Proceedings of CAV 2002, Vol. 2404 of LNCS, Springer, 2002, pp. 359–364.

[27] R. K. Brayton, A. Mishchenko, ABC: An academic industrial-strength verification tool, in: Proceedings of CAV 2010, Vol. 6174 of LNCS, Springer, 2010, pp. 24–40.
URL `http://dx.doi.org/10.1007/978-3-642-14295-6`

[28] B. Sterin, Personal Communication (2014).

[29] A. Biere, The aiger and-inverter graph (aig) format, Available at fmv.jku.at/aiger.

[30] A. Biere, Available at fmv.jku.at/smvflatten (2014).

[31] Z. Hassan, A. R. Bradley, F. Somenzi, Incremental, inductive CTL model checking, in: Proceedings of CAV 2012, Vol. 7358 of LNCS, Springer, 2012, pp. 532–547.
URL `http://dx.doi.org/10.1007/978-3-642-31424-7`

[32] E. A. Emerson, A. P. Sistla, Symmetry and model checking, Formal Methods in System Design 9 (1996) 105–131.

[33] J. Las Fargeas, B. Hyun, P. Kabamba, A. Girard, Persistent visitation under revisit constraints, in: Proceedings of ICUAS 2013, IEEE Press, 2013, pp. 952–957.

[34] H.-J. Böckenhauer, J. Hromkovic, J. Kneis, J. Kupke, The parameterized approximability of TSP with deadlines, Theory of Computing Systems 41 (3) (2007) 431–444.
[35] M. W. Savelsbergh, Local search in routing problems with time windows, Annals of Operations Research 4 (1) (1985) 285–305.
[36] E. A. Feinberg, M. T. Curry, Generalized pinwheel problem, Mathematical Methods of Operations Research 62 (1) (2005) 99–122.
[37] W. Zhao, M. H. Ammar, E. W. Zegura, A message ferrying approach for data delivery in sparse mobile ad hoc networks, in: Proceedings of MobiHoc 2004, ACM Press, 2004, pp. 187–198.
[38] M. V. Marathe, H. B. Hunt III, R. E. Stearns, V. Radhakrishnan, Approximation algorithms for PSPACE-hard hierarchically and periodically specified problems, SIAM Journal on Computing 27 (5) (1998) 1237–1261.
URL http://dx.doi.org/10.1137/S0097539795285254