Swarm Intelligence (2023) 17:5–26
https://doi.org/10.1007/s11721-022-00214-z

Noise-resistant and scalable collective preference learning via ranked voting in swarm robotics

Qihao Shan[1] · Sanaz Mostaghim[1]

Received: 22 December 2021 / Accepted: 4 August 2022 / Published online: 5 September 2022

Abstract
Swarm robotics studies how to use large groups of cooperating robots to perform designated tasks. Given the need for scalability, individual members of the swarm usually have only limited sensory capabilities, which can be unreliable in noisy situations. One way to address this shortcoming is via collective decision-making, and utilizing peer-to-peer local interactions to enhance the behavioral performances of the whole swarm of intelligent agents. In this paper, we address a collective preference learning scenario, where agents seek to rank a series of given sites according to a preference order. We have proposed and tested a novel ranked voting-based strategy to perform the designated task. We use two variants of a belief fusion-based strategy as benchmarks. We compare the considered algorithms in terms of accuracy and precision of decisions as well as the convergence time. We have tested the considered algorithms in various noise levels, evidence rates, and swarm sizes. We have concluded that the proposed ranked voting approach is significantly cheaper and more accurate, at the cost of less precision and longer convergence time. It is especially advantageous compared to the benchmark when facing high noise or large swarm size.

Keywords Swarm robotics · Collective decision-making · Collective preference learning · Self-organized systems

1 Introduction and related works

Swarm robotics refers to a design paradigm that employs the intelligent collective behavior of a group of robots to achieve a designated task (Brambilla et al., 2013). Inspirations are taken from natural intelligent swarms, such as insect colonies and fish schools, who can effectively pool information from agents with poor individual capabilities and

All data generated or analyzed during this study are included in this published article.

✉ Qihao Shan
qihao.shan@ovgu.de

Sanaz Mostaghim
sanaz.mostaghim@ovgu.de

[1] Faculty of Computer Science, Otto von Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Saxony-Anhalt, Germany

display complex collective behaviors without centralized control mechanisms (Camazine et al., 2020). Collective decision-making is a field within the study of swarm intelligence, which focuses on the process where a swarm of intelligent agents achieve a global decision via only local interactions among each other and with the environment. This field of study has its roots in attempting to model and understand natural intelligent swarms (Garnier et al., 2007) and has also been increasingly utilized to construct decision-making strategies for artificial intelligent swarms (Dorigo et al., 2021).

Best-of-n problems refer to collective decision-making problems that focus on discrete consensus forming (Valentini et al., 2017). Site selection is a long-standing studied scenario among best-of-n problems. It takes inspirations from the house-hunting behaviors of honey bees (Garnier et al., 2007), which is an example of decentralized decision-making in natural intelligent swarms. Similar scenarios are used to gauge the capabilities of artificial intelligent swarms in collective decision-making. The experimental setup of site selection problems started with binary environments with two sites in the arena (Parker & Zhang, 2009, 2011) and have gradually evolved into multi-site environments (Lee et al., 2018; Talamali et al., 2019). Recently, there has been a trend in the broader collective decision-making research to move from simple binary decision-making scenarios and toward enabling the agents to make more complicated decisions, such as multicolor collective perception (Ebert et al., 2018; Bartashevich & Mostaghim, 2021), collective estimation (Strobel et al., 2018; Shan & Mostaghim, 2021), and the aforementioned multi-option site selection.

In swarm robotics, learning the ranking of a number of options according to their relative preference is an important operation, that has many real-life applications and can also serve as building blocks for more complex behaviors. To perform such preference learning in a swarm robotics setting, the robots need to converge to a consensus regarding the ranking of available options using a distributed and localized strategy. In this paper, we tackle such a collective preference learning problem, where the group of agents is tasked with ranking the available options from best to worst. A similar problem has been addressed in a non-physics-based environment by Crosscombe and Lawry (2021). They have proposed a belief fusion-based algorithm to achieve consensus in the ranking.

Another potential source of inspiration for collective decision-making strategies is the election process, where the voters collectively decide among the available candidates (Tideman, 2017). However, for the distributed decision-making processes of intelligent swarms, a centralized tallying of all the ballots cannot be performed. Thus, in swarm intelligence settings, majority-voting-based decision-making strategies usually implement local-scale voting among small groups of agents, such as in Direct Modulation of Majority-based Decisions (Valentini et al., 2015, 2016), which gives good performances in various binary decision-making scenarios. However, in more complex scenarios, a simple majority voting tends to be insufficient. Here, we focus on a ranked voting system, Borda count, which was proposed by Jean-Charles de Borda in the eighteenth century (Emerson, 2013). Consensus formation using iterative pairwise voting has been studied in the context of social networks (Hassanzadeh et al., 2013; Brill et al., 2016; Guha & Dasgupta, 2021), where its ability to converge the agents with different opinions to a consensus was proven. Similar ranked voting techniques have also already been utilized in another collective decision-making scenario, discrete collective estimation, by Shan et al. (2021).

In this paper, we seek to apply a ranked voting-based decision-making strategy to perform collective preference learning, with the aforementioned belief fusion strategy as a baseline. We will test the considered algorithms in physics-based simulated environments with different noise values, rate of evidence, and swarm size.

The structure of the paper is as follows. In Sect. 2, we will introduce the collective preference learning problem we are investigating in this paper and the background of the algorithms investigated. In Sect. 3, we will show the two considered algorithms in this paper in detail. Section 4 includes the experimental results. Section 5 is our analysis and discussion on the experimental results. And finally, Sect. 6 is the conclusion.

2 Problem statement

We investigate a preference learning problem inspired by classical site selection scenarios as well as the collective preference learning scenario investigated in Crosscombe and Lawry (2021). An illustration of the environment is shown in Fig. 1. There are K sites distributed over the arena. A swarm of N_{robot} robots is shown in red. They roam the experimental environment and are tasked with ranking the sites from best to worst.

In our experimental settings, each site is associated with a fix index and quality, the latter indicated by the intensities of the gray color in Fig. 1. When an agent is over a site area, it has a probability of detecting the site per control loop, referred to as the evidence rate r_e. When a site is detected, the agent records the index and the quality of the site. The former is measured accurately, while the latter is subject to an additive Gaussian noise, with the mean being 0 and the standard deviation being the noise level σ_{noise}.

An agent has limited computational and memory resources and can only record the indices and measured qualities of two sites. The agent will thus obtain a pairwise comparison between them. Depending on the decision-making strategy, the agent records its own computed ranking among all sites. In addition, an agent has a limited communication radius and can only broadcast and receive messages to its peers within the communication radius.

We use the belief fusion algorithm proposed by Crosscombe and Lawry (2021) as a baseline. They encode the full ranking in a $K \times K$ matrix that indicates pairwise comparison between all available pairs of sites. An element $o_{i,j}$ in the matrix can take one of three values, 0 and 1 mean that the agent believes that $quality_i < quality_j$ and $quality_i > quality_j$, respectively, while 1/2 means that the comparison is unknown. Their experiments have

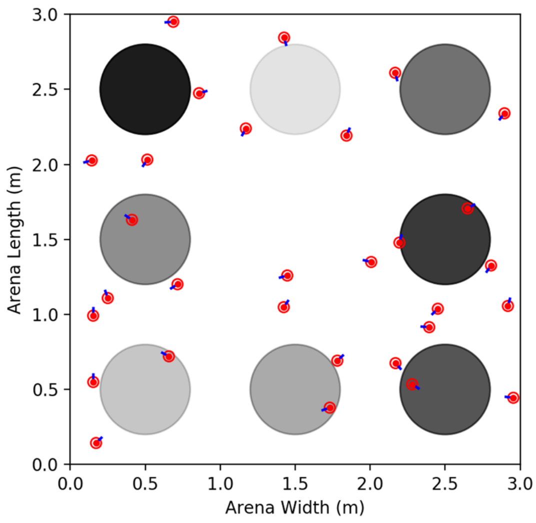


Fig. 1 Illustration of the simulated experimental environment used in this paper; gray circular areas represent the $K = 8$ sites, their color intensities represent their qualities; red dots indicate mobile robots roaming the arena (Color figure online)

been conducted in a non-physics-based environment. At every control loop, every agent tries to obtain an unknown pairwise comparison between two sites with a success probability. Two agents from the swarm also perform belief fusion and combine their beliefs together. In our paper, we have applied this algorithm to our aforementioned environment.

Additionally, we seek to apply a ranked voting-based decision-making strategy to this problem. We have chosen Borda count (Emerson, 2013) as a promising ranked voting system. The original voting system works as follows. Each voter ranks all candidates according to the own preference, from best to worst. During the tallying process, every candidate receives a number of points according to every ballot. If there are n candidates, the most preferred candidate on a ballot receives n points, the next most preferred $n-1$ points, and the least preferred 1 point. These points are added up for all candidates, and the candidate with the most points wins the voting. The same voting system can also be used to obtain a consensus in the ranking of the available candidates by looking at the ranking of the final tallied points. We use this approach in the collective preference learning problem. The details of our algorithm are shown in the Methodology section.

3 Methodology

In this section, we describe the algorithms considered in detail. We start with how the robots obtain pairwise quality comparisons from the raw quality reading. Then, we cover the state-of-the-art approach in solving similar collective preference learning problems. After that, we will introduce our proposed ranked voting algorithm. Finally, we define the computation method we use for the performance metrics.

3.1 Obtaining pairwise quality comparisons

In both considered algorithms, we use the same assumptions used by Crosscombe and Lawry (2021) on the cognitive capabilities of the robots considered. Robot n keeps track of the indices and qualities of two sites, expressed as follows:

$$\begin{aligned} Indices &: D_n = \{d_{n,1}, d_{n,2}\} \\ Qualities &: Q_n = \{q_{n,1}, q_{n,2}\} \end{aligned}$$

All four variables are initialized to -1. At every control loop, every robot who is in the area of a site has a probability (evidence rate r_e) to detect the site and updates its recorded pairwise comparison using Algorithm 1.

Algorithm 1 Update Pairwise Comparisons

```
1: $D_n = \{-1, -1\}$; $Q_n = \{-1, -1\}$; $n = 1..N_{robot}$
2: procedure $update\_pair(n, d*, q*)$
3:     if $d_{n,1} = d*$ or $d_{n,1} = -1$ then
4:         $d_{n,1} = d*$; $q_{n,1} = q*$
5:     else if $d_{n,2} = d*$ or $d_{n,2} = -1$ then
6:         $d_{n,2} = d*$; $q_{n,2} = q*$
7:     else
8:         $ind = RandomChoice\{1, 2\}$
9:         $d_{n,ind} = d*$; $q_{n,ind} = q*$
10:    end if
11:    if $q_{n,1} < q_{n,2}$ then
12:        $Switch(d_{n,1}, d_{n,2})$; $Switch(q_{n,1}, q_{n,2})$
13:    end if
14: end procedure
```

The detected site index $d*$ and measured quality $q*$ are recorded. If $d*$ is present in D_n, the robot updates the associated quality value with $q*$ (line 3–6). In this paper, the robots do not take repetitive measurements of the noisy site quality to determine the true value, as it is assumed that necessary procedures to minimize the noise have been implemented at low level. If one value in D_n is -1, indicating the position is empty, a new $d*$ also fills the position (line 3–6). If both values in D_n are filled and are not equal to $d*$, then one of the two positions is selected at random and filled with $d*$ and $q*$ (line 8–9). Finally, the robot always preserves the ordering $q_{n,1} \geq q_{n,2}$, so that if this is no longer the case after updating then the values in both D_n and Q_n will be switched (line 12).

3.2 Benchmark algorithm: belief fusion

The state-of-the-art strategy to solve a collective preference learning problem is a belief fusion-based algorithm proposed by Crosscombe and Lawry (2021). They have experimented on two variants of it, one with an operation that preserves the transitivity in pairwise comparisons, and the other without, producing different results. We use a modified version of it with both variants considered as benchmarks in this paper. The detailed pseudocode is shown in Algorithm 2.

Algorithm 2 Collective Preference Learning using Belief Fusion

```
1: $B_n = zeros(K, K)$; $n = 1..N_{robot}$
2: Initialize: $r_e$, $\sigma_{noise}$, $f_t$
3: procedure $preference\_learning\_fusion(n)$
4:     if Robot n is on site k & Site detected with probability $r_e$ then
5:         $update\_pair(n, k, sample(N(quality_k, \sigma^2_{noise})))$ # shown in Algorithm 1
6:         if $logical\_and(D_n >= 0)$ then
7:             $B_n[d_{n,1}, d_{n,2}] = 1$; $B_n[d_{n,2}, d_{n,1}] = -1$
8:         end if
9:     else if Other robots in communication radius of robot n then
10:        $m = RandomChoice\{$Indices of neighboring robots$\}$
11:        $B_n = B_n + B_m$
12:        $B_n[B_n \neq 0] = B_n[B_n \neq 0]/abs(B_n[B_n \neq 0])$
13:    end if
14:    if $f_t$ i.e. transitivity needs to be preserved then
15:        for $k = 1..K$ do
16:            for $k1 = 1..K$ do
17:                for $k2 = 1..K$ do
18:                    if $B_n[k, k1] = 1$ & $B_n[k, k2] = -1$ & $B_n[k1, k2] = 0$ then
19:                        $B_n[k1, k2] = -1$; $B_n[k2, k1] = 1$
20:                    end if
21:                end for
22:            end for
23:        end for
24:    end if
25: end procedure
```

The belief matrix B_n records the pairwise relationship between all possible pairs of sites. Element $B_n[k1, k2]$ can take one of three values, 1 when $quality_{k1} > quality_{k2}$, -1 when $quality_{k1} < quality_{k2}$, and 0 when the pairwise relationship is unknown or when $k1 = k2$. The overall behavior of the robot is similar to in Algorithm 2. One important difference is in Algorithm 2 line 7, where the robot updates the belief matrix B_n by modifying the corresponding elements. In the original version of the algorithm, a belief fusion operation changes the belief matrices of both robots, thus requiring bidirectional communication. We have modified this feature to keep the hardware requirements on the same level as our proposed algorithm. The robot broadcasts its belief matrix B_n constantly to its nearby neighbors. In practice, due to $B_n[a, b] = -B_n[b, a]$, only half of the matrix needs to be transmitted. At every control loop, it picks up the belief matrix of a random neighbor and performs belief fusion to update its own belief matrix (Algorithm 2 line 11–12). The message transfers are peer-to-peer and pairwise. There are no requirements for the robots to be uniquely identifiable.

Another important operation in the belief fusion algorithm is the preservation of transitivity in pairwise comparisons in the belief matrix (Algorithm 2 line 14–24). Here, f_t is a Boolean variable that marks this setting. The operation makes sure when the belief matrix B_n records $q_a > q_b$ and $q_b > q_c$, it will automatically also record $q_a > q_c$. Since the operation needs to traverse the whole matrix K times, it is an expensive operation with complexity scaling to K^3 and presents a trade-off between performance and computational resources needed. We have thus experimented on the benchmark belief fusion algorithm both with and without operations to preserve transitivity in the decision-making process for a full comparison with the proposed ranked voting algorithm.

3.3 Collective preference learning via ranked voting with Borda count tallying process

We will now introduce the proposed ranked voting-based decision-making strategy. The decision-making behavior of the robot *n* using the proposed strategy is shown in Algorithm 3, while Algorithms 4 and 5 are subroutines used in the algorithm. In this algorithm, the robot *n* encodes the ranking of all known sites in a list $\mathbf{ranking}_n$, which is empty at initialization.

$$Initial : \mathbf{ranking}_n = []$$

The maximum length of $\mathbf{ranking}_n$ is the total number of sites *K*. At every control loop, the robot attempts to detect a potential site. A site will only be detected when the robot is in the marked area of the site and a random variable satisfies the evidence rate r_e (Algorithm 3 line 4).

Algorithm 3 Collective Preference Learning using Ranked Voting

```
1: $\mathbf{ranking}_n = []$; $n = 1..N_{robot}$
2: Initialize: $r_e$, $\sigma_{noise}$
3: procedure $preference\_learning\_voting(n)$
4:     if Robot n is on site k & Site detected with probability $r_e$ then
5:         $update\_pair(n, k, sample(N(quality_k, \sigma^2_{noise})))$ # shown in Algorithm 1
6:         if $d_{n,1} >= 0$ and $d_{n,2} >= 0$ then
7:             $update\_ranking(n, D_n)$ # shown in Algorithm 4
8:         end if
9:     else if Other robots in communication radius of robot n then
10:        $m = RandomChoice\{$Indices of neighboring robots$\}$
11:        $\mathbf{ranking}_n = election(\mathbf{ranking}_n, \mathbf{ranking}_m)$ # shown in Algorithm 5
12:        if $d_{n,1} >= 0$ and $d_{n,2} >= 0$ then
13:            $update\_ranking(n, D_n)$ # shown in Algorithm 4
14:        end if
15:    end if
16: end procedure
```

If a site is detected, the robot updates its internal record of a pairwise comparison using the procedure *update_pair* (Algorithm 3 line 5) introduced in Sect. 3.1 Algorithm 1, and with the index and measured quality of the detected site as input. After that, if both positions in its pairwise comparison are filled, the robot updates its computed ranking of all sites $\mathbf{ranking}_n$ using the recorded pairwise comparison following the procedure *update_ranking* (Algorithm 3 line 6–7), which is shown in Algorithm 4.

Algorithm 4 Update Ranking

```
procedure update_ranking($n$, $D_n$)
    if $d_{n,1} \notin \mathbf{ranking}_n$ then
        if $d_{n,2} \notin \mathbf{ranking}_n$ then
            Randomly insert $d_{n,1}$; Randomly insert $d_{n,2}$ after $d_{n,1}$
        else
            Randomly insert $d_{n,1}$ before $d_{n,2}$
        end if
    else
        if $d_{n,2} \notin \mathbf{ranking}_n$ then
            Randomly insert $d_{n,2}$ after $d_{n,1}$
        else if index($d_{n,1}$) > $index(d_{n,2})$ then
            Switch $d_{n,1}$ and $d_{n,2}$ in $\mathbf{ranking}_n$
        end if
    end if
end procedure
```

In procedure *update_ranking*, the robot seeks to insert the two sites in its recorded pairwise comparison D_n into its computed ranking $\mathbf{ranking}_n$, while preserving the pairwise relationship (Algorithm 4 line 4,6,10). For example, inserting an element after that of $d_{n,1}$ is done by inserting an element in a random position marked by downward arrows.

$$\textit{Inserting after } d_{n,1} : \mathbf{ranking}_n = [s_a \; s_b \; s_c \; d_{n,1} \downarrow s_d \downarrow s_e \downarrow]$$

If both sites are present in $\mathbf{ranking}_n$, the robot checks if the rankings are complying with the pairwise relationship, and switches the rankings if not (Algorithm 4 line 12). An example of the switching operation is as follows.

$$\begin{aligned}\textit{Switching } d_{n,1}, d_{n,2} : \mathbf{ranking}_n &= [s_a \; s_b \; s_c \; d_{n,2} \; s_d \; d_{n,1} \; s_e] \\ \Rightarrow \mathbf{ranking}_n &= [s_a \; s_b \; s_c \; d_{n,1} \; s_d \; d_{n,2} \; s_e]\end{aligned}$$

The robot constantly broadcast its current computed $\mathbf{ranking}_n$ to its neighbors in its communication radius. If a site is not detected, it randomly picks up a message sent by its neighbor, if one is present, and it performs an election to generate a new $\mathbf{ranking}_n$ (Algorithm 3 line 9–11). We keep all interactions among the robots to a peer-to-peer and pairwise fashion similar to in the benchmark algorithm, such that the communication paradigms of the considered algorithms in this paper can be roughly similar. The differences between the message sizes of the considered algorithms depend on how the messages are encoded. For the benchmark belief fusion algorithm, the messages have $3^{K(K-1)/2}$ possible values, while for the proposed ranked voting algorithm, the messages have roughly $(K + 1)!$ possible values. In this paper $K = 8$, thus the possible message values are 3^{28} and 362,880, respectively. When represented in binary, they can be represented in a minimum of $45 bits$ and $19 bits$, respectively. However, this encoding method needs significant computational and storage resources to decode the messages during the operation of the algorithms. On the other hand, using the simplest method of encoding, where every value used is stored in a *short int* variable of $16 bits$. The messages' sizes would be $8K(K-1)\,bits$ and $16K\,bits$, respectively, and in this paper $448 bits$ and $128 bits$. Thus, compared to the benchmark algorithm, the proposed ranked voting algorithm not only has lower requirements on the communication bandwidth in the settings of this paper, the bandwidth also scales less rapidly when facing higher number of options.

Algorithm 5 Election

```
1:  function $election(\mathbf{ranking}_n, \mathbf{ranking}_m)$
2:      $\mathbf{candidates} = sort(\mathbf{ranking}_n \cup \mathbf{ranking}_m)$
3:      for $i = 1..size(\mathbf{candidates})$ do
4:          if $\mathbf{candidates}[i] \in \mathbf{ranking}_n$ then
5:              $\mathbf{score}_n[i] = search(\mathbf{ranking}_n, \mathbf{candidates}[i])$
6:          end if
7:          if $\mathbf{candidates}[i] \in \mathbf{ranking}_m$ then
8:              $\mathbf{score}_m[i] = search(\mathbf{ranking}_m, \mathbf{candidates}[i])$
9:          end if
10:     end for
11:     $\mathbf{score}_n[\mathbf{score}_n < 0] = size(\mathbf{ranking}_n)/2$
12:     $\mathbf{score}_m[\mathbf{score}_m < 0] = size(\mathbf{ranking}_m)/2$
13:     $\mathbf{result\_ranking} = \mathbf{candidates}[argsort(\mathbf{score}_n + \mathbf{score}_m)]$
14:     Return result_ranking
15: end function
```

An election in this context is held with only two voters, the robot *n* and its chosen neighbor *m*. The detailed process is shown in Algorithm 5. In the election process, the rankings need to be transformed into the scores of all considered sites, which are stored in $\mathbf{score}_n$ and $\mathbf{score}_m$ for the two voters, respectively. The transformation is done in Algorithm 5 line 3–10. The corresponding score of a considered site is the ranking of it in $\mathbf{ranking}_n$ or $\mathbf{ranking}_m$ (Algorithm 5 line 5,8). The two score vectors must then be padded to contain the same sites, which are tracked by the vector **candidates**. The unranked candidates' indices are selected using Boolean indexing in Algorithm 5 line 11–12. This is different from when ranked voting is utilized in real-life elections. This is because when a real-life ranked voting ballot has missing entries, it means that the unranked candidates have lower preferences than all ranked candidates and hence can be given the highest rankings. However, in our algorithm, an unranked site has an unknown quality relative to the ranked sites. Therefore, we assigned them a temporary ranking that is half of the number of ranked sites (Algorithm 5 line 11–12), such that the resulting ranking of unranked sites only considers the opinion of the other robot.

The following example illustrates the aforementioned operations.

$$\begin{aligned}
\mathbf{ranking}_n =& [3,1,5,6,2],\ \mathbf{ranking}_m = [5,1,2,7] \\
\mathbf{candidates} =& [1,2,3,5,6,7] \\
& \textit{Ranking} \Rightarrow \textit{Score}\ : \\
\mathbf{score}_n =& [1,4,0,2,3,-1],\ \mathbf{score}_m = [1,2,-1,0,-1,3] \\
& \textit{Pad Score Vectors}\ : \\
\mathbf{score}_n =& [1,4,0,2,3,2.5],\ \mathbf{score}_m = [1,2,2,0,2,3]
\end{aligned}$$

The padded score vectors are then added together, applied with an *argsort* operation and used as indices in an indexing operation of **candidates** vector to obtain the new ranking vector **result_ranking** (Algorithm 5 line 13). Here, we are performing the inverse of traditional Borda count (Emerson, 2013) and use the rankings directly as the associated points and sort the candidates in ascending order of their received points. The produced ranking is also randomized in the event of a tie anywhere in the ranking of received points.

Keeping up with the example above, the following is an example of how **result_ranking** is produced.

$$\begin{aligned} & \textit{Adding Score Vectors}: \\ \mathbf{score}_{total} &= [2, 6, 2, 2, 5, 5.5] \\ & \textit{Ranking of Considered Sites' Indices in } \mathbf{Candidates}: \\ argsort(\mathbf{score}_{total}) &= [2, 3, 0, 4, 5, 1] \\ & \textit{Resulting Ranking of Considered Sites}: \\ \mathbf{Candidates}[\mathbf{Indices}] &= [3, 5, 1, 6, 7, 2] \end{aligned}$$

Finally, the election results also need to be checked if they comply with the recorded pairwise comparison using *update_ranking* (Algorithm 3 line 12–13).

Overall, at the design level, the proposed algorithm uses less communication, storage, and computational resources compared to the benchmark algorithm based on belief fusion, especially the variant of it with the transitivity-preserving operation.

3.4 Evaluation metrics

In order to evaluate the performances of the two considered algorithms, we have to unify their outputs to the same format. The proposed ranked voting algorithm encodes the ranking in a vector with length of K, while the benchmark belief fusion algorithm records all pairwise relationships using a $K \times K$ matrix. Since the conversion from the latter to the former can result in information loss, we convert the rankings produced by the proposed ranked voting algorithm into a same-sized matrix containing all known pairwise relationships. The conversion is done using Algorithm 6.

Algorithm 6 Converting Ranking Vector into Belief Matrix

```
function $conversion(\mathbf{ranking}_n)$
    $B_n = zeros(K, K)$
    for $i = 1..K$ do
        for $j = 1..K$ do
            if $\mathbf{ranking}_n[i] < \mathbf{ranking}_n[j]$ then
                $B_n[i, j] = 1$
            else if $\mathbf{ranking}_n[i] > \mathbf{ranking}_n[j]$ then
                $B_n[i, j] = -1$
            end if
        end for
    end for
    Return $B_n$
end function
```

After unifying the outputs from the two considered algorithms, the output is compared to the belief matrix produced by the pairwise relationships of the true values of the sites B^*. The error is defined as follows:

$$Error = (\Sigma_{n=1..N_{robot}} sum(abs(B^* - B_n)))/N_{robot}$$

At initialization, all elements in B_n are set to 0; hence, the error at initialization is $Error_0 = K(K-1)$. In this paper $K = 8$, thus the error at initialization is 56. The maximum error that can theoretically be reached is $2K(K-1)$, where every pairwise relationship in the matrix is the opposite of the correct value. In our paper, this value is 112. The lowest error that can be achieved is 0, where the ranking of every robot is exactly correct.

We also pay attention to the level of scatter in the produced decisions within the swarm. We define the quantity *Scatter* as the average error between the belief matrices computed by every robot and those of every other robot, as follows:

$$Scatter = \frac{\Sigma_{n_1=1..N_{robot}} \Sigma_{n_2=1..N_{robot}} sum(abs(B_{n_1} - B_{n_2}))}{(N_{robot}-1)N_{robot}}$$

4 Experiments and results

In this section, we explain the experimental settings and results in detail. A swarm of N_{robot} robots is simulated in a $3\,\mathrm{m} \times 3\,\mathrm{m}$ $2D$ environment as shown in Fig. 1. The individual robots are programmed with the same low-level control mechanism to perform a random walk in the arena. A robot alternates between two modes of movement, walking forward in a straight line and rotating in place in a random direction. The two modes of movement have lengths that are randomly distributed, sampled from $exp(40)s$ and $unif(0, 4.5)s$, respectively. In order to avoid collisions, a robot moving forward will abort its current movement and start turning if another robot or the edge of the arena is detected in front of it. The robots here are simulated with the mechanical specification of e-puck robots (Mondada et al., 2009) and have a linear speed of 0.16 m/s and a rotational speed of 0.75 rad/s. The control loops are 1 s long where the aforementioned decision-making algorithms are executed.

The $K = 8$ sites in the experimental environment are in fixed positions of (0.5, 0.5), (1.5, 0.5), (2.5, 0.5), (0.5, 1.5), (2.5, 1.5), (0.5, 2.5), (1.5, 2.5), and (2.5, 2.5), all with radii of 0.3 m. Their qualities are chosen from the array $[0, 1, \dots, 7]$ randomly in every experimental instance. Noise $N(0, \sigma^2_{noise})$ is added to the true qualities of the sites to simulate different levels of inaccuracies in the cognitive abilities of the robots. We observe the performances of considered algorithms in different environments by changing the experimental parameters σ_{noise}, r_e, and N_{robot}. We gauge the performances via error and scatter at convergence, which is determined by the lowest error achieved during an experimental instance within a time limit of 2400 s. We compute the convergence time as the time step taken for the whole swarm to reach 90% of its peak performance, i.e., reach an error lower than $Error_{Conv} + (Error_0 - Error_{Conv}) * 0.1$.

4.1 Performances of ranked voting algorithm with respect to noise and evidence rates

The mean error and scatter at convergence, together with the mean convergence time, across 20 experiments at every parameter combination for the proposed ranked voting algorithm at various noise level σ_{noise} and evidence rate r_e settings are shown in Table 1.

Table 1 Performances of the proposed ranked voting algorithm at different noise levels σ_{noise} and evidence rates r_e; $N_{robot} = 30$

		Evidence Rate r_e						
	σ_noise	0.01	0.02	0.05	0.1	0.2	0.5	1
Mean Error at Convergence	0	1.227	0.040	0.187	0.000	0.000	0.007	0.307
	0.5	2.513	0.287	0.073	0.100	0.060	0.027	0.173
	1	4.453	2.093	0.973	1.480	1.173	1.280	1.087
	1.5	7.533	4.933	4.173	2.913	3.287	3.280	3.593
	2	8.053	9.287	5.860	5.820	5.187	4.940	6.127
	2.5	11.453	9.400	7.393	7.480	7.167	8.240	7.573
	3	15.453	11.873	10.040	10.380	9.273	10.213	11.127
Mean Scatter at Convergence	0	0.404	0.071	0.202	0.000	0.000	0.013	0.177
	0.5	1.155	0.428	0.096	0.179	0.105	0.048	0.270
	1	2.983	2.397	1.314	2.228	1.831	1.806	1.838
	1.5	4.862	4.844	4.538	3.857	4.925	4.592	5.070
	2	6.679	7.429	7.087	6.411	6.587	6.363	7.952
	2.5	8.090	9.447	8.782	9.229	8.371	10.116	9.596
	3	9.971	10.643	11.887	11.074	10.523	13.116	12.800
Mean Convergence Time (s)	0	1453.1	990.1	763.2	569.2	688.6	658.6	602.2
	0.5	1349.7	1173.2	741.6	687.4	631.1	622.1	710.7
	1	1400.8	1210.7	765.7	763.2	661.3	667.6	617.4
	1.5	1471.1	1114.0	816.0	728.0	697.2	745.9	740.4
	2	1638.2	1061.5	891.6	741.7	738.3	587.2	801.7
	2.5	1565.8	1324.9	1024.1	832.4	954.6	782.3	928.1
	3	1472.6	1135.8	1163.2	1176.3	997.7	863.9	840.8

It can be observed from the mean error and mean scatter results that the noise level has a significant impact on the accuracy and precision performances of the proposed algorithm. As the noise level σ_{noise} increases, there is a very clear increase in both mean error and mean scatter at convergence. However, for most noise level and evidence rate combinations, the mean scatter is consistently higher than the mean error at convergence. This shows an accurate but imprecise decision distribution from the proposed algorithm. At low r_e values below 0.02 and at especially high noise levels, the relationship above can be reversed, and the error could be higher than the scatter at convergence. This is to be expected as at these r_e values, the robots get very few observations. Coupled with a high noise level, erroneous pairwise observations tend not to be challenged, leading to inaccurate results.

At a particular noise level, the lowest mean errors and mean scatters are quite likely to be found on the middle range of evidence rates from 0.05 to 0.5, while both too low and too high an evidence rate can negatively affect the decision-making accuracy. Due to the stochasticity in the proposed algorithm's decision-making process, especially the random inserting of observed pairwise relationships in Algorithm 4, the proposed algorithm needs a certain number of pairwise opinion combination relative to the evidence input to enforce a consensus, which is harder to meet when the evidence rate is too high.

On the other hand, the mean convergence time is more affected by the evidence rate r_e than by the noise level. When r_e increases from 0.01 to 0.1, there is a very apparent drop in mean convergence time at every noise level. However, beyond an evidence rate of 0.1, the change in mean convergence time is more irregular. This, combined with evidence rate's effects on errors and scatters at convergence, shows that for the proposed algorithm, a lack of evidence can hamper the decision-making process, but too high an influx of evidence does not necessarily have a positive effect.

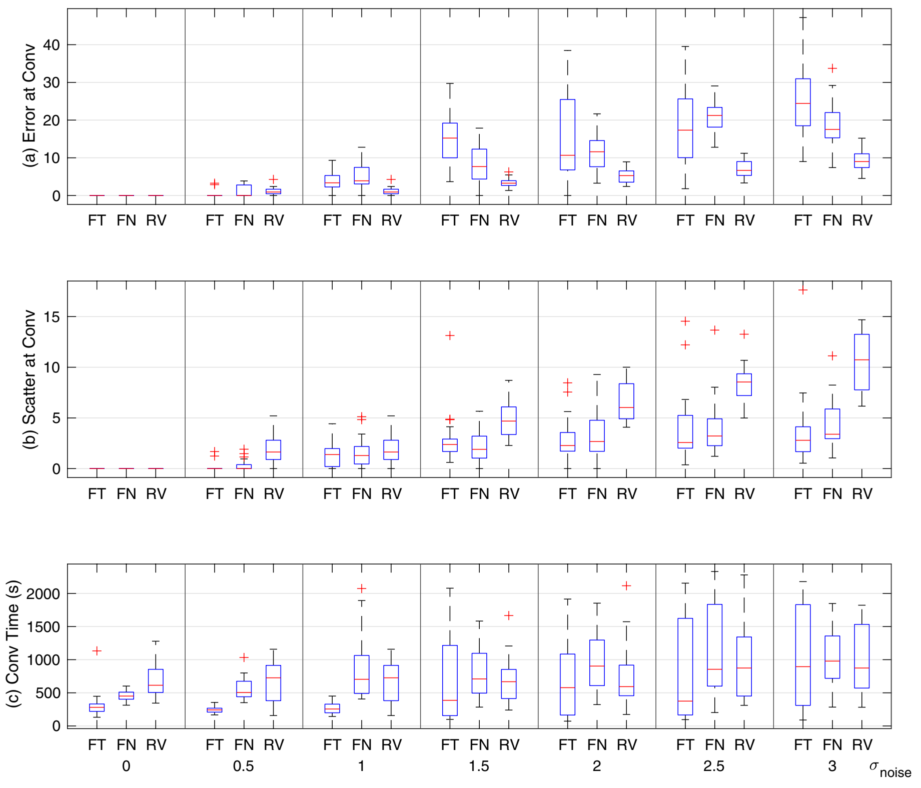


Fig. 2 Box plots of **a** error at convergence, **b** scatter at convergence, **c** convergence time (s) for the proposed ranked voting algorithm (RV) and both variants of the benchmark algorithm (FT: fusion with transitivity preserved, FN: fusion without transitivity preserved) at different σ_{noise} settings; $r_e = 0.2$, $N_{robot} = 30$ (Color figure online)

4.1.1 Comparison with the belief fusion benchmark at different noise levels

The performance distribution across 20 experimental runs of considered algorithms under different noise levels is shown in Fig. 2. The evidence rate r_e is set to 0.2. The swarm size N_{robot} is set to 30. We have also performed linear regression of the mean performances across all experimental runs at individual parameter settings against noise level and computed the gradient of the best-fitting linear function and the coefficient of determination (R^2), the latter of which measures the level of linear relationship observed in the data. The results are shown in Table 2. In Fig. 2a, b, we see that all the three algorithms produce comparable errors and scatters at convergence when the noise is low at 0 or 0.5. As shown

Table 2 Gradient and R^2 values obtained from linear regression of mean performances against σ_{noise}

Algo	LR of Error	LR of Scatter	LR of Conv Time
BF w Tr	$G = 9.4\ R^2 = 0.918$	$G = 1.48\ R^2 = 0.916$	$G = 276\ R^2 = 0.881$
BF w/o Tr	$G = 7.39\ R^2 = 0.936$	$G = 1.58\ R^2 = 0.979$	$G = 209\ R^2 = 0.839$
RV	$G = 3.77\ R^2 = 0.994$	$G = 3.78\ R^2 = 0.975$	$G = 118\ R^2 = 0.75$

in Fig. 2c, both variants of belief fusion are also able to converge within a shorter time compared to the proposed ranked voting algorithm. Among them, belief fusion with transitivity-preserving operations is the fastest.

However, when the noise increases, the advantages of both belief-fusion-based algorithms begin to diminish. As shown in Fig. 2a, when noise level σ_{noise} is in the range between 1 and 3, the error at convergence increases significantly for both variants of belief fusion, as the median error increases from around 0 to 24.4 for belief fusion with transitivity-preserving operation, and 17.5 without. The reduction in accuracy in the face of noise is also observed in the proposed ranked voting algorithm; however, the increase in median error at convergence here is much milder and the median value only hit 9 at the highest experimented noise level of 3. This is substantiated by the statistical analysis in Table 2, where the proposed ranked voting algorithm obtains the lowest gradient of mean error with respect to σ_{noise} at 3.77.

On the other hand, as observed in Fig. 2b, the proposed ranked voting algorithm produces a progressively higher scatter than the two variants of belief fusion as the noise level increases, reaching a median value of 10.7. As noted in the previous subsection, the scatter produced by the proposed ranked voting algorithm is consistently on roughly the same scale as the error. However, both variants of belief fusion, although experiencing a significant increase in error, only have a mild increase in scatter, to a median of 2.79 when transitivity is preserved and 3.39 when it is not, as noise increases. This is also shown in Table 2, where the proposed ranked voting algorithm obtains the highest gradient of mean scatter with respect to σ_{noise} at 3.78.

From the aforementioned experimental data, we can conclude that as noise increases, the proposed ranked voting algorithm experiences a drop in precision, producing a higher scatter as the noise increases. Although the error also increases, it is consistently on the same scale or smaller than the scatter, confirming the fact that the proposed ranked voting algorithm keeps a high accuracy and much of the increasing error can be ascribed to scatter. In contrast, both variants of belief fusion experience a smaller increase in scatter, but they experience a much larger increase in error compared to the proposed ranked voting algorithm, demonstrating the fact that belief fusion can lead to consistent consensus among the swarm but is unable to reliably obtain the correct ranking at high-noise scenarios.

As shown in Fig. 2c, the convergence time for both variants of belief fusion experiences in general can increase as the noise level increases. Its variance also rises for both algorithms. At higher levels of noise from 2 to 3, the convergence time of all the three algorithms is roughly on the same level and the advantage in fast convergence of belief fusion does not hold anymore. As shown in Table 2, the linear relationships between convergence time and noise level are not as strong as for the previous two performance metrics, shown by lower R^2 values. However, the proposed ranked voting algorithm still obtains the lowest gradient at 118.

Taking an integrated look at the performances of the considered algorithms with respect to the noise level, the differences in their performances can be explained by looking at their decision-making mechanisms. Both variants of belief fusion use a deterministic fusion function that encodes every pairwise relationship, making it easy for the whole swarm to converge their individual beliefs. However, it is also vulnerable to being misled by erroneous information at high-noise scenarios. On the other hand, the proposed ranked voting algorithm limits the number of decision variables faced by the individual robots by using a more compact way of encoding the decisions. Its method of opinion combination also introduces a degree of stochasticity into the decision-making process, hence allowing the swarm to correct itself from wrong ordering easily, albeit at a cost of reducing the precision of the decisions made.

To better illustrate the differences in the decision-making mechanisms of the considered algorithms, Fig. 3 shows two toy examples of the benchmark belief fusion algorithm when there are only one pairwise relationship and three robots considered, and also in the absence of evidence input. The three robots are assumed to be within communication distance of each other. Every robot randomly receives a belief message from a random neighbor and performs its decision-making process. The top rows in both subfigures show the initial state in the locality, and the bottom rows show the possible states in the next time step. It can be seen in Fig. 3a for belief fusion that all three possible transitions eliminate the minority opinion -1, and the first two transitions will result in all three robots picking the opinion $+1$ in the following time steps. In contrast, in Fig. 3b for ranked voting it can be observed that only the first and last outcome with a combined probability of 5/16 result in loss of information. In addition, no robots are left with the unknown status of 0 and the spread of a particular single opinion is significantly slowed.

4.1.2 Comparison with the belief fusion benchmark at different evidence rates

We then compare the impact on the operations of the considered algorithms from evidence rate r_e. The performance distribution at different r_e values is plotted in Fig. 4. The noise level σ_{noise} is set to 1.5, and the swarm size N_{robot} is set to 30. The results from linear regression of the mean performances against the natural log of the evidence rate $ln(r_e)$ are shown in Table 3.

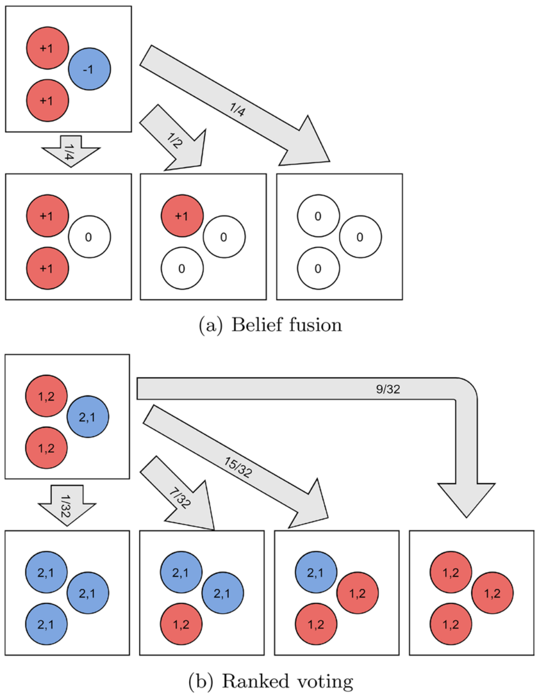


Fig. 3 Toy examples of the state transition only considering 1 pairwise comparison within a small locality of three robots of **a** the benchmark belief fusion algorithm and **b** the proposed ranked voting algorithm; circles represent robots, numbers, and color codings represent robot opinions, arrows & fractions represent possible next states and transition rates (Color figure online)

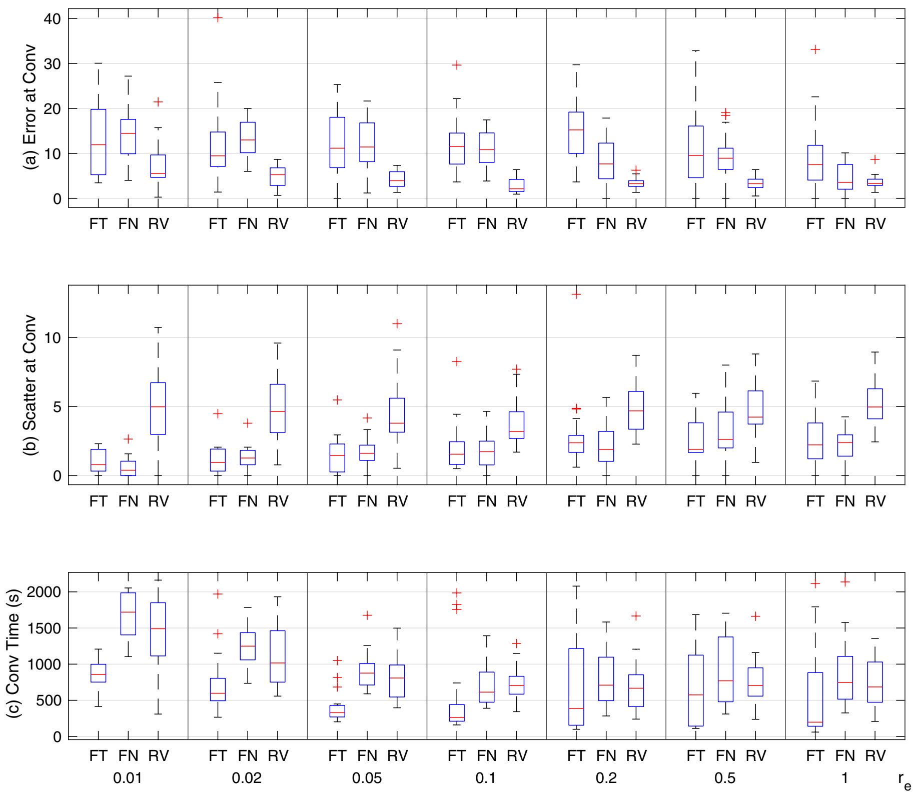


Fig. 4 Box plots of **a** error at convergence **b** scatter at convergence **c** convergence time (s) for the proposed ranked voting algorithm (RV) and both variants of the benchmark algorithm (FT: fusion with transitivity preserved, FN: fusion without transitivity preserved) at different r_e settings; $\sigma_{noise} = 1.5$, $N_{robot} = 30$ (Color figure online)

Table 3 Gradient and R^2 values obtained from linear regression of mean performances against $ln(r_e)$

Algo	LR of Error	LR of Scatter	LR of Conv Time
BF w Tr	$G = -0.42\ R^2 = 0.146$	$G = 0.386\ R^2 = 0.791$	$G = -34.2\ R^2 = 0.144$
BF w/o Tr	$G = -1.91\ R^2 = 0.901$	$G = 0.409\ R^2 = 0.738$	$G = -144\ R^2 = 0.522$
RV	$G = -0.333\ R^2 = 0.429$	$G = 0.0204\ R^2 = 0.007$	$G = -141\ R^2 = 0.665$

From Fig. 4a, we can see that all considered algorithms experience a general reduction in error when the evidence rate increases. The reduction is the least apparent in belief fusion with transitivity-preserving operations. For the proposed ranked voting algorithm, there is also a significant drop in the variance of the error at convergence. This is also substantiated by the statistical analysis shown in Table 3, where belief fusion with transitivity preserved obtains a very weak linear relationship between mean error and $ln(r_e)$ with $R^2 = 0.146$, as well as between mean convergence time and $ln(r_e)$ with $R^2 = 0.144$. Figure 4b shows that both variants of belief fusion see higher scatter in their results as the evidence rate increases. There is also more variance in the scatter observed. However, this feature is not observed in the proposed ranked voting algorithm. Instead, the median scatter decreases when the evidence rate increases from 0.01 to 0.1 and starts increasing beyond that. There is also an observable increase in the variance of the scatter when evidence rate reduces beyond 0.1. As shown in Table 3, both variants of belief fusion obtain moderately strong linear relationships between mean scatter and $ln(r_e)$ with R^2 being 0.791 and 0.738, respectively. On the other hand, for the proposed ranked voting algorithm, mean scatter is largely independent of evidence rates with $G = 0.0204$ and $R^2 = 0.007$.

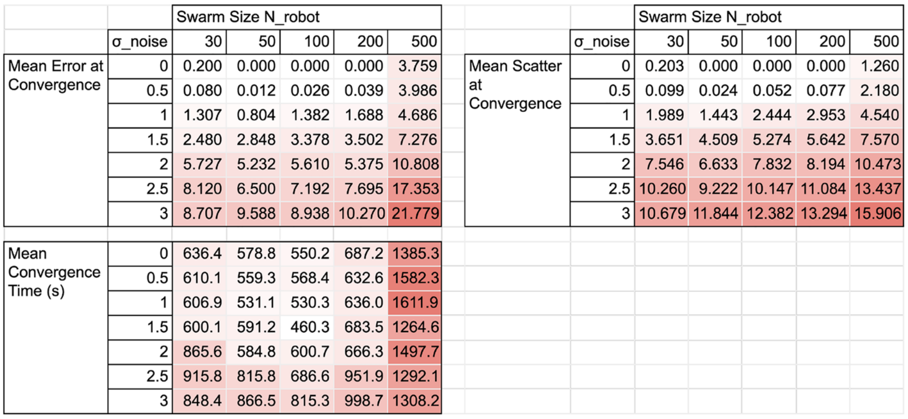
In terms of convergence time, all considered algorithms experience a significant increase in decision speed when the evidence rate increases from 0.01 to 0.1. Beyond 0.1, the median convergence time either experiences a slight increase as in the case of the two variants of belief fusion, or does not see much change as in the case of the proposed ranked voting algorithm. At the same time, both variants of belief fusion experience an increase in the variance of the convergence time at high evidence rate. The same holds true for the proposed ranked voting algorithm when comparing to the variance at $r_e = 0.1$.

Overall, the performances of the proposed ranked voting approach generally improve as the evidence rate increases, with a reducing error and convergence time. It is also more resistant to the effects of low evidence rates in terms of error compared to both variants of belief fusion. Its convergence time also increases at a slower rate than belief fusion without transitivity preserved when the evidence rate reduces, while being more vulnerable in this aspect compared to belief fusion with transitivity preserved. For the belief fusion benchmark, both variants see reducing error when the evidence rate increases, but both also see increasing scatter and a much higher uncertainty in convergence time as evidence rate increases beyond 0.2.

4.2 Performances of ranked voting algorithm with respect to swarm sizes

Afterward, we examine the impact of swarm sizes N_{robot} on the performances of the proposed ranked voting algorithm. The mean performances across 20 experimental runs at every parameter combination are shown in Table 4. It can be observed that for all three metrics, optimal behaviors are more likely to be observed at medium ranges of swarm sizes

Table 4 Performances of proposed ranked voting algorithm at different noise levels σ_{noise} and swarm sizes N_{robot}; $r_e = 0.2$



		Swarm Size N_robot				
	σ_noise	30	50	100	200	500
Mean Error at Convergence	0	0.200	0.000	0.000	0.000	3.759
	0.5	0.080	0.012	0.026	0.039	3.986
	1	1.307	0.804	1.382	1.688	4.686
	1.5	2.480	2.848	3.378	3.502	7.276
	2	5.727	5.232	5.610	5.375	10.808
	2.5	8.120	6.500	7.192	7.695	17.353
	3	8.707	9.588	8.938	10.270	21.779
Mean Convergence Time (s)	0	636.4	578.8	550.2	687.2	1385.3
	0.5	610.1	559.3	568.4	632.6	1582.3
	1	606.9	531.1	530.3	636.0	1611.9
	1.5	600.1	591.2	460.3	683.5	1264.6
	2	865.6	584.8	600.7	666.3	1497.7
	2.5	915.8	815.8	686.6	951.9	1292.1
	3	848.4	866.5	815.3	998.7	1308.2

		Swarm Size N_robot				
	σ_noise	30	50	100	200	500
Mean Scatter at Convergence	0	0.203	0.000	0.000	0.000	1.260
	0.5	0.099	0.024	0.052	0.077	2.180
	1	1.989	1.443	2.444	2.953	4.540
	1.5	3.651	4.509	5.274	5.642	7.570
	2	7.546	6.633	7.832	8.194	10.473
	2.5	10.260	9.222	10.147	11.084	13.437
	3	10.679	11.844	12.382	13.294	15.906

of 50 and 100, while the performances at extreme swarm sizes are often worse off. This is similar to the effects produced by varying the evidence rate r_e. However, there is a more clear worsening of all considered metrics at higher swarm sizes compared to evidence rates. This is to be expected as a higher swarm size not only introduces more evidence but also introduces more agents that need to be brought into convergence for a consensus to form.

4.2.1 Comparison with the belief fusion benchmark at different swarm sizes

We now compare the impact from swarm size N_{robot} on the performances of the considered algorithms, as shown in Fig. 5. The noise level σ_{noise} is set to 1.5, and the evidence rate r_e is set to 0.2. The results from linear regression of the mean performance against the natural log of the swarm size $ln(N_{robot})$ are shown in Table 5.

It can be observed that all considered algorithms experience an increase in error when the swarm size increases. This is substantiated by the statistical analysis in Table 5, where the proposed ranked voting algorithm obtains the lowest gradient of mean error against $ln(N_{robot})$ at 1.53. Both variants of belief fusion also see a general reduction in scatter as the

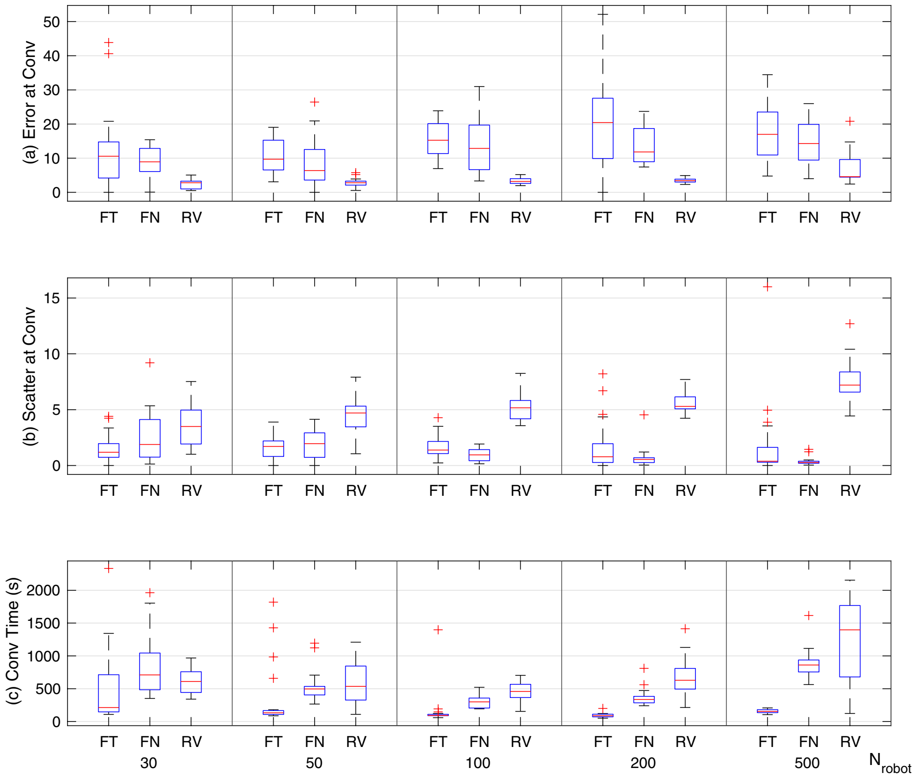


Fig. 5 Box plots of **a** error at convergence **b** scatter at convergence **c** convergence time (s) for the proposed ranked voting algorithm (RV) and both variants of the benchmark algorithm (FT: fusion with transitivity preserved, FN: fusion without transitivity preserved) at different N_{robot} settings; $\sigma_{noise} = 1.5$, $r_e = 0.2$ (Color figure online)

Table 5 Gradient and R^2 values obtained from linear regression of mean performances against $ln(N_{robot})$

Algo	LR of Error	LR of Scatter	LR of Conv Time
BF w Tr	$G = 2.74\ R^2 = 0.53$	$G = -0.138\ R^2 = 0.247$	$G = -137\ R^2 = 0.689$
BF w/o Tr	$G = 2.37\ R^2 = 0.782$	$G = -0.777\ R^2 = 0.892$	$G = 5.38\ R^2 = 0.001$
RV	$G = 1.53\ R^2 = 0.777$	$G = 1.29\ R^2 = 0.961$	$G = 218\ R^2 = 0.598$

swarm size increases, while for the proposed ranked voting algorithm, there is still a clear linear relationship between scatter and swarm size. It is thus shown that as the number of agents increases, both variants of belief fusion see a stronger push toward consensus, which produces lower scatter but higher error. For belief fusion with transitivity preserved, this also translates to a lower convergence time, with a gradient of −137. The same effects are not observed in the proposed ranked voting algorithm, which sees its error scales much slower to swarm size. However, this comes at the cost of a higher and scaling convergence time with a gradient of 218. The proposed ranked voting algorithm also uses less communication bandwidth, storage, and processing power compared to the benchmarks, which makes it viable in large swarm sizes.

5 Discussion

Based on our experiments, we can characterize the performances of the proposed ranked voting algorithm as being, in general, slower and less precise, but more accurate and cheaper than the benchmark belief fusion algorithms. There is especially a clear advantage of the ranked voting at high noise and high swarm size scenarios.

The differences in their performances are due to the different decision-making mechanisms used. The proposed ranked voting algorithm uses a more compact encoding method to represent the ranking among the sites. It is able to give a compromising result when agents of different opinions are combining their opinions, while in contrast, the benchmark belief fusion algorithms revert all entries in conflict back to the initial unknown status of value 0, resulting in information loss. This feature, combined with the mechanism in belief fusion operation to always assign any available +1 or −1 entry values to entries with the unknown status, results in a positive feedback loop within the swarm. Thus, swarms using the belief fusion algorithm can come to a consensus rapidly, but when most of the belief matrices are filled, it is very hard for dissenting agents to spread their opinions, even if they hold correct pairwise information.

Most of the classical opinion-based collective decision-making strategies have been built on similar positive feedback mechanisms, such as in Valentini et al. (2015), Valentini et al. (2016), and Ebert et al. (2020). In these decision-making strategies, the adoption of a particular opinion by an agent increases the probability of the same opinion being adopted by other agents. Such positive feedback has also been replicated using a probability fusion algorithm in Shan and Mostaghim (2021). However, in these problems, the number of possible options is small. Thus, it is possible to accurately track every single potential option and use positive feedback to create fast consensus.

In contrast, in the collective preference learning problem among eight sites investigated in this paper, there are $8! = 40320$ possible results. Therefore, the two algorithms considered in this paper also do not seek to accurately track all possible options, rather they both try to approach the collective preference learning problem as an optimization problem and the individual agents seek to make incremental changes in the form of single pairwise relationships to approach the true preference order. In such an approach, the existence of a positive feedback loop in the decision-making process can cause the swarm to be stuck on a local optimum, where a few agents have more accurate ranking information, but could not overpower the established consensus, leading to premature convergence. The impact of such premature convergence on the accuracy of the consensus depends on two factors, the level of dynamism in the environment, and the level of sensory capabilities of individual agents. In a dynamic environment, the established consensus can potentially prevent the swarm from responding to changes in the environment. On the other hand, this could also negatively impact the accuracy in a static environment when the individual agents have poor sensory capabilities, in terms of the environment being noisy or observations being hard to collect, due to establishing a consensus before the agents can make enough observations. This is substantiated by the performances of considered algorithms at high noise levels and low evidence rates, respectively.

On the other hand, the proposed ranked voting algorithm employs a degree of stochasticity in its election process. The ordering among options with tied points in the election result is random. Since there are only two voters, ties are fairly common. This leads to a higher scatter in the final result, as conflicting information needs many pairwise robot interactions to be eliminated. However, it also means that dissenting opinions have an opportunity to spread within the swarm. The whole swarm can thus readily shift in opinions and has a much better chance in approaching the true result. It is also less likely for a pairwise robot interaction to result in loss of information, and the swarm can thus avoid being dominated by a single opinion.

6 Conclusion

In this paper, we investigate a collective preference learning scenario that can potentially be faced by an autonomous robot swarm. The swarm is tasked with ranking a series of potential sites in the order of preference. We have proposed a ranked voting algorithm with Borda count tallying to enable the simulated swarm to perform the designated task. We have then tested the viability of the proposed approach in collective preference learning scenarios with different noise levels, evidence rates, and swarm sizes. We have compared the performances of our proposed approach against those of two variants of a belief fusion-based benchmark algorithm, in terms of accuracy, precision, and speed.

On the design level, our proposed ranked voting algorithm is cheaper in memory usage, processing power required, and communication bandwidth needed. However, it can outperform the benchmarks in terms of decision accuracy and, in some cases, convergence speed, especially in high-noise and high swarm size situations. Its downsides include a higher scatter of the swarm's results at convergence and longer convergence time in low-noise situations.

In future works, we aim to implement the proposed ranked voting decision-making strategy in dynamic environments as well as on real robotic systems and investigate its performances. We also plan to further improve the ranked voting strategy so that it can achieve stronger convergence and also deal with cases where the robots are only interested in the rankings of the higher-quality subset of the available sites. In addition, we aim to integrate path planning and active searching by individual agents into the algorithm to further improve the performance.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Bartashevich, P., & Mostaghim, S. (2021). Multi-featured collective perception with evidence theory: Tackling spatial correlations. *Swarm Intelligence, 15,* 83–110. https://doi.org/10.1007/s11721-021-00192-8.

Brambilla, M., Ferrante, E., Birattari, M., et al. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence, 7,* 1–41. https://doi.org/10.1007/s11721-012-0075-2.

Brill M, Elkind E, & Endriss U, et al (2016) Pairwise diffusion of preference rankings in social networks. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 130–136).

Camazine, S., Deneubourg, J. L., Franks, N. R., et al. (2020). *Self-organization in biological systems*. Princeton University Press.

Crosscombe, M., & Lawry, J. (2021). Collective preference learning in the best-of-n problem. *Swarm Intelligence, 15,* 145–170. https://doi.org/10.1007/s11721-021-00191-9.

Dorigo, M., Theraulaz, G., & Trianni, V. (2021). Swarm robotics: Past, present, and future [point of view]. *Proceedings of the IEEE, 109*(7), 1152–1165. https://doi.org/10.1109/JPROC.2021.3072740.

Ebert JT, Gauci M, & Mallmann-Trenn F, et al (2020) Bayes bots: Collective bayesian decision-making in decentralized robot swarms. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 7186–7192). https://doi.org/10.1109/ICRA40945.2020.9196584.

Ebert JT, Gauci M, & Nagpal R (2018) Multi-feature collective decision making in robot swarms. In *Proceedings of the 17th international conference on autonomous agents and multiagent systems. International foundation for autonomous agents and multiagent systems, Richland, SC, AAMAS'18* (pp. 1711–1719).

Emerson, P. (2013). The original Borda count and partial voting. *Social Choice and Welfare, 40*(2), 353–358. https://doi.org/10.1007/s00355-011-0603-9.

Garnier, S., Gautrais, J., & Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intelligence, 1,* 3–31. https://doi.org/10.1007/s11721-007-0004-y.

Guha A, & Dasgupta A (2021) Consensus from distributed iterative voting. In *2021 IEEE 11th annual computing and communication workshop and conference (CCWC)* (pp. 0829–0833). https://doi.org/10.1109/CCWC51732.2021.9376080.

Hassanzadeh FF, Yaakobi E, & Touri B, et al (2013) Building consensus via iterative voting. In *2013 IEEE international symposium on information theory* (pp. 1082–1086). https://doi.org/10.1109/ISIT.2013.6620393.

Lee, C., Lawry, J., & Winfield, A., et al. (2018). Negative updating combined with opinion pooling in the best-of-n problem in swarm robotics. In M. Dorigo, M. Birattari, & C. Blum (Eds.), *Swarm intelligence* (pp. 97–108). Springer. https://doi.org/10.1007/978-3-030-00533-7_8.

Mondada F, Bonani M, & Raemy X, et al (2009) The e-puck, a robot designed for education in engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, 1*(1), 59–65. http://infoscience.epfl.ch/record/135236

Parker, C. A. C., & Zhang, H. (2009). Cooperative decision-making in decentralized multiple-robot systems: The best-of-n problem. *IEEE/ASME Transactions on Mechatronics, 14*(2), 240–251. https://doi.org/10.1109/TMECH.2009.2014370.

Parker, C. A. C., & Zhang, H. (2011). Biologically inspired collective comparisons by robotic swarms. *The International Journal of Robotics Research, 30*(5), 524–535. https://doi.org/10.1177/0278364910397621.

Shan Q, Heck A, & Mostaghim S (2021) Discrete collective estimation in swarm robotics with ranked voting systems. In *2021 IEEE symposium series on computational intelligence (SSCI)* (pp. 1–8). https://doi.org/10.1109/SSCI50451.2021.9659868

Shan, Q., & Mostaghim, S. (2021). Discrete collective estimation in swarm robotics with distributed Bayesian belief sharing. *Swarm Intelligence, 15,* 377–402. https://doi.org/10.1007/s11721-021-00201-w.

Strobel V, & Dorigo M (2018) Blockchain technology for robot swarms: A shared knowledge and reputation management system for collective estimation. In *Swarm intelligence: 11th international conference, ANTS 2018, Rome, Italy, October 29–31, 2018. Proceedings* (p. 425). Springer.

Talamali MS, Marshall JAR, & Bose T, et al (2019) Improving collective decision accuracy via time-varying cross-inhibition. In *2019 international conference on robotics and automation (ICRA)* (pp. 9652–9659). https://doi.org/10.1109/ICRA.2019.8794284

Tideman, N. (2017). Collective decisions and voting: The potential for public choice. *Routledge*. https://doi.org/10.4324/9781315259963.

Valentini, G., Hamann, H., & Dorigo, M. (2015). Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In *Proceedings of the 2015 international conference on autonomous agents and multiagent systems. International foundation for autonomous agents and multiagent systems, Richland, SC, AAMAS'15* (pp. 1305-1314).

Valentini, G., Ferrante, E., & Dorigo, M. (2017). The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives. *Frontiers in Robotics and AI, 4,* 9. https://doi.org/10.3389/frobt.2017.00009.

Valentini, G., Ferrante, E., Hamann, H., et al. (2016). Collective decision with 100 kilobots: Speed versus accuracy in binary discrimination problems. *Autonomous Agents and Multi-Agent Systems, 30*(3), 553–580. https://doi.org/10.1007/s10458-015-9323-3.