Decentralized Adaptive Path Selection for Multi-Agent Conflict Minimization

Andrew Kimmel and **Kostas Bekris**
Department of Computer Science, Rutgers University
Piscataway, New Jersey 08854

Abstract

Consider multiple robots moving towards individual goals in a cluttered environment. While contacts between robots in these situations can be averted by reactive collision avoidance methods, deadlocks may arise in tight spaces if robots move along precomputed, conflicting paths. To resolve these issues, methods have been proposed which consider robots that employ communication, or centralized planning, or follow predefined rules. This work considers only decentralized planning solutions that employ minimum information, i.e., each robot has access only to the current position of its neighbors, without using any form of prediction, intent recognition or agent modeling. This leads to a study of several methods for minimum-conflict path selection among dynamic obstacles. The evaluation of these methods in varying simulated benchmarks, provides the following insights: (a) considering the "minimum-conflict" path given the other agents' current positions is critical for deadlock avoidance, (b) reasoning over a diverse set of paths that provide multiple alternatives improves path quality, and (c) accumulating cost over alternative paths by finding the agent's best path selection in hindsight allows robots to learn effective high-level strategies in a computationally efficient way that is adaptive to the other agents' behavior, reducing task completion time and average path lengths. [1]

Introduction

The proliferation of robotic technology allows for applications where multiple autonomous systems effortlessly interact in the same cluttered environment, while solving individual tasks. For example, consider ground vehicles that operate in a mine or a construction facility and which need to move efficiently to solve their assigned task while avoiding collisions. In many interesting challenges, it is also highly likely that people or animals are navigating in the same space and need to be considered. Explicit coordination with the other agents in the environment, especially when humans or animals are involved, may not feasible nor desirable. Similarly, it may be difficult to model or predict the actions of moving obstacles. This necessitates the consideration of decentralized methods that allow a robot to make progress towards its goal in a safe manner with minimal information and without strong assumptions about the intentions of its moving neighbors. It is also desirable for the solution to minimize certain parameters, such as executed path length or task completion time, and for the motions to look natural to people that operate in the same space.

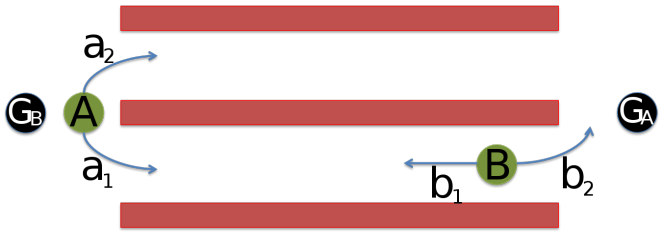


Figure 1: The base case for this study: if both robots A and B insist on following the same corridor, then given the environment characteristics and their geometry, a reactive collision avoidance method may not allow them to make progress towards their goals, G_A and G_B, which are on opposite sides.

In environments without obstacles, reactive methods are able to perform well even for large numbers of agents. In the presence of obstacles, however, planning is also needed to compute a global solution path. Solving problems that combines complex geometry and dense distribution of moving agents requires the use of both a high-level planner, operating in a replanning framework with a reasonably small planning cycle, so as to adapt to the frequent changes, as well as a low-level reactive collision avoidance technique.

Challenges, Foundations and Objectives

Avoiding collisions with unexpected obstacles or unpredictable, self-interested mobile agents, can be effectively addressed by reactive collision avoidance methods, such as those based on the popular Velocity Obstacle framework [Fiorini and Shiller1998, van den Berg et al.2011] or trajectory deformation methods [Fraichard and Delsart2009, Karamouzas, Geraerts, and Overmars2009]. These methods generally provide smooth, natural-looking paths, but they are primarily local techniques and do not reason about the robot's global path. If the agents select conflicting paths in a decentralized manner, reactive collision avoidance can still give rise to deadlocks and poor performance. For instance,

[1] A related paper to this report has been submitted to IROS 2014.

consider the situation in Figure 1, where two robots on opposing sides of two corridors need to exchange positions. If both robots decide to move along the lower corridor, e.g., because it corresponds to their individual shortest paths, the space is narrow enough to prevent the robots from swapping positions.

Robots in such situations that replan [Petti and Fraichard2005] and change their path to a different homotopic class [Bhattacharya, Kumar, and Likhachev2010, Bhattacharya, Likhachev, and Kumar2011, Jaillet and Siméon2006] can potentially resolve such conflicts. This type of coordination can be achieved by assuming that the robots share information [Bekris et al.2012], or follow a form of centralized planning [Qutub, Alami, and Ingrand1997], or by respecting a set of pre-specified "social" rules [Knepper and Rus2012, Trautman and Krause2010], or performing sophisticated prediction [Large et al.2004, Thompson, Horiuchi, and Kagami2009, Ziebart et al.2009], agent modeling [Shi et al.2008, Sisbot et al.2007] or learning [Bennewitz et al.2005, Henry et al.2010]. The type of information required in order to be able to use such solutions may correspond a) to the actions selected by neighbors, b) the utilities of different motions, c) the goals of neighbors, or d) extensive prior experience interacting with other agents. Such information is difficult to attain quickly and reliably, especially when a robot interacts with a human, since the robot has little knowledge about the human's future actions without explicit communication. This work employs strictly decentralized methods while utilizing minimal information to solve this problem. Each robot has access only to the current position of its neighbors from sensor data. A further objective is to identify what can be achieved without any prediction, intent recognition or modeling of the moving agents. This method employs learning, corresponding primarily to online learning of appropriate strategies in response to environmental conditions.

This planning-based work is complementary to reactive collision avoidance methods and shares the desire of requiring minimal information and assumptions about the other agents. The Velocity Obstacles framework requires knowledge of the velocity of neighboring agents, which is not utilized by the methods proposed here. Reciprocity between the agents utilizing the same method is desirable, as illustrated with Reciprocal Velocity Obstacles [Snape et al.2011, van den Berg et al.2011], so that in situations as the one depicted in Figure 1, one of the agents will reliably decide to switch corridor even if both agents originally selected to move along the same one.

Considered Methodology

The above objectives lead to the consideration of various methods for computing decentralized, minimum-information, minimum-conflict path selection strategies. These methods are evaluated in terms of their effectiveness in various simulated benchmarks. The basic framework assumes that the robots follow a replanning approach to compute paths frequently [Petti and Fraichard2005] and then they employ reciprocal velocity obstacles [van den Berg, Lin, and Manocha2008, Snape et al.2011] so as to follow these paths while avoiding collisions. If agents make greedy choices to use the shortest path to their goal and ignore the presence of other agents, this approach leads to deadlocks in the considered environments.

One idea is to compute a family of diverse paths instead of only the shortest one. The notion of path diversity has been shown to be helpful in different challenges, but frequently corresponds to a local concept [Green and Kelly2007, Knepper and Mason2009]. If a roadmap for the scene is available, then an effective way to compute this set of paths is to compute trajectories belonging to different homotopic classes, using search-based primitives [Bhattacharya, Kumar, and Likhachev2010, Bhattacharya, Likhachev, and Kumar2011]. This work provides a method for selecting a minimally conflicting path out of this set given the other agents' current positions. This process is designed so as to effectively and reliably address the issue in the prototypical example provided in Figure 1. If the replanning cycle of the robot is short, this process yields a high-level of reactivity and deconfliction.

While this is an improvement over considering only the shortest path, it also causes computational issues in scenes with many homotopic classes. If only a small set of short in length, homotopically-distinct paths are generated each planning cycle, deadlocks still arise. For instance, this occurs if none of the paths permit the robot to backtrack so as to allow other agents to make progress. Nevertheless, there is a way to address this issue according to simulated results. If the "minimum-conflict" homotopy is always included in the set of considered actions, deadlocks are not observed. The notion of minimum conflict considered here is related to the "minimum constraint displacement" problem, which has attracted attention recently in motion planning [Hauser2013]. For the current work, constraints on the minimum-conflict homotopy correspond to the observed locations of other agents. This homotopy can be discovered in a computationally effective manner. Using an underlying roadmap, the method computes the shortest path that also minimizes the number of collisions with other agents.

While minimum conflict paths achieve deadlock avoidance, they are also conservative, and may take a long time to reach the robots' goals. Considering both the minimum-conflict path and a set of shortest, homotopically-distinct paths towards the goal that ignore the other agents typically results in better performance. Under the "garage" assumption, where agents essentially "disappear" from the workspace when they reach their goals, and assuming that the workspace is unbounded, by strictly navigating using only minimum conflict paths, results seem to indicate that liveness can be guaranteed. This work considers both a deterministic approach for choosing between these paths, as well as a learning-based, probabilistic approach. Both solutions always find a solution and avoid deadlocks in simulations. The probabilistic approach is also able to provide adaptivity to various behaviors of neighbors.

The probabilistic solution is based on the Polynomial Weights `PW` algorithm [Littlestone and Warmuth1994, Nisan et al.2007], which is a learning-based approach for achieving regret minimization. Given an individual agent's action set, `PW` accumulates a weight on each action, which directly

corresponds to the action's probability of being selected. At each step of the framework, an agent evaluates its most recently executed action and computes the regret for that action, which is evaluated by considering the current state of the world compared to the previous state. Essentially, the agent thinks "What would have happened if I selected a different action, given the current observed positions of the other agents?", and in doing so, the agent can compute its best action in hindsight. The difference in action cost between the best action in hindsight and the actual selected action, corresponds to the agent's regret for the selected value, which will directly reduce the action's weight and thus its probability of selection.

Regret minimization is advantageous in this context, as it results in high expected utilities against unpredictable agents, without knowledge of the other agents' goals, utilities, intents, or beliefs. The application of the PW algorithm here accumulates regret for the "minimum-conflict" strategy and the "greedy" choice strategy at each replanning cycle by observing the choices of the other agents in the same workspace and assigning a loss to each strategy. A probability is then assigned for selecting each strategy based on the accumulated losses.

Such learning-based approaches to coordination and game theoretic challenges have been considered by others in the robotics literature, where a reinforcement learning method has been used to create adaptive, loosely coupled, agents [Kaminka, Erusalimchik, and Kraus2010]. Some approaches qualitatively measure the effectiveness of coordination between agents offline to provide action selection online [Excelente-Toledo and Jennings2004]. There has been work in coordinating with agents that are not necessarily rational [Stone, Kaminka, and Rosenschein2010]. All of these approaches require knowledge of the agents' actions, which is not assumed here. Although there are models for predicting the actions of an agent, such as in an adversarial setting [Wunder et al.2011], by utilizing regret minimization [Filiot, Le Gall, and Raskin2010], it is possible to solve certain games without knowing the other agent's actions.

Contribution and Overview of Results

The key observations from this work are the following:

a) Computing minimum-conflict paths is critical for avoiding deadlocks under a minimum information setup for planning among dynamic obstacles and is interesting to further analyze the properties of this strategy.

b) Computing paths in different homotopies is a useful primitive for providing diverse alternatives to robots to improve the resulting path quality and execution time.

c) Regret minimization is computationally efficient and allows robots to optimize and learn over time an appropriate strategy given the characteristics of the underlying challenge, without explicit communication.

Simulations show that the considered solutions allow robots to avoid deadlocks, minimize completion time and path length for solving such problems with minimal information requirements.

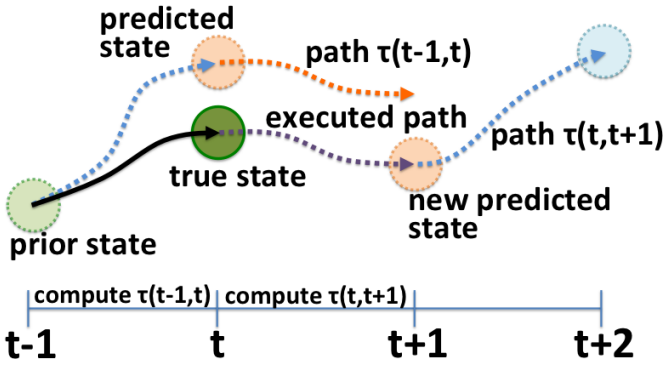


Figure 2: An illustration of the replanning framework. Here the path computed between time t-1 and t is executed during time t to t+1. Execution of the plan means the state at time t deviates from the predicted state, so the framework begins planning for t+1 to t+2 from an updated predicted state.

Problem Setup

Path deconfliction problems can be defined in general configuration spaces but this discussion will focus on holonomic navigation as it provides an easy way to describe the framework and corresponds to the accompanying simulations.

Consider a set of m planar, holonomic agents $\{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$ that move with bounded velocity $v \in [0, v_{max}]$ in the same workspace $\mathcal{W}$. The configuration space of an agent is $\mathcal{Q} = \mathbb{R}^2$, where $\mathcal{Q}_{free}$ represents the obstacle free subset given static obstacles. Given a configuration $q_i \in \mathcal{Q}$, the expression $\mathcal{A}(q_i)$ corresponds to the collision volume of agent $\mathcal{A}_i$ in $\mathcal{W}$.

Then a path $\pi_i = \{q_i | q_i : [0,1] \rightarrow \mathcal{Q}_{free}\}$ for agent $\mathcal{A}_i$ corresponds to a continuous curve in $\mathcal{Q}_{free}$. Given a time scaling function $\sigma_i : \mathbb{R}^{\geq 0} \rightarrow [0,1]$ it is also possible to define the agent's trajectory $\tau_i = \pi_i \circ \sigma_i$, which defines the configurations that the agent visits at each point in time.

The problem formulation assumes that all agents want to solve a similar problem, as in each agent $\mathcal{A}_i$ wants to reach a desired goal $q_i^{\mathcal{G}} \in \mathcal{Q}$ without conflicts. The objective then is for the agents to select trajectories $\{\tau_1, \ldots, \tau_m\}$ in a decentralized manner, such that each $\mathcal{A}_i$ follows an individual τ_i and in finite time T: $\forall\, i \in [1,m] : \tau_i[T] = q_i^{\mathcal{G}}$. Collisions between agents must be avoided, unless one of the agents has reached its goal, i.e. $\forall\, t \in \mathbb{R}^{\geq 0}, \forall\, i,j \in [1,m]$:

$$\mathcal{A}(\tau_i[t]) \cap \mathcal{A}(\tau_j[t]) = \emptyset \;\vee\; \mathcal{A}(\tau_i[t]) = q_i^{\mathcal{G}} \;\vee\; \mathcal{A}(\tau_j[t]) = q_j^{\mathcal{G}}.$$

The above expression implies the so called "garage" assumption, where agents which reach their goal are freed from the workspace and are not considered for collisions when other agents pass through their goal.

Agents are never aware of the goal of any other agent or the trajectory selected by another agent. At any point in time an agent can only observe the position of other agents as long as their configurations are within a certain sensing radius: $||(q_i, q_j)|| \leq d^{sense}$. Furthermore, the agents use a collision avoidance method (e.g., Reciprocal Velocity Obstacles [van den Berg, Lin, and Manocha2008]), which is used to follow their selected trajectory while still avoiding

collisions with other agents. This means that the planned trajectory may not be executed perfectly due to the influence of neighboring agents.

Note that the above discussion can be easily extended to include the case where one agent is the planning robot that employs a method for achieving deconfliction while all the other agents are unpredictable dynamic obstacles that ignore the presence of the planning agent. In these situations, the relative velocity of the planning agent and the dynamic obstacles should be such so that the collision avoidance method can always guarantee the safety of the planning agent.

The above fact, together with the need to adapt to the unpredictable behavior of neighboring agents, motivates a replanning framework for recomputing trajectories given the latest observed configurations of agents. This replanning approach forms the basis of the overall methodology that is described in the following section.

Methods

This section first describes a classical method for integrating global path planning and local collision avoidance, which can lead to deadlocks in certain environments. Then a sequence of alternative strategies for computing the global path are considered so as to avoid such situations.

Replanning Framework

During the execution of a plan, a robot's trajectory will deviate from the planned trajectory due to the reactive collision avoidance utilized. Naïvely following the original planned trajectory is therefore not sufficient, as the robot will most likely not be able to reach its goal as intended. A straightforward replanning framework [Petti and Fraichard2005,Hauser2011] as illustrated in Figure 2 is used to address such issues. The framework follows related work, where first, a roadmap is precomputed using a sampling-based motion planning method and then integrated with a collision-avoidance method [van Den Berg et al.2008]. The sampling-based planner used in this work is PRM* [Karaman and Frazzoli2011].

The trajectory computed for time t-1 to t will not be executed perfectly, as shown in Figure 2; however, the framework updates the predicted state of the robot accordingly. From the agent's state at time t-1, the robot created a plan which should have brought it to the predicted state at time t, but the collision avoidance led it instead to its true state. The plan for time t to t+1 is then propagated from the true state of the robot to obtain a new predicted state, and this state for time t+1 is used to plan for time t+1 to t+2. In this way, the robot can iteratively correct deviations from its plan, applying this continuously until it is able to reach its goal.

By using such a replanning framework, where the agent assumes its previous selected plan will be executed perfectly and plans for the next time step, the agent becomes robust to perturbations in the solution path caused by these reactive methods. When integrated with a reactive collision avoidance method, the traditional framework selects the shortest path to the goal ignoring other agents. As argued before, this choice can lead to deadlocks despite the availability of the reactive collision avoidance method.

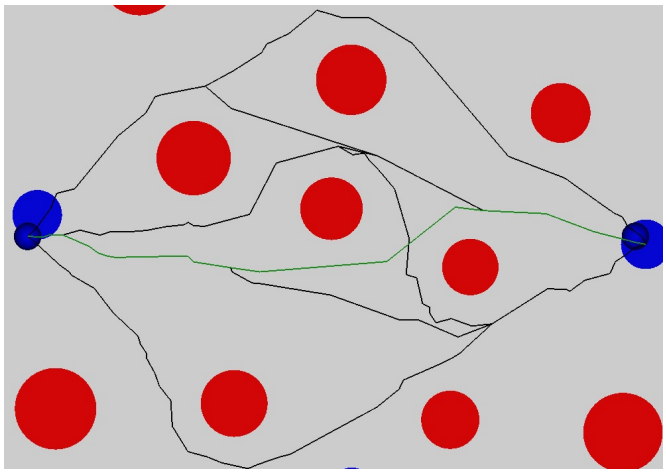


Figure 3: An example of a set of paths belonging in different homotopic classes for the right-side agent, bringing it to the left-side of the environment.

K-best Paths from Different Homotopic Classes

Rather than simply selecting the greedy path at each planning cycle, one alternative is for each agent to consider a diverse set of paths which bring the agent to its goal. This work accomplishes this by restricting the set of paths generated, such that they all must belong to different homotopic classes [Bhattacharya, Likhachev, and Kumar2011]. When considering 2D problems, trajectories are in different homotopy classes when the area between them contains an obstacle. A complete definition for homotopies can be found in the related literature [Hatcher2002].

By ignoring paths that loop around obstacles, the set of non-homotopic paths describes all of the shortest-length paths that bring the agent from its current position to the goal. These computations take place over an underlying roadmap, and use a set of search-based primitives. An example of the resulting set of computed paths in a simulated environment is shown in Figure 3.

The "k-best" strategy therefore corresponds to the following: at each replanning cycle, agents compute a set of k paths belonging to k different homotopies and select a single path from this set as the agent's action. Considering such a set of actions, instead of just the shortest path, provides the agent with more choices. It is then possible to select a path, not just greedily in terms of its length, but also in terms of its interactions with neighbors.

Minimizing Interaction Cost

The question now arises on how agents can differentiate among a set $\mathcal{S}$ of available paths from different homotopic classes in order to select motions that will allow the agents to make progress towards their goals. To describe the process employed by this work, consider the situation depicted in Figure 1, where agent A has actions a_1 to move through the lower corridor and a_2 to move through the upper corridor towards its goal. These actions correspond to two solutions of the homotopic computation described in the previous section, regardless of the current configuration of the robot q_A.

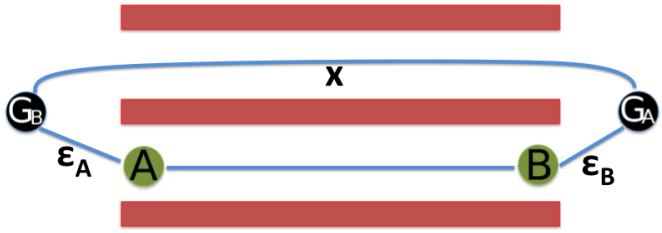


Figure 4: For the base case: for both of the robots the length to backtrack out of the current corridor and move to the other corridor is ϵ_A and ϵ_B for robots A and B respectively, while remaining in the current corridor reduces path length by ϵ_A and ϵ_B respectively.

Similarly, agent B has choices b_1 and b_2.

Then the question is how costs $C(a_1)$, $C(a_2)$, $C(b_1)$, $C(b_2)$ can be computed appropriately, and in a decentralized manner, so that in any situation the two agents will decide to follow different corridors when they try to select the action with minimum cost. A conflict arises when both robots are already following the path down the same corridor. Without loss of generality set both agents to be inside corridor 1, i.e. the lower corridor.

Assume that the goals for the agents are symmetrically placed at the end of each side of the corridor. Then the shortest path between the two goal points through the corridors is x, as illustrated in Figure 4. If the corridors are too narrow, then the paths will go through the current configurations of robots A and B. Assume that the distance between the goal G_B and q_A is ϵ_A and the distance between the goal G_A and q_B is ϵ_B along the path that goes through corridor 1. Then the lengths of the shortest paths for the robots to reach their goals via the corresponding homotopic paths can be computed as follows:

$$P_1^A = x - \epsilon_A, \quad P_2^A = x + \epsilon_A$$

$$P_1^B = x - \epsilon_B, \quad P_2^B = x + \epsilon_B$$

Where P_i^X corresponds to the length of the shortest path for robot X from its current configuration q_X to its goal G_X via corridor i.

The proposed approach also considers an interaction cost along each action for every agent. The interaction cost of an action is 0 if there is no other agent occupying the corresponding path given the latest observation. If there is an agent occupying the path, then the interaction cost is computed as follows:

$$I_i^A = 1 - \frac{\text{distance between A and B along } \pi_i}{\text{length of } \pi_i} \tag{1}$$

The reasoning behind this definition is that agents closer to the current position of an agent should incur a higher interaction cost. Then for the above scenario the interaction costs are:

$$I_1^A = \frac{\epsilon_B}{x - \epsilon_A}, \quad I_2^A = 0$$

$$I_1^B = \frac{\epsilon_A}{x - \epsilon_B}, \quad I_2^B = 0$$

Then the proposed cost function for actions is the following:

$$C_i^X = P_i^X(1 + 2 \cdot I_i^X) \tag{2}$$

which translates to the following cost in the above scenario:

$$C_1^A = x - \epsilon_A + 2 \cdot \epsilon_B, \quad C_2^A = x + \epsilon_A$$

$$C_1^B = x - \epsilon_B + 2 \cdot \epsilon_A, \quad C_2^B = x + \epsilon_B$$

Then, note that in order for A to select action 1 it has to be the case that:

$$C_1^A = x - \epsilon_A + 2 \cdot \epsilon_B < x + \epsilon_A = C_2^A \Rightarrow$$

$$A \text{ selects corridor 1 iff: } \epsilon_B < \epsilon_A \tag{3}$$

Similarly for robot B to select action 1 it has to be the case that:

$$C_1^B = x - \epsilon_B + 2 \cdot \epsilon_A < x + \epsilon_B = C_2^A \Rightarrow$$

$$B \text{ selects corridor 1 iff: } \epsilon_A < \epsilon_B \tag{4}$$

From Eqs. 3 and 4 it becomes apparent that the agents are not able to simultaneously pick the same corridor given the above definitions for the interaction cost and the overall cost functions. The agent who is farther away from its goal will have to pick the other homotopic class.

It is easy to check that if the agents had picked different corridors to enter then they would have stuck with their original choices as they would have incurred no interaction cost along the corridors that they would be moving. Similar reasoning can take place for the case that the environment has multiple corridors or the environment is the same and there are three agents A, B and C that are competing for the same corridor. Without loss of generality, if one assumes that in this case B is in the middle of the other two agents and C wants to move towards the same direction as B, there are two possible outcomes given the above definitions for the interaction cost and depending on the exact distances the three agents have traveled down the corridor:

- either A and C will be forced to change homotopic class and B continues down the same corridor,
- or A will continue moving along the same corridor and both B and C are forced to move to another corridor.

This means that C, which has the maximum number of conflicting agents along its path, will never decide to continue moving along the same corridor and the choice of A and B is forced to be different as in the case of two agents competing for the same corridor.

The entire above discussion was based on the assumption that the goal locations of the two agents were symmetrical relative to the corridors, i.e., the length of the path connecting the agents that goes through corridor 1 is the same as the length of the path through corridor 2. If the goals are not symmetrical, then instead of a common path length of x, the initial path costs P_i^X should include different lengths x_1 and x_2 for the connections of the goals via corridor 1 and 2 respectively. Then, the cost of actions should be defined in a general manner: $C_i^X = P_i^X(1 + \alpha \cdot I_i^X)$ for a constant α, which will depend on the relative difference $\Delta x = |x_1 - x_2|$. This is information, however, that is not

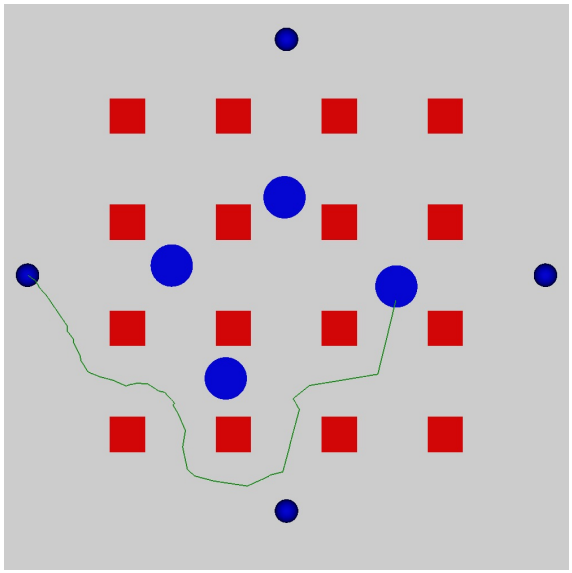


Figure 5: An example of a "minimum-conflict" path computed for the right-most agent for a goal on the left side of the image. In this situation the shortest path that does not conflict with any of the other agents is returned. In simulations with a larger number of agents, e.g., 32 in the above scene, it is typically the case that the "minimum-conflict" path still intersects with neighboring agents.

available to the robots, since it requires knowledge of the goals for the other agents.

In practice, using the value $\alpha = 2$ as was defined originally in this section, results in good performance in the classification of different homotopic paths in terms of their interaction cost. So, in the context of the replanning framework in order to replace the greedy choice, a "k-best" choice is the following:

- Use a homotopy computation algorithm to extract the k-shortest paths that belong to k different homotopic classes.
- For each one of these paths, compute their costs according to Eq. 2, where the interaction cost is computed according to Eq. 1.
- Return the action with minimum cost.

The action with minimum cost both minimizes distance from the goal as well as interaction with other agents. The above "k-best" strategy is superior to the "greedy" strategy of always selecting the shortest path, since it allows multiple alternative choices to the robot and considers interactions with neighbors given the reasoning that was presented here for the basic "corridor" challenge under the assumption that the goals of the two agents are symmetric.

Minimum Conflict (MC) Path

The "k-best" strategy does not result in deadlocks in simulations as long as the number of simple homotopic classes does not significantly exceed k, where simple homotopic classes correspond to those that do not include loops. When this property is true, then the consideration of interaction costs with other agents, as described in the previous section, results in the selection of homotopy classes which allow the team to make progress overall. Even in relatively simplistic scenes, however, the number of homotopic classes required to satisfy this condition can quickly become large. This introduces a computational challenge, since the $k+1$ homotopy class corresponds to a longer path, which translates to a longer search time on the underlying roadmap. To keep the proposed method effective, however, it is important to keep a small planning cycle and perform each path computation as fast as possible.

In order to address this issue, the value of k is kept relatively small, and to accommodate the potential lack of a desirable path, the current work proposes that the "minimum-conflict path" should always be included as an available action to the agents. To compute such a path, each agent $\mathcal{A}_i$ considers the current set of configurations for the agents it can observe: $\{q_1, \ldots, q_{i-1}, q_{i+1}, \ldots, q_m\}$. For each one of those configurations q_j, agent $\mathcal{A}_i$ marks edges in the roadmap that intersect q_j. Edges that are marked then have their weights inflated, effectively removing them from consideration during the heuristic search to find the shortest path on the roadmap from q_i to q_i^G. This means that the heuristic search process will first return the shortest path that does not collide with any agents. If no such path exists, then one which collides only with one agent will be returned and so on.

The inclusion of such paths in the set of available strategies results in methodologies that always solve challenges where the "greedy" or the "k-best" method failed. Interestingly, a strategy which only considers the "minimum conflict" action, constructed at each replanning cycle using the process described above, is also able to always solve all the challenges considered.

Even so, the resulting paths may not be as desirable when all the agents follow their minimum conflict action. As shown in Figure 5, this action may be significantly longer than the shortest path to the goal. Thus, it is interesting to consider the combination of the "k-best" strategy with the "minimum conflict" one. In this case, the process works as follows:

- Use a homotopy computation algorithm to extract the k-shortest paths that belong to k different homotopic classes.
- Compute the "minimum-conflict" action.
- For each one of the above paths, compute their costs according to Eq. 2, where the interaction cost is computed according to Eq. 1.
- Return the action with minimum cost.

This "deterministic" approach for combining the agent's greedy choices, i.e., k-shortest paths, and the safe choice, i.e., minimum conflict, takes again advantage of the process described in the previous section for evaluating a weighted cost of path length and interaction cost. It allows the agent to sometimes make the greedy choice and select one of the shortest paths, even if they conflict with other agents, as long as these paths are significantly shorter than the minimum

conflict path and do not overlap with other agents over a short horizon.

A Probabilistic Selection Strategy

To allow some adaptability to varying conditions, this work considers an online learning method to compute a non-deterministic policy for selecting the appropriate strategy out of the following: (a) the "minimum conflict", i.e., the one that returns a path with the minimum number of collisions with other agents given their current configuration and among these paths, the one with the smallest length, and (b) a greedy strategy, where we have considered two versions:

- Always return the shortest path ignoring other agents and
- Return the action selected by the "k-best" strategy.

The idea is that the probabilistic selection strategy will learn during the execution of a path whether it is better to play the "minimum conflict" strategy or the greedy alternative, given the cost that it experiences for the outcomes of these strategies over time.

The learning algorithm used is the Polynomial Weights method, which is following the principle of regret minimization [Littlestone and Warmuth1994, Nisan et al.2007]. It begins by assigning uniform weights on the two strategies: $w^{min\_conflict} = w^{greedy} = 1$. Then, when the agent must choose an action, one of the strategies is chosen at random proportionally to their weights, i.e.,

$$Pr(\text{"minimum-conflict"}) = \frac{w^{min\_conflict}}{w^{min\_conflict} + w^{greedy}}$$

During each planning cycle, the method updates these weights by calculating a loss value for each one of them: $l^{min\_conflict}, l^{greedy}$, in hindsight, i.e., assuming that all the other agents would have acted the same way, the method computes a value that corresponds to the regret of choosing that value. Given the other agents' motion, one of the two pure strategies would have performed better. This action causes low regret and its weight is not reduced, while the worse performing strategy incurs regret, and thus receives a lowered weight. The implementation of the Polynomial Weights algorithm in the context of this challenge computes loss as follows:

$$l_i = \frac{C_i - min_i(C_i)}{max_i(C_i) - min_i(C_i)}$$

Where again the term C_i corresponds to the weighted cost computed according to Eq. 2. The weights are then updated according to the following rule and the computed loss value:

$$w_i = w_i \cdot (1 - \eta \cdot l_i)$$

This means that the action with the highest weighted cost in hindsight gets its weight reduced by η, while the other action is not penalized. A value of $\eta = 0.2$ was used for the simulations presented here.

The Polynomial Weights method has several advantages. First, it does not require knowledge of the other agent's utilities and requires no information to be passed from the other agents. Furthermore, as the weights are learned, the expected utility is guaranteed to be within a bound of the best pure strategy [Littlestone and Warmuth1994, Nisan et al.2007]. Lastly, it allows a high degree of adaptability to changing conditions, as large regret costs will be quickly accumulated for choosing a sub-optimal strategy.

Simulations

Each of the strategies presented in the previous sections,Greedy (`Greedy`), k-best (`KBest`), minimum-conflict (`Min Conf`), deterministic (`Determ`) (i.e., combination of "minimum-conflict" with "k-best"), Polynomial-Weights Greedy (`PWGreedy`), and Polynomial-Weights Best (`PWBest`), was evaluated experimentally in simulation.

The experiments were run in a computing cluster, where each agent was allowed access to a single computing core on an Intel Xeon E5-4650 2.70GHz, and given 1 GB of memory. This was done to simulate the fact that each agent represented a separate robot, so there was no competition between agents for computing resources. Each experiment had a homogeneous setup of agents, i.e. every robot ran the same strategy, within a variety of scenes, such as a grid and a random obstacle environment as shown in Fig. 6.

The following metrics were used to evaluate each strategy's performance:

- average completion time in seconds for all agents,
- average length of the solution path for all agents

Evaluating the average experimental solution time provides a good measure of the performance of the method, as it directly indicates how much progress agents are making towards their goals. The purpose of examining the average path length is to have some measurement of how much "effort" an agent must spend to achieve its desired solution time.

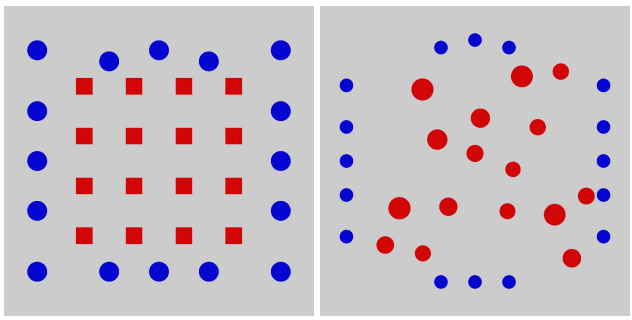


Figure 6: The environments used to evaluate the proposed methods. The blue disks are the agent's initial positions, when they are not randomized.

Corridor Experiments: Evaluating Validity

The experiments begin with a simple corridor setup, with only two agents attempting to reach opposite sides of the corridor. The purpose of such a simple setup is to find whether the proposed methods, including the `Greedy` approach, are able to solve simple congestion problems. The

results are averaged over 5 runs, with the average path lengths and solution times shown in Figure 7.

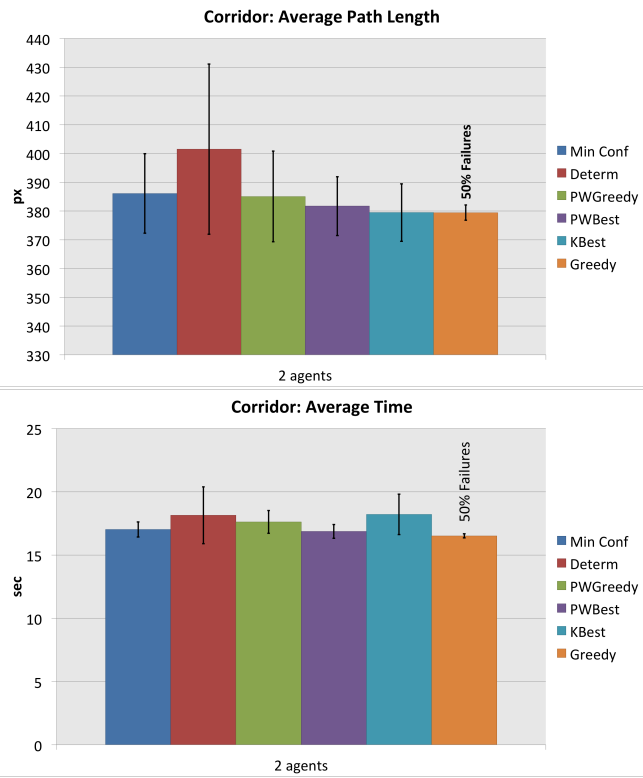


Figure 7: Results for testing the validity of the approaches in the corridor environment as shown in Figure 6.

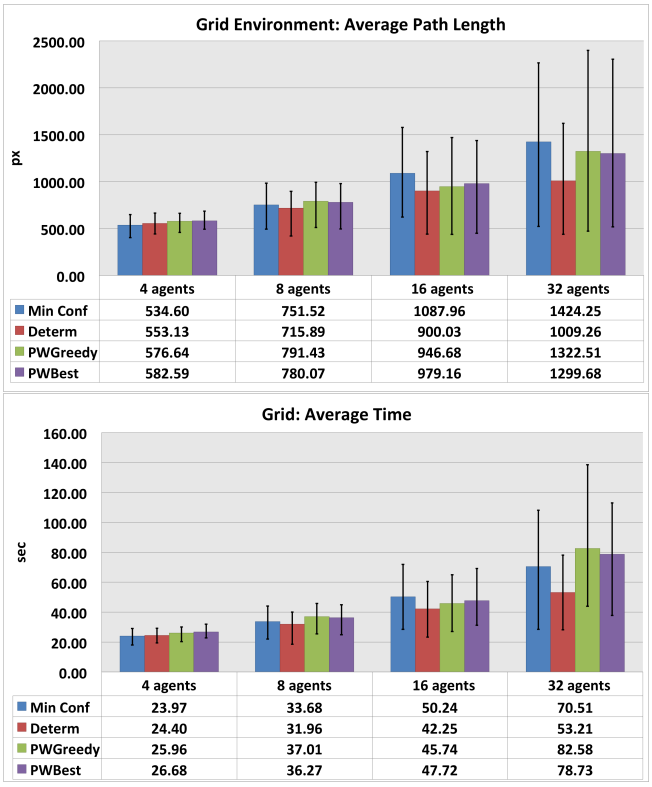
Although `Greedy` always had the lowest averages, it failed to solve even a simple deconfliction problem such as this 50% of the time. This is again due to the fact that no other paths are considered by the agent. All of the other strategies were able to solve the corridor problem without a single failure.

Evaluating Performance

The performance of the four methods is evaluated using two environments, the grid environment and a forest-like environment as shown in Fig. 6. Since the `Greedy` strategy failed to consistently solve the corridor problem, it is omitted from the rest of the experiments. An important observation of the `KBest` strategy is for small values of k, and for large numbers of homotopy classes, it is possible for the strategy to become deadlocked/livelocked. Such was the case in the grid and forest environments, so accordingly the `KBest` strategy is no longer considered in further experiments.

Agents are given a pseudo-random start location, and a fixed goal location, with the intention of having agents swap locations with one another, which promotes conflicts and congestion in the environment. The results are averaged over 5 runs and presented in Fig. 8 (for the grid environment) and Fig. 9 (forest).

The deterministic approach, `Determ`, which considers



	4 agents	8 agents	16 agents	32 agents
Min Conf	534.60	751.52	1087.96	1424.25
Determ	553.13	715.89	900.03	1009.26
PWGreedy	576.64	791.43	946.68	1322.51
PWBest	582.59	780.07	979.16	1299.68

	4 agents	8 agents	16 agents	32 agents
Min Conf	23.97	33.68	50.24	70.51
Determ	24.40	31.96	42.25	53.21
PWGreedy	25.96	37.01	45.74	82.58
PWBest	26.68	36.27	47.72	78.73

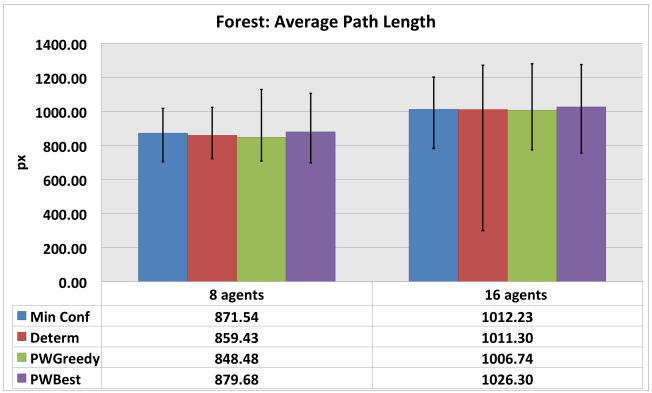
Figure 8: Results for randomly selected starting positions in the grid environment. Data is shown for increasing number of agents in the same environment, where error bars signify the variance over all simulations.

both the "minimum-conflict" and the "k-best" strategies, always selects the action that minimizes the proposed interaction cost. In the Grid environment, `Determ` outperformed the other approaches. In the random obstacle forest scene, however, the methods seem to be competitive. The explanation for this is that the random placement of obstacles, combined with a larger workspace, does not cause a constrained enough environment, hence there are not frequent conflicts between agents. This allows agents to consider a larger set of possible actions that are conflict-free, so each of the strategies presented can provide an equivalent solution quality.

Grid Experiments: Evaluating Scalability

In these experiments the start location of the agents were set to be symmetrical, so as to promote conflicts and congestion quickly. The results are averaged over 15 different runs and are shown in Figure 10. The purpose of this set of experiments was to evaluate the scalability of the adaptive-strategies, `PWGreedy` and `PWBest`, as both of these approaches utilize the other deconfliction methods, and are consequently the most computationally complex.

The results show that for increasingly larger number of agents, the average solution time and the average path lengths for the methods scales sublinearly.



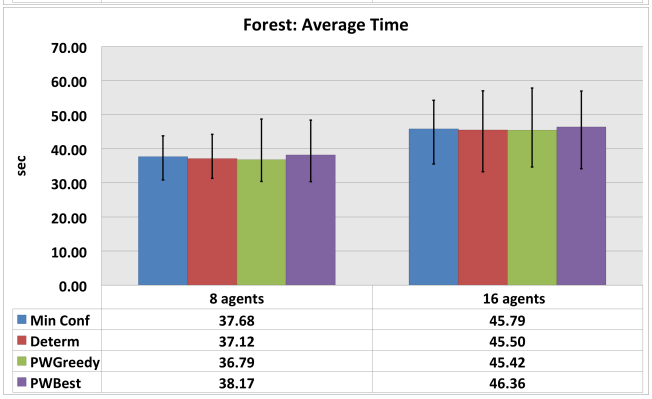
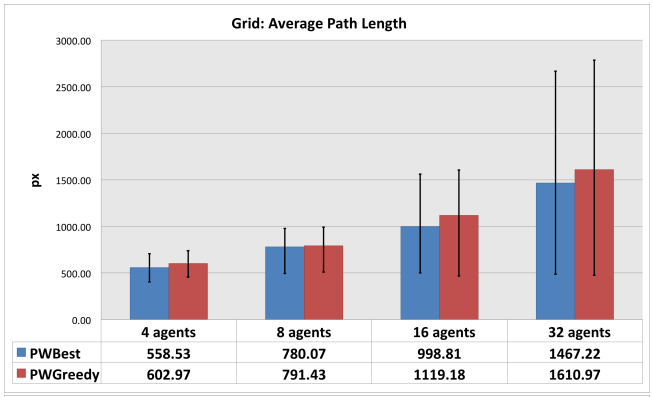


Figure 9: Results for randomly selected starting positions in the random obstacle, forest environment.



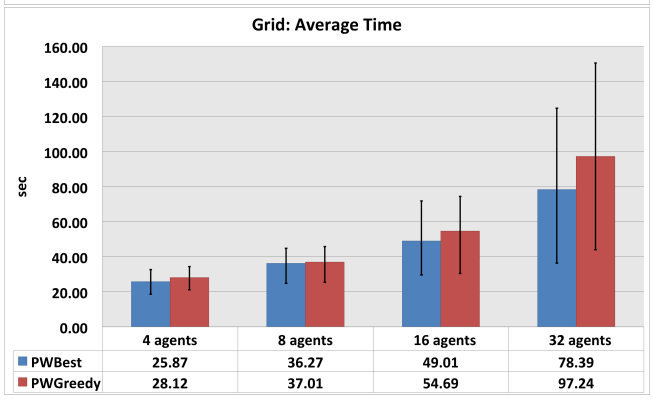


Figure 10: The average path length and average time to finish for simulations using the polynomial weight with greedy (`PWGreedy`) and with k-best selection (`PWBest`).

Heterogeneous Setups

A set of experiments was conducted among heterogeneous agents in the grid environment, where 7 agents were assigned the "minimum-conflict" strategy and 1 agent was assigned the PWBest strategy. The idea here was to examine the probabilistic learning algorithm, PWBest, and see if it was able to adapt its weights according to the strategies the other agents were playing. Interestingly, over a course of 5 separate runs, PWBest selected the "k-best" strategy 65% of the time on average. Since the 'minimum-conflict" agents were actively attempting to avoid interaction with other agents, it makes sense that the PWBest agent is able to be more "greedy" in its selection of paths.

Carrying on with this line of thought, another set of experiments was run where 4 agents were given the pure "greedy" strategy, and the other 4 agents ran PWBest. In this case, the PWBest agents adapted and chose to select the "k-best" strategy only 41% of the time. Since the 4 purely greedy agents caused a deadlock in the center of the environment, the PWBest agents had to adapt and select the safer "minimum-conflict" strategy more often. Together these results seem to show promise for the adaptability of the learning strategy, as well as motivating its use over the "deterministic" strategy.

A video of the experiments can be found at:

`http://www.cs.rutgers.edu/~kb572/videos/icaps_PlanRobWorkshop_2014.mp4`

Discussion

The proposed framework brings together path planning primitives, such as search-based methods for computing paths in different homotopic classes [Bhattacharya, Likhachev, and Kumar2011] and sampling-based motion planners for computing roadmaps [Karaman and Frazzoli2011], reactive obstacle avoidance methods [van den Berg, Lin, and Manocha2008, Snape et al.2011] as well as game theoretic and learning tools [Nisan et al.2007] to provide an algorithmic framework capable of computing acceptable solutions to motion coordination challenges in a decentralized, communication-less way.

Some interesting directions for this work include: removing the "garage" assumption from the framework - this would require agents to continue reasoning about their observed states, potentially adapting a "passive" mode to more easily allow other agents through their goal positions; extensively evaluate the adaptive methods in a larger set of heterogeneous setups, as well as imposing a stricter sensing range on the agents; and analyzing the conditions under which the current framework is able to guarantee that the robots are free of deadlocks and livelocks, using tools that have been developed towards this direction [Knepper and Rus2013]. Currently, the work is in the process of being evaluated on real systems in a construction challenge, with the eventual hope of running experiments including humans.

References

Bekris, K. E.; Grady, D. K.; Moll, M.; and Kavraki, L. E. 2012. Safe Distributed Motion Coordination For Second-Order Systems With Different Planning Cycles. *International Journal of Robotics Research (IJRR)* 31(2).

Bennewitz, M.; Burgard, W.; Cielniak, G.; and Thrun, S. 2005. Learning Motion Patters of People for Compliant Robot Motion. *International Journal of Robotics Research (IJRR)* 24(1):31–48.

Bhattacharya, S.; Kumar, V.; and Likhachev, M. 2010. Search-based Path Planning with Homotopy Class Constraints. In *Third Annual Symposium on Combinatorial Search*.

Bhattacharya, S.; Likhachev, M.; and Kumar, V. 2011. Identification and Representation of Homotopy Classes of Trajectories for Search-based Path Planning in 3D. In *Proc. of Robotics: Science and Systems*.

Excelente-Toledo, C. B., and Jennings, N. R. 2004. The dynamic selection of coordination mechanisms. *Autonomous Agents and Multi-Agent Systems* 9(1-2):55–85.

Filiot, E.; Le Gall, T.; and Raskin, J.-F. 2010. Iterated regret minimization in game graphs. In *Mathematical Foundations of Computer Science 2010*. Springer. 342–354.

Fiorini, P., and Shiller, Z. 1998. Motion planning in dynamic environments using velocity obstacles. *Int. Journal of Robotics Research* 17(7).

Fraichard, T., and Delsart, V. 2009. Navigating Dynamic Environments with Trajectory Deformation. *Journal of Computing and Information Technology* 17(1).

Green, C., and Kelly, A. 2007. Toward Optimal Sampling In the Space of Paths. In *International Symposium on Robotics Research (ISRR)*.

Hatcher, A. 2002. *Algebraic Topology*. Cambridge University Press.

Hauser, K. 2011. Adaptive time stepping in real-time motion planning. In *Algorithmic Foundations of Robotics IX*. Springer. 139–155.

Hauser, K. 2013. Minimum Constraint Displacement Motion Planning. In *Proc. of Robotics: Science and Systems*.

Henry, P.; Vollmer, C.; Ferris, B.; and Fox, D. 2010. Learning to Navigate Through Crowded Environments. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*.

Jaillet, L., and Siméon, T. 2006. Path Deformation Roadmaps. In *Workhop on the Algorithmic Foundations of Robotics (WAFR)*.

Kaminka, G. A.; Erusalimchik, D.; and Kraus, S. 2010. Adaptive multi-robot coordination: A game-theoretic perspective. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 328–334. IEEE.

Karaman, S., and Frazzoli, E. 2011. Sampling-based Algorithms for Optimal Motion Planning. In *IJRR*.

Karamouzas, I.; Geraerts, R.; and Overmars, M. 2009. Indicative Routes for Path Planning and Crowd Simulation. In *The 4^{th} Intern. Conference on the Foundations of Digital Games (FDG)*, number 113-120.

Knepper, R. A., and Mason, M. T. 2009. Path Diversity is Only Part of the Problem. In *Proc. of the IEEE Intern. Conf. on Robotics and Automation (ICRA)*.

Knepper, R. A., and Rus, D. 2012. Pedestrian-Inspired Sampling-based Multi-Robot Collision Avoidance. In *Proc. of the International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 94–100. Paris, France: IEEE.

Knepper, R. A., and Rus, D. 2013. On the Completeness of Ensembles of Motion Planners for Decentralized Planning. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*.

Large, F.; Vasquez, D.; Fraichard, T.; and Laugier, C. 2004. Avoiding Cars and Pedestrians Using Velocity Obstacles and Motion Prediction. In *IEEE Intelligent Vehicles Symposium*.

Littlestone, N., and Warmuth, M. K. 1994. The Weighted Majority Algorithm. *Information and Computation* 108:212–261.

Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. *Algorithmic game theory*. Cambridge University Press.

Petti, S., and Fraichard, T. 2005. Partial Motion Planning Framework for Reactive Planning within Dynamic Environments. In *ICINCO*, 199–204.

Qutub, S.; Alami, R.; and Ingrand, F. 1997. How to Solve Deadlock Situations within the Plan-Merging paradigm for Multi-Robot Cooperation. In *Proc. of the Inter. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, 1610–1615.

Shi, D.; Collins, E. G.; Donate, A.; Liu, X.; Goldiez, B.; and Dunlap, D. 2008. Human-aware Robot Motion Planning with Velocity Constraints. In *IEEE International Symposium on Collaborative Technologies and Systems*, 490–497.

Sisbot, E. A.; Marin-Urias, L. F.; Alami, R.; and Siméon, T. 2007. A Human-aware Mobile Robot Motion Planner. *IEEE Transactions on Robotics* 23(5):874–883.

Snape, J.; van Den Berg, J.; Guy, S.; and Manocha, D. 2011. The Hybrid Reciprocal Velocity Obstacle. *IEEE Transactions on Robotics* 27(4):696–706.

Stone, P.; Kaminka, G. A.; and Rosenschein, J. S. 2010. Leading a best-response teammate in an ad hoc team. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*. Springer. 132–146.

Thompson, S.; Horiuchi, T.; and Kagami, S. 2009. A Probabilistic model of Human Motion and Navigation Intent for Mobile Robot Path Planning. In *Proc. of the 4^{th} International Conference on Autonomous Robots and Agents*.

Trautman, P., and Krause, A. 2010. Unfreezing the Robot: Navigation in Dense, Interacting Crowds. In *Proc. of the IEEE Intern. Conf. on Intelligent Robots and Systems (IROS)*.

van Den Berg, J.; Patil, S.; Sewall, J.; Manocha, D.; and Lin, M. 2008. Interactive Navigation of Individual Agents in Crowded Environments. In *Proc. of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*.

van den Berg, J.; Snape, J.; Guy, S.; and Manocha, D. 2011. Reciprocal Collision Avoidance with Acceleration-Velocity Obstacles. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*.

van den Berg, J.; Lin, M.; and Manocha, D. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*.

Wunder, M.; Kaisers, M.; Yaros, J. R.; and Littman, M. 2011. Using iterated reasoning to predict opponent strategies. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 593–600. International Foundation for Autonomous Agents and Multiagent Systems.

Ziebart, B. D.; Ratliff, N.; Gallagher, G.; Mertz, C.; Peterson, K.; Bagnell, J. A.; Hebert, M.; Dey, A.; and Srinivasa, S. 2009. Planning-based Prediction for Pedestrians. In *Proc. of the IEEE Intern. Conf. on Intelligent Robots and Systems (IROS)*.