Construction of Optimal Control Graphs in Multi-Robot Systems

Gal A. Kaminka and Ilan Lupu and Noa Agmon

Abstract Control graphs are used in multi-robot systems to maintain information about which robot senses another robot, and at what position. Control graphs allow robots to localize relative to others, and maintain stable formations. Previous work makes two critical assumptions. First, it assumes edge weights of control graphs are deterministic scalars, while in reality they represent complex stochastic factors. Second, it assumes that a single robot is pre-determined to serve as the global anchor for the robots' relative estimates. However, optimal selection of this robot is an open problem. In this work, we address these two issues. We show that existing work may be recast as graph-theoretic algorithms inducing control graphs for more general representation of the sensing capabilities of robots. We then formulate the problem of optimal selection of an anchor, and present a centralized algorithm for solving it. We evaluate use of these algorithm on physical and simulated robots and show they very significantly improve on existing work.

1 Introduction

Control graphs are used in multi-robot systems to maintain information about which robot senses another robot, and at what position. In such control graphs, nodes represent robots in given positions. Weighted edges represent sensing capabilities; an edge from node A to node B, with weight w, represents the fact that robot A can sense robot B, with preference w (typically, smaller weight indicates stronger pref-

Gal A. Kaminka
Computer Science Department and Gonda Brain Research Center, Bar Ilan University, Israel e-mail: galk@cs.biu.ac.il

Ilan Lupu
Computer Science Department, Bar Ilan University, Israel

Noa Agmon
Computer Science Department, Bar Ilan University, Israel e-mail: agmon@cs.biu.ac.il

erence). On the basis of such graphs, it is possible to build a shared coordinate system (e.g., [11]), compute message passing paths in ad-hoc networks, and maintain stable formations (e.g., [3, 7]).

Existing work utilizing control graphs raises several open challenges. First, it offers no systematic treatment of the edge weights, how they are determined, and how they should be utilized in the computation of optimal control graphs. Different tasks (e.g., building a coordinate system versus formation maintenance) utilizes the edge weights differently. Second, it makes the assumption that a single robot is given, chosen to serve as the *global anchor* of the shared coordinate system, *leader* of the formation, or origin of a message whose position is taken as the basis for the robots' relative positioning and location estimates. Third, it ignores uncertainty in the weights of edges, such that, for instance, if the edge weight denotes a distance, it assumes the distance is known with certainty, despite the inherent uncertainty that exists in real-world sensing. In this work, we tackle these open challenges.

First, we synthesize from existing work, and then generalize the notion of control graphs and their uses. We begin by refining the definition of *monitoring multi-graphs* [7], which distinguish between different sensing configurations of robots. We show how existing techniques (e.g., for computing shared coordinate systems) can be optimized by re-casting them in terms of graph-theoretic algorithms for inducing directed trees from the multi-graphs, such that the trees optimize for a given criteria (e.g., team costs, individual position error). Each such tree is an *optimal control graph* for a given task (e.g., message passing, formation maintenance).

Second, on the basis of this more general understanding of how control graphs are generated from monitoring multi-graphs, we formulate the problem of optimal selection of *leader* or *global anchor* in a given monitoring multi-graph. A leader robot serves as the root of the control graph (tree) generated from it. We present a centralized algorithm that efficiently determines the optimal leader for a given task, as well as the resulting control graph.

We evaluate use of the novel algorithms on physical and simulated robots equipped with depth and image sensors (RGB-D cameras), and contrast them with results obtained from existing work. The results show very significant improvements from using these algorithms for coordinate frame alignment, in both simulated and real robots, in static and dynamic settings.

2 Related Work

The use of graph theory for reasoning about roles of robots in cooperative multi-robot tasks has a long history. We survey below only the most related, recent work.

Formation maintenance. Here the robots move while maintaining a shape, dictated by their relative positions. Desai et al. [3] defined a *control graph* as an unweighted directed graph (digraph) whose vertices are the robots in the formation. An edge from A to B represents that robot A monitors robot B's position. They show that

a formation can be stably maintained if the control graph implies each robot (except a single *leader*) maintains its bearing (angle) and separation (distance) with respect to one other robot (*target*). This type of formation control is known as SBC (Separation-Bearing Control). Without referring to control graphs, Fredslund and Matarić [4] propose a distributed algorithm for generating SBC monitoring rules (i.e., which robot monitors whom) given a target placement of the robots and the leader. In contrast, we consider weighted edges in our control graphs, and show how to induce *optimal* control graphs for different tasks (not just formations). We also address the question of leader selection. However, our algorithms here are centralized.

Kaminka et al. [7] generalized on these works. They defined a weighted monitoring multi-graph, which compactly represents all possible SBC control graphs for a given placement of robots. Each edge represents a possible configuration of the follower robot by which it can sense a target robot, and its weight represents its cost. They present a centralized algorithm for inducing a specific control graph, which optimizes the selection of targets, assuming a pre-determined leader. We show that their representation and algorithm is in a special case of a broader definition of monitoring multi-graphs, and we address the question of leader selection, which they leave open.

Lemay et al. [8] present a distributed method of assigning robots to formation positions. The computation relies on a cost function that considers distances and angles to the teammates; it outputs the lowest-cost assignment of robots to positions, and a leader that minimizes costs over all possible assignments. In contrast, we begin with robots already assigned to positions, and only then select a leader and SBC targets. However, we explicitly consider sensor capabilities, including errors.

Shared Coordinate Systems (Coordinate Frame Alignment). Another common task is that of multiple robots agreeing on a common coordinate system (axes and origin), e.g., as the basis for multi-robot mapping. There are several studies regarding the construction and alignment of coordinate systems (e.g., [5, 10, 12, 15]). Briefly, the task here is for robots to identify their alignment (translation and rotation) with respect to each other (typically one of the robots serves as a global anchor). As not all robots can sense the global anchors, they may instead localize via *anchor chains*, i.e., localize with respect to local anchors, who sense other anchors, etc. This is also referred to as coordinate frame alignment.

Most such work focuses on the filtering mechanisms able to cope with the uncertainty inherent to this process, and with various types of errors (e.g., receiving only range information). However, recently, Nagavalli et al. [11] presented a distributed method for improving the accuracy of such alignments, by utilizing a breadth-first search (BFS) to minimize the number of anchors in anchor chains, all beginning with a selected global anchor. In this paper we present a centralized algorithm for selecting an optimal global anchor in this task, and show that this further improves (significantly) the position estimates of the robots. Moreover, this works with anchors that are not part of the team, such as objects in the team surroundings that the robots can identify.

3 Optimal Construction of Control Graphs

We begin with robots placed in fixed relative positions, and no leader assigned. In Section 3.1 we show how to compactly represent all the different possibilities for robots to sense each other in their positions, using a refined definition of *monitoring multi-graphs*, originally presented in [7]. Then, in Section 3.1 we show how existing work can be re-cast in terms of graph-theoretical algorithms, properly extended to run on monitoring multi-graphs. Existing work leaves open the question of optimal leader selection, which we address in Section 3.2.

3.1 Monitoring Multi-Graphs

A monitoring multigraph captures all the potential control graphs for a group of robots in fixed positions. As defined in [7], it is a directed, weighted multigraph $G = \langle V, E \rangle$, where V is a set of vertices representing robots, and E is *a bag* (multi-set) of weighted edges between vertices.

Each $v_i \in V$ represents a unique robot i, identified by its index, and having a specific pose in space. The function $pos : V \mapsto \Re^n$ identifies the unique pose of each robot $v \in V$ (typically, $n = 3$, with the pose determined by the position and orientation of the robot v).

Let $v_i, v_j \in V$ be two robots. Suppose v_i can use a specific configuration of its sensors to sense v_j, i.e., v_i computes an estimate of $pos(v_j)$, denoted by $\hat{pos}(v_j)$. Denote the specific configuration by x. For instance, it may refer to a specific pan of a camera or Lidar, combined with a specific sensor processing algorithms (e.g., visual marking recognition, depth perception), or a specific choice of resolution or focus. [7] propose using a single scalar value c_{ij}^x as the edge weight, indicating preferences for using the sensor in this configuration, e.g.. based on reliability. We depart from this definition in two ways. First, we distinguish between *directly measurable resource costs* (such as expenditure of power, computation time, or sensor processing latency), and errors in the estimate $\hat{pos}(v_j)$, which are given in terms of deviations from the ground truth. Second, we accept that realistically, costs and errors can only be estimated with uncertainty. Thus we model them as random variables, with a known probability distribution function.

More precisely, with each measurable cost factor k in the operation of the sensor, and each component of error m resulting from it in $\hat{pos}(v_j)$, we associate a known probability distribution $C_{ij}^{x,k}$ ($R_{ij}^{x,m}$, respectively), explicitly or parametrically represented. For instance, if the perception latency l is known to be *uniformly* distributed in the range 20ms–30ms, this may be explicitly represented by setting $C_{ij}^{x,l} \equiv \mathscr{U}(20,30)$. If the distance from v_i to v_j is d, measured by a Lidar with a 3% error, we may set $R_{ij}^{x,d} \equiv \mathscr{U}(-0.015d, +0.015d)$. As v_i only approximates the true position of v_j with $\hat{pos}(v_j)$), we use an approximate distance measure d, and update it as additional measurements are made. The overall costs associated with the

edge e_{ij} are then drawn from the joint distribution of all $C_{ij}^{x,k}$, denote $\overline{C_{ij}^{x}}$. Likewise, we denote the errors by $\overline{R_{ij}^{x}}$.

Given these definitions, we define the edges in E as follows. An edge $e_{ij}^{x} \in E$ is a tuple $e_{ij}^{x} = \langle v_i, v_j, \overline{C_{ij}^{x}}, \overline{R_{ij}^{x}} \rangle$. When clear from the context, we omit the superscript ^{x}. This definition departs from [7] in that we add the representation of errors, and distinguish multiple components in costs and errors. We also depart from [7] in that we assume that the sensing robot can identify the sensed robot id and contrast the graph with the existing edges without assuming all possible edges can exist and eliminating edges that are occluded by other robots. Alternative configurations may result in improved costs or lower errors; often a robot may trade these off, e.g., by spending more computation time or more energy to improve its position estimate of the other robot. Given $|X|$ configurations for robot v_i to monitor v_j (which are usually determined by the number of different sensors the robot has), there exist edges $e_{ij}^{1}, e_{ij}^{2}, \ldots, e_{ij}^{|X|} \in E$.

Inducing Control Graphs with Uncertainty: Managing Risk. Following [9], we refer to a multigraph with random-variable weights as a *stochastic multigraph*. Different tasks, such as *formation maintenance*, may reduce to selecting paths in the multigraph. The length of a path in a stochastic graph is a function of random events characterized by the probability distributions associated with the cost along the path. We therefore have to decide how we would like to deal with the uncertainty. The common approach to dealing with uncertainty is by considering the risk involved in the decision. Standard policies include *risk-aversion* (hoping to reduce risk, even at higher cost, i.e., minimize the expected maximal cost/error); *risk-seeking* (inversely); and risk-neutrality (perfectly balancing risk and costs). Different decision strategies can lead to different shortest path selections.

Several such algorithms appear elsewhere [6, 9], and are outside the scope of this paper. However, it has been shown that risk-neutral selection both works correctly [9], and is safe, in the sense that it minimizes notions of regret [14]. For the remainder of the work, and in the experiments, we therefore used the risk-neutral policy, by using the expected (mean) value of the distributions $E[\overline{C_{ij}^{x}}]$ (or, as needed, $E[\overline{R_{ij}^{x}}]$) as the edge weights. Here $E[P]$ is the expected (mean) value of the probability distribution P.

Inducing Control Graphs (for a Given Robot). Monitoring multigraphs compactly represent all potential ways in which robots could monitor each other in their positions. Given a task which requires robots to monitor each other's positions (e.g., formation maintenance), we want to induce a *control graph*: a subset of the monitoring graph, which specifies for each robot which sensor configuration to use, and what other robot(s) to monitor, in order to improve task performance.

Table 1 summarizes the progression in previous work. In the column marked "Arbitrary leader, arbitrary control graph" we list previous works which utilize heuristic algorithms for constructing control graphs which are not guaranteed to be optimal (in the sense of reducing accumulating errors or costs). In the next column, marked "Arbitrary leader, Optimal control graph", we list investigations which, for a pre-

determined leader, generate an optimal control graphs minimizing accumulating errors or costs (assuming scalar edge weights). A variant of Dijkstra's single-source shortest path (S3P), described in [7] is optimal for such cases.

	Arbitrary leader, arbitrary control graph	Arbitrary leader, Optimal control graph	Optimal leader, Optimal control graph
Algorithm type	Heuristic	Dijkstra's	All Pairs Shortest Path
Formation maintenance	[4]	[7]	This
Relative Localization	[5, 15]	[11]	Work

Table 1: Related work utilizing accumulating factors, re-cast by type of algorithm and problem settings. [7] uses costs to represent errors. [11] assumes uniform errors, allowing use of BFS instead of Dijkstra's algorithm.

3.2 Inducing Control Graphs with Optimal Global Anchor

Thus the challenge remains of determining the optimal leader (i.e., one whose associated control graph is superior to those of other leaders). Our task here is to select a single robot which will serve as a leader of a formation, or the origin point (global anchor) for an agreed-upon shared coordinate system. We will therefore optimize the leader selection and associated control graph to reduce the errors $\overline{R^x_{ij}}$.

3.2.1 Problem Formulation

K robots are positioned in space. Each robot is equipped with sensors, allowing it to identify (some) other robots in its vicinity, and to estimate their position with respect to itself (i.e., their position in its own ego-centric coordinate frames). Furthermore, we assume robots are able to communicate with their peers, at least with those they are able to observe. The settings are captured by a monitoring multi-graph G_K. The task is to extract a control graph where the coordinate frame of a single robot (global anchor) is used as the origin, and all robots align their coordinate frames to it. Because not all robots can directly sense the global anchor, each robot can decide to align its coordinate system with respect to one other robot (called local anchor), who aligns itself to the global anchor, or to another local anchor. Thus a coordinate frame alignment control graph has the following properties:

- The vertex representing the global anchor has an out-degree of 0.
- All other vertices (robots) have an out-degree of 1.
- There exist a path from every vertex (robot) to the vertex representing the global anchor.

A coordinate frame alignment control graph is optimal with respect to the selected global anchor v_A if it minimizes the errors in position estimates of the robots. Suppose we have a robot v_0. Its position estimate in the shared coordinate system accumulates errors with every local anchor it uses on a path from itself to the global anchor in the control graph. It thus seeks to minimize the sum of expected errors $\sum_{e_{ij}} E[\overline{R_{ij}}]$ where e_{ij} is an edge on the path from v_0 to v_A. The question is how to choose v_A.

3.2.2 Optimal Global Anchor Selection

A global anchor v_A is called optimal, if its associated control graph is superior to the control graphs associated with any other potential global anchor. We consider two different ways a control graph may be superior to another: It may reduce the average position error for the group (a societal view of errors), or it may reduce the maximal position error (an individual view of errors). Our task here is to determine the optimal global anchor for both definitions. The process includes the following steps (see details next).

1. Transform the stochastic monitoring multigraph G_K into an intermediate representation, G'_K, which is a deterministically-weighted regular digraph (embedding errors, and reversing direction of edges). This step is carried out in time $\mathcal{O}(|E|)$, where E is the bag of edges in G_K.
2. Apply an All Pairs Shortest Path (APSP) algorithm to the graph G'_K. The time needed depends on the algorithm chosen, but is generally $\mathcal{O}(|V|^3)$, where V is the set of vertices in G'_K (normally, $|V| = K$).
3. Determine for each robot $v \in V$ the set of shortest paths leading from it P_v. For each such set P_v, determine the sum of the path lengths S_v, or the maximal path length M_v, depending on the global anchor selection criteria. This is carried out in time $\mathcal{O}(|V|^2)$.
4. The global anchor v_A is one that minimizes S_{v_A} or M_{v_A}. This is determined in time $\mathcal{O}(|V|)$.

Transformation of G_K into G'_K. This step is carried out to transform the stochastic directed monitoring multigraph into a deterministic graph, which embeds the necessary information, yet amenable to the execution of familiar graph-theoretic algorithm. The graph $G'_K = \langle V', E' \rangle$ is built as follows.

First, we set $V' \leftarrow V$. Then, for each pair of vertices $v_i, v_j \in V$, we do the following: (1) If an edge e^x_{ij} exists, with error distribution $\overline{R^x_{ij}}$, then create a temporary reversed edge, e'^x_{ji}, with scalar weight $r^x_{ji} = E[\overline{R^x_{ij}}]$. (2) Among all edges e'^x_{ji}, select the one with minimum r^x_{ji}, i.e., $e_{ji} = \arg\min_{e'^x_{ji}}(r^x_{ji})$. Finally, (3) add e_{ji} to E'. The result is a directed graph, with scalar deterministic edge weights, in which all errors have

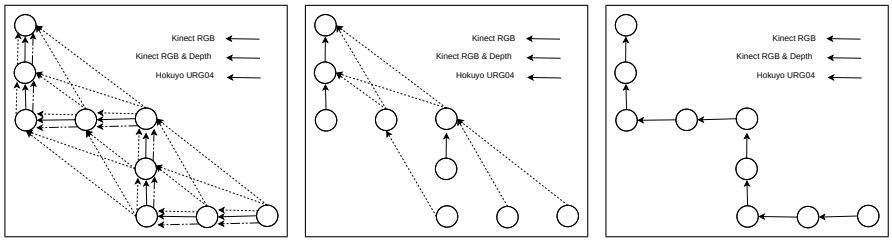


Fig. 1: An example for a monitoring multigraph (left), and two resulting monitoring graphs: one that minimizes the maximal path length (middle), and one that minimizes the sum of path lengths (right).

been folded into the edge weights using the risk-neutral policy, redundant edges in the multigraph removed, and edge direction reversed.[1]

All Pairs Shortest Paths. We now run an algorithm for determining the shortest paths for all pairs of vertices. In our implementation we utilized Johnson's algorithm [2]. Given the size of V' is the number of robots K, the algorithm runs in $\mathcal{O}(K^2 \log K + K|E|)$. The result is often represented in a matrix L, such that matrix cell l_{ji} contains the length of the shortest path from vertex j to vertex i (or ∞ if none exists). As edges are reversed in direction compared to the sensing direction, l_{ji} is the accumulating error in position estimates, from robot v_i to robot v_j, where $v_i, v_j \in V$.

Determine S_v and/or M_v. We propose two different criteria for selecting a global anchor that, if used as the origin for a shared coordinate system, would result in smaller position estimate errors for the team of K robots. One possible criterion is to minimize the mean position error of all K robots. This is a societal criterion, as it balances the errors across all robots. An alternative criterion is to minimize the worst-case error of any single robot, possibly resulting in some robots accepting a larger error than individually needed, in order to reduce the error of the other robots.

We examine the matrix L. Let S, M be vectors of dimension K. We denote S_v the component of S associated with a given v (and similarly, M_v). For all $v \in V$, $S_v = \frac{1}{K} \sum_{i=1}^{K} l_{vi}$, i.e., the sum of all cells in row v divided by K, or more intuitively, the mean length of shortest paths from all robots i to robot v. As these shortest path represent smallest errors, this is the mean smallest error in position estimates, if v is selected as global anchor. Similarly, for all $v \in V$, $M_v = \max_{i=1}^{K} l_{vi}$, i.e., the maximal smallest error in position estimate for any robot i, if v is the global anchor.

Determine global anchor v_A. Finally, a new global anchor can be chosen, by setting $v_A = \arg\min_{v \in V'} S_v$, if we prefer a global anchor that minimizes the average position error, or $v_A = \arg\min_{v \in V'} M_v$, if we prefer to minimize the maximal error instead. If

[1] Note that one can decide at this step to use any function combining C and R.

there are ties, they can be broken by preferring according to the other criterion, or arbitrarily.

4 Evaluation

To evaluate the effects of using the techniques presented in this work, we implemented the algorithms for optimal global-anchor selection and coordinate frame alignment in ROS (Robot Operating System), to be used on Gazebo-simulatedand real RoboTICan Lizirobots (shown in Figure 2(b)). All robots in the team were marked with unique visual markers identifying each robot. Using image and depth data from an RGB-D sensor, each robot identified its neighbors and measured their relative position in its reference frame. A calibrated sensor model was used to estimate the error measurements R_{ij}.

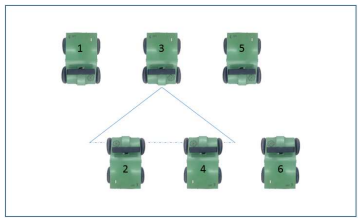
We compared the global position errors resulting from using the optimal v_A algorithm above, to the errors resulting from using an arbitrary robot [11]. Specifically, we contrast the robots' estimates with the ground truth measured externally. This was done by carrying out five repeated trials in each setting, each lasting two minutes, resulting in thousands of data points, *for each robot*.

We have carried out experiments in three types of settings: robots standing still, robots moving while maintaining a static formation, and robots moving while changing formation. In the first two settings, the relative positions of the robots are maintained: by definition in the first setting, and using feedback control in the second. In the third setting, moving robots changed their initial formation, requiring them to select a new global anchor.

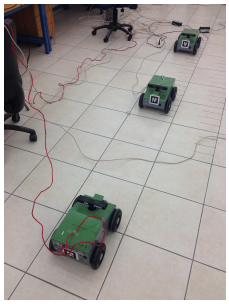
Our first experiment recreates an experiment in [11]. Six Lizi robots are placed as shown in Figure 2(a). All robots are static, and align their coordinate system with respect to the selected global anchor. Similar experiments involve placing three robots as shown in Figure 2(b). These were conducted both in simulation, as well as in real robots. Robot 1 (bottom of the image) could monitor robot 2 (center) and vice versa; robot 3 could see robot 2.

We then turned to experiments where robots moved while continually estimating their position based on a shared coordinate system, with the origin at the selected global anchor. We placed four robots in the formation shown in Figure 3(a), again both in simulation as well as in the lab. Robot 1 (front of the formation) could monitor robot 2 (center) and vice versa, robots 3 and 4 (side by side, bottom) could monitor robot 2. Figure 3(b) shows the real robots in one of the trials. In the arbitrary ID settings, robot 1 was selected as the global anchor. In the optimal settings, our algorithm chose robot 2 as the global anchor.

As a final experiment, we tested the ability of the algorithm to adjust the global anchor while moving, when the relative position of robots is changed. Four simulated robots were placed as shown in Figure 3(c). All robots moved forward; robots 1–3 at constant speed, and robot 4 three time faster, along the dotted path shown in the figure, and until it pulled ahead of everyone else. While moving, the robots

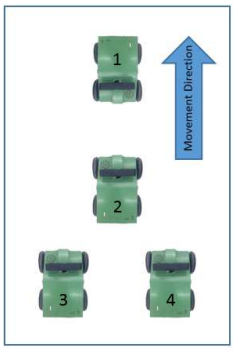


(a) Six simulated robots.

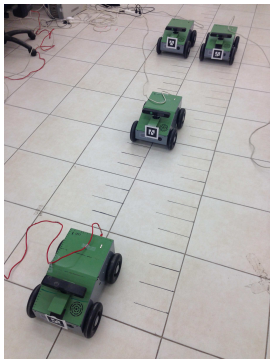


(b) Three real robots. Simulated robots placed likewise.

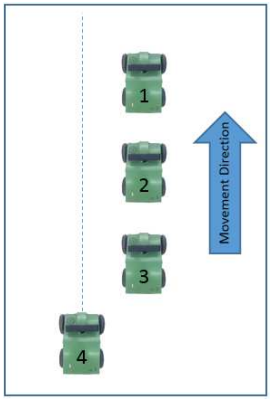
Fig. 2: Formation in static experiments.



(a) Static formation placement.



(b) Static real robots.



(c) Dynamic formation. Robot #4 overtaking others.

Fig. 3: Formations maintained while moving.

continually checked and recomputed the global anchor appropriate to their current settings. At the beginning of each run, robot 1 was chosen as global anchor v_A, and the algorithm chose local anchors for all other robots: robot 4 monitored 3, which monitored 2, which monitored 1. However, as robot 4 begins to overtake it peers, its local anchor changes from 3 to 2, then to 1, until finally it overtakes robot 1, at which point it becomes the global anchor, and root 1 switches to monitor it.

Figure 4 shows the mean error (error bars indicate standard deviation) of robot 4 during the experiment. It shows that between 0.1 minutes and 0.5 minutes into a trial, when robot 4's local anchor is robot 3, the error in position (in the shared coordinate system where robot 1 is the origin) is around 40cm. After passing robot 3, robot 4 changes local anchor based on the optimal selection, first to robot 2 and

then to robot 1. Approximately 0.95 minutes into the run, and until 1.15 minutes in it, robot 4's local anchor is robot 1 which is still the global anchor v_A. We see a corresponding decrease in robot 4's position error as it now monitors the global anchor directly. After 1.15 minutes, robot 4 cannot see any other robot and its error increases due to moving and assuming location in its last position. With real robot it is possible to change the localization method to less accurate one such as GPS in this situation. After robot 4 enters robot 1's field of view, the algorithm sets robot 4 to serve as v_A.

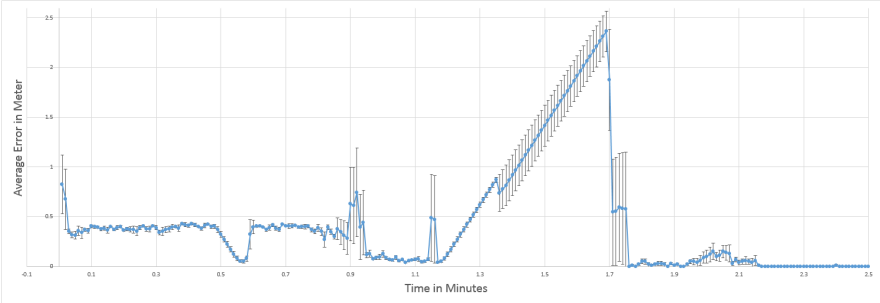


Fig. 4: Changing control graph in real time

Results. The results, summarized in Table 2, show the use of the leader-selection algorithm leads to very significant improvements in the position estimates of the robots in the shared coordinate system. In many cases, the mean error is reduced by 50% or more. For example, in the experiment with six standing robots, when using the minimal robot ID as a global anchor the farthest robot (#6) was located five hops away, and accumulated approximately 13cm in error. However, using the global anchor selected by our algorithm, the average error for the same robot, now located 3 hops away, drops to 6cm. This improvement is statistically significant (one tailed t-test, $p < 7.49 \times 10^{-16}$). Similar improvements can be seen in the other experiments, both in simulated and real robots. Over all trials, these results are over approximately 5000 measurements in each settings, for each robot.

5 Conclusions and Future Work

Control graphs are used in multi-robot systems to maintain information about which robot senses another robot, and at what position. On the basis of such graphs, it is possible to compute a shared coordinate system, localize relative to others, and maintain stable formations. In this work, we demonstrated that previous work assumes that a robot is pre-determined, to serve as global anchor (origin point) for coordinate frame alignment. We extended previous notions of *monitoring multigraphs*,

Type	Experiment	Robot ID	Arbitrary v_A Error in meters	**Optimal** v_A Error in meters	Significance p value (one-tailed t-test)
Standing	3-line (simulation)	3	0.058 (0.102)	**0.036 (0.009)**	7.12×10^{-15}
	3-line (real robots)	3	0.107 (0.019)	**0.049 (0.001)**	0 (below excel limit)
	6 zigzag (simulation)	2	0.031 (0.021)	**0.014 (0.006)**	2.62×10^{-78}
		4	0.073 (0.142)	**0.030 (0.005)**	4.12×10^{-15}
		5	0.086 (0.181)	**0.032 (0.005)**	4.90×10^{-15}
		6	0.134 (0.239)	**0.061 (0.033)**	7.49×10^{-16}
Moving	Simulation 4 center	3	0.036 (0.014)	**0.019 (0.018)**	$1.72 \times 10{-}98$
		4	0.032 (0.017)	**0.013 (0.012)**	5.92×10^{-143}
	Real moving 4 center	3	0.155 (0.076)	**0.095 (0.009)**	8.31×10^{-6}
		4	0.140 (0.105)	**0.084 (0.038)**	0.00056

Table 2: All experiment results, including mean errors in meters (standard deviations), and t-test significance testing. Robot ID is shown for robots not acting as global anchor v_A in either setting. The optimal global anchor column shows significant improvement in *all experiments*.

a construct intended to compactly represent all possible control graphs. We focused on risk-neutral decision policy, which allows us to replace stochastic edge weights with the deterministic expected value of the distributions. Second, we demonstrated that an All Pairs Shortest Path algorithm can be utilized, on the extended monitoring multi-graph, through some transformations. This facilitates the automatic determination of an optimal robot to lead a formation or serve as a global anchor. We conducted extensive experiments in real and simulated robots; these show very significant improvement to the robots' position estimates. In future work, we hope to examine alternative methods for dealing with decision policies that are risk-averse, or risk-seeking.

The algorithms presented herein assume that all information about the sensing capabilities and location of the robots is known - either to a centralized unit, or to one of the robots. Using this information, the optimal local and global anchors are determined. It would be interesting to extend these results to a decentralized setting. In this case, choosing a local anchor may be straightforward, yet choosing an optimal global anchor would require using innovative methods.

Acknowledgements We gratefully acknowledge support by ISF grants #1511/12 and #1337/15. As always, thanks to K. Ushi.

References

1. T. Balch and R. Arkin. Behavior-based formation control for multi-robot teams. *IEEE Trans. on Robotics and Automation*, 14(6):926–939, 1998.
2. T. T. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

3. J. P. Desai. Modeling multiple teams of mobile robots: A graph-theoretic approach. In *IROS*, volume 1, pages 381–386, 2001.
4. J. Fredslund and M. J. Mataric. A general algorithm for robot formations using local sensing and minimal communications. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.
5. A. Howard, M. J. Matarić, and G. S. Sukhatme. Putting the 'i' in 'team': an ego-centric approach to cooperative localization. In *ICRA*, pages 868–892, 2003.
6. L. K. Hwang. Stochastic shortest path algorithm based on lagrangian relaxation. Master's thesis, University of Illinois at Urbana-Champaign, 2010.
7. G. A. Kaminka, R. Schechter-Glick, and V. Sadov. Using sensor morphology for multi-robot formations. *IEEE Transactions on Robotics*, pages 271–282, 2008.
8. M. Lemay, F. Michaud, D. Létourneau, and J.-M. Valin. Autonomous initialization of robot formations. In *ICRA-04*, 2004.
9. R. P. Loui. Optimal paths in graphs with stochastic or multidimensional weights. Technical Report TR115, Computer Science Department, University of Rochester, 1982.
10. A. Martinelli, F. Pont, and R. Siegwart. Multi-robot localization using relative observations. In *ICRA-05*, pages 2797–2802. IEEE, 2005.
11. S. Nagavalli, A. Lybarger, L. Luo, N. Chakraborty, and K. Sycara. Aligning coordinate frames in multi-robot systems with relative sensing information. In *IROS-14*, pages 388–395. IEEE, 2014.
12. G. Piovan, I. Shames, B. Fidan, F. Bullo, and B. D. O. Anderson. On frame and orientation localization for relative sensing networks. *Automatica*, 49(1):206–213, 2013.
13. E. C. S. Y Chiem. Vision-based robot formations with bézier trajectories. In *IAS-8*, volume IOS Press, 2004.
14. M. Traub, G. A. Kaminka, and N. Agmon. Who goes *there*? using social regret to select a robot to reach a goal. In *AAMAS-11*, 2011.
15. N. Trawny, X. S. Zhou, K. Zhou, and S. I. Roumeliotis. Interrobot transformations in 3-d. *IEEE Trans. on Robotics*, 26(2):226–243, April 2010.