Probabilistic Road Map sampling strategies for multi-robot motion planning

Christopher M. Clark

Department of Mechanical Engineering, University of Waterloo, Waterloo, Ont., Canada N2L 3G1

Abstract

This paper presents a Probabilistic Road Map (PRM) motion planning algorithm to be queried within *Dynamic Robot Networks*—a multi-robot coordination platform for robots operating with limited sensing and inter-robot communication.

First, the Dynamic Robot Networks (DRN) coordination platform is introduced that facilitates centralized robot coordination across ad hoc networks, allowing safe navigation in dynamic, unknown environments. As robots move about their environment, they dynamically form communication networks. Within these networks, robots can share local sensing information and coordinate the actions of all robots in the network.

Second, a fast single-query Probabilistic Road Map (PRM) to be called within the DRN platform is presented that has been augmented with new sampling strategies. Traditional PRM strategies have shown success in searching large configuration spaces. Considered here is their application to on-line, centralized, multiple mobile robot planning problems. New sampling strategies that exploit the kinematics of non-holonomic mobile robots have been developed and implemented. First, an appropriate method of selecting milestones in a PRM is identified to enable fast coverage of the configuration space. Second, a new method of generating PRM milestones is described that decreases the planning time over traditional methods. Finally, a new endgame region for multi-robot PRMs is presented that increases the likelihood of finding solutions given difficult goal configurations.

Combining the DRN platform with these new sampling strategies, on-line centralized multi-robot planning is enabled. This allows robots to navigate safely in environments that are both dynamic and unknown. Simulations and real robot experiments are presented that demonstrate: (1) speed improvements accomplished by the sampling strategies, (2) centralized robot coordination across Dynamic Robot Networks, (3) on-the-fly motion planning to avoid moving and previously unknown obstacles and (4) autonomous robot navigation towards individual goal locations.

Keywords: Multi-robot systems; Robot coordination; Motion planning; Probabilistic Road Maps; Robot networks; Ad hoc communication networks

1. Introduction

Motion planning is the construction of collision-free trajectories that connect robots to their individual

goal destinations. Motion planning performance can be characterized by several algorithm properties: speed, completeness and optimality. For robots operating in dynamic, unknown environments, planning must occur on-the-fly and the primary requirement is algorithm *speed*.

For multi-robot motion planning, coupled planning is beneficial because the motion of each robot can be planned while considering the motion of all robots. However, coupled planning can be slow, making on-the-fly planning difficult to achieve. Decoupled planning is fast, but is not complete. This paper first presents a robot coordination platform called Dynamic Robot Networks to enable centralized motion coordination despite limitations in sensing and communication. While the motion coordination is centralized, the platform allows for both decoupled and coupled PRM motion planning to occur in parallel, distributed across the robot network. Thus, taking advantages of both approaches.

In the past, *Probabilistic Road Map* (PRM) planners have shown the ability to plan quickly for systems with many of degrees of freedom. Here, PRMs are applied to coupled multiple mobile robot motion planning problems. Several PRM sampling strategies are evaluated with a particular single-query PRM planner (originally introduced by Hsu et al. [16]). The planner in ref. [16] can construct feasible, collision-free trajectories for robots operating in dynamic environment, but does not address planning for more than two robots. This research adds new techniques that improve upon existing sampling strategies when applied to coupled multi-robot planning. Listed below are the key steps in the single-query PRM algorithm, in which these techniques are implemented:

- Selecting milestones from the road map for expansion—Special techniques of selecting milestones for expansion to ensure fast configuration space coverage and sampling uniformity.
- Generating new milestones for the road map—The average number of collision-checks necessary to successfully generate a new milestone is exponential with the number of robots. This paper presents a new milestone generation technique that decreases this exponential complexity.
- Checking for endgame region inclusion—An endgame region of greater size will improve the chance of finding a solution. Also, determining if a road map milestone belongs to the endgame region must be easily calculated to reduce computation time. A new endgame region is presented that is substantially larger than traditional definitions, allowing for increased chance of finding a solution when goal configurations are highly constrained.

Implementing these techniques leads to decreased planning time and allows for on-the-fly robot planning. What follows is an overview of related motion planning research, an overview of the DRN platform, a description of the PRM algorithm, a description of the new sampling strategies and results.

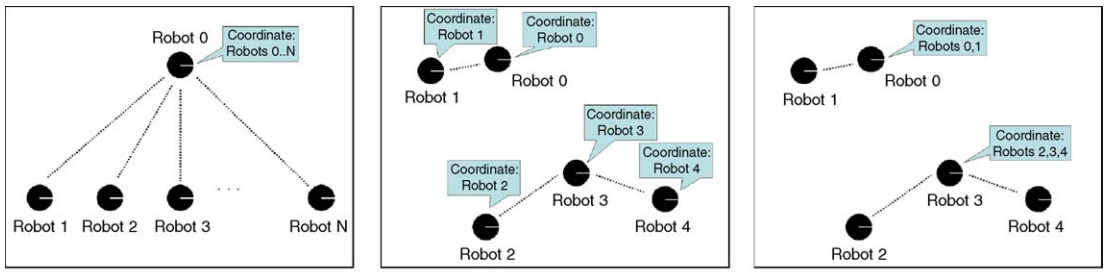
2. Related work

The many approaches to multi-robot motion planning are usually compared based on their algorithm's speed, completeness and optimality. For complex problems, it is difficult to meet all of these requirements. Probabilistic Road Map planners have recently gained popularity because of their speed. However, effective sampling strategies are crucial to achieving successful PRM planning. Presented below is an overview of multi-robot group architectures, multi-robot motion planning, PRMs and PRM sampling strategies.

2.1. Group architecture

The method of coordinating robots will depend heavily on the *group architecture* of the multi-robot system. Most architectures are classified as being centralized or decentralized.

Within a *centralized architecture*, a single agent has information about the entire system and controls all agents in the system. Because this agent has complete information, *centralized coordination* algorithms can be used. Fig. 1(a) provides an illustration in which one agent, robot 0, plans actions for all robots. One example is the NANOWALKERS multi-robot system developed for nano-scale manipulation and inspection [27]. Unfortunately, centralized architectures are usually not scalable because a single agent is responsible for communicating with and processing the control over every other robot. They suffer from single-point failures in that the whole system will fail if the central agent fails. They are also not practical for many applications



(a) (b) (c)

Fig. 1. Centralization vs. decentralization.

where no single agent has complete knowledge of the environment and the other agents, as is the case when limitations in communication are present.

Within *decentralized architectures*, control responsibility is distributed and each agent uses local sensing and communication for control [10,14,28,29,39]. Fig. 1(b) provides an illustration in which each agent plans its own actions based on information about neighboring robots (i.e. they use a type of *decentralized coordination*). These approaches have been shown to be scalable and fault-tolerant. One example is Behavior-Based Systems [28], in which robots are equipped with a set of primitive behaviors (e.g. corridor-finding). If individual robots employ the appropriate behavior(s), desirable group behaviors can result. Related to this approach are Robot Ant Colony systems [39]. Robots within these systems have been shown to cooperate and accomplish complex tasks, despite the fact that individual robots are simple (i.e. they have limited sensing, communication and computation capabilities). The main issue is that robots do not generally have complete system information or communication with all robots in the system.

While the group architecture defines the inter-robot relationships of a system, it is largely a function of the communication structure. Furthermore, within the group architecture, robot coordination can be facilitated by implementing a *coordination platform*—a communication infrastructure that determines how robots coordinate their actions through Data Exchange.

In this research, a coordination platform is proposed that uses a communication infrastructure based on Mobile Ad-Hoc Networks (MANETs) [34]. Equipped with MANET communication capabilities, robots can act as routers in a network to pass information between robots which might not otherwise be able to communicate, e.g. robots 2 and 4 in Fig. 1(b). This provides individual robots with more information about the environment and the other robots. This information could be used to improve the performance of any of the core capabilities required by autonomous robots including planning, sensing and control.

Coordination across an ad hoc network can benefit robots operating in dynamic, unknown environments where sensing and communication are limited. Consider the example in Fig. 1(c). Communication limitations prohibit any communication link between the two groups of robots. While centralized coordination cannot occur between all five robots, it can occur within each of the two distinct groups of robots. Also, because a decentralized architecture is used, the system is scalable and fault-tolerant to single-point failures.

To implement this type of coordination, several issues must be resolved to ensure the coordination is (1) fault-tolerant to network communication drops caused by network breaks, (2) tolerant to communication delays caused by information having to hop through the network and (3) equipped with a planning algorithm that is fast enough to be run on-line. This research aims to provide a coordination platform, i.e. DRN that addresses these issues.

2.2. Multi-robot motion planning

Multi-robot motion planners are usually classified according to whether the planning is *decoupled* or *coupled* [2,36]. Decoupled planners construct plans for each robot separately before coordinating the individual plans [2,3,21,22,26,30,38]. The coordination step can be accomplished by tuning the robot velocities

along their respective paths (e.g. [21]). Consider two robots whose paths cross. If both these robots follow their paths with some nominal velocity, there is possibility of collision. However, by tuning velocities so one robot slows down and the other robot speeds up to pass through the intersection first, a collision-free pair of trajectories can result. This coordination can be done globally, in which complete information is available to the planner or locally (i.e. when robots come close to one another).

A variant of decoupled planning, called prioritizing planning, plans for one robot at a time, in some sequence, considering the robots whose trajectories have already been planned as moving obstacles [7,10,13]. In ref. [7], trajectories were constructed for each robot in a specific order such that each trajectory is collision-free of previously constructed trajectories. A search routine was used to find the order that provides shorter paths and in some cases was essential to finding a solution.

Decoupled planning algorithms can be advantageous because they do not require robots to have complete system information and are generally fast enough for planning on-the-fly. However, they are inherently not complete and often cannot find solutions when robots must be tightly coordinated [36].

Reactive style planning is one type of decoupled planning that has proven suitable for many applications because it is fast, enabling real-time planning. A common reactive approach is potential fields [23]. This approach has been applied to both single robots and extended to multi-robot applications [41] including robot soccer [19]. A major drawback of potential fields, is their susceptibility to deadlock.

Another drawback of decoupled planners is that they usually fail to find globally optimal solutions because they do not use global knowledge. Hence, many algorithms exist that search for near-optimal solutions. One example [14], uses the method of altering velocities with D^* to produce a distributed planner that tries to optimize trajectories. Also in ref. [3], negotiations between localized groups of robots are used to assign priority orders to robots, that when applied to the planning algorithm, results in reduced trajectory lengths. The negotiation scheme in ref. [3] demonstrates the benefits of localized inter-robot communication, and is the research most closely related to the robot network system presented later in this paper.

Coupled planning considers all robots together as if they were forming a single multi-body robot [4,9,24,31,35,40,41]. Coupled planning is beneficial because the motion of each robot can be planned while considering the motion of all robots. Unfortunately, coupled planning is often slow and requires that at least one robot be provided with complete system information. This becomes a problem when robots are operating in dynamic unknown environments where there is a requirement for fast, online planning.

Recently, there has been research into using mixed integer linear programming to solve multi-robot path planning (e.g. [6,33]). These methods result in optimal trajectories, but still require longer planning times not practical for some on-line implementations.

In ref. [37], a non-linear model predictive control (NMPC) is used for the control of autonomous helicopters. Simulation results exhibited trajectory generation for helicopters operating in complex 3D environments, multiple vehicle collision avoidance and predator evasion. Computation times ranged from 41 to 173 s.

To handle the requirement for speed, a single-query Probabilistic Road Map planner is proposed. In this case, the planner is queried within DRNs as a centralized planner that concurrently employs both coupled and decoupled approaches to plan trajectories for all robots in the network.

2.3. Multi-robot planning with PRMs

Probabilistic Road Maps have been used to solve path planning problems with many degrees of freedom successfully [18,35,36]. They have also been shown to construct plans that satisfy various constraints (e.g. dynamic, non-holonomic, etc.) [20]. They are not complete in the traditional sense. However, under certain assumptions (e.g. the free space is *expansive* [16]), they are *probabilistically complete.* That is, the probability of failure decreases quickly (e.g. exponentially) to zero with time.

PRMs have been applied to multi-robot motion planning problems, many of which use decoupled planners. One example is [10], where a single-query PRM algorithm is used with prioritized planning. Each robot calculates a priority number based on the occupancy of its neighborhood (i.e. the more robots/obstacles in its neighborhood, the higher the planning priority). As:

Algorithm 1. Single-Query PRM Planner

1. Add initial milestone m_0 to road map M
2. **Until** timeout
3. Randomly select a milestone m from M
4. m_{new} = PROPAGATE(m)
5. Add m_{new} to the road map M
6. **If** m_{new} is connected to goal state
7. **Return** plan connecting m_0 to the goal state
8. **Return** null

robots move into one another's neighborhood, the robot with lower priority plans to avoid the higher priority robot. The higher priority robot continues on its original path. Results demonstrate on-the-fly planning for up to 15 robots in a cluttered environment.

One example of a coupled approach is presented in ref. [40], where a multi-query PRM is used. First, a road map is constructed for one robot. Then, several of these road maps are combined into a road map for the composite robot. The approach worked well in planning for up to five car-like robots in static environments, and has the advantage of being probabilistically complete.

In ref. [36], coupled and decoupled planning are compared using PRMs. Both approaches were applied to test scenarios involving two to six robot manipulators (12–36 degrees of freedom). Given those scenarios, decoupled planning often failed to find any solution. This research demonstrated the advantage of coupled planning when the motion of multiple robots requires tight-coordination. Aside from ref. [36], few have investigated how different sampling strategies can affect planning for multiple robots.

2.4. Background on PRMs

PRMs are usually classified according to whether they are *single-query* or *multi-query.* To construct a multi-query PRM, a time-intensive pre-processing step is required to construct the road map. Once completed, this road map can be queried many times to search for trajectories from any pair of start/goal configurations. However, for many applications the road map construction step is too slow for on-line implementation (e.g. to avoid moving obstacles).

For a single-query PRM planner, a new road map is constructed for each query. In these planners, less time is spent constructing the road map because only a restricted subset of the configuration space is sampled. This is usually accomplished by a *single-directional* search or a *bi-directional search.* For a single-directional search, a tree of milestones in grown from the initial configuration until a connection is found with the goal configuration. Two trees are grown for a bi-directional search, one from the initial configuration and one from the goal configuration, until a connection between them is found.

In Hsu et al. [15], a single-query PRM planner was developed to successfully plan trajectories for a robot operating in dynamic environments. Results demonstrated on-the-fly planning for real robots that are operating among moving obstacles. Hsu's et al. algorithm is represented as Algorithm 1. In this representation, the motion of the robot is governed by Eq. (1). The state of the robot is x such that $x \in X$, an n-dimensional manifold called the state space. Control inputs to the robot are represented as u.

$$\dot{x} = f(x, u) \tag{1}$$

A milestone of the road map is defined by $m = (t, x)$, where x represents the state of the robot r at time t. The initial milestone m_0 defines the initial state of the robot at time zero.

To start, the road map M is rooted at m_0 by adding it as the first milestone in M (step 1 in Algorithm 1). The algorithm iteratively tries to expand M by first selecting an existing milestone m from M and then propagating it to a new milestone m_{new} (step 4). Within the PROPAGATE function, a candidate path from m is generated by integrating Eq. (1) with randomly selected values for u. The function iterates until a collision-free path is found, whereby it returns a milestone m_{new} defined by the path endpoint. In step 5, m_{new} is added to the road map M. If there exists a simple path from m_{new} to the goal state, then planner returns a path connecting m_0 to the goal state (step 6).

This algorithm can be extended to planning for multi-robot planning using a coupled or decoupled planning approach (e.g. [10]).

Within DRNs, a centralized planning approach is taken in which all robots are planned for at once (e.g. [11]). Each robot has information about all other robots in the network, and can then plan the trajectories of all robots using a coupled or decoupled approach. The decoupled approach is a direct extension of ref. [15], in which the planner constructs trajectories for one robot at a time. In the coupled approach, the milestones must define the configuration of all robots being planned for

$m = (t, x_1, x_2, \ldots, x_r)$, where x_r represents the state of robot r at time t. This approach will be slower than a decoupled approach (due to the increased size of the configuration space) but maintains the property of *probabilistic completeness.* Section 4 of this paper concerns the development of new sampling strategies that decrease the algorithm's running time when a coupled approach is taken. Section 3 describes the DRN coordination platform.

3. Dynamic Robot Networks

Dynamic Robot Networks is a coordination platform that functions within a *decentralized group architecture*, but maximizes the *centralization of coordination* between robots.

Dynamic Robot Networks are mobile ad hoc communication networks in which the robots become nodes in the network and can act as routers to relay information through the network. Such networks are formed by robots establishing communication links whenever possible. This can result in many different networks of robots located in different parts of the workspace. The networks are dynamic in that they can break or merge with other networks over time.

Information is distributed within networks to the point where all robots in a network share a common model of the world (although each network in the workspace will have a different model). Over time, this model will change as new information about the environment is gained from on-board sensing. In response to these changes in the model, robots may adapt their navigation plans. In such cases, the network of robots will respond as a whole, by re-planning coordinated motion for all robots in that network.

3.1. Platform description

Within the Dynamic Robot Networks coordination platform, every robot will belong to one network (which could include only that one robot). As robots move about the environment, they will enter and leave each others communication range. This causes *network merges* and *network breaks*, respectively.

Within each network (not between networks), information can be passed between any two robots by way of ad hoc network routing algorithms. Assuming world models can be encoded concisely (a possible issue for some applications), robots can use information exchange to share a common world model. This allows for a centralized *coordination process* to occur across the network in which the actions are planned for all robots within that particular network. A coordination process is a defined series of steps that robots must take to coordinate their actions. Steps include Event Detection, Data Exchange, Model Fusion, Planning and Plan Execution (see Fig. 3).

A coordination process can be initiated by any robot in a network, at any time. A robot will initiate such a process in response to changes in the environment (e.g. two robot networks merge). Once the process is initiated, all robots in the network participate in each step of the process. The platform allows for several processes to occur concurrently.

3.2. Network merges/breaks

When any two robots are within communication range of each other, they establish a communication link. Define G to be the graph whose nodes are the robots and edges are the communication links. A network of robots is any group of $k \geq 1$ robots forming a maximally connected component of G. So, any two robots in a network can communicate through one or several communication links, but two robots from different networks cannot. Fig. 2(a) shows an environment with five robots, where two networks have formed. In the network on the right, the top and bottom robots can exchange information via their communication links with the middle robot.

Because robots and objects are moving, the networks are dynamic. The networks may merge (see Fig. 2c) and/or break apart. Ad hoc network protocols [8] ensure that edges in G are established when possible, and that information can be routed efficiently across these edges. With G established, robots within the network can communicate and conduct a coordination process.

To facilitate information exchange between robots in a network, it is assumed that each robot is assigned a unique identification number. Also, when two networks merge, let the robot with the lower identification number of the two robots that caused the merge be known as the *Lead robot* and the other robot that caused the merge be known as the *Secondary robot*.

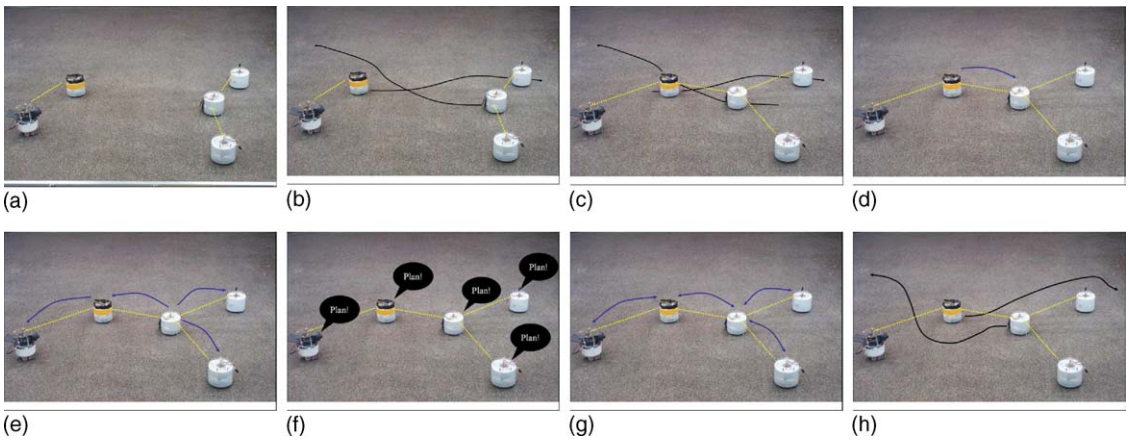


Fig. 2. Robot coordination example.

3.3. Coordination process

The coordination process that takes place across a robot network is a series of steps as shown in Fig. 3. The process is initialized with an *Event Detection* step. Such events may include changes to the: (1) Network Topology, e.g. a new robot is in communication range and joins the network, (2) world model, e.g. the sensing of new obstacles in the environment and (3) goal state, e.g. a new goal state is requested by one of the robots in the network.

Information regarding the detected event will be routed across the network with the *Data Exchange* step. This information will include world state information (i.e. object state estimates, estimate confidence levels, object sizes and object trajectories), with which each robot's world model must be updated. Using the network topology information gained from implementing a table driven routing algorithm [34], the amount of information broadcasted can be minimized. An example of the Data Exchange that occurs when two networks merge is depicted in Fig. 4. When robots receive world model information obtained from other robots, they must fuse it with their own world model (i.e. the *Model Fusion* step).

Along with the world state information will also be sent a "plan request" message (if required). This informs robots to start constructing a new plan that takes the new event into account. This starts the *Planning* step in which robots construct a plan that schedules actions of *all robots* in the network. Here, a Probabilistic Road Map motion planning algorithm augmented with new sampling strategies [11] has been implemented. To carry out the planning step, each robot in the network calls a PRM planner to construct trajectories. Some robots can call a coupled PRM planner to maximize completeness, while others can call a decoupled PRM planner to maximize planning speed. Because

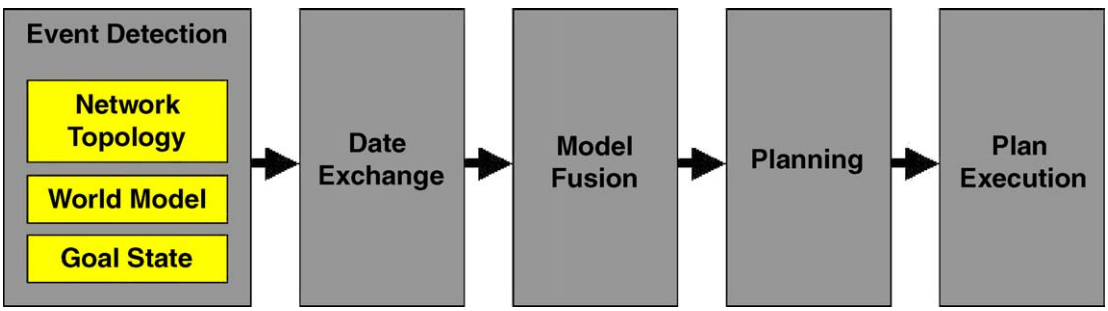


Fig. 3. Coordination process.

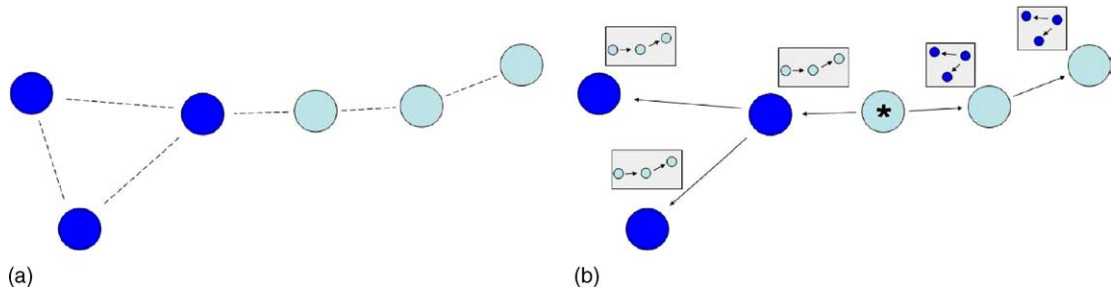


Fig. 4. Data Exchange step: after two networks merge (a), the information within the each of the previous networks is distributed so all robots in the newly formed network have a common world model (b).

the algorithm uses a random search, each robot will produce a different plan (i.e. a different set of trajectories). This step is followed by robots broadcasting their newly constructed plans to all other robots.

Each robot in the network will receive the plan constructed by all other robots in the network. Robots will then implement the best plan of those received to carry out the *Plan Execution* step. Further details of each step can be found in ref. [11].

An example of the coordination process involving five robots is illustrated in Fig. 2. Initially, two robot networks are present. Two robots, one within each network, are following trajectories to their respective goal locations (b). Note that these trajectories collide, but this is undetected because robots are not close enough to communicate. As the robots follow their trajectories (c), they eventually can communicate (Event Detection). They begin the Data Exchange step of the process when the follower robot broadcasts its world model (d). The lead robot then broadcasts a "plan request" message to all robots in the network (e). Upon receiving this message, robots merge the newly acquired information (Model Fusion step) and query their planners (i.e. the Planning step) to construct a set of trajectories for all robots in the newly formed network (f). As each robot completes its plan, it broadcasts it for other robots to receive (g). Once a robot receives a plan from every robot in the network, it picks the best plan based on some established criteria and uses it for motion (h) to complete the Plan Execution step.

3.4. Multiple coordination processes

One of the main challenges of implementing centralized coordination across an ad hoc network is that the robots are continuously moving and hence the network topology is dynamic. Difficulties arise when robots enter and leave one another's communication range within a short period of time (e.g. less than a second). In these cases, continuous network communication might not be possible throughout the entire coordination process, which can last on the order of 500 ms. The planning system must be robust to such difficulties. What follows is a description how such events are handled, so as to continue providing responsive, distributed planning across the network.

3.4.1. Network breaks

In the case where a network breaks into two different networks of reduced size, the coordination process must continue. Because messages are queued and processing of them is synchronized, it can be assumed that the plan manager will not realize such a break until after a robot begins its actual planning (i.e. it has queried the planning algorithm).

At this point, the robot's planner will continue constructing trajectories, even for those robots that no longer belong to the same network as the robot. However, once the robot finishes planning, it waits to receive plans from only those robots that are currently in its new reduced network. For example, if five robots in a network are planning and one robot leaves, then the four remaining robots will distribute their plans and implement the best of the four. The fact that the plans consist of trajectories for five robots will not hinder the coordination process. Note that this does require robots to update the network with the information that another robot has left communication range and robots should not wait to receive a plan from it. This can be accomplished through means of a network level

routing algorithm protocol (i.e. the Data Exchange step).

If the network breaks after plans are completed (i.e. during the plan execution phase of a coordination process), there will be no ill effects. Each robot executes only its own plan and does not consider the other robot plans at this point.

3.4.2. Multiple triggers

It is possible for a new plan trigger (i.e. new desired goal state, new network merge or new object state estimates), to occur during a coordination process. In these cases, it is desirable to plan with this new information as soon as possible. However, robots cannot simply halt their current coordination process to start a new process based on the most recent information. This can lead to endless planning with no plan execution (i.e. the system may repeatedly halt plan searches as a robot continually receives new plan triggers).

The solution presented is as follows. As new triggers occur during a coordination process (or any time after a coordination process has been initiated), they are stored until the first completed plan from the original coordination process is received. At this point, the robots execute the first plan and initiate the next coordination process, which takes into account all stored trigger information. This ensures that plans are given time to finish, but starts the next process promptly.

This system allows for several new triggers to be stored until the next coordination process begins. Also, it allows for different triggers to be heard by different robots at different times. Consider an example, where two robots, located at opposite ends of a network, each detect a different plan trigger. Each robot will initiate a separate coordination process and send out its own "plan request" message with information regarding the trigger event it detected. Each robot will also begin the planning stage for the coordination process it initiated. As each robot receives the other robot's plan request, it will store it until it gets the first solution to its own plan request. Once receiving this first plan, it will begin executing the plan and immediately start planning again to incorporate the trigger received from the other robot. In this manner, each robot will execute a plan that responds to the trigger it detects, then construct and execute a plan that responds to both triggers. See Figure 5 for an example time-line.

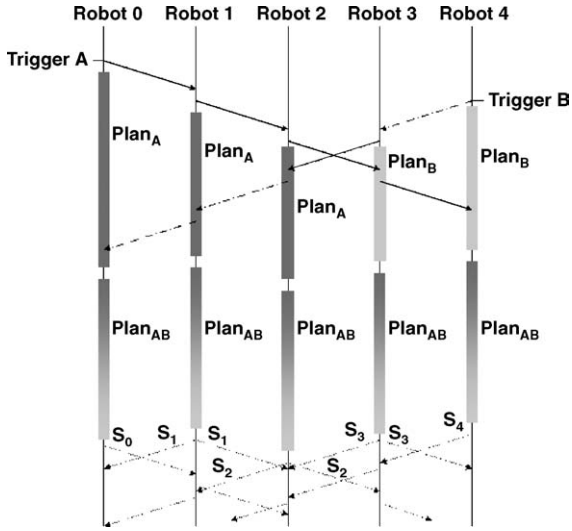


Fig. 5. Multiple trigger time-line.

For this protocol, the maximum time before a plan is executed for any given trigger is always less than double the time to carry out one coordination process. This may occur if a new trigger is detected immediately after the start of a coordination process initiated by an earlier trigger. This ensures a finite planning time for any new trigger. Note that due to communication delays, numerous completed plans for a coordination process may have been sent after the first plan, only to be received after a new coordination process has begun. In these cases, robots will simply implement them if they are better than the first, without interrupting the new coordination process (Fig. 5).

4. PRM sampling strategies

In PRM planning, a large amount of time is spent collision-checking. One way to reduce the amount of collision checking is use better sampling strategies.

These strategies avoid milestone generation in uninteresting areas of the free space. Connecting new milestones to the road map in such areas requires costly collision-checks, without greatly expanding the road map.

Examples of different sampling strategies that have been applied to multiple-query PRM planners include multi-stage strategies [19], obstacle-sensitive strategies [1] and narrow-passage strategies [17]. Several

sampling strategies have also been applied to single-query PRM planners. Both single-directional and bi-directional searches require *diffusion* strategies to avoid over-sampling certain areas of the free space. More specifically, the road map must eventually diffuse through the reachable component of the free space, and result in a uniform distribution of milestones across the components. This uniform distribution is required to prove the planner's fast convergence property [16].

There are two main approaches to diffusion. One approach is to first select a milestone m from the road map with probability inverse to the density of milestones in the neighborhood of m. Then, a new milestone m_{new} is obtained with a random but uniform sampling of the neighborhood of m.

To speed up the selection of m, milestone density calculations are approximated through a discretization of the configuration space. A common technique is to use a *hyper-grid* of the configuration space [16]. In this technique, the configuration space is divided into a grid of cells. A milestone is selected by (1) randomly selecting a cell c from all those cells, which are occupied and (2) randomly selecting a milestone from within c.

This method has been extended for multi-robot planning by using *hashtables* to dynamically allocate the memory for the cells that discretize the large configuration space. Hashtables can also provide an efficient means of weighting the gridcells further [12].

A technique similar to *hyper-grid* milestone selection has been applied to planning the motion of multiple robot manipulators with many degrees of freedom N_{dof} [35]. First, h degrees of freedom are randomly selected, where $h \ll N_{\text{dof}}$. Then, local milestone densities are calculated based only on the closeness of milestones within the h degrees of freedom. Using these densities for weighting milestone selection, a milestone m is picked to generate m_{new}. This technique, *multi-grid* selection, is also applied to multiple mobile robot planning in ref. [10].

The other main diffusion approach derives techniques from the closely related Rapidly exploring Random Trees (RRTs) [25] (a variant of PRM planning). In these techniques, a configuration q is randomly selected from the configuration space. Then, the milestone m, which is closest to q is obtained. Finally, a new milestone m_{new} is selected along the line connecting m to q. A drawback of these techniques is that a search for the milestone with the shortest distance must be done for each expansion. One way to minimize the effects of this drawback is to only consider a small sample of randomly selected milestones in the road map for each expansion. Also, instead of picking a point, one can randomly select a gridcell c_{random} from a discretized grid of the configuration space, then find the occupied gridcell c that is closest to c_{random} using the Manhatten distance metric. From c, a milestone is selected randomly.

Based on Algorithm 1, this research invokes sampling strategies broken down into the three components used to (1) select new milestones for expansion, (2) generate new milestones by expanding from an existing milestone and (3) defining the endgame region that determines if newly generated milestones are connected to the goal configuration. The next three sections describe these components, with improved techniques to speed up planning.

4.1. Road map milestone selection

In identifying an appropriate selection technique for multi-robot planning, the different diffusion techniques mentioned above were compared via simulations. The simulations involved three robots, each with one degree of freedom. The resulting joint configuration space C is a cube. Portions of C are non-free to simulate robot collisions. To establish a comparison metric, the joint configuration space (i.e. the cube) is divided into 3375 smaller *occupancy cubes.* The coverage of the configuration space is then measured by the number of these smaller cubes occupied by at least one milestone. Simulations are conducted by expanding the PRM from a randomly selected point in C, using the different sampling techniques above. To summarize, each technique is compared based on how quickly the road map expands over C. The faster the expansion, the faster a path to any goal in C can be found. Fig. 1 plots the different expansions, or amount of C covered by the road map, as a function of time.

Illustrated in Fig. 6(a) and (b) are the average configuration space coverages from expanding a road map using each of the above mentioned sampling techniques. Aside from the unweighted case, each technique demonstrates an initial region of fast expansion, followed by a region of slower expansion. However, the ratio of these two regions differs greatly between

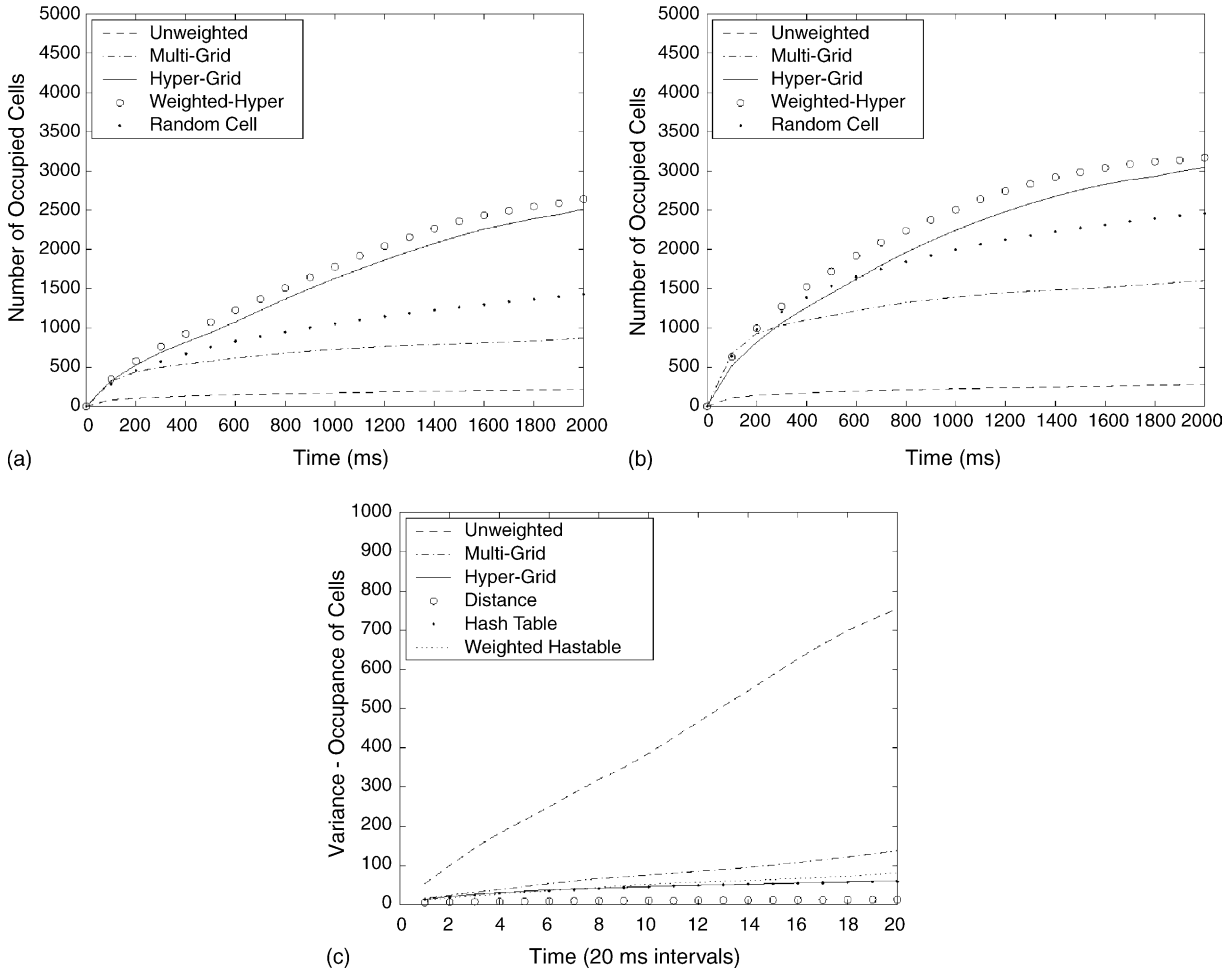


Fig. 6. Milestone selection techniques—coverage: the coverage of 3375 cells hyper cube are shown for various sampling techniques. In (a), the coverage from a single planner is plotted. In (b), the composite coverage of three different planners running in parallel is plotted. In (c), the uniformity of the configuration space coverage is measured as the variance of the occupancy of cells.

sampling techniques. The multi-grid approach tapers off quickly to a very slow expansion. The random cell technique (from RRT) provides a good rate of coverage, especially when considering the composite of three planners running in parallel. The hyper-grid techniques (including the dynamically allocated hyper-grid) demonstrated superior performance. It was not until a majority of the configuration space was covered before their rate of expansion decreased significantly.

A second metric for comparing these sampling techniques is the uniformity of the expansion. To measure uniformity, the variance of *occupancy cubes*—the square of the average difference between the occupancy of the cubes and the average occupancy, was used. In Fig. 6(c), the variance of occupancy cube milestone density is plotted as a function of time. It is clear that the unweighted approach leads to a very non-uniform milestone expansion. The variance increases with time indicating that some *occupancy cubes* are occupied by many more milestones than others. Other techniques demonstrated a slightly increasing variance, indicating a more uniform milestone expansion (i.e. most areas of the configuration space have generally the same density of milestones).

4.2. Milestone generation

In ref. [16], a two-step sampling diffusion technique was introduced where new milestones are generated in vicinities of the road map that have a low density of milestones. Discussed in the previous section was the first step: the random selection of a milestone m from the road map. This section presents a new method of accomplishing the second step: the generation of new milestone in the neighborhood of m. This method, called *serial expansion*, increases the likelihood of successfully generating milestones by decreasing the number of required collision-checks.

Within the PROPAGATE function of Algorithm 1, several candidate paths from m are generated by integrating Eq. (1) with randomly selected values for u. The function iterates until u induces a collision-free path, whereby it returns a milestone m_{new} defined by the path endpoint. It is important to note that *the order in which the different control inputs of u are randomly selected can affect the number of collision-checks necessary to successfully generate a new milestone*.

Previous research has used a *parallel* approach to milestone generation in that all control inputs are selected simultaneously, followed by collision checking [20]. If the trajectories connecting states in the existing milestone to states in the newly generated milestone are collision-free, then the new milestone is added to the road map.

In this research, a *serial* approach is introduced. For each robot, the control inputs are randomly selected and collision-checking is carried out between it and all previously expanded robots. For example, consider generating a new milestone by expanding a milestone defined by $m(t, x_A, x_B, x_C)$ for robots A, B and C. First, the amount of time Δt between milestones is randomly selected. Second, a new state x'_A is generated by applying random inputs to state x_A. Then x'_B is generated and a check is made to ensure that the trajectory from x_B to x'_B is collision-free with the trajectory from x_A to x'_A. Random inputs are continually used to obtain a new x'_B until collision-free trajectories are obtained. Finally, a new state x'_C is generated and a check is made to ensure that the trajectory between x_C and x'_C is collision-free with the trajectories from x_A to x'_A and from x_B to x'_B. Again, candidate states for x'_C are randomly generated until collision-free trajectories are obtained. What results is a collision-free milestone defined by $m'(t', x'_A, x'_B, x'_C)$, where $t' = t + \Delta t$.

A problem with the serial approach is that more search freedom is given to robots whose motion is expanded first. To deal with this problem, two measures are taken. First, the order of robots is randomly selected at each milestone expansion. Second, there is a timeout check. This is used to ensure that the algorithm does not get stuck in a particularly difficult expansion. For example, the first robot state expanded could result in a trajectory for which all other robot state expansions will lead to collision.

The purpose of using serial expansions over parallel expansions is that information from previous failed state expansions is used for future expansion attempts. That is, as each individual robot state is expanded, the previous successful robot state expansions are reused. In contrast, parallel expansion throws out this information at every expansion attempt. Equations that predict the performance of each expansion type can be found in ref. [12].

To compare the two methods of expansion, 50 simulations were run in which a road map was expanded continuously for 0.5 s. At each milestone expansion, both the parallel and serial methods were implemented. Data was recorded for each simulation, including the number of collision checks during each expansion. With this information, the average number of collision checks necessary for a successful expansion were predicted (see ref. [12] for prediction calculations) and compared with the recorded number for each expansion. Results are plotted in Fig. 7.

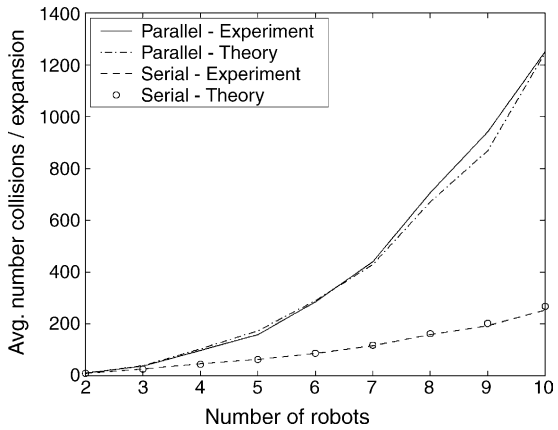


Fig. 7. Parallel vs. serial expansion.

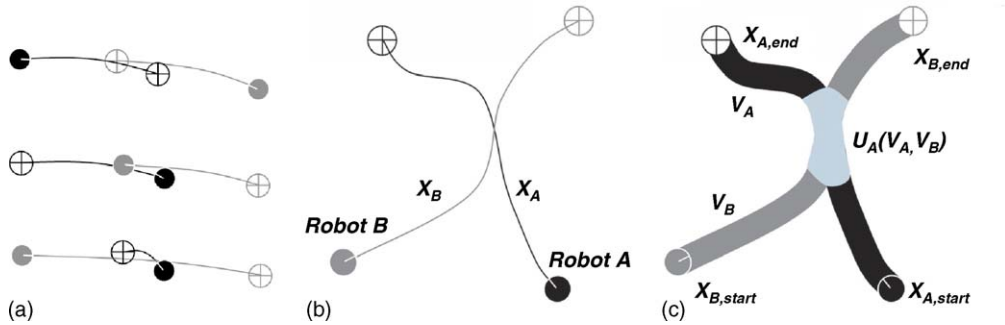


Fig. 8. Velocity-tuning: in (a), three examples of paths that cannot use velocity-tuning to become collision-free are provided. In (b), a sample pair of trajectories are provided for which variables are defined in (c) for *Leadability*.

As the number of robots increases, the number of collision-checks required with parallel expansion grows more quickly than with serial expansion. Note that there is a direct correlation between the number of collision checks necessary for an expansion and the time taken to complete an expansion. Thus, on average, serial expansions take less time than parallel expansions.

4.3. Defining the endgame region

For single-query PRM planning using a single directional search, a tree of milestones is grown until it connects with the goal state. How the tree connects to the goal state is determined by how one defines the *endgame region E*: a region of the free space in which configurations have a simple connection with the goal configuration. This region is not calculated explicitly. Instead, admissibility tests are conducted to determine if a configuration belongs to *E*.

The method in which an endgame region is defined for a specific planning problem can significantly alter the success of the planner. A key to successful planning is to enlarge the endgame region as much as possible [20]. This increases the possibility that a road map will intersect with the endgame region and provide a feasible solution, i.e. the larger the endgame region, the higher the probability a milestone in the road map will belong to the endgame region and hence the higher the probability of finding a solution. A second desired characteristic of the endgame region is that the admissibility test be easily calculated. This test will occur for every new milestone added to the road map and will greatly affect the speed of the planner.

Previous approaches to defining the endgame region fail to meet the above mentioned requirements when applied to multi-robot planning problems. In ref. [5], the endgame region is defined to be a ball of small radius centered at the goal. This works well for configuration spaces of low dimensionality. However, as the dimensionality increases, the likelihood of sampling a milestone within the ball of fixed radius decreases rapidly.

For some robots, it is possible to analytically compute one or several canonical control functions that exactly connect two given points while obeying the kinodynamic constraints (e.g. [32]). If such control functions are available, one can test if a milestone belongs to *E* by checking if the canonical control function generates a collision-free trajectory connecting *m* to the goal state. A similar example method is found in [20], where cubic splines take the place of the control function. The cubic splines were generated based on *k* randomly selected end-times. If any of the *k* splines were collision-free and satisfied all kinodynamic constraints, the milestone was said to belong to the endgame region.

This section presents a new endgame region for multiple mobile robot planning that exploits some geometric properties of a multi-rover system. In doing so, it provides a region that is not only larger than that described in ref. [10], but easily calculated. The endgame region presented is based on the concept of *velocity-tuning*—prescribing a time parameterization to path to produce collision-free trajectories [21]. This is accomplished by discretizing the path into trajectory points defined by both space and time.

The new endgame region presented here aims to include those milestones from which the simple paths

that connect them to goal states can be velocity-tuned to produce a collision-free trajectory set. Specifically, to check if a candidate milestone m belongs to the endgame region, a test is done to see if the simple paths connecting robot states in m to their respective goal states can be velocity-tuned. It is essential that this test rule out non-admissible cases (see Fig. 8(a)), but still be fast so as not to slow down the road map expansion.

The test is based on the property of *Leadability*, defined below, that indicates when paths can be velocity-tuned. Simply stated, robot paths are *Leadable* if one robot can take the lead and pass through the intersection(s) of the paths before the other robot. Provided below are two easy-to-calculate conditions that sufficiently (not necessarily) demonstrate *Leadability* for wheeled mobile robots. These conditions are used to develop the endgame region test.

Given that x_i is a candidate path for robot i, let V_i be the volume of the workspace swept by the path x_i. The intersection of two paths can be described by $U(V_i, V_j)$, the union of V_i and V_j. Also let $t_{i,U-}$ and $t_{i,U+}$ be the times that robot i, respectively, enters and leaves $U(V_i, V_j)$.

Definition. Consider a pair of paths $\{x_A, x_B\}$ for robots A and B (see Fig. 8(b)). The paths intersect at $U(V_A, V_B)$, the union of volumes V_A and V_B swept out by the respective robot paths (see Fig. 8(c)). The path pair $\{x_A, x_B\}$ is said to be (A, B) *Leadable* if there exists a time parameterization for the paths in which robot A can pass through $U(V_A, V_B)$ before robot B enters it, thus forming a collision-free trajectory set.

Given initial states of the robots are far enough away from $U(V_A, V_B)$, and given that enough variability exists in their velocities, then it is fairly easy to show whether or not a path pair is (A, B) *Leadable*. The core requirement is that finite values for times $t_{A,U+}$ and $t_{B,U-}$ exist such that $t_{B,U-} > t_{A,U+}$. That is, the time at which robot B enters $U(V_A, V_B)$ is after the time at which robot A leaves $U(V_A, V_B)$.

Here, it is assumed that robots have allowable velocity $\upsilon \in [0, \upsilon_{\max}]$. Furthermore, it is also assumed that robots have infinite acceleration (e.g. stop on the spot). Under these assumptions, it is straightforward to show that sufficient (not necessary) conditions for a path pair $\{x_A, x_B\}$ to be (A, B) *Leadable* are:

(1) Robot A's path end location $x_{A,\text{end}}$ does not intersect V_B.
(2) Robot B's path start location $x_{B,\text{start}}$ does not intersect V_A.

While this property helps determine whether two paths can be velocity-tuned, it alone will not provide information on whether a set of $R > 2$ paths can be velocity tuned to be collision-free. For this reason, the definition of Leadability is generalized to any number of robots:

Definition. A path set $\{x_A, x_B, x_C, \ldots, x_R\}$ for R robots is said to be $(A, B, C, \ldots, R)$ *Leadable* if there exists a time parameterization for the paths in which each robot g from the list $A, B, C, \ldots, R$ can pass through the path union $U(V_g, V_h)$ before any subsequent robot h from the list $A, B, C, \ldots, R$ enters the union, thus forming a collision-free trajectory set.

To check whether a milestone belongs to the new velocity-tuneable endgame region, a test is made as to whether the simple paths connecting robot states in the milestone to the goal states make up a path set that is Leadable. While no formal proof is presented, it should be clear that a Leadable path set requires each path pair in the set to be Leadable (e.g. (Q, R) *Leadable*, (Q, S) *Leadable* and (R, S) *Leadable* imply the path set $\{Q, R, S\}$ is (Q, R, S) *Leadable*).

To accomplish the endgame region test on a milestone, several steps are carried out on the set of paths that connect the robot states to their goal states. First, each path within the set must be tested for collisions with obstacles in the environment. If a collision exists, the milestone is rejected.

Second, each pair of paths $\{x_i, x_j\}$ within the set is checked whether or not it is (i, j) *Leadable* or (j, i) *Leadable*. If it is neither, the milestone is rejected. Moving obstacles are also considered in this step as robots that can only be Leadable in one direction (i.e. the obstacle must lead the robots).

Finally, if all the pairs are Leadable in at least one direction, then the test continues to see if the set is Leadable. For each path pair that is only Leadable in one direction, a consistency check is made to ensure that no ordering conflicts would prevent the set from being Leadable (e.g if the only lead conditions are (Q, R) *Leadable*, (R, S) *Leadable* and (S, T) *Leadable*, then $\{Q, R, S\}$ is not a Leadable set). If an ordering conflict

exists the milestone is rejected, otherwise the milestone is determined as belonging to the endgame region.

Given n robots, $R = 0.5n(n-1)$ Leadable pair checks are required. To check consistency between pairs, let $X_{\text{unidirectional}}$ be the set of all trajectory pairs that are Leadable in only one direction. Clearly the size of $X_{\text{unidirectional}}$ is limited by R. For every pair in $X_{\text{unidirectional}}$, a maximum of n checks are done to see if combining multiple unidirectional constraints will create more (e.g. if Q must lead R and R must lead S then Q must lead S). If any such constraints lead to an inconsistency (e.g. Q must lead R and R must lead Q), then the consistency check fails. This requires an upper limit of $0.5n^3$ checks for consistency. In practical implementations, this limit is rarely approached.

The endgame region is summarized below. Note that only once the set is determined as being Leadable (i.e. a solution to the planning problem is found) does the planner actually assign a velocity profile to the paths. Once the solution is found, those robots, which lead all other robots are given the fastest velocity possible. From this assignment, it can be calculated at what time the lead robots will leave the intersection of other robot trajectories. To prevent collisions, these times are set as the minimum time that following robots can enter the intersections, dictating a maximum velocity for the following robot.

Definition. Let the *endgame region* be defined as the set of all milestones such that the arc paths connecting robots to their respective goals form a *Leadable* set. The following criteria must be satisfied to determine if a milestone belongs to the endgame region:

(1) Each arc path connecting a robot to its respective goal is collision-free with obstacles.
(2) Each pair of arc paths connecting robot states to their respective goals are Leadable.
(3) The leadability constraints force no ordering conflicts.

4.3.1. Endgame region simulation results

Simulations of two different scenarios were used to evaluate the use of velocity-tuned endgame regions. Four robots and four obstacles were placed in a bounded workspace and the planner was run for 0.5 s. For each scenario, two sets of simulations were run: one set where a velocity-tuned endgame region was used and one where no velocity-tuning was used in the endgame region definition. During these simulations, the number of expanded milestones that belong to the respective endgame regions was recorded.

To highlight the advantage of the new endgame region, results from two planning scenarios are compared in which one goal state is more confined than the other. The two scenarios are depicted in Fig. 9, in which the environment in (a) has been created by randomly selecting robots, obstacles and goal locations. In (b), a more constrained goal state was created. In 0.5 s of road map expansion, the average planner for case (a) produced 111 milestones belonging to the non-velocity-tuned endgame region, and 144 milestones belonging to the velocity-tuned endgame region. In this case, the increase in size of the velocity-tuned endgame region was largely offset by the increase in time taken to check for admissibility.

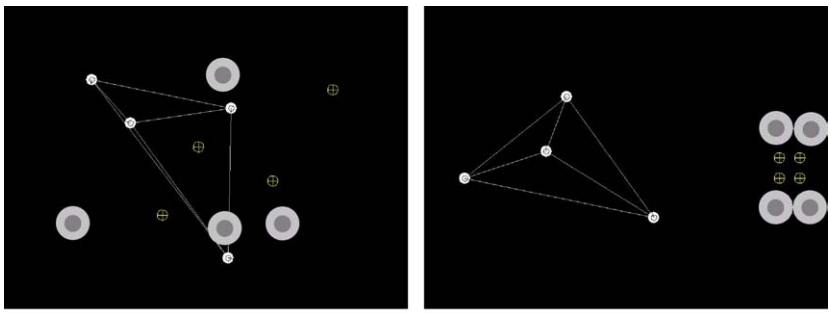


Fig. 9. Velocity-tuned endgame region: sample scenarios used to illustrate increased size of the endgame region attained when using velocity-tuning. The scenarios are illustrated as top-down views of environments involving four robots (white circles) and four obstacles (gray circles). Goal locations are depicted as gold cross-hairs.

However, in case (b), the average planner produced 1.5 milestones belonging to the non-velocity-tuned endgame region and 33 milestones belonging to the velocity-tuned endgame region. In many simulations, the planner never found a solution when no velocity-tuning was used. This illustrates a clear advantage of using a velocity-tuned endgame region when tight-coordination is required to attain the goal state.

5. Robot planning results

Simulations were run to characterize the performance of the planner for a multi-robot system with up to 12 robots. To accomplish this, a particular test scenario was chosen that highlights the characteristics of the coordination platform and motion planner.

In this scenario, 12 rovers of diameter 5 cm are operating in a 2 m × 3 m flat workspace amidst six stationary and six moving circular obstacles of diameter 7 cm. To add complexity to the scenario, four of the moving obstacles were directed towards a network of two robots with little room to maneuver. Also, two networks of two robots were placed between a row of three obstacles and a workspace boundary. The scenario was run 25 times with different initial random seeds. The planner demonstrated fast planning times (an average of 17.3 ms), while planning for up to five robots in a network. This speed enables the on-the-fly planning that is required for operation in dynamic, unknown environments.

Throughout the simulations that lasted several minutes, robots formed on average 49 different networks. This illustrates the ability for centralized planning despite the continuous merging and breaking of networks.

To illustrate the applicability of the planner to a 3D environment, simulations with up to eight free-floating space robots and eight obstacles were carried out. A test scenario was used in which robots must cross paths several times. The test scenario was simulated 25 times to produce the results in Table 1. From these results, it is clear that the planner was capable of planning on the fly with average planning times of 67 ms. An average of 12.2 networks were formed throughout each simulation.

Relative to the rover simulations, the planner was slower despite planning for fewer robots. This is attributed to the requirement for a different endgame region definition. A bang-off-bang control sequence was used to connect milestones to the goal. This produced efficient trajectories, but the overhead in calculating them was substantial. In the future, it is recommended that robots use a spline function to connect candidate milestones to the goal state [15].

Table 1
Simulation data for rover and free-floating robot test scenarios

Simulation	Rovers	Free-floaters
Average number of robots per plan	2.12	1.84
Average planning time (ms)	17.3	67.0
Average number of plans per robot simulation	5.07	4.77
Average number of networks formed per simulation	49.4	12.2

In Fig. 10, a visualization of robots navigating in a walled-in, multi-level environment is provided. Within these scenarios, robot coordination within networks is not only triggered through Event Detection, but by a single robot that requests new coordination plans with a set frequency. Not only does this demonstrate the platform's ability to coordinate robot actions at a frequent rate, but that re-planning can be used to attain better trajectories (according to some pre-determined cost-function). The example involves four rovers. The goal locations for the rovers are located in the middle of the environment's central platform. As shown in Fig. 10, initial robot trajectories lead robots over drop-offs in unexplored regions of the environment. However, as the rovers traverse these areas, they learn more about the environment. With new information, robots construct new plans that allow for safe movement. This process continues until robots eventually reach their goals.

In attempt to optimize trajectories, one robot within each network (e.g. that with the lowest priority number), calls for a new plan every 2.0 s regardless of whether there is new information. Robots compare the newly constructed plan with the currently implemented plan. They implement the better of these plans, where the better plan is determined by some predetermined cost function. This assumes the previous constructed

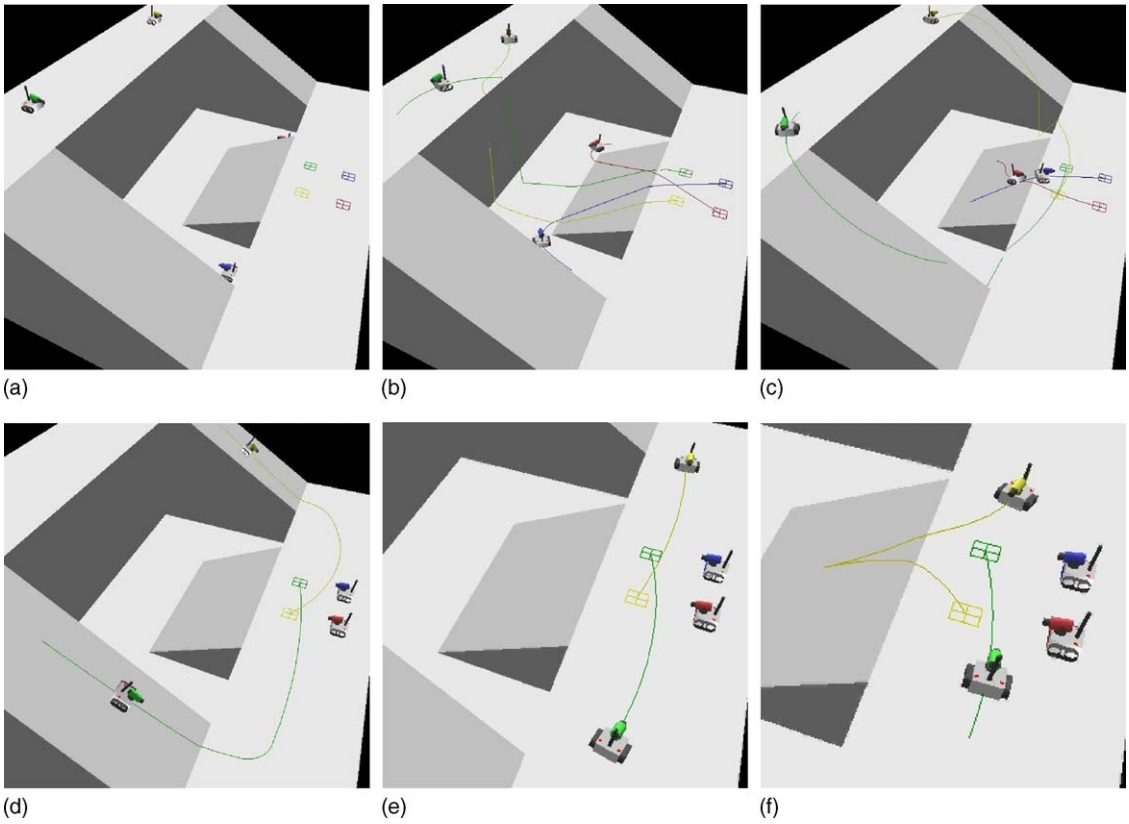


Fig. 10. Robot motion planning simulation.

plan is still feasible. If not, then no comparison is carried out and the new plan is implemented.

5.1. Rover experiments

To exemplify the system's ability to function on real hardware, an experiment is documented below involving five rovers and four obstacles. The experiment is depicted in Fig. 11, where a series of screen-shots of the GUI are on the left with the corresponding hardware photos on the right. Four of the robots are lined up on the left rail of the test-platform and their goals are located in a line on the right side. The top two of these four robots are close enough to form a local communication network. The goal locations for these two robots are located on the other side of the platform, but swapped such that the lines connecting these two robots to their goal locations will intersect. Likewise, the bottom two of these four robots are also close enough to form their own network and have a similar "swapped" goal configuration. The fifth robot, located in the upper right, has a goal location in the upper left. Initially, there are three static obstacles in a line down the middle of the test-platform, and another obstacle located in the bottom right that moves across the table.

This experiment not only illustrates that the planner can function on real robots, but it highlights the planner's ability to handle:

(1) On-the-fly centralized motion coordination—Planning times were all less than 50 ms which enabled robots to plan new trajectories as they moved. One example of this occurred between Fig. 11a and b, when the top two robots on the left had to replan to avoid the middle stationary obstacle that was initially out of sensing range.
(2) Avoidance of moving and previously unknown obstacles—The two bottom robots planned together within their network to avoid an obstacle heading directly for them (see bottom of Fig. 11d).

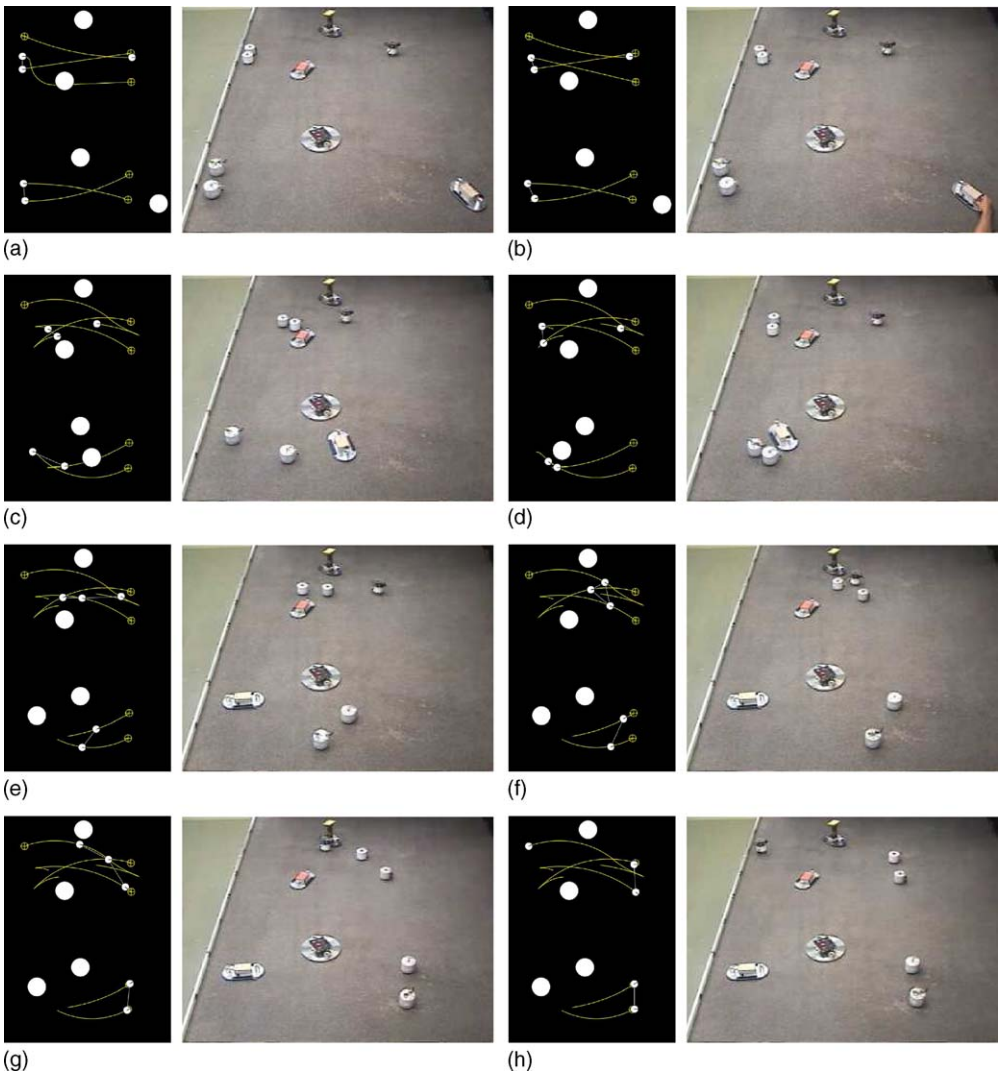


Fig. 11. Dynamic robot network experiment.

5.2. *Probabilistic completeness*

Given certain assumptions, Hsu's et al. algorithm is proven to probabilistically complete [16]. That is, it has an exponentially fast convergence for general motion planning problems, including multi-robot planning problems. The analysis is based on two simplifying assumptions: that the configuration space is expansive, and that the coverage converges to a uniform distribution over the configuration space. These assumptions are difficult to verify. Hence, simulations were conducted to demonstrate the exponential convergence rate of the *coupled* planner presented in this paper.

Simulations were run for six different scenarios of varying complexity, involving up to 5 robots and 10 obstacles within in a 2D workspace (only four sets of results are provided here, see ref. [12] for additional results). For each simulation, the planner was allowed to expand until a certain number of milestones, say x milestones, were added to the road map. The value of x was varied for each scenario, with 100 searches run for each value of x.

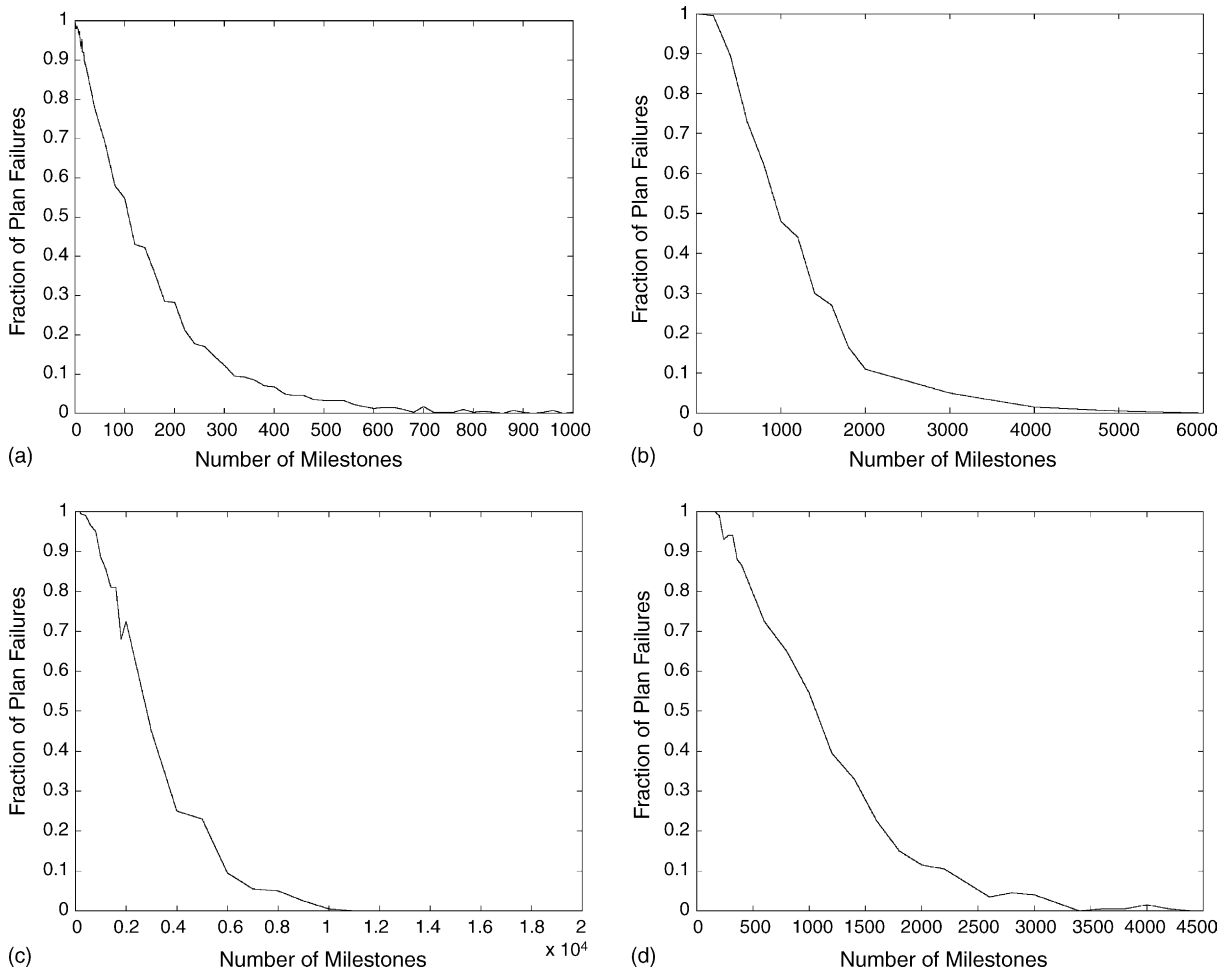


Fig. 12. Exponential decay of planner failure.

A summary of the simulation results are plotted in Fig. 12 as the ratio of failure for increasing values of x. As expected for probabilistic complete planners, there is an exponential decay in the failure rate.

6. Conclusions

This paper presents a new approach to multi-robot motion planning based on implementing Probabilistic Road Map planning techniques within the Dynamic Robot Network (DRN) coordination platform. Results indicate the DRN platform functions well even when frequent network merges or breaks occur. Robot coordination was carried out successfully under such conditions, allowing robots to achieve their goal states.

Also presented were new strategies for increasing the speed of a PRM motion planner when used to plan trajectories for multiple mobile robots. First a method of sampling PRM milestones for expansion was identified for multi-robot motion planning. The hyper-grid method was extended to provide fast coverage of the configuration space.

Second, the serial expansion method of milestone generation was introduced. As predicted, this method proved to require fewer collision-checks than the traditional parallel expansion method. This resulted in faster road map expansions.

Finally, a new endgame region was defined based on the concept of velocity-tuning. This definition demonstrated improved likelihood of finding solutions when goal configurations are highly constrained.

With the help of these new sampling strategies, the PRM motion planner was implemented within the Dynamic Robot Network coordination platform. Successful on-line trajectory planning was demonstrated with average planning times on the order of 20 ms. This enabled on-the-fly planning for avoidance of moving obstacles and allowed multiple robots to navigate in environments that are both unknown and dynamic.

References

[1] N.M. Amato, Y. Wu, A randomized roadmap method for path and manipulation planning, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1996, pp. 113–120.

[2] T. Arai, J. Ota, Motion planning of multiple mobile robots, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 1992, pp. 1761–1768.

[3] K. Azarm, G. Schmidt, Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1997, pp. 3526–3533.

[4] J. Barraquand, B. Langlois, J.C. Latombe, Numerical potential field techniques for robot path planning, IEEE Trans. Syst., Man Cybern. (2) (1992) 224–241.

[5] J. Barraquand, J.C. Latombe, Nonholonomic multibody mobile robots: controllability and motion planning in the presence of obstacles, Algorithmica 10 (2–4) (1993) 121–155.

[6] J.S. Bellingham, M. Tillerson, M. Alighanbari, J.P. How, Cooperative path planning for multiple uavs in dynamic and uncertain environments, in: Proceedings of the IEEE Conference on Decision and Control, December, 2002.

[7] M. Bennewitz, W. Burgard, S. Thrun, Optimizing schedules for prioritized path planning of multi-robot systems, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2001.

[8] J. Broch, D.A. Maltz, D.B. Johnson, Y.-C. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: Proceedings of the 4th ACM/IEEE International Conference on Mobile Computing and Networking, Dallas, TX, 1998, pp. 85–87.

[9] S.J. Buckley, Fast motion planning for multiple moving robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1989, pp. 1419–1424.

[10] C. Clark, S. Rock, Randomized motion planning for groups of nonholonomic robots, in: Proceedings of the International Symposium. of Artificial Intelligence, Robotics and Automation in Space, 2001.

[11] C. Clark, S. Rock, J.C. Latombe, Dynamic networks for motion planning in multi-robot space systems, in: Proceedings of the International Symposium of Artificial Intelligence, Robotics and Automation in Space, 2003.

[12] C.M. Clark, Dynamic Robot Networks: A Coordination Platform for Multi-Robot Systems, Ph.D. Thesis, Stanford University Press, 2004.

[13] M. Erdmann, T. Lozano-Perez, On multiple moving objects, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1986, pp. 1419–1424.

[14] Y. Guo, L.E. Parker, A distributed and optimal motion planning approach for multiple mobile robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, Washington, DC, 2002, pp. 2612–2619.

[15] D. Hsu, R. Kindel, J.C. Latombe, S. Rock, Randomized kinodynamic motion planning with moving obstacles, Int. J. Robotic Res. 21 (3) (2002) 233–255.

[16] D. Hsu, J.C. Latombe, R. Motwani, Path planning in expansive configuration spaces, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1997, pp. 2719–2726.

[17] D. Hsu, L.E. Kavraki, J.C. Latombe, R. Motwani, S. Sorkin, On finding narrow passages with probabilistic roadmap planners. In: P.K. Agarwal, L.E. Kavraki, M.T. Mason (Eds.), Robotics: The Algorithmic Perspective, Workshop on Algorithmic Foundations of Robotics, A.K. Peters, Natick, MA, 1998, pp. 141–153.

[18] L.E. Kavraki, P. Svestka, J.C. Latombe, M. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Trans. Robotic Automation 12 (4) (1993) 566–580.

[19] L.E. Kavraki, Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, Ph.D. Thesis, Computer Science Department, Stanford University, 1994.

[20] R. Kindel, Motion Planning for Free-Flying Robots in Dynamic and Uncertain Environments, Ph.D. Thesis, Stanford University Press, 2001.

[21] K. Kant, S. Zucker, Toward efficient trajectory planning: the path-velocity decomposition, Int. J. Robotic Res. 5 (3) (1986) 72–89.

[22] S. Kato, S. Nishiyama, J. Takeno, Coordinating mobile robots by applying traffic rules, in: Proceedings of the IEEE/RSH International Conference on Intelligent Robots and Systems, 1992, pp. 1535–1541.

[23] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Int. J. Robotic Res. 5 (1) (1986) 90–98.

[24] S.M. LaValle, S.A. Hutchinson, Optimal motion planning for multiple robots having independent goal, IEEE Trans. Robotic Automation 14 (1998) 912–925.

[25] S.M. LaValle, J.J. Kuffner, Randomized kinodynamic planning, Int. J. Robotic Res. 20 (5) (1998) 278–300.

[26] V.J. Lumelsky, K.R. Harinarayan, Decentralized motion planning for multiple mobile robots: the cocktail party model, Autonomous Robots J. 4 (1997) 121–135.

[27] S. Martel, I. Hunter, Nanofactories based on a fleet of scientific instruments configured as miniature autonomous robots, J. Micromechatronics (2003).

[28] M.J. Mataric, Using communication to reduce locality in distributed multi-agent learning, J. Exp. Theor. Artif. Intell. 10-3 (1998) 357–369.
[29] L.E. Parker, ALLIANCE: an architecture for fault tolerant multi-robot coordination, IEEE Trans. Robotics Automation 14 (2) (1998) 220–240.
[30] D. Parsons, J. Canny, A motion planner for multiple mobile robots, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1992, pp. 8–13.
[31] J. Peng, S. Akella, J.D. Boissonnat, J. Burdick, K. Goldberg, S. Hutchinson, Coordinating multiple robots with kinodynamic constraints along specified paths, in: Algorithmic Foundations of Robotics V (WAFR 2002), Springer-Verlag, 2003, pp. 221–237.
[32] J.A. Reeds, L.A. Shepp, Optimal paths for a car that goes forwards and backwards, Pacific J. Math. 145 (2) (1990) 367–393.
[33] A. Richards, J.P. How, Aircraft trajectory planning with collision avoidance using mixed integer linear programming, in: Proceedings of the American Control Conference, May, 2002.
[34] E. Royer, C.K. Toh, A review of current routing protocols for ad-hoc mobile wireless networks, IEEE Pers. Commun. Mag. (1999) 46–55.
[35] G. Sanchez, J.C. Latombe, On delaying collision checking in PRM planning: application to multi-robot coordination, Int. J. Robotics Res. 21 (1) (2002) 5–26.
[36] G. Sanchez, J.C. Latombe, Using a PRM planner to compare centralized and decoupled planning for multi-robot systems, Proc. IEEE Int. Conf. Robot. Autom. (2002).
[37] D.H. Shim, H.J. Kim, S. Sastry, Decentralized reflective model predictive control of multiple flying robots in dynamic environment, in: Proceedings of the IEEE Conference on Decision and Control, December, 2003.
[38] T. Simeon, S. Leroy, J.P. Laumond, Path coordination for multiple mobile robots: a geometric algorithm, in: Proceedings of the International Joint Conference on Artificial Intelligence, 1999.
[39] J. Svennebring, S. Koenig, Trail-laying robots for robust terrain coverage, in: Proceedings of the International Conference on Robotics and Automation, Taipei, Taiwan, 2003.
[40] P. Svestka, M.H. Overmars, Coordinated motion planning for multiple car-like robots using probabilistic roadmaps, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1995, pp. 1631–1636.
[41] C.W. Warren, Multiple path coordination using artificial potential fields, in: Proceedings of the IEEE International Conference on Robotics and Automation, 1990, pp. 500–505.

Dr. Clark is an assistant professor at the University of Waterloo, Ont., Canada. His education includes a BASc in engineering physics from Queen's University, a MASc in mechanical and industrial engineering from the University of Toronto and a PhD in aeronautics and astronautics with a minor in computer science from Stanford University. Dr. Clark's professional experience includes working as a control systems designer for Sterner Automation Limited and as a consulting software architect for robot control applications. Within the University of Waterloo's Lab for Autonomous and Intelligent Robotics (LAIR), Dr. Clark directs research in the areas of multi-robot navigation, 3D localization, underwater robotics, modular reconfigurable robotics and collaborative vehicle control.