Contents lists available at ScienceDirect

Theoretical Computer Science

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

# Parameterized pursuit-evasion games

Allan Scott [*], Ulrike Stege

*Department of Computer Science, University of Victoria, Engineering/Computer Science Building (ECS), Room 504 PO Box 3055, STN CSC Victoria, BC, Canada V8W 3P6*

ARTICLE INFO

*Article history:*
Received 25 September 2009
Received in revised form 13 April 2010
Accepted 3 July 2010
Communicated by A. Fraenkel

*Keywords:*
Parameterized complexity
Cops and robber
Algorithmic combinatorial game theory

ABSTRACT

We study the parameterized complexity of four variants of pursuit-evasion on graphs: Seeded Pursuit Evasion, Short Seeded Pursuit Evasion, Directed Pursuit Evasion and Short Directed Pursuit Evasion. Both Seeded Pursuit Evasion and Short Seeded Pursuit Evasion are played on undirected graphs with given starting positions for both the cops and the robber. Directed Pursuit Evasion and its short variant are played on directed graphs, with the players free to choose their starting positions. We show for Seeded Pursuit Evasion and Directed Pursuit Evasion that finding a winning strategy for the cops is AW[*]-hard when we parameterize by the number of cops. Further, we show that the short ($k$-move) variants of these problems (Short Seeded Pursuit Evasion and Short Directed Pursuit Evasion) are AW[*]-complete when we parameterize by both the number of cops and turns.

## 1. Introduction

The *pursuit-evasion* game (also known as *cops and robber*) takes place on a graph. The cops and robber occupy nodes in the graph and alternate taking turns where they may move to adjacent nodes. The cops win if any cop captures the robber by occupying the same node as him. The robber wins if he or she is able to evade the cops indefinitely.

This problem was initially studied for only one cop by Quilliot [11,12] and independently by Nowakowski and Winkler [10]; both characterized graphs on which one cop was sufficient to capture the robber. Additional cops were added by Aigner and Fromme [2], who showed that three cops are sufficient to capture a robber in a planar graph, and that it is possible to construct graphs requiring an arbitrary number of cops to capture a robber. The former result was expanded upon by Quilliot, who proved that $3 + 2k$ cops are sufficient to capture a robber in a graph of genus $k$ [13]. Goldstein and Reingold considered a seeded variant (i.e., one in which starting positions were given) and showed it to be EXPTIME-complete [8]. Surveys of the topic include [3,9,7].

Many variants of the pursuit-evasion problem are connected to interesting graph properties. If the cops are allowed to choose their starting positions and have only one move to catch a single visible robber, the number of cops needed to ensure capture in a graph $G$ is equal to the size of a minimum dominating set of $G$ [10]. In [2] it is shown that three cops are sufficient to guarantee the capture of a single robber in a planar graph, and that in a graph of girth at least five the number of cops necessary to guarantee capture is at least the minimum vertex degree in that graph. If the robber has infinite velocity (meaning it can move any distance as long as there is a cop-free path from his initial position to its destination), the cops travel by helicopter (each can move from its current vertex to any other vertex without visiting any vertices between), and both sides move simultaneously, the number of cops needed to guarantee capture is equivalent to the treewidth of $G$ [15].

In the first variant of the pursuit-evasion problem we study, Seeded Pursuit-Evasion, the cops and robber both move with a maximum velocity of one, both sides have perfect information (e.g., they can see each other at all times), the starting

[*] Corresponding author. Tel.: +1 250 598 7228.
*E-mail addresses:* aescott@cs.uvic.ca (A. Scott), stege@cs.uvic.ca (U. Stege).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.07.004

positions are given (not chosen), and the parameter is the number of cops $|C|$. Seeded Pursuit-Evasion is known to be hard for EXPTIME when unparameterized [8]. The parameterized complexity of the pursuit-evasion game when both sides are free to choose their starting position has been considered in [6] and shown to be W[2]-hard when parameterized by $|C|$, the number of cops. We formally define Seeded Pursuit-Evasion as follows:

Seeded Pursuit-Evasion
*Input*:
- A simple, undirected graph $G = (V, E)$
- A set $C \subseteq V$ of starting positions for the cops
- Starting position $a \in V$, $a \notin C$, for the robber

*Rules*:

1. $C$ defines the starting positions of the cops. Exactly one cop starts at each $v \in C$.
2. The robber begins at vertex $a$.
3. The cops and robber alternate taking turns, starting with the cops.
4. During the cops' turn, each cop either moves to a vertex adjacent to his current position, or stays put.[1] Multiple cops may occupy the same node at a time.
5. On his turn, the robber either moves to an unoccupied vertex adjacent to his current position, or stays put.
6. If, at any time, the robber occupies the same vertex as a cop, the robber is captured and the cops win.
7. Both players have complete information. That is, the cops know the positions of the robber and their fellow cops at all times, and vice-versa.
8. You cannot add or subtract pieces (cops or robbers) from the game at any time. Further, any given piece occupies exactly one vertex at a time.

*Parameter*: $|C|$
*Question*: Can the cops guarantee capture of the robber?

We also consider three other variants of this problem: in Short Seeded Pursuit-Evasion the cops have only $t$ turns to capture the robber and $t$ is an additional parameter. In Directed Pursuit-Evasion the graph is directed and the players choose their starting positions at the beginning of the game. Finally, Short Directed Pursuit-Evasion is Directed Pursuit-Evasion with a $t$-turn limit (as in Short Seeded Pursuit-Evasion).

In Section 2 of this article we review the parameterized class AW[*]. Section 3 proves that Seeded Pursuit Evasion is AW[*]-hard when parameterized by the number of cops. In Section 4 we prove that hypercubes can serve as the escape subgraphs needed for our reduction in Section 3. Section 5 extends our AW[*]-hardness reduction from Seeded Pursuit-Evasion to Short Seeded Pursuit Evasion and also shows that the latter problem is in AW[*], making this variant AW[*]-complete. We also show that Short Pursuit-Evasion is in AW[*]. In Section 6 we show that Directed Pursuit-Evasion and Short Directed Pursuit-Evasion are AW[*]-hard and AW[*]-complete respectively. Finally, we close with a brief summary and some open questions in Section 7.

## 2. The class AW[*]

Like classical complexity, parameterized complexity has its own hierarchy of complexity classes (see Fig. 1, [5]). In this case, the base class for tractability is FPT—the set of parameterized problems which can be solved in $O(f(k) \cdot n^c)$ time where $f$ is an arbitrary function, $k$ is the parameter (or the tuple of parameters), $n$ is the unparameterized input size, and $c$ is a positive constant.

The parameterized complexity class AW[*] is a subset of XP and a superset of the A-hierarchy. Each class of the A-hierarchy is in turn a superset of the corresponding W-hierarchy[2] class (that is, $W[t] \subseteq A[t]$ for $t \geq 1$), and W[1] contains FPT. Thus, if an AW[*]-complete problem were shown to be in FPT, the resulting collapse would consume both the A- and W-hierarchies.

Intuitively, AW[*] can be considered a parameterized version of the classical complexity class AP (the set of problems solvable in polynomial time by an alternating Turing machine). There is, however, a striking difference in that AP = PSPACE while AW[*] does not appear to correspond to any notion of parameterized space [4].

We now review the definition of the problem Unitary Parameterized QBFSAT$_2$, which is known to be AW[*]-complete [1].[3]

[1] Note that we say the cops and robber *may* move, but they can also stay at the vertex they currently occupy. This is equivalent to the variant where all pieces *must* move and every vertex has an edge to itself, since in that variant a piece can stay at its current vertex by moving along the reflexive edge.

[2] The W-hierarchy is the most well-known parameterized hierarchy, and is often viewed as the parameterized analog of NP [4]. Famous examples of complete problems in the W-hierarchy include the NP-complete problems Clique, which is W[1]-complete when parameterized by the size of the clique, and Dominating Set, which is W[2]-complete when parameterized by the size of the dominating set.

[3] Specifically, Unitary Parameterized QBFSAT$_t$ was shown to be AW[*]-complete for any fixed $t$ and we use $t = 2$.

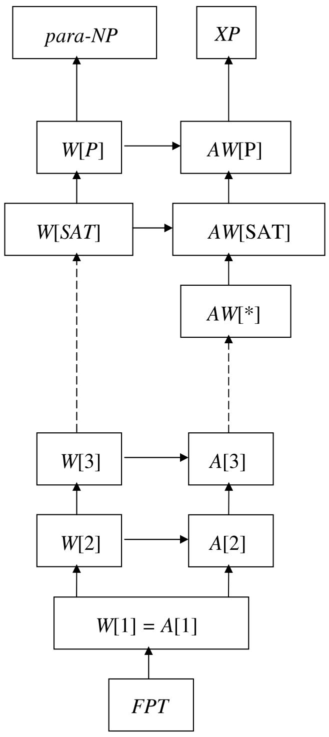

**Fig. 1.** Parameterized complexity hierarchies. An arrow from class $X$ to class $Y$ indicates that $X$ is known to be contained within $Y$.

UNITARY PARAMETERIZED QBFSAT$_2$
*Instance:* An integer $r$; a sequence $s_1, \dots, s_r$ of pairwise disjoint sets of boolean variables; a boolean formula $F$ in conjunctive normal form (CNF) over the variables $s_1 \cup \dots \cup s_r$ with negations applied only to variables.
*Parameter:* $r$
*Question:* Does there exist a variable $q_1 \in s_1$ such that for every variable $q_2 \in s_2$, there exists a variable $q_3 \in s_3$ such that . . . (alternating qualifiers) such that, when the variables $q_1, q_2, \dots, q_r$ are set to true and all other variables are set to false, the formula $F$ is true?

This problem serves as the basis for all our AW[*]-hardness reductions, contained in Sections 3, 5 and 6.

## 3. Hardness of seeded pursuit-evasion

In this section we prove that SEEDED PURSUIT-EVASION is AW[*]-hard. We obtain this result by using a parameterized reduction from UNITARY PARAMETERIZED QBFSAT$_2$. A parameterized reduction is much like a classical one, but the time permitted to perform the reduction is bounded by a fixed-parameter tractable function rather than a polynomial one, and we must not introduce any aspect of the input size into the parameters for the target problem [4]. We explain the mechanics of the reduction first; the detailed proof follows after.

### *3.1. Reduction*

Let the entire UNITARY PARAMETERIZED QBFSAT$_2$-instance given as input, including the quantifiers in the question (ordered from 1 to $k$), variable sets, and formula $F$, be referred to as $A$.

Our strategy is to construct an instance of SEEDED PURSUIT-EVASION which simulates setting the variables in $A$ and then tests the setting on the formula $F$. This means the cops can guarantee capture of the robber if and only if they can produce a satisfying assignment in the simulated setting. Therefore, the cops are guaranteed to catch the robber in the reduced instance if and only if $A$ is satisfiable. For our parameter, we set $|C| = r + 1$.

Our reduction employs four component gadgets: escape subgraphs, runways, the assignment gadget, and a set of clause vertices. *Escape subgraphs* enable the robber to evade the cops indefinitely (in other words, the robber wins the game if he enters an escape subgraph), *runways* simulate the variables of $A$, the *assignment gadget* enables the robber to effectively set the values of the (simulated) universally-quantified variables, and the *clause vertices* enable the robber to test whether a clause is satisfied by the simulated variable setting. The starting positions for the cops and the robber are defined in the gadgets.

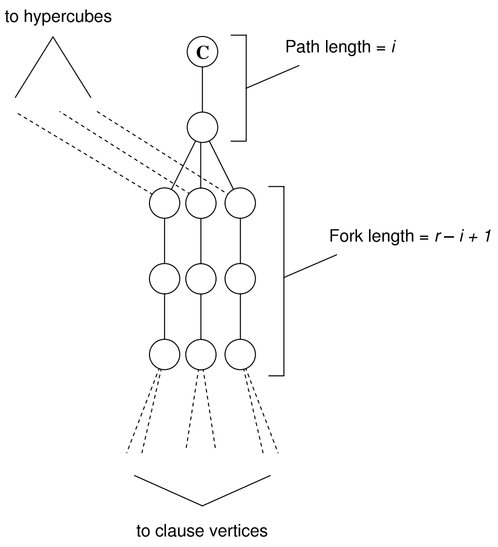

**Fig. 2.** Runway gadget construction for the $i$th quantifier. The cop ($C$) is shown in his starting position. Note that there is a one-to-one correspondence between the forks and the variables of $s_i$.

The terms *move* and *turn* are used interchangeably in this reduction. Both mean that one player makes his full move. In the case of the robber this entails moving from his current node to an adjacent one (or choosing to stay put). In the case of the cops, this means that *all* of the cops have either moved or committed to staying put. A *round* consists of two turns: one for all the cops, followed by a turn for the robber.

We now describe the construction of each gadget in detail.

### 3.1.1. *Escape subgraphs*

The sole purpose of the escape subgraphs is to provide the robber with a place in which he can evade the cops indefinitely. Any graph of fixed-parameter-tractable size in which a robber could perpetually evade $r + 1$ cops would suffice for our purpose. We use hypercubes.

**Definition 1.** The *hypercube graph of degree $n$, $Q_n$*, is the graph whose vertices lie on the corners of the $n$-dimensional unit cube. An edge exists between two vertices in $Q_n$ if and only if their coordinates differ in exactly one position.

To ensure that the hypercube fulfills its function as an escape subgraph we rely on the following corollary, which is proved in Section 4.

**Corollary 1.** *A robber can evade $k$ cops indefinitely in $Q_{2k}$.*

We use hypercubes of degree $2r + 2$ as escape subgraphs in our reduction, since the number of cops is the instance constructed is $r + 1$.

### 3.1.2. *Runways*

For each quantifier in $A$ we create a corresponding runway in our Seeded Pursuit-Evasion instance (see Fig. 2). The runway corresponding to the $i$th quantifier of $A$ consists of a path of length $i$ which then forks into $|s_i|$ paths, making the number of paths equal to the number of variables associated with the $i$th quantifier. For each of the variables associated with the quantifier, there is a path of length $r - i + 1$; we call these paths the *forks* of the runway. Moving the cop into a fork corresponds to a setting in which the variable associated with the fork is true and all the other variables in $s_i$ are false. We refer to the first vertex of the length-$i$ path as the *start* of the runway, while the fork vertices furthest from this start are referred to as *end vertices* of the runway. A single cop begins at the start of the runway (see Fig. 2), and the distance the cop must travel from the start to any end of that runway is at least $r$.

### 3.1.3. *The assignment gadget*

We construct the assignment gadget starting from a path of length $r+2$ (remembering that $r$ is the number of quantifiers). There is one cop, which starts on the first vertex of this path, while the robber starts on the third. The remaining $r$ vertices correspond to the quantifiers of $A$; specifically, the $i+2$nd vertex corresponds to quantifier $i$. Quantifiers are indexed from 1, so the robber's starting position corresponds to the first quantifier. Each vertex which corresponds to a universal quantifier (which, by the definition of Unitary Parameterized QBFSAT$_2$, is every even-numbered quantifier) is replaced with a clique of size equal $|s_i|$. We call these *assignment cliques*. For an example of an assignment gadget, we refer to Fig. 3.

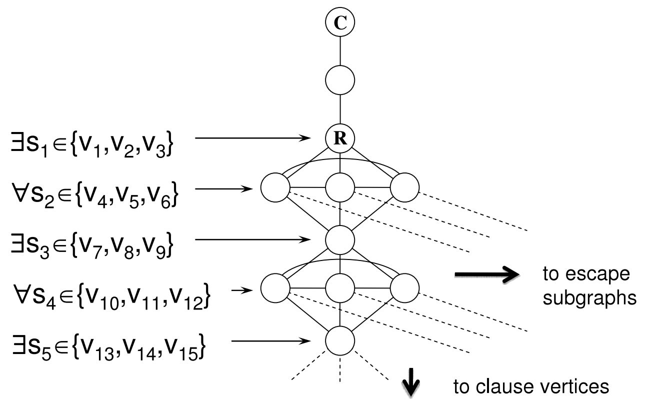
C
$\exists s_1 \in \{v_1,v_2,v_3\}$
R
$\forall s_2 \in \{v_4,v_5,v_6\}$
$\exists s_3 \in \{v_7,v_8,v_9\}$
$\forall s_4 \in \{v_{10},v_{11},v_{12}\}$
$\exists s_5 \in \{v_{13},v_{14},v_{15}\}$
to escape subgraphs
to clause vertices

**Fig. 3.** An example of the assignment gadget for $r = 5$. The cop ($C$) and robber ($R$) are shown in their starting positions.

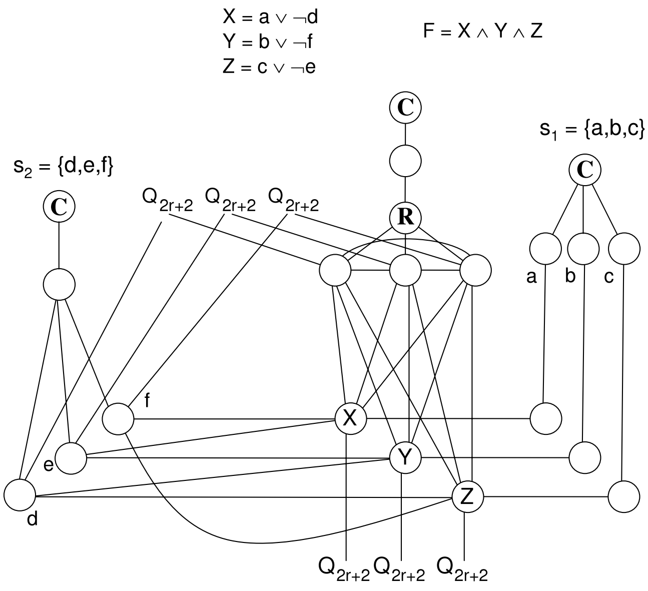

**Fig. 4.** A Seeded Pursuit-Evasion instance constructed by our reduction from the Unitary Parameterized QBFSAT$_2$ instance $s_1 = \{a, b, c\}$, $s_2 = \{d, e, f\}$, $F = (a \wedge \overline{d}) \vee (b \wedge \overline{f}) \vee (c \wedge \overline{e})$. The cops ($C$) and robber ($R$) are shown in their starting positions.

Next, we connect the assignment cliques with the runway forks. We do this as follows: for the assignment clique corresponding to quantifier $i$, assign to each vertex a unique variable in $s_i$ (this results in a one-to-one mapping from the elements of $s_i$ to the clique's vertices). For each vertex $v$ in the assignment clique, create a unique escape subgraph $Q^v_{2r+2}$. Choose a vertex $w \in Q^v_{2r+2}$ and add an edge connecting $w$ to $v$ and an edge connecting $w$ to the first vertex of the runway fork from the $i$th runway which corresponds to the same variable as $v$. Fig. 5 shows an example of how the assignment gadget is connected to a runway.

### *3.1.4. Clause vertices*

Finally we describe the clause vertices ($X$, $Y$, and $Z$ in Fig. 4). Each of these corresponds to a clause in $F$. Each clause vertex has an edge to every runway fork which represents at least one literal that satisfies the vertex's corresponding clause. Every clause vertex is also connected to the end of the assignment gadget (the opposite end of the path from where the cop starts). If the last quantifier in $A$ is existential, then the end of the assignment gadget is a single vertex. On the other hand, if the last quantifier in $A$ is universal, then the end of the assignment gadget is a clique and each clause vertex is connected to every vertex in that clique. Lastly, we create a unique escape subgraph for each clause vertex and connect each clause vertex to a single vertex of its unique escape graph.

## *3.2. Sequence of play*

The general idea of this reduction is that the runways simulate the quantifiers. For existential quantifiers, the cops are free to choose the true variable for that quantifier, while for universal quantifiers the connections between the assignment

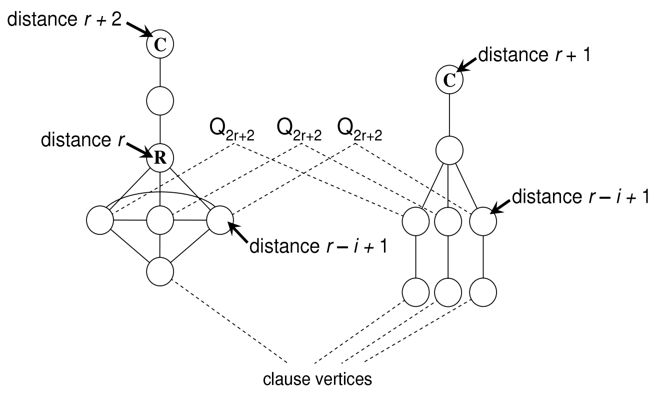

**Fig. 5.** Distances from the clause vertices in the assignment gadget and runways.

gadget and the runway enable the robber to manipulate the choice of variable. Our goal is that when the robber reaches the clause vertices, he will have a safe path to an escape subgraph if and only if there is an unsatisfied clause. For an example of an instance constructed by this reduction, see Fig. 4.

When a cop chooses a fork to follow, this can be viewed as setting the variable corresponding to that fork to true while setting the rest of the variables in the same quantifier set $s_i$ to false. In runways corresponding to existential quantifiers, the cops position themselves as they choose. For universal quantifiers, the runway cop is forced to block the robber: when the robber positions himself in one of the vertices in an assignment clique, the cop must move into the corresponding fork (thus enabling him to catch the robber with his next move should the robber attempt to enter the escape subgraph) or the robber will be able to move into the adjacent escape subgraph where he can elude the cops indefinitely.

Once all the variables are set, the robber moves from the end of the assignment gadget into a clause vertex. Since the clause vertices are connected to the runway forks, the cops will be able to capture the robber if he moves into a vertex corresponding to a satisfied clause. However, if he moves into an vertex corresponding to an unsatisfied clause, then the cops will be unable to catch him on their next turn and he can flee into an escape subgraph on his subsequent turn.

### *3.2.1. Proving the sequence of play*

We now show the correctness of our reduction with a sequence of lemmas. The first four are independent of instance $A$, while the last three depend on the satisfiability of $A$.

**Lemma 1.** *The cops can either capture the robber or force him into a clause vertex on round $r$.*

**Proof.** The cop in the assignment gadget starts only two vertices away from the robber (see Fig. 3) and moves first. Thus if this cop always moves toward the robber, the robber must always move away from the cop. If the cops in the runways corresponding to universally-quantified variables move to guard the escape subgraphs adjacent to the assignment cliques, then the robber cannot move into any of them because he would be captured. Thus the robber can only leave the assignment gadget by entering a clause vertex and he will do so in round $r$, since that is the distance from his starting position to any clause vertex. □

**Lemma 2.** *The robber can make it into any clause vertex on round $r$ without getting captured, regardless of the cops' strategy.*

**Proof.** As this proof depends heavily on distances to clause vertices, Fig. 5 may be of assistance.

By our definition for the assignment gadget, the robber starts at distance $r$ from the clause vertices. Let us assume the robber's evasion strategy decreases his distance from the clause vertices each turn, since the cop in the assignment gadget can catch the robber otherwise. The cop in the assignment gadget which starts at distance $r + 2$ from the clause vertices cannot catch the robber in the first $r$ turns since the cop's distance to the clause vertices will always be greater than the robber's. The cop does have the advantage of moving first, but this only enables him to temporarily close the gap since the robber starts out two vertices ahead.

If a runway cop moves to the end of his runway and enters the clause vertices from there, then the distance covered is at least $r + 1$. Further, his distance from the clause vertices will always be higher than the robber's at the end of each of the first $r$ rounds. Thus, the only remaining possible way a cop could capture the robber before round $r + 1$ is if a runway cop can find a shorter path to the clause vertices by leaving the runway early and entering the assignment gadget. However, this is not the case: for any universal quantifier,[4] if that quantifier is numbered $i$ then both the assignment clique and the connected vertices in the associated runway forks are at distance $r - i + 1$ from the clause vertices. It takes two turns to move between

[4] We do not consider existential quantifiers as their corresponding runways are not connected to the assignment gadget—only the clause vertices. Thus the cops in these runways can only leave through clause vertices.

the runway forks and the assignment gadget, but these moves do not decrease the cop's distance to the clause vertices and thus these moves are lateral and not part of any shortest path to the clause vertices. Even with the advantage of moving before the robber, a cop using this move would arrive in the assignment gadget at distance $r - i + 1$ from the clause vertices on turn $i + 2$, while the robber would be at distance $r - (i + 1) = r - i - 1$ from the clause vertices and would also be next to move.

Thus, any shortest path to the clause vertices for a runway cop is of length at least $r + 1$ and no runway cop can catch the robber in the first $r$ turns even if they leave their runways. This leaves no other cop who could possibly catch the robber. □

**Lemma 3.** *If the robber is on a clause vertex $v$ at the end of round $r$, the cops have a winning strategy if and only if there is a cop at the end of a runway fork which corresponds to a literal that satisfies the clause $B$ corresponding to $v$.*

**Proof.** By definition, each runway fork corresponds to a set of literals. The end of each runway fork is connected to every clause vertex for which the associated clause contains at least one of its associated literals. Therefore, if there is a cop at the end of a runway fork corresponding to a literal $b$ that satisfies $B$, then that cop can simply follow the edge to $v$ and capture the robber.

On the other hand, let us assume that no cop is at the end of a runway fork corresponding to a literal that satisfies $B$. Other than runway forks, the only vertices adjacent to $v$ are those at the end of the assignment gadget and those in the adjacent hypercube. To show that there is no cop adjacent to $v$, observe that no cop can reach the hypercube in $r$ rounds since it is distance $r + 2$ from the starting position for runway cops and $r + 3$ from the starting position for the cop in the assignment gadget. Similarly, the cop in the assignment gadget starts at distance $r + 2$ from the clause vertices, and the runway cops start at distance $r + 3$ from the end of the assignment gadget. □

**Lemma 4.** *Once a runway cop has entered a runway fork, he cannot reach the end of any other runway fork by the end of round $r$.*

**Proof.** As noted in the definition of the runway gadget, the cop starts at distance $r$ from the ends of the runways. Once he enters a fork, he increases the distance to the end of every other fork. Thus his distances to the ends of those forks are more than the number of rounds he has left. □

The following lemma shows that when the robber enters an assignment clique, he chooses which fork the cop in the associated runway must start down. This renders the cop unable to reach the end of any other fork.

**Lemma 5.** *For runways corresponding to universally-quantified variables, the robber decides which fork the cop can reach the end of.*

**Proof.** When the robber enters a vertex $v$ in an assignment clique,[5] the cop in the runway associated with that clique must move into $u$, the first vertex of the runway fork associated with $v$. This prevents the robber from moving into the escape subgraph adjacent to $v$, since both $u$ and $v$ are adjacent to the same vertex in the escape subgraph (see Figs. 4 and 5). Otherwise, if the cop does not move in this manner the robber can move into the escape subgraph unimpeded and evade the cops indefinitely once in it.

The cop cannot undo the robber's selection by moving back and into another fork as he will be unable to reach the end of another fork with his remaining turns in such a situation by Lemma 4. □

**Lemma 6.** *If $A$ is satisfiable, then the cops can guarantee capture of the robber.*

**Proof.** By Lemma 1 the cops can force the robber into a clause vertex in round $r$ (or capture him if he does not comply). Since $A$ is satisfiable, it is possible to create a satisfying assignment taking into account how the universal variables get set. Therefore, regardless of how the robber manipulates the cops in the runways corresponding to universal quantifiers, the cops can choose runway forks in the runways corresponding to the existential quantifiers of a satisfying assignment.

Assuming the cops have played following a satisfying assignment, and every cop has moved to the end of his runway fork, there must be at least one cop adjacent to each clause vertex. This is because the set of literals corresponding to the runways occupied by the cops satisfies $F$, which means that, by definition, for each clause vertex, there is at least one adjacent runway end (representing a literal which satisfies the clause) which is occupied by a cop. □

**Lemma 7.** *If $A$ is unsatisfiable, then the robber can evade the cops indefinitely regardless of their strategy.*

**Proof.** By Lemma 2, the robber can get to a clause vertex safely in round $r$.

By Lemma 3, if the robber is in clause vertex $v$ corresponding to clause $B$, he can survive round $r$ and enter the adjacent escape subgraph in round $r + 1$ (where he can evade the cops indefinitely) if and only if there is no cop at the end of any runway corresponding to a literal which satisfies $B$.

By Lemma 5, the robber can decide for each universal variable which fork the runway cop can reach the end of. Since $A$ is unsatisfiable, he can force the cops into runway forks such that the set of literals corresponding to the runways the cops can reach the ends of does not satisfy $F$. Therefore, even if the cops move to the ends of their runways they will not be able to cover every clause vertex. The robber can survive round $r$ in that clause vertex and then move into the adjacent escape subgraph on his next turn, guaranteeing that he cannot be captured. □

[5] Remember that we only use assignment cliques when the associated quantifier is universal.

### *3.3. Correctness*

With the following proof, we conclude our hardness result:

**Theorem 1.** Seeded Pursuit-Evasion *is AW[*]-hard.*

**Proof.** Lemmas 6 and 7 show that the cops can force a win in the constructed instance if and only if $A$ is satisfiable. All that remains to show is that our reduction is parameterized.

If $A_C$ is the number of clauses in $A$ and $A_V$ is the number of variables in $A$, then for the number of vertices in the constructed instance we have:

- At most $A_C + A_V$ hypercubes, each of size $2^{2r+2}$.
- $A_V$ runways, each of size $r + 1$, with some overlap.
- $A_C$ clause vertices, each of size 1.
- one assignment gadget of size at most $A_V + 2$.

While the size of the constructed instance is not polynomial in the size of the input because of the exponentially-large hypercubes, the size of the hypercubes is dependent only on $r$, the parameter. There is only a polynomial number of hypercubes (and other gadgets) so the size of the graph as a whole remains fixed-parameter tractable. The parameter in our target reduction is $|C|$, which is equal to $r+1$ (one cop in each of the $r$ runways, and one in the assignment gadget), so our new parameter remains independent of the input size. Thus we have a parameterized reduction from Unitary Parameterized QBFSAT$_2$ to Seeded Pursuit Evasion. □

## 4. Pursuit in a hypercube

We have omitted one crucial proof from the reduction in our previous section: that our robber can evade the cops indefinitely once he enters a hypercube. We provide that proof in this section.

We use coordinates of the hypercube (which are all 0 or 1) as a mapping from vertices to subsets of an $n$-element set. In this case, an edge between two vertices exists if and only if the sets corresponding to those vertices differ by exactly one element.

We now define a notion of a local dominating set. We use $N[v]$ to denote the closed neighborhood of vertex $v$ (which includes $v$), while $N(v)$ is the open neighborhood of $v$ (which excludes $v$).

**Definition 2.** A graph $G = (V, E)$ has a *local dominating set of size k* if and only if there exists some $v \in V$ and $V' \subseteq V - \{v\}$ such that $|V'| \leq k$ and $V'$ dominates $N[v]$.

Note that $V'$ must always contain at least one vertex of $N(v)$ in order to dominate $v$. You always have a local dominating set when $k$ is larger than or equal to the degree of the lowest-degree vertex in the graph, since for that vertex the local dominating set can consist of all its neighbors.

If a graph does not have a local dominating set of size $k$ (meaning that no vertex in the graph meets the criteria given above) then the robber can evade $k$ cops as follows:

**Lemma 8.** *If a graph $G$ does not have a local dominating set of size $k$, then a single robber can evade $k$ cops in $G$ indefinitely.*

**Proof.** If graph $G = (V, E)$ has no local dominating set of size $k$ and the robber is on a vertex $v$, then by definition the $k$ cops will not be able to position themselves in such a way that they are on or adjacent to every vertex in $N[v]$. Thus, the robber can escape to an undominated vertex and therefore the cops will not be able to capture him on their next turn. Since this property applies to every vertex in the graph, the robber can evade the cops indefinitely. □

All that remains is for us to show that a hypercube $Q_{2k}$ does not have a local dominating set, thus allowing the robber to employ this simple evasion strategy.

**Theorem 2.** *$Q_{2k}$ has no local dominating set of size $k$.*

**Proof.** We associate with each vertex a length $2k$ bit string—namely its coordinates on the $2k$-dimensional unit cube. In this representation, two vertices in $Q_{2k}$ are adjacent if and only if their bit strings differ in exactly one position. When we refer to a weight-$x$ vertex, we mean a vertex whose bit string has Hamming weight $x$.

Now, assume without loss of generality that vertex $v$ is represented by the all-zero string. $N[v]$-the closed neighborhood of $v$-consists of all $2k$ weight-1 vertices and the weight-0 vertex ($v$) itself. Any weight-1 vertex dominates itself and $v$. Any weight-2 vertex also dominates exactly two weight-1 vertices (its two weight-1 neighbors being the two strings you can get by changing either of the weight-2 bitstring's ones to zeros). You are not allowed to use the weight-0 vertex since it is $v$, and vertices of weight 3 or higher are not adjacent to any relevant vertices. Thus, all vertices besides $v$ dominate at most two vertices of $N[v]$.

Since each vertex can dominate at most 2 elements of $N[v]$, a set of $k$ vertices can dominate at most $2k$ vertices. However, $N[v]$ contains $2k + 1$ vertices and therefore $k$ vertices are insufficient to dominate it. □

Since, by Theorem 2, $Q_{2k}$ has no local dominating set, and by Lemma 8 a robber can evade $k$ cops indefinitely in a graph with no local dominating set of size $k$, we can derive the result we need for our reduction:

**Corollary 1.** *A robber can evade $k$ cops indefinitely in $Q_{2k}$.*

## 5. Short seeded pursuit-evasion

In [1], Abrahamson, Downey, and Fellows sought to parameterize games by considering *short* variants which ask whether the player to move can force a win within the next $t$ turns. This new input $t$ becomes a parameter. Abrahamson, Downey, and Fellows applied this approach to Generalized Geography and Node Kayles in [1] and showed the resulting problems to be AW[*]-hard for parameter $t$. Downey and Fellows later conjectured that AW[*] was the natural home of $k$-move (short) games [4]. In this section, we now apply this technique to Seeded Pursuit-Evasion.

Short Seeded Pursuit-Evasion
*Input*:

- A simple, undirected graph $G = (V, E)$
- A set $C \subseteq V$ of starting positions for the cops.
- Starting position $a \in V$ for the robber.
- Positive integer $t$.

*Rules*: *As for* Seeded Pursuit-Evasion.
*Parameters*: $t, |C|$
*Question*: Can the cops guarantee capture of the robber within $t$ moves?

We parameterize by both $t$ and $|C|$ because our hardness result then holds for any subset of those parameters (e.g. for parameterizing by $t$ or by $|C|$ alone).

### *5.1. Hardness of short seeded pursuit-evasion*

We use our reduction from Section 3 again to inform us as to the parameterized complexity of Short Pursuit Evasion:

**Lemma 9.** Short Seeded Pursuit-Evasion *is AW[*]-hard.*

**Proof.** We reuse the reduction from Unitary Parameterized QBFSAT$_2$ to Seeded Pursuit-Evasion. By Lemma 6, if $A$ (the original Unitary Parameterized QBFSAT$_2$ instance) is satisfiable then the cops have a winning strategy. The winning strategy given in that lemma takes at most $r + 1$ rounds to execute. In all other cases (that is, when $A$ is not satisfiable) the cops do not have a winning strategy, as per Lemma 7. Since $r$ is the original parameter of $A$, setting $t = r + 1$ is both correct and parameterized. Thus we have a parameterized reduction to Short Seeded Pursuit-Evasion and can conclude that the problem is AW[*]-hard. □

### *5.2. Membership of short seeded pursuit-evasion*

We reduce Short Seeded Pursuit-Evasion to the AW[*]-complete problem Parameterized QBFSAT$_t$ [1]. Once again, we first outline our parameterized reduction, then follow with a proof of correctness.[6]

Parameterized QBFSAT$_t$
*Instance:* An integer $r$; a sequence $s_1, \ldots, s_r$ of pairwise disjoint sets of boolean variables; a boolean formula $F$ over the variables $s_1 \cup \cdots \cup s_r$ that consists of $t$ alternating layers of conjunctions and disjunctions with negations applied only to variables ($t$ is a fixed constant).[7]
*Parameter:* $r, k_1, k_2, \ldots, k_r$
*Question:* Does there exist a subset $q_1 \subseteq s_1$ of size $k_1$ such that for every subset $q_2 \subseteq s_2$ of size $k_2$, there exists a subset $q_3 \subseteq s_3$ of size $k_3$ such that ... (alternating qualifiers) such that, when all the variables in $q_1, q_2, \ldots, q_r$ are set to true and all other variables are set to false, formula $F$ is true?

Given an instance of Short Seeded Pursuit-Evasion, our goal is to create a formula $F$ which encodes the rules of the game and is satisfied if and only if there is a winning strategy for the cops. As before, this must be a parameterized reduction.

We first create two sets of variables to represent the positions of the cops and robber:

- $c_{dvj}$ is true if and only if cop $d$ is at vertex $v$ in round $j$.
- $r_{vj}$ is true if and only if the robber is at vertex $v$ in round $j$.

We use the quantifiers to capture the alternating nature of the pursuit problem. Since the cops must choose their own moves, we use the existential quantifiers for the cops' moves, and since a winning strategy for the cops must take into account any possible move by the robber we use the universal quantifiers for his moves. Thus, for the existential quantifiers where $i$ is odd, we set $s_i = \{c_{dv((i+1)/2)} | 1 \leq d \leq |C|, v \in V\}$ and $k_i = |C|$. For universal quantifiers where $i$ is even, we set $s_i = \{r_{v(i/2)} | v \in V\}$ and $k_i = 1$.

[6] A preliminary version of this proof appeared in [14].

[7] Note that for $t = 2$ we require that $F$ be in CNF.

The $k_i$ parameters cause the correct number of cops to occupy vertices. However, this does not preclude the possibility of one cop occupying multiple vertices while others occupy none. We will prevent this by adding a rule for the cops to our formula $F$ (specifically $R_8$ below).

We use formula $F$ to encode the movement rules and the winning condition. We write the robber's constraint rules such that any variable setting which corresponds to a game in which the robber either cheated or was captured satisfies the formula — corresponding to a cop-win. Thus the robber's rules are negated so that a violation results in a clause being true. These clauses are then arranged in disjunctive normal form so that if any clause is true the entire formula is true as well. For the cops we want a single violation to result in the entire formula evaluating to false, so their clauses are arranged in CNF.

We now restate the rules of Short Seeded Pursuit-Evasion, encode them as formulas, and prove that these encodings properly enforce them. Negations are represented with overlines (e.g., the negation of $x$ is $\overline{x}$).

**Rule 1:** Exactly one cop starts at each $v \in C$.
Implementation: $R_1 = \bigwedge_{v \in C} c_{f(v)v0}$, where $f(v)$ as an arbitrary one-to-one function that maps the vertices of $C$ to the set $\{1, \ldots, |C|\}$.

**Proof.** By definition, $c_{f(v)v0}$ is true if and only if cop $f(v)$ is on vertex $v$ on turn 0. Therefore, $R_1$ is true if and only if the cops started on all the vertices in $C$. □

**Rule 2:** The robber begins at vertex $a$.
Implementation: $R_2 = \overline{r}_{a0}$

**Proof.** By definition, $r_{a0}$ is true if and only if the robber is on vertex $a$ on turn 0. Thus $R_2$ is true if and only if the robber starts on some vertex other than $a$. □

**Rule 3:** The cops and robber alternate taking turns, starting with the cops.

**Proof.** The cops' turns correspond to existential quantifiers and the robber's turns correspond to universal quantifiers, which alternate as desired. For each existential quantifier $i$ is odd and $s_i$ is the complete set of variables for cop positions on the corresponding turn. For each universal quantifier $i$ is even and $s_i$ is the complete set of variables for robber positions on the corresponding turn. □

**Rule 4:** During the cops' turn, each cop either moves to a vertex adjacent to his current position, or stays put. Multiple cops may occupy the same node at a time.
Implementation: $R_4 = \bigwedge_{\substack{(u,v)\notin E, u\neq v \\ 1\leq s\leq t \\ 1\leq i\leq |C|}} (\overline{c}_{iu(s-1)} \vee \overline{c}_{ivs})$.

**Proof.** It may be easier to see the correspondence with $R_4$ if we state Rule 4 as its complement: namely that no cop may move to a non-adjacent vertex.

Multiple cops (say $i$ and $j$) occupying the same node is accomplished by setting both $c_i vs$ and $c_j vs$ to true. In essence, we permit this behavior by never disallowing it. □

The following rule is the robber's analog to Rule 4.

**Rule 5:** On his turn, the robber either moves to an unoccupied vertex adjacent to his current position, or stays put.
Implementation: $R_5 = \bigvee_{\substack{(u,v)\notin E, u\neq v \\ 1\leq s\leq t}} (r_{u(s-1)} \wedge r_{vs})$.
Now we add the winning conditions for the cops. For implementation, these are lumped in with the robber's rules and thus phrased in DNF.

**Rule 6:** If, at any time, the robber occupies the same vertex as a cop, the robber is captured and the cops win.
For simplicity, we split this rule into two:

**Rule 6a:** The cops win if a cop enters the vertex occupied by the robber.
Implementation: $R_{6a} = \bigvee_{\substack{v\in V \\ 1\leq s\leq t \\ 1\leq i\leq |C|}} (r_{v(s-1)} \wedge c_{ivs})$.

**Proof.** Since the cops move first, they can capture the robber by entering the vertex he was in at the end of the last round $(s-1)$. Thus, if the robber ends round $s-1$ on vertex $v$, the cops can capture him if and only if some cop $i$ (for any $i$ such that $1 \leq i \leq |C|$) enters vertex $v$ in round $s$. This is true if and only if $r_{v(s-1)} \wedge c_{ivs}$ is true. Thus $R_{6a}$ is true if and only is a cop moves into the robber's vertex. □

As an alternative to the situation in $R_{6a}$, the robber may move into a cop's position. This, of course, gets him caught and thus we must enforce this possibility as well.

**Rule 6b:** The cops win if the robber enters a vertex occupied by a cop.
Implementation: $R_{6b} = \bigvee_{\substack{v\in V \\ 1<s\leq t \\ 1\leq i\leq |C|}} (r_{vs} \wedge c_{ivs})$.
The proof of this rule is similar to that of $R_{6a}$, except that since the robber moved last we check his position in round $s$ instead of $s-1$.

**Rule 7:** Both players have complete information.

**Proof.** This is due to the fact that the quantifiers are resolved sequentially in a left-to-right manner. That is, when it is time to resolve the $i$th quantifier, the variables corresponding to quantifiers 1 through $i - 1$ have already been decided and are known. Thus complete information is encoded into the formula. □

**Rule 8:** You cannot add or subtract pieces (cops or robbers) from the game at any time. Further, any given piece occupies exactly one vertex at a time.
Implementation: $R_8 = \bigwedge_{\substack{u,v \in V \\ u \neq v \\ 1 \leq s \leq t \\ 1 \leq i \leq |C|}} (\overline{c}_{ius} \vee \overline{c}_{ivs})$.

**Proof.** The first part of this statement is enforced by the quantifier weights. For every universal quantifier numbered $i$ (robber's turns), we defined the weight $k_i$ to always be 1 and the set $s_i$ to be the set of all $r_{v(i/2)}$. Thus, for any given round $t$, exactly one of the robber variables $r_{vt}$ is true, indicating that the robber is on vertex $v$.

Similarly, for the existential quantifiers (cops' turns) we defined the weight to always be $|C|$. However, this is not sufficient to ensure that all $|C|$ cops are placed as it is possible that the variable assignment could set one cop in multiple positions while another is not given any position. Rule $R_8$ ensures that no cop occurs twice: if a cop occupies two vertices, say $u$ and $v$, on turn $i$ then the clause containing both $c_{ius}$ and $c_{ivs}$ is false and thus $R_8$ would be false. If no cop occupies two vertices on the same turn, then for any turn $i$ and any pair of vertices $u$, $v$, one of $c_{ius}$ and $c_{ivs}$ is false. Thus, every clause of $R_8$ would be true, rendering the entire formula true. Therefore, $R_8$ is true if and only if no cop occupies two different vertices on the same turn. This means that there must be $|C|$ different cops placed, and thus every cop is placed once. □

Now we assemble the formulas for the cops and the robber:

| | |
|---|---|
| Robber's Formula: | $F_R = (R_2 \vee R_5 \vee R_{6a} \vee R_{6b})$ |
| Cops' Formula: | $F_C = (R_1 \wedge R_4 \wedge R_8)$ |

Combining these, we get the formula $F$ for our PARAMETERIZED QBFSAT$_t$ instance: $F = F_R \wedge F_C$.
Given the preceding sequence of proofs, we also have:

**Corollary 2.** *Our constructed* PARAMETERIZED QBFSAT$_t$ *instance encodes all the rules of* SHORT SEEDED PURSUIT EVASION.

Now we are ready to state the main result of this section.

**Theorem 3.** SHORT SEEDED PURSUIT-EVASION *is in AW[*].*

**Proof.** We show that $F$ is true if and only if the cops won the game, either by capturing the robber in a legal game or because the robber cheated.

As we have already verified the rules individually above, the robber's formula ($F_R = R_2 \vee R_5 \vee R_{6a} \vee R_{6b}$) is true if at least one of the following is true: he chooses to start somewhere other than his assigned starting position ($R_1$), performs an illegal move ($R_5$), or is caught ($R_{6a}$, $R_{6b}$). Otherwise, if the robber moves legally without being captured, $F_R$ evaluates to false for the corresponding variable setting. Similarly, the cops' formula ($F_C = R_1 \wedge R_4 \wedge R_8$) is false if and only if they break a rule. Thus formula $F$ is satisfied if the cops move legally ($F_C$ is true) and the robber is caught or he cheats ($F_R$ is true).

Conversely, $F$ is unsatisfied if the cops move illegally ($F_C$ is false) or the robber moves legally and is not caught ($F_R$ is false). Note that the formula is still unsatisfied if the cops cheat and the robber is captured since $F_C$ would be false under those conditions. The formula is polynomial in size, and does not introduce the input size into the parameters, so our reduction is complete. □

We can also use this to derive a result for SHORT PURSUIT-EVASION, defined as per SHORT SEEDED PURSUIT-EVASION except that the players choose their starting positions with their first moves.

**Lemma 10.** SHORT PURSUIT-EVASION *is in AW[*].*

**Proof.** The only difference between SHORT PURSUIT-EVASION and SHORT SEEDED PURSUIT-EVASION is that the former allows the players to choose starting positions while in the latter the starting positions are given as part of the input. As such, we can take the membership reduction for SHORT SEEDED PURSUIT EVASION and turn it into a membership reduction for SHORT PURSUIT-EVASION by removing the rules regarding starting positions from the formula $F$, namely $R_1$ and $R_2$. With these rules removed the players are free to select any starting position they wish on their first turns of the simulated game. Thus, we can reduce SHORT PURSUIT-EVASION to PARAMETERIZED QBFSAT$_t$ by reusing the reduction from SHORT SEEDED PURSUIT-EVASION, with the only change being that $F = (R_5 \vee R_{6a} \vee R_{6b}) \wedge (R_4 \wedge R_8)$. □

## 6. Directed pursuit-evasion

In this section we consider Directed Pursuit-Evasion, an unseeded variant of the Seeded Pursuit-Evasion problem where the edges of the input graph are directed and the players choose their starting positions. In [8], Goldstein and Reingold showed this problem to be hard for EXPTIME.

Directed Pursuit-Evasion
*Input*: directed graph $G = (V, E)$, positive integer $k$.
*Rules*:

- At the beginning of the game, each of the $k$ cops chooses a vertex of $G$ to be deployed to.
- Once the cops are deployed, the robber chooses a vertex in $V$ to start on.
- For the remainder of the game the cops and robber alternate taking turns, starting with the cops.
- During the cops' turn, each cop either moves to a vertex adjacent to his current position, or stays put. Multiple cops may occupy the same node at a time.
- On his turn, the robber either moves to an unoccupied vertex adjacent to his current position, or stays put.
- If, at any time, the robber occupies the same vertex as a cop, the robber is captured and the cops win.
- Both players have complete information. That is, the cops know the positions of the robber and their fellow cops at all times, and vice-versa.
- You cannot add or subtract pieces (cops or robbers) from the game at any time. Further, any given piece occupies exactly one vertex at a time.

*Parameter*: $k$
*Question*: Can the cops guarantee capture of the robber?

We show that this problem is AW[*]-hard by extending our previous reduction from Section 3. Specifically, we take the output instance from that reduction (from Unitary Parameterized QBFSAT$_2$ to Seeded Pursuit-Evasion) and create an instance of Directed Pursuit-Evasion.

Given a Seeded Pursuit-Evasion instance generated by our reduction, we add gadgets and direct the edges resulting in a Directed Pursuit-Evasion instance in which the players will lose if they do not move into the vertices corresponding to the starting position(s) of the Seeded Pursuit-Evasion instance. We avoid modifying the rest of the graph so that the remainder of the game plays out as before. The reduction creates $G'$ from $G$ as follows:

- Let $s$ be the vertex in $G'$ corresponding to the starting position in $G$ for the cop in the assignment gadget as given by the Seeded Pursuit-Evasion instance (see Fig. 3).
- Let $r'$ be the vertex between $r$ (the starting position of the robber in the original instance) and $s$ (again, see Fig. 3). Let $U = V - C - r'$.
- For each vertex $v \in C$ (the set of starting positions for the cops), change all the undirected edges incident to $v$ into out-edges.
- Create a set of new vertices $D$ with $|D| = |C|$.
- Add directed edges to $G'$ such that each vertex in $D$ has one out-edge to a unique vertex in $C$ (that is, a one-to-one mapping).
- Let $s'$ be the vertex in $D$ that is adjacent to $s$.
- Add directed edges from each vertex in $D - \{s'\}$ to every vertex in $U$.
- For each $v \in C - \{s\}$, create an escape subgraph called $R_1^v$. Let $R_1 = \{R_1^v : v \in C - \{s\}\}$.
- For each $v \in D - \{s'\}$, create an escape subgraph called $R_2^v$. Let $R_2 = \{R_2^v : v \in D - \{s'\}\}$.
- For each $R_1^v$ in $R_1$, pick a vertex $w \in R_1^v$ and add directed edges from $r'$ and $v$ to $w$.
- Add directed edges from $s'$ to every vertex in each escape subgraph in $R_1$.
- For each $R_2^v$ in $R_2$, pick a vertex $w \in R_2^v$ and add directed edges from $s$ and $r'$ to $w$.
- For each $R_2^v$ in $R_2$, add directed edges from $v$ to every vertex of $R_2^v$.

Fig. 6 illustrates how the components of $G'$ are connected.

Besides the graph, we also adjust the rest of the input from Seeded Pursuit-Evasion. While we do not input the set $C$, we set $k = |C|$. As the robber can choose his starting position, we omit vertex $a$ from the input.

**Theorem 4.** Directed Pursuit-Evasion *is AW[*]-hard.*

**Proof.** This reduction forces the cops to start on the vertices of $D$: If when the cops deploy, there is some vertex $v \in D$ which is is not occupied by a cop, then the robber can evade the cops indefinitely simply by starting on $v$ and never leaving that vertex. The cops cannot capture the robber if he uses this strategy since the vertices in $D$ have no in-edges.

If the cops do start on the vertices of $D$, the robber must start at $r'$ because every other vertex in the graph is adjacent to a starting position for a cop.

At this point, the cops must all move into $C$ on the same turn or else the robber can escape. If the *chaser cop* (the one starting at $s'$) moves to $s$ while some other cop stays in $D$ and thus leaves vertex $v \in C - \{s\}$ unoccupied, then $R_1^v$ will be unguarded and the robber can escape into it. On the other hand, if the cop on $v \in D - \{s'\}$ (for arbitrary $v$) moves into $C$

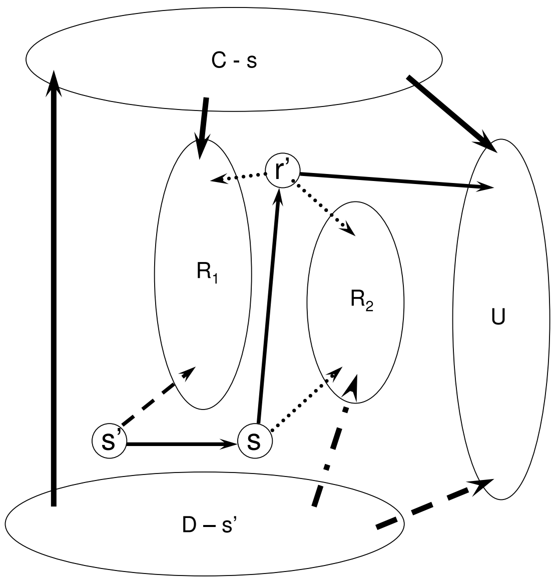

**Fig. 6.** Connections in $G'$. Solid arrows indicate a directed one-to-one mapping from the vertices in the origin set to the vertices in the destination set. Dashed arrows indicate that every vertex in the origin set has an edge to every vertex in the destination set. Dotted arrows indicate that each vertex in the origin set has one edge to each escape subgraph in the destination set. Arrows that alternate between dashes and dots indicate that each vertex in the origin set has an edge to every vertex of its unique escape subgraph in the destination set.

before the chaser cop moves to $s$, then the hypercube $R_2^v$ will be left unguarded. To prevent the robber from escaping via one of these two situations, the cops must either take their positions in $C$ all at once, or stay put. In the latter case, the robber can simply remain in his current position. In the former, the robber's only escape is to $r$.

Once the cops are on the vertices of $C$ and the robber is on $r$, we have reached the starting position of the Seeded Pursuit-Evasion instance. The remainder of the game plays out as it did before (since the vertices in $U$ have not gained any new out-edges). Thus we have a winning strategy in this instance if and only if we had one in the Seeded Pursuit-Evasion instance. This gives us a parameterized reduction from Unitary Parameterized QBFSAT$_2$ to Directed Pursuit-Evasion since $k = |C|$ and $|C| = r + 1$, showing that Directed Pursuit-Evasion is AW[*]-hard. □

### 6.1. *Short directed pursuit-evasion*

The last problem we consider is the short variant of Directed Pursuit-Evasion.

Short Directed Pursuit-Evasion
*Input*: directed graph $G = (V, E)$, positive integers $k$ and $t$.
*Rules*:

- At the beginning of the game, each of the $k$ cops chooses a vertex of $G$ to be deployed to.
- Once the cops are deployed, the robber chooses a vertex in $V$ to start on.
- For the remainder of the game the cops and robber alternate taking turns, starting with the cops.
- During the cops' turn, each cop either moves to a vertex adjacent to his current position, or stays put. Multiple cops may occupy the same node at a time.
- On his turn, the robber either moves to an unoccupied vertex adjacent to his current position, or stays put.
- If, at any time, the robber occupies the same vertex as a cop, the robber is captured and the cops win.
- Both players have complete information. That is, the cops know the positions of the robber and their fellow cops at all times, and vice-versa.
- You cannot add or subtract pieces (cops or robbers) from the game at any time. Further, any given piece occupies exactly one vertex at a time.

*Parameters*: $k$, $t$
*Question*: Can the cops guarantee capture of the robber within $t$ rounds?

We derive AW[*]-completeness for this problem using Theorem 4 and the techniques from Section 5.

**Theorem 5.** Short Directed Pursuit-Evasion *is AW[*]-complete.*

**Proof.** The proof of Lemma 9 shows that if an instance of Seeded Pursuit-Evasion admits a winning strategy for the cops, then the cops require at most $r + 1$ rounds to win. Our reduction to Directed Pursuit-Evasion starts from Seeded Pursuit-Evasion. The gadgets added by the reduction force the cops to spend exactly two additional rounds getting into the starting positions for the Seeded Pursuit-Evasion instance (one round to move into the new starting vertices $D$, and one round to move from the new starting positions into the old positions $C$), after which the game is played out as before. Therefore, if the cops have a winning strategy in the Short Directed Pursuit-Evasion instance they require at most $r+3$ rounds to win. This gives us a parameterized reduction, making Short Directed Pursuit-Evasion AW[*]-hard.

To show membership in AW[*], we reuse the reduction from Theorem 3 in Section 5. As we did before in Lemma 10, we omit the rules enforcing starting positions ($R_1$ and $R_2$) to enable the players to their starting positions on the first turn. For $R_4$ and $R_5$ (the movement rules) note that these have already been written in a manner suitable for directed graphs. The remaining rules do not concern edges or starting positions, and therefore function as intended without further modification, giving us $F = (R_5 \vee R_{6a} \vee R_{6b}) \wedge (R_4 \wedge R_8)$ in our modified reduction. This gives us a reduction from Short Directed Pursuit-Evasion to Parameterized QBFSAT$_t$. Thus, by this reduction, Short Directed Pursuit-Evasion is in AW[*]. □

## 7. Summary and open questions

We have shown that Seeded Pursuit-Evasion and Directed Pursuit-Evasion are both AW[*]-hard, and that short variants of both problems are AW[*]-complete. We also showed that Short Pursuit-Evasion is in AW[*]. Not yet mentioned are Directed Seeded Pursuit-Evasion and Short Directed Seeded Pursuit-Evasion, to which our Seeded Pursuit-Evasion and Short Seeded Pursuit-Evasion results apply respectively (as they are subproblems). Thus Directed Seeded Pursuit-Evasion is also AW[*]-hard and Short Directed Seeded Pursuit-Evasion is AW[*]-complete. We are, however, left with a number of open questions.

We can argue for Seeded Pursuit-Evasion and Directed Pursuit-Evasion that the number of positions in a game of either is $O(|V|^k)$. Standard game-theoretic algorithms can solve a game in a time which is polynomial in the number of valid game positions, so with such algorithms we can show these problems to be in XP. However, this leaves us with a gap between hardness and membership. What is the exact classification of these parameterized problems?

We know that Short Pursuit-Evasion is in AW[*]. Can we adapt the techniques we used in this paper to find an AW[*]-hardness reduction for Short Pursuit-Evasion as well? Or is the problem easier? Similarly, what about Pursuit-Evasion? We know that the problem is W[2]-hard [6], but given that so many variants of the problem are hard for AW[*] it seems likely that it is not in W[2] and thus we are interested in its exact characterization.

Finally, for symmetric graphs (such as the hypercubes we use), can we apply the local dominating set concept any further? Specifically, the lack of a local dominating set in a graph ensures that a robber can evade cops indefinitely, but is the existence of a local dominating set of size $k$ indeed sufficient to ensure that $k$ cops can capture the robber?

## References

[1] K. Abrahamson, R. Downey, M. Fellows, Fixed-parameter tractability and completeness IV: on completeness for W[P] and PSPACE analogues, Annals Pure Applied Logic 73 (1995) 235–276.
[2] A. Aigner, M. Fromme, A game of cops and robbers, Discrete Applied Mathematics 8 (1984) 1–11.
[3] B. Alspach, Searching and sweeping graphs: A brief survey, Le Matematiche 59 (2004) 5–37.
[4] R. Downey, M. Fellows, Parameterized Complexity, Springer, 1998.
[5] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer, 2006.
[6] F. Fomin, P. Golovach, J. Kratochvíl, On tractability of cops and robbers game, in: G. Ausiello, J. Karhumäki, G. Mauri, C.-H. Luke Ong (Eds.), IFIP TCS, in: IFIP, vol. 273, Springer, 2008, pp. 171–185.
[7] F. Fomin, D. Thilikos, An annotated bibliography on guaranteed graph searching, Theoretical Computer Science 399 (3) (2008) 236–245.
[8] A. Goldstein, E. Reingold, The complexity of pursuit on a graph, Theoretical Computer Science 143 (1995) 93–112.
[9] G. Hahn, Cops, robbers and graphs, Tatra Mountains Mathematical Publications 36 (2007) 1–14.
[10] R. Nowakowski, P. Winkler, Vertex-to-vertex pursuit in a graph, Discrete Mathematics 43 (1983) 235–239.
[11] A. Quilliot, Jeux et Points Fixes sur les graphes, Thèse de 3ème cycle, Université de Paris VI, 1978, p. 131–145.
[12] A. Quilliot, Problèmes de jeux, de point fixe, de connectivité et de représentation sur des graphes, des ensembles ordonnés et des hypergraphes. Thèse d'Etat, Université de Paris VI, 1983.
[13] A. Quilliot, A short note about pursuit games played on a graph with a given genus, Journal of Combinatorial Theory (B) 38 (1985) 89–92.
[14] A. Scott, Short pursuit-evasion, in: Texts in Algorithmics 7: Algorithms and Complexity in Durham 2006, 2006, pp. 141–152.
[15] P. Seymour, R. Thomas, Graph searching, and a min–max theorem for tree-width, Technical Report 89-1, The Center for Discrete Mathematics and Theoretical Computer Science, 1989.