XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGENEERING

Swarm Robotics for Caging-based Collaborative Transport of Multiple Objects

Jeany N. A. Laranjeira[1], Nadia Nedjah[2], Luiza de M. Mourelle[3]

[1]*Postgraduation Program in Electronics Engineering, State University of Rio de Janeiro, Brazil Rua São Francisco Xavier, 20550-013, Maracanã, Rio de Janeiro, Brazil laranjeira.jeany@posgraduacao.uerj.br*
[2]*Department of Electronics Engineering and Telecommunications, State University of Rio de Janeiro Rua São Francisco Xavier, 20550-013, Maracanã, Rio de Janeiro, Brazil nadia@eng.uerj.br*
[3]*Department of Systems Engineering and Computation, State University of Rio de Janeiro Rua São Francisco Xavier, 20550-013, Maracanã, Rio de Janeiro, Brazil ldmm@eng.uerj.br*

Abstract. This work presents a swarm robotics approach for caging-based transportation of multiple objects in automated environments. The motivation stems from the challenge of enabling a swarm of autonomous robots to efficiently locate, coordinate, and transport multiple objects simultaneously, while avoiding collisions and maintaining group cohesion. The proposed model simulates a group of mobile robots that explore a predefined arena using a foraging strategy to search for objects distributed across the environment. Upon locating an object, the detecting robot assumes the role of recruiter, estimating the required number of collaborators based on the object's mass and dynamically selecting the nearest available robots to form a caging formation. Once the formation is established, the swarm initiates the transport operation, maintaining alignment and coordination throughout the trajectory. To support multiple simultaneous transport tasks, the system incorporates a path conflict detection and resolution mechanism. This mechanism prevents collisions at intersection points by prioritizing the group nearest to the crossing, while others temporarily yield until safe passage is confirmed. Key challenges addressed include real-time inter-robot synchronization, precise agent positioning around objects, and the management of intergroup priorities to prevent collisions without compromising efficiency. These challenges are addressed through a combination of geometric path planning, signal-based communication, and modular control logic, resulting in a robust and scalable coordination framework. The conflict resolution mechanism effectively prevents collisions between robot groups, preserving the integrity of transport operations even in densely populated environments. The methodology is implemented in the CoppeliaSim simulation environment using Khepera-III mobile robots. Simulation results, in terms of time requirements regarding object search, robots' recruitment and object transport as well as transport path error, demonstrate that multiple swarms can successfully and efficiently transport multiple objects to their designated targets. The system also exhibits strong adaptability, maintaining coordinated behavior under varying robot and object densities.

Keywords: Swarm Robotics; Collaborative Object Transport; Caging-based Transport; Mobile Robot Coordination; Collision Avoidance.

1 Introduction

The use of multi-robot systems offers a promising solution for executing complex tasks in automated environments. Among these tasks, collaborative object transportation enables robots to jointly move loads that are too heavy for individuals. While many works explore the transport of a single object, managing multiple objects with simultaneous transport groups still poses challenges.

This paper proposes a swarm robotics approach for the caging-based transportation of multiple objects, with a mechanism to avoid collisions between transport groups. Robots explore the arena in a decentralized way, detect objects, and form transport groups. A key feature is the verification of path crossings, which defines priorities and applies waiting and release rules to prevent group collisions.

The methodology is simulated in CoppeliaSim with Khepera III robots, integrating alignment, communication, synchronization, and modular control. Results show high delivery success and effective collision avoidance in dynamic, dense environments.

The paper presents a progressive structure: Section 2 reviews related works; Section 3 introduces the proposed method, detailing foraging, recruitment, formation, and crossing checks. Section 4 presents simulation results validating efficiency and scalability. Section 5 concludes with suggestions for future developments.

2 Related Works

Koung et al. (2021) propose a control method based on Hierarchical Quadratic Programming (HQP) to manage formation, navigation, and obstacle avoidance in multi-robot systems. Validated with industrial robots, it proves suitable for logistics scenarios. In contrast, Jurt et al. (2022) present a decentralized approach for transporting large, irregular objects using robotic swarms. Inspired by ants behavior, it allows on-demand recruitment without prior shape knowledge, showing promising results in logistics and disaster response. Unlike these, our work addresses multi-object transport with dynamic path-crossing checks and group coordination.

Decugnière et al. (2008a,b) introduce the cart-bot, a system inspired by insect colonies that collects and stores multiple items before transport. Decugnière et al. (2008a) analyzes its effectiveness, while Decugnière et al. (2008b) highlights the superlinear performance gain as more robots are added, reinforcing its scalability.

Nath et al. (2020) propose social negotiation among physically linked robots with heterogeneous perceptions to navigate and transport cooperatively. Herranz et al. (2022) evolve decentralized control via genetic programming, enabling swarm subgroups to handle multiple objects. Differently, our work enhances coordination with dynamic path-crossing validation and independent group transport in shared arenas.

3 Collective Caging-Based Transportation in Swarm Robotics

Cardoso (2023) presents a caging-based approach for collective object transportation in swarm robotics, structured in three phases: object foraging, coordinated recruitment and caging, and transportation. Implemented in CoppeliaSim with Khepera-III robots, the method proved effective in maintaining formation and ensuring uniform trajectories. However, it lacked solutions for multiple objects and inter-group path conflicts.

Built upon this, the present work extends the approach to support simultaneous multi-object transport, dynamic recruitment based on object mass, and path-crossing verification. These enhancements improve scalability and robustness in larger arenas.

Key challenges addressed include: recruiting the appropriate number of robots based on object mass; preventing collisions between simultaneously active transport groups; and ensuring safe, known return paths. The proposed model introduces decentralized recruitment, reuse of foraging paths, and dynamic coordination rules to overcome these limitations.

3.1 Proposed Method for Collaborative Transportation of Multiple Objects

The proposed method organizes the collective transportation of multiple objects into three phases: (i) foraging and recruitment, (ii) caging, and (iii) coordinated transportation. Each phase is executed in a distributed manner by a set of autonomous mobile robots that communicate via global signals to coordinate their actions.

Algorithm 1 presents the main system flow for the collective transportation of multiple objects. Initially, the arena is configured with landmarks and a random distribution of objects. The main loop runs while the number of delivered objects remains below the total expected. The logic is state-driven: (i) foraging ($state = 0$), where the robots explore the arena to locate objects; (ii) caging ($state = 1$), where recruited robots align around the object in preparation for transportation; and (iii) transportation ($state = 2$), where the group performs a coordinated movement toward the destination. If, during the transport phase, two or more groups perform object transportation simultaneously, a path-crossing verification is carried out. If an intersection point is detected between their trajectories, the group that is farther from the crossing point is temporarily blocked to prevent collisions between transportation groups. At the end of each cycle, the system updates the count of delivered objects. While there are still objects available in the arena that have not been transported, the robots return to state 0 and resume exploration to locate new objects. Only when all objects have been delivered, do the robots return to the base. This structure ensures the orderly execution of phases.

3.2 Foraging, Recruitment, and Caging

During the foraging phase, each robot performs a decentralized movement to move to a random reference point in its neighborhood. This ensures a broader arena coverage and increasing the chances of object detection in partially unknown environments. We adopted this approach instead of a greedy strategy, as randomness favors trajectory diversity and reduces robot overlap. In this work, the recruiter does not need to follow the shortest path, only to locate the object, and transport takes place along the same path used in the search, which ensures a known route back to the base.

In the foraging phase, as described in Algorithm 2, the robots explore a discretized arena marked with landmarks, detect available objects, and signal their discovery. The detecting robot assumes the role of *recruiter*, initiating a process to evaluate the mass of the object and calculate the number of robots required for collaborative

Algorithm 1 Collective transportation of multiple objects

initialize arena, landmarks, and objects
while $delivered_objects < total_objects$ **do** ▷ continue until all objects are delivered
  if $state = 0$ **then**
    initialize local variables
    forage for objects for all robots ▷ robots explore to detect objects
    $state \leftarrow 1$
  else if $state = 1$ **then**
    cage object for all recruited robots ▷ position robots around the object
    $state \leftarrow 2$
  else if $state = 2$ **then**
    transport object for all caging robots ▷ perform cooperative transport toward destination
    $state \leftarrow 0$
  end if
  $(delivered_objects) \leftarrow$ count delivered objects ▷ update number of completed deliveries
end while
return to base ▷ robots return to the end position after delivering all objects

caging-based transportation. The recruiter then identifies, based on distance, the nearest available robots and sends recruitment signals to form the transportation group according to the received availability responses.

Algorithm 2 Foraging for objects by robot i

while search = **true do** ▷ until object is not found
  go to a random landmark in the vicinity
  if object detected **then**
    check object found
    if object available **then**
      signal object O_k found ▷ notifies other robots about the found object
      $m \leftarrow$ compute object mass using Equation (1) for O_k
      $n \leftarrow$ compute required robots using Equation (2) for O_k
      $availableRobots \leftarrow$ request all robot availability ▷ send signal asking if free or busy
    end if
  end if
  verify received availabilities ▷ verify responses with distances
  if $availableRobots \geq n$ **then**
    recruit robots (O_k, $responses$, n) ▷ recruiter sends recruitment message to robots
  end if
  robot verifies message of recruitment ▷ check if recruitment message received
  if $O_k \neq \emptyset$ **then** ▷ if no objects available, stop search
    $search \leftarrow$ **false**
  end if
end while

The mass m of the detected object is computed based on its geometric characteristics, considering object o as a solid cylinder. Its value is calculated using Equation 1:

$$m_o = d_o \cdot V_o = d \cdot \pi r_o^2 h_o, \tag{1}$$

where d is the material density of the object, V is its volume, r_o is its radius, and h_o is its height. This formulation assumes cylindrical symmetry to estimate the mass from geometric and material properties. Once the mass m_o is known, the minimum number of robots n_o required to perform the transportation of objetct o is computed using Equation 2:

$$n_o = \left\lceil \frac{m_o}{c} \right\rceil, \tag{2}$$

where c is the maximum capacity of each robot to contribute to the collective transport, defined as a proportion of its own mass (typically 30%). The use of the ceiling function ensures that a sufficient number of robots is selected, even when the division is not exact.

The control of object localization and availability is managed through a state mechanism associated with each object present in the arena. Initially, all objects are classified as available, allowing any robot that detects them to initiate the recruitment process. When a robot identifies an object during foraging, it queries the object's current state to verify whether it has already been located by another group. If the object is still marked as $available$, the robot updates its state to $found$, broadcasting a global signal indicating that the object is in the allocation phase and preventing concurrent recruitment attempts. Once transportation is successfully completed and the object reaches its destination, its state is updated to $delivered$, preventing further interaction during the remainder of the process. This state flow, $available$, $found$, and $delivered$, is managed through global signals in the simulation environment, ensuring consistency and avoiding conflicts between multiple active transportation groups.

During the foraging phase, the robots continuously monitor availability requests from other agents, allowing them to be recruited to assist in the collective transportation of objects detected by others. Upon recruitment, they receive the object identifier (O_k), coordinates (x, y), radius (r), recruiter identifier (ρ_r), required number of robots (n_{O_k}), angular position in the caging formation, and the return path vector (Γ). These data ensure alignment and the coordinated execution of the transportation process.

The recruitment, as described in Algorithm 3, and caging, as described in Algorithm 4, processes have been modified with respect to Cardoso (2023), resulting in a significant improvement in performance. Robots are recruited simultaneously and directed to a waiting position, preconfigured to form the caging circle around the object while maintaining a predefined small distance from it. This spacing prevents interference and ensures that other robots can reach their caging positions without obstacles. Once all robots have reached their respective positions, the recruiter emits a signal that prompts the group to initiate the caging phase in a coordinated and simultaneous manner.

Algorithm 3 Recruiting robots for collaborative transportation

get $O_k, responses, n, \Gamma$; $recruited_group \leftarrow \emptyset$
$dist \leftarrow$ calculate distance from each robot in $responses$ to object O_k
$sorted_robots \leftarrow$ sort $responses$ by $dist$ in ascending order
for $i = 1$ to n **do** ▷ recruit the closest robots up to the required number
    send recruitment message $(O_k, x, y, r, \rho_r, i, \Gamma)$
    $recruited_group \leftarrow sorted_robots_i$
end for
return $recruited_group$

This approach increases efficiency and considerably reduces recruitment and caging delays. Regardless of the number of robots involved, the timings remain nearly constant since all actions occur in parallel. The main factor directly influencing the overall recruitment duration is the distance between the recruited robots and the object at the moment of detection. The key improvement lies in selecting the closest robots to the object for recruitment, further optimizing performance.

Algorithm 4 Caging of object for all recruited robots

compute alignment and standby positions
move robot to standby position
if robot reached standby position **then**
    $\beta \leftarrow$ all robots have reached standby position
end if
if β **then**
    move robot to caging position
    $\psi \leftarrow$ all robots have reached caging position
end if
return ψ

The recruited robots respond to the signal, updating their state, assuming the caging state, and moving toward the object, positioning themselves symmetrically around it to ensure balanced force distribution. The formation is only considered complete when all recruited robots reach their transport positions and their aligments, regarding the next waypoint they should go to, is confirmed. Only after this stage does the group initiate the object transport phase toward the destination. This process is repeated in parallel for different objects distributed across the arena, enabling the formation of multiple transportation groups operating simultaneously while performing the path-crossing verification described in the Section 3.3.

3.3 Transportation and Path-Crossing Verification

Algorithm 5 describes the main transportation cycle. While the final destination has not been reached, each robot retrieves its next position λ from its list of landmarks in the return path γ, performs path-crossing verification, applies PID (Proportional-Integral-Derivative) control for position and orientation, and checks whether the current target position λ has been reached.

Algorithm 5 Transportation to the origin point in robot i

```
wait for transport signal (i, Γ)          ▷ waits for transport start message from recruiter
while destination not reached do
    λ ← get next position in path (Γ)     ▷ read next landmark λ in path Γ
    ω ← Caminhos de outros grupos
    check path crossing (Γ, ω)            ▷ verifies if path Γ intersects with another group ω
    apply PID control (i, λ)              ▷ uses PID control to move robot i to position of next landmark λ
    verify arrival (i, λ)                 ▷ checks if robot i has reached the target position of next landmark λ
end while
```

In the transportation of multiple objects, the groups follow trajectories Γ, defined during the foraging stage along the landmarks toward the delivery point. During cooperative movement, the PID controller continuously adjusts each robot's linear and angular velocities, minimizing distance and angular errors relative to the target position.

A path-crossing verification mechanism is triggered at each new waypoint, monitoring the next two segments of each group's route. This mechanism detects both geometric intersections between consecutive segments and landmark overlaps, preventing collisions in scenarios involving simultaneous transportation. The geometric verification of path crossing is performed in two main steps. The first step uses the counterclockwise criterion, to verify whether two line segments intersect, as defined in Equation 3:

$$\text{pcv}(p_1, q_1, q_2) \neq \text{pcv}(p_2, q_1, q_2) \wedge \text{pcv}(p_1, p_2, q_1) \neq \text{pcv}(p_1, p_2, q_2), \tag{3}$$

where $\text{pcv}(a, b, c)$ is a boolean function that returns true if the points (a, b, c) form a counterclockwise turn; p_1 and p_2 are the endpoints of the first segment; and q_1 and q_2 are the endpoints of the second segment. This criterion evaluates the relative orientation of triplets of points to determine whether the endpoints of one segment lie on opposite sides of the other segment.

If a crossing is confirmed, the exact coordinates of the intersection point $\zeta = (x_\zeta, y_\zeta)$ are calculated using Equations 4 and 5:

$$x_\zeta = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \tag{4}$$

$$y_\zeta = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \tag{5}$$

where (x_1, y_1) and (x_2, y_2) are the endpoints of the first segment, and (x_3, y_3) and (x_4, y_4) are the endpoints of the second segment.

To determine which group has priority at the crossing, the algorithm estimates the time τ that each group would take to reach the intersection point ζ. This estimate uses the Euclidean distance δ between the current position of object o and the intersection point, defined in Equation 6:

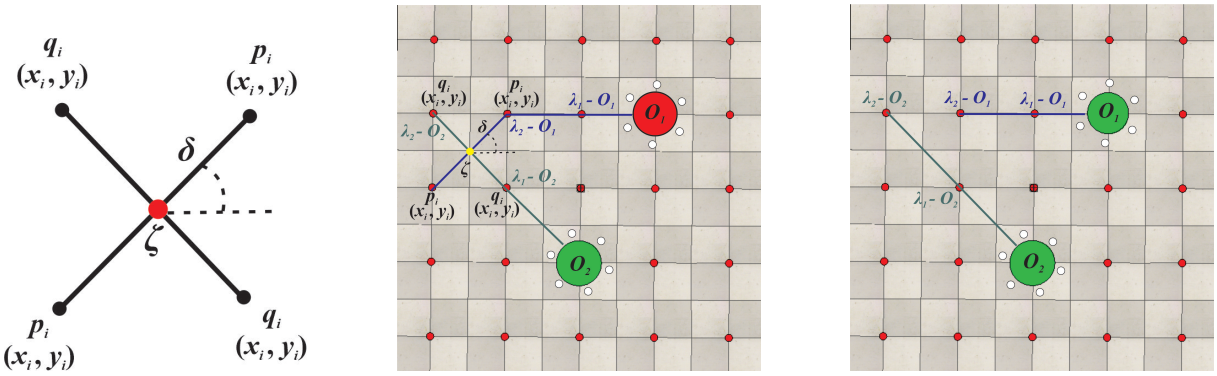
$$\delta = \sqrt{(x_o - x_\zeta)^2 + (y_o - y_\zeta)^2}, \tag{6}$$

and the average transport speed v, resulting in the estimated time to the crossing point given by Equation 7:

$$\tau = \frac{\delta}{v}, \tag{7}$$

where (x_o, y_o) is the current position of the transported object o; (x_ζ, y_ζ) are the coordinates of the intersection point. Figure 1a illustrates this situation. This mechanism ensures that the group formed to transport the objetct closest to the intersection proceeds first, while the other group waits safely, preventing collisions and maintaining efficient system coordination. To reduce the risk of premature blocking, the path-crossing analysis considers only the next two segments in each group's remaining trajectory. Premature blocking occurs when a crossing point ζ is identified too early, even if it is located far ahead in the path. Blocking a group at this stage may cause unnecessary waiting and reduce system performance. In Figure 1b, the full paths are analyzed, and the group caging object O_1 is blocked prematurely, despite the crossing occurring only near the end of the route. In contrast, Figure 1c shows

the intelligent blocking strategy, where only the upcoming segments are evaluated. In this case, blocking is applied only when the crossing is imminent, ensuring better synchronization and avoiding unnecessary delays.



(a) Geometry of crossing check.

(b) Early detection with distant crossing.

(c) Block only when crossing is imminent.

Figure 1. Path crossing verification: (a) Geometric scheme of segments and intersection point ζ; (b) Premature blocking due to full-path detection; (c) Segment-based analysis reducing unnecessary blocking.

The group with the smallest τ is granted priority, while the other enters a waiting mode via a global blocking signal. The release of the blocked group occurs only after the priority group has passed the crossing point ζ. This passage verification compares the current position of the transported object to the coordinates of ζ to ensure that the unlock is triggered only when there is no risk of collision, thereby enhancing coordination robustness.

4 Performance Results

The simulations were carried out in CoppeliaSim with Khepera-III robots and cylindrical objects. The Khepera-III, widely adopted in studies of differential locomotion, has a mass of 0.30Kg and a radius of 0.30m. Simulations were run in 5×5 m and 10×10 m arenas with 0.5 m and 1.0 m discretizations, respectively, allowing analysis in both compact and large-scale environments. Table 1 shows that increasing arena size and number of objects N_o leads to higher average search time T_S, due to longer distances. Recruitment time T_R and its deviation σ_T remain stable, as robots are selected based on proximity. Transport time T_T grows moderately with distance, and total time T_O increases accordingly, without significant efficiency loss. This initial version of the study employs cylindrical objects. In future improvements of this work, different geometric shapes will be used to broaden the analysis and evaluate the performance of the mechanism in more complex configurations. It is important to note that the proposed conflict detection and resolution mechanism is independent of both the robot model and the object shape.

Table 1. Impact of arena size on search and recruitment performance with 12 robots and objects of equal weight 1.1kg

Arena	N_o	**Search**		**Recruitment**		**Transportation**		**Overall Time**	
		T_S (s)	σ_{T_S}	T_R (s)	σ_{T_R}	T_T (s)	σ_{T_T}	T_O (s)	σ_{T_O}
5 x 5 m	1	70.87	13.97	49.42	2.78	118.97	9.84	236.68	20.24
5 x 5 m	2	72.47	6.03	58.81	10.32	116.47	5.89	270.68	17.58
10 x 10 m	1	75.45	21.17	76.13	9.43	89.23	19.92	315.52	22.51
10 x 10 m	2	98.42	15.50	77.50	5.80	123.96	15.50	330.45	19.50

The recruitment stage shows consistent durations, as summarized in Table 2, where groups are dynamically based on object mass and robot proximity. The results T_R and σ_{T_R} reflect low variability in recruitment times, even as the number of recruited robots ρ and object mases vary. Result L_{actual} represents the actual path length and ε the difference between expected and actual paths, measured by the object's center of mass, does not affect performance. Execution time scales proportionally with distance to the destination. Similarly, transportation analysis in Table 3 highlights the relationship between the number of objects N_o, the number of robots ρ, the overall time T_O, and standard deviation σ_T.

Some simulations showed divergent values, increasing the standard deviation — an expected effect of the simulation's stochastic nature. In object search, detection time varies due to random foraging. Recruitment is usually effective but can be slower when robots are far from the object, especially in simultaneous transports,

Table 2. Simulation results: dynamic recruitment and trajectory error

ρ	r (m)	m_o (kg)	T_R (s)	σ_{T_R} (s)	L_{actual} (m)	L_{ideal} (m)	$\varepsilon(\%)$	T_O (s)
7	0.65	1.3	46.42	2.8	6.094	6.325	0.501	320.84
8	0.67	1.5	45.41	2.4	5.972	6.268	0.313	329.43
9	0.70	1.7	46.80	2.5	5.898	6.064	0.531	336.10

Table 3. Impact of the number of objects N_o and robots ρ

N_o	ρ	T_O (s)	σ_{T_O} (s)	v (m/s)
1	7	235.31	10.71	0.045
2	12	270.68	17.58	0.044

where the recruiter with higher priority recruits the closest robots and, thus, the corresponding object is caged faster, while the lower priority relies on the remaining robots that might be further away. Transport times remain stable for similar path lengths but differ when return routes vary, as the object follows the recruiter's search path.

Studies with larger arenas (10 × 10 m) with high number of objects distributed, forming several transportation groups coordinated by a large swarm of robots, are ongoing. These additional scenarios aim to assess the robustness of the methodology in complex environments involving group interactions and trajectory crossings. As proof of the model's functionality, simulation videos are available at: **`https://linktr.ee/roboticsSwarm`**.

5 Conclusion

This work proposes a method for implementing collective caging-based transportation of multiple cylindrical objects in a simulated environment. The method, distributed and coordinated through message exchanges among a swarm of robots, demonstrates efficiency in forming dynamic groups and maintaining the caging formation during transportation to the destination points. The analysis of the paths traversed by the objects' center of mass revealed that the average trajectory error remained low, highlighting the robustness of the strategy even with varying swarm sizes and object masses. The results indicate that the system is capable of handling more complex scenarios, with multiple objects distributed in a larger arena (10 m × 10 m) and transportation groups operating simultaneously.

As future work, we intend to compare the obtained results with approaches reported in the literature to evaluate the performance of the proposed method and identify potential improvements. Furthermore, improving the foraging phase through trajectory optimization algorithms is planned, since the first robot to detect the object may not have followed the shortest path. More complex arena topologies with static and dynamic obstacles will also be investigated, as well as scaling the system to scenarios with a larger number of objects and robots, enabling the analysis of emergent behavior and the scalability of the proposed approach.

References

D. Koung, O. Kermorgant, I. Fantoni, and L. Belouaer. Cooperative multi-robot object transportation system based on hierarchical quadratic programming. *IEEE Robotics and Automation Letters*, vol. 6, n. 4, pp. 6466–6472, 2021.

M. Jurt, M. Jurt, E. Milner, E. Milner, M. Sooriyabandara, M. Sooriyabandara, S. Hauert, and S. Hauert. Collective transport of arbitrarily shaped objects using robot swarms. *Artificial Life and Robotics*, vol. 27, n. 2, pp. 365–372, 2022.

A. Decugnière, A. Decugnière, B. Poulain, B. Poulain, A. Campo, A. Campo, C. Pinciroli, C. Pinciroli, B. Tartini, B. Tartini, M. Osée, M. Osee, M. Dorigo, M. Dorigo, M. Dorigo, M. Birattari, and M. Birattari. The cart-bot and the cooperative transport of multiple objects in the swarmanoid project. *ANTS Conference*, 2008a.

A. Decugnière, A. Decugnière, B. Poulain, B. Poulain, A. Campo, A. Campo, C. Pinciroli, C. Pinciroli, B. Tartini, B. Tartini, M. Osée, M. Osee, M. Dorigo, M. Dorigo, M. Dorigo, M. Birattari, and M. Birattari. Enhancing the cooperative transport of multiple objects. *ANTS Conference*, 2008b.

A. Nath, A. Nath, A. Ar, A. R. Arun, R. Niyogi, and R. Niyogi. A distributed approach for autonomous cooperative transportation in a dynamic multi-robot environment, 2020.

G. L. Herranz, G. L. Herranz, S. Hauert, S. Hauert, S. Jones, and S. Jones. Decentralised negotiation for multi-object collective transport with robot swarms. *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, 2022.

K. d. S. Cardoso. Transporte coletivo por enjaulamento em robótica de enxame. Dissertação de mestrado, Universidade do Estado do Rio de Janeiro, Rio de Janeiro. Dissertação (Mestrado em Engenharia Eletrônica), 2023.