An Ant-Inspired Algorithm for Multi-Robot Navigation and Distributed Simultaneous Task Allocation

Faraz Shamshirdar, Rakesh Shrestha and Richard Vaughan

***Abstract*— This paper describes multi-robot foraging in an unknown environment with an unknown number of sources and a single sink (home). The goal is to effectively allocate robots to sources with least interference and maximum rate of resource collection. The navigation between source and sink is achieved by using trails of virtual landmarks inspired from ant-like trail laying behaviour.**

The algorithm is tested in simulation with varying number of robots and sources. The results indicate that our method achieves better performance than the original algorithm in case of multiple sources.

I. INTRODUCTION

We present an algorithm for the classical *resource transportation* task in an unmapped environment. Robots start from arbitrary positions and search for a supply of resources. On reaching the source, they receive a unit of resource and return home with it, then return back to fetch more resource repeatedly for the length of a trial. Our main focus in this paper is finding task allocation that maximizes the rate of collection of the resources (*throughput*) and with minimum congestion in trails. These two goals are mostly overlapping, hence we achieve the former by trying to solve the latter.

The earlier works [1] for this task present Localization-Space Trails (LOST) which is an implementation of ant-inspired trail following for imperfectly-localized robots. In that algorithm, robots generate and share trail data structures composed of waypoints specified with reference to task-level features. The trails are continuously refined online, and maintain the ant-algorithm property [2] of converging to near-optimal paths from source to home.

In the LOST and other ant-algorithm based methods, time is used as the distance function to be optimized, by which the discovered paths can be longer in space but shorter in time since they spread robots out to minimize spatial interference between robots. When the population size is increased, these interferences eventually dominate the traveling cost. However, since ant algorithms tend to converge to a single best trail, in a large population this trail can become congested. Since the behaviour of the ant algorithms (including LOST) is to explore the environment to discover the target while finding the shortest path to that, in the case of multiple sources – which we are trying to explore – they do not perform well as they try to minimize the path to a random source (which might be the closest one). Hence, we modify the original algorithm such that it does not converge to a single source leading to better throughput.

Autonomy Lab, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada

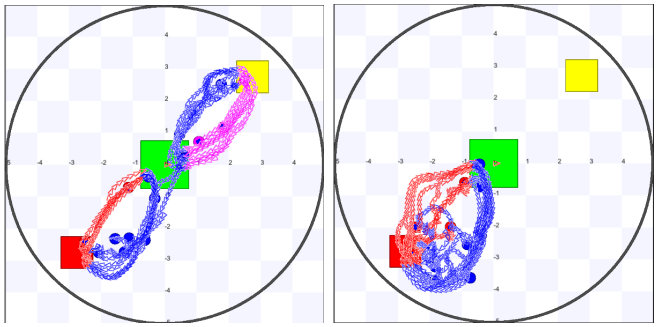


Fig. 1: Trails formed in an empty environment using our embodied apprach outlined in section IV (left) and SO-LOST (right)

In this paper we describe two modifications to the existing LOST algorithm that improve performance in the case of multiple sources and a large number of robots as well as maintaining similar performance in a single-source, single-sink setting with less number of robots. We were able to get superior performance in most of the test-cases we examined.

II. RELATED WORKS

Various works of ant-inspired trail following algorithms have been published, ranging from leaving real chemical trails [3] to infra-red message transmission [4].

Our work closely follows from Localization-Space Trails (LOST) [1] and its variants which is suitable for imperfectly localized robots. A brief review of LOST will be provided in subsection II-A. The method is to generate and share short-lived waypoints (*crumbs)* by robots, which contain some information (estimated distance to target, time to target, density of waypoints, to-source or to-sink label). By following these crumbs, robots can find optimal path between source and home. In the Spread-Out LOST (SO-LOST) [5] which is a modified version of LOST, scalability is increased by minimizing mutual spatial inference between robots which is done by shifting crumbs in a trail from source to home and home to source for a single source, single sink.

The Multi-Objective LOST (MO-LOST) [6] tries to reduce mutual spatial inference for paths with multiple sources, multiple sinks by shifting the crumbs of a trail which is crossing other crumbs of the other trails. Blinkered LOST [7] modulates the field of the view (FOV) of the robots' trail-detecting sensor. It shows that the global throughput is a function of robot FOV, means that with narrower FOVs, it performs better in large populations, since narrrower FOV causes multiple trails to be maintained, so that the system

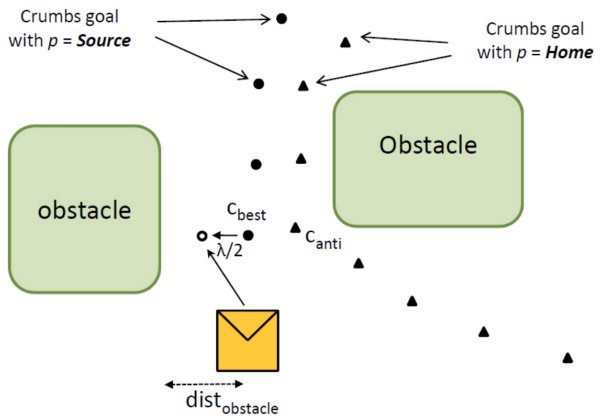


Fig. 2: Trail laying behavior in SO-LOST. The robot is following trail to source (filled circle) and changes its direction to new point (empty circle) to keep distance from trail leading to home (triangles). Obtained from [5]

can support larger population sizes before saturating due to inference.

A. LOCALIZATION-SPACE TRAILS REVIEW

LOST generates trails between the locations of *Events* which in our case is reaching source and reaching home. When an Event occurs to robot, its pose in localization space is recorded as *Place* represented by tuple $[Event, Pose]$. Other robots can interpret the coordinates of a Place in their own frame of reference. To guide robots between these Events, waypoints called *Crumbs* are laid in every S seconds. A Crumb is a four-tuple $[P_c, L_c, d_c, D_c, t_c]$ containing Place name P_c of place of interest, Localization-space pose L_c, estimated distance (in some distance function) between P_c and L_c, and t_c the Crumb creation time. A series of Crumbs that lead to the place P_c is referred to as a $Trail$. A robot broadcasts its Trail when an Event occurs to it (i.e. when it reaches source or home). All trails are periodically scanned and Crumbs with timestamp older than age threshold a seconds are discarded. This is analogous to ant's pheromone trails evaporating over time.

A robot follows trail to a destination place P_g by heading towards the best crumb C_L within its *field of view*, i.e. within radius d_f of its current position. We use the modified version of this algorithm SO-LOST which lays crumbs certain distance λ away from crumbs leading to a different place. Figure 2 shows the crumb laying behavior of the SO-LOST algorithm.

III. CONTRIBUTION

The main contribution of this paper is modification of LOST algorithm to have a simultaneous task allocation which performs better in multiple-source, single-home setting in both large and small population sizes. Two approaches were explored:

1) Embodied approach: Subtle modification to LOST using trail density as a proxy for congestion which has the emergent effect of maintaining multiple trails leading to different sources
2) Explicit allocation approach: Allocate robots to different sources with a goal of uniform distribution of robots along different trails (with consideration of the distances of the sources from home).

In the following sections, we will describe these approaches with quantitave and qualitative evaluations.

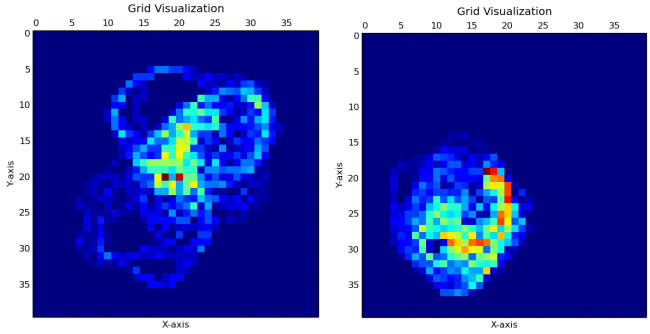


Fig. 3: Grid visualization in embodies approach (left) and original version of SO-LOST (right). The home is at center while the sources are at top-right and bottom-left

IV. EMBODIED APPROACH

This approach maintains trails to multiple sources without keeping track of the number of sources discovered so far. The general workings of this approach is same as that of LOST algorithm where the robots try to follow the best crumb in their immediate neighbourhood. The crumbs are not labelled with source identification and the robots still make local decision without explicitly considering the multiple sources available.

The reason LOST and its other variants converge to a single congested trail is that the robots try to minimize the distance to the trail without considering the congestion in the trail. Even when time to target is using as distance metric instead of physical distance, the final trail is still congested for large population sizes.

The conflict here is whether to follow the shortest path discovered so far or avoid following congested trail potentially leading to the target with shortest distance. To make a better decision, we take both these factors into account and make decision based on both. For this, the distance function is formulated as follows:

$$d = w_{time\_to\_target} \cdot time\_to\_target + w_{crumb\_density} \cdot crumb\_density \quad (1)$$

where $time\_to\_target$ is the estimated time to target encoded in the crumbs and $crumb\_density$ local density of trail around a crumb (to be discussed shortly).

The best values of $w_{time\_to\_target}$ and $w_{crumb\_density}$ were found to be 0.3 and 0.7 respectively from experiments[1]. Note

[1]The scaling of the weights does not affect the performance. So experiments were done with weights that summed to one

that when $w_{crumb\_density}$ is set to 0, the algorithm is the same as SO-LOST.

This method not only maintains trails leading to multiple sources, each trail isn't just a single curve of crumbs (Figure 3). Hence, there are multiple side trails that lead to the same source.

The main part of our approach is outined in Algorithm 1

Algorithm 1 Embodied Approach

procedure DIST(C)
return $w_{d_c} \cdot c.d_c + w_{density} \cdot c.density$

Θ = all the crumbs in the robot's FOV
$\Sigma = \{c | c \in \Theta \ \Lambda \ (c.p_c = robot.goal)\}$
$\Pi = \{c | c \in \Theta \ \Lambda \ (c.p_c \neq robot.goal)\}$

$\lambda = Min(crumb\_avoid, \ dist\_obstacle)$

$c_{best} = c$ s.t. $(c \in \Sigma) \ \Lambda \ (\nexists c' \in \Sigma$ s.t. $DIST(c) > DIST(c'))$

if $(\exists c_{anti} \in \Pi$ s.t. $dist(c_{anti}, robot) < crumb\_avoid)$ **then**
Shift direction λ orthogonal to current trail
else
Follow the c_{best}

A. CRUMB DENSITY IMPLEMENTATION

For finding crumb density, we implemented an additional data structure which is a 2D grid that spans the simulation world. Each cell of the grid contains the number of crumbs within it. Hence, the grid represent a histogram of crumbs. The $crumb\_density$ around a crumb is then the number of crumbs in the cell it belongs to divided by total number of crumbs present. We used cells of size 0.25 x 0.25 m. This is approximation of a robot's size. Figure 3 shows a visualization of our grid data structure. A brighter value represents higher number of crumbs. The fact that SO-LOST implementation converges to a single source is clearly visible while our approach maintains trails leading to both the sources.

B. DRAWBACKS OF THIS APPROACH

A major drawbacks of this approach is that there are a lot of parameters that need to be handtuned. These parameters are evaporation rate of crumbs, grid size and weights of density and time to target in distance function. Even though we were able to get moderate invariance to size of the world, number of robots, number of sources and distances of the sources, these parameters still need fine tuning when used in drastically different setting.

A future extension would be to change these parameters dynamically. For example, we can set evaporation rate based on distance between source and home such that multiple cycles on the same trail does not give a false impression of congestion. We would also like to note that the need to finetune the evaporation rate on different distances between source and home is not inherent to our approach and almost all ant-trail based algorithms would require this. But performance our method is particularly sensitive to this parameter.

Also, there is no guarantee of convergence to multiple sources. If by chance all the robots discover the same source, there will only be a single trail leading to that source (although this is highly unlikely if we have large number of robot).

V. EXPLICIT ALLOCATION APPROACH

This method was explored to mitigate the shortcomings of our previous approach. Using crumb density as a heuristic of congestion introduces unwanted dependence on evaporation rate and trail length as discussed in subsection IV-B. This approach, instead of using approximation, utilizes the actual congestion in different trails. It also modifies the crumb data structure to include the source the crumb is leading to and the total Euclidean Distance from that crumb to the target, so the new crumb data structure with respect to its definition in the LOST review section is a tuple $C = [P_c, S_c, L_c, d_c, D_c, t_c]$ which contains the name of the Place P_c to which it lead to (source or sink), Source Id that the crumb is leading to S_c (when P_c is source), a localization space pose L_c, an estimate d_c of the distance from L_c to the target P_c which comes from a distance function that assume time as distance in the original version of LOST, an estimate of total Euclidean Distance from the crumb to the target D_cs, and the time t_c containing the creation time of the crumb.

We also maintain a global data structure to keep track of numbers of robot currently following different sources. It is implemented as a hash table with all the discovered sources as its key and the number of robots in trail leading to a source as its value. This structure is updated every time a robot reaches home and is allocated to a source.

When a robot reaches home, it checks congestion in every trail. We formulation our *congestion_factor* as follows:

$$congestion\_factor = \frac{(R_{SIZE} + D_{SAFE}) \cdot n_t}{d_t} \quad (2)$$

where R_{SIZE} is the size of robot (diameter) in meters, D_{SAFE} is a safe distance (meters) we need to maintain between the robots, n_t is the number of robots currently in that trail and d_t is the length of the trail (meters)

The ideal $congestion\_factor$ is 1 where the robots form a tight loop in the trail with a safety distance D_{SAFE} between each other. Whenever a robot reaches home, it calculates $congestion\_factor$ for each source and decides to choose the source with minimum $congestion\_factor$ that is less than 1. If no trail is available that has $congestion\_factor$ less than 1, the robot decides not to follow any trail and moves randomly to find other food sources. After choosing the source (or deciding to explore randomly), the hash table is updated if needed.

Although this method maintains trail leading to multiple sources and increases our throughput, in the case of having

paths that are almost equally congested, robots can switch back and forth between sources. We introduce an *inertia* factor for convergence of robots to a particular source; a robot decides to switch from one source to another only if the $congestion\_factor$ is better than that of previous source by a fixed threshold. Our method is outlines in Algorithm 2.

This approach will have a competitive performance even in single source setting as the trail between source and home is maintained such that there is minimum change of interference between the robots. Experimental results showing this will be provided in the following section.

In our implementation, we do not restrict the wandering robots that were not assigned to any known source to cross existing trails. A better performance will likely be achieved by having such restrictions.

Algorithm 2 Explicit Allocation Approach

Congestion factor of a source s
procedure $C_f(s)$
  $\Sigma =$ all the crumbs
  $c_{farthest} = c$ s.t. $(c \in \Sigma)\ \Lambda\ (c.Sc = s)\ \Lambda$
  $(\nexists c' s.t. dist(c', home) < dist(c, home))$

  return $\dfrac{(R_{SIZE} + D_{SAFE}) s.num\_robots}{c_{farthest}.d_t}$

$\Theta =$ Known sources
$s_{prev} =$ previous source the robot was following
$s_{best} = s$ s.t. $(s \in \Theta)\ \Lambda\ (\nexists s' \in \Theta$ s.t. $C_f(s') < C_f(s))$

if $(s \neq \phi)$ **then**
  if $(C_f(s_{best}) - C_f(s_{prev}) < threshold)$ **then**
    $s_{chosen} = s_{best}$
  else
    $s_{chosen} = s_{previous}$
else
  Random move

A. POSSIBLE MODIFICATIONS FOR DISTRIBUTEDNESS

In the previous section we mentioned a new data structure that is used to keep number of robots in each path, which makes this approach centralized. The following defintion of crumbs is suggested: $C = [P_c, S_c, L_c, d_c, D_c, t_c, R_{UDID}]$ in which we have a new information containing a random constant unique identity for each robot that is generated when they start working. In this case, instead of having a hash table which stores the number of robots in each path, each robot can read crumbs and see how many unique robots are in this path. However, persistence of the crumbs still depend on the evaporation rate and the length of the trail. In order to solve this issue, since the path converges after a while, each robot knows an estimate of traveling time, so they can count the number of robots based on the crumbs where the creation timestamp is greater than the average traveling time.

Another way of making this approach distributed is having an agent at home which is incharge of maintaining the hash table and relays the information when a robot reaches home, based on which the robot decides the source to pursue. For this, we can either place a server like setup near home or arbitrarily have one of the robots stay at home to assume the role of conductor.

VI. EXPERIMENTS

A. SIMULATION SETUP

We used Stage simulator [8] to implement SO-LOST, Embodied Approach and Explicit Allocation Approach. We tested in the empty world of size 10x10. Stage's Irobot Roomba robots were used with SICK LMS200 laser rangefinder. The middle green square is the home, the red square is source 1, the blue square is source 2 and the yellow square is source 3. Red robots are carrying resources from source 1, magenta robots are carrying resources from source 2 and the cyan robots are carrying resources from source 3. Robots that are not carrying any resource are colored blue. Each trial was run for 10 minutes and the total number of resources delivered at the end of the trial is our performance metric. Due to stochastic nature of all the methods, we run 5 trials for each setting.

The experiments we conducted used the following parameter settings:

Parameter	Embodied	Explicit	SO-LOST
$Evaporation\ rate$	10Hz	10Hz	10Hz
d_f	1.5m	1.5m	1.5m
S	.2s	.2s	.2s
$Minimum\ Trail\ Distance$	0.75m	0.75m	0.75m
$Grid\ Cell\ Size$	0.25	N/A	N/A
$W_{crumb\_density}$	0.7	N/A	0.0
$W_{time\_to\_target}$	0.3	N/A	1.0
$Inertia\ Factor$	N/A	0.3	N/A
$Max\ Congestion\ Factor$	N/A	1.2	N/A

TABLE I: Parameters Settings for Experiments

Our source code is available at [2].

B. RESULTS AND DISCUSSION

In figures 5, 6 and 7, we show the screenshots of three algorithms Explicit Allocation Approach, Embodied Approach and SO-LOST with 1, 2 and 3 sources respectively (with 16 robots). This gives us qualitative evaluation of our these three algorithms.

Figure 4 shows the performance of the three algorithms in terms of total collected resources at the end of 10 minutes of simulation time with respect to population size. All of the methods scale in similar manner with the population size. In the case of only one source, SO-LOST performs slightly better than both our approaches since in our methods some robots do not follow trail to known source to find other sources (which are not present). With 2 sources, both of our methods perform better with larger population size as they exploit both the sources and have less congested trails. We see the same trend with 3 sources.

[2] https://github.com/fshamshirdar/moorche

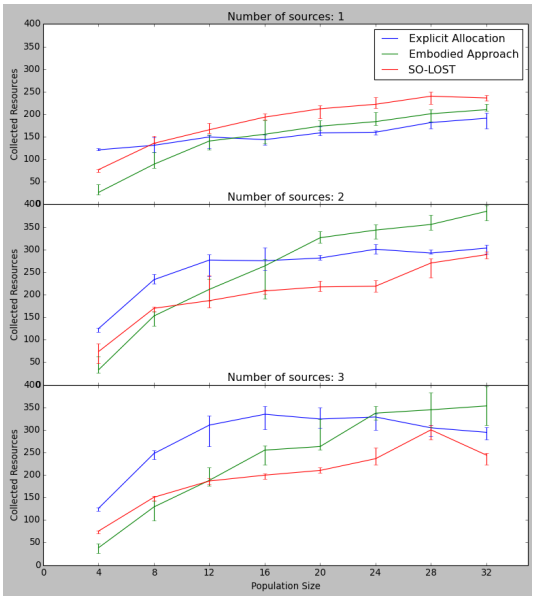


Fig. 4: Comparison in performance between Explicit Allocation, Embodied and SO-LOST algorithms

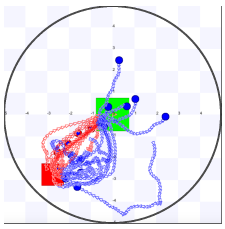
Comparing Explicit Allocation and Embodied Approach, we find that the performance of Embodied Approach is superior at higher number of robots. This can be attributed to the fact that increasing the number of robots, more robots won't be assigned to any source and left to wander. But these wandering robots eventually interfere with the robots in existing trails and degrade the performance.

The performance of Embodied Approach is highly dependent on values of parameters (especially the evaporation rate and distance between source and home). It has higher variance in performance compared to other methods. Also, the fact that it performs significantly better in larger population size (except when there is only one source) and consistently lower in smaller population size compared to other approaches can be partly explained by the parameters setting.

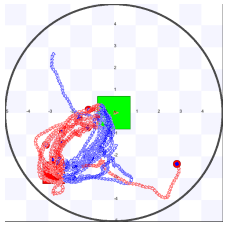
VII. CONCLUSIONS AND FUTURE WORKS

In this paper, we outlined two approaches for improving LOST algorithm for better throughput in a multiple-source, single-home setting. We have provided experimental results showing improved performance compared to a variant of LOST (SO-LOST) both quantitavely and qualititavely. The robots take into account the congestion in the trails either explicitly or implicitly to make better decisions regarding which trail to follow.

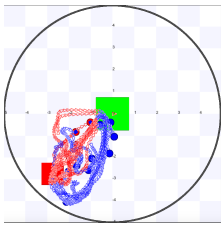
In our future works we will implement a fully distributed version of our algorithm, potentially with real robots. Another extension can be implementing Maximum Sustainable Yield (MSY) [9] where the sources have a finite (and possibly different) resource growth rate. Previous works in multi-robot MSY foraging [10] and [11] rely on known source position. We can extend our current work to deal with simultaneous source finding and MSY in multiple-souce, single-home setting. We also plan to improve our navigation strategies. There are lot of room for improvement in our obstacle avoidance and trail-following behavior which could lead to better performance.



(a) Explicit Allocation Approach



(b) Embodied Approach



(c) SO-LOST

Fig. 5: Screenshot after 10 minutes of simulation time: single source, 16 robots

REFERENCES

[1] R. Vaughan, K. Støy, A. Howard, G. S. Sukhatme, and M. J. Matarić, "Lost: Localization-space trails for robot teams," *IEEE Transactions on Robotics and Autonomous Systems*, vol. 18, no. 5, pp. 796–812, 2002.

[2] M. Dorigo, "Optimization, learning and natural algorithms," *Ph. D. Thesis, Politecnico di Milano, Italy*, 1992.

[3] A. Russell, D. Thiel, and A. Mackay-Sim, "Sensing odour trails for mobile robot navigation," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, May 1994, pp. 2672–2677 vol.3.

[4] D. W. Payton, M. J. Daily, B. Hoff, M. D. Howard, and C. L. Lee, "Pheromone robotics," in *Intelligent Systems and Smart Manufacturing*. International Society for Optics and Photonics, 2001, pp. 67–75.

[5] A. Sadat and R. Vaughan, "SO-LOST: An ant-trail algorithm for multi-robot navigation with active interference reduction," in *Proceedings of the Twelfth International Conference on Artificial Life (ALife XII)*, August 2010.

[6] Z. Song, S. A. Sadat, and R. Vaughan, "MO-LOST: Adaptive ant trail untangling in multi-objective multi-colony robot foraging," in *Proceedings of the Eleventh International Conference on Autonomous Agents and Multiagent systems*, Valencia, Spain, June 2012.

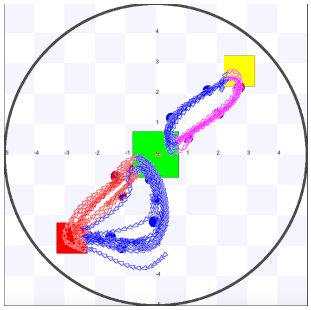
[7] A. Sadat and R. Vaughan, "Blinkered lost: Restricting sensor field of view can improve scalability in emergent multi-robot trail following," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2010.

[8] R. Vaughan, "Massively multi-robot simulations in stage," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 189–208, December 2008.

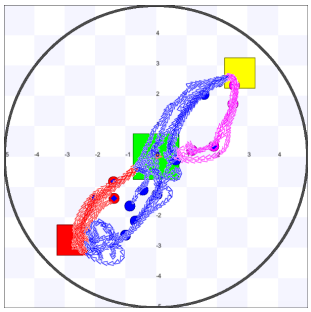
[9] J. Hjort, G. Jahn, and P. Ottestad, "The optimum catch," *I kommisjon hos J. Dybwad*, 1933.

[10] Z. Song and R. Vaughan, "Sustainable robot foraging: adaptive fine-grained multi-robot task allocation for maximum sustainable yield of biological resources," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS'13)*, Tokyo, Japan, November 2013.

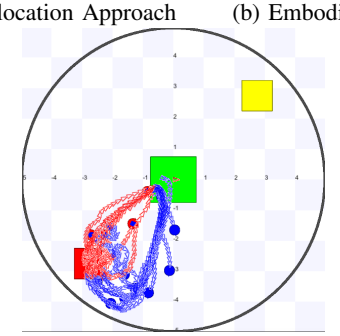
[11] R. Zhang and Z. Song, "Maximum sustainable yield problem for robot foraging and construction system," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, 2013.



(a) Explicit Allocation Approach

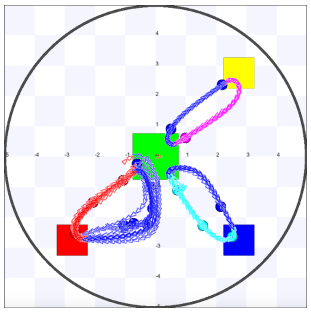


(b) Embodied Approach

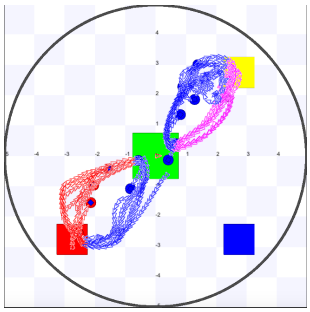


(c) SO-LOST

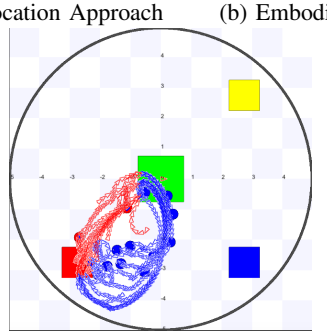
Fig. 6: Screenshot after 10 minutes of simulation time: 2 sources, 16 robots



(a) Explicit Allocation Approach



(b) Embodied Approach



(c) SO-LOST

Fig. 7: Screenshot after 10 minutes of simulation time: 3 sources, 16 robots