Coalescing Swarms of Limited Capacity Agents: Meeting and Staying Together (without trust)

Luther A. Tychonievich and James P. Cohoon

***Abstract*—We consider the *coalescence* or *gathering* problem: a set of k mobile agents (robots) with distinct identifiers have to locate one another and gather at a common location. Agents may face several challenges: swarms of agent may be too large to simultaneously observe one another, agent sensors may be subject to noise, and some agents may be malicious.**

For small swarms, we design an algorithm coalescing all trustworthy agents despite sensor noise and an arbitrary number of malicious agents. This algorithm requires that all coalescing agents be able to simultaneously observe a single agent, limiting its practical scalability. For arbitrarily large swarms we design an algorithm achieving coalescence as long as the number of malicious agents near any given trustworthy agent is bounded. In both cases, our algorithms work by reduction to black-box solutions to well-studied problems, allowing them to be used in any environment and for any type of agents for which deterministic rendezvous, agent state estimation, and pair-wise cohesion algorithms are available.

Keywords: Swarms, coalescence, gathering, rendezvous, malicious agents

I. Introduction

We consider the *coalescence problem* defined as follows: given a set of k independent mobile agents (robots) with distinct identifiers, each agent shall move independently into a state where all agents are connected in a single network of mutual sensation. Coalescence is closely related to the *rendezvous problem* which has pairs of agent meet at a common location. In the situation where all agents are trustworthy, coalescence can be reduced to rendezvous: when agents meet, one agent follows the other. When some agents may be faulty or malicious, this simple reduction no longer achieves coalescence.

We solve the coalescence problem in the face of malicious agents by reducing it to several problems that have received significant attention in the literature. These include the rendezvous problem; the *cohesion constraint* problem, which requires that pairs of agents not move too far from one another; and the *estimation* problem, which has one agent determine conservative bounds on another agent's state by observing its actions.

It is common to distinguish between various classes of environments in which an agent might operate; these include structural classifications such as geometric spaces, grids, and graphs, as well as classifications of the timing information available to each agent. By utilizing existing rendezvous, cohesion, and estimation processes in a black-box manner, we are able to avoid all of these considerations and provide coalescence algorithms that will work anywhere the underlying processes are available.

Manuscript received March 18, 2012.

L. Tychonievich and J. Coohon are with the Department of Computer Science, University of Virginia, Charlottesville, VA, 22903 USA e-mail: {lat7h, jpc}@cs.virginia.edu.

The remainder of this paper is organized as follows. Section II contains basic definitions and background theorems. Section III describes coalescence for swarms small enough to all observe a single agent; we arrive at an algorithm for coalescing despite sensor noise and an arbitrary number of malicious agents. Section IV describes coalescence for larger swarms, presenting algorithms for limited-information global cohesion and for maintaining trustworthy communication in a swarm containing malicious agents as well as a large-swarm coalescence algorithm. Section V contains a survey of other approaches to coalescence and of existing methods for achieving cohesion and rendezvous. We conclude with a summary of results and some suggestions for future work.

II. Terminology

A. Agents and Coalescence

We assume that an *agent*

- is autonomous,
- can move under its own power,
- can sense its local environment (up to distance r under some distance function, which may depend on obstacles and other environmental parameters),
- can sense other agents within distance r, and
- can perform computations according to a supplied algorithm.

Definition 1 (Connected). Two agents are *connected* if the distance between them is no greater than r.

A set of agents is *connected* (or *k-connected*) if the graph with edges between connected pairs of agents is a connected (or k-connected) graph.

Recall that a k-connected graph is one in which every vertex cut contains at least k vertices.

Definition 2 (Neighbors). An third agent is *between* two agents if it is closer to each than the two are to each other.

Two agents are *neighbors* (or *k-neighbors*) if they are connected and there are no (or fewer than k) agents between them.

A graph of neighbor relationships corresponds to the relative neighborhood graph with agents for nodes and all edges longer than r removed.

We define a *swarm* of agents to be a set of agents that are currently. We say that agents *coalesce* when they become part of the same swarm.

A fundamental building-block to coalescence is cohesion, or keeping the swarm connected while moving.

Definition 3 (Cohesion). A *cohesion constraint* between two agents keeps them connected as they move.

A swarm exhibits *local cohesion* if no agent A will break connection with an agent B unless A can see a connected path to B that is no breaking.

A swarm exhibits *global cohesion* if no agent A will break connection with an agent B unless there exists a connected path to B that is no breaking.

A variety of cohesion algorithms have been published and are reviewed in Section V. Section IV contains techniques for achieving local and global cohesion from a cohesion constraint as well as extensions that ensure k-connectivity with k trustworthy agents in every cut.

Agents that can communicate with one another and have a common reference frame for identifying locations can coalesce by reaching consensus on a "home" location. For other agents, coalescing requires the agents be able to rendezvous.

Definition 4 (Rendezvous). An algorithm achieves *rendezvous* if, for any two agents executing the algorithm, there exists some time $t \geq 0$ at which the agents are connected. It achieves *repeating rendezvous* if for any time t_0 there exists some $t \geq t_0$ at which the agents are connected.

Some of our algorithms require that the *worst-case* runtime of the underlying rendezvous algorithm be finite. A variety of rendezvous algorithms, including repeating rendezvous algorithms with finite worst-case runtimes, have been published and are reviewed in Section V.

We assume agents can make *estimates* of various values by processing their observations. We define the following notation to refer to various levels of estimation.

Definition 5 ($A\lceil x\rceil_k$). Let predicate $A\lceil x\rceil$ be be true if and only if agent A's observations are sufficient to establish the truth of predicate x. Define $A\lceil x\rceil_k$ inductively as

$$A\lceil x\rceil_k \triangleq \begin{cases} \neg A\lceil x\rceil & k = 0 \\ A\lceil x\rceil & k = 1 \\ \forall X : A\lceil X\lceil x\rceil_{k-1}\rceil & k > 1 \end{cases}$$

where X is taken from the set of agents that have observed the same events A observed in establishing x.

If $A\lceil x\rceil_{i+1} \neq A\lceil x\rceil_i$ then two agents having observed the same events must be able to arrive at different conclusions, implying there is noise in the observations. We thus say observations of x are *noise-free* if $A\lceil x\rceil_{i+1} = A\lceil x\rceil_i$ for all $i > 0$; otherwise they are *noisy*. An agent A has *perfect knowledge* of x if $A\lceil x\rceil = x$.

B. Malice and Identity

Let the *behavior*[1] of an agent over a time window T be the set of all pairs $(t \mapsto s)$ where $t \in T$ and s is the state of the agent at time t. Let $\mathcal{G}_A^T$ be the set behaviors agent A might exhibit while following a specified algorithm in time window T; $\mathcal{G}_A^T$ is a singleton set for fully deterministic synchronous noise-free algorithms.

Definition 6 (Malice). Let A^T be the behavior of agent A over some time window T. A is *malicious in* T if $A^T \notin \mathcal{G}_A^T$. A is *malicious* if it is malicious for all T. A is *trustworthy* if it is not malicious.

We say B is *k-sure* that A is malicious if $B\lceil A^T \notin \mathcal{G}_A\rceil_k$ for the T during which B observed A's behavior. By definition 5 it follows that if any agent is k-sure A is malicious then any other agent also observing A^T will be at least $(k-1)$-sure no more than $(k+1)$-sure that A is malicious.

A finite observation can at most verify that an agent is not malicious in some time window T; this provides no guarantee it will not be malicious after T. Thus, there is no way to determine that an unknown agent or swarm is trustworthy without external stimuli or *a priori* knowledge about trustworth. This leads to the following theorem.

Theorem 1 (Identifiability). Coalesence in the presence of an unbounded but finite number of malicious agents requires that each agent can reliably identify each other agent and have sufficient storage to remember the identity of all malicious agents.

Proof: At least one agent must adjust its behavior when agents rendezvous or they cannot coalesce. By the pigeonhole principle, any change in behavior will delay the time to rendezvous with some other agents. If the same malicious agent can repeatedly rendezvous with an agent, it can repeatedly delay that agent rendezvousing with other, non-malicious agents.

Preventing repeatedly reacting to encounters with the same malicious agent requires that each agent be able to identify agents with which it rendezvous and to remember that it met at least the malicious subset of the agents it encounters. ■

We assume that every agent has a unique *identity* that is visible to all other agents; this means malicious agents are at most weakly Byzantine. An identity is *communicable* if it can be serialized into the same form by every observer, including the agent possessing the identity. Agent identities are *comparable* if all agents agree on a total ordering of identities. All of our algorithms assume comparable identities; those for large swarms also assume communicable identities.

III. Small Swarms

We define a *small swarm* to be a group of agents that all simultaneously observe and remain connected with a single *leader* agent. When all agents are capable of maintaining a single small swarm, cohesion can be achieve despite an arbitrary number of malicious agents. Our algorithm for doing so requires comparable (but not necessarily communicable) identities; repeating rendezvous; and either noise-free observations of other agents or the ability to broadcast messages to other agents.

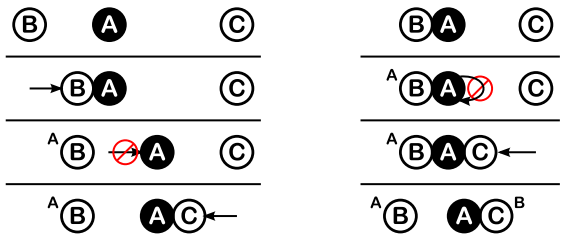
A. Full Trust

The simplest coalescence algorithm works for small swarms with no noise or malice. When n agents rendezvous, the one with the maximum identity becomes the leader. The leader continues its rendezvous search uninterrupted, while the other agents follow it. This general approach has been discussed by several authors, often in the context of motivating the rendezvous problem; we give it no further discussion here.

B. Noise-Free Malice

When observations of agents are noise-free, each agent in a swarm will agree upon the malice of the leader provided they have all observed a sufficient quantity of the leader's actions. The simplest extension of the full-trust rendezvous in this case is to have each agent maintain a blacklist of agents it

[1]We use "behavior" to formalize the notion of "what an agent does;" this is not the same as a "behavior" in a behavior-based algorithm.



(a) Leader blacklisting. (b) Follower blacklisting.

Fig. 1: Failure of two example approaches to individual blacklisting where $A > B > C$ and A is malicious. If agents don't join swarms with blacklisted leaders (a) then A plays nice with C to ensure B and C don't coalesce. If agents join swarms with any nonblacklisted members (b) then A pretends to lead B until C comes along; then, when A and B's rendezvous searches differ, C incorrectly thinks that B maliciously left the swarm.

has observed acting maliciously. Individual blacklists, such as those illustrated in Fig. 1, fail because a malicious agent can manipulate different agents' blacklists separately. Individual blacklists that track entire swarms instead of individual agents do work, but require exponential space.

We propose instead a shared-blacklist approach that makes three guarantees: trustworthy agents never leave a swarm, sufficiently malicious agents never lead a swarm for too long, and each swarm increases in size between times a malicious agent leads it.

Algorithm 1 (Noise-free Small Swarms)**.** All agents are initially leaders of their own swarms of one agent. Swarm leaders execute a rendezvous search algorithm. We handle the following events:

Agent leaves swarm: Other agents blacklists that agent.

Two swarms meet: If all of the agents in the other swarm are blacklisted, ignore them. Otherwise clear all blacklists and leader estimation information and follow the best leader of the new, larger swarm.

Leader acts maliciously: Blacklist the old leader; the new leader is the non-blacklisted agent with the greatest identity.

Theorem 2. Algorithm 1 guarantees coalescence of small swarms despite any number of malicious agents provide observations are noise-free, identities are comparable, rendezvous is repeating, and any agent observing a malicious behavior that would prevent rendezvous eventually becomes 1-sure it has done so.

Proof: By clearing all estimation information each time a new agent arrives, the swarm is guaranteed (by the noise-free assumption) to agree on when a leader should be blacklisted. By clearing the blacklists each time a new agent arrives, the swarm is guaranteed all agents' blacklists are identical. By blacklisting agents that leave the swarm, malicious agents cannot arbitrarily reset the estimation information and blacklists unless they return with or after a new agent.

Thus, each malicious leader can delay coalescence by at most one detectably-malicious act between each growth of the swarm. If there are m malicious and t trustworthy agents, a given swarm can be delayed by at most mt detectably-malicious acts before full coalescence. Since the underlying rendezvous algorithm is repeating, finite delays cannot prevent coalescence. ■

Algorithm 1 discards each swarm's knowledge of malicious agents when two swarms meet. This potential inefficiency can be avoided if each agent keeps an individual blacklist as well as the swarm blacklist. When swarms meet or agents are blacklisted, the remaining agents use message passing to establish the intersection of the individual blacklists and add that intersection to the shared blacklist. Whether this optimization improves the expected- or worst-case runtime depends on the details of the underlying rendezvous algorithm; a full investigation of runtime performance is left to future work.

C. Noise and Malice

Algorithm 1 does not work when agents may disagree on when to blacklist a leader. In such a noisy environment we consider agents that can unambiguously share some information. In particular, we consider agents that can broadcast the k by which they are k-sure that the leader of the swarm is malicious.

If every agent is at least 1-sure that the leader is malicious then the leader may be blacklisted by the entire swarm and a new leader selected.

An agent that is k-sure that the leader is malicious should not desert an agent that is $(k-1)$-sure, who in turn should not desert an agent that is $(k-2)$-sure, etc. We define the *surety-chain* of an agent broadcasting k recursively as the agents broadcasting k and the surety-chains of any agents broadcasting $k+1$ or $k-1$. Agents cannot convince a surety chain to separate.

Algorithm 2 (Noisy Small Swarms)**.** All agents are initially leaders of their own swarms of one agent. Swarm leaders execute a rendezvous search algorithm. Agent A should handle the following events:

B leaves swarm: A adds B to its private blacklist.

Two swarms meet: If all of the agents in the other swarm are in either A's private or shared blacklist, A ignores them. Otherwise, A clears both of its blacklists and follows the best leader of the new, larger swarm.

A is k-sure that the leader is malicious: A broadcasts k.

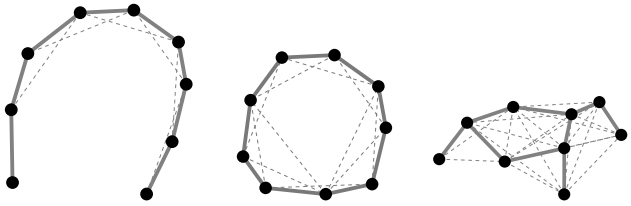
B broadcasts: A updates its surety-chain. Agents outside the surety-chain are added to A's shared blacklist.

A's surety-chain contains no 0-sure agents: A adds the swarm leader to its shared blacklist.

The leader is blacklisted: Begin following the next-best agent that is not on the shared blacklist.

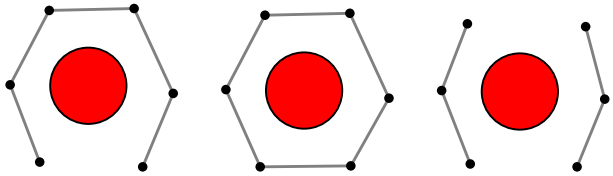
Theorem 3. Algorithm 2 guarantees coalescence of small swarms despite any number of malicous agents provided identities are comparable, rendezvous is repeating, and any agent observing a behavior that would prevent rendezvous eventually becomes k-sure that it has done so for arbitrary k.

Proof: All agents within a surety-chain agree on which other agents are within it and that it contains all of the trustworthy agents. Because surety-chains stay with a leader until all agree the leader is malicious, no surety chain will ever leave a trustworthy leader. Because swarms contain a finite number of agents, there is some k at which an agent



(a) Open (b) Ring (c) General

Fig. 2: Several swarms; neighbors are displayed in bold solid lines, connections in dashed lines. While open configurations (a) can change into rings (b) freely, and both may change to and from general swarms (c), rings cannot readily change into open configurations.



(a) one swarm before trap (b) trapped (c) two swarms before trap

Fig. 3: A swarm becoming trapped around an obstacle. Local cohesion prevents the the swarm from leaving the obstacle.

that is k-sure the leader is malicious must be part of a chain containing no 0-sure agents.

Agents might not agree on the definition of "leave the swarm;" hence, agents that leave cannot be added to the shared blacklist and excluded from leadership. However, any agent noticing a malicious agent leaving will ignore its return, and any agent not noticing its departure won't be aware of its return. Hence, leaving-and-returning agents do not reset any agent's information.

The rest of the proof mirrors the proof of Theorem 2. ■

One antecedent to Theorem 3 is that any agent observing a behavior that would prevent rendezvous eventually becomes k-sure it has done so for arbitrary k. Most rendezvous algorithms can be made to handle finite errors by decreasing the perception radius used in the algorithm below that available to the agent. For any asymptotically-accurate estimation filter, this finite buffer will provide the requisite k-surety.

IV. Large Swarms

We define a *large swarm* to be a set of connected agents (see Definition 1) that cannot all simultaneously observe a single leader. For large swarms, coalescence requires both rendezvous and cohesion. We first discuss coalescence of trustworthy agents; we then discuss how to extend it for swarms with some malicious agents. In both cases we require that agents are able to broadcast messages to their neighbors and that each agent has a communicable, comparable identity.

A. Trustworthy Agents

Large swarm coalescence differs from small swarm coalescence principally in handling cohesion. As in small-swarms, it is sufficient to have the agent with the maximum identity (as determined by message passing) continue with rendezvous while the others follow along. What differs is how the other follow.

The core of swarm cohesion is the cohesion constraint, which ensures that a pair of agents not separate from one another while moving. By applying the cohesion constraint to every pair of neighbors in a swarm we achieve local cohesion; locally cohesive swarms can reconfigure themselves, but cannot break redundant connections in non-local loops (see Fig. 2). However, these loops can be formed by the same process that allows swarms to merge. This irreversible creation of loops leads to a trapping behavior in the presence of obstacles, as illustrated in Fig. 3.

The simplest technique for preventing trapping behaviors is to forbid irreversible events within a swarm: only already-connected agents may move into a neighboring configuration. This technique may be implemented by having agents that become neighbors without first being connected share their swarm leaders' identities and move apart if they are part of the same swarm. However, such an approach does not prevent trapping when two swarms meet in multiple locations (see Fig.3(c)). Overcoming simultaneous joins through distributed mutual exclusion is possible, but is beyond the scope of this work.

An alternative technique to preventing trapping allows agents to break neighborhood connectivity whenever doing so does not break swarm connectivity; these breaks result in global rather than local cohesion.

Algorithm 3 (Global Cohesion)**.** Each agent may move freely provided it does not break connection with any neighbor. If an agent desires to break with a neighbor, it broadcasts a message seeking an alternate connection to that neighbor within the swarm. This takes the form (from, source, destination, timestamp). When A wishes to separate from neighbor B it broadcasts $(i_A, i_A, i_B, t = now + dt)$ and waits to receive (i_B, i_A, i_B, t) before t arrives.

When agent A receives message (i_f, i_s, i_d, t) it sends message (i_A, i_s, i_d, t) unless any of the following are true:

1) A and f are not connected;
2) $t < now$ (messages expire);
3) $i_f = i_s$ and $i_d = i_A$ (edges can't approve themselves);
4) A already sent (i_A, i_s, i_d, t) (messages don't cycle);
5) A has received $(i_f, i_f, i_A, t_1 \geq now)$ and $i_f > i_s$; or
6) A has sent $(i_A, i_A, i_f, t_1 \geq now)$ and $i_A > i_s$

(Items 5 and 6 together prevent higher-priority edges requesting a break from forwarding lower-priority messages.)

When an agent sends a message with timestamp t, it is not permitted to initiate a new message until after t expires.

Algorithm 3 makes explicit reference to a timestamp, implying a globally-synchronized clock. Asynchronous clocks may be used instead if communication across the entire network of agents has an known upper time bound T. In that case, timestamps may be omitted and agent should not initiate any messages until T after sending a message.

Theorem 4. Algorithm 3 allows only breaks that do not split the swarm into disconnected pieces. If messages do not time out, it forbids only one break for each set of requests that would split the swarm in two.

Proof: Let $N = (V, E)$ be the graph with edges between connected agents. It is never the case that, for $e_i, e_j \in E$,

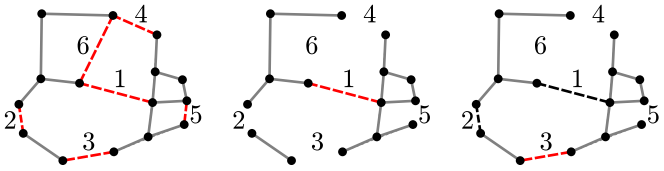


Fig. 4: Illustration of Theorem 4. On the left is N with break requests in red dashed lines. In the center, e_1 is a bridge on N_1 so it cannot break. On the right e_3 is not a bridge in N_3 so it can break. The forbidden breaks are precisely those required to connect the graph.

both break request i crosses edge e_j and break request j crosses edge e_i unless either i or j has expired.

Consider break request e_i. Let $N_i = (V, E \backslash \{e_j : j > i\})$ be the graph containing only edges that will forward break request e_i. If e_i is a bridge on N_i, the break request will be denied and e_i will be a bridge after all break requests are resolved. Otherwise, the break request will be approved and not break the graph. ■

Undiscussed by Algorithm 3 is how to determine when to send a message and with what initial timestamp. Timestamp doubling can be used to ensure that eventually messages last long enough to reach their destinations; alternatively, old timestamps can be forwarded so that edges realize their message expired rather than being blocked. The frequency with which messages are initiated is limited only by the communication medium utilized; heuristics for selecting message initiations and durations are left to future work.

B. Limited Malice

When some agents cannot observe other agents directly, they are dependent on intermediaries to convey information about the swarm status. We require some portion of each swarm to be trustworthy so that this conveyance can be trusted.

Communication within a swarm is interrupted if a vertex cut of the swarm is malicious. Swarms with malicious vertex cuts of can be constructed for k-connected graphs with arbitrary k, and may be created with only k malicious agents and with an arbitrary minimum path length between malicious agents; one such construction is illustrated in Fig. 5 We thus require that in any vertex cut of the graph there be at least $m+1$ trustworthy agents, and that each trustworthy agent is connected to no more than m malicious agents. These requirements mean that if messages forwarded by any agent that hears it from at least $m+1$ agents, then all such messages can be trusted and will reach all trustworthy agents in the graph.

Theorem 5. If there are no more than m malicious agents connected to each trustworthy agent then applying a cohesion constraint between each pair of $(2m+1)$-neighbors ensures there are $m+1$ trustworthy agents per vertex cut in the connectivity graph.

Proof: Consider any vertex cut in the graph. Consider two agents, one on each side of the cut, who are connected to the same agent in the cut. All of the agents between those two agents are in the cut. By the local cohesion constraint the agents would never have agreed to fewer than $2m+1$ agents between them. Since a malicious agent cannot impact the neighbor relationship of agents with which it is not connected, one malicious agent is required per missing intermediary. Thus, at most m of those $2m+1$ agents could be missing or malicious, leaving at least $m+1$ that must be trustworthy. ■

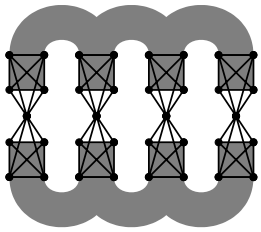


Fig. 5: Example construction of a four-vertex cut using four widely separated agents in a four-connected graph. Wide gray curves represent arbitrarily long four-connected paths of agents.

The following modification of Algorithm 3 also maintains $m+1$ trustworthy agents per cut provided no more than m malicious agents are connected to any given agent.

Algorithm 4 (Non-local Trustworthy Cuts)**.** Each agent may move freely provided it does not break connection with any $(2m+1)$-neighbor. Agents seeking to break with a neighbor broadcast a message as in Algorithm 3.

When agent A receives message (i_f, i_s, i_d, t) it ignores messages per items 1–6 in Algorithm 3; non-ignored messages are added to an internal set of non-expired messages. If the set now contains either $(2m+1)$ entries (i_X, i_s, i_d, t) with distinct i_X or a single entry (i_s, i_s, i_d, t) then A broadcasts (i_A, i_s, i_d, t).

Agents A and B may break when B sends message (i_B, i_A, i_B, t) provided A had earlier sent message (i_A, i_A, i_B, t).

When an agent sends a message with timestamp t, it is not permitted to initiate a new message until after t expires.

We assume that agents are not able to lie about their own identity when sending messages.

Theorem 6. Breaks permitted by Algorithm 4 will never introduce cuts with fewer than $m+1$ trustworthy agents.

Proof: Consider a cut $\mathcal{C}$ introduce when Algorithm 4 approved a break between agents A and B; assume the break request was initiated by A.

Suppose either A or B is malicious. Since both $\mathcal{C} \cup \{A\}$ and $\mathcal{C} \cup \{B\}$ were cuts before the break, each had $m+1$ trustworthy agents; since either A or B is malicious, $\mathcal{C}$ must contain $m+1$ trustworthy agents.

Suppose both A and B are trustworthy. An agent will only have approved the break if it received the request from $2m+1$ connected agents. Since each agent is connected to at most m malicious agents, each agent that approved the break must have received the request from $m+1$ trustworthy agents. Since neither A nor B would have transfered the message across $\mathcal{C}$, for B to have approved the message some agent X on Bs side of $\mathcal{C}$ must have received the message from at least $m+1$ trustworthy agents in $\mathcal{C}$. Thus, $\mathcal{C}$ must contain $m+1$ trustworthy agents. ■

Given trustworthy communication and a trustworthy quorum observing a leader agent, coalescence for swarms with limited trust is relatively straightforward. The agents

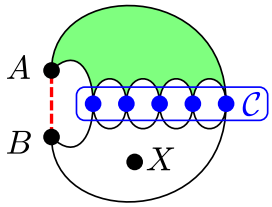


Fig. 6: Illustration of proof to Theorem 6. A break request is sent by A and propagates through A's side of vertex cut $\mathcal{C}$. If the request is to make it to B, it must reach some arbitrary first node X on B's side of $\mathcal{C}$, requiring $m+1$ trustworthy requests from agents in $\mathcal{C}$.

observing the leader broadcast when they notice the leader behaving maliciously; once $m+1$ agents make this broadcast it will propagate to the entire swarm and the entire swarm can blacklist the previous leader. Election of a new leader and recognition of new members of the swarm can be handled via similar trustworthy broadcasts. The blacklist reset, etc., can be handled per Algorithm 1.

V. Related Work

Coalescing trustworthy agents in a graph with malicious agents has been investigate in [1]. They provide algorithms for noiseless agents to coalesce in the presence of a minority of malicious agents. Their algorithms also allow the agents to detect that they have completed coalescing by having *a priori* information about the number of trustworthy and malicious agents. Our algorithms extend theirs in the number of malicious agents we can handle, the types of (possibly noisy) environments the agents can inhabit, and the practical size of the swarms. These advances are gained at the expense of handling only weakly Byzantine agents and of not detecting when coalescence has been achieved.

The problem of coalescing, also called gathering or aggregating, is often presented as a reduction to rendezvous; we outline this common approach in Section III-A.

The rendezvous problem was introduced in [2] and has been investigated in many settings. In interest of space we mention only deterministic algorithms for agents without *a priori* knowledge of one another's location or the quantity of agents involved. See [3] for a survey of randomized rendezvous algorithms; see also [4]–[7] for asymptotic analysis of coalescence based on randomized rendezvous algorithms. On a line or ring, tight bounds are known and both deterministic and randomized coalescence algorithms are known to achieve them (e.g., [7]–[10]). A tight $O(\log n)$ space-bound was demonstrated for deterministic rendezvous in a tree in [11]. Rendezvous in a graphs or networks has been extensively studied and deterministic algorithms developed for partial and full asynchrony, for indistinguishable agents in visibly-distinct starting locations, and for agents with distinct identities (e.g., [1], [10], [12]–[15]). In a geometric setting, deterministic approaches have been posed using several approaches including landmarks [16], stigmergy [17], reduction to a graph [14], and repeated deterministic exploration [13]. Variable-speed clocks, position noise, and position drift have also been investigated [10], [13], [14].

Cohesion of groups of moving agents has been investigated in many contexts. In addition to many flocking and formation algorithms, local cohesion has been provided using switching laws that alternate between mission objectives and connection reinforcement (e.g., [18]–[20]); using potential fields to combine mission objectives and cohesion (e.g., [21], [22]); and using algebraic approximations of arbitrary geometric constraints (e.g., [23]). Global cohesion has been added in [20], [21] by communicating the entire network connectivity to each agent and then selecting a set of cuts to be made; [21] also discusses market-based consensus algorithms for selecting cuts. Our approach to global cohesion is based on comparable identities instead of market-based decisions, and only requires each agent to recall a set of messages received in a finite time window: no agent needs to accumulate knowledge of the connectivity graph. We also handle cohesion in the presence of malicious agents in both a local and global sense.

Estimating bounds on agent state based on observed behavior has been widely studied for many classes of agents; entire fields of research are devoted to signal processing and estimation and are not reviewed here.

VI. Conclusion

We investigated the coalescence problem and show that it can be solved in finite time no matter how many malicious agents are present as long as all agents are able to simultaneously observe a single leader. For larger swarms where decentralized activities are essential, we demonstrated that global cohesion can be achieved using only a simple message-passing protocol without any agent modeling the structure of the connectivity graph. We also proved that we can ensure that trustworthy communication is available within a swarm only if the number of trustworthy agents in an arbitrary vertex cut of the swarm is subject to a lower bound. We showed how, if the number of malicious agents in the neighborhood of each agent is bounded, local cohesion constraints and simple message passing protocols can to keep trustworthy agents in every vertex cut.

Several theoretical questions remain open. While we showed that our message passing protocols for global cohesion is tight in the sense that it will approve any break request that does not violate connectivity, we have not demonstrated similar tightness for the trustworthy vertex cut version of the protocol. Likewise, while we demonstrated that trustworthy vertex cuts are required for large-group message passing, we do not know if message passing itself is necessary for large-group cohesion.

Other work has investigated a stronger notion of malice in noise-free graphs with known numbers of agents [1]. It would be interesting to see if that model of strongly-Byzantine agents can be adapted to noisy geometric environments and/or swarms too large to be simultaneously observed.

While the theory of rendezvous, cohesion, and coalescence improve, there remains significant work to be done before these algorithms become practical for real robots. Among the issues to be investigated before such tests can be made are details of message passing frequency, models of agent state estimation, and budgeting of physical resources.

We have made significant steps in improving the theoretic bounds on coalescence in the presence of malicious agents, observation noise, and swarm size constraints. We look forward to further advances in this unfolding field of inquiry.

REFERENCES

[1] Y. Dieudonné, A. Pelc, and D. Peleg, "Gathering despite mischief," in *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, 2012, pp. 527–534.

[2] T. Schelling, *The strategy of conflict*. Oxford, England, UK: Oxford University Press, 1960.

[3] S. Alpern and S. Gal, *The theory of search games and rendezvous*, ser. International Series in Operations Research and Management Science, 2002.

[4] P. Dykiel, "Asymptotic properties of coalescing random walks," Uppasala University, Tech. Rep. 2005:15, December 2005.

[5] S. Poduri and G. S. Sukhatme, "Achieving connectivity through coalescence in mobile robot networks," in *RoboComm '07: Proceedings of the 1st international conference on Robot communication and coordination*. Piscataway, NJ, USA: IEEE Press, 2007, pp. 1–6.

[6] ——, "Latency analysis of coalescence in robot groups," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3295–3300.

[7] E. J. Anderson and S. Essegaier, "Rendezvous search on the line with indistinguishable players," *SIAM Journal on Control and Optimization*, vol. 33, pp. 1637–1642, 1995.

[8] S. Alpern, "The rendezvous search problem," *SIAM Journal of Control and Optimization*, vol. 33, no. 3, pp. 673–683, 1995.

[9] S. Alpern and S. Gal, "Rendezvous search on the line with distinguishable players," *SIAM Journal on Control and Optimization*, vol. 33, pp. 1270–1276, 1995.

[10] G. D. Marco, L. Gargano, E. Kranakis, D. Krizanc, A. Pelc, and U. Vaccaro, "Asynchronous deterministic rendezvous in graphs," *Theoretical Computer Science*, no. 335, pp. 315–326, 2006.

[11] P. Fraigniaud and A. Pelc, "Deterministic rendezvous in trees with little memory," in *Proceedings of the 22nd International Symposium on Distributed Computing (DISC 2008)*, ser. Springer Lecture Notes in Computer Science, vol. 5218, 2008, pp. 242–256.

[12] D. Kowalski and A. Malinowski, "How to meet in anonymous network," in *13th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2006)*, ser. Springer Lecture Notes in Computer Science, vol. 4056, pp. 44–58.

[13] L. A. Tychonieivch and J. P. Cohoon, "Guaranteeing rendezvous of oblivious limited-capability mobile agents," University of Virginia, Tech. Rep. CS-2012-04, 2012.

[14] J. Czyzowicz, A. Labourel, and A. Pelc, "How to meet asynchronously (almost) everywhere," in *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, 2010, pp. 22–30.

[15] J. Czyzowicz, A. Kosowski, and A. Pelc, "How to meet when you forget: Log-space rendezvous in arbitrary graphs," in *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing (PODC 2010)*, 2010, pp. 450–459.

[16] N. Roy and G. Dudek, "Collaborative exploration and rendezvous: Algorithms, performance bounds and observations," *Autonomous Robots*, vol. 11, no. 2, pp. 117–136, September 2001.

[17] A. Shiloni, N. Agmon, and G. A. Kaminka, "Of robot ants and elephants," in *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 81–88.

[18] G. A. S. Pereira, A. K. Das, and V. Kumar, "Decentralized motion planning for multiple robots subject to sensing and communication constraints," in *Proceedings of the 2003 International Workshop on Multi-Robot Systems*, 2003, pp. 267–278.

[19] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor, "Maintaining network connectivity and performance in robot teams," *Journal of Field Robotics*.

[20] J. Vazquez and C. Malcom, "Distributed multirobot exploration maintaining a mobile network," in *Proceedings of the 2nd International IEEE Conference on Intelligent Systems*, vol. 3, 2004, pp. 113–118.

[21] M. M. Zavlanos and G. J. Pappas, "Distributed connectivity control of mobile networks."

[22] X. Li, D. Su, J. Yang, and S. Liu, "Connectivity constrained multirobot navigation with considering physical size of robots," in *Proceedings of the International Conference on Automation and Logistics*, Chongquing, China, August 2011, pp. 24–29.

[23] L. A. Tychonieivch and J. P. Cohoon, "Cohesion: Keeping independently-moving agents close together," University of Virginia, Tech. Rep. CS-2012-04, 2012.