The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Coordinated Navigation of Multi-Robot Systems with Binary Constraints

Bernd Brüggemann and Dirk Schulz

Abstract— **In this paper we present a method for navigating a multi-robot system through an environment while additionally maintaining a predefined set of constraints. Possible constraints are the requirement to keep up the direct line-of-sight between robots or to ensure that robots stay within a certain distance. Our approach is based on graph structures that model movements and constraints separately, in order to cover different robots and a large class of possible constraints. Additionally, the partition of movement and constraint graph allows us to use known graph algorithms like Steiner trees to solve the problem of finding a target configuration for the robots. We construct so called separated distance graphs from the Steiner tree and the underlying roadmap graph, which allow assembling valid navigation plans fast.**

I. Introduction

A. Motivation

The use of a multi-robot system (MRS) has several advantages over single-robot solutions: a group of robots is more robust against failure, it can perform many tasks faster, and specialized robots for specific sub-tasks can be employed to more effectively achieve a common goal. However, to exploit these advantages, the task execution of the group of robots needs to be coordinated. In this work, we look at planning the joint task execution. More specifically, we investigate the problem of coordinated multi-robot navigation, if additional constraints need to be fulfilled. Such constraints are not directly related to the goal to be achieved, but have to hold throughout task execution, to ensure a successful mission.

In this paper we propose a graph-based approach that allows specifying and maintaining a large class of such constraints, like

- ensuring that each individual robot never departs from the group by more than a certain distance,
- maintaining a formation in which every robot can be seen by its neighbors, or
- keeping up the wireless communication within a complete group of robots.

Note that maintaining such constraints may require more robots than fulfilling the direct goals of a mission alone, e.g. if additional robots are required as relays to ensure the communication. Our planning approach also allows estimating the number of robots required to fulfil such a task.

In this article we will explain the graph construction underlying our planning approach, mostly using the problem of reaching a given set of goal positions while maintaining additional constraints as an illustrating example. The main idea of our approach is to use two separate models for the actual motion planning and for ensuring a global constraint. In a nutshell we construct a constraint graph from all given sets of binary constraints between robots and employ a Steiner tree algorithm for computing the final robot placements that fulfil the global constraint and achieve the mission objective. The actual navigation of the robots is planned using a navigation roadmap graph. Generating a navigation plan towards a target placement of the robots requires linking both graph structures in order to find possible paths for all robots. We introduce a third graph structure called *separated distance graph* (SDG) for this purpose that allows to quickly find feasible paths, which do not violate the group constraint. Based on the SDG we develop a search algorithm for coordinated navigation plans for a group of robots. We present several simulation experiments illustrating the properties and the performance of our approach in different types of environments.

The reminder of this paper is organized as follows: after a short overview showing robot navigation and coordination approaches, we define the important graph structures needed for our approach, and show some of their basic properties in Section II. Section III describes the planning problem and a possible solution. In Section IV some planning results of representative toy problems are presented to illustrate the algorithm. We conclude in Section V with a summary and some suggestions for future work.

B. Related Work

Only a well coordinated multi-robot system is able to fully utilize its capabilities. In the last years, as MRS got into focus of research, different kinds of coordination systems for MRS were developed.

When coordinate a MRS several different aspects have to be taken into account. Usually there are several different tasks to fulfil. Not only reaching a target position is important but also to achieve a mission goal. Especially when several robots have to work together because a single robot is not able to solve the problem (e.g. building up a communication line to get information from a specific robot) So task allocation is one important problem MRS have to cope with (see e.g. [4], [10], [11]).

When moving several robots in the same physical environment to different targets the problem of deadlocks as well as congestion occurs. If several robots have to reach different target locations in an environment with narrow passages, the robots may block or hinder themselves and non-optimal

B. Brüggemann and Dirk Schulz are with Unmanned Systems Group, Fraunhofer FKIE, 53343 Wachtberg, Germany {bernd.brueggemann|dirk.schulz}@fkie.fraunhofer.de

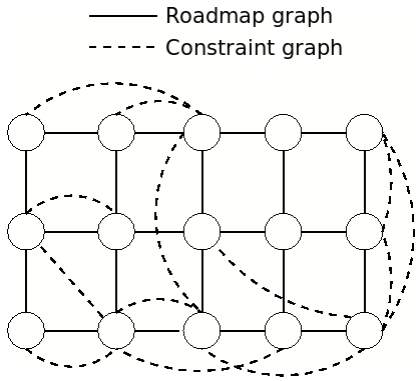

Fig. 1. An example for a roadmap graph and a constraint graph. Both have exactly the same vertices. The weight of each connection is 1.

paths have to be chosen. So the coordinated execution of a navigation plan in a MRS is a subject of research. There are several approaches to solve this problem in an either centralized or decoupled way (see e.g. [2], [6], [12]).

The here presented problem of moving a group of robots to certain destinations bear analogy to the so called multi-robot routing [5]. While we want to position one robot at each target, in multi-robot routing each target has to be visited by at least one robot. Nevertheless some problems occurring are similar e.g. finding the overall shortest path. Mosteo et al dealt in [9] with the problem of solving the multi-robot routing with communication constraint. In their algorithm *connect tree*, in contrast to our approach, the constraint problem is solved locally. To reach a target a group of robots is send to it. If the signal quality drops below a certain threshold a robot from this group has to stay as relay (detailed described in [8]). If a target can be reached depends on the number of robots in the group which is assigned to this target. As this number is not known in advance and can only be estimated, re-planning is necessary if a group contains to few robots. In this approach the plan consist of an intelligent way of assigning targets to the groups. Beside the slightly different task, there are some difference in the abilities of the connect tree algorithm and our approach. Although both algorithm guarantees a plan which do not violated the constraint our approach depends heavily on the world model to predict if the constraint between to points in the environment holds. in the current state of development it cannot deal with local changes. Furthermore, the connect tree algorithm is limited to the communication constraint while the here presented algorithm supports a larger group of constraints. In the end our global planning approach prevents the necessaries of re-planning during execution.

II. Basics and Definitions

In the following we will briefly introduce the notation used throughout the remaining article and formulate the actual task.

Our approach basically computes coordinated navigation plans on a roadmap graph $G_m = (V_m, E_m, d)$ within the free space of the environment, where d denotes the edge costs. Each vertex represents a position in (or a small area of) the environment and the edge costs correspond to the path length between the corresponding positions in the environment. The maximum "distance" between two neighbouring vertices is crucial in our approach, because it must not exceed the limits of the physical constraints we want to ensure, e.g. the reliable communication range of a Wi-Fi device if we want to maintain group communication.

Our aim is to move a group of robots from a given start configuration S_0 to some target configuration S_t. Here, a *configuration* $S = (v_1, \ldots, v_K)$ for a group of K robots is a vector that assigns each robot to a vertex of G_m. A *motion action* $a(S, S')$ transfers a configuration S of the multi-robot system into a new configuration S' by "moving" a single robot along an edge of G_m.

Within G_m motion planning for the MRS can be carried out by searching for a sequence of motion actions leading from S_0 to S_t, e.g. using the A*-algorithm. However, we want to ensure that a global constraint on the multi-robot configuration holds during the whole motion sequence, which requires additional efforts.

We introduce a *constraint graph* $G_c = (V_c, E_c, w)$ for this purpose. The vertex set V_c of this graph coincides with V_m and the edges in E_c formalize binary constraints between two positions in the environment. A binary constraint $(v, v') \in E_c$ holds in a configuration S if two robots are located at v and v' in S. Intuitively, such a constraint models a physical requirement in the real world. For example, if we want to ensure the wireless communication between two robots located at positions v and v', we have to make sure that the expected signal strength $L(v, v')$ between the two vertices exceeds a certain threshold L_{thresh}. In that case we add constraints to all pairs of adjacent vertices in the navigation roadmap, $(v, v') \in E_m$, for which $L(v, v') \geq L_{\text{thresh}}$.

For both graphs (see Fig. 1), we set edge weights w and d to 1. For the navigation roadmap this is achieved by arranging the vertices in a regular grid. For the constraint graph, the arbitrary choice of 1 has some convenient effects as we shall see later.

In our approach we now use the constraint graph to constrain complete configurations of the MRS. We say a configuration S is *valid* if G_c restricted to the vertices in S is a single-component sub-graph of G_c. Additionally, we say a motion action $a(S, S')$ is valid if it transfers the valid configuration S into a valid configuration S'.

The constrained motion coordination task now is to find a sequence of valid motion actions that transfers a valid initial configuration S_0 into a valid target configuration S_t. However, the user generally specifies only a set of target locations $Z \subset V_m$, $|Z| \leq k$, for the robots, based on the mission to be accomplished. Note that Z, interpreted as a multi-robot configuration, might not be valid, because the global constraint between the robots is generally not taken into account. The overall task can therefore be divided into two separate parts:

1) Given the constraint graph G_c and the set of targets Z, find a valid target configuration S_t.
2) Given a valid start configuration S_0 for k robots, the

constraint graph G_c, the navigation roadmap G_m and the valid target configuration S_t, find a sequence of valid motion actions, $a_1, \ldots, a_t$ that transfers S_0 into S_t via a sequence of valid intermediate configurations.

III. Constrained Motion Coordination for MRS

A. Finding a Valid Target Configuration

The task of finding a valid target configuration for the multi-robot system involves placing robots at each mission target $z_i \in Z$, and placing additional robots on vertices of V_c in such way, that the resulting configuration is valid. For a valid configuration S there always exists at least one holding constraint in the constraint graph for each vertex in S, and the corresponding edge of the constraint graph connects this vertex with some other vertex in S. The vertices in S together with the subset of holding constraint edges build a single-component sub-graph within the constraint graph. For this reason, one can obtain a valid target configuration by constructing its corresponding single-component constraint sub-graph. For this purpose we have to place the additional robots to achieve single-connectedness. This task is closely related to the Steiner tree problem.

The Steiner tree problem is a well known problem in graph theory, often encountered in communication and in technical computer science. Let $G = (V, E, c, T)$ be a single-component, non-directed graph. V is the set of vertices, E the set of edges and c the weight associated with the edges. The set T, called terminals, is a subset of the vertices. A tree in graph G is called Steiner tree if it connects all vertices from T and the sum of all weights is minimal. A Steiner tree normally consists of more than the terminal vertices. The additional vertices are called Steiner vertices.

Considering the set of targets Z as terminals, the Steiner tree problem exactly matches the task of finding the target configuration S_t. The Steiner tree $G = (V_c, E_c, w, Z)$ for the constraint graph G_c is one possible valid target configuration S_t. In fact, this solution achieves optimality with respect to the number of robots required. As G_c has only equal edge weights of one, the Steiner tree minimizes the number of Steiner vertices.

Proof: The total weight of a Steiner tree is $\Sigma_{i=1}^{|G|} c_i$ where $|G|$ is the number of edges in the tree. If all weights are 1, the sum is equal to the number of edges. As the Steiner tree is a tree, the number of edges is equal to the number of vertices minus 1. As the number of terminals is fixed, the Steiner tree minimizes the number of Steiner vertices. ■

To sum up, finding a valid target configuration where all targets are reached by a robot and the global constraint is fulfilled, is equal to computing the Steiner tree on the constraint graph with the robots' mission targets as terminals. Unfortunately, computing the Steiner tree is NP-complete [3], but several heuristics exist which compute near optimal Steiner trees in polynomial time (see e.g. [1]).

Knowing that the Steiner tree is a solution for the problem of finding S_t, we can state two properties of the problem:

First, you need at least $|Z|$ plus the number of Steiner vertices robots to fulfil the task.

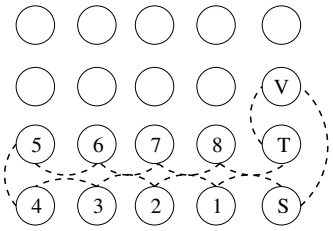

Fig. 2. There is no non-trivial upper limit for the number of relay robots needed. Constraint Graph edges are dashed lined and connections of the roadmap are not displayed. Each vertex is connected by an edge to its direct neighbour in horizontal and vertical direction. Depending on how the constrain graph is build relay robots may be needed on each vertex.

Proof: As stated above, for graphs with all weights equal to one, the Steiner tree minimizes the number of Steiner vertices. Since the Steiner tree is one solution for S_t, it is also the solution with the minimal number of vertices. Therefore, you need at least that many robots to obtain a valid target configuration. ■

This limit is hard and gives the number of robots needed for any valid target configuration. But it is not guaranteed that you can reach this configuration with the same number of robots. More robots might be necessary for intermediate configurations.

Second, as can be seen in Figure 2, if you allow any set of binary constraints, in the worst case the number of robots required to obtain a valid plan is equal to $|V_c|$.

This shows that it may be difficult to move a robot from vertex a to vertex b without violating the constraint. The problem to find positions for relay-robots if needed arises. This leads to the question how many robots are need to get the robot to b.

B. Finding Valid Navigation Plans

As can be seen in Figure 2. All robots have to reach certain target vertices (either a Steiner vertex or a terminal) in the target configuration S_t and the global constraint between the robots has to be maintained throughout the navigation. While the robots' motions are planned based on the navigation roadmap G_m, the constraint can only be checked based on the constraint graph G_c. To plan valid paths, we therefore frequently have to map between constraint paths in G_c and motion paths in the navigation roadmap G_m. As shown in Figure 2, such paths are not necessarily shortest paths to the target vertices; they may have any topology. Additionally, to navigate a robot to its target vertex, further robots may be needed to build up relays to ensure the global constraint.

To ease this planning problem we introduce a graph structure that we call *S*eparated Distance Graph (SDG). A $SDG(v) = (V_S(v), E_S(v), W)$ is constructed for a specific vertex v; its vertices $V_S(v)$ are v together with all vertices directly connected to v in G_c. Let $G_{\mathrm{m}}^{\mathrm{sub}}(v)$ be the navigation roadmap restricted to the vertices in $V_S(v)$. There is an edge between v and a vertex in $SDG(v)$ if there is a path in $G_{\mathrm{m}}^{\mathrm{sub}}(v)$ between both. The weight of this edge is equal to the number of vertices on this path.

Intuitively, an edge $(v, x) \in E_S(v)$ states that a robot can travel without violating its binary constraint to x as long

as a robot is located at the vertex v. The set of separated distance graphs for a given planning problem now allows us to derive a search algorithm for an abstract navigation plan that achieves the target configuration. This algorithm performs a search for valid intermediate robot configurations instead of searching in the much larger space of possible motions in the navigation roadmap G_m.

Formally, we want to find a sequence of valid configurations $S_0, \ldots, S_K$ that leads to the target configuration $S_t = (s_t^1, \ldots, s_t^K)$ starting from the start configuration $S_0 = (s_0^1, \ldots, s_0^K)$. We make two additional assumptions:

1) All robots start at the same location s_0.
2) One robot stays fixed, i.e. $s_t^j = s_0$ for some robot.

These two conditions allow to directly use the Steiner tree corresponding to S_t as an abstract plan for moving the robots. The idea is to sequentially cover the nodes of S_t in the sequence of a tree traversal through the Steiner tree starting from s_0.

However, implementing this approach is not straight forward. Although two neighbouring vertices in the Steiner tree always fulfil the constraint that has to be ensured, it is generally not possible to navigate a single robot on a path through G_m to this vertex, without violating the constraint at some intermediate position. This happens, e.g. if reaching a position requires lengthy detours due to obstacles in the environment. So in the general case, additional temporary relay-robots are required. To compute a valid navigation plan from a vertex a of the Steiner tree to a neighbouring vertex b, we therefore perform a breadth-first search of valid motions through G_m.

To find the path and the positions for the relay robots we build up a search tree. The root of the tree is the start vertex a. At the root we add those vertices as leafs which are connected with a in SDG(a). If b is now within the search tree, there is a direct way from a to b without the need of a relay robot. Otherwise we perform following steps until b is in the search tree:

1) Let x be the leaf of the search tree with the lowest depth
2) Add all vertices from SDG(x) which are not already in the search tree as leafs to x

If b is added in that way, the positions of the relay robots can be determined by following the path from b to a in the search tree. Every vertex visit on that path is a position for one relay-robot.

By construction, the breadth-first search returns the path from a to b with the smallest number of intermediate relay-robots required, i.e. the number of intermediate nodes on the path. On each temporary relay position, we leave one robot behind, in order to maintain the constraint. As soon as the remaining robots reach the next target location, the temporary relay-robots can catch up, without violating the constraint, e.g. by moving the farthest robot first.

Following this procedure, the complete target configuration can be reached without violating the global constraint. A sufficient number of robots to execute this plan is K plus the maximum number of temporary relay-robots required to traverse a single edge of the Steiner tree.

C. Summing up

We want to plan the coordinated navigation of a group of robots which have the task to cover a set of target locations $Z = \{z_1, \ldots, z_l\}$. We assume that all robots start at the same location z_1, that is also a target location – think of a base station that is supposed to remain fixed. During navigation, the robots must maintain a global constraint that can be expressed by a graph of binary constraints between robots. This constraint is fulfilled if the graph of holding constraints between robots has a single-component. An example for such a global constraint is maintaining wireless group communication during navigation. Due to the additional constraint, the task might require more than $|Z|$ robots. To solve this problem, we carry out the following steps:

1) Given the targets Z, a navigation roadmap G_m for the environment and a symmetric binary relation that should hold between robots, we compute the corresponding constraint graph G_c for the environment.
2) Given G_m and G_c we compute SDGs for all vertices of G_c.
3) Given Z and G_c, we compute a valid target configuration $S_t = (s_t^1, \ldots, s_t^K)$ by approximately solving the associated Steiner tree Problem.
4) Given s_0, S_t and the set of SDGs, we compute the temporary relay positions required to traverse the edges of the Steiner tree corresponding to S_t. This procedure also gives an upper bound on the number of robots required to execute the task.
5) The final plan now consists of the tree traversal of the Steiner tree with all robots, placing a robot at each target location, as well as placing and removing temporary relay-robots.

D. Computational Complexity

The resulting planning process can be divided into two parts: The pre-processing and the query. During the pre-processing, the constraint graph G_c and the set of all SDGs are build up. When building the constraint graph each pair of vertices has to be checked whether the binary constraint is satisfied. Although certain constraints allow to reduce the number of pairs to check, the overall complexity is in $O(n^2)$, with n being the number of vertices in G_c. The cost of building the set of SDGs is rather dependent on the number of neighbors a vertex in G_c has. For each vertex v its SDG has to be set up. Therefore, each vertex connected with v in G_c has to be put into the SDG and it has to be checked whether there is a connection in G_m. Let k be the number of vertices in the SDGs, and s the number of edges in $G_{\mathbf{m}}^{sub}(v)$. As the needed information can be obtained e.g. by a Dijkstra-Algorithm, the time needed to build up one SDG is in $O(k \cdot log(k) + s)$. As it has to be done for each vertex in G_c the overall computational complexity is in $O(n \cdot (k \cdot log(k) + s))$.

The query step consist also of two tasks: finding the valid target configuration and the way between the vertices of that positioning. The computational complexity of finding the valid target positioning depends on the chosen algorithm for approximated Steiner trees. When choosing e.g. the algorithm of Mehlhorn [7] we have to expect the approximated Steiner trees inferior to the exact Steiner tree by a factor of two, but we can compute it in a complexity equal to $O(n \cdot log(n) + E)$ where E is the number of edges in G_c. The search for the paths between two vertices in the target positioning is easy due to the usage of SDGs. Suppose the target configuration consists of t edges, the path finding is possible in $O(t \cdot n)$. So the query step can be performed in $O(n \cdot (log(n) + t) + E)$.

IV. Experiments and Results

For the experiments, we have chosen four different maps, in order to illustrate different properties of the algorithm. These four maps are:

- Empty world: an empty world with no obstacles,
- Bars: a map with two parallel linear obstacles,
- Open box: a map with a box shaped obstacle with open corners,
- Spiral: a spiral shaped map.

We applied our algorithm to three different kinds of binary constraints:

- Distance constraint: a robot is not allowed to move further away from the next robot than a certain threshold.
- Vision constraint: a robot is not allowed to move out of sight of the following robot.
- Wireless connection constraint: a robot is not allowed to lose its communication connection to the other robots. To predict the connectivity we used a simple log-distance path loss model for large scale fading combined with a attenuation of obstacles linear to their thickness.

The different constraints have specific characteristics. The distance constraint is the simplest one. If there is no obstacle in the environment, the target configuration can be reached without any temporary relay-robots (see Figure 3). More interesting are environments with obstacles. As the constraint is not affected by any obstacles, the Steiner tree representing the target configuration, can have edges crossing obstacles (see Figure 4). In this case, it is not possible to follow some paths to target locations with a single robot without violating the distance constraint, and temporary relay-robots are needed. In the spiral shaped world the difference between the Steiner vertices and the temporary relay-robots required is significant. There is only a single final position and only a few Steiner vertices, but a lot of temporary relay-robots are needed (see Figure 5).

The vision constraint is, in contrast to the distance constraint, directly affected by obstacles in the environment. Here, it is essential that the robot can follow an edge of the Steiner tree, because there is no obstacle between two vertices in the target configuration. Therefore, temporary relay-robots are not needed (see Figure 6). Within a blank environment, this constraint is trivial, as the target configuration

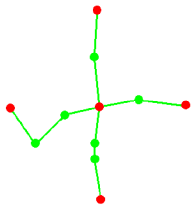

Fig. 3. Distance Constraint, empty world. Resulting target configuration. Red dots are terminals (starting position in the middle and destination chosen by the operator), green dots are Steiner vertices.

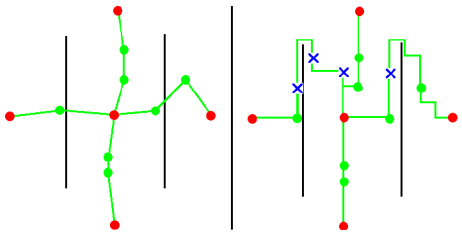

Fig. 4. Distance constraint, bar-shaped obstacles. *Left:* Resulting target configuration. Note that the distance constraint is not affected by obstacles. The Result is similar to the empty world. *Right:* The resulting plan. Blue crosses are the temporary relay-robots, which appear if a path has to be build around obstacles.

consists only of the target positions. In Figure 6 you can also see, that the number of robots is minimized, not the distance covered. If different kinds of obstacles are introduced, like e.g. windows, this constraint will become more complex.

As an example of constraints for keeping up the communication, we used a log-distance path loss model together with attenuation in obstacles. The signal strength is not the only indicator for a working wireless communication, nevertheless gives a hint. In Figure 7 the target configuration and the plan for the bar world are illustrated. Here you can see ,that in contrast to the distance constraint, the obstacles limit the range of influence of each vertex. So it is possible to have edges of the target configuration crossing obstacles, their maximum length is shorter than the maximum length of those

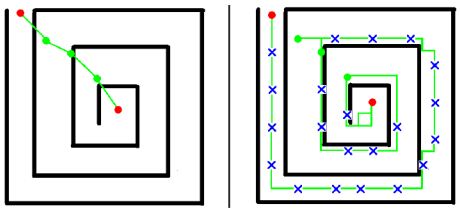

Fig. 5. Distance constraint, spiral. *Left:* The target configuration. Unaffected by obstacles, only few Steiner vertices are needed. *Right:* The resulting plan. Several temporary relay-robots are needed to reach the target position at the entrance.

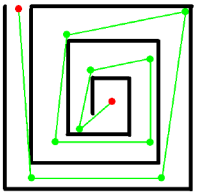

Fig. 6. Vision constraint, spiral. The target configuration is equal to the plan. A robot have to be placed at every corner, otherwise the line-of-sight will be lost.

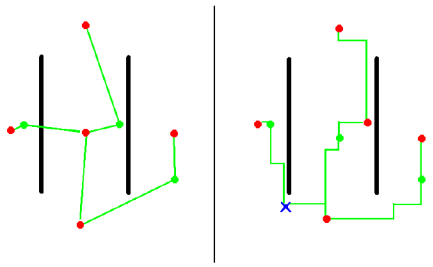

Fig. 7. Log-distance constraint. *Left:* The target configuration. This constraint shows characteristics of both other presented constraints. Edges may pass obstacles and some edges looks like a visual constraint with limited range of view. *Right:* The resulting plan. As in the distance constraint additional relay robots may be needed to pass an obstacle.

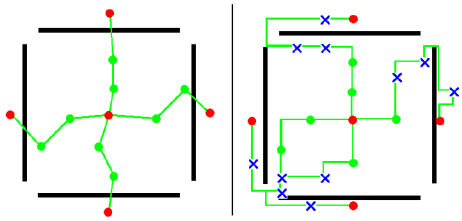

Fig. 8. Distance constraint, open box. *Left:* The target configuration. The distance constraint is not affected by obstacles. The resulting positioning is similar to the positioning in the empty world. *Right:* The resulting plan. The number of used robots is not optimal. The crossing in the left bottom corner is not necessary.

edges passing only free space. Here again, you can see that, at the moment the algorithm optimizes the number of robots for the target configuration not the length of the path.

Although the algorithm is build to minimize the number of used robots in the target configuration, there are cases in which the path to this minimal target configuration is not optimal. One of these cases can be seen in Figure 8. Here, the target configuration is a Steiner tree with the root as starting position. Four sub-trees expand from the root to the target positions. The algorithm is designed to treat those four ways independently. This explains the resulting plan: the path to the bottom and left target areas are calculated separately. For each of these paths different temporary relay-robots are needed. This is not optimal because the robots could use the same way to the bottom left corner and split up. Additional optimization is necessary to identify such short cuts.

V. Conclusion

In this paper we presented a graph based approach to navigate a MRS while guaranteeing to fulfil a global group constraint during navigation. For this purpose, the possible movements of the robots and the constraint are represented by an independent constraint graph and a navigation roadmap graph. We showed that the problem of finding valid target configurations, that fulfil the global constraint can be solved using an approximate solution of the Steiner tree Problem. The Steiner tree can also be used as an abstract plan to reach the target configuration, but generally additional relay-robots are required to ensure the constraint during navigation. We introduced a new graph structure called Separated Distance Graph (SDG) that links the constraint graph with the navigation roadmap and allows identifying the additional relay positions using a breadth-first search technique. We provided several simulation experiments showing the capabilities and the behaviour of our navigation approach in different types of environments.

As also shown in the experiments, the algorithm is not optimal with respect to the number of robots required to solve a task. Here, optimization after planning may be promising. A post-processing step will be necessary to enhance the result.

This approach is intended to be used also in partly known environments. Therefore, it has to react to changes in environments. We kept this in mind during the development of the presented algorithm. For example, there are known heuristics to compute a Steiner tree for dynamic graphs, so that the target configuration can be maintained over time. It will be important to analyse how the SDGs will behave for dynamic graphs. Because of the chosen representation of environments, we expect that local changes in environments will require only local re-planning. Our goal is to be capable of doing this re-planning on-line

References

[1] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel. Lower bounds for approximation algorithms for the steiner tree problem. In *Proceedings of 27th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 2001, Springer LNCS*, pages 217–228. Springer Verlag, 2001.

[2] Y. Guo and L. E. Parker. A distributed and optimal motion planning approach for multiple mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2612–2619, 2002.

[3] F. K. Hwang and D. S. Richards. Steiner tree problems. *Networks*, 22(1):55–89, 1992.

[4] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[5] M. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, pages 343–350. Citeseer, 2005.

[6] S. M. Lavalle and S. A. Hutchinson. Optimal motion planning for multiple robots having independent goals. *IEEE Trans. on Robotics and Automation*, 14:912–925, 1996.

[7] K. Mehlhorn. A faster approximation algorithm for the steiner problem in graphs. *Inf. Process. Lett.*, 27(3):125–128, 1988.

[8] A. R. Mosteo, L. Montano, and M. G. Lagoudakis. Multi-robot routing under limited communication range. In *IEEE International Conference on Robotics and Automation (ICRA) 2008*, 2008.

[9] A. R. Mosteo, L. Montanoand, and M. G. Lagoudakis. Guaranteed-performance multi-robot routing under limited communication range. In *Distributed Autonomous Robotic Systems 8*, pages 491–502. Springer Berlin Heidelberg, 2009.

[10] L. Parker. Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1), 2008.

[11] L. E. Parker and A. Howard. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *International Journal of Robotics Research*, 25:431–447, 2006.

[12] L. E. Parker, B. Kannan, F. Tang, and M. Bailey. Tightly-coupled navigation assistance in heterogeneous multi-robot teams. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1016–1022, 2004.