Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots

Cap, Michal; Novak, Peter; Kleiner, Alexander; Selecky, Martin; Pechoucek, Michal

DOI
10.1109/TASE.2015.2445780

Publication date
2015

Document Version
Final published version

Published in
IEEE Transactions on Automation Science and Engineering

Prioritized Planning Algorithms for Trajectory Coordination of Multiple Mobile Robots

Michal Čáp, Peter Novák, Alexander Kleiner, and Martin Selecký

***Abstract*—In autonomous multirobot systems one of the concerns is how to prevent collisions between the individual robots. One approach to this problem involves finding coordinated trajectories from start to destination for all the robots and then letting the robots follow the preplanned coordinated trajectories. A widely used practical method for finding such coordinated trajectories is "classical" prioritized planning, where robots plan sequentially one after another. This method has been shown to be effective in practice, but it is incomplete (i.e., there are solvable problem instances that the algorithm fails to solve) and it has not yet been formally analyzed under what circumstances is the method guaranteed to succeed. Further, prioritized planning is a centralized algorithm, which makes the method unsuitable for decentralized multirobot systems.**

The contributions of this paper are: a) an adapted version of classical prioritized planning called *revised prioritized planning* with a formal characterization of a class of instances that are provably solvable by this algorithm and b) an asynchronous decentralized variant of both classical and revised prioritized planning together with a formal analysis showing that the algorithm terminates and inherits completeness properties from its centralized counterpart.

The experimental evaluation performed in simulation on real-world indoor maps shows that: a) the revised version of prioritized planning reliably solves a wide class of instances on which both classical prioritized planning and popular reactive technique ORCA fail and b) the asynchronous decentralized implementation of classical and revised prioritized planning finds solution in large multirobot teams up to 2x-faster than the previously proposed synchronized decentralized approach.

***Note to Practitioners*—Consider a large warehouse in which the goods are stored and retrieved by autonomous mobile robots. One way to deal with possible collisions between the robots is to ignore interactions between the vehicles during the route planning for each robot and handle the conflicts only during the route execution. However, such an approach is prone to deadlocks, i.e., to situations during which some of the robots mutually block each other, cannot proceed and fail to complete their transportation task. An alternative approach would involve planning collision-free routes for each robot before the robots start executing them. However, the current methods that guarantee ability to find a solution to any such coordination problem are not applicable in practice due to their high computational complexity. Instead, a simple and computationally efficient approach in which robots plan their routes sequentially one after another (classical prioritized planning) is often used for finding coordinated trajectories even though the algorithm is known to fail on many dense problem instances. In this paper, we show that a simple adaptation of this classical algorithm called revised prioritized planning is guaranteed to find collision-free trajectories for a well-defined class of practical problems. In particular, if the system resembles human-made transport infrastructures by requiring that the start and destination position of each vehicle must never obstruct other vehicles from moving, then the proposed approach is guaranteed to provide a solution. For instance, in our warehouse multirobot system example, the collision-free routes can be efficiently computed by the revised prioritized planning approach. This paper formally characterizes the problem instances for which the method is guaranteed to succeed.**

Further, we propose a new asynchronous decentralized adaptation of both classical and revised prioritized algorithm that can be used in multirobot systems without a central solver. This technique can be used to find coordinated trajectories just by running a simple asynchronous negotiation protocol between the individual robots. This paper provides an analysis showing that the asynchronous decentralized implementations of classical and revised prioritized planning exhibit desirable theoretical properties and an experimental comparison of performance of different variations of centralized and decentralized prioritized planning algorithms.

***Index Terms*—Collision avoidance, decentralized algorithms, multirobot systems, trajectory planning.**

I. Introduction

In RECENT years, several successful demonstrations of production-quality mobile robots, autonomous unmanned aerial vehicles (UAVs), and self-driving cars fueled excitement about the future opportunities offered by autonomous multivehicle systems both for transportation of goods and people. Clearly, the efficiency and safety of such systems will depend on the existence of guaranteed methods for reliable collision avoidance between the individual vehicles. As an example of an autonomous multivehicle system, consider a factory where intermediate products are moved between workplaces by autonomous robots. The worker at a particular workplace calls a robot, puts an object to a basket mounted on the robot and orders the robot to autonomously deliver the object to another workspace where the object will be retrieved by another worker. An important requirement in such a system is that each robot must be able to avoid collisions with other robots that autonomously carry out their tasks at the same floor. The

Manuscript received August 31, 2014; revised January 06, 2015, May 06, 2015, and May 20, 2015; accepted May 21, 2015. Date of publication June 29, 2015; date of current version July 17, 2015. This paper was recommended for publication by Associate Editor F. Ehlers and Editor L. Sabattini upon evaluation of the reviewers' comments. This work was supported in part by the Czech Science Foundation under Grant 13-22125S) and in part by the Grant Agency of the Czech Technical University in Prague under Grant SGS14/143/OHK3/2T/13.

M. Čáp and M. Selecký are with the Department of Computer Science, Czech Technical University in Prague, Praha 121 35, Czech Republic (e-mail: cap@agents.fel.cvut.cz; martin.selecky@agents.fel.cvut.cz).

P. Novák is with the Department of Software and Computer Technology, Delft University of Technology, Delft 2628 CC, Netherlands (e-mail: p.novak@tudelft.nl).

A. Kleiner is with iRobot, Pasadena, CA 91125 USA (e-mail: alexander.kleiner@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2015.2445780

problem of avoiding collisions between individual robots can be approached from a control engineering perspective by employing reactive collision avoidance and from AI perspective by planning coordinated trajectories for the robots.

In the reactive approach, the robot follows the shortest path to its current destination and attempts to resolve collision situations as they appear. Each robot periodically observes positions and velocities of other robots in its neighborhood. If there is a potential future collision, the robot attempts to avert the collision by adjusting its immediate heading and velocity. A number of methods have been proposed [4], [7], [14] that prescribe how to compute such collision-avoiding velocity in a reciprocal multirobot setting, with the most prominent one being ORCA [13]. These approaches are widely used in practice thanks to their computational efficiency—a collision-avoiding velocity for a robot can be computed in a fraction of a millisecond [13]. However, they resolve collisions only locally and therefore they cannot guarantee that the resulting motion will be deadlock-free and that the robot will always reach its destination.

In the planning approach, a multirobot motion planner is used that takes into consideration the start and goal position of each robot and plans coordinated trajectories for all robots that are mutually conflict-free. If the robots track the resulting joint trajectories precisely (or within some given tolerance that has been accounted for during the planning phase), it is guaranteed that the robots will reach their goals without a collision. However, it is known that even the simplest variants of the multirobot path planning problem are computationally intractable. Finding coordinated collision-free paths for multiple circular robots moving amidst polygonal obstacles is known to be strongly NP-hard [10]; the same task involving rectangular robots in an empty room is known to be in PSPACE-hard [5]. This implies that it is possible to construct instances of coordination problems that are not efficiently solvable under the current computation paradigm.

The existing multirobot planners are typically based either on the coupled heuristic search in the joint state space of all robots or on decoupled planning. The coupled approaches are able to find optimal solution [11], [12], [19], but their worst-case time complexity grows exponentially with the number of robots involved in the conflict, which makes them impractical for coordination of more than a few robots.

On the other hand, decoupled approaches plan independently for each robot. They can be fast enough for real-time applications, but they typically suffer from incompleteness, i.e., there are solvable coordination problems that these algorithms fail to solve. A widely used decoupled scheme for the multirobot motion planning that has been shown to be effective in practice is prioritized planning [3]. In prioritized planning, each robot is assigned a unique priority and the algorithm proceeds sequentially from the highest-priority robot to the lowest priority one. In each iteration, one of the robots plans its trajectory such that it avoids the higher priority robots. Such a greedy approach is clearly incomplete if we allow arbitrary maps and arbitrary start and goal locations for each robot, but in relatively sparse environments, the technique tends to perform well. However, it has not yet been formally studied under what circumstances is prioritized planning guaranteed to provide a valid coordinated solution.

Recently, Velagapudi *et al.* presented a decentralized version of prioritized planning technique for teams of mobile robots [18], which is able to utilize the distributed computational resources to reduce the time needed to find a solution. However, since the algorithm proceeds in globally-synchronized rounds, faster-computing robots have to wait at the end of each round for the longest-computing robot and thus the distributed computational power is not used efficiently.

The contribution of this paper is twofold. First, we propose a revised version of the prioritized planning scheme and show that for this version it is possible to formulate sufficient conditions under which the algorithm is guaranteed to provide a solution. Second, we propose a novel asynchronous decentralized variant of both classical and revised prioritized planning scheme that is guaranteed to terminate and inherits completeness properties from the respective centralized counterpart. We experimentally show that asynchronous decentralized algorithm exhibits better utilization of the distributed computational resources and thus provides faster convergence times compared to the previously presented synchronized approach.

Partial results of the presented work appeared in [2] and [17], where the focus was on the design of an asynchronous version of the decentralized prioritized planning. Here, we extend our previous work by proposing a revised version of prioritized planning scheme, by theoretical analysis of the properties of all discussed algorithms, by performing experimental comparison on real-world indoor maps using idealized simulation and by including reactive techniques into the comparison.

II. Problem Definition

Consider n circular robots operating in a 2-D workspace $\mathcal{W} \subseteq \mathbb{R}^2$. The subset of $\mathcal{W}$ occupied by the body of a robot i when its center is on position x is denoted as $R_i(x)$. The maximum speed robot i can move at is denoted as v_i. Each robot is assumed to be assigned a *task* that involves moving from its start position s_i to some goal position g_i and stay there. We assume that the start and goal positions of all robots are mutually disjunct, i.e., the bodies of robots do not overlap when the robots are on their start positions and when they are on their goal positions.

A path $p : [0, 1] \to \mathcal{W}$ of robot i in workspace $\mathcal{W}$ is called *satisfying* if it starts at the robot's start position s_i, ends at robot's goal position g_i, and the body of the robot whose center follows the path p always lies entirely in $\mathcal{W}$. A trajectory $\pi : [0, \infty) \to \mathcal{W}$ is a mapping from time points to positions in workspace and unlike a path, it carries information about how it should be executed in time. Analogically, a trajectory of robot i is called *satisfying* if it starts at the robot's start position s_i, finally reaches and stays at the goal position g_i, the body of robot i whose center follows the trajectory π always lies entirely in $\mathcal{W}$, and the robot never moves faster than its maximum speed v_i.

The trajectories π_i, π_j of two robots i, j are said to be *conflict-free* if and only if the bodies of the robots i, j never intersect when they follow the trajectories π_i and π_j.

Problem 1 (Trajectory Coordination Problem): Given a workspace $\mathcal{W}$ and tasks $\langle s_1, g_1\rangle, \ldots, \langle s_n, g_n\rangle$ for robots $1, \ldots, n$, find trajectories $\pi_1, \ldots, \pi_n$ such that each trajectory

π_i is satisfying for robot i and trajectories π_i, π_j of every two different robots i, j are mutually conflict-free.

1) Notation: The following shorthand notations will be used to denote regions occupied by a different subsets of robots at their start and goal positions:

$$S^i := R_i(s_i) \qquad G^i := R_i(g_i)$$
$$S^{>i} := \bigcup_{j=i+1,\ldots,n} R_j(s_j) \qquad G^{<i} := \bigcup_{j=1,\ldots,i-1} R_j(g_j)$$
$$S := \bigcup_{j=1,\ldots,n} R_j(s_j) \qquad G := \bigcup_{j=1,\ldots,n} R_j(g_j).$$

Further, we will work with the concept of a space–time region: When a spatial object, such as the body of a robot, follows a given trajectory, then it can be thought of as occupying a certain region in space–time $\mathcal{T} := \mathcal{W} \times [0, \infty)$. A dynamic obstacle Δ is then a region in such a space–time $\mathcal{T}$. If $(x, y, t) \in \Delta$, then we know that the spatial position (x, y) is occupied by the dynamic obstacle Δ at the time t. The function

$$R_i^{\Delta}(\pi) := \{(x, y, t) : t \in [0, \infty) \wedge (x, y) \in R_i(\pi(t))\}$$

maps trajectories of a robot i to regions of space–time that robot i occupies when its center point follows given trajectory π. As a special case, let $R_i^{\Delta}(\emptyset) := \emptyset$.

A. Assumptions on Communication

We assume that each robot is equipped with an independent computation unit and a wireless device for communication with other robots. Wireless communication channels are typically implemented as broadcast channels, where each communicated message is broadcast, but ignored by the nodes that are not among the declared recipients of the message. In such a channel a single broadcast message uses the same channel capacity as a single point-to-point message and thus we will prefer to perform a single broadcast instead of sending several point-to-point messages. Further, in the following discussion, we will assume that such a communication channel is reliable, i.e., each broadcast messages is eventually received by all robots in the system, and that the communication channel preserves the ordering of messages that were sent in.

III. Prioritized Planning

A straightforward approach to solve trajectory coordination problems is to see all robots in the system as one composite robot with many degrees of freedom and use some path planning algorithm to find a joint path for all the robots. However, the size of the joint configuration space that has to be searched during the planning is exponential in the number of robots and thus this approach quickly becomes impractical if one wants to plan for more than a few robots. A pragmatic approach that is often useful even for large multirobot teams is prioritized planning. The idea of prioritized planning has been first articulated by Erdman and Lozano–Perez in [3]. Since the quality of the returned solution depends on the order in which the robots plan, later works such as [1] and [15] focused on heuristics for choosing a suitable priority sequence for the robots.

Algorithm 1: Classical Prioritized Planning

```
1 Algorithm PP
2     Δ ← ∅;
3     for i ← 1...n do
4         π_i ← Best-traj(W, Δ);
5         if π_i = ∅ then
6             report failure and terminate
7         Δ ← Δ ∪ R_i^Δ(π_i);
8 Function Best-traj(W', Δ)
9     return optimal satisfying trajectory for robot i in W'
      that avoids regions Δ if it exists, otherwise return ∅
```

A. Classical Prioritized Planning

In classical prioritized planning each robot is assigned a unique priority. The trajectories for individual robots are then planned sequentially from the highest-priority robot to the lowest priority one. For each robot a trajectory is planned that avoids both the static obstacles in the environment and the higher priority robots moving along the trajectories planned in the previous iterations.

Algorithm 1 lists the pseudocode of classical prioritized planning. The algorithm iterates over the robots, starting from the highest-priority robot 1 to the lowest-priority robot n. During ith iteration the algorithm computes a trajectory for robot i that avoids the space–time regions occupied by robots $1, \ldots, i-1$. The trajectory of robot i is computed in `Best-traj` $(\mathcal{W}', \Delta)$ function, which returns a trajectory for robot i such that the body of the robot stays inside the static workspace $\mathcal{W}'$ and avoids dynamic regions Δ occupied by other robots. Such a function would be in practice implemented using some application-specific technique for motion planning with dynamic obstacles, e.g., [8] or [16]. For the reasons that will be discussed later, it is desirable that this function is realized using an algorithm offering some form of completeness, since this property will be inherited also by the multirobot algorithm.

B. Properties

Classical prioritized planning terminates either with success or with failure in at most n iterations. The successful termination occurs in exactly n iterations if in each iteration i a valid trajectory for robot i has been found. The termination with failure occurs if there exists a robot for which no satisfying trajectory that avoids higher priority robots can be found.

When the algorithm terminates successfully, each robot is assigned the computed trajectory, which is conflict-free with respect to the trajectories of the other robots. This follows from the fact that the trajectory planned for each robot i is conflict-free with the higher priority robots, since it was planned to avoid collision with them and also with the lower priority robots since their trajectories were planned to avoid collisions with the trajectory of robot i.

Prioritized planning is in general incomplete, consider the counter-example [9] depicted in Fig. 1.

Let us now analyze when the classical prioritized planning algorithm is bound to fail. The algorithm fails to find a trajectory for robot i if: 1) no satisfying path exists for robot i, i.e., the robot cannot reach its destination even if there are no other robots in the workspace; 2) every satisfying trajectory of robot

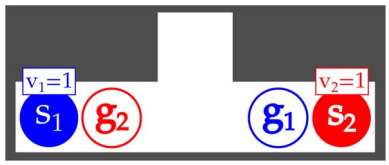

Fig. 1. **Incompleteness of classical prioritized planning**: In this scenario, robot 1 travels from s_1 to g_1 and robot 2 travels from s_2 to g_2 in a corridor that is only slightly wider than a body of a single robot. Both robots can travel at identical maximum speeds. Observe that irrespective of which robot starts planning first, its trajectory will be in conflict with all satisfying trajectories of the robot that plans as the second.

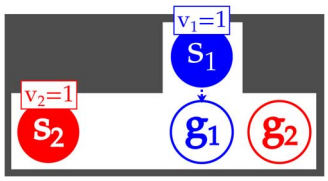

Fig. 2. **Type A conflict**: Robot 1 travels from s_1 to g_1 and robot 2 travels from s_2 to g_2. The gray areas represent obstacles. Both robots can travel at identical maximum speed. The trajectory for robot 1 is planned first and the straight line trajectory from s_1 to g_1 at the maximum speed is found. Consequently, all satisfying trajectories of robot 2 are in Type A conflict with robot 1.

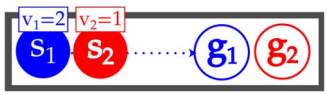

Fig. 3. **Type B conflict**: Robot 1 travels from s_1 to g_1 and robot 2 travels from s_2 to g_2. The gray areas represent obstacles. Robot 1 can travel *twice as fast* as robot 2. The trajectory for robot 1 is planned first and the straight line trajectory from s_1 to g_1 at the maximum speed is found. The trajectory for robot 2 is planned second, but all satisfying trajectories for robot 2 are in Type B conflict with robot 1 and thus trajectory planning for robot 2 fails.

i is in conflict with some higher priority robot. There are two types of conflicts that can occur between a satisfying trajectory π of robot i and a higher priority robot:

Type A: Occurs if the trajectory π is in conflict with a higher priority robot which has reached and is "sitting" at its destination, i.e., it is blocked by a static higher priority robot. Fig. 2 depicts a scenario in which all satisfying trajectories of one of the robots are in Type A conflict.

Type B: Occurs if the trajectory π of robot i is in conflict with a higher priority robot which is moving towards its destination, i.e., it is "run over" by a moving higher priority robot. Fig. 3 shows a scenario in which all satisfying trajectories of one of the robots are in Type B conflict.

A question that naturally arises is whether it is possible to restrict the class of solvable instances or to alter the classical prioritized planning algorithm in such a way that there is always at least one trajectory without neither Type A nor Type B conflict for each robot.

One way to ensure that there is a satisfying trajectory without Type A conflict for every robot is to only consider instances, where each robot has a path to its goal that avoids the goal regions of all higher priority robots. When each robot follows such a path, then they cannot be engaged in a Type A conflict, because the Type A conflict can only occur at the goal region of one of the higher priority robots.

Unfortunately, the existence of a trajectory without Type B conflict is difficult to guarantee in classical prioritized planning algorithm, because interactions with lower priority robots are completely ignored during trajectory planning at each iteration of the algorithm. If we want to ensure that each robot has a satisfying trajectory without Type B conflict, the trajectories of all higher priority robots have to be planned such that the lower-priority robots are always left with some alternative trajectory that can be used to avoid the potential conflicts of this type.

One way to ensure that there is a satisfying trajectory without Type B conflict for every robot is to consider only instances in which each robot has a path to its goal that avoids start region of lower priority robots and enforce that the trajectory of each robot will avoid the start regions of all lower priority robots. When this is ensured, then any robot always has a fall-back option to wait at its start position (since no higher priority robot can run over its start region) until its desired path is clear of the all higher priority robots. Thus, it can always avoid Type B conflicts.

Moreover, if the robot continues by following a path that avoids the goal regions of higher priority robots, then the resulting trajectory is also guaranteed to avoid Type A conflicts.

The result of the above analysis can be intuitively summarized as follows: If we have a trajectory coordination problem instance in which every robot can reach its goal without crossing: a) the regions occupied by the lower priority robots at their start positions and b) the regions occupied by the higher priority robots at their goal positions, then such an instance can be in the worst-case resolved by moving the robots sequentially, one after another, to their destinations. The following theorem states this property formally.

Theorem 2: Let us have a trajectory coordination problem with workspace $\mathcal{W}$ and tasks $\langle s_1, g_1\rangle, \ldots, \langle s_n, g_n\rangle$ for robots $1, \ldots, n$. If for every robot i there exists an $S^{>i}$-avoiding and $G^{<i}$-avoiding satisfying path, then a sequential conflict-free solution can be constructed.

Proof: We will construct the solution inductively as follows:

Induction assumption: The trajectories of robots $1, \ldots, i-1$ are satisfying and $S^{>i-1}$-avoiding.

Base step (robot 1): Robot 1 is the highest-priority robot. There are no higher priority robots that robot 1 needs to avoid. From our assumption there exists a path p that is satisfying for robot 1 and $S^{>1}$-avoiding. A satisfying and $S^{>1}$-avoiding trajectory for robot 1 can be simply constructed by following the path p at an arbitrary positive speed. Such a trajectory can always be constructed, therefore the algorithm will not report failure when planning for robot 1.

Induction step (robot i): From our assumption there exists a path p that is satisfying for robot i, $S^{>i}$-avoiding and $G^{<i}$-avoiding. Since all the trajectories for robots $1, \ldots, i-1$ are satisfying (i.e., eventually reach the goal and stay there), there must exists a time point $\overline{t}$ after which all robots $1, \ldots, i-1$ have reached and will stay at their goal. A satisfying and $S^{>i}$-avoiding trajectory for robot i that is conflict-free with all robots $1, \ldots i-1$ can be constructed as follows:

- In interval $[0, \overline{t}]$ stay at s_i. The trajectory cannot be in conflict with the higher priority robots during this interval, be-

Algorithm 2: Revised Prioritized Planning

```
1 Algorithm RPP
2     Δ ← ∅;
3     for i ← 1...n do
4         S ← ⋃_{j>i} S^j
5         π_i ← Best-traj(W \ S, Δ);
6         if π_i = ∅ then
7             report failure and terminate;
8         Δ ← Δ ∪ R_i^Δ(π_i);
```

cause all trajectories of robots $1, \ldots, i-1$ are $S^{>i-1}$ and thus also S^i-avoiding.

- In interval $[\overline{t}, \infty]$ follow path p until the goal position g_i is reached. The path p avoids regions $G^{<i}$ and thus the trajectory cannot be in collision with any of the higher priority robots $1, \ldots, i-1$ because they are at their goal positions during this time interval, which the path p avoids.

Such a trajectory can always be constructed.

The trajectories of robots $1, \ldots, i-1$ are satisfying and $S^{>i-1}$-avoiding, which implies that they are also $S^{>i}$-avoiding. The newly computed trajectory for robot i is satisfying and $S^{>i}$-avoiding. By taking the union of the old set of trajectories and the new trajectory, we have a set of trajectories for robots $1, \ldots, i$ that are satisfying and $S^{>i}$-avoiding. ■

As we can see from the constructive proof of Theorem 2, the instances that admit $S^{>i}$-avoiding and $G^{<i}$-avoiding paths for each robot can be solved by a simple sequential algorithm that navigates all robots one after another along their shortest $S^{>i}$-avoiding and $G^{<i}$-avoiding paths. Albeit simple, such an approach never lets two robots move concurrently and thus it typically generates solutions of poor quality. The solution quality can be improved if we adopt the prioritized planning approach and find for each robot a best $S^{>i}$-avoiding trajectory that avoids conflicts with higher priority robots.

C. Revised Prioritized Planning

Using insights for the preceding discussion, we propose a Revised version of Prioritized Planning (RPP) in which a trajectory for each robot is sought so that both: a) start position of all lower priority robots are avoided and b) conflicts with higher priority robots are avoided. The pseudocode of RPP is listed in Algorithm 2.

D. Properties

The RPP algorithm inherits the termination and soundness properties from the PP algorithm. The algorithm terminates successfully after n iterations if a trajectory for each robot has been found. The algorithm terminates with a failure at iteration i if there is a robot i for which a satisfying trajectory in $\mathcal{W} \setminus S$ has not been found.

In general, it is not guaranteed that a trajectory that avoids both start positions of lower priority robots and regions occupied by higher priority robots will exist for each robot. Thus, the algorithm may fail to provide a solution to a solvable problem instance. Consider the example in Fig. 1 once again. However, for instances characterized by the following condition, the solution is guaranteed to exists and RPP will find it.

Corollary 3: If there is a $S^{>i}$-avoiding, $G^{<i}$-avoiding satisfying path for every robot i and a complete algorithm is used for the single-robot trajectory planning in `Best-traj` function, then RPP is guaranteed to terminate with a conflict-free solution.

Proof: Consider the inductive argument from the proof of Theorem 2. The argument states that at every iteration, there exists a $S^{>i}$-avoiding satisfying trajectory for robot i that avoids all higher priority robots. Since the single-robot planning algorithm is assumed to be complete, it cannot fail in finding such a trajectory. ■

E. Well-Formed Infrastructures

In this section, we introduce a class of multirobot systems that can be coordinated using the RPP algorithm in a guaranteed way. Consider a situation when one designs a closed multirobot system such, as a warehouse, where mobile robots store and retrieve goods. In such systems Theorem 2 and Corollary 3 can be exploited to design the environment and the allowed tasks of the robots in a way that $S^{>i}$-avoiding and $G^{<i}$-avoiding paths will always exists and RPP will be consequently guaranteed to provide a conflict-free solution to all possible trajectory coordination queries. An important class of systems that satisfy the condition of having a $S^{>i}$-avoiding and $G^{<i}$-avoiding paths for every possible task of every robot are *well-formed infrastructures*:

To model systems such as warehouses, factories, rail roads, road networks, etc., we introduce the notion of an infrastructure. An infrastructure is a pair $(\mathcal{W}, E)$, where $\mathcal{W}$ is a workspace (described by a set of obstacle-free coordinates $\mathcal{W} \subseteq \mathbb{R}^2$) and a set of points $E \subset \mathcal{W}$ represents distinguished locations in the environment called endpoints (modeling, e.g., storage locations in a warehouse, workplaces in a factory, parking places, road stops, etc.). Vehicles operating in such an infrastructure are assumed to only move between the endpoints of the infrastructure, i.e., $\forall i : s_i \in E$ and $g_i \in E$.

A *well-formed* infrastructure has its endpoints distributed in such a way that any robot standing on an endpoint cannot completely prevent other robots from moving between any other two endpoints. In a well-formed infrastructure, a robot is always able to find a collision-free trajectory to any other unoccupied endpoint by waiting for other robots to reach their destination endpoint, and then by following a path around the occupied endpoints, which is in a well-formed infrastructure guaranteed to exist.

In the following, we will describe the idea more formally. First, let us introduce the necessary notation. Let $D(x, r)$ be a closed disk centered at x with radius r. Then, $\mathrm{int}_r X := \{x : D(x, r) \subseteq X\}$ is an r-interior of a set $X \subseteq \mathbb{R}^2$.

Definition 4: An infrastructure $(\mathcal{W}, E)$ is called *well-formed* for circular robots having body radii $r_1, \ldots, r_n$ if any two endpoints $a, b \in E$ can be connected by a path in workspace $\mathrm{int}_{\overline{r}}(\mathcal{W} \setminus \bigcup_{e \in E \setminus \{a,b\}} D(e, \overline{r}))$, where $\overline{r} = \max\{r_1, \ldots, r_n\}$.

That is, there must exist a path between any two endpoints with at least $\overline{r}$-clearance with respect to static obstacles and at least $2\overline{r}$-clearance to any other endpoint. See Fig. 4 for an illustration.

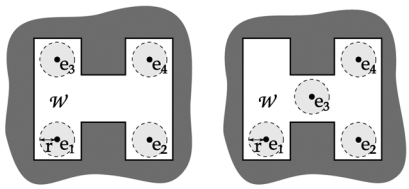

Fig. 4. Well-formed and ill-formed infrastructure. (a) Well-formed infrastructure: The workspace $\mathcal{W}$ and endpoints $\{e_1, e_2, e_3, e_4\}$ for robots having radius r form a well-formed infrastructure. (b) Ill-formed infrastructure: The workspace $\mathcal{W}$ and endpoints $\{e_1, e_2, e_3\}$ do not form a well-formed infrastructure because there is no path from e_1 to e_2 with $2r$-clearance to e_3 for a robot having radius r.

The notion of well-formed infrastructures follows the structure typically witnessed in man-made environments that are intuitively designed to allow efficient transit of multiple persons or vehicles. In such environments, the endpoint locations where people or vehicle stop for long time are separated from the transit area that is reserved for travel between these locations.

In a road network, for example, the endpoints would be the parking places. The system of roads is then built in such a way that any two parking places are reachable without crossing any other parking place. Similar structure can be witnessed in offices or factories. The endpoints would be all locations, where people may need to spend longer periods of time, e.g., surroundings of the work desks or machines. As we know from our every day experience, work desks and machines are typically given enough free room around them so that a person working at a desk or a machine does not obstruct people moving between other desks or machines. We can see that real-world environments are indeed often designed as well-formed infrastructures.

Suppose a particular workspace $\mathcal{W}$ and a set of endpoints E form a well-formed infrastructure for robots having radius r. An instance of multirobot trajectory coordination problem in such an infrastructure would then involve a number of robots traveling from one endpoint to another so that each endpoint is used by at most one robot. From the well-formed infrastructure property, we know that for each robot there is a path p from its start endpoint to its goal endpoint that avoids all other endpoints in the infrastructure with $2r$ clearance. Since all other robots' start and goal positions lie at endpoints, path p is also avoiding start and goal position of every other robot. In other words, for every robot there is a path that is S^{-i}-avoiding and G^{-i}-avoiding, which implies that the path is also $S^{>i}$-avoiding and $G^{<i}$-avoiding. Therefore, any trajectory coordination query in which all robots move between endpoints of a well-formed infrastructure will be successfully solved by the RPP algorithm, given that a complete algorithm is used for the single robot trajectory planning.

F. Checking Solvability

In some systems, it can be hard or even impossible to ensure that all coordination queries will always take place between endpoints of a well-formed infrastructure. An obvious case are systems that cannot be structured as a well-formed infrastructure. Another example are systems that can be structured as well-formed infrastructures, but in which the precise tracking of trajectories cannot be guaranteed due to unexpected events, e.g., when a robot has to yield to a human who crosses its path. After such an unexpected interruption, a new coordinated motion for all robots must be found from the robots' current positions to their destinations. However, at the time of interruption some of the robots can be outside of endpoints. Yet another example is a well-formed infrastructure to which a new robot with a particular task is added and has to be coordinated with the other robots that are executing their previously planned coordinated trajectories. In such a scenario, the motion of all robots has to be interrupted and the coordinated trajectories need to be computed for all the robots from their current positions. Again, such positions may be in general outside of an endpoint.

In these scenarios, the start positions of some of the robots may be outside of an endpoint and thus RPP is not guaranteed to succeed. Nevertheless, Theorem 2 and Corollary 3 can be exploited to quickly determine if the resulting coordination problem is guaranteed to be solvable by RPP, without actually running the algorithm. This can be determined by verifying that a $S^{>i}$-avoiding and $G^{<i}$-avoiding satisfying path exists for each robot, which amounts to planning n spatial paths amidst static obstacles. We note that existence of a spatial path among static obstacles can be verified significantly faster than finding a spatio-temporal trajectory amidst dynamic obstacles (which has to be done if RPP is executed), because in the latter case the search space is augmented with an extra time dimension. Then, in the case of a negative result, the system can decide to reject or delay the adding of a new robot task, switch to a different coordination algorithm, or ask a human operator to resolve the coordination situation.

G. Limitations

We have shown that there is a class of instances that RPP completely covers, but PP does not. See Fig. 5 for an example of such an instance. However, outside this class we can find instances that PP solves, but RPP does not. For an example, consider the scenario depicted in Fig. 6. None of the algorithm is therefore superior to the other in terms of instance coverage.

Further, since RPP avoids start regions preemptively, even when they can be safely passed through, the solutions generated by RPP tend to be slightly longer than the ones generated by PP. This is demonstrated in Fig. 7.

We can see that despite the theoretical guarantees of RPP, there exist situations in which PP can be a more appropriate choice than RPP.

IV. Decentralized Algorithms

Consider a multirobot system consisting of tens or hundreds of heterogeneous autonomous robots. In such a scenario, a decentralized implementation of (revised) prioritized planning may be more desirable than a centralized one. In a decentralized implementation, each robot runs its own instance of the algorithm and exchanges messages with the other robots according to a prescribed communication protocol. If an inconsistency is detected by a robot, it recomputes the best trajectory for itself using its own on-board computation resources. The process should eventually converge to a state where all robots hold mutually conflict-free trajectories.

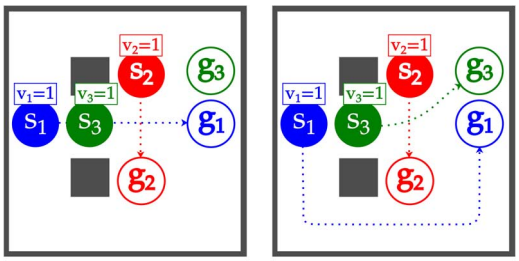

Fig. 5. **Scenario, where PP fails, but RPP succeeds**. Robot 1 travels from s_1 to g_1, robot 2 travels from s_2 to g_2, robot 3 travels from s_3 to g_3. The gray areas represent obstacles. All robots can travel at an identical maximum speed. **Left**: In PP, the straightline trajectory between s_1 and g_1 at maximum speed is found for robot 1. Similarly, the straight line trajectory at maximum speed is found also for robot 2. Consequently, all satisfying trajectories for robot 3 are in Type B conflict with robot 1 or robot 2 and thus trajectory planning for robot 3 fails. **Right**: In RPP, a trajectory that avoids the region occupied by robot 3 at its start position is found for robots 1 and 2. This allows robot 3 to avoid collisions with robot 2 by waiting at its start position and then by following the shortest path to g_3.

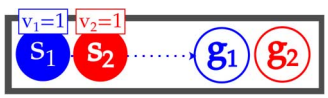

Fig. 6. **Scenario, where PP succeeds, but RPP fails**. Robot 1 travels from s_1 to g_1, robot 2 travels from s_2 to g_2. The gray areas represent obstacles. Assume that both robots can travel at the same maximum speed. Robot 1 searches for a trajectory that avoids the start position of robot 2; such a trajectory does not exist and thus RPP finishes with failure. Note that PP will successfully find a solution to this instance: Robot 1 will plan the trajectory that follows the straight line at maximum speed. Robot 2 will search for a trajectory that avoids the trajectory of robot 1 and finds that it suffices to travel at maximum speed to its goal g_2.

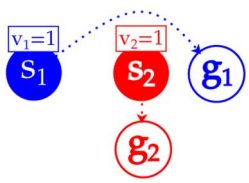

Fig. 7. **Scenario, where PP finds a higher quality solution than RPP**. Robot 1 travels from s_1 to g_1, robot 2 travels from s_2 to g_2. There are no obstacles in this scenario. When RPP searches for a trajectory for robot 1 it has to avoid the start position of robot 2, resulting in the curved trajectory as depicted in the picture. On the other hand, PP generates a shorter straightline trajectory connecting start and destination of robot 1.

An advantage of such an approach is that several robots can compute their trajectories in parallel and thus a conflict-free solution can be computed faster. Another advantage for multi-robot systems with heterogeneous robots is that the kinematic and other potentially implicit constraints on the trajectory of a particular robot remain local to that robot and do not need to be formalized or communicated, which simplifies the design of the communication protocol and allows each robot to use a custom robot-specific planner for planning its trajectory.

A. Synchronized Decentralized Implementation

A decentralized implementation of the classical prioritized planning scheme, where robots concurrently proceed in synchronized rounds, has been first presented by Velagapudi *et al.* [18]. We use their approach as a baseline decentralized implementation of both classical and revised prioritized planning and denote the resulting algorithm as *synchronized decentralized implementation of prioritized planning*, SD-PP and *synchronized decentralized implementation of revised prioritized planning SD-RPP*. The SD-(R)PP abbreviation is used when a statement holds for both variants.

Algorithm 3: Synchronized Decentralized Implementation of (Revised) Prioritized Planning. Pseudocode for robot i

```
1  Algorithm SD-(R)PP
2      π_i ← ∅;
3      H_i ← ∅;
4      S ← { ∅            for SD-PP
           { ⋃_{j>i} S^j   for SD-RPP ;
5      repeat
6          π* ← Find-consistent(π_i, W \ S, Δ(H_i));
7          if π* = ∅ then
8              report failure and terminate;
9          else if π* ≠ π_i then
10             π_i ← π*;
11             broadcast INFORM(i, R_i^Δ(π*));
12         wait for INFORM messages from all other
           robots, wait for all other robots to finish
           processing INFORM messages ;
13     until not global termination detected;
14 Handle-message INFORM(j, Δ_j)
15     if j < i then
16         H_i ←
           (H_i \ {(j, Δ'_j) : (j, Δ'_j) ∈ H_i}) ∪ {(j, Δ_j)};
17 Function Find-consistent(π, W, Δ)
18     if π = ∅ ∨ ¬consistent_i(π, Δ) then
19         π* ←Best-traj_i(W, Δ);
20         return π*;
21     else
22         return π;
```

The algorithm proceeds in synchronized rounds. In every round, each robot ensures that its current trajectory is consistent with the trajectories of higher priority robots from the previous round. If the current trajectory is consistent, then the robot keeps its current trajectory and remains silent. Otherwise, it finds a new consistent trajectory for itself and broadcasts the trajectory to all other robots. When a robot finishes its computation in the current round, it waits for all other robots to finish the round and all robots simultaneously proceed to the next round. The algorithm successfully terminates if none of the robots changes its current trajectory during a single round. The SD-PP algorithm terminates with failure if there is a robot that fails to find a trajectory that avoids the higher priority robots following their respective trajectories. The SD-RPP algorithm, on the other hand, finishes with failure if there is a robot that fails to find a satisfying trajectory that avoids the start positions of lower priority robots. The pseudocode of SD-(R)PP is listed in Algorithm 3.

In SD-(R)PP, each robot i maintains a database of space–time regions occupied by higher priority robots. We call such a database a trajectory store and model it as a set of pairs $H_i = \{(j, \Delta_j)\}$, where Δ_j is the space–time region occupied by robot j.

Function $\Delta(H)$ represents the region of the space–time occupied by all robots stored in a trajectory store H

$$\Delta(H) := \bigcup_{(j,\Delta_j)\in H} \Delta_j.$$

Further, we use predicate $\text{consistent}_i(\pi, \Delta)$ to express that the trajectory π of robot i is collision-free against dynamic obstacles Δ, defined as

$$\text{consistent}_i(\pi, \Delta) := R_i^{\Delta}(\pi) \cap \Delta = \emptyset.$$

B. Properties

In order to facilitate and simplify the exposition of the later introduced asynchronous algorithm, we developed an alternative proof of termination of the SD-(R)PP algorithm, which deviates from the original one devised by the authors of SD-PP in [18]. Further, in this section, we show that SD-(R)PP inherits the soundness and completeness properties from its respective centralized counterpart.

The SD-(R)PP algorithm is guaranteed to terminate. First, we need to define what termination means for a decentralized algorithm. A decentralized algorithm:

- **terminates** when all robots stop computing;
- **terminates with a failure** if it terminates and there is at least one robot that reported a failure during the computation;
- **terminates successfully** if it terminates without a failure.

To show that SD-(R)PP terminates, we first show that robots running SD-(R)PP cannot exchange messages forever.

Proposition 5: All robots running the SD-(R)PP algorithm eventually stop sending INFORM messages.

Proof: We proceed by induction on the robot priority i.

Inductive hypothesis: Robots $1, \ldots, i-1$ eventually stop sending messages.

Base step (robot 1):

Robot 1 is the highest-priority robot and as such it does not receive any message from a higher priority robot. Therefore, its trajectory store will stay empty. During the initialization, robot 1 either successfully finds its initial trajectory and broadcasts a single message, or reports a failure and terminates. Since its trajectory store is empty, its initial trajectory will never become inconsistent, the robot will therefore never replan and send any further INFORM messages.

Induction step (robot i):

From the inductive hypothesis, we have that each of the robots $1, \ldots, i-1$ eventually stops broadcasting messages. After the last message from robots $1, \ldots, i-1$ has been received by robot i, its trajectory store gets updated for the last time, since from our assumption there are no more messages from higher priority robots. After the trajectory store changes for the last time, the robot either: a) keeps its current trajectory if it is consistent with the last trajectory store; b) finds a new consistent trajectory, or c) terminates with failure. In cases a) and b), the current trajectory will never become inconsistent again because the trajectory store does not change anymore and thus the robot will never have to replan and communicate a new trajectory. In case c), the robot has terminated the computation and thus it will not send any more INFORM messages in the future. We see that after robots $1, \ldots, i-1$ stopped sending messages, also robot i eventually ceases to send messages. ■

Corollary 6: SD-(R)PP always terminates.

Proof: We have that all robots in the system will eventually stop sending messages. Assume that the last message is broadcast during round k. In round $k+1$, no robot changes its trajectory, otherwise, a message would have to be broadcast which is a contradiction. If no robot changes its trajectory during the round, the global termination condition is satisfied and the system terminates. ■

Unless a failure is reported by one of the robots, the solution computed when SD-(R)PP terminates is sound.

Proposition 7: When SD-(R)PP successfully terminates, all robots hold trajectories that are mutually conflict-free.

Proof: Let π_i be the trajectory of robot i after the algorithm has terminated. We need to show that

$$\forall i, j : i \neq j \Rightarrow \pi_i \text{ and } \pi_j \text{ are conflict-free.}$$

Take two arbitrary, but different robots i,j. Since the conflict-free relation is symmetrical, we can assume $j < i$ w.l.o.g. If robot i stopped computing without a failure, it must have received the INFORM message from a higher priority robot j carrying its last trajectory π_j at some point before its termination. Since there are no further INFORM messages broadcast by robot j, the trajectory store of robot i will contain π_j from that point on. Every trajectory returned by `Find-consistent` function for robot i from that point on will be conflict-free with π_j and thus also its last trajectory π_i will be conflict-free with π_j. ■

The synchronized decentralized implementations of PP and RPP inherit completeness properties from the respective centralized implementations. In general, both SD-PP and SD-RPP are incomplete. However, if an $S^{>i}$-avoiding and $G^{<i}$-avoiding satisfying path exists for each robot, then the SD-RPP is guaranteed to terminate successfully.

Lemma 8: If there is a $S^{>i}$-avoiding and $G^{<i}$-avoiding satisfying path for every robot i and a complete algorithm is used for the single-robot trajectory planning in `Best-traj` function, then SD-RPP is guaranteed to terminate with a conflict-free solution.

Proof: The argument used in the proof of Theorem 3, which shows that RPP will never fail during planning, can be extended to decentralized implementations of RPP: Take an arbitrary replanning request for robot i. All trajectories of each higher priority robot $j < i$ have been generated to be $S^{>j}$-avoiding and thus such trajectory will be also S^{i}-avoiding. All trajectories in the trajectory store of robot i are therefore S^{i}-avoiding. An $S^{>i}$-avoiding satisfying trajectory consistent with trajectories of higher priority robots can be constructed as follows: Wait at the starting position s_i until all higher priority robots reach their goal position and then follow the $S^{>i}$-avoiding $G^{<i}$-avoiding satisfying path from the assumption. Since such a trajectory is guaranteed to exist for robot i and a complete replanning algorithm is used, the replanning cannot report failure. The algorithm must terminate with success. ■

C. Asynchronous Decentralized Implementation

Due to its synchronous nature, the SD-(R)PP algorithm does not fully exploit the computational resources distributed among

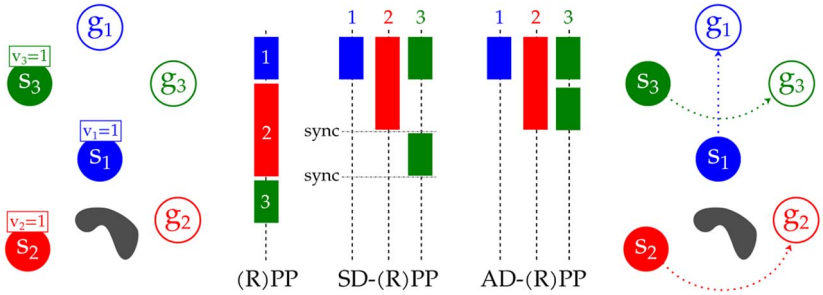

Fig. 8. Example problem in which AD-(R)PP converges faster than SD-(R)PP. **Left**: The task of robots 1, 2, and 3. Robot i travels from s_i to g_i. The gray area represents an obstacle. **Middle**: Sequence diagrams showing the planning process in (R)PP, SD-(R)PP, and AD-(R)PP. Suppose that robot 2 needs to plan longer, because it has to plan around the gray obstacle. In SD-(R)PP, robot 3 starts resolving the conflict with robot 1 only when robot 2 has finished computing its trajectory in the first round. In AD-(R)PP, robot 3 stars resolving the conflict with robot 1 immediately after it becomes aware of it, therefore it finds the solution faster. **Right**: The final solution.

Algorithm 4: Asynchronous Decentralized Implementation of (Revised) Prioritized Planning

Algorithm AD-(R)PP
  $\pi_i \leftarrow \emptyset$;
  $H_i \leftarrow \emptyset$;
  $S \leftarrow \begin{cases} \emptyset & \text{for AD-PP} \\ \bigcup_{j>i} S^j & \text{for AD-RPP} \end{cases}$;
  $\pi_i \leftarrow$ `Find-consistent` $(\pi_i, \mathcal{W} \backslash S, \Delta(H_i))$;
  if $\pi_i = \emptyset$ **then**
    report failure and terminate;
  else
    broadcast INFORM$(i, R_i^{\Delta}(\pi_i))$;
    wait for global termination;
Handle-message INFORM(j, Δ_j)
  if $j < i$ **then**
    $H_i \leftarrow (H_i \setminus \{(j, \Delta_j') : (j, \Delta_j') \in H_i\}) \cup \{(j, \Delta_j)\}$;
    $\pi^* \leftarrow$ `Find-consistent` $(\pi_i, \mathcal{W} \setminus S, \Delta(H_i))$;
    if $\pi^* = \emptyset$ **then**
      report failure and terminate;
    else if $\pi^* \neq \pi_i$ **then**
      $\pi_i \leftarrow \pi^*$;
      broadcast INFORM$(i, R_i^{\Delta}(\pi^*))$;

individual robots. In every iteration, the robots that finished their trajectory planning routine earlier, or did not have to replan at all, idle while waiting for the slower computing robots in that round. However, they could use the time to resolve some of the conflicts among themselves and speed up the overall process. An example of a situation, where the asynchronous algorithm would be beneficial is illustrated in Fig. 8.

To deal with such an inefficiency, we propose an asynchronous decentralized implementation of classical prioritized planning, abbreviated AD-PP, and revised prioritized planning scheme, abbreviated AD-RPP. The AD-(R)PP abbreviation is used when a statement holds for both variants.

The pseudocode of AD-(R)PP is shown in Algorithm 4. The asynchronous algorithm replaces the concept of globally synchronized rounds (while loop in Algorithm 3) by a reactive approach in which every robot reacts merely to incoming INFORM messages. Upon receiving an INFORM message (**Handle-message** INFORM(j, Δ_j) routine in Algorithm 4), the robot simply replaces the information about the trajectory of the sending robot in its trajectory store and checks whether its current trajectory is still consistent with the new contents of its trajectory store. If the current trajectory is inconsistent, the robot triggers replanning and informs other robots about its new trajectory. Otherwise, the robot keeps its current trajectory and remains silent.

D. Properties

AD-(R)PP inherits all the desirable properties from its synchronized and centralized counterparts, i.e., it terminates and if it terminates with success, then all the robots will hold conflict-free trajectories. Further, AD-RPP is guaranteed to solve instances that admit an $S^{>i}$-avoiding and $G^{<i}$-avoiding path for each robot.

Proposition 9: AD-(R)PP always terminates.

Proof: Recall that the inductive argument demonstrating that robots running SD-(R)PP will eventually stop sending messages (Lemma 5) does not make use of the synchronization points in SD-(R)PP and thus it is also valid for AD-(R)PP. By this argument, we know that there is a finite number of messages being sent. We can observe that a robot running AD-(R)PP performs computation only during initialization or when it processes an incoming message. When all robots process their last incoming messages, the system terminates. ■

Proposition 10: When AD-(R)PP successfully terminates, all robots hold trajectories that are mutually conflict-free.

Proof: The proof of soundness of SD-(R)PP (Proposition 7) is directly applicable also to AD-(R)PP. ■

Proposition 11: If an $S^{>i}$-avoiding, $G^{<i}$-avoiding satisfying path exists for each robot i and a complete algorithm is used for the single-robot trajectory planning in `Best-traj` function, then AD-RPP terminates.

Proof: The proof of Proposition 8, where this property is demonstrated to hold for SD-RPP, is directly applicable also for AD-RPP. ■

V. Experiments

In this section, we report the results of empirical comparison between algorithms discussed in the previous sections (i.e., PP, RPP, SD-PP, SD-RPP, AD-PP, and AD-RPP). The *effectiveness* of the algorithms is compared by: a) measuring their ability to successfully solve a set of randomly generated problem instances (i.e., instanceset coverage) in three real-world environments and b) measuring the quality of the returned solutions.

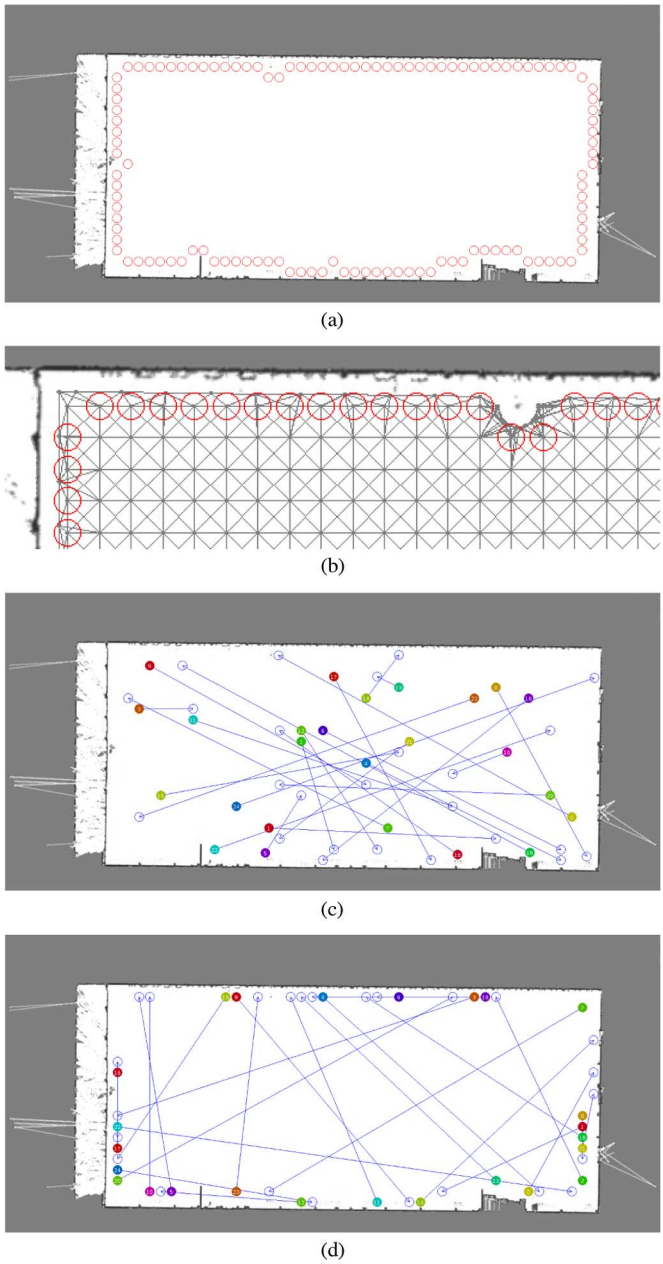

Fig. 9. Empty hall environment. (a) *Empty hall* environment. The endpoints used to create a well-formed infrastructure in the environment are shown as red circles. (b) A selected part of the roadmap used for trajectory planning. (c) Example instance with free-formed tasks for 25 robots. Task of each robot shown in blue. (d) Example instance with tasks for 25 robots in the well-formed infrastructure. Task of each robot shown in blue.

Their *efficiency* is compared a) by measuring the wall-clock time each algorithm requires to compute a valid solution and b) by counting the number of messages communicated by each algorithm.

These performance criteria are compared both a) on general "free-formed" instances that do not guarantee existence of $S^{>i}$-avoiding, $G^{<i}$-avoiding satisfying path for each robot and b) on instances, where robots move between endpoints of a well-formed infrastructure, i.e., on instances where existence of $S^{>i}$-avoiding, $G^{<i}$-avoiding satisfying path for each robot is guaranteed.

A. Environments

The comparison was performed in three real-world environments [see Fig. 9(a), 11(a), and 10(a)]. For each of the environments, we generated two sets of problem instances: a) In the *free-formed tasks* instance set, each robot is assigned a task to move from a randomly selected start position to a randomly selected goal position. b) In the *tasks in well-formed infrastructure* instance set, we generated a set of endpoints that together with a particular roadmap discretization of the environment form a well-formed infrastructure; each robot is then assigned a random endpoint as a start position and a randomly chosen endpoint as a destination position.

1) Empty Hall Environment: The map of the *empty hall* environment and the endpoints of the well-formed infrastructure embedded in the environment are shown in Fig. 9(a). The roadmap used during trajectory planning is depicted in Fig. 9(b). For both instance sets and each number of robots ranging from $n = 1$ to $n = 50$, we generated 50 random instances of the problem containing the given number of robots.

2) Office Corridor Environment: The map of the *office corridor* environment and the endpoints of the well-formed infrastructure embedded in the environment are shown in Fig. 11(a). The roadmap used during trajectory planning is depicted in Fig. 11(b). The map of the environment is based on the laser rangefinder log of the *Cartesium* building at the University of Bremen.[1] For both instance sets and each number of robots ranging from $n = 1$ to $n = 30$, we generated 50 random problem instances containing the given number of robots.

3) Warehouse Environment: The map of the *warehouse* environment and the endpoints of the well-formed infrastructure embedded in the environment are shown in Fig. 10(a). The roadmap used during trajectory planning is depicted in Fig. 9(b). For both instance sets and each number of robots ranging from $n = 1$ to $n = 60$, we generated 50 random problem instances containing the given number of robots. The tasks in the well-formed infrastructure instance set represent a scenario of an automated logistic center where robots move goods between the gates and the storage shelves.

4) Experiment Setup: For each problem instance, we create a simulated multirobot team and let all the robots coordinate the trajectories from the given start positions to given goal positions using each of the tested algorithms. For the centralized algorithms (PP and RPP), the robots communicate their start position and goal position to a central solver that uses the information to compute a coordinated solution on 1 CPU and, consequently, sends a message with the resulting trajectory to each robot. For the decentralized algorithms (SD-PP, SD-RPP, AD-PP, and AD-RPP), we assume that each robot uses its own on-board CPU to compute its trajectory. To measure the runtime characteristics of the execution of decentralized algorithms, we emulate the concurrent execution of the algorithms using a discrete-event simulation. The simulation measures the execution time of each message handling and uses the information to simulate the concurrent execution of the decentralized algorithm as if it were executed on n independent CPUs, where n is the

[1]We thank Cyrill Stachniss for providing the data through the Robotics Data Set Repository [6].

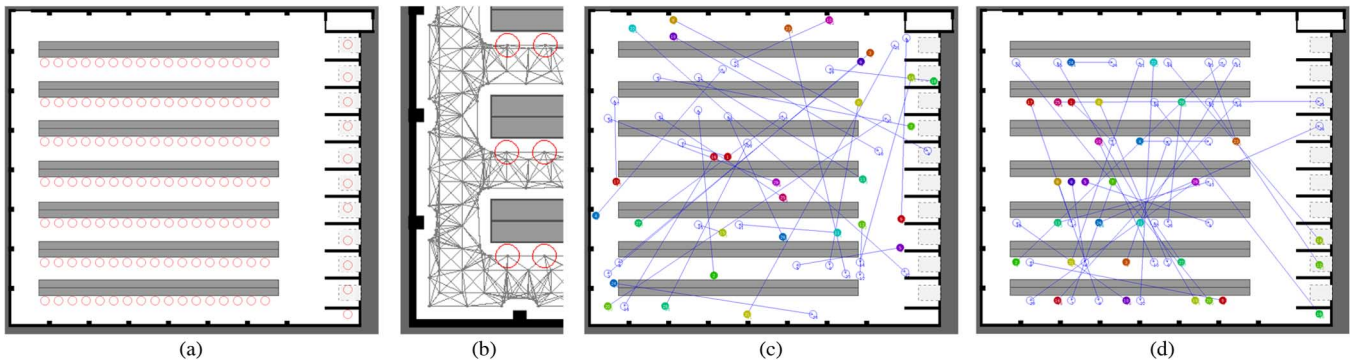

Fig. 10. Warehouse environment. (a) *Warehouse* environment. The endpoints used to create a well-formed infrastructure in the environment are shown as red circles. (b) A selected part of the roadmap used for trajectory planning. (c) Example instance with free-formed tasks for 30 robots. Task of each robot shown in blue. (d) Example instance with tasks for 30 robots in the well-formed infrastructure. Task of each robot shown in blue.

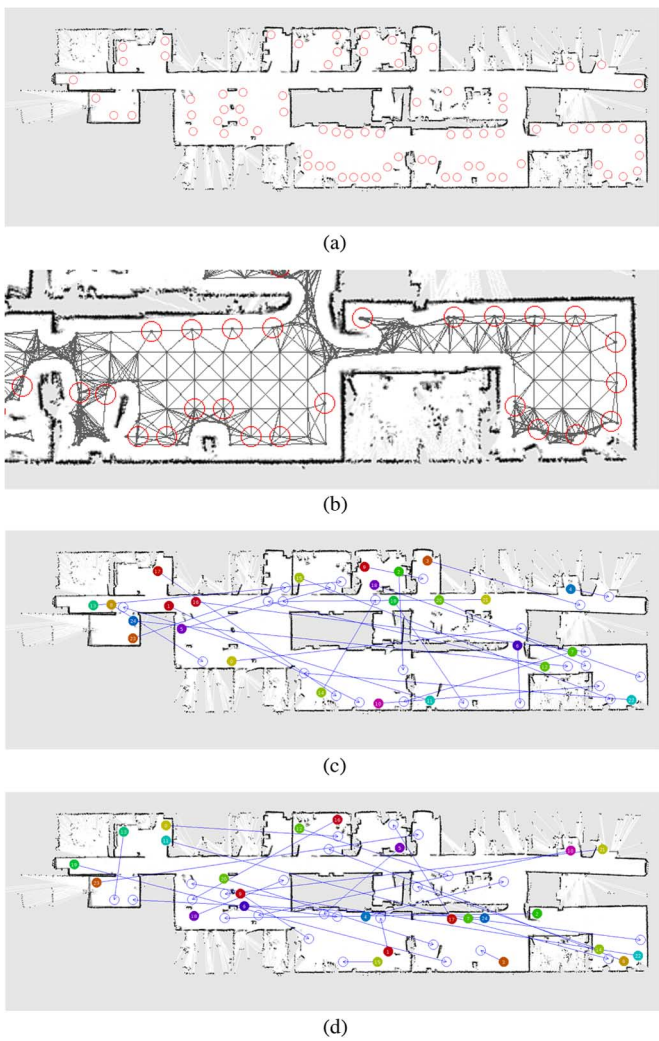

Fig. 11. Office corridor environment. (a) *Office corridor* environment. The endpoints used to create a well-formed infrastructure in the environment are shown as red circles. (b) A selected part of the roadmap used for trajectory planning. (c) Example instance with free-formed tasks for 25 robots. Task of each robot shown in blue. (d) Example instance with tasks for 25 robots in the well-formed infrastructure. Task of each robot shown in blue.

number of robots. The individual robots communicate via an idealized simulated communication channel modeled as a perfectly reliable channel with zero latency. All compared algorithms use an identical best trajectory planner. The best trajectory for each robots is obtained by searching a roadmap extended with a discretized time-dimension using the A* algorithm, where the heuristic function is the time to travel along the shortest path on the roadmap from the given node to the goal node when dynamic obstacles are ignored. The simulation of the decentralized system was implemented using the Alite multiagent simulation toolkit. The test instances, the simulator, and Java implementation of individual algorithms can be downloaded from http://agents.cz/~cap/adpp/. A video demonstrating the performance of ADPP and ADRPP algorithms on six selected instances is available at https://youtu.be/dFm-JJhyuv0.

The experiments have been performed on a computer with AMD Opteron 8356 2.3GHz CPU and 8 GB RAM. For each algorithm, we measure the following characteristics.

Coverage: We waited until each algorithm returns either a success or a failure and counted the number of instances each of the algorithm successfully solved.

The following average characteristics were computed on all instances that were *successfully solved* by all compared algorithms. To obtain reasonably precise estimate of the average, the values were computed only when there were at least ten such instances for a particular number of robots.

Avg. time to solution: We measured the wall-clock runtime needed to compute a solution. For the centralized planner we recorded the time of termination of the centralized planner. For the decentralized algorithms, we recorded the time when the last robot detected global termination of the computation.

Avg. speed-up: In order to be able to judge the effect of asynchronous execution of AD-(R)PP algorithm we also computed the speed-up ratio for both decentralized algorithms over their centralized counterparts. The speed-up for the algorithm A on an instance i is computed as

$$\frac{\text{runtime of centralized variant of alg. } A \text{ on instance } i}{\text{runtime of alg. } A \text{ on instance } i,}$$

where the centralized variant of AD-PP and SD-PP is PP, and the centralized variant of AD-RPP and SD-RPP is RPP.

Avg. messages sent: Every time a robot running a decentralized algorithm adopts a new trajectory (replans), the trajectory is broadcast to all other robots. Therefore, the number of INFORM messages sent by a robot directly corresponds to the number of replannings performed by the robot.

Avg. prolongation: The objective criterion we minimized is the sum of goal arrival times for each robot. We measured the duration of the trajectory for each robot and computed the prolongation coefficient for each instance as

$$\text{prolongation of alg. } A \text{ on instance } i = \frac{\sum_{i=1}^{n} t_i^A - t_i'}{\sum_{i=1}^{n} t_i'}, \quad (1)$$

where t_i^A is the time robot i needs to reach its goal position when it follows the trajectory computed by the algorithm A and t_i' is the time robot i would need to reach its goal following the shortest path on the roadmap if the collisions with other robots were ignored.

B. Comparison With Reactive Planning

In order to evaluate the practical advantages of the proposed planning approaches, we also compared their coverage against a popular reactive technique ORCA in our environments. When using ORCA, each robot continuously observes positions and velocities of other robots in a defined neighborhood. Should any potential collision be detected, a linear program is solved to obtain a new collision averting velocity that the robot should follow. If there are no eminent collisions, the robot follows its preferred velocity. In our implementation the preferred velocity vector points at the shortest path from the robot's current position to the goal.

C. Results

The plots showing the results of the comparison on instances with free-formed tasks are in Fig. 12. The results in the respective well-formed infrastructures are in Fig. 13.

D. Coverage

On instances with free-formed tasks, all tested algorithms exhibit incomplete coverage of the instance space. Generally, RPP-based algorithms solved fewer instances than the PP-based algorithms due to the requirement on an $S^{>i}$-avoiding path used in RPP. Unlike PP, RPP requires that each robot follows a path that avoids regions occupied by lower priority robots at their start position. Since the start positions for all robots were randomly generated, it can indeed happen that the generated start positions block some higher priority robot from reaching its goal. For an example instance where such a phenomena occurs consult Fig. 6.

For tasks in well-formed infrastructures, the existence of an $S^{>i}$-avoiding path for each robot is guaranteed and thus the RPP-based algorithms show full-instance coverage in accordance with our theoretical findings. Further, we can see [best in Fig. 13(a)-1] that some of those instances remain unsolved both by PP-based algorithms and ORCA.

Note that in the Office Corridor and Warehouse environments with free-formed tasks, all tested algorithms exhibit low success rates on instances with high number or robots. That is, it is increasingly hard to randomly generate an instance on which all tested algorithms succeed. Therefore, the following aggregate characteristics are only computed if there is enough data from successfully solved instances.

E. Time to Solution/Speed-Up

The asynchronous decentralized implementation of both PP and RPP consistently achieves higher speed-up than the synchronized implementation in accordance with our prediction. The higher speed-up is exhibited on instances with higher number of robots, where it is more likely that several independent conflict clusters will occur. On such instances, it is often beneficial that the conflict clusters can resolve conflicts between the individual robots at different pace and thus converge faster. The phenomena can be seen clearly in Figs. 12(a)-4, 13(a),-4 13(b)-4, and 13(a)-4.

F. Replannings/Communication

AD-(R)PP broadcast higher number of messages than SD-(R)PP. To see how this can be explained, suppose that at some point of the computation new conflicts arise between the trajectory of one particular robot and the trajectories of two other higher priority robots. If the two conflicts occur in a single round, SD-(R)PP solves both conflicts during one replanning at the end of the round and therefore broadcasts only a single INFORM message. However, in such a situation AD-(R)PP may need to replan twice because it triggers replanning immediately after each of the conflicts is detected and thus it will broadcast two INFORM messages.

G. Prolongation

There are two phenomena influencing the quality of returned solutions. First, RPP-based algorithms generate slightly longer trajectories than PP-based algorithms. This is due to the fact that RPP preemptively avoids start positions of the lower priority robots. Second, decentralized approaches generate slightly longer trajectories than the centralized approaches. The reason lies in the replanning condition used by the decentralized algorithms. The condition states that a robot should replan its trajectory only if the trajectory is inconsistent with the trajectories of other robots. Thus, the robot may receive an updated trajectory from a higher priority robot that allows an improvement in its current trajectory, but since its current trajectory may be still consistent, the robot will not exploit such an opportunity for an improvement.

H. Sensitivity of Results

The experiments were performed using a multirobot simulation that uses an idealized model of communication and trajectory tracking. The first issue is that in our model the message exchange among the robots is assumed to be reliable and instant. In the context of indoor multirobot systems, the reliable communication can be realized by covering the entire workspace by a wireless LAN network and ensuring that a reliable protocol on MAC or transport layer (e.g., TCP) is used. In other contexts, e.g., when robots communicate using ad-hoc peer-to-peer communication, such a reliable wireless network might not be available and the robots will face communication unavailability or message loss. In practice, the ADPP algorithm can be altered

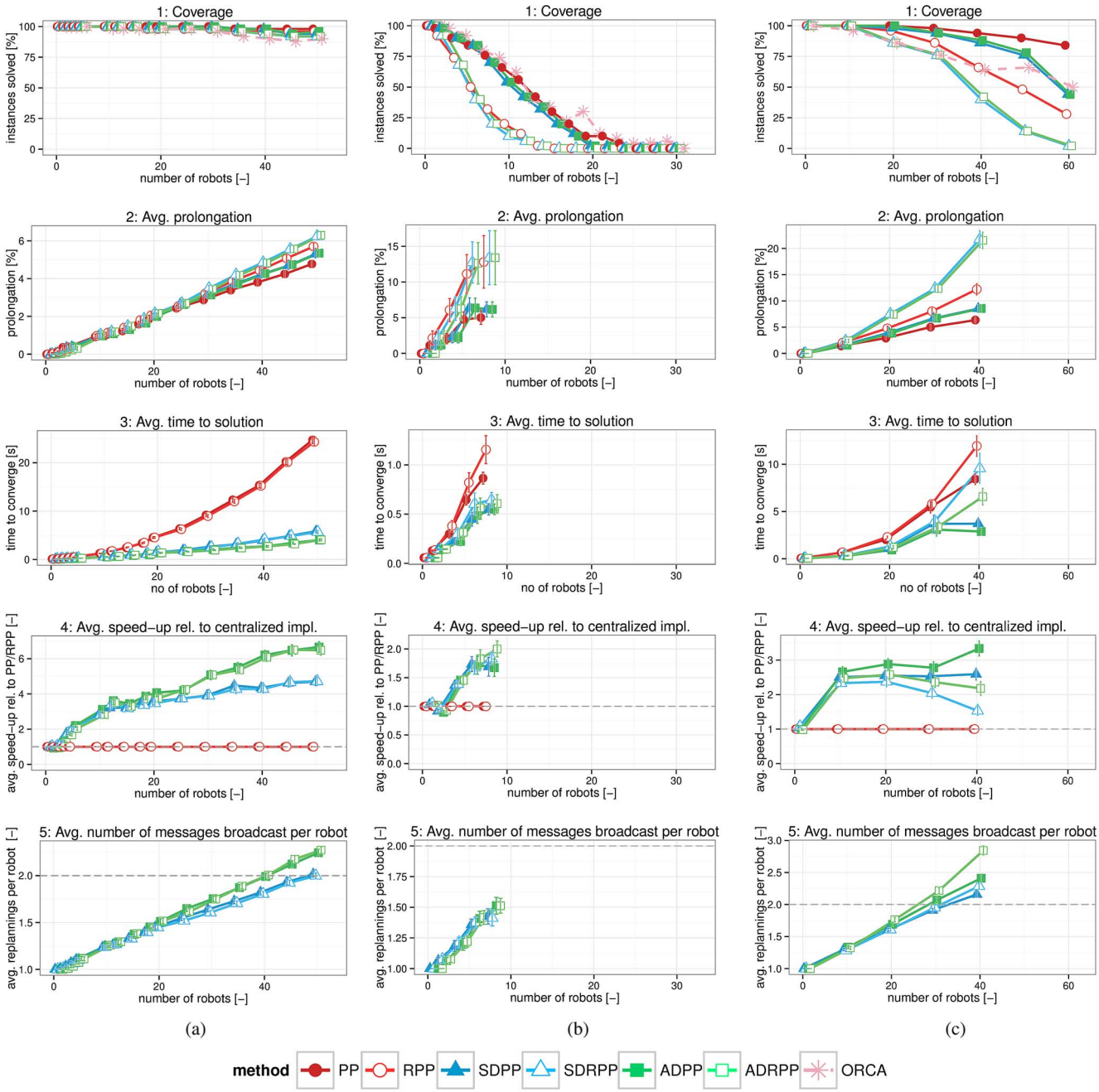

Fig. 12. Results: free-formed tasks (bars indicate standard error). (a) **Empty hall environment**. (b) **Office corridor environment**. (c) **Warehouse environment**.

to fit such systems by making each robot to periodically broadcast its current trajectory. Due to the asynchronous stateless nature of the ADPP protocol, the algorithm is able to recover from the message loss and resolve conflicts with another robot immediately after at least one of the periodically broadcast messages from that robot is successfully delivered.

Further, real-world communication has latency. If the latency cannot be neglected relative to the time required to replan a trajectory, one should expect a lower speed-up ratio than the one reported by our simulation.

The second issue is that real robots suffer from imprecise sensing and actuation, which leads to errors in localization and trajectory tracking. This problem must be addressed by all open-loop planning algorithms and can to a certain extent be resolved by accounting for such uncertainties during the planning. A simple approach is to consider a sufficiently large buffer around the body of the robot that will empirically embed the uncertainty in localization and tracking.

VI. Conclusion

Prioritized planning is a practical approach for multirobot trajectory planning. In this paper, we have summarized properties and compared the performance of six different algorithms employing the idea of prioritized planning. While PP and SD-PP are existing algorithms that have previously appeared in the literature, the remaining four algorithms are our novel contributions.

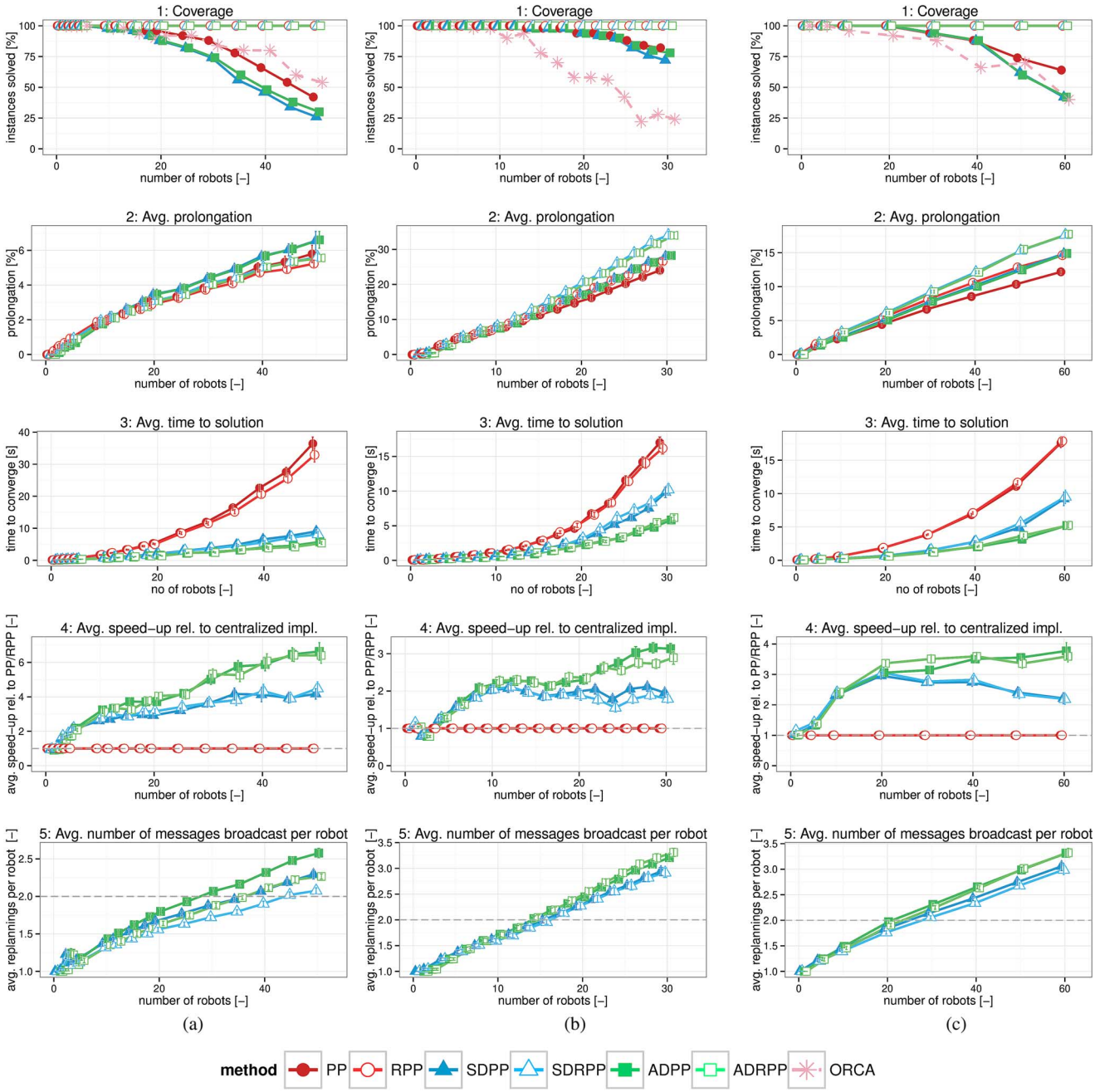

Fig. 13. Results: Tasks in well-formed infrastructures (bars indicate standard error). (a) **Empty hall environment**. (b) **Office corridor environment**. (c) **Warehouse environment**.

The first contribution of this paper is the revised version of prioritized planning (RPP) and a formal proof that this algorithm is guaranteed to provide a solution if there exists a path for every robot that reaches its goal position, avoids start positions of lower priority robots, and avoids goal positions of higher priority robots. We have shown that this condition is satisfied if the individual robots move between two endpoints of a well-formed infrastructure. The significance of this result lies in the fact that human-made environments are usually build as well-formed infrastructures and thus the RPP algorithm can be used to efficiently find coordinated trajectories for the robots operating in such environments. We have experimentally demonstrated that in well-formed infrastructures, the RPP algorithm solves instances that would otherwise be unsolvable by state-of-the-art techniques such as the classical prioritized planning or ORCA.

The second contribution of this paper is a novel asynchronous decentralized implementation of both the classical and the revised prioritized planning scheme and a formal proof that the algorithm is guaranteed to terminate. Furthermore, we proved that the asynchronous decentralized implementation of the revised prioritized planning is guaranteed to provide a solution under the same conditions as its centralized counterpart and thus it can be used to reliably plan coordinated trajectories in well-formed infrastructures. Finally, using extensive experimental evaluation in simulation on three real-world maps, we have shown that

the asynchronous approach converges up to 2x-faster than the previously known synchronized approach.

In the future, we plan to study extensions of the presented decentralized algorithms to large-scale multirobot systems with local communication.

ACKNOWLEDGMENT

The authors greatly appreciate access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the program "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005).

REFERENCES

[1] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robot. Auton. Syst.*, vol. 41, no. 2, pp. 89–99, 2002.

[2] M. Cap, P. Novak, J. Vokrinek, and M. Pechoucek, "Asynchronous decentralized algorithm for space-time cooperative pathfinding," in *Spatio-Temporal Dynamics Workshop (STeDy)SFB/TR 8 Spatial Cognition Center Rep. 030-08/2012*, 2012.

[3] M. Erdmann and T. Lozano-Pèrez, "On multiple moving objects," *Algorithmica*, vol. 2, pp. 1419–1424, 1987.

[4] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: Highly parallel collision avoidance for multi-agent simulation," in *Proc. ACM SIGGRAPH/Eurograph. Symp. Comput. Animation, SCA'09*, New York, NY, USA, 2009, pp. 177–187, ACM.

[5] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, "On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem"," *Int. J. Robot. Res.*, vol. 3, no. 4, pp. 76–88, Dec. 1984.

[6] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/, accessed: May 20, 2015

[7] E. Lalish and K. A. Morgansen, "Distributed reactive collision avoidance," *Auton. Robot.*, vol. 32, no. 3, pp. 207–226, 2012.

[8] V. Narayanan, M. Phillips, and M. Likhachev, "Anytime safe interval path planning for dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS'12)*, 2012, pp. 4708–4715.

[9] D. Silver, "Cooperative pathfinding," in *Proc. 1st Artif. Intell. Interactive Digital Entertainment Conf.*, 2005, pp. 117–122.

[10] P. G. Spirakis and C.-K. Yap, "Strong np-hardness of moving many discs," *Inf. Process. Lett.*, vol. 19, no. 1, pp. 55–59, 1984.

[11] T. S. Standley, "Finding optimal solutions to cooperative pathfinding problems," in *Proc. 24th AAAI Conf. Artif. Intell. (AAAI)*, M. Fox and D. Poole, Eds., Jul. 2010, pp. 173–178.

[12] T. S. Standley and R. E. Korf, "Complete algorithms for cooperative pathfinding problems," in *Proc. 22nd Int. Joint Conf. Artifi. Intell. (IJCAI/AAAI)*, T. Walsh, Ed., 2011, pp. 668–673.

[13] J. Van Den Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robot. Res.*, pp. 3–19, 2011.

[14] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA'08)*, 2008, pp. 1928–1935, IEEE.

[15] J. van den Berg and M. Overmars, "Prioritized motion planning for multiple robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot.*, 2005, pp. 430–435.

[16] J. van den Berg and M. Overmars, "Kinodynamic motion planning on roadmaps in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot.*, 2007, pp. 4253–4258.

[17] M. Čáp, P. Novák, M. Selecký, J. Faigl, and J. Vokřínek, "Asynchronous decentralized prioritized planning for coordination in multi-robot system," in *Proc. Intell. Robot. Syst. (IROS'13)*, 2013.

[18] P. Velagapudi, K. P. Sycara, and P. Scerri, "Decentralized prioritized planning in large multirobot teams," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot.*, 2010, pp. 4603–4609.

[19] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artif. Intell.*, vol. 219, pp. 1–24, 2015.

Michal Čáp, photograph and biography not available at the time of publication.

Peter Novák, photograph and biography not available at the time of publication.

Alexander Kleiner, photograph and biography not available at the time of publication.

Martin Selecký, photograph and biography not available at the time of publication.