ARMO: Adaptive Road Map Optimization for Large Robot Teams

Alexander Kleiner*, Dali Sun* and Daniel Meyer-Delius* *

Abstract

Autonomous robot teams that simultaneously dispatch transportation tasks are playing more and more an important role in present logistic centers and manufacturing plants. In this paper we consider the problem of robot motion planning for large robot teams in the industrial domain. We present adaptive road map optimization (ARMO) that is capable of adapting the road map in real time whenever the environment has changed. Based on linear programming, ARMO computes an optimal road map according to current environmental constraints (including human whereabouts) and the current demand for transportation tasks from loading stations in the plant. We show experimentally that ARMO outperforms decoupled planning in terms of computation time and time needed for task completion.

1 INTRODUCTION

Recent trends in logistics and manufacturing clearly indicate an increasing demand for flexibility, modularity, and re-configurability of material flow systems. Whereas in the past plant installations have been used for decades without change, nowadays product life cycles and the demand for product variety rely on innovative technologies that allow to flexibly reconfigure automation processes without reducing their availability. Therefore, distributed and self-organized systems, such as teams of robots that autonomously organize transportation tasks, are playing an increasingly important role in present logistic centers and manufacturing plants.

Besides the task assignment problem, i.e., allocating robots to different tasks [14], another challenge in this domain is to efficiently coordinate the simultaneous navigation of large robot teams in confined and cluttered environments. In general, multiple robot motion planning can be solved by either considering the joint configuration space of the robots [2] or by deploying decoupled techniques that separate the problems of motion planning and coordination [10]. Whereas the first approach is intractable for large robot teams since the dimension of the joint configuration space grows linearly and thus the search space grows exponentially with increasing number of robots, the second approach yields typically sub-optimal solutions, for example, requiring the robots to perform larger detours in order to avoid collisions. Road map planners are a popular method for single robot planning in static environments [9] that compute during a pre-processing phase a connectivity graph in free configuration space that is then used for efficient path planning during runtime. However, dynamic domains, such as industrial environments, are more challenging due to permanent changes in the environment, e.g., due to the placement and removal of objects such as pallets and boxes, and the co-location of human workers.

In this paper we present adaptive road map optimization (ARMO) for large robot teams that is capable of adapting the road map in real time whenever the environment has changed. In short, the planner computes an optimal road map according to current environmental constraints (including human whereabouts) and the current demand for transportation tasks from the loading stations. We describe the environment of the robot with a spatial grid map in which a hidden Markov model (HMM) is used to represent dynamic changes [11]. From the continuously updated grid map the computation of a Voronoi Graph [4] is triggered whenever significant changes have been detected. The Voronoi graph, representing free space connectivity, is taken as a starting point to extract road segments (as shown in Figure 1) for the final road map. We use a Linear Programming (LP) approach for computing the optimal configuration of these segments with respect to minimal travel costs and maximal compactness of the network. Figure 1 depicts the re-arrangement of the road map after local changes of the environment have been detected. We show experimentally that ARMO outperforms decoupled planning in terms of computation time and time needed for task completion.

Kallman et al. used dynamic roadmaps for online motion planning based on Rapidly-exploring Random Trees (RRTs) [8]. Velagapudi et al. introduced a distributed version of prioritized planning for large teams where each robot plans simultaneously and re-plans in case a conflict has been detected [18]. Berg et al. presented a method for road map based motion planning in dynamic environments [16]. In contrast to our method, which learns changes of the environment online, their approach discriminates between static and dy-

** Department of Computer Science, University of Freiburg, Georges-Koehler-Allee 52, 79110 Freiburg, Germany, {kleiner,sun,meyerdel}@informatik.uni-freiburg.de

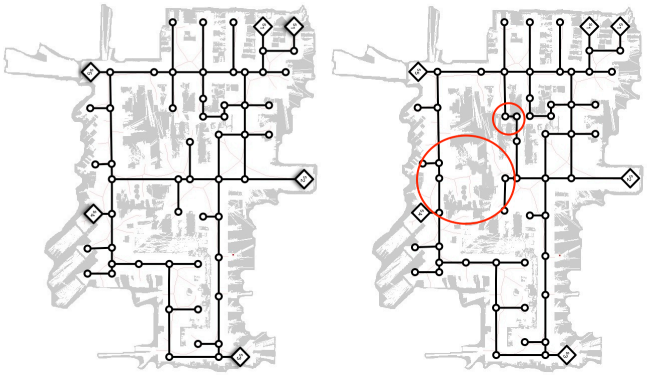

Figure 1: Motivating example: In industrial environments the map can locally change due to replaced objects, such as pallets, as well as gathering humans. Adaptive road map optimization facilitates the simultaneous navigation planning of large robot teams while respecting these changes.

namic objects, e.g. walls and robots, in advance, which might fail when also portions of the map have to be considered as dynamic. Bellingham et al. proposed a method for solving the cooperative path planning for a fleet of UAVs [3]. They formulate the task allocation problem as a mixed-integer linear program (MILP). Sud et al. developed an approach for path planning of multiple virtual agents in complex dynamic scenes [13]. They utilize first- and second-order Voronoi diagrams as a basis for computing individual agent paths. While computational efficient, their method does not focus on optimizing the global efficiency of the multi agent team.

The reminder of this paper is organized as follows. In Section 2 the problem is formally described and in Section 3 a description of the target system is provided. In Section 4 the algorithm for adaptively recomputing the road map are described, and in Section 5 results from experiments are presented. We finally conclude in Section 6.

2 PROBLEM FORMULATION

We consider the problem of coordinating the execution of delivery tasks by a team of autonomous robots, e.g., the transportation of crates containing goods, between a set of fixed stations $\mathcal{S}$. For each delivery task $d^{kl} \in \mathcal{D}(t)$ a robot has to be assigned to finalize the delivery by transporting the corresponding crate from station $k \in \mathcal{S}$ to station $l \in \mathcal{S}$. We assume that the assignment problem has been solved (e.g. as shown in our previous work [14]), and hence restrict our attention to the problem of solving the multiple robot motion planning problem as defined in the following. Let $\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$ be the set of n robots navigating simultaneously on a two-dimensional grid map. During planning, each robot has a start configuration $s_i \in \mathcal{C}_{free}$ and a goal configuration $g_i \in \mathcal{C}_{free}$, where $\mathcal{C}_{free}$ is the subset of configurations robots can take on without colliding with static obstacles. Note that in our case these configurations directly map to locations and orientations on the discrete grid map which are collision free given the footprint of the robot.

Figure 2: The target system: Robots equipped with convoyer and RFID reader for autonomously handling transportation tasks: (a) approaching a station for loading. (b) safe navigation among humans.

The problem is to compute for each robot $R_i \in \mathcal{R}$ a path $\pi_i : [0, T_i] \rightarrow \mathcal{C}_{free}$ such that $\pi_i(0) = s_i$ and $\pi_i(T_i) = g_i$ which is free of collisions with the trajectory π_j of any other robot $j \neq i$. Note that T_i denotes the individual path length of robot R_i.

We consider environments with dynamic obstacles such as pallets and larger crates that might change their locations over time. Therefore, $\mathcal{C}_{free}$ is a function of time which we denote by $\mathcal{C}_{free}(t)$. Note that we assume that $\mathcal{C}_{free}$ is static during each planning cycle.

3 SYSTEM OVERVIEW

Our system is based on the KARIS (Kleinskalige Autonomes Redundantes Intralogistiksystem) [7] platform developed by a joint effort of several companies and universities of the "Intralogistic Network" in south Germany. The long-term goal of this project is to deploy hundreds of these elements to solve tasks in intra-logistics and production, such as autonomously organizing the material flow between stations. The element has a size of 50×50 cm, a payload of 60 kg, and is capable to recharge its batteries via contact-less rechargers let into the ground. Furthermore, it contains a high precision mechanism for enabling automatic docking maneuvers, either with other elements or a loading station. Each element is equipped with a holonomic drive to facilitate docking behaviors and a conveyor for loading and unloading crates when docked with a loading station. The convoyer has an integrated RFID reader for directly reading from the crates their destination, e.g. the target station ID, when they are placed on the conveyer.

For the purpose of autonomous navigation the element is equipped with two SICK S300 laser range finders (LRFs) mounted in two opposing corners, wheel odometry, and an inertial measurement unit (IMU). Navigation is based on grid maps, which are generated from data collected by once steering a single robot manually through the environment. We use Monte-Carlo localization [6] with wheel odometry, IMU, and range readings from the two LRFs for localizing robots on the grid map. Furthermore, the typical hybrid architecture is deployed consisting of two components, which are a deliberative planning layer based on the grid map and a reactive safety layer based on LRF data directly. Figure 2 depicts the demonstration of the system during the *Logimat* fair in Stuttgart 2010. At the current stage, the system is capable of

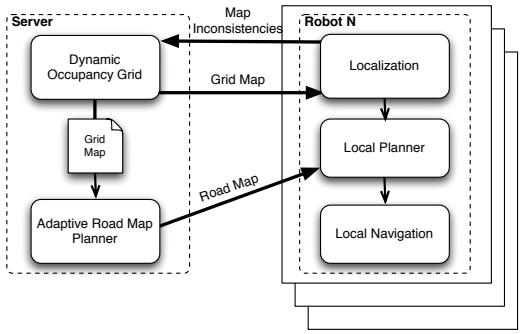

Figure 3: System Overview

safe autonomous navigation in human workspaces for team sizes of up to four robots.

The work presented in this paper has the goal to extend the planning system for the simultaneous navigation of large robot teams in dynamically changing environments. Figure 3 depicts the overall system architecture and modules of the considered extension. The localization module reports inconsistencies between sensor observations and the current grid map to the *Dynamic Occupancy Grid Module* which computes an updated version of the grid map [11]. The updated grid map is published to the localization module of each robot, and also to the *Adaptive Road Map Planner* (see Section 4) that computes a new road map, which is then published to the local planner of each robot. The local planner computes then based on the road map a path that is executed by the navigation module. The overview does not contain the mechanism for task allocation, i.e., to assign robots to delivery taks. In the current system this task is solved by the *contract net protocol* [12], however, also more sophisticated approaches, such as the one presented in our previous work [14] can be deployed.

4 ADAPTIVE ROADMAP PLANNER

In this section we describe the procedure for computing the adaptive road map given a dynamic occupancy grid map, a set of stations $s \in \mathcal{S}$, where $loc(s)$ denotes the location (x_s, y_s) of station s on the grid map, and a set of *delivery tasks* $\mathcal{D}$, where each $d^{kl} \in \mathcal{D}$ requires the routing of packages from station $k \in \mathcal{S}$ to station $l \in \mathcal{S}$.

4.1 Computation of the connectivity network

Our goal is to compute a road map that is optimal in terms of efficiency and compactness for the simultaneous routing of robots executing delivery tasks. For this purpose we first compute the Voronoi graph [4] from the dynamic grid map, which then serves as a basis for computing the connectivity network $\mathcal{C} = (V, E)$ consisting of nodes $v \in V$ that correspond either to station locations $loc(s)$ or crossings, and edges $e \in E$ that connect all stations and crossings on the map. The computation of $\mathcal{C}$ is carried out by three steps. First, we determine for each tuple $(i, j) \in \mathcal{S} \wedge i \neq j$ the set of alternative paths A_{ij} connecting station i and j on the Voronoi graph. Second, according to the method described in [5], we replace each A_{ij} by orthogonal straight lines (either horizontal or vertical) under the constraint that they have to be within a minimum safety distance to obstacles including the maximal extent of robots from their rotational center. Third, we add all straight lines to E while merging parallel lines if they exceed the double size of the robots. Besides station locations $loc(s)$, for each crossing line a node is created and added to V. Finally, we compute for each $e_{ij} \in E$ the maximal number of possible lanes w_{ij} for this connection according to the distance to the nearest obstacle, and the time needed to travel this segment c_{ij} according to its length.

4.2 Definition of the LP problem

Based on the connectivity network $\mathcal{C}$, we define our logistics problem similar to the *minimum cost flow problem* [1], however, with the difference that the number of lanes in both directions between two nodes and thus the capacities are variable. The goal is to find a network structure by which packages are optimally routed between the stations in the network. At each time there exists a set of simultaneous *delivery tasks* $d^{kl} \in \mathcal{D}(t)$ that require the routing of packages from station $k \in \mathcal{S}$ to station $l \in \mathcal{S}$. We denote by $b^{kl} = b(d^{kl})$ the requested throughput rate, i.e., the amount of packages per minute that have to be delivered from station k to station l.

Given the connectivity network C, we associate with each edge a cost c_{ij}, the maximal number of lanes w_{ij} allowed in the real world, and the capacity of a single lane connection u_{ij}. Whereas the cost c_{ij} expresses the time needed to travel from i to j, capacity u_{ij} expresses the maximal number of robots that can travel on this connection via a single lane at the same time without causing congestions. The number of lanes in both directions between two nodes i and j is expressed by the decision variables y_{ij} and y_{ji}, respectively. For example, $y_{ij} = 2, y_{ji} = 1$ denotes a single lane connection from node j to node i and a double lane from node i to node j. The quantity w_{ij} constraints the set of possible assignments to y_{ij} and y_{ji} according to the space available in the the real world. For example, if $w_{ij} = 4$, then some of the possible assignments are $(0, 0)$, $(0, 1)$, $(1, 0)$, $(2, 1)$, $(1, 2)$, $(2, 2)$, ... In general, it has to be assured that $y_{ij} + y_{ji} \leq w_{ij}$. Note that there exists the same limit in both directions and thus $w_{ij} = w_{ji}$.

The decision variables x_{ij}^{kl} define the flow assigned to an edge due to the delivery from k to l. The total flow x_{ij} has to be bigger or equal to zero and below the maximal flow $u_{ij}y_{ij}$, where u_{ij} is the capacity of a single lane and y_{ij} the number of activated lanes.

We associate for each delivery task d^{kl} the requested throughput $b(i)$ with the respective station nodes i. For each node $i \in V$, $b(i) = b^{kl}$ if $i = k$, i.e., vertex i is a source, and $b(i) = -b^{kl}$ if $i = l$, i.e., vertex i is a sink. All other nodes for which $b(i) = 0$ are functioning as transition nodes. The problem formulation can then be stated as follows:

$$\text{Minimize} \sum_{(i,j)} \sum_{k} \sum_{l} c_{ij} x_{ij}^{kl} + \sum_{(i,j)} u_{ij} y_{ij} \tag{1}$$

subject to:

$$\sum_{j:(j,i)} x_{ij}^{kl} - \sum_{j:(i,j)} x_{ji}^{kl} = \begin{cases} -b^{kl}(i) & (i = k) \quad \forall\, i,k,l \\ b^{kl}(i) & (i = l) \quad \forall\, i,k,l \\ 0 & otherwise. \end{cases} \tag{2}$$

$$y_{ij} + y_{ji} \leq w_{ij} \quad \forall\, (i,j), \tag{3}$$

$$x_{ij}^{kl} \geq 0 \quad \forall\, (i,j), k, l, \tag{4}$$

$$\sum_{k} \sum_{l} x_{ij}^{kl} \leq u_{ij} y_{ji} \quad \forall\, (i,j), k, l \tag{5}$$

$$\sum_{j:(i,j)} x_{ji}^{kl} \leq C_{max} \quad (i \neq k \wedge i \neq l) \quad \forall\, i,k,l \tag{6}$$

Equation 1 minimizes over the total travel costs and the physical space occupied by the road network. Equation 2 enforces the flow conservation in the network, i.e., the summed flow from all incoming edges $j : (j,i)$ and all outgoing edges $j : (i,j)$ has to be equal $-b_{kl}$ if i is a sink, b_{kl} if i is a source, and zero otherwise. Equation 3, Equation 4, and Equation 5 are constraining the maximal number of lanes, minimal and maximal flow, respectively. Finally, Equation 6 ensures that the total flow through crossings does not exceed the maximal crossing capacity C_{max} which depends on the spacial size of crossings, i.e., how many robots can be located there at the same time. Note that delivery tasks for which the node operates as source or sink have no influence on the capacity.

The above formulation can efficiently be solved by linear programming solvers, such as CPLEX, when defining the decision variables x_{ij}, y_{ij} by continuous values and rounding up the y_{ij} from which then the road map can directly be constructed. Furthermore, we yield for each delivery task d^{kl} a subset of edges from the road map having positive flow assignments $x_{ij}^{kl} > 0$. These quantities are directly utilized by the local planner (see Section 3) for extracting individual robot plans by finding the shortest path on the road map by the following successor state expansion: For each node i, we perform random sampling over all outgoing edges weighted according to their normalized flow values x_{ij}^{kl}. If there exists only one edge with $x_{ij}^{kl} > 0$ for node i, the edge is expanded directly. Finally, the local navigation module follows this plan while coordinating locally at crossings with other robots when needed.

5 EXPERIMENTAL RESULTS

The system has been tested in several different environments. Figure 4 depicts some of these environments that were used for the results presented in this paper. The *PLANT* map has a size of $51m \times 56m$, the *ASE* map a size of $94m \times 82m$, and the KNO map a size of $88m \times 43m$. On each map we defined locations of loading stations: 8 on PLANT, 16 on ASE, and 8 on KNO.

The robot platform shown in Figure 2 has been presented during the *Logimat* fair in Stuttgart, 2010, where the task of

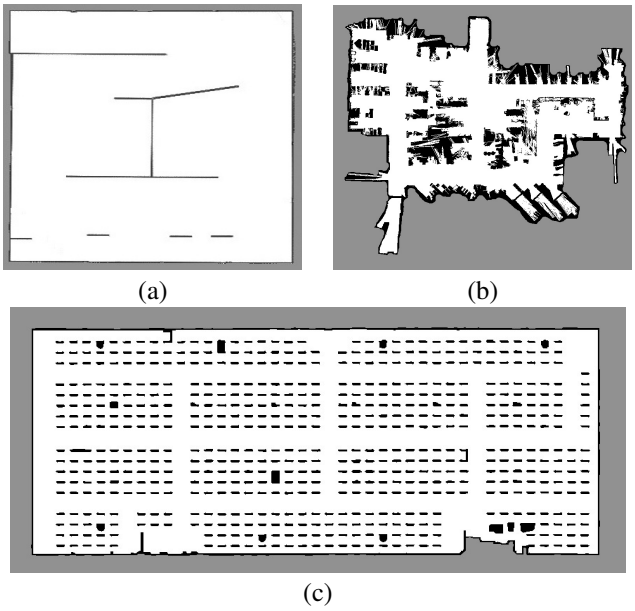

Figure 4: Grid maps utilized for experiments: (a) the *PLANT* map generated from a simulated environment, (b) the *ASE* map generated in a real logistic center, (c) the *KNO* map generated in a large distribution center.

the robot team was to deliver freshly prepared coffee cups to visitors waiting at the delivery stations, and to return used cups back to the coffee kitchen. During this demonstration up to four robots were continuously running for three days without any interruption. The robots were driving in average four kilometers per day without causing collisions or deadlocks. Due to the small team size we utilized for this demonstration a decoupled planning technique together with the local navigation module. In the following a comparison with large robot teams between ARMO and the decoupled technique based on prioritized planning from Berg and colleagues [15] will be presented. In prioritized planning, robot trajectories are planned iteratively after a pre-defined priority scheme. When planning for the i'th robot trajectories of the i-1 robots that were planned previously are considered as dynamic obstacles. Berg and colleagues define the *query distance* as the distance for each robot to reach its goal configuration on the shortest path when ignoring the other robots. In order to minimize the maximum of arrival times, priorities are assigned according to this distance: the longer the query distance the higher the priority assigned to a robot. The planner is complete under the assumption that start and goal locations of each robot are so called *garage configurations*, i.e., configurations that are not part of $\mathcal{C}_{free}$ of any other robot. The method efficiently avoids the intractable computation of $n!$ possible priority schemes, however, requires at least $|\mathcal{R}|$ sequential calls of the motion planner.

We utilized the Stage software library [17] for simulating large robot teams. In our experiments we used the same navigation software that is used on the real robots together with a model of our real platform, including the simulation of laser beams and odometry. One advantage of Stage is that it allows

Map	#Rob.	Method	# C	(m/s)	CTime (s)
ASE	20	ARMO	1432	0.47	969
		PRIO	2142	0.43	926
	50	ARMO	5040	0.37	545
		PRIO	10625	0.28	631
	100	ARMO	8307	0.3	369
		PRIO	16983	0.2	496
PLANT	20	ARMO	1471	0.42	628
		PRIO	1346	0.38	610
	50	ARMO	5563	0.34	426
		PRIO	11601	0.25	481
	100	ARMO	15145	0.22	383
		PRIO	107700	0.21	874
KNO	20	ARMO	506	0.38	1383
		PRIO	5638	0.35	1346
	50	ARMO	2951	0.31	815
		PRIO	70799	0.16	1371
	100	ARMO	7729	0.29	513
		PRIO	102836	0.11	1167

Table 1: Comparing prioritized planning (PRIO) with adaptive road map optimization (ARMO).

to build simulation worlds directly from grid maps that were generated from real environments. For the following experiment we used the grid maps shown in Figure 4.

We generated 100 delivery tasks for each map that were handled by 20, 50, and 100 robots during different runs. Table 1 provides the results from comparing prioritized planning (PRIO) with adaptive road map optimization (ARMO) on different maps with different numbers of robots. We measured the number of conflicts (C) of the optimal path in $\mathcal{C}_{free}$ with trajectories of the other robots. In the case of ARMO these were the situations in which a robot had to wait for other robots before entering a segment, and in the case of prioritized planning these were the situations were robots had to plan around a conflicting path of a higher prioritized robot. Furthermore, we measured the average velocity of all robots (avg. v) and the total time needed by all robots to complete the task (CTime). As can be seen from Table 1 and Figure 5, while leading to slightly longer completion times for small robot teams, ARMO notably reduces this time when the team size increases. This is also reflected by the number of conflicts and the average velocities of the robots. Prioritized planning minimizes the final completion time after a heuristically determined order, whereas LP-based planning in ARMO minimizes the global flow of robots, leading to a more efficient distribution of the vehicles over time.

The computation times of both methods were measured in seconds on an Intel DualCore running at 2.13 GHz. We measured for prioritized planning with 50 robots an average computation time of 0.03 ± 0.03 on PLANT, 0.05 ± 0.04 on ASE, and 0.1 ± 0.17 on KNO, and with 100 robots 0.1 ± 0.08 on PLANT, 0.13 ± 0.08 on ASE, and 1.1 ± 0.7 on KNO. ARMO required for the road map computation $0.9 + 0.6$ on PLANT, $0.82 + 0.84$ on ASE, and $1.2 + 10.3$ on KNO, where the first number denotes the time for extracting the fully connected graph, and the second number the time for solving the

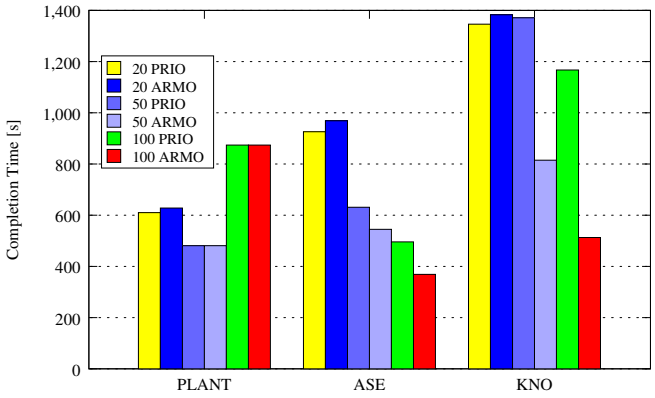

Figure 5: Comparing the CTime of prioritized planning (PRIO) and adaptive road map optimization (ARMO).

LP problem. Within each planning cycle ARMO needed in average only 0.002 on PLANT, 0.004 on ASE, and 0.01 on KNO for any number of robots. In summary, the number of robots has nearly no effect on the computation time needed by ARMO, however, we measured a significant growth of the time needed by prioritized planning. On the contrary, ARMO requires much more time for computing the road map when the environment is very large and complex, such as the KNO map, which however needs only to be performed at low frequency, i.e., when the environment was significantly changed.

We also evaluated ARMO with respect to dynamic changes of the grid map. For this purpose we modified the ASE map step wise by adding successively obstacles that were updated in the map by the dynamic occupancy grid approach. Figure 6 depicts two snapshots taken at successive points in time. As can be seen, the road map adjusts to the changes at the cost of higher completion times. For 100 robots the completion time increased from 378s (no modifications) to 410s (first modification) and 420s (second modification). We performed several more experiments for evaluating the adaptivity of our approach. Also after changing the distribution of delivery tasks between the stations, the road map dynamically adjusted by removing or adding links between the stations. Note that in this case only the LP solver is restarted without re-computing the connectivity network $\mathcal{C}$.

6 CONCLUSION

We proposed an adaptive road map planner based on a linear programming formulation which can be used for motion planning of large robot teams in dynamically changing environments. Experimental results have shown that ARMO leads to more efficient multi-robot plans than decoupled techniques while keeping the demand for computational resources low. In fact, the computation time needed by ARMO depends mainly on the complexity of the environment rather than on the number of robots. We believe that the computation of the road map could further be improved by splitting the map into independent areas that are interconnected via *fixed* crossing points similar to the stations. Then, only a part of the road map would have to be recomputed after local changes have been detected.

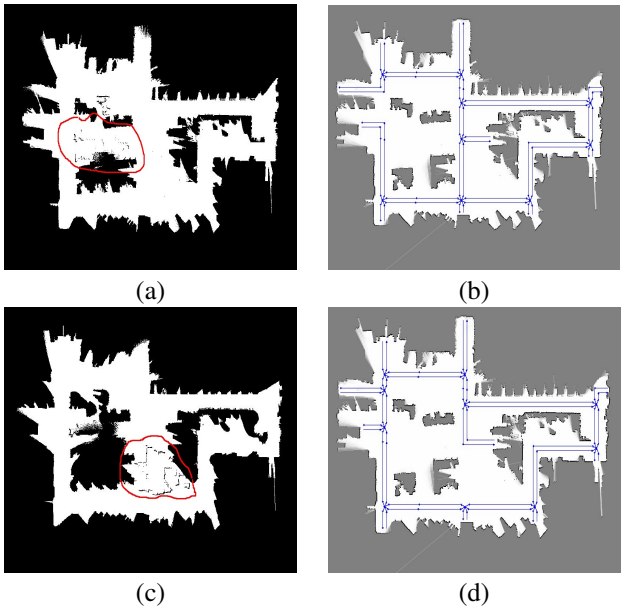

Figure 6: Adjustment of the road map according to dynamic changes in the map (a,c) source of disturbance and (b,d) resulting modifications reflected in the road map.

Furthermore, we have shown that ARMO is adaptive to dynamic changes in the map, i.e., the road map is reconstructed accordingly, whereas changes in the map are detected by dynamic grid maps, an extension of conventional grid maps.

We conducted several more experiments and conclude that our method is capable to efficiently solve a wide variety of problems. One restriction of our current implementation is the fact that our road map planner only returns a solution when the overall throughput demanded by the stations can be routed given the environmental constraints, i.e., does not exceed the capacity of the network. One future extension will be to introduce priorities for deliveries and to construct the network from a subset of tasks sampled according to their priority in case the requested throughput is higher than the capacity of the network. Furthermore, when a larger number of real robots is available, ARMO will be used with the real platform deployed in one of the logistic centers of our partners.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications*, volume 1. Englewood Cliffs, N.J.: Prentice Hall, 1993.

[2] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International journal of robotics research*, 10:628–649, 1991.

[3] J.S. Bellingham, M. Tillerson, M. Alighanbari, and J.P. How. Cooperative path planning for multiple uavs in dynamic and uncertain environments. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on*, volume 3, pages 2816 – 2822 vol.3, 2002.

[4] H. Choset, , and Burdick J. Sensor-based exploration: The hierarchical generalized voronoi graph. *The International Journal of Robotics Research*, 19(2), 2000.

[5] Xavier Décoret and François X. Sillion. Street Generation for City Modelling. In *Architectural and Urban Ambient Environment*, Nantes France, 2002.

[6] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 1998.

[7] H. Hippenmeyer, K. Furmans, T. Stoll, and F. Schönung. Ein neuartiges Element für zukünftige Materialflusssysteme. *Hebezeuge Fördermittel: Fachzeitschrift für Technische Logistik*, (6), 2009.

[8] M. Kallman and M. Mataric. Motion planning using dynamic roadmaps. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, volume 5, pages 4399–4404, 2004.

[9] L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566 –580, August 1996.

[10] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[11] D. Meyer-Delius, J. Hess, G. Grisetti, and W. Burgard. Temporary maps for robust localization in semi-static environments. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, Taipei, Taiwan, 2010.

[12] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1981.

[13] A. Sud, E. Andersen, S. Curtis, M.C. Lin, and D. Manocha. Real-time path planning in dynamic virtual environments using multiagent navigation graphs. *Visualization and Computer Graphics, IEEE Transactions on*, 14(3):526 –538, 2008.

[14] D. Sun, A. Kleiner, and C. Schindelhauer. Decentralized hash tables for mobile robot teams solving intra-logistics tasks. In *Proc. of the 9th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 923–930, Toronto, Canada, 2010.

[15] J.P. van den Berg and M.H. Overmars. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots & Systems (IROS)*, pages 430–435, 2005.

[16] J.P. van den Berg and M.H. Overmars. Roadmap-based motion planning in dynamic environments. *Robotics, IEEE Transactions on*, 21(5):885–897, 2005.

[17] R. Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2):189–208, 2008.

[18] P. Velagapudi, K. Sycara, and P. Scerri. Decentralized prioritized planning in large multirobot teams. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4603–4609. IEEE.