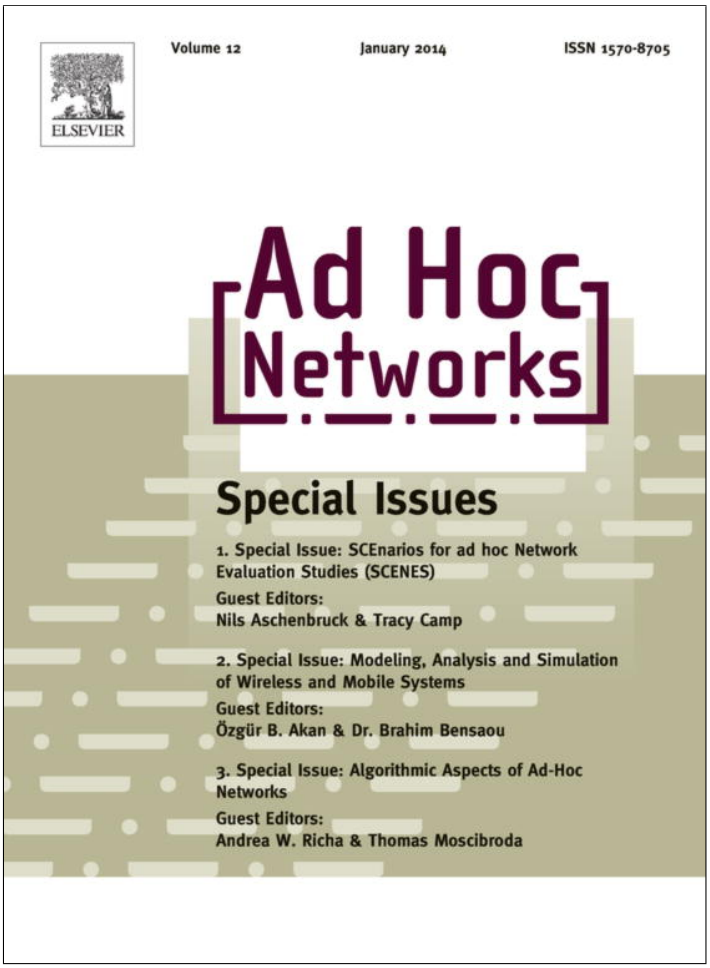


Ad Hoc Networks 12 (2014) 250–258

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

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

Direction election in flocking swarms ☆

Ohad Ben-Shahar [a], Shlomi Dolev [a], Andrey Dolgin [b], Michael Segal [b,*]

[a] Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
[b] Department of Communication Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

ARTICLE INFO

Article history:
Received 28 February 2011
Received in revised form 8 April 2012
Accepted 3 May 2012
Available online 11 May 2012

Keywords:
Direction election
Flocking
Swarms
Mobile robots

ABSTRACT

Swarm formation and swarm flocking may conflict each other. Without explicit communication, such conflicts may lead to undesired topological changes since there is no global signal that facilitates coordinated and safe switching from one behavior to the other. Moreover, without coordination signals multiple swarm members might simultaneously assume leadership, and their conflicting leading directions are likely to prevent successful flocking. To the best of our knowledge, we present the first set of swarm flocking algorithms that maintain connectivity while electing direction for flocking, under conditions of no communication. The algorithms allow spontaneous direction requests and support direction changes.

1. Introduction

Coordinating motion, cooperative formation, and flocking control of multiple autonomous entities is of great theoretical and practical interest. One (now classical) approach is Reynolds's Boids [13], where each entity updates its movement based on the distances and velocities of neighboring entities in order to ensure the *alignment* and *cohesion* of the swarm while avoiding unsafe distances between the entities. The three corresponding behavioral rules that each entity follows indeed address many practical situations, however there are still several pathological cases where partition of the swarm is possible. For example, if it happens that all entities move exactly towards (or away from) their center of mass, no convergence to stable flocking can occur and it becomes impossible to break the symmetry without using randomization. In this paper, an intrinsic bounded random variable is indeed incorporated, though its primary use goes beyond symmetry breaking, handling leadership election when no explicit information exchange is allowed between the swarm's entities.

Clearly, convergence of a swarm motion while avoiding topological changes like partitioning is possible using explicit (e.g., wireless) communication, and indeed early work has explored such solutions, in particular when the communication is employed in a predefined time interval [5,14,18]. Later work relaxed the defined time interval to variable-length intervals [4,8,10] or employed probabilistic networks [17]. However, unlike previous work, we seek to solve these problems in "silent networks", where there is no explicit communication between the swarm members beyond passive observation (of position, motion, etc.). Furthermore, unlike much of the previous work, we are also interested in maintaining robust convergence results despite uncertainties in both the measurement and execution of motion. All these conditions mimic constraints of artificial swarms.

Much work on the stability and convergence of flocking in silent swarms has been based on the use of *potential functions*. Doing so, several algorithms and convergence results have been obtained for different flocking mechanisms and various connectivity assumptions (e.g., [6,11,12,19]). While most effort was put to investigate leaderless

☆ Research supported in part by the ICT Programme of the European Union under Contract No. FP7-215270 (FRONTS), US Air Force European Office of Aerospace Research and Development, Grant No. FA8655-09-1-3016, Microsoft, Deutsche Telekom, European project FLAVIA, Israeli Ministry of Industry, Trade and Labor (consortium CORNET), and Rita Altura Trust Chair in Computer Sciences.

* Corresponding author.
E-mail addresses: ben-shahar@cs.bgu.ac.il (O. Ben-Shahar), dolev@cs.bgu.ac.il (S. Dolev), dolgin@cse.bgu.ac.il (A. Dolgin), segal@cse.bgu.ac.il (M. Segal).

1570-8705/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2012.05.001

swarms, some research has extended this scope, presenting potential field algorithms for safe, connectivity-preserving flocking in the presence of a single leader. However, thus far no potential function that guarantees the desired flocking behavior (and in particular, collision avoidance and connectivity preservation) when multiple leaders may co-exist, is known. Still, the case of multiple leaders has been considered in several studies, typically employing local rules and information passing between neighbors [15,7]. Only few recent papers study the problem in the scope of "silent networks". For example, Jiang-Ping and Hai-Wen [7] show that all agents will flock in the polytope region formed by the leaders. Su et al. [16] have studied the case of a fixed set of leaders and showed how the swarm will steadily converge to the dynamical center of mass of the leaders. While suitable under specific assumptions (e.g., symmetry does not exist) for solving different agreement problems like swarming, schooling, flocking, or rendezvous, existing methods do not support direction election. Furthermore, the majority of previous works bear on a strict assumption that connectivity is preserved and collisions between entities are avoided all the time, while no technique to satisfy these properties is described.

An important issue related to flocking swarms is their temporal convergence rate to a stable configuration. Results on this problem have been obtained by Chazelle in [1,2], who showed an exponential time bound for the convergence of the motion of a *single entity* to a fixed motion vector. Furthermore, the time bound for the convergence of the entire swarm was shown to be an iterated exponential of height logarithmic in the number of agents. Since the flocking rules in our model are superset of the rules in the model of Chazelle [1,2], we suspect that the convergence rate (denoted later by T) proved in [1,2] is an upper bound on convergence rate in our model.

In this paper we present (what we believe are) the first practical and provable flocking schemes with silent direction election from several candidate leaders. We develop simple and efficient algorithms for silent direction election, taking into account (bounded) environmental and parametric uncertainties, while providing a mechanism for connectivity preservation and collision avoidance. These latter capacities are obtained by introducing the notion of a *spring*, which resembles a potential function with restrictions and provides flexibility in the presence of uncertainties. The same mechanism also withstands temporary coexistence of multiple leaders. We do note that although direction election techniques do exist in the context of mobile robotics [3,9], these methods assume explicit communication between entities, which is outside the scope of our study.

The rest of this paper is organized as follows. Section 2 briefly describes basic settings and definitions. Section 3 describes the intuition behind our new spring network approach and proves its effectiveness for direction election in autonomous mobile swarms under uncertainties, assuming initial connectivity of swarm entities. Section 4 describes a method for monitoring of swarms motion by every entity and an alternative direction election approach based on it. Conclusions appear in Section 5.

2. Swarm settings

We begin our theoretical discussion with several definitions and notions, the first of which relates to the different swarms configurations that may be considered in the context of flocking. In particular, we will first consider *synchronous swarms*, where all entities perform their measurements at the same points in time based on one global clock. We will then move to discuss *partially asynchronous swarms*, which relax the synchronous swarms assumption to allow each entity to perform its measurements and motion updates after some arbitrary time phase relative to global time pulses (as the local clocks measure time periods in the same rate but are not synchronized to hold the same global time). We note that the case of asynchronous network, where each entity has its own independent clock and cycle duration, is not considered here.

Definition 2.1 (*Cycle*). Cycle duration dt is a unit of time used by all entities for their measurement and position adjustments.

A *period* is a sequence of numbered cycles starting from 1, and its *size* is measured in time slots. As we emphasized in the Introduction, the focus of this paper is *silent swarms*, i.e. network of entities *without* explicit communication of information transfer. In particular, the only measurement allowed for different entities is the position of neighboring entities. Let r_i be the position vector of entity i. We therefore define:

Definition 2.2 (*Distance*). Let r_i and r_j be the position vectors of entities i and j, respectively. The distance r_{ij} between the two entities is therefore defined as the Euclidean norm of the corresponding difference vector r_{ij}, i.e. $||r_i - r_j||$.

Due to the measurement errors, the measurement made by entity i for the distance to its neighbor j may be different from the real value r_{ij}.

Definition 2.3 (*Measured distance*). Let $r_i^{m_i}$ and $r_j^{m_i}$ be the position vectors of entities i and j, respectively, measured by entity i. The distance $r_{ij}^{m_i}$ between the two entities is therefore defined as the Euclidean norm of the corresponding difference vector $r_{ij}^{m_i}$, i.e. $||r_i^{m_i} - r_j^{m_i}||$.

To better support practical applications, we do allow some uncertainty in distance measurements and the execution of motion commands, and, in particular, we assume that distance measurement errors (compared to the true distances) are bounded by a known constant, which includes all possible noise sources, such as a randomization used in all the algorithms for symmetry breaking, etc. Formally:

Definition 2.4 (*Error*). Error e_{ij} is the difference between the measurement made by entity i for the distance to its neighbor j and their actual distance, i.e. $e_{ij} = r_{ij}^{m_i} - r_{ij} + RV$, where RV stands for a random variable value, bounded by a predetermined constant, which is used to break possible pathological symmetry position of entities. The error is bounded by some known constant e, i.e. $|e_{ij}| \leqslant e \ \forall i,j$.

Given a swarms of entities, one may associate a graph topology based on distances. In particular, we define the maximum distance, under which entities can be considered connected (or neighbors in the swarm graph), and the minimum distance, under which the safety of entities is compromised.

Definition 2.5 (*Connectivity limit*). Connectivity limit R between two entities is the maximal distance, where those entities are considered connected to each other (e.g., by visibility).

Definition 2.6 (*Proximity limit*). Proximity limit r between two entities is the minimal safety distance that prevents the entities from collision.

Of course, the connectivity limit entails a neighborhood relationship between entities, i.e.

Definition 2.7 (*Neighbors*). The set $N(i)$ of all neighbors of entity i is the set of all entities whose distance from entity i as measured by entity i does not exceed the connectivity limit value R, i.e. $r_{ij}^{m_i} \leqslant R$.

In order to obey behavioral rules such as Reynolds' cohesion and separation, we propose to describe the interaction between swarm neighbors via the following intuitive mechanism: two neighboring entities i and j, such that $i \in N(j)$ and $j \in N(i)$, are connected by (virtual) spring, which applies force on its end points (i.e. the entities) based on its length. Formally,

Definition 2.8 (*Spring*). Spring is a virtual structure connecting any two neighboring entities. Spring size, measured by entity i, equals the distance $r_{ij}^{m_i}$ between the two entities i and j, where $r \leqslant r_{ij}^{m_i} \leqslant R$. This important property of the spring states that it can neither stretch above R, nor shorten under r. Each pair of entities i,j that become neighbors (i.e. $i \in N(j)$ and $j \in N(i)$) obtains a spring. The force that the spring applies on its ends is proportional to its size, i.e. the force on each end is equal to $(r_{ij}^{m_i} - (R+r)/2)/2$ and $(r_{ij}^{m_j} - (R+r)/2)/2$, so the spring attains its equilibrium state in the middle between R and r.

Note that the definition above does not necessarily mimic a physical spring. However, regardless of the implementation, the force applied on each entity by its springs affects it velocity, which one can define in the standard way, i.e.

Definition 2.9 (*Velocity*). The velocity $v_i^{m_i}$ of entity i is the time derivative of its position vector $v_i^{m_i} = \dot{r}_i^{m_i}$ and it is assumed to be constant within a cycle.

Finally, we note that one of the main purposes of the algorithm described in the bulk of this paper is to preserve initial connectivity of the swarm network. We therefore define.

Definition 2.10 (*Connectivity*). Let $G(V,E)$ be the graph whose nodes are the swarm entities and the bidirectional edges are the swarm bidirectional springs (both entities on the end of the spring are aware of its existence). The swarm is said to be connected if its corresponding graph is connected.

Definition 2.11 (*Non-critical starting point*). Non-critical starting point is a sufficient condition to start using direction election algorithms. It states that $\forall i,j | j \in N(i), r + 2e \leqslant r_{ij}^{m_i} \leqslant R - 2e$ that is all the entities are on the safe distance from each other, and also the swarm is connected.

3. Direction election in silent swarms

3.1. Synchronous systems

Leadership of an entity in silent swarms can only be achieved once we violate the basic Reynold's rules. However, such arbitrary motions should be done with care in order to avoid changes to the connectivity (or more strictly, the graph topology) of the swarm. We begin our investigation of how this can be done in synchronous swarms, where all entities share the same global clock and all perform their distance measurements at the same time (but otherwise exchange no other information by any other means). The main advantage of a synchronized system is that by carefully assigning time slots to the entities we avoid a situation when any two would try to lead simultaneously. Notice that in our solutions once entity i sees entity j and entity j sees entity i, both entities will never lose a bidirectional connection between them.

3.1.1. Leadership in flocking swarms

By Definitions (2.9) and (2.1), the velocity between subsequent measurements is assumed to be constant. We observe that when an entity j decides to lead, it may move a distance of at most $(R - r_{ij}^{m_i})/2 - 2e$ in the direction of stretching each one of the springs connected to it, and $(r_{ij}^{m_i} - r)/2 - 2e$ in the direction of shortening the springs. We note that the halving distance is necessary because i's neighbor on the other end of a spring may decide to lead simultaneously (this may happen in semi-synchronous model described later). For the same reason, the error bound e is doubled in all the equations. Hence, in order to make the movement in the desired direction DIR_i, the leader must obey

$$r_i^{m_i^{next}} \leftarrow r_i^{m_i} + \min(ST, SH) \cdot DIR_i \tag{1}$$

where

$$ST \leqslant \min_{j \in N(i)}((R - r_{ij}^{m_i})/2 - 2e) \cdot u_j \cdot DIR_i \tag{2}$$

is the minimal move among all neighbors of entity i in the direction of stretching the spring (u_j is the unit vector in the direction of stretching the spring between entities i and j), and

$$SH = \min_{j \in N(i)}((r_{ij}^{m_i} - r)/2 - 2e) \cdot u_j \cdot DIR_i \tag{3}$$

is the minimal move among all neighbors of entity i in the direction of shortening the spring. Recall that e is the error bound on the measured distance of neighboring entities, and $N(i)$ is the set of neighbors of entity i. Since this policy

is maintained by all pairs of neighboring entities, we are guaranteed that springs do not stretch over the connectivity limit R and at the same time do not shorten below the proximity limit r, assuming a non-critical starting point. Hence, the swarm remains connected and safety distances are maintained at all times.

Naturally, entities that do not wish to lead, should simply follow the regular Reynolds rules. Formally, in our spring system, this can be expressed in the following manner:

$$r_i^{m_i^{next}} \leftarrow r_i^{m_i} + dt^2 \sum_{j\in N(i)} u_j \cdot (r_{ij}^{m_i} - (R+r)/2)/2 + dt \sum_{j\in N(i)} v_j^{m_i} - \sum_{j\neq i} Correction_i^j \widehat{r_{ij}^{m_i}}, \quad (4)$$

where

$$Correction_i^j = max[0, Short_i^j, Stretch_i^j] \quad (5)$$

is the correction in the direction from $\widehat{r_{ij}^{m_i}}$ to entity's i movement in order to prevent the spring between entity i and entity j to shorten below r or stretch above R. Here we defined the shortening and stretching spring violations as

$$Short_i^j = r + (r_{ij}^{m_i} - r)/2 - |r_i^{m_i^{next}} - r_j^{m_i}| + 2e \quad (6)$$

$$Stretch_i^j = |r_i^{m_i^{next}} - r_j^{m_i}| - (R - (R - r_{ij}^{m_i})/2) + 2e \quad (7)$$

This way we make sure that entities from both ends of the spring will correct their movements in order to prevent violation of the spring Definition 2.8.

While the above set of behavioral rules for leaders and non leaders guarantees that no bidirectional edges in the swarm graph are being disconnected, it does allow the formation of new edges. Formally, each swarm member should update its neighborhood set at all cycles by executing

$$\forall j\ r_{ij}^{m_i} > R \Rightarrow N^{next}(i) \leftarrow N(i) - \{j\}$$
$$\forall j\ r_{ij}^{m_i} \leqslant R \Rightarrow N^{next}(i) \leftarrow N(i) \cup \{j\} \quad (8)$$

Notice, that Eq. 8 is applied only on directed links. It is never applied on bidirectional links.

3.1.2. Direction election in swarms with labeled entities

With the basic motion rules for leaders and non leaders formulated above, we are now ready to deal with direction election. Assume first that all the entities in the swarm are *labeled* and aware of their identity. In such cases, entities can be numbered and ordered by total ordering, and each entity can then be allocated unique time slots in which only it may become a leader. Let $ORDER_i$ be the label of entity i. For fairness, this time slot allocation can be done via round robin, using the global clock T^{global} to allocate T consecutive cycles to entity i, such that $ORDER_i == (T^{global} \bmod) + 1$, and T is the convergence rate of a swarm. It is an intrinsic constant in all algorithms presented below that is used to represent time slot duration.

Following the previous subsection, we hence obtain the following algorithm:

Algorithm 3.1. Direction election for entity i who wants to lead in flocks with n labeled entities

1: *for cycle* $= 1, \ldots, PT$ *do*
2: *if* $ORDER_i == \lfloor (cycle - 1)/T) + 1 \rfloor$ then *lead according to Eq. (1)*
3: else *obey Reynolds rules according to eq. (4)*
4: update *neighbor list according to eq. (8)*
5: end
6: go back *to step 1*.

Note that convergence rate properties are directly dependent on the constant T. In order to provide each leader with the best possibility to move in any desired leading direction, we may first wait until all the springs in the system are near their equilibrium. Note that this is the optimal position to allow each leader to move in any desired direction, since the equilibrium state of each spring is in the middle between r and R. So, we divide the time slot of duration T into 2 parts: one for spring network convergence to equilibrium state, and the second for leadership itself. Naturally, entity that does not want to lead in its designated time slots, simply obeys Reynolds rules according to Eq. 4.

Bidirectional springs are only allowed to be formed, but they are never removed under the direction election algorithm. Since the initial swarm graph is assumed connected, and the connectivity is defined on bidirectional springs only, then connectivity is preserved. Collisions are avoided by the definition of the unidirectional and bidirectional spring, which cannot become shorter than the proximity limit distance. Using this reasoning, we can state the following theorem:

Theorem 3.1. *The initial connectivity of a swarm is preserved and collisions between all entities are avoided under the direction election Algorithm 3.1, i.e. there is a path of springs connecting any pair of entities of a swarm during any stage of the algorithm.*

The time of starting the leading period for each entity is predetermined by its label, i.e. by its sequential number. Such a leader will lead starting from its first designated time slot for a duration of T cycles, a time at which the leading opportunity is passed to the next entity. Since the equilibrium state of all springs is in the middle between R and r, then any leader will obtain the possibility to move during its leading time slot, when it leads alone. Hence, we obtained the following result:

Theorem 3.2. *Direction election algorithm will cause swarm members to move in a single leading direction for periodic time slots of duration T cycles.*

3.1.3. Flocks of unlabeled entities

Assume now that the swarm consists of identical unlabeled entities which cannot be ordered. Clearly, while leading slots cannot be allocated deterministically in this case, entities can randomly choose their leading slot. However, this random selection should be done with care to ensure a similar fair chance to all entities to lead, and to prevent two entities from leading at the same time.

In our proposed solution, each entity runs the standard direction election algorithm for flocks with labeled entities, when a sequence of numbers $ORDER_i$ is randomly generated from the uniform distribution on the range $(1,\ldots,P)$, where P is the predetermined size of a period, whose optimal choice is described in Eq. (11).

Algorithm 3.2. Direction election for entity i who wants to lead in flocks with n anonymous entities

1: *Uniformly generate* $ORDER_i$ *on the range [1,P]*
2: for *cycle = 1, ..., PT* do
3: if $ORDER_i == \lfloor (cycle-1)/T)+1 \rfloor$ then *lead according to Eq. (1)*
4: else *obey Reynolds rules according to* Eq. (4)
5: update *neighbor list according to* Eq. (8)
6: end
7: go back *to step 1*.

Clearly, connectivity is preserved again and network topology is never reduced, hence we obtain:

Theorem 3.3. *The initial connectivity of a swarm is preserved and collisions between all entities are avoided under the direction election algorithm, i.e. there is a network of bidirectional springs connecting any pair of entities of a swarm during any stage of the algorithm.*

However, as is, the selection of $ORDER_i$ allows a small number of time slots, where multiple leaders compete for leadership. In such time slots, the swarm will flock according to the average of the leader directions [16].

Theorem 3.4. *Direction election algorithm will cause the swarm to follow a single leader for periodic time slots of duration T with predetermined probability, given the period duration P, where each entity can take leadership.*

Proof. By construction, the leading starting time of each entity is distributed uniformly in the range 1, ..., P. Let us first assume that a particular slot is selected by a *single* entity. Then, such a leader will lead by itself, starting from its time slot onward and until the slot of a next potential leader begins. Hence, in such a case, a single leader is elected and stable flocking of the swarm is achieved for time slot of duration T.

Of course, it is critical to understand how likely it is for any single candidate leader to find itself selecting a leading slot without conflicting with others. The number of ways to distribute k different entities into P different slots, where maximum a single entity is allowed in any slot is $P(P-1)\ldots(P-k+1)$. The total number of ways to distribute n different entities into P different slots is P^n. Let us assign $Number^{alone}_{leading}$ to the number of entities that lead alone during the period P. So, the probability that this number is at least k is:

$$Prob(Number^{alone}_{leading} \geqslant k) = \frac{P(P-1)\ldots(P-k+1)(P-k)^{n-k}}{P^n} \quad (9)$$

Increasing P for given n also increases the probability, due to the fact that $lim_{P\to\infty} Prob(Number^{alone}_{leading} \geqslant k)) = 1$. Let us assume that $Prob(Number^{alone}_{leading} \geqslant n) = \frac{1}{q}$, where q must be determined for each system, as stated below. Then, obviously, $\log_{(1/(1-q))}(n)$ sequential periods are needed to obtain the probability of any particular entity out of n to lead alone approximately approaching unity. □

Fig. 1 gives an example of the increasing probability of all the entities leading alone as a function of the period P duration in this case, for number of entities $n = 1\ldots20$. For the case of $n = 1$ the probability is exactly 1, and it drops down as n increases.

In order to choose the period length P consider the following argument:

$$\begin{aligned}\log Prob(Number^{alone}_{leading} \geqslant k) &= \log P(P-1)\ldots(P-k+1) \\ &+ (n-k)\log(P-k) - n\log P \\ &\geqslant n\log(P-k) - n\log P \\ &= -n\log\frac{P}{P-k} \geqslant -\log q \end{aligned} \quad (10)$$

Furthermore, one can choose P according to

$$P \geqslant \frac{kq^{\frac{1}{n}}}{q^{\frac{1}{n}}-1} \quad (11)$$

We should choose $k = n$, since this will reduce the total number of periods needed, as for larger k the lower bound for P in Eq. (10) is less strict.

In order to choose the last unknown variable q in Eq. (11) we need to solve the following optimization problem:

$$minimize_q \left[\log_{(1/(1-q))}(n)\frac{nq^{\frac{1}{n}}}{q^{\frac{1}{n}}-1}\right] \quad (12)$$

Taking the derivative of the minimization function with respect to q we realize that it vanishes at a single minimum of the minimization function (see Fig. 2), which is the desired optimal value.

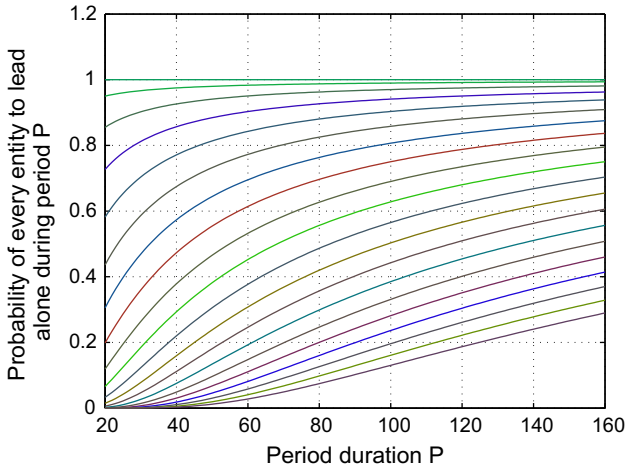


Fig. 1. Probability of all n entities leading alone vs. period duration P, $n = 1\ldots20$.

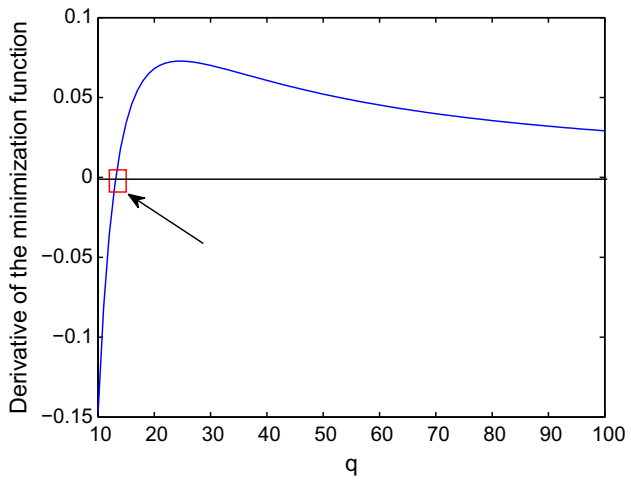


Fig. 2. Derivative of the minimization function with respect to q, $n = 10$.

Fig. 2 gives an example of the derivative of the minimization function with respect to q, for number of entities $n = 10$.

3.2. Semi-synchronous networks

Consider a relaxed version of our synchronous network, in which time slots of different entities are allowed to have unknown, but bounded size, phase shifts. We argue that the direction election algorithm for flocks with labeled entities 3.1 described in subsection 3.1 as well as direction election algorithm for flocks with anonymous entities described in subsection 3.2 work when no synchronization between entities exists, while an additional synchronization part in time slot T should be devoted, in order to preserve the probabilistic properties of these algorithms. So, the structure of the time slot T contains now: the convergence part, as earlier, the synchronization part that is equal to the maximal allowed phase shift, and the leading part.

Indeed, allowing entities to lead only after the additional synchronization part of the time slot T, we obtain the leadership for duration T, while no two entities with different $ORDER_i$ values can compete for leadership. In addition, Eq. (5) implies that no spring violations can occur when cycle size dt is constant among different entities and only the measurement time starts with a different phase. But note that this situation is covered by the synchronous case, since here the maximal possible movement size is less than the similar quantity in the synchronous case.

Theorem 3.5. *Direction election algorithm 3.1 preserves connectivity and avoids collision between entities for partially asynchronous systems.*

Proof. We show that by running the synchronous direction election algorithm in partially asynchronous system no violation of the spring Definition 2.8 happens. Here, each entity performs its measurements with the same time periods dt, but with different starting time phase. In particular, let us assume, without loss of generality that entity i performs its measurements at time sequence $\{0, dt, 2dt, \ldots, kdt, \ldots\}$, while entity j performs its measurements at time sequence $\{ph, dt + ph, 2dt + ph, \ldots, kdt + ph, \ldots\}$, where $0 < ph < dt$ stands for the phase shift of entity j. Phase shifts may affect the algorithm only in two situations: when making a movement as a leader, and while obeying Reynolds rules. In both cases, the maximal movement for entity i starting at time 0 is $(R - r_{ij}^{m_i})/2$ in the direction of stretching the spring between i and j, and $(r_{ij}^{m_i} - r)/2$ in the direction of shortening the spring between i and j, where $r_{ij}^{m_i}$ is the length of the spring at time 0 measured by entity i. By Definitions (2.9) and (2.1) the velocity between subsequent measurements is assumed to be constant. Thus, until the time ph entity i passes a distance of

$$S_i^{ph} = \frac{ph}{dt}\left(R - r_{ij}^{m_i}\right)\Big/2 \tag{13}$$

in the direction of stretching the spring between i and j, and

$$s_i^{ph} = \frac{ph}{dt}\left(r_{ij}^{m_i} - r\right)\Big/2 \tag{14}$$

when it takes the direction which shortens the spring. Then, entity j makes its movement, according to the measurement at time ph. Entity j made until the time dt a distance of exactly

$$S_j^{dt} = \frac{dt - ph}{dt}\left(\left[R - \frac{ph}{dt}(R - r_{ij}^{m_i})\Big/2\right] - r_{ij}^{m_i}\right)\Big/2 \tag{15}$$

in the direction of stretching the spring between i and j, or

$$s_j^{dt} = \frac{dt - ph}{dt}\left(\left[r_{ij}^{m_i} - \frac{ph}{dt}(r_{ij}^{m_i} - r)\Big/2\right] - r\right)\Big/2 \tag{16}$$

unit distance if it takes the direction which shortens the spring. Thus, at time dt, after a complete cycle of entity i, the difference between R and the maximal length of the spring between i and j is

$$\begin{aligned} NV^{dt} &= R - L^{dt} \\ &= R - \left\{ r_{ij}^{m_i} + (R - r_{ij}^{m_i})/2 + \frac{dt - ph}{dt} \right. \\ &\quad \left. \times \left(\left[R - \frac{ph}{dt}(R - r_{ij}^{m_i})\Big/2\right] - r_{ij}^{m_i}\right)\Big/2 \right\} \\ &\geqslant (R - r_{ij}^{m_i})/2 - \left(\left[R - \frac{ph}{dt}(R - r_{ij}^{m_i})\Big/2\right] - r_{ij}^{m_i}\right)\Big/2 \\ &= \frac{ph}{dt}(R - r_{ij}^{m_i})/4 \geqslant 0 \end{aligned} \tag{17}$$

It means that the length of the spring between i and j is less or equal to R for this case. Also, at time dt, after a complete cycle of entity i, the difference between the minimal length of the spring between i and j and r is

$$
\begin{aligned}
n v^{dt} &= l^{dt} - r \\
&= r_{ij}^{m_i} - (r_{ij}^{m_i} - r)/2 \\
&\quad - \frac{dt - ph}{dt}\left(\left[r_{ij}^{m_i} - \frac{ph}{dt}(r_{ij}^{m_i} - r)\Big/2\right] - \right)\Big/2 - r \\
&\geqslant (r_{ij}^{m_i} - r)/2 - \left(\left[r_{ij}^{m_i} - \frac{ph}{dt}(r_{ij}^{m_i} - r)\Big/2\right] - \right)\Big/2 \\
&= \frac{ph}{dt}(r_{ij}^{m_i} - r)/4 \geqslant 0
\end{aligned} \tag{18}
$$

Thus, the spring's length between i and j is larger than or equal to r for this case.

Proceeding by mathematical induction, the spring size between i and j is less than or equal to R and greater or equal to r after any number of cycles. The same is true for all other springs in the network, since the spring between i and j has been chosen arbitrarily. □

Theorem 3.6. *Given P, direction election Algorithm 3.1 will make the swarm follow a single leader at least k times in a period P for periodic time slots of duration at least T with predetermined probability.*

Proof. A time of starting the leading slot for each leader is uniformly generated upon the range $1, \ldots, P$. Such a leader will lead alone starting from its time slot on, until the next leader in the sequence wants to lead. Hence, single leader is elected and stable flocking of the swarm is achieved for time slot of duration T. The probability that at least k entities lead alone is equal to the probability in Eq. (11), since exactly the same probabilistic rule is applied in this case. Since the equilibrium state of all springs is in the middle between R and r, then after the spare part of the time slot T, dedicated for spring network convergence, all the springs will stay near their equilibrium in the middle between r and R. Then any leader will obtain the possibility to move during its leading time slot, when it leads alone. □

3.2.1. Real semi-synchronous networks

In the semi-synchronous networks, entities can start counting period P at different times. The *liveness* of an algorithm ensures that all system constraints stay inviolated when algorithm is applied to the system. The next theorem shows that synchronous direction election algorithm 3.1 is also applicable for this case.

Since the statement of liveness for networks with nonsynchronized clocks Theorem 3.5 and its proof do not depend on the period starting time, we obtain:

Theorem 3.7. *Liveness for networks with nonsynchronized clocks Theorem 3.5 is applicable for semi-synchronous networks.*

Theorem 3.8. *Progress for synchronous network with time shift Theorem 3.6 is applicable for semi-synchronous networks.*

Proof. Let us assume without loss of generality that entity i starts counting its period P from a time $t = -\varepsilon$. Its $ORDER_i$ value is uniformly distributed on the range $[1 - \varepsilon, P - \varepsilon]$ that is $Prob(ORDER_i = o, o \in [1 - \varepsilon, P - \varepsilon]) = \frac{1}{P}$. Immediately after the period of duration P, another period of duration P starts, where $ORDER_i$ value of entity i is also uniformly distributed that is $Prob(ORDER_i = m, m \in [P - \varepsilon, 2P - \varepsilon]) = \frac{1}{P}$. Then $ORDER_i$ value of entity i must be uniformly distributed on the range $[1, P]$. Indeed, $Prob(ORDER_i = q, q \in [1, P]) = \frac{1}{P}$. The same is true for every entity in the system. Since the only assumption on period P was the uniform distribution of $ORDER_i$ value of each entity on the range $[1, P]$ in the statement and the proof of theorem 3.4, then it is also applicable for Semi-synchronous Networks. □

4. Monitoring and leader following

Let us show that all the proposed direction election algorithms allow all flock members monitor the movement of the swarm in the leadership direction, while they perform fast convergence to the leader velocity. We base this conclusion on the fact that the movement of the center of mass of the system is affected by leading external force alone, while all other internal forces in the system cancel each other. This reasoning will bear on the symmetry properties of any heavily populated network around its center of mass.

It can be realized that due to the possible measurement errors, the newly created network edges may not be bidirectional. In this situation, these new edges can be broken by the unawared neighbor. Note that this case is not worse than the case without the newly created directed neighborship as the connectivity is maintained by the links for which both endpoint entities measure distance less than R.

For heavily populated networks, a reasonable assumption one can make is that entities are located uniformly around the center of mass of the system. Due to this fact, we obtain symmetry properties regarding the unidirectional springs formation in any direction in the system. These properties lead to the conclusion that in the case of a single leader the overall additional external force induced by the unidirectional springs vanishes. So, practically, only the leading force is applied to the center of mass of the system.

To conclude this elementary case, we do note that an upper bound for the convergence rate of a swarm to a single leader may permit slight deviation from the exact speed and direction of the leader. Furthermore, in many practical situations, explicit convergence of all entities to the exact velocity of the leader is not crucial, but the movement of the swarm's center of mass in the direction of a leader, up to a small predetermined deviation, is sufficient. While the limits r and R are not approached, which is prohibited by the spring definition, the center of mass of the swarm is given under the external leading force only, as stated above. In this ideal case, the convergence of entities to the approximate leader velocity (up to an error) will happen immediately, because they are given under the single leading force only. Due to the real system symmetry

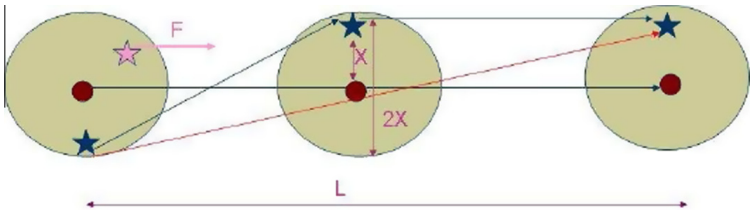


Fig. 3. Relative error bound change with time.

imperfections, the convergence should take time, which might be polynomial in heavily populated networks.

In Fig. 3 the leader applies the leading force F for the time t_1 until the velocity of the flock attains the desired value. Another entity is moving between the most extremal possible positions under the influence of internal forces in the network. Still, the measure of self velocity of this last entity converges to the velocity of center of mass as time passes.

While using the spring network abstraction, real masses and, therefore, accelerations of different entities are not taken into account, but only their effective values and inter-entity distances. The movement of a particular entity is limited only by its physical characteristics. Let us assume that due to physical limitations the maximal allowed acceleration is a_{max}, so that the maximal force each entity can apply is limited by F and its effective mass is thus m_{eff}. Here, $F = m_{eff}a_{max}$. Recall that only the leader's force affects the center of mass of the system. Using this fact, our algorithm allows monitoring of movement of the swarm in the desired direction by all its members. Indeed, from Fig. 3 we conclude that the relative angular error in the leadership direction measurement by any entity is bounded by $\frac{2X}{L}$, where X is the maximal deviation of an entity from the center of mass, and L is the way passed by the center of mass in the direction of the leadership. We note that the relative error decreases with increase in L. Let us assume the total mass of the network to be one unit. By regular laws of accelerated motion,

$$L = V_0 t_0 + F t_1^2 + V^{leader} t \tag{19}$$

where V^{leader} is the desired leadership velocity. X by itself is bounded by nR, since no spring can stretch beyond R by definition. Hence, the relative angular error is bounded by

$$\frac{2nR}{Ft_1^2 + V^{leader}t} \approx \frac{2nR}{V^{leader}t} \tag{20}$$

Thus, after a predetermined time period each entity can infer the leadership direction and speed, up to a known error bound.

The stated above possibility of monitoring the leading direction by every entity gives an opportunity of the most wanted direction election (which has been chosen majority of entities during the algorithm) by among different leading directions. All the potential leaders may be allowed to lead simultaneously for a given period of time, while monitoring. After this time period, swarm's velocity will attain, up to a bounded error, the most wanted velocity direction among all leaders. So, the leader with closest leading goal (in other words, the leader with the direction that is closest to the most wanted direction) will obtain the maximal possibility to lead.

Clearly, the algorithm outlined so far would operate in the presence of multiple leaders, but it does not guarantee that these leaders would not conflict each other to stall the swarm as a whole or to act in a diverging behavior forever.

5. Conclusions

In this work we enhance Reynolds Boids rules to enable symmetry breaking, handle measurement errors and support direction election. We prove correctness and believe that the schemes presented can be used in practice. The constant value T that is used in our algorithms can be experimentally (or using simulations) measured for the specific swarm settings; as we pointed out we suspect that the value of T proved in [1,2] is an upper bound on convergence rate in our model. The algorithms presented can be tuned for the cases in which various threats or flocking goals with different urgencies coexist in the system. To address this scenario we can introduce priority mechanism. The priority variable N_P will have a predetermined scale of values, dividing the priority to different scenarios present in the system. Then, direction election is influenced by the priority value of each entity. For this scheduler, we have that for a random generation of $ORDER_i$ a period of P is multiplied by the priority category number N_P, starting from $N_P = 1$ for the highest priority.

The priority of each entity should be calculated asynchronously at the time of an appropriate scenario arrival to the entity. Each entity would perform such priority calculations all the time according to a predetermined formula. At this moment, new period starts for this entity, and the period duration for it is influenced appropriately.

Using the following reasoning, we argue that all presented direction election algorithms will preserve their properties under this new priority scheme. In particular, they will make the swarm to follow a single leader for periodic time slots of duration T with arbitrarily high predetermined probability, since the time of starting the leading time slot for each leader is uniformly generated upon the range $1, \ldots, N_P P$. Such a leader will lead alone starting from his time slot on, until the next leader in the sequence wants to lead. This may happen when another leader was previously scheduled or the priority scheduling took place. So, single leader is elected and stable flocking of the swarm is achieved, for time slot of duration T.

Also, the probability that at least k entities lead alone is greater or equal to the probability in Eq. (11), since exactly the same probabilistic rule is applied in the case, where all

entities have equal priority, otherwise, the probability is higher, since the number of entities in each period P is less than n.

Acknowledgement

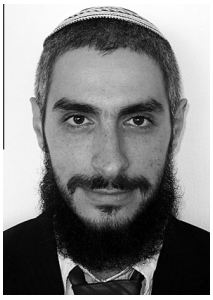
We thank anonymous reviewers whose comments significantly improved the presentation of the paper.

References

[1] B. Chazelle, Natural algorithms, in: Proc. 20th SODA, pp. 422–431, 2009.
[2] B. Chazelle, The geometry of flocking, in: Proc. ACM Symp. on Comp. Geometry, 2010, pp. 19–28.
[3] M. Fischer, H. Jiang, Self-stabilizing direction election in networks of finite-state anonymous agents, OPODIS (2006) 395–409.
[4] J.M, Hendrickx, V.D. Blondel, Convergence of different linear and non-linear Vicsek models, in: Proc. 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS2006), Kyoto (Japan), July 2006, pp. 1229–1240.
[5] A. Jadbabaie, J. Lin, A.S. Morse, Coordination of groups of mobile autonomous agents using nearest neighbor rules, IEEE Trans. Autom. Control 48 (6) (2003) 988–1001.
[6] M. Ji, M. Egerstedt, Distributed formation control while preserving connectedness, IEEE Trans. Robotics 23 (4) (2007).
[7] H. Jiang-Ping, Y. Hai-Wen, Collective coordination of multi-agent systems guided by multiple leaders, IEEE Trans. Robotics 18 (9) (2009).
[8] S. Li, H. Wang, Multi-entity coordination using nearest neighbor rules: revisiting the Vicsek model, arXiv:cs/0407021v2, 2004.
[9] N. Malpani, J. Welch, N. Vaidya, Direction Election Algorithms for Mobile Ad-hoc Networks, DIAL-M: Workshop in Discrete Algorithms and Methods for Mobile Computing and Communications, 2000.
[10] L. Moreau, Stability of multiagent systems with time-dependent communication links, IEEE Trans. Autom. Control 50 (2005) 169–182.
[11] R. Olfati-Saber, R.M. Murray, Consensus problems in networks of agents with switching topology and time-delays, IEEE Trans. Autom. Control 49 (9) (2004) 1520–1533.
[12] R. Olfati-Saber, Flocking for multi-entity dynamic systems: algorithms and theory, IEEE Trans. Autom. Control 51 (2006) 401–420.
[13] C. Reynolds, Flocks, birds and schools: a distributed behavioral model, Comput. Graphics 21 (1987) 25–34.
[14] H. Shi, L. Wang, T. Chu, Coordination of multiple dynamic agents with asymmetric interactions, in: Proc. 2005 IEEE International Symposium on Intelligent Control, Cyprus, pp. 1423–1428, 2005.
[15] N. Sorensen, W. Ren, A unified formation control scheme with a single or multiple leaders, in: Proceedings of the American Control Conference, New York, 2007, pp. 5412–5418
[16] H. Su, X. Wang, W. Yang, Flocking in multi-entity systems with multiple virtual leaders, Asian J. Control 10 (2) (2008) 238–245.
[17] G. Tang, L. Guo, Convergence of a class of multi-entity systems in probabilistic framework, J. Syst. Sci. Complex. 20 (2007) 173–197.
[18] H.G. Tanner, A. Jadbabaie, G.J. Pappas, Stable flocking of mobile agents, Part II: dynamic topology, in: Proc. IEEE Conference on Decision and Control, Maui, Hawaii, 2003, pp. 2016–2021.
[19] H.G. Tanner, A. Jadbabaie, G.J. Pappas, Flocking in fixed and switching networks, IEEE Trans. Autom. Control 52 (5) (2007).

Ohad Ben-Shahar received the B.Sc. and M.Sc. degrees in computer science from the Technion, Israel Institute of Technology, and the Ph.D. degree in computer science from Yale University, Connecticut, in 2003. He is an associate professor in the Department of Computer Science at Ben Gurion University (BGU) of the Negev and the director of the BGU interdisciplinary Computational Vision Lab (iCVL). His main area of research is in computational vision and image analysis, where he is focusing primarily on issues related to the differential geometrical foundations of perceptual organization and early vision. His work is explicitly multidisciplinary and his computational research is often endowed by investigations into human perception, visual psychophysics, and computational neuroscience of biological vision.

Shlomi Dolev received his B.Sc. in Engineering and B.A. in Computer Science in 1984 and 1985, and his M.Sc. and D.Sc. in computer Science in 1990 and 1992 from the Technion Israel Institute of Technology. From 1992 to 1995 he was at Texas A&M University postdoc of Jennifer Welch. In 1995 he joined the Department of Mathematics and Computer Science at Ben-Gurion University where he is now an full professor. He was a visiting researcher/professor at MIT, DIMACS, and LRI, for several periods during summers. He is the author of the book "self-stabilization" published by the MIT Press. He published more than two hundreds journal and conference scientific articles, and patents. He served in the program committee of more than eighty and he is an associate editor of several journals including the IEEE Transactions on Computers and the AIAA Journal of Aerospace Computing, Information and Communication. His research grants include IBM faculty awards, Intel academic grants, the ISF and the NSF. He is the founding chair of the computer science department at Ben-Gurion university, where he now holds the Rita Altura trust chair in computer science and the head of the Frankel center for computer science. Shlomi serves as the dean of the natural sciences of Ben Gurion University, and the chair of the inter-university computation center of Israel. His current research interests include distributed computing, distributed systems, security and cryptography and communication networks; in particular the self-stabilization property of such systems. Recently, he is involved in optical computing research.



Andrey Dolgin received B.Sc. M.Sc. and Ph.D. degrees in Computer Science, Electrical Engineering, and Industrial Engineering and Management departments in Technion – Israel Institute of Technology. Afterwards, he proceeded to a postdoctoral fellowship in Computer Science and Communications in Ben-Gurion University of the Negev. His research interests include randomized algorithms, mobile communications, optical and classical communications.

Michael Segal finished B.Sc. M.Sc. and Ph.D. degrees in computer science from Ben-Gurion University of the Negev in 1994, 1997, and 1999, respectively. During a period of 1999–2000 Prof. Michael Segal held a MITACS National Centre of Excellence Postdoctoral Fellow position in University of British Columbia, Canada. Prof. Segal joined the Department of Communication Systems Engineering, Ben-Gurion University, Israel in 2000 where he served as department's Chairman between 2005–2010. He published more than 120 journal and conference papers on topics including algorithms (sequential and distributed), data structures with applications to optimization problems, mobile wireless networks, scheduling and efficient networking.