Progress on a perimeter surveillance problem

Jeremy Avigad and Floris van Doorn

June 4, 2021

Abstract

We consider a perimeter surveillance problem introduced by Kingston, Beard, and Holt in 2008 and studied by Davis, Humphrey, and Kingston in 2019. In this problem, n drones surveil a finite interval, moving at uniform speed and exchanging information only when they meet another drone. Kingston et al. described a particular online algorithm for coordinating their behavior and asked for an upper bound on how long it can take before the drones are fully synchronized. They divided the algorithm's behavior into two phases, and presented upper bounds on the length of each phase based on conjectured worst-case configurations. Davis et al. presented counterexamples to the conjecture for phase 1.

We present sharp upper bounds on phase 2 which show that in this case the conjectured worst case is correct. We also present new lower bounds on phase 1 and the total time to synchronization, and report partial progress towards obtaining an upper bound.

1 Introduction

In 2008, Kingston, Beard, and Holt [8] considered a problem in decentralized control in which a group of small unmanned aerial vehicles (UAVs) or drones is required to surveil a linear interval with changing borders. In their model, the drones all move along the interval at the same uniform speed and can exchange information only when they meet. Because the borders of the interval and the number of operant drones can change over time, the drones have imperfect information as to the global state.

Kingston et al. described an algorithm for coordinating the drones and considered the problem of bounding the time to synchronization in a setting where the parameters remain fixed. They divided the algorithm's behavior into two phases which we will call *phase 1* and *phase 2*. Normalizing units so that a single drone can traverse the interval in one unit of time, they claimed an upper bound of 3 units of time for phase 1 and an upper bound of 2 units of time for phase 2. In each case, the bounds were based on the behavior of what they took to be the worst-case starting configurations.

In 2019, Davis, Humphrey, and Kingston [5] pointed out that the previous work did not justify the claimed characterizations of the worst-case behavior, and, in fact, they provided a counterexample to the bound for phase 1. As a result, there is currently no rigorously established bound on the time to synchronization that is independent of the number of drones.

We describe the problem more precisely in Section 2 and discuss the previous results in greater detail. In Section 3, we present sharp upper bounds on the length of phase 2, showing that the originally claimed worst-case behavior is correct. In Section 4, we present improved lower bounds on the length of phase 1 as well as the total time to synchronization. Finally, in Section 5, we present some partial results towards bounding the length of phase 1.

It is by now well understood that decentralized coordination of UAVs raises interesting combinatorial challenges [1, 3, 9]. What the Kingston–Beard–Holt example shows is that difficult combinatorial problems arise even when dealing with fairly simple models, and that new mathematical ideas and techniques are needed to handle them.

David Greve at Collins Aerospace has independently obtained a substantially different proof of our Theorem 2.1 with 2 replacing $2 - 1/n$ [6], and he has recently formalized the proof presented here (personal communication) in the ACL2 verification system [2].

2 The problem

To describe the model under consideration, it is convenient to normalize units so that the drones are surveilling the unit interval $[0, 1]$ and moving at a velocity of one unit per unit time. At each point in time, each drone has a direction $d = \pm 1$, where 1 indicates that the drone is moving to the right and -1 indicates that it is moving to the left. Each drone also has an estimate of the form $((a, \ell), (b, m))$ where a is the left endpoint of the interval, ℓ is the number of drones to the left, b is the right endpoint, and m is number of drones to the right. Kingston et al. wanted to consider a scenario where the data keeps changing, so these estimates may be wrong; in particular, a and b do not need to be in the unit interval. Each drone recognizes the leftmost or rightmost border when it reaches it. Two drones can only exchange information when they meet, that is, occupy the same position in the interval.

Suppose we have n drones on the unit interval, numbered from left to right $1, \ldots, n$. The ith drone's *left endpoint* is $(i - 1)/n$, its *right endpoint* is i/n, and its *interval* is the closed interval with those endpoints. We say the *common endpoint* of drones i and $i + 1$ is the right endpoint of drone i, which is equal to the left endpoint of drone $i + 1$. The desired situation is that each drone remains in its interval, moving back and forth between its left and right endpoints.

Kingston et al. proposed the following algorithm to attain this behavior. Write $(\alpha, \beta) = ((a, \ell), (b, m))$ for the drone's estimates. Based on this data, each drone can calculate the interval it *thinks* it is supposed to surveil as follows:

- The size of the interval is $I(\alpha, \beta) = (b - a)/(\ell + m + 1)$, that is, the length of the estimated interval divided by the number of drones.
- The left endpoint is

$$\begin{aligned} L(\alpha, \beta) &= a + \ell I(\alpha, \beta) = b - (m + 1)I(\alpha, \beta) \\ &= \frac{a(m + 1) + b\ell}{\ell + m + 1}. \end{aligned}$$

- The right endpoint is

$$\begin{aligned} R(\alpha, \beta) &= a + (\ell + 1)I(\alpha, \beta) = b - mI(\alpha, \beta) \\ &= \frac{am + b(\ell + 1)}{\ell + m + 1}. \end{aligned}$$

According to the algorithm, each drone continues in the direction it is moving until one of these events occurs:

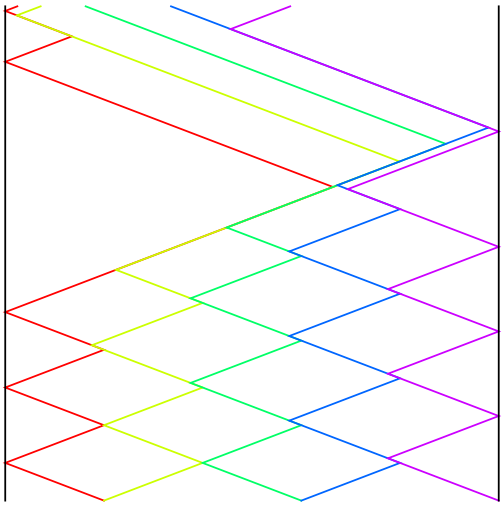

Figure 1: A sample run of the algorithm with $n = 5$ drones.

- If a drone hits the left border, it updates its left estimate with the correct left endpoint of the interval and the fact that there are no drones to the left, and then it turns around. The case where a drone hits the right border is handled similarly.
- If two drones meet, the left one adopts the right estimate from the drone to its right (adding 1 to the number of drones to the right), and vice-versa. As a result, the two drones agree as to their estimates of the intervals they are supposed to surveil. The two then set their directions so that they are headed to their common endpoint.
- If two drones are traveling together (with consistent estimates) and reach their common endpoint, they split, i.e. one of them reverses direction in order to stay in its estimated interval.

We assume that when two or more drones start together, they all share their estimates at time 0. We call the first type of event a *border* event, the second type of event a *meet* event, and the third type of event a *separation* event. Notice that, as a special case, the second and third can happen simultaneously, if two drones meet at their common endpoint. We call that a *bounce* event. Fig. 1 depicts a sample run of the algorithm with five drones, with the interval extending left to right and time flowing downward.

The arguments below can be made rigorous by formalizing the notion of a *configuration* (that is, a time, t, and the sequence of positions, directions and estimates of all the drones at that time), and, given a configuration, the next configuration at which one of the events above occurs. It then makes sense to talk about the sequence of eventful configurations from a given start configuration, and all the informal claims below can be interpreted in terms of that.

One should also establish that the algorithm does not exhibit *Zeno* behavior, i.e. that for every time t there is a finite sequence of events that extends past time t. To prove this, suppose otherwise, consider the infinite sequence of events determined by the algorithm, and let T be the least upper

bound on their times. Then for every $\varepsilon > 0$, there are infinitely many events in the time interval $(T - \varepsilon, T)$. Since there are only finitely many drones, at least one drone has to be involved in infinitely many events, which means that one drone has to change direction infinitely many times. But it is not hard to show that if drone $i + 1$ makes two consecutive left turns, then drone i must turn right in the interim; so if drone $i + 1$ changes direction infinitely often, then so does drone i. With the symmetric argument, it therefore follows that *all* the drones change direction infinitely often in the interval $(T - \varepsilon, T)$. But now if we take $\varepsilon < 1/2n$, then at time $T - \varepsilon$ either there is a pair of drones i and $i + 1$ that are not within 2ε of each other, or the leftmost drone is not within ε of the left border, or the rightmost drone is not within ε of the right border. In the first case, drone i can turn at most once in $(T - \varepsilon, T)$, and in the last two cases, the relevant drone can turn at most once. Thus we have contradicted the assumption that the algorithm exhibits Zeno behavior.

Given a particular start configuration, say a drone is *left synchronized* at time t if beyond that point it never goes to the left of its left endpoint, and similarly for *right synchronized*. A drone is *synchronized* at time t if it is left and right synchronized. Kingston et al. conjectured the following:

Conjecture 2.1. *From any start configuration, all drones have correct estimates by time 3.*

Conjecture 2.2. *If all drones have correct estimates at time t, then all drones are synchronized by time $t + 2$.*

We call the time between the start and the moment that all drones have correct estimates *phase 1*, and the time after phase 1 until the moment that the drones are synchronized *phase 2*.

Kingston et al. sketched a proof of each conjecture, in each case based on a claims that a certain start configurations gave rise to the worst-case behavior. The two conjectures imply that for any start configuration, the drones are synchronized by time 5.

Davis et al. showed that Conjecture 2.1 is false, by exhibiting counterexamples with $n = 3$ that require up to $3 + 1/2$ units of time before all the drones have correct estimates. For that purpose, they used the AGREE model checker [4], which required fixed bounds on all the parameters. In particular, they had to limit the estimates of the number of drones to the left or right at 20. With those restrictions, the tool reported upper bounds of $3 + 2/3$ on the time until all three drones have complete information, and $4 + 1/3$ units of time until full synchronization. The tool also reported absolute upper bounds of 2 on phase 2, with $n \leq 6$; they report that the verification for $n = 6$ required about 20 days of computation using 40 cores. They do not report any results for larger n. In particular, there was no rigorously established bound on the length of either phase, or total time to synchronization, that is independent of n.

Conjecture 2.2 is clearly implied by the following statement: if all drones start with correct estimates then they are synchronized by time 2. The implication follows, because we can consider the configuration at time t as the new start configuration. In Section 3, we prove the following:

Theorem 2.1. *Assuming all the drones have the correct estimates, they are all synchronized at time $2 - 1/n$.*

This shows that the conjecture by Kingston et al. as to the worst-case configurations is correct. In Section 3, we also obtain the following additional information:

Theorem 2.2. *If all drones start with incorrect estimates, and they all have correct estimates at time t, then all drones are synchronized by time $t + 1 - 1/n$.*

In Section 4, we improve the lower bounds as follows:

Theorem 2.3. *For every $n \geq 3$ and $\varepsilon > 0$, there is a start configuration such that drones do not have correct estimates before time $4-1/n-\varepsilon$, and are not fully synchronized before time $5-3/n-\varepsilon$.*

We mention in passing that the case $n = 1$ is trivial; a single drone is already synchronized, though it may not have correct estimates until time 2. The case $n = 2$ is also easy to analyze; drones have correct estimates by time 2 and are synchronized by time $2 + 1/2$. Both these bounds are sharp, which can be seen by having both drones start together near the left border of the interval, moving right, and having them separate near the right border of the interval. So $n = 3$ is the first interesting case.

It is important to note that Kingston et al. were not looking for an algorithm to synchronize all the drones as quickly as possible. For that, having all drones move all the way to the left to get the correct information about the left border and then move all the way to the right does better than the one proposed. Rather, they were independently interested in the behavior of that particular algorithm for updating information in the face of changing borders and addition or subtraction of drones. Given that, the question about worst-case behavior even under fixed conditions is natural.

The description of the algorithm leaves two things unspecified. First, it does not specify whether the information that each drone has must be consistent with its current position. For example, it does not specify whether a drone can think that the right border is at position 0.8 when the drone itself it is at position 0.9. Second, it does not specify what happens when a group of three or more drones come together and determine that three of them are within the middle drone's interval; in that case, the middle drone can escort either neighbor to their common border. Neither of these issues bears on the results reported below, since our upper bound only concerns phase 2, where these issues do not arise, and our lower bounds meet the more stringent requirement that all drones have information consistent with their positions. Regarding the first issue, we note in passing that one can show that if the drones start with consistent information, temporary inconsistencies do not affect the behavior of the algorithm. Regarding the second issue, we note that the strongest upper bound will allow for nondeterminism and allow the middle drone to go to either endpoint.

3 An upper bound on phase 2

Once the drones have the correct estimates as to the left and right endpoints and the number of drones on either side, each drone knows its proper interval, and the behavior of the algorithm from that point on can be described more simply: when two drones meet, they escort each other to their common endpoint and then separate. Our goal is to prove Theorem 2.1, which guarantees that the drones are all synchronized within $2 - 1/n$ units of time.

By symmetry, it suffices to show that all the drones are left synchronized by time $2 - 1/n$. Kingston, Beard, and Holt [8] gave a short argument that all drones are eventually left synchronized, although the bound that is implicit in that argument is linear in n. The argument goes as follows: suppose at some time, t, drones $1, \ldots, j$ are left synchronized. Eventually, drone j will meet drone $j + 1$, and then they will travel to their common endpoint and separate. It suffices to show that at this point, $j+1$ in left synchronized, because then by induction we have that all drones are eventually left synchronized. We present their proof of this in Lemma 3.2. Our proof of Theorem 2.1 is based on a subtle refinement of their argument.

Lemma 3.1. *Suppose that at time t drone j is moving to the right. Then the next time drone j changes direction, it is at or to the right of its right endpoint. The same is true with "right" replaced by "left."*

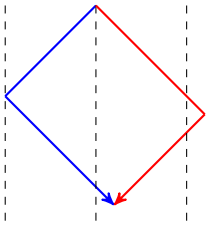

Figure 2: A picture proof of Lemma 3.2. The two intervals between the vertical lines indicate the intervals of drone j (blue) and drone $j+1$ (red).

Proof. If drone $j < n$ is moving to the right, the only two events in which it can change direction is when meeting drone $j+1$ or separating from drone $j+1$. If the next time drone j turns left is when meeting drone $j+1$, then at that point they are to the right of their common endpoint, which is the right endpoint for drone j. If the next time drone j turns left is when separating from (or bouncing off) drone $j+1$, then at that point they must both be at their common endpoint.

If drone n is moving to the right, the only event in which it changes direction is a border event, which is at its right endpoint. The last observation follows by symmetry. □

Lemma 3.2. *Suppose at time t, drone j is left synchronized and drone j and $j+1$ separate at their common endpoint. Then drone $j+1$ is left synchronized at time t.*

Proof. By induction, we show that every subsequent meet, bounce, and separation event involving j and $j+1$ occurs to the right of their common endpoint.

After the separation, drone j is moving to the left. By Lemma 3.1, the next time it changes direction, it is at or to the left of its left endpoint. Since it is left synchronized, we know it is *at* its left endpoint. Similarly, after the separation, drone $j+1$ is moving to the right, and the next time it changes direction is it at or to the right of its right endpoint. So the next time drone j and $j+1$ meet, they are at or to the right of their common endpoint, since drone $j+1$ must has taken at least as long to turn around as drone j; see Fig. 2 for a visual depiction. They then travel left to their common endpoint and separate, and the situation repeats. □

We now draw out two useful consequences of Lemma 3.2:

Lemma 3.3. *Suppose at time t drone j and $j+1$ are together moving to the left, and drone j is left synchronized. Then drone $j+1$ is also left synchronized at time t.*

Proof. If they are together and moving to the left, they are to the right of their common endpoint. Eventually they will reach their common endpoint and separate, say, at time t'. By Lemma 3.2, drone $j+1$ is left synchronized at time t'. But since drone $j+1$ is to the right of its left endpoint between time t and t', it is in fact left synchronized at time t. □

Lemma 3.4. *Suppose at time t drone $j < n$ is at or to the right of its right endpoint, moving right, and left synchronized. Then drone $j+1$ is left synchronized at time t.*

Proof. Since drone j is moving right and is to the right to its right endpoint, it cannot be together with drone $j+1$. Eventually drone j and $j+1$ will meet to the right of their common endpoint, say at time t', and then they will move left together. At that point, by Lemma 3.3, drone $j+1$ is left synchronized. Since drone $j+1$ is to the right of its left endpoint between time t and t', it is already left synchronized at time t. □

We have now arrived at the key refinement of the argument by Kingston et al.Ẇe say that drones j and $j+1$ *have met* by time t if either they started together, moving in the same direction, or they have been involved in a meet or bounce event. It turns out that we have much more information about the behavior of the drones once this is the case. Fortunately, it is not hard to show that this happens within one unit of time, for all the drones uniformly.

Lemma 3.5. *For every $j < n$, drones j and $j+1$ have met by time 1.*

Proof. Intuitively, the worst case is where j and $j+1$ start close together with j moving to the left and $j+1$ moving to the right. Eventually, j turns around at or before it reaches 0 and $j+1$ turns around at or before it reaches 1, and then j and $j+1$ will meet. At that point, together they have traveled at most the twice the length of the interval, which means that each one has traveled at most one unit of distance.

We can make this argument more rigorous as follows. Suppose drone j starts at position x and drone $j+1$ starts at position $y \geq x$. Furthermore, let w be the position of drone j when it first moves right (so $w = x$ if j starts moving right, and otherwise w is the position where drone j first turns around), and let z be the position of drone $j+1$ when it first moves left. Then the total distance traveled by both drones before they meet is $2(z-w)-(y-x)$, which means the drones meet at time $z - w - (y-x)/2 \leq z - w \leq 1$. □

Lemma 3.6. *Suppose that at time t, drone j is moving to the left and drone $j+1$ is not together with drone j. Suppose also that j and $j+1$ have met by time t. Let t' be the last time before time t that drones j and $j+1$ bounced or separated. Then j has been moving left since time t'.*

Proof. If at some point between t' and t drone j was moving to the right, something must have turned it to the left. But that can only have been a meet or separation or bounce event. If it was a meet event, the fact that j and $j+1$ are not together at time t means there was also a separation event. Both situations contradict the fact that t' is the last time before time t that drones j and $j+1$ bounced or separated. □

Lemma 3.7. *Suppose $j < n$ and at time t, drones $1, \ldots, j$ are left synchronized and drone j and $j+1$ have met. Then at time $t + 1/n$, drones $j+1$ is left synchronized as well.*

Proof. Suppose drone j is left synchronized. If it is moving to the right, it will be at or to the right of its right endpoint within time $1/n$, possibly having met drone $j+1$ along the way. At that point drone $j+1$ is left synchronized, by Lemmas 3.2 and 3.4. If drone j is moving to the left and it is together with drone $j+1$, drone $j+1$ is left synchronized at time t by Lemma 3.3.

Finally, suppose drone j is moving to the left and is not together with drone $j+1$. Since we are assuming drones j and $j+1$ have met by time t, there is a $t' < t$ where drones j and $j+1$ bounced or separated last. By Lemma 3.6, drone j has been moving left since time t'. Since drone j is at or to the right of its left endpoint at time t, it was to the right of its left endpoint between time t' and t. Since drone j is left synchronized at time t, this shows that it was already left synchronized at t'. By Lemma 3.2, drone $j+1$ was also left synchronized at time t', and hence is left synchronized at time t. □

Since drone 1 is always left synchronized and all the drones have met by time 1, by induction on $i < n$ we have that drones $1, \ldots, i$ are left synchronized at time $1 + (i-1)/n$. Taking $i = n$ yields Theorem 2.1.

It is not hard to show that Theorem 2.1 is sharp. To attain the worst-case behavior, let all n drones start arbitrarily close to the left border, moving right independently. After close to one unit of time they reach the right border, at which point the rightmost drone turns left and quickly meets all the others. The group then moves to the left, with each drone separating from the group at its left endpoint. Drones 1 and 2 separate at their common endpoint at time arbitrarily close to $2 - 1/n$, at which point all the drones are synchronized. This is exactly the worst-case scenario presented by Kingston et al.

We end this section by proving Theorem 2.2. In that theorem we assume that all drones start with incorrect information. Without such an assumption, the theorem is false; for example, if all drones start with correct information, then the fact that Theorem 2.1 is sharp means that Theorem 2.2 does not hold in that case. However, the condition is not very strong, and we can find worst-case configurations for phase 1 where this condition holds, since we can modify the correct information by adding a small amount to the estimated left and right endpoints of the interval.

Lemma 3.8. *Suppose all drones start with incorrect information, $j < n$, and at time t all drones have correct information and drones $1, \ldots, j$ are left synchronized. Then at time $t + 1 - j/n$, all drones are synchronized.*

Proof. The only way that all drones have correct information at time t is that drone 1 hits the left border, and then this correct left information propagates through all drones to drone n. Similarly, the correct right information has propagated through all drones from drone n to drone 1. For two consecutive drones i and $i+1$ this means that at some time $t' \leq t$ they were together with both correct left and correct right information. Any time after t' the drones i and $i+1$ had correct information. Inspecting the proof of Lemma 3.7, we see that it also holds in this case (the lemmas in this section only use that drones j and $j+1$ have correct information, not that the other drones have correct information). Therefore, by induction we see that if drones $1, \ldots, j$ are left synchronized, then at time $t + 1 - j/n$, all drones are synchronized. □

Since drone 1 is always left synchronized, this implies Theorem 2.2. In Section 4 we present an example where all drones get correct information at time $4 - 1/n$, but drone 2 is already left synchronized at that time. Theorem 2.2 then guarantees that the drones will be synchronized at time $(4 - 1/n) + (1 - 2/n) = 5 - 3/n$.

4 Lower bounds

Recall that Davis, Humphrey, and Kingston [5] produced counterexamples to Conjecture 2.1 by presenting configurations of three drones that do not all have correct information about the state of affairs until time $3 + 1/2$. In this section, we improve this lower bound for three drones to $3 + 2/3$ and modify their example to provide new lower bounds for every $n \geq 3$. Our strategy is similar to theirs. We let all the drones start near the left border of the interval, moving to the right. We put the drones in groups that separate just before they hit the right border, so that most drones in each group do not learn the information about the right border from the group to their right. After this, the drones will move back in groups to the left border, and then we try to repeat this

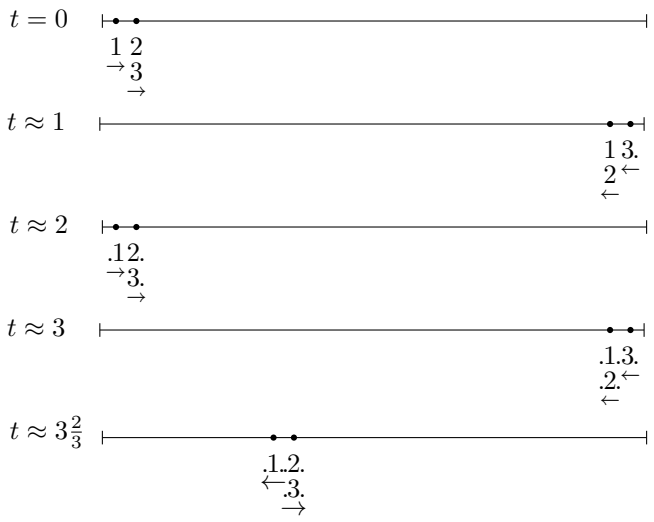

Figure 3: A depiction of the worst-case scenario for phase 1 that we found for $n = 3$. A dot to the right (left) of a drone indicates that it has correct right (left) information.

process: let all groups separate just before they hit the left border, and then the drones move to the right again.

It is conceivable that things can be arranged so that n drones shuttle back and forth on the order of n times before information about the right border has propagated all the way to the left and vice-versa. To rule out such a possibility we have to take into consideration the specific algebraic calculations described in Section 2, and the extent to which they constrain the drones' estimates. In Section 5, we consider a simplified version of the calculation, and show that, with that version (which does not directly apply to the original problem), the type of behavior described above cannot occur. But our counterexamples show that with the actual calculations presented in Section 2, we can obtain slightly worse behavior. In particular, it can take close 4 units of time before the drones have correct estimates.

Let us start with the smallest interesting situation, with $n = 3$ drones. Our counterexample is summarized in Fig. 3.

- The drones start near the left border, moving to the right, with drones 2 and 3 moving as a group.
- Just before the drones hit the right border, drones 2 and 3 separate, and drone 2 meets drone 1. Drone 3 hits the border, learning the true position of the right border, and all drones move to the left.
- Just before the drones hit the left border, drones 1 and 2 separate, and drone 2 meets drone 3 (learning the true position of the right border). Drone 1 hits the border, learning the true position of the left border, and all drones move to the right again.
- This process repeats at the right border, after which drones 1 and 2 have complete knowledge

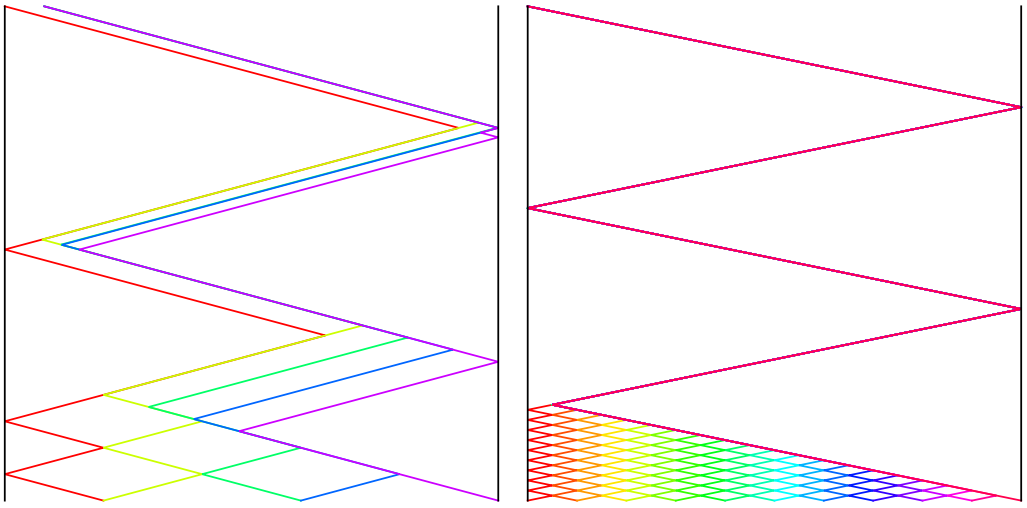

Figure 4: A depiction of the worst-case scenario we found for $n = 5$ (left) and $n = 20$ (right). In the left diagram $N = 25$ and $N = 10^6$ in the right diagram. In the left diagram N is chosen to be small to show the behavior better, but this choice of N gives rise to an artifact that doesn't happen for high N, which is that drone n turns around a second time around time 1.

of the left and the right border.

- Drones 1 and 2 move to their actual common endpoint, and separate. Then drone 3 also learns the true left endpoint, and moves to its common endpoint with drone 2.

To achieve this, choose N large, and let $\delta = 1/N$. Let drone 1 start at $x_1 = 0$ and drone 2 start at $x_2 = \delta$, both with direction $d_i = 1$. The initial information for drones 1 and 2 are as follows:

$$((a_1, \ell_1), (b_1, m_1)) = ((-2 + 4\delta, N), (1, N))$$
$$((a_2, \ell_2), (b_2, m_2)) = ((0, N), (2 - 2\delta, N))$$

The initial information for drone 3 is fully determined by the fact that it starts in the same group as drone 2:

$$x_3 = x_2$$
$$((a_3, \ell_3), (b_3, m_3)) = ((a_2, \ell_2 + 1), (b_2, m_2 - 1)).$$

This initial information results in the behavior in Fig. 3. Drone 2 turns around at positions $1 - \delta$, then δ, then $1 - 3\delta$ and then $1/3$. Only after the last time that drone 2 turns around do all drones have correct estimates. This means that by taking N large enough, it can require (arbitrarily close to) $3 + 2/3$ units of time before every drone has correct estimates. It takes another $1/3$ unit of time until drone 3 is synchronized. Therefore, the drones need time (arbitrarily close to) 4 before all drones are synchronized.

We can generalize this by adding more drones at the same position as drone 2 and 3. If there are more than 2 drones, the positions at which the intermediate drones turn are still roughly the

same, except that the fourth turn is at $x = 1/n$. This means that it takes time (arbitrarily close to) $4 - 1/n$ until all drones have correct estimates. After that it takes time $1 - 2/n$ until drone n is synchronized, for a total of (arbitrarily close to) $5 - 3/n$ for all drones to be synchronized.

The situation is displayed in Fig. 4 for $(n, N) = (5, 25)$ and $(n, N) = (20, 10^6)$. In these diagrams every drone has a unique color (though if drones travel close to each other only the rightmost drone is drawn), time flows down along the vertical axis, and the interval is on the horizontal axis.

We have explored other methods for finding lower bounds with more than 3 drones using Mathematica [7]. One method we used was to manually pick all starting points and desired points of separation, enter all constraints into Mathematica, and use Mathematica's FindInstance function or Reduce function to find a solution. However, this problem quickly becomes intractable for these functions because of the subtle interplay between the equations like $R(\alpha_2, \beta_2) = 1 - \delta$ and the inequalities like $a_i \le x_i \le b_i$ and $m_i \ge 0$. We resorted to heuristics, strategically choosing some values and letting Mathematica solve for the remaining variables. For example, we often set the estimated number drones to the left or right to a specific drone to be fixed large integers. We have found an example with five drones in three groups, which also takes time roughly time $4 - \frac{1}{5}$ for phase 1, and time $5 - \frac{3}{5}$ for phase 1 + 2, similar to our previous example. Drone 1 starts alone at position 0, drones 2 and 3 start together at position $\varepsilon = \frac{1}{100}$ and drone 4 and 5 start together at position 2ε. The initial estimates for drones 1, 2 and 4 are

$$(\alpha_1, \beta_1) = ((-250098, 10^8), (1, 10^6)$$
$$(\alpha_2, \beta_2) = ((-249999, 10^8), (2501, 10^6))$$
$$(\alpha_4, \beta_4) = ((-232446, 10^8), (13.94, 5569))$$

The initial estimates for drones 3 and 5 are determined by those of drones 2 and 4. The resulting diagram, not shown, is similar to that of Fig. 4.

5 Towards an upper bound on phase 1

The results described in Section 4 involve crafting examples where incorrect estimates on the part of the drones lead groups of drones to misjudge their common endpoints and shuttle back and forth across the interval. From a combinatorial perspective, it is not hard to imagine sequences of events where n drones keep regrouping in pairs and traveling back and forth up to n times before all the border information has propagated to all the drones. The question is whether the algebraic calculations of the common endpoints make it possible to realize this behavior. Getting a bound on phase 1 that is independent of n requires ruling this out.

In this section, we take small steps towards obtaining a better understanding of the algebraic constraints. If $\alpha = (a, \ell)$ is a pair consisting of a real number and a nonnegative integer, we will write α^+ for $(a, \ell + 1)$ and α^- for $(a, \ell - 1)$. Remember that a pair (α, β) represents a drone's estimates as to the left border, the number of drones to the left, the right border, and the number of drones to the right. Remember also that we write $L(\alpha, \beta)$ for the left endpoint of the drone's interval based on that estimate, and $R(\alpha, \beta)$ for the right endpoint. We will also write 0 for the estimate $(0, 0)$ adopted by the leftmost drone when it reaches the left border, and 1 for the estimate $(1, 0)$ adopted by the rightmost drone when it reaches the right border.

Consider the example in Fig. 3. The estimates of the three drones on each line can be represented

as follows:

(α_1, β_1)	(α_2, β_2)	(α_2^+, β_2^-)
(α_1, β_2^+)	(α_1^+, β_2)	$(\alpha_2^+, 1)$
$(0, \beta_2^+)$	$(\alpha_1^+, 1^+)$	$(\alpha_1^{++}, 1)$
$(0, 1^{++})$	$(0^+, 1^+)$	$(\alpha_1^{++}, 1)$
$(0, 1^{++})$	$(0^+, 1^+)$	$(0^{++}, 1)$

What makes the example effective is that:

- $R(\alpha_2, \beta_2) = L(\alpha_2^+, \beta_2^-)$ is close to 1.
- $R(\alpha_1, \beta_2^+) = L(\alpha_1^+, \beta_2)$ is close to 0.
- $R(\alpha_1^+, 1^+) = L(\alpha_1^{++}, 1)$ is close to 1.
- $R(0, 1^{++}) = L(0^+, 1^+) = 1/3$.

To understand the extent to which we can or cannot improve the lower bound, we need to understand the constraints that arise when drones share information in such a way.

The $+1$ and -1 terms make calculation more difficult, so we focus on a simpler approximation to the problem. Given $\alpha = (a, \ell)$ and $\beta = (b, m)$, write

$$P(\alpha, \beta) = \frac{am + b\ell}{\ell + m}$$

for an approximation to $L(\alpha, \beta)$ and $R(\alpha, \beta)$. In other words, we ignore the terms $+1$ in the calculations in Section 2, which is reasonable when ℓ and m are large.

Now, suppose three groups of drones start with estimates that are roughly (α_1, β_1), (α_2, β_2), and (α_3, β_3). Sharing information between the first two yields roughly (α_1, β_2), and sharing information between the second two yields roughly (α_2, β_3). Sharing information between these again yields roughly (α_1, β_3). Under these simplifications, we can ask the following question: are there choices of (α_1, β_1), (α_2, β_2) and (α_3, β_3) such that

- $P(\alpha_1, \beta_1)$, $P(\alpha_2, \beta_2)$, and $P(\alpha_3, \beta_3)$ are close to 1,
- $P(\alpha_1, \beta_2)$ and $P(\alpha_2, \beta_3)$ are close to 0, and
- $P(\alpha_1, \beta_3)$ is close to 1?

The following theorem shows that the answer is negative.

Theorem 5.1. *If*

$$\max(P(\alpha_1, \beta_2), P(\alpha_2, \beta_3)) \leq \min(P(\alpha_2, \beta_2), P(\alpha_1, \beta_3))$$

then

$$P(\alpha_1, \beta_2) = P(\alpha_2, \beta_3) = P(\alpha_2, \beta_2) = P(\alpha_1, \beta_3)$$

In this theorem only the four values $P(\alpha_2, \beta_2)$, $P(\alpha_1, \beta_2)$, $P(\alpha_2, \beta_3)$, and $P(\alpha_1, \beta_3)$ are mentioned, and the constraints are weaker than stated in the original question. We provide three proofs of this theorem.

First proof. The statement $P(\alpha, \beta) = c$ is equivalent to $(c - a)/\ell = (b - c)/m$. If we ignore the length of the drone's own interval, $(c - a)/\ell$ is the length of the interval for each drone to the left of a, and $(b - c)/m$ is the length of the interval for each drone to the right of b. So the statement $P(\alpha, \beta) = c$ says that c has the property that these two lengths are the same.

Let $v(\alpha, c) = (c - a)/\ell$ and $w(\beta, c) = (b - c)/m$, so $v(\alpha, c)$ is strictly increasing in c and $w(\beta, c)$ is strictly decreasing in c. Write

$$\begin{aligned} P(\alpha_1, \beta_3) &= r_1, & P(\alpha_2, \beta_2) &= r_2, \\ P(\alpha_1, \beta_2) &= \ell_1, & P(\alpha_2, \beta_3) &= \ell_2. \end{aligned}$$

The hypothesis of the lemma is that $\ell_i \le r_j$ for each i and j. We have the following chain of inequalities:

$$\begin{aligned} v(\alpha_2, r_2) &= w(\beta_2, r_2) \le w(\beta_2, \ell_1) \\ &= v(\alpha_1, \ell_1) \le v(\alpha_1, r_1) \\ &= w(\beta_3, r_1) \le w(\beta_3, \ell_2) \\ &= v(\alpha_2, \ell_2) \le v(\alpha_2, r_2). \end{aligned}$$

Therefore, all inequalities must be equalities, and the fact that v and w are strictly monotone implies $\ell_2 = r_2 = \ell_1 = r_1$. □

Second proof. Write

$$I'(\alpha, \beta) = \frac{b - a}{\ell + m}$$

for the length of each drone's interval based on the information α, β, and notice that

$$P(\alpha, \beta) = a + \ell I'(\alpha, \beta) = b - m I'(\alpha, \beta). \qquad (\star)$$

This enables us to do comparisons when either α or β does not change:

- From $P(\alpha, \beta) \le P(\alpha, \beta')$ we can infer $I'(\alpha, \beta) \le I'(\alpha, \beta')$.
- From $P(\alpha, \beta) \le P(\alpha', \beta)$ we can infer $I'(\alpha', \beta) \le I'(\alpha, \beta)$.

So from

$$\begin{aligned} P(\alpha_1, \beta_2) &\le P(\alpha_1, \beta_3), & P(\alpha_2, \beta_3) &\le P(\alpha_1, \beta_3), \\ P(\alpha_2, \beta_3) &\le P(\alpha_2, \beta_2), & P(\alpha_1, \beta_2) &\le P(\alpha_2, \beta_2) \end{aligned}$$

we can conclude

$$\begin{aligned} I'(\alpha_1, \beta_2) \le I'(\alpha_1, \beta_3) \le I'(\alpha_2, \beta_3) \le I'(\alpha_2, \beta_2) \\ \le I'(\alpha_1, \beta_2) \end{aligned}$$

and so all these quantities are the same. Using ($\star$), this means that any two expressions $P(\alpha_i, \beta_j)$ that have either α_i or β_j in common are equal, so we have $P(\alpha_1, \beta_2) = P(\alpha_1, \beta_3) = P(\alpha_2, \beta_3) = P(\alpha_2, \beta_2)$, as required. □

We are grateful for Reid Barton for providing us with the following physical interpretation.

Third proof. If we think of $\alpha = (a, \ell)$ as representing a mass of $1/\ell$ at position a on the real number line and $\beta = (b, m)$ as representing a mass of $1/m$ at position b, then $P(\alpha, \beta)$ is their combined center of mass. The hypothesis of the lemma says that the center of mass of α_1 and β_2 and the center of mass of α_2 and β_3 are both less than the center of mass of α_2 and β_2 and the center of mass of α_1 and β_3.

But the combined the center of mass of the first two pairs and the center of mass of the second two pairs are both equal to the center of mass of α_1, α_2, β_2, and β_3 all together. Since two objects combined have a center of mass between the center of mass of each, all four quantities have to be equal. □

Unfortunately, the omissions of the +1s in the definition of $P(\alpha, \beta)$ are an oversimplification. When we restore them to the model, the example at the end of Section 4 is a counterexample to statements analogous to Theorem 2.3. So we still need to both refine our calculations and figure out how to use them to obtain an upper bound to phase 1.

6 Conclusions

These problems are harder than we thought it would be. Even bounding phase 2 behavior was difficult, since it was hard to control the combinatorial explosion of possibilities induced by the discrete decision points in the algorithm. A natural strategy is to look for a complexity measure and show that it is decreasing over time, but we were unable to find such a measure. Another idea is to argue that configurations can, without loss of generality (for example, without decreasing the time to synchronization), be reduced to simpler ones, possibly in a larger space of states and actions. We also tried using induction on the number of drones, or focusing on some salient feature of the event history. We could not get any of these strategies to work. It was surprising to us that the argument presented in Section 3 works, because it involves ignoring everything about the first unit of time other than the fact that each pair of drones has met. We stumbled upon it only by finding a more complicated argument and simplifying it to the shorter one, and then we determined that the longer argument was incorrect.

Bounding phase 1 promises to be even harder, since it requires taking the algebraic constraints into consideration as well as the combinatorial behavior. Despite the difficulty of the problem, we are hopeful that the method used to bound the phase 2 behavior, together with a better understanding of the algebraic constraints, will lead to a solution.

Acknowledgments. This work was supported in part by the AFOSR under grant FA9550-18-1-0120 and the Sloan Foundation under grant G-2018-10067.

References

[1] Lubomír Bakule. Decentralized control: An overview. *Annual Reviews in Control*, 32(1):87 – 98, 2008.

[2] Robert S. Boyer and J Strother Moore. *A Computational Logic Handbook*. Academic Press international series in formal methods. Academic Press, second edition, 1998.

[3] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[4] Darren D. Cofer, Andrew Gacek, Steven P. Miller, Michael W. Whalen, Brian LaValley, and Lui Sha. Compositional verification of architectural models. In Alwyn Goodloe and Suzette Person, editors, *NASA Formal Methods (NFM) 2012*, pages 126–140. Springer, 2012.

[5] Jennifer A. Davis, Laura R. Humphrey, and Derek B. Kingston. When human intuition fails: Using formal methods to find an error in the "proof" of a multi-agent protocol. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification (CAV) 2019*, pages 366–375. Springer, 2019.

[6] David Greve. A hierarchical proof of DPSS-A. in progress, 2021.

[7] Wolfram Research, Inc. Mathematica, Version 12.0. Champaign, IL, 2019.

[8] Derek B. Kingston, Randal W. Beard, and Ryan S. Holt. Decentralized perimeter surveillance using a team of UAVs. *IEEE Trans. Robotics*, 24(6):1394–1404, 2008.

[9] P. B. Sujit and Randy Beard. Multiple UAV exploration of an unknown region. *Ann. Math. Artif. Intell.*, 52(2-4):335–366, 2008.