SWARM COMPACTNESS MAINTENANCE USING ONLY LOCAL COMMUNICATION

BRANDTNER Daniel
CHATAIN Julia
LEDERREY Gael
Presentation

- Sub-microscopic model

  ![Webots™](image)

- Following a beacon

- Macroscopic model

![Diagram](image)
Sub-microscopic model
α-algorithm

- Finite State Machine
- Alpha threshold
- Local communication

(i) Neighbor < ALPHA_THRESHOLD && Neighbor < Previous_neighbor
(ii) Counter == TC && Neighbor > Previous_Neighbor
(iii) Counter == TC && Neighbor <= Previous_Neighbor

Fig. 1: Relations between the states for the α-algorithm.
Webots simulation
Webots: $\alpha = 10$
Webots: $\alpha = 20$
Robots and States

![Graphs showing the number of robots in a state against the number of connections for different states: FORWARD, COHERENCE, OBSTACLE_AVOIDANCE, and SUM.](image-url)
Macroscopic model

% Definitions
xSpace = linspace(0,36,31); % Linspace for the plotting

% If geometricModel || movingTowardProbs || statesProbs || monteCarloProbs
fA = @(x) compute_pA(x, dmax); % Computes the probability of moving toward for Avoidance state
fC = @(x) compute_pC(x, alpha); % Computes the probability of moving toward for Coherence state
Pac = monte_carlo(Ra + Rp, Ra + Rp + 2 * v * T, 0, Ra + Rp, fA, 0, V, T); % Probability of moving for zone A to zone C
Pin_A = @(v) (Ra + Rp + 2 * v * T)^2 - (Ra + Rp)^2 / (v - Ra + Rp)^2; % Probability for robot to be in area A
fF = @(x) compute_pF(x, alpha, dmax, Pac, Pin_A); % Computes the probability of moving toward for Forward state
fAx = @(x) compute_pAx(x, Ra, Rw, Rp, V, T, dmax); % Computes the probability of collision with another robot
fLx = @(x) compute_pLx(x, Ra, Rw, Rp, V, T, Tc, alpha, dmax, Pac, Pin_A); % Computes the probability of losing a connection when in Forward state
fGx = @(x) compute_pGx(x, Ra, Rw, Rp, V, T, Tc, alpha); % Computes the probability of gaining a connection
fRx = @(x) compute_pRx(x, Ra, Rw, Rp, V, T, Tc, alpha); % Computes the probability of recovering a connection
fFx = @(x) compute_pFx(x, Ra, Rw, Rp, V, T, Tc, alpha); % Computes the probability of failing in recovering a connection
fLx = @(x) compute_pLx(x, Ra, Rw, Rp, V, T, Tc, alpha); % Computes the probability of losing a connection when in Coherence state
fPpA = @(x) monte_carlo(Ra + Rp, Ra + Rp + 2 * v * T, 0, Ra + Rp, fA, x, V, T); % Computes the probability of moving from zone A to zone C
fPpC = @(x) monte_carlo(Ra + Rp, Ra + Rp + 2 * v * T, 0, Ra + Rp, fC, x, V, T); % Computes the probability of moving from zone C to zone A
fPpF = @(x) monte_carlo(Ra - 2 * v * Tc * T, Rw, Rw, Rw + 2 * v * Tc * T, fF, x, V, T); % Computes the probability of moving from zone A to zone F
fPpF = @(x) monte_carlo(Rw - 2 * v * Tc * T, Rw, Rw + 2 * v * Tc * T, fF, x, V, T); % Computes the probability of moving from zone F to zone A
fPpC = @(x) monte_carlo(Rw - 2 * v * Tc * T, Rw, Rw + 2 * v * Tc * T, fC, x, V, T); % Computes the probability of moving from zone C to zone F
fPpF = @(x) monte_carlo(Rw + 2 * v * Tc * T, Rw - 2 * v * Tc * T, fF, x, V, T); % Computes the probability of moving from zone F to zone C
end

if movingTowardProbs
plot(xSpace, arrayfun(fA, xSpace), 'o', 'LineStyle', '--', 'Color', [0.8 0.2 0.3]);
endplot(fA,[0,30], '-b');
hold on;
plot(xSpace, arrayfun(fC, xSpace), 'o', 'LineStyle', '--', 'Color', [0.8 0.2 0.3]);
endplot(fC,[0,30], '-r');
hold on;
endpax = 0.02;
comp_pAx = @(x) compute_pAx(x, Ra, Rw, Rp, V, T);
endfF = @(x) compute_pF(x, alpha, dmax, comp_pAx);
endplot(xSpace, arrayfun(fF, xSpace), 'o', 'LineStyle', '--', 'Color', [1 0 0]);
endplot(fF,[0,30], '-r');
grid;
title('Probabilities of moving towards in avoidance');
xlabel('Probabilities');
ylabel('Connections');
legend('P_a(x)', 'P_c(x)', 'P_f(x)');
end
Description

- \( P_a \): collision with another robot
- \( P_f \): loss of a connection in forward state
- \( P_g \): gain of a connection
- \( P_r \): recovery of a connection
- \( P_f \): failure to recover a connection
- \( P_{la} \): loss of a connection in coherence state
Comparison between models

Number of connections vs. Number of robots in a state

- FORWARD
- COHERENCE
- OBSTACLE_AVOIDANCE
- SUM
Convergence of the model
Beacon Following
Algorithm

- Expending on the $\alpha$-algorithm
- A beacon illuminates some of the robots
- The rest are trying to stick to them, making the whole swarm move toward the beacon
Swarm trajectory

- Average of the swarm trajectory over the 30 runs
- Trajectory of the swarm for one of the runs
- Position of the beacon
Optimize light sensitivity

Threshold for the number of illuminated sensors : 3
Threshold for the number of illuminated sensors : 4
Optimize $\alpha$ threshold

![Graph showing the impact of different alpha thresholds on distance over time. The graph has two axes: Time [s] on the x-axis and Distance [m] on the y-axis. Three lines are plotted, each corresponding to a different alpha threshold: 10, 15, and 20. The line for alpha threshold 10 is green and dashed, the line for alpha threshold 15 is red and dashed, and the line for alpha threshold 20 is blue. The graph shows that as the alpha threshold increases, the distance decreases more rapidly.]
Standard Deviations

![Graph showing standard deviations over time and distance with three curves for different values: Alpha 10, Alpha 15, and Alpha 20. The x-axis represents time in units of $10^4$, and the y-axis represents distance. The curves illustrate the variation in standard deviation over time for each alpha value.]
Moving Beacon
Webots simulation
Questions?