Distributed Path Planning for Collective Transport Using Homogeneous Multi-Robot Systems

Golnaz Habibi, William Xie, Mathew Jellins, James McLurkin

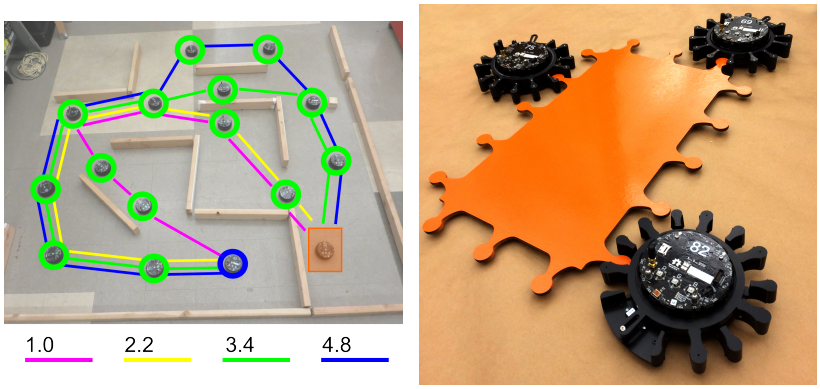
Abstract We present a scalable distributed path planning algorithm for transporting a large object through an unknown environment using a group of homogeneous robots. The path is optimal given the sampling of the robots and user input parameters. The robots are randomly scattered across the terrain and collectively sample the environment in a distributed fashion. Using the dimensions of the payload, the robots first construct a configuration space. With a variant of the distributed Bellman-Ford algorithm, we then construct a shortest-path tree using a custom cost function from the goal location to all other connected robots. The cost function encompasses the work required to rotate and translate the object in addition to an extra control penalty to navigate close to obstacles. Our approach sets up a framework that allows the user to to balance the trade-off between the safety of the path and the mechanical work required to move the object. We implemented our algorithm in both simulated and real-world environments. Our approach is robust to the size and shape of the object and adapts to dynamic environments.

1 Introduction

Object transportation is a well-studied problem in the multi-robot domain. Multi-robot system offers a scalable solution to move a large object traditionally required a much more sophisticated robot to move without collaboration. This problem is especially challenging when traversing through an unknown environment where the path is undefined. It is also important to avoid moving obstacles and adapt to changing environments in the process. For the object manipulation part of the problem, researchers had experimented with an ant-inspired approach [1], granular con-

Golnaz Habibi, William Xie, James McLurkin
Rice Univ., Houston, TX e-mail: {gh4, wxie, jmclurkin}@rice.edu

Mathew Jellins
Purdue Univ., West Lafayette, IN e-mail: mjellins@purdue.edu



(a) (b)

Fig. 1: **(a)** An environment is sampled by 16 r-one robots. Our path planning algorithm finds optimal paths given the sampling of robots and user parameters to transport the object (orange) to the goal location (blue). The different paths in various colors are generated based on different control costs when navigating through in narrow corridors. **(b)** Multiple r-one robots with engaged manipulator attachment can translate and rotate an object.

vection [2], attractor dynamics [3], and quadrotors [4] to collectively transport an object using multiple robots. Obstacle avoidance and path planning for collective transport have been done through using a potential-field based approach [5], leader-follower [6], reinforced learning [7], and localization using imagery [8]. In previous work, collective transport has been studied for objects with different shapes and sizes [1]. However, the study did not include obstacles during the transportation. Li et al., [9] used stationary sensor nodes and a distributed algorithm to provide a obstacle avoiding path. O'Hara and Balch [10] also used distributed network to plan paths for environment with dynamic obstacles. Neither incorporates the dimension, shape, or orientation of the payload and the sensors used are immobile. This could lead to collision during the actual transportation.

The goal of this paper is to develop and test a scalable path planning algorithm for transporting a large object, one that cannot easily be transported by a single robot, using a multi-robot system. We are interested to solve the path planning problem in environments larger than the robots communication range. Additionally, the robots have no prior knowledge of the environment or other global information, mimicking real-world situations where services such as global positioning system are not readily available or are too expensive to implement. These robots are scattered randomly, and perhaps non-uniformly, to detect obstacles in the environment. Therefore, we design a *distributed* path planning algorithm. Using local obstacle information, robot distributively construct a configuration space for the environment in a distributed fashion. With a robot near the goal position as the root, a shortest-path tree based on a custom cost function is constructed spanning all traversable

positions. The cost for the tree takes into account of the amount of mechanical work required to transport the object and the additional control cost to navigate in constricted, less safe areas. The resulting tree gives an optimal path with respect to the sampling of the environment from every possible starting position to the goal. The path-planning algorithm is self-stabilizing and runs continuously to reconstruct the tree as the environment changes. This makes the algorithm robust to dynamic obstacles and changing robot population.

Our path planning algorithm is practical on real robots. We adopt the r-one platform as our main environment. The r-one is a low-cost robot designed for research and teaching [11, 12]. It has a large sensor suite and is capable of motion and communicating with other r-ones. We plan to develop distributed controllers and algorithms based on the r-one platform to form a complete object transport solution. We have designed and built manipulator attachments for the r-ones to control and transport objects as shown in Figure 1b [12]. Using this attachment, multiple r-ones can attach, rotate, and translate an object collectively. The methods of utilizing this attachment for object manipulation will be discussed in future work. This paper focuses on the path planning aspect in an unknown environment.

2 Model and Assumptions

We assume that the multi-robot network starts in a dispersed, but connected state. The network is modeled as a undirected unit disk graph $G = (V, E)$ [13]. Each robot in the network used to construct the configuration space is represented by a node, $u \in V$, where V is the set of all robots and E is the set of all robot-to-robot communication links. The neighbors of each robot $u \in V$ are the set of robots with line-of-sight connections and within communication range r from robot u.

Each robot is at the origin of its local coordinate system. The x-axis is aligned with the robot's current heading. Each robot can measure the relative orientation between its heading and its neighbors' headings along with the distance from each of its neighbors and obstacles. Each robot has knowledge of the dimension of the object being transported.

Each robot is modeled as a small disk with a pose defined by $u_{pose} = (x, y, \theta)$. The robots have a differential drives that allow them to rotate in place or translate. The algorithm executes in a series of synchronous *rounds*. With a synchronizer, we can model an asynchronous distributed system as a synchronous one. This simplifies analysis and is straightforward to implement in a physical system.

The object is transported by identical robots. The object is assumed to be rigid and can be in any shape. Our model treats the object as a rectangular bounding box fully enclosing the original shape and robots as illustrated in Figure 2. We assume that the object can translate in any direction but rotate only with respect to the intersection of the enclosing bounding box's diagonals, the center of the rectangle. Preliminary results in distributed controllers support this assumption (see video attachment of [12]). Future work will focus on these controllers. We define two

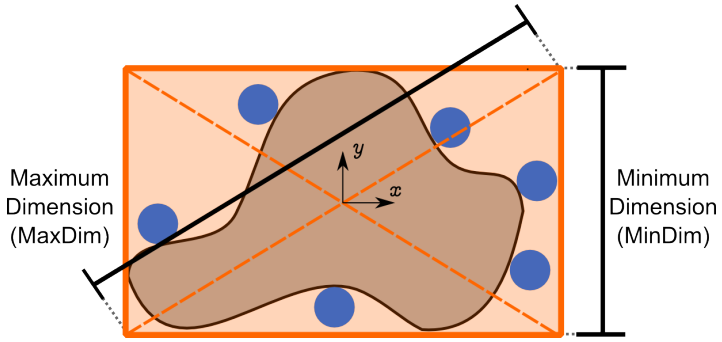


Fig. 2: The object and the transporter robots are modeled as an fully enclosing rectangular box with the center of rectangle as the axis of rotation. MaxDim is the maximum dimension of the object during rotation. MinDim is the minimum dimension of the object.

measures for the object: *MaxDim* and *MinDim*. They are the maximum and minimum dimensions of the simplified object. The object is also at the origin of its local coordinate system. The x-axis is orthogonal to the side of MinDim.

3 Distributed Path Planning

We assume there is a disperse but connected networks of robots. The robots covered the environment by using an expansion algorithm [**?**]. There are also two distinguished robots as *start* and *goal* robots. We aim to generate a path among the network of robots from start robot to the goal robot. Our path planning algorithm executes in three phases. First, the robots build a configuration space in a distributed fashion. Then, they construct a shortest-path tree using a variant of the distributed Bellman-Ford algorithm from the goal robot to every other connected robot in the network. Lastly, we select the starting location to transport the object.

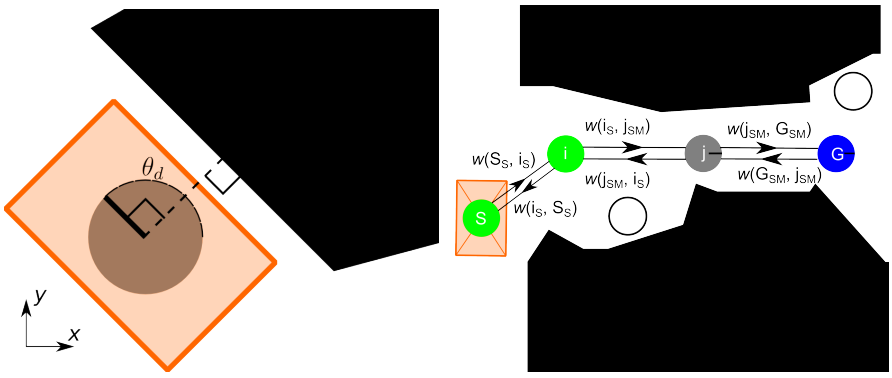
3.1 Configuration Space

We measure the safety of a location based on the likelihood of the object colliding with an obstacle when it's centered at that location. The robots independently sample the environment and construct a configuration space based on the safety classifications of their locations. The configuration space is categorized into three types of safety regions: safe (**S**), safe-minor (**SM**), and unsafe (**U**). When the object is at a **S** region, the object can rotate freely. In **SM** regions, the object has a limited range of orientation that it can assume without colliding. In **U** regions, the object collides

with obstacles in any orientation. Each robot categorizes itself into one of the three safety classifications using the center of the robot as the center of the object. Each robot also has a desired orientation θ_d for the object to align its local x-axis to when the object is at the robot's location to avoid collision. The classification $C(x,\ y)$ and desired orientation $\theta_d(x,\ y)$ for a robot at global coordinates $(x,\ y)$ are defined according to the following conditions:

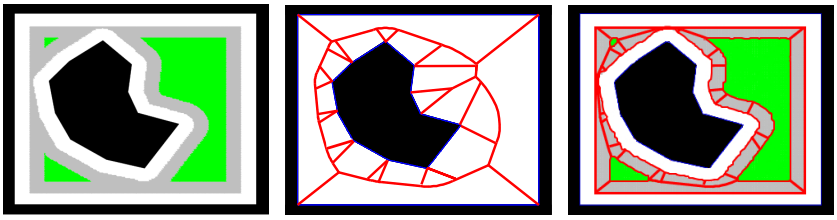
$$\begin{array}{llcll} \textbf{if} & MinDim < d_{min} & \Rightarrow & C(x,\ y) = \textbf{S}, & \theta_d(x,\ y) = \textbf{undefined} \\ \textbf{if} & MinDim \leq d_{min} < MaxDim & \Rightarrow & C(x,\ y) = \textbf{SM}, & \theta_d(x,\ y) = f(x,y) \\ \textbf{if} & d_{min} < MaxDim & \Rightarrow & C(x,\ y) = \textbf{U}, & \theta_d(x,\ y) = \textbf{undefined} \end{array}$$

Where $d_{min}(x,\ y)$ is the minimum distance from the robot's global coordinate $(x,\ y)$ to the nearest obstacle in the environment. θ_d is undefined for **S** and **US** regions. In **S** regions the object can rotate freely with no risk for collision. There is no need for a specific orientation. Similarly, in **U** regions the object collides in any orientation so θ_d is also undefined. However, in **SM** regions, the orientation of the object is constrained to a range of angles. We define $f(x,\ y)$ as a function that computes θ_d in **SM** regions. To be conservative, we limit θ_d to be orthogonal to the normal of the nearest wall as shown in Figure 3a. Orienting the object at this angle maximizes the distance between the object and the wall by making the MinDim side of the bounding box parallel to the normal of the wall. There are two possible angles at each **SM** location that satisfy this constraint. We select the θ_d that minimizes the bearing of the current node to the next hop along the minimum cost path.



(a) (b)

Fig. 3: **(a)** θ_d of **SM** robots is orthogonal to the normal of the nearest obstacle. **(b)** The bi-connected weighted direct graph model for the robot network. Robots in **S**, **SM**, and **U** are shown in green, gray, and white respectively. Costs are modeled as the weight of each corresponding edge.



(a) Safety classification (b) Voronoi diagram (c) Segmented **SM** regions

Fig. 4: **(a)** The configuration space constructed by an infinite number of robots. The locus of **S**, **SM**, and **U** points are colored green, gray, and white respectively. The wall and obstacles are colored black. **(b)** Voronoi tessellation separates the configuration space into regions based on the location of the closest obstacle edge or vertex. **(b)** Combining configuration space and Voronoi tessellation, we split the **SM** regions into sections with the same θ_d.

Figure 4 shows an environment sampled by a large number of infinitesimally small robots achieving an infinite resolution configuration space. The configuration space can be segmented using Voronoi tessellation [14, 15] into regions that share the same closest obstacle—either an edge or a vertex. Combining the two, we can visualize the SM regions that have the same θ_d. The individual robots have no global information about the environment and thus cannot rely on this method. Instead, each robot finds the closest obstacle edge or vertex, classifies its safety, and sets θ_d individually. Using distance and bearing information from its neighbors, each robot can localize its neighbors and determine their θ_d in its own coordinate system.

In addition to configuration space safety classifications, we introduce *safety factor* γ to quantify safety at a higher precision:

$$\gamma = \frac{d_{min}(x,\, y)}{MinDim/\, 2} \tag{1}$$

where $MinDim/\, 2$ is the shortest distance from the object's center to the object's boundary. The greater γ, the safer the location. In S and SM regions $\gamma \geq 1$.

3.2 Distributed Bellman-Ford Algorithm

The cost of an edge $w(u,\, v)$ is calculated based on the mechanical work required to translate and rotate the object from node u to neighbor node v. Note that $w(u,\, v)$ and $w(v,\, u)$ are not necessarily the same due to the bi-connected property of the graph (Figure 3b). This happens when translating between **S** and **SM** locations. The cost is defined by the following equation:

$$w(u,\, v) \;=\; \alpha F d(u,\, v) + \tau \theta_{\Delta} \tag{2}$$

where $d(u,\ v)$ is distance between nodes u and v. θ_Δ is the change in angle for the object to line up its local x-axis to the desired heading θ_d. F and τ are constants of force and torque that scale $d(u,\ v)$ and θ_Δ to the mechanical work required to translate and rotate the object respectively. They depend on the dimensions and weight of the object. α is the *minor translation control multiplier*, an additional weight for extra control cost when the object is being transported in confined regions (**SM** locations). α is an user-defined value and is ≥ 1 when moving to a **SM** region where the object's motion needs to be controlled more tightly to avoid collision. A rotation cost is also imposed to orient the object to θ_d. When moving to a **S** region, since the object can rotate freely without collision, the α is 1 and rotation cost is 0. We use the worst case θ_Δ (change of object's orientation) of $\frac{\pi}{2}$ when moving from a **S** region to a **SM** region's θ_d. The cost function for all different safety region transitions is summarized by Table 1.

Table 1: Translation and rotation cost for different safety regions

Edge type	Translation cost	Rotation cost
S→ S	$Fd(u,\ v)$	0
SM→ S	$Fd(u,\ v)$	0
S→ SM	$\alpha Fd(u,\ v)$	$\tau\frac{\pi}{2}$
SM→ SM	$\alpha Fd(u,\ v)$	$\tau\theta_\Delta$

To quantify the efficiency of the generated path, we define *path efficiency* as the ratio of the shortest path length to the planned path length. We use this ratio to evaluate the trade-off between distance and safety.

$$\psi = \frac{P_{shortest}}{P_{planned}} \tag{3}$$

where $P_{shortest}$ is the sum of all edges for the shortest distance path. $P_{planned}$ is the sum of all edges for the path planned by our algorithm.

We use a variant of the distributed Bellman-Ford Algorithm [16, 17] to build the shortest-path tree using the above cost function to find a minimum cost path from any **S** or **SM** connected robot in the network to the goal position. **U** robots are not traversable locations and are excluded in the tree. We assume that there is a sufficiently large population of robots that are scattered to sample the environment and there is at least one safe path from the starting location to the goal. The algorithm starts building a shortest path tree using the goal robot as the root. For every communication round, each robot calculates the shortest-path to goal (starts from 0 at goal) and records the associated next hop using local information. It then broadcasts its distance to neighbor, relative orientation, and minimum cost to goal to its neighbors. The resulting tree contains the minimum cost paths from any starting location to the goal.

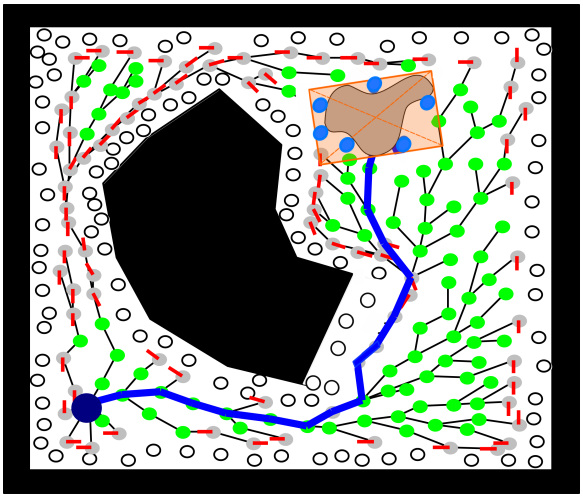


Fig. 5: The optimal path (blue) is found using a variant of the distributed Bellman-Ford algorithm for a simulated environment sampled by 275 robots. The goal robot is indicated by a large blue circle. All other robots are designated by small circles and are classified as S (green), SM (gray), and U (white). Red lines on SM robots denote the desired orientation for the object at that location.

The algorithm is self-stabilizing and allows change in network topology. Additional robots introduced to the environment are quickly integrated into the network. Figure 5 shows the result of our algorithm in simulated environment.

3.3 Transporting the Object

After the minimum-cost paths are generated, the object starts from any location on the tree. The initial orientation of the object is first determined by setting the object's local x-axis to one that minimizes the bearing to the next hop robot's location. The object then follows the path. When the object moves to a **SM** location, it aligns its x-axis to the robot's θ_d or $\theta_d + \pi$. Each robot knows the object's desired orientation at that location, and the location and distance to the next robot along the path. A distributed controller will only have access to this information to move the object. The prescriptions of the controller is outside of the scope of this work, but preliminary results show that translation, rotation, and combined controllers can successfully transport an object with multiple robots. Physical interference from the robots will be a significant problem remained to be solved in future work.

4 Experimental Results

We tested our path planning algorithm in a simulated environment. Figure 6 shows a complex environment sampled by a large number of robots. Using our algorithm, different paths are generated by altering the minor translation control multiplier α. From right to left, the images show paths with increasing focus on safety over minimizing the distance (decreasing α). The result shows the path becoming shorter by including more SM robots. Path efficiency ψ and safety factor γ for the three paths are plotted in Figure 7. Path efficiency decreases and the safety factor increases with increasing α. By adjusting α, we can balance the trade-off between safety and overall distance traveled by altering alpha. The path efficiency directly depends on the number of the samples(robots). Figure 7 shows the effect of degree of the network on path efficiency. By increasing the degree of the network, which implies network size, the path efficiency approaches to the unity. For the degree larger than 12 , the environment is sampled enough so that our algorithm generates the path close to the shortest path. For careful analytic of this behavior see the work by Kleinrock [**?**]. The population with degree less than 5 is disconnected and the algorithm cannot find any path from start to the goal and path efficiency is zero.

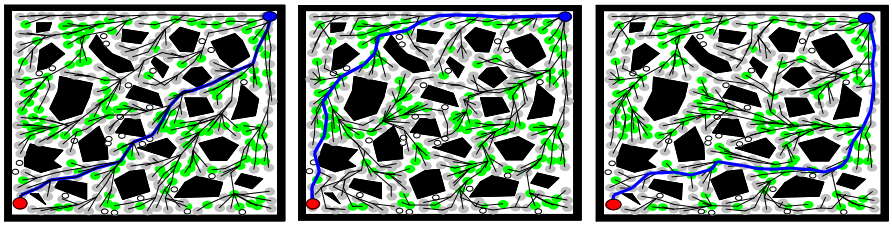


Fig. 6: Simulation of the path planning algorithm for an environment sampled by 339 planner robots. Robots have safety designation of **S** (green), **SM** (gray), and **U** (white). Start and goal positions for the object are marked red and blue respectively. The blue line shows the planned path found using our algorithm. By altering α in the cost function, the algorithm generates different topology for the shortest-path tree.

We conducted several experiments using the r-one robots. They have knowledge of their pose and their neighbors' in their local coordinate frames. They can communicate with their neighbors through line-of-sight infrared transmitters and sensors (maximum range of 0.375 meters). They can sense nearby obstacles with infrared ranging to find their safety classifications. The **SM** robots orient themselves to their respective θ_d angles based on the nearest obstacle. This aids debugging, simplifies the coordinate systems, and is aesthetically pleasing. For the experiments, the cost function is simplified. We assume each neighboring robots are equal distance from one another and the distance-work scaling factor is 1 ($Fd(u, v) = 1$).

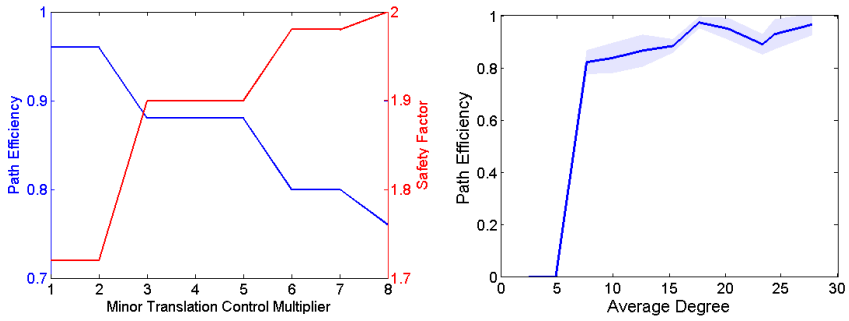


Fig. 7: (left)Plot for path efficiency ψ (blue) and safety factor γ (red) with respect to minor translation control multiplier α for the environment used in the simulation in Fig 6 with 1000 robots. (right) The effect of network degree on the average of path efficiency of algorithm in the environment used in Fig 6, with minor translation control multiplier $\alpha = 0$, the result is for the average path efficiency of 10 networks for each degree.

We tested self stabilization in a dynamic environment by constructing an environment with a "room" and two "doors" that lead to the destination (Figure 8). The doors serve as movable obstacles. When one door is shut, the initial path is blocked. The path planning network finds a different way. Our algorithm first selects the shorter of the two paths. After the environment is altered and the path is destroyed, the algorithm successfully regenerates an alternate path to goal.

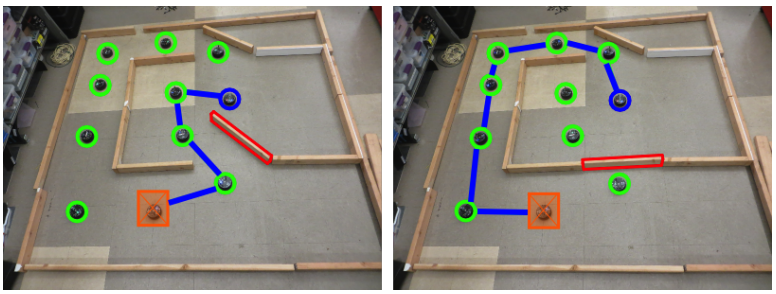


Fig. 8: Experiment shows our algorithm's ability to respond to changing network topology and to regenerate the path when the initial path is broken. The algorithm is flexible and robust to changing environments.

We constructed three larger, more complex environments with multiple paths to get from source to goal. Each environment consists of paths with different length or robot safety classifications. The dimensions of the environments are 2.6 x 2.6 meters. Figure 9 shows the path planning algorithm running with different α values.

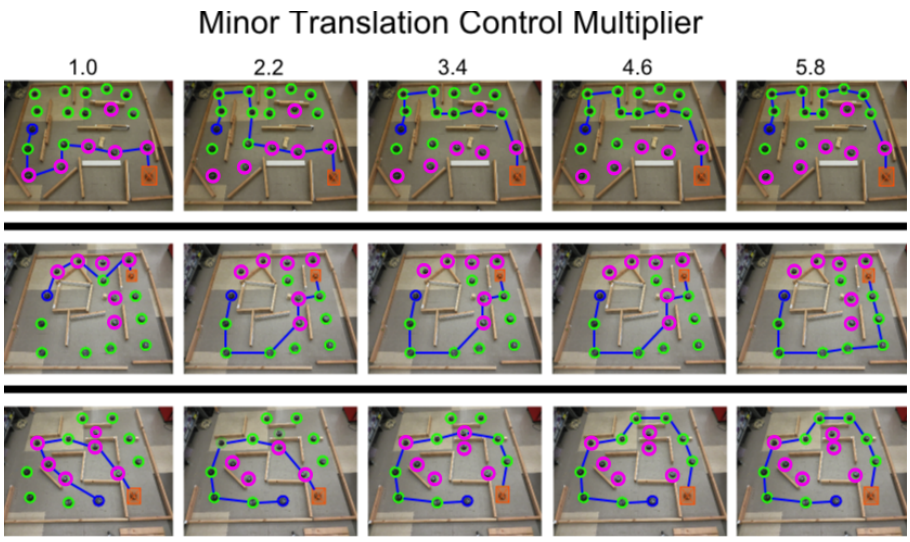


Fig. 9: The path planning algorithm tested on three different complex environments with a varying minor translation control cost α for each. **S** and **SM** robots are labeled green and magenta respectively. Goal robot is blue and starting robot is orange. α increases from left to right. The path changes with increasing α value to include fewer **SM** robots for a safer transport.

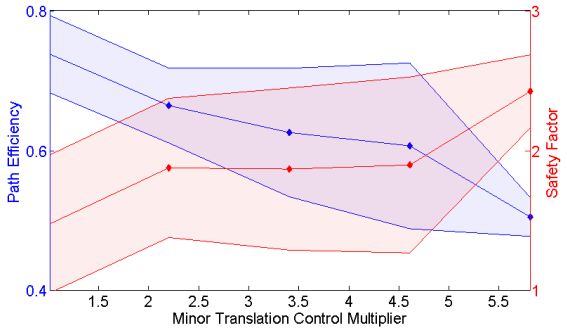


Fig. 10: Path efficiency (blue) and safety factor (red) are plotted against minor translation control cost. The region around each plot shows one standard deviation from the mean for the three tested environments. The data was gained from the experiments shown in Figure 9

By measuring the path distance and the average distance for all robots on the path from the nearest obstacle, we are able to measure the path efficiency, ψ, and the safety factor, γ for each path. The data can be found in Figure 10. Similar to Figure 7, ψ decreases and γ increases with α. The path planning algorithm is robust to changing environments and changing robot network topology. By modifying α,

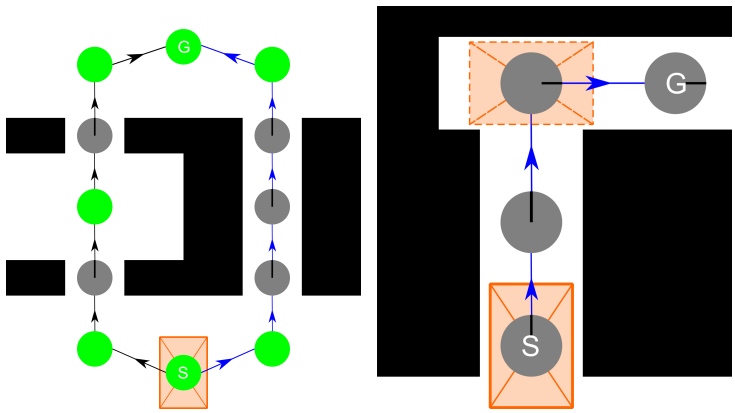
we could use the algorithm to find a safer path in exchange for additional distance traveled. Our algorithm executed in rounds of two seconds. For each round, a 9-byte message is sent to each of the robot's neighbors. Because execution time of the algorithm is much shorter than a round and the distance between two nodes is the same as the number of hops going from one node to the other, our algorithm has a running time complexity of $O(diam(G))$, the diameter of the graph. The number of neighbors per robot for our system is limited to 10. For stability, we update the next hop to goal every three rounds. When the topology of the shortest-path tree changed due to changing environment, it still has the same upper bound of $O(diam(G))$ to reconfigure the tree. The analysis has been confirmed by experimental data.

5 Scope and Limitations

Our models and algorithms have made some simplifications and assumptions to make the path planning computationally feasible. However, they do have limitations and drawbacks. In particular, our path planning algorithm fails to address two types of environments. Figure 11a shows an environment sampled by 10 planner robots. There are two valid paths with equal distance to the goal. Because we used the worst case assumption for object moving from **S** to **SM**, the θ_Δ in the cost function is $\frac{\pi}{2}$. This penalty makes our algorithm selects the right path with more SM robots over the left for small α. In general, paths with large number of S to SM edges would incur a large aggregate rotation cost even when the actual θ_Δ during the transportation is small. One solution to this problem is to include the heading of the object as part of the minimum-cost path calculation. This method requires the algorithm to store the state of the object for each robot. It adds additional complexity to the problem and makes the algorithm computationally taxing. The second scenario where our algorithm fails is illustrated in Figure 11b. Our algorithm omits rotational safety classifications when constructing the configuration space. This results in the shortest-path tree wrongly includes sub-paths that could cause collision due to rotational constraints as part of the tree. This problem could be addressed in future work by adding additional rotational metrics when constructing the configuration space. The path given by the figure should be disconnected due to rotational constraints. Alternatively, the model for the object could be modified into a larger rectangular enclosing box. While this method would not solve the problem completely, it could alleviate the degree of the problem at the expense of efficiency.

6 Conclusion and Future Work

In this paper, we investigated a path planning problem that provided a framework for the user to balance the trade-off between safety and mechanical work required to move the object using a multi-robot system. The cost for safety is captured by



(a) (b)

Fig. 11: **(a)** The path generated by our path planning algorithm is sub-optimal due to the worst-case assumption for S to SM edges. **(b)** Our model fails to address rotational safety constraints which can cause collision on some tightly bounded paths while rotating the object.

the minor translation control multiplier in the cost function applied to paths that put the object near obstacles. We use a large number of homogeneous robots to sample an unknown environment and construct a configuration space based on safety constraints. We then used a variant of the distributed Bellman-Ford algorithm with a custom cost function to find a path for the object to transport from the start to goal location. We implemented our algorithm both in simulated and real-world environments. As the results show, our approach robustly constructs safe paths and adapts to dynamic environments.

The proposed path planning algorithm simplifies many aspects of the transportation. A more extensive study could be done in the future to model the system in greater details. The manipulation and transportation of the object are still left for future investigation although preliminary results are promising. Because we are using physical robots to construct the configuration space and to create a path, some measures need to be taken during the actual transportation of the object to disperse the robots that create the path. The dimensions, shape, and heading of the object should be measured dynamically by the manipulator robots and propagate through the network in future work. In conclusion, our proposed path planning method is the first step for an effective object transportation using a multi-robot system. More research still need to be done in order to utilize it to its full capacity.

References

1. M. Rubenstein, A. Cabrera, J. Werfel, and J. Mclurkin, "Collective Transport of Complex Objects by Simple Robots : Theory and Experiments," *International Conference on Autonomous Agents and Multiagent Systems*, pp. 47–54, 2013.
2. K. Sugawara, N. Correll, and D. Reishus, "Object Transportation by Granular Convection Using Swarm Robots," *Proc. of the 10th International Symposium on Distributed Autonomous Robotic Systems*, pp. 1–13, Nov. 2012.
3. R. Soares, E. Bicho, T. Machado, and W. Erlhagen, "Object transportation by multiple mobile robots controlled by attractor dynamics: theory and implementation," *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 937–944, Oct. 2007.
4. K. Sreenath and V. Kumar, "Dynamics, control and planning for cooperative manipulation of payloads suspended by cables from multiple quadrotor robots," *Robotics: Science and Systems*, Jun. 2013.
5. F. E. Schneider and D. Wildemuth, "A potential field based approach to multi robot formation navigation," *Proc. of the 2003 IEEE International Conference on Robotics, Intelligent Systems and Signal Processing*, vol. 1, pp. 680–685, Oct. 2003.
6. Y. Le, H. Kojima, and K. Matsuda, "Cooperative Obstacle-Avoidance Pushing Transportation of a Planar Object with One Leader and Two Follower Mobile Robots," *Journal of Robotics and Mechatronics*, vol. 17, pp. 77–88, 2004.
7. C. M. Vigorito, "Distributed path planning for mobile robots using a swarm of interacting reinforcement learners," *Proc. of the 6th international joint conference on Autonomous agents and multiagent systems - AAMAS '07*, vol. 5, p. 1, 2007.
8. J. Spletzer, A. K. Das, C. J. Taylor, V. Kumar, and J. P. Ostrowski, "Cooperative Localization and Control for Multi-Robot Manipulation," *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference*, vol. 2, pp. 631–636, Oct. 2001.
9. Q. Li, M. De Rosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," *Proc. of the 9th annual international conference on Mobile computing and networking - MobiCom '03*, p. 313, 2003.
10. K. J. O. O'Hara and T. R. Balch, "Distributed Path Planning for Robots in Dynamic Environments Using a Pervasive Embedded Network," *Proc. of the Thrid International Joint Conference on Autonomous Agent and Multiagent Systems*, pp. 1538–1539, Jul. 2004.
11. J. McLurkin, A. Lynch, S. Rixner, T. Barr, A. Chou, K. Foster, and S. Bilstein, "A Low-Cost Multi-Robot System for Research, Teaching, and Outreach," *Proc. of the Tenth Int. Symp. on Distributed Autonomous Robotic Systems DARS-10*, pp. 597–609, 2013.
12. J. McLurkin, A. McMullen, N. Robbins, A. Chou, W. Li, M. John, C. Licato, N. Okeke, J. Rykowski, S. Kim, G. Habibi, W. Xie, Y. Zhou, J. Shen, N. Chen, Q. Kaseman, A. Becker, L. Langford, J. Hunt, A. Boone, and K. Koch, "A Robot System Design for Low-Cost Multi-Robot Manipulation," *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.
13. B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, no. 1–3, pp. 165 – 177, 1990.
14. S. P. Fekete, T. Kamphans, A. Kröller, J. S. B. Mitchell, and C. Schmidt, "Exploring and triangulating a region by a swarm of robots," in *14th International Workshop, 2011, and 15th International Workshop, 2011, Princeton, NJ, USA*, 2011, pp. 206–217.
15. N. Mayya and V. T. Rajan, "Voronoi diagrams of polygons: A framework for shape representation," *Journal of Mathematical Imaging and Vision*, vol. 6, no. 4, pp. 355–378, Dec. 1996.
16. R. Fabbri and L. F. Estrozi, "On Voronoi Diagrams and Medial Axes," *Journal of Mathematical Imaging and Vision*, vol. 17, pp. 27–40, 2002.
17. L. Ford, D. Fulkerson, and R. Bland, *Flows in Networks*, ser. Princeton Landmarks in Mathematics. Princeton University Press, 2010.
18. C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Aceves, "A loop-free extended Bellman-Ford routing protocol without bouncing effect," *SIGCOMM '89 Symposium proceedings on Communications architectures and protocols*, vol. 19, pp. 224–236, 1989.

19. L. Kleinrock and J. Silvester, “Optimum transmission radii for packet radio networks or why six is a magic number,” in *Conference Record, National Telecommunications Conference*, Birmingham, Alabama, December 1978, pp. 4.3.2–4.3.5.