Collective Decision Making with Heterogeneous Agents

Syam Ajay Simha Gullipalli
Universität Paderborn
syam@mail.uni-paderborn.de

April 26, 2015

1 Introduction

Choosing an option from a choice of N alternatives based on quality metrics is easy if every alternative is processed. The agents used to evaluate the quality metrics should have immense processing capabilities, because the time and computational power it takes to choose the best increases with the number of alternatives which is known as *best-of-n* decision problem. Multiple distributed agents with simple capabilites may dominate a single central agent with enormous capabilities. Natural computing, an emerging technology which takes inspiration from nature, has its own advantages in solving complex problems. This paper focuses on swarm robotics, a part of natural computing which uses relatively simple, distributed and self-organized robots (agents) to accomplish complex tasks. The biggest advantage of distributed agents over a central agent is, the distributed agents are not prone to single point of failure.

This paper presents a solution to tackle a problem when the agents are heterogeneous. There are four rooms r_i where $i \in [0, 3]$ connected to the edges of a central room. The agents can differentiate the room by finding a color at the entrance of the romm. The central room has an entrance to each room. Each room has its quality metrics; ground color (v_G), light intrensity (v_L), and objects (v_O). The task is to choose the room with high average value of these three metrics. The heterogeneous robots used here are of two types; Robots with ground sensor to evaluate ground color (type G), and robots with light sensor to evaluate the light intensity (type L). Robots of both type G and L have omnidirectional camera to detect objects around it. Each metric is a value between 0 and 1. Each room can have a minimum of 2 and maximum of 12 objects. The objects evaluation metric (v_O) using the number of objects (n_O) is calculated using equation (1) to have a value between 0 and 1.

$$v_O = 0.1(n_O) - 0.2 \tag{1}$$

2 Solution from Honeybee Swarms

Exchanging information among the agents in a swarm during collective decision making is an important task. An agent assess the quality of information upon new information to take individual decision. The decisions of individual agents emerges to a global behavior of every agent to agree on the same decision. Since, the communication capabilities of swarm robots are limited, it is impossible to exchange the complete knowledge of every single metric among all robots. The decision making is difficult when the robots are heterogeneous, because each agent can percieve a limited knowledge. Due to limited communication and no complete knowledge, the robots have to depend on simple information like opinion.

The solution implemented follows the ideas of "The Weighted Voter Model" presented in literature [1]. It is a hybrid method of combining house hunting behavior of honeybees and opinion dynamics. During house hunting, honeybees individually *survey* and collect information about the available *sites*. When a honeybee chooses a site, it comes back to the *nest* and tries to influence the neighbors to survey the site by *waggle dances*. The waggle dance encodes the information about available metrics like distance, direction, quality etc. This house hunting behavior is applied on classic voter model in which a random opinion is adopted by a random agent in each step. When waggle dance ends, the honeybee adopts an *opinion* of random neighbor and surveys the site. This procedure eventually makes all the honeybees to have the same opinion of site. The positive feedback here is the duration of the waggle dance which is directly proportional to the quality of the opinion. Therefore, if the qulatiy of site is high, the agent invests larger time on waggle dance to influence more number of agents to have the same opinion.

3 State Machine

For the convenience of comparing the configuraion of current task to honeybees behavior, the central room is considered as nest and the rooms connected to it's edges are considered as sites. The transitions of different states for the solution construction is shown in figure 1. Initially, robot has no opinion, so it is set to INIT to have an opinion. AVOID state is used to avoid collisions. SURVEY state sets the behavior of robot to go to the site of opinion and evaluate metrics perceived. WAGGLE state sets the behavior of robot to get into the nest and broadcast it's opinion. All possible transitions of these states are:

3.1 Any state to AVOID

Collision avoidance mechanism is a high priority task to avoid physical damages to robots. The robot transforms to AVOID state from any state when a collision is detected.

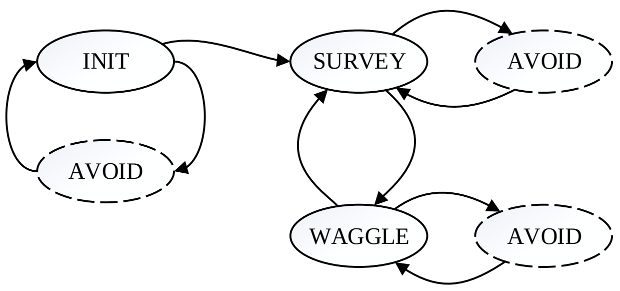


Fig. 1: State machine for collective decision making using house hunting behavior of honeybees.

3.2 AVOID to previous state

When a collision is avoided, the robot transforms to the previous state to perform desired task. The robot has to preserve the current state before transforming to AVOID to resume the state again after collision avoidance.

3.3 INIT to SURVEY

Initially, a robot has no opinion which is initialized to INIT state. Upon having an opinion, the robot transforms to SURVEY state to percieve the metrics of the opinion. This transition is performed only once during exection.

3.4 SURVEY to WAGGLE

Upon time out of the SURVEY state, robot transforms to WAGGLE state to broadcast opinion and influence it's neighbors.

3.5 WAGGLE to SURVEY

Upon time out of the WAGGLE state, robot adopts opinion from a random neighbor and enters SURVEY state to perceive the quality of site. When ever a robot transforms from WAGGLE to SURVEY state, it considers the problem as *binary decision problem* instead of *best-of-n* decision problem, because it can choose only between two opinions, the opinion it has already and the opinion of a (random) single nighbor.

4 Implementation

The behavior of the robot is changed by set and reset of the basic flags `is_room_sensed`, `is_at_entrance`, and `is_entered_room` flags as in flowchart shown in figure 2.

This flowchart can be easily transformed into a basic structure of algorithm depicted in 1. The flags are set and reset within the function definitions to change the behavior of robots. For simplicity, the collision avoidance mechanism is considered as an internal mechanism of robots, therefore not shown in the flowchart or algorithm, although it is designed as a separate state.

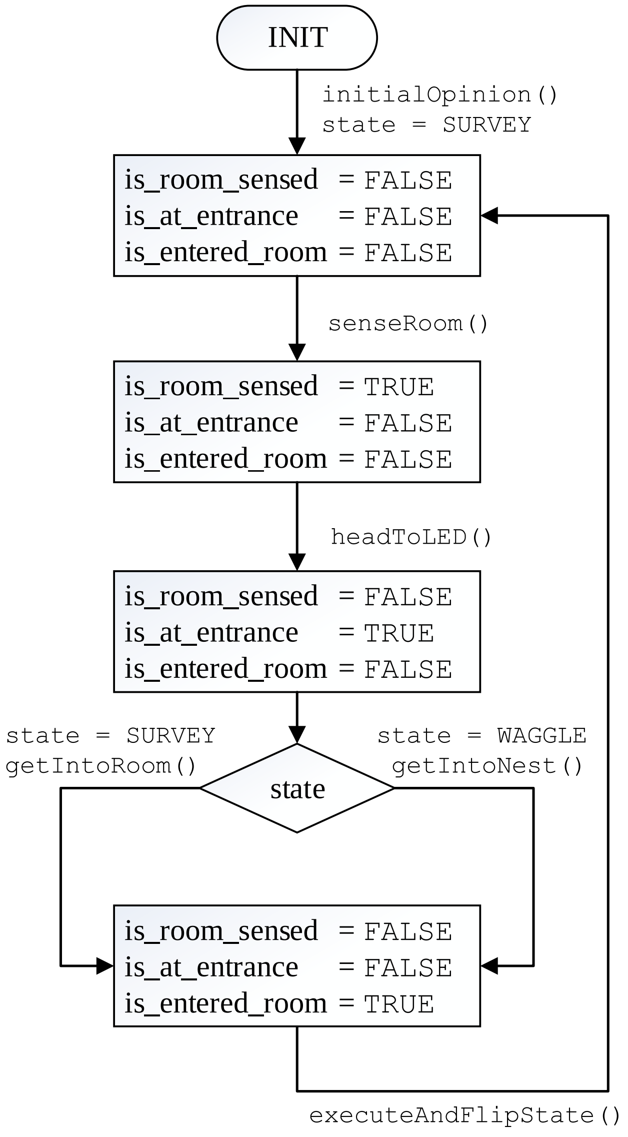


Fig. 2: Flowchart showing transitions of important flags to change state and behavior of a robot.

Algorithm 1: Transformed pseudocode from the flochart in image 2.

```
state = INIT;                                          /* Initial state */
initialOpinion();                               /* Have an initial opinion */
state = SURVEY;  /* Change to SURVEY state when robot has an opinion */
if is_room_sensed then /* Recognized the room of opinion              */
    headToLED();                              /* Go towards the room sensed */
else if is_at_entrance then /* Close to the room of opinion          */
    if state == SURVEY then /* SURVEY state                            */
        getIntoRoom();                               /* Enter into the room */
    else /* WAGGLE state                                                 */
        getIntoNest();                  /* Exit the room to enter the nest */
else if is_entered_room then /* In the desired room                    */
    executeAndFlipState();   /* Perform activities of current state and
    flip the state on timeout */
else /* Room not sensed                                                  */
    senseRoom();          /* Search for the room matching the opinion */
```

The basic functions of the robt are:

`detectCollision()`

Detect collisions ahead. If a collision is detected, preserve the current state before changing to AVOID state.

`avoidCollision()`

Avoid collision if a collision is detected by `detectCollision()`. Resume the previous state when collision is avoided.

`initialOpinion()`

Get an initial opinion of which site to survey. For the experiment results presented, each room has assigned equal (or nearly equal if not possible) number of of robots.

`senseRoom()`

Find the site of the robot's opinion by detecting the LED color placed at the entrance of the site to change the heading. Set the flag `is_room_sensed` when detected the site to perform the next action.

`headToLED()`

Go near the LED placed at the entrance of the site to make sure that the robot is moving into desired site or nest. When the robot is close enough to the desired LED, set `is_at_entrance`.

`getIntoRoom()`

Get into the site (room) if the robot is close enough to the LED. This function is executed only when the robot is in SURVEY state. The robot finds the objects in the site and tries to go near the object to confirm that it has entered the site. Upon confirmation, set the flag `is_entered_room` to execute the actual behavior of SURVEY state.

`getIntoNest()`

Get into the nest (central room) if the robot is close enough to the LED. This function is executed only when the robot is in WAGGLE state. The robot finds the LEDs placed at the entrance of the site other than the current opinion and tries to go away from the site towards those LEDs. When the robot is away a certain distance from the LED of current opinion, confirm that it has entered the nest. Upon confirmation, set the flag `is_entered_room` to execute the actual behavior of WAGGLE state.

`executeAndFlipState()`

Set timer, the number of steps the robot has to execute the current state. If the current state is SURVEY, a uniform random integer value between 150 and 300 is taken as the number of steps the robot has to execute. If the current state is WAGGLE, an exponential value proportional to the average of metrics v calculated using equation (2) is set to the number of time steps t the robot has to move before changing the state to SURVEY. The values of rate parameter λ and weight W are set to -2 and 40 respectiveley.

$$t = W\lambda(1 - e^{-\lambda v}) \tag{2}$$

If the current state is SURVEY, assess the quality of the rooom by calling `updateMetrics()`. If current state is WAGGLE, broadcast it's opinion. On timeout, if the current state is WAGGLE then stop broadcasting the opinion and adopt an opinion from random neighbor. Flip the current state from SURVEY/WAGGLE to WAGGLE/SURVEY and reset all the flags `is_room_sensed`, `is_at_entrance`, `is_entered_room`.

`updateMetrics()`

Update the metrics of the robot when best values found. The average of the metrics found v is calculated as in equation (3).

$$v = \begin{cases} (v_G + v_O)/2 & \text{if type G robot} \\ (v_L + v_O)/2 & \text{if type L robot} \end{cases} \tag{3}$$

5 Analysis and Results

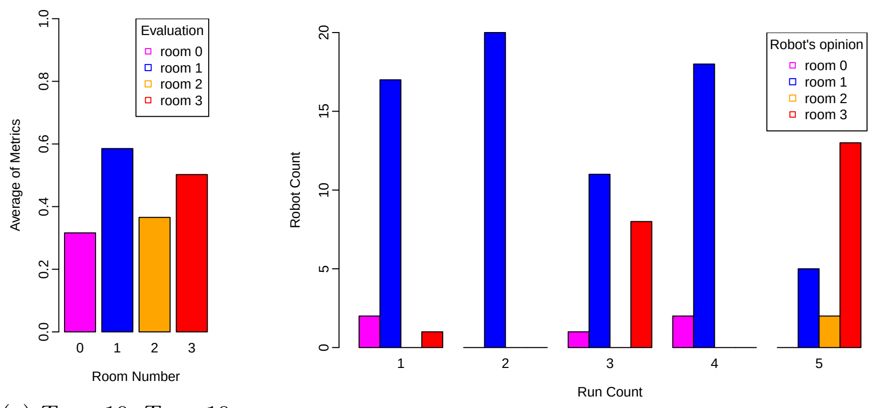
The results for 9 experiments by varying the robot count, each executed 5 sequential runs is presented in figures 3, 4, 5. Every run in each experiment is executed for 5000 ticks. T_G is the number of type G robots and T_L is the number of type L robots. The left side bar graph on each plot shows the average of metrics of the original configuration on each room. The right side bar graph shows the opinions of the robots on each run.

The results show that the robots try to choose room with high average of metrics irrespective of the average of only perceivable metrics (a robot may have a ground or light sensor value, but not both). Eventually, almost every robot agrees on the same room. If the difference between the average of metrics v of each room is high, the robots perform better and may give optimal solution and also take less time for consensus. If these values are closer, the robots try to find near optimal solutions and may take larger time for consensus. Too many robots increases the time for the robots to agree on the same room, but initially they give many (near optimal) solutions, slowly decreasing the number of solutions by forming clusters. Too many robots is a good choice to find many better solutions rather than the single best solution.

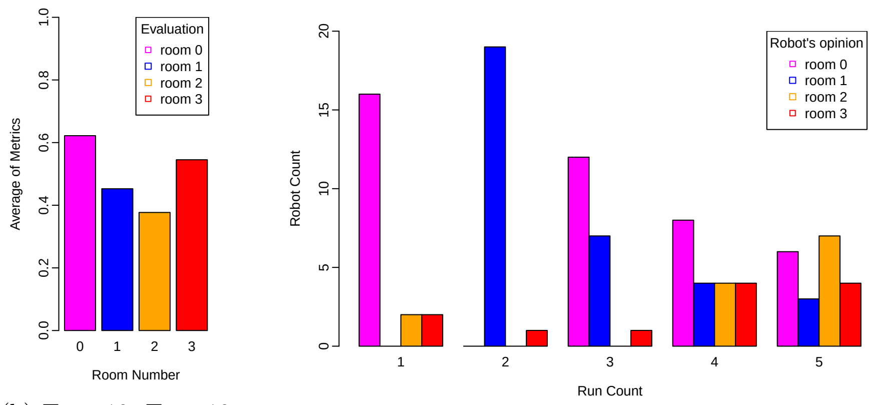
For the given configuration 20-25 robots of equal sizes of T_G and T_L is a good value to have a (near) optimal solution. The results vary by varying the ratio of T_G and T_L. If the robot count of type G (T_G) is more compared to type L (T_L), the results favor the room with high ground sensor values.

6 Conclusion

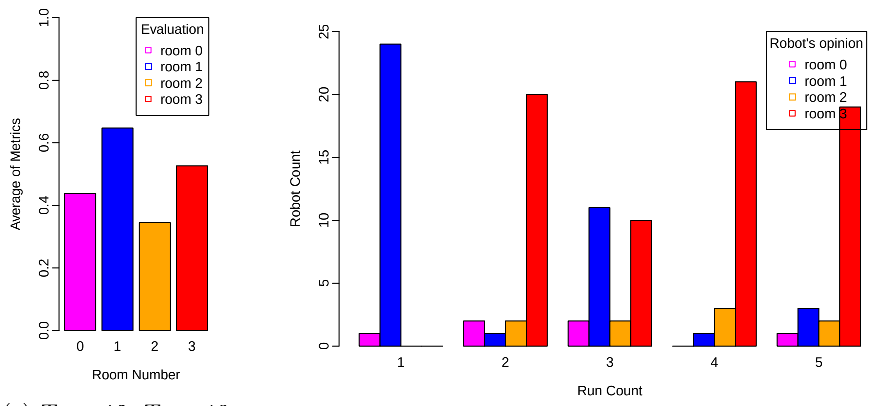
The hybrid method using house hunting behavior of honeybees and opinion dynamics for collective decision making suits well where the agents are heterogeneous, and have noisy sensors and limited communication capabilities. Since, the room entrance for each room in the given experiment configuration is a narrow passage, the random walk leads to congestion problem where robots spend a lot of time on collision avoidane if too many robots try to enter and exit the same room. Forming patterns to enter and exit the room for congestion conrtol instead of random walk may improve the quality and time complexity. The positive feedback mechanism here is, the amount of time a robot waggle dance is directly proportional to the quality of metrics. Exponential time for the waggle dance improves the quality of decision by spending much time on influencing other to choose better solution.



(a) $T_G = 10$, $T_L = 10$

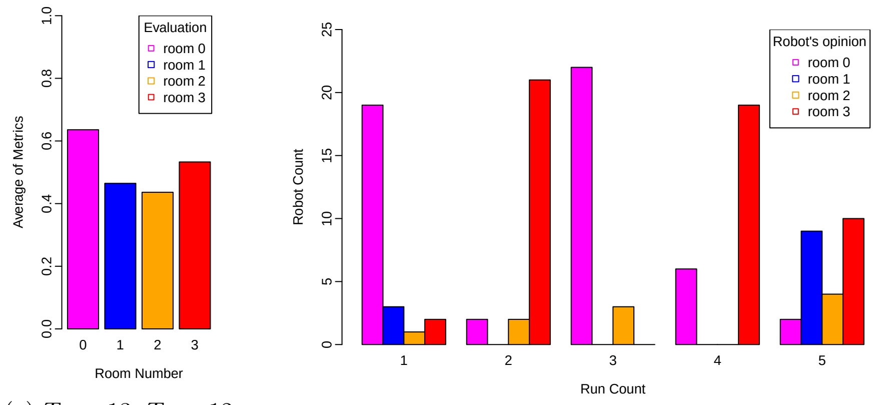


(b) $T_G = 10$, $T_L = 10$

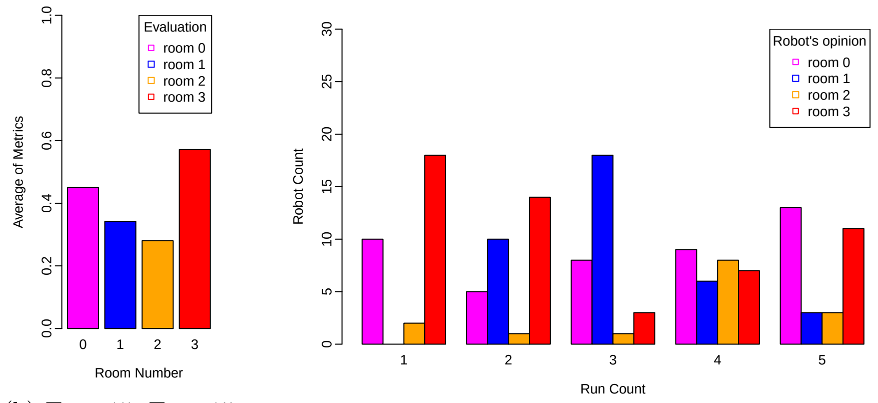


(c) $T_G = 12$, $T_L = 13$

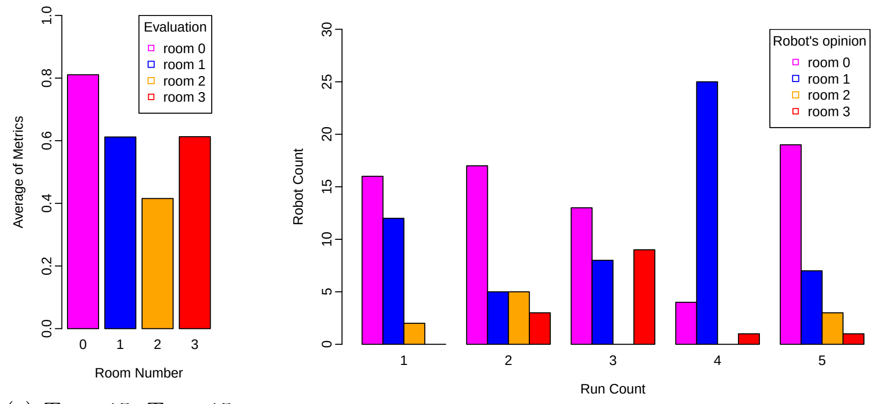
Fig. 3: Results of experiments on different configurations with (a) 20 robots, (b) 20 robots, and (c) 25 robots.



(a) $T_G = 13$, $T_L = 12$

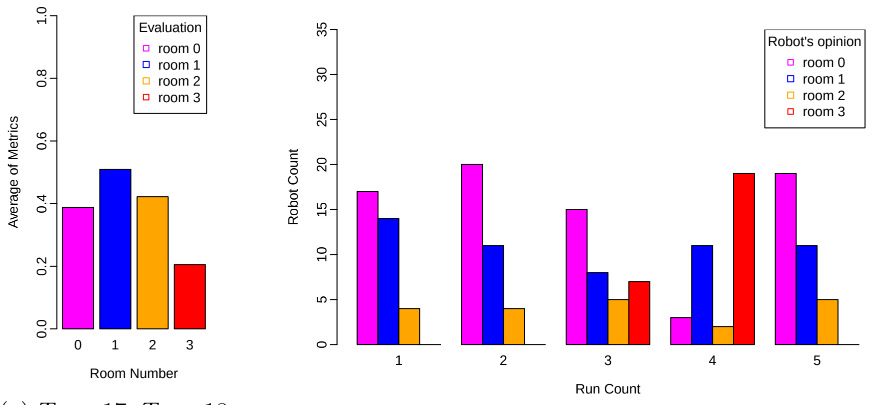


(b) $T_G = 15$, $T_L = 15$

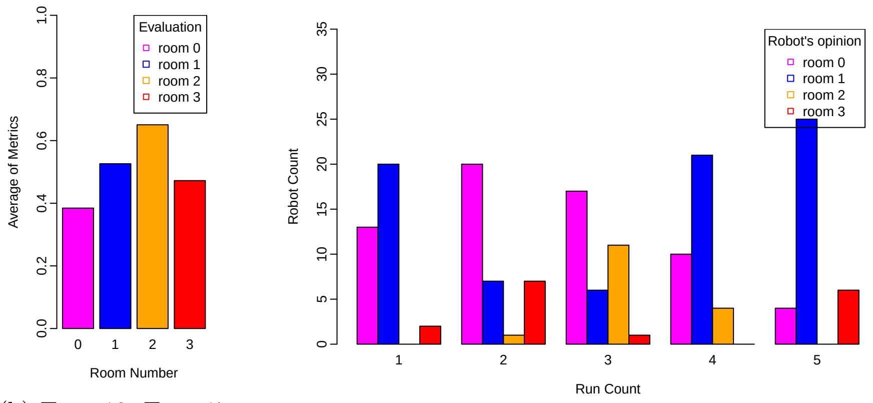


(c) $T_G = 15$, $T_L = 15$

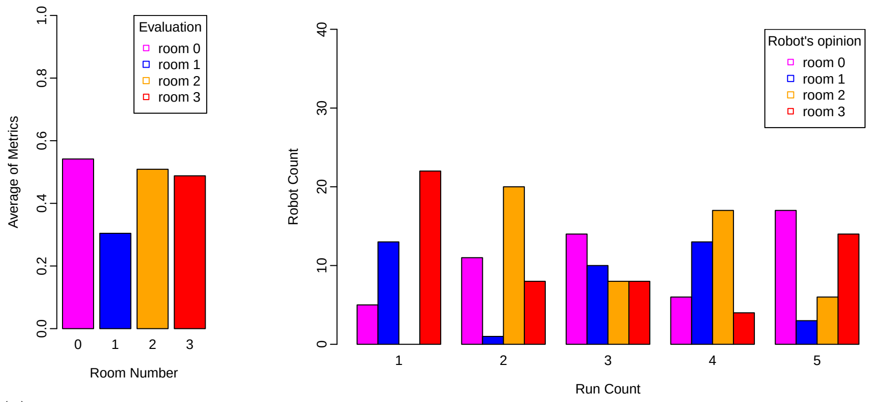
Fig. 4: Results of experiments on different configurations with (a) 25 robots, (b) 30 robots, and (c) 35 robots.



(a) $T_G = 17$, $T_L = 18$



(b) $T_G = 18$, $T_L = 17$



(c) $T_G = 20$, $T_L = 20$

Fig. 5: Results of experiments on different configurations with (a) 35 robots, (b) 35 robots, and (c) 40 robots.

References

1. Valentini, G., Hamann, H., Dorigo, M.: Self-organized collective decision making: The weighted voter model. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems. AAMAS '14, Richland, SC, International Foundation for Autonomous Agents and Multiagent Systems (2014)