Decentralized Multi-agent Path Selection Using Minimal Information

Andrew Kimmel and Kostas Bekris

Abstract This work studies conflict avoidance between moving, non-communicating agents with minimum sensing information. While safety can be provided by reactive obstacle avoidance methods for holonomic systems, deadlock avoidance requires reasoning over different homotopic paths in cluttered scenes. A method to compute the "interaction cost" of a path is proposed, which considers only the neighboring agents' observed positions. Minimizing the interaction cost in a prototypical challenge with two agents moving through two corridors from opposing sides guarantees the selection of non-conflicting paths. More complex scenes, however, are more challenging. This leads to a study of alternatives for decentralized path selection. Simulations indicate that following a "minimum-conflict" path given the other agents' observed positions provides deadlock avoidance. A scheme that selects between the minimum-conflict path and a set of shortest paths given their interaction cost improves path quality while still achieving deadlock avoidance. Finally, learning to select between the minimum-conflict and one of the shortest paths allows agents to be adaptive to the behavior of their neighbors and can be achieved using regret minimization.

Keywords Multi-agent · Decentralized · Coordination · Path planning

1 Introduction

Advances in robotic technology allow applications where multiple robots operate in the same cluttered environment, potentially also in the presence of people or animals. Explicit communication with the other agents, especially when humans are involved, may not be feasible or desirable. Similarly, it may be difficult to model or predict the actions of the other agents. This motivates decentralized methods that allow a robot

A. Kimmel · K. Bekris (✉)
Rutgers University, Piscataway, NJ, USA
e-mail: andrew.kimmel@cs.rutgers.edu

K. Bekris
e-mail: kostas.bekris@cs.rutgers.edu

N.-Y. Chong and Y.-J. Cho (eds.), *Distributed Autonomous Robotic Systems*,
Springer Tracts in Advanced Robotics 112, DOI 10.1007/978-4-431-55879-8_24

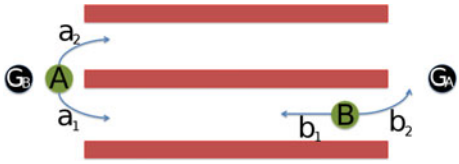

Fig. 1 If both agents A and B insist on following the same corridor, a reactive collision avoidance method may not allow them to make progress to their goals, G_A and G_B

to reach its goal safely given minimal information and without strong assumptions about the intentions of its moving neighbors. Such solutions should avoid deadlocks and minimize executed path length or task completion time.

Challenges, Foundations and Objectives: Avoiding collisions with unexpected obstacles or mobile agents, can be effectively addressed by reactive collision avoidance methods, such as those based on the popular Velocity Obstacle framework [5, 31] or trajectory deformation methods [6, 15]. These methods generally provide smooth, natural-looking paths, but they are primarily local techniques and do not reason about the robot's global path. Although local motion coordination can be achieved [16], if the agents select conflicting paths, reactive collision avoidance can still give rise to deadlocks and poor performance. For instance, consider the situation in Fig. 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 path, the space is narrow enough to prevent the robots from crossing.

Robots in such situations that replan [23] and change their path to a different homotopy class [3, 12] can potentially resolve conflicts. Such coordination can be achieved by assuming that the robots share information [1], or follow a form of centralized planning [24], or by respecting a set of pre-specified "social" rules [19], or performing sophisticated prediction [28, 32], agent modeling [25, 26], learning [11] or game-theoretic reasoning [13]. The information required 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. Furthermore, it is interesting to study what is achievable without any prediction, intent recognition or modeling of the moving agents.

Considered Methodology: This work employs strictly decentralized methods while utilizing minimal information. Each robot has access only to the current position and velocity of its neighbors from sensing data. The basic framework assumes that robots replan paths frequently [23] and employ reciprocal velocity obstacles [27, 30] so as to follow these paths while avoiding collisions. The velocity information is actually used only for the adopted reactive obstacle avoidance method based on Velocity Obstacles [31] and not for the proposed path planning techniques. Learning is also considered, corresponding to online learning of appropriate strategies in response

to the behavior of other agents. The considered techniques are evaluated in various simulated benchmarks.

If agents greedily select the globally shortest path to their goal, this frequently results in deadlocks as in the case of Fig. 1. To address this issue, one alternative is to consider a set of diverse paths instead of only the shortest one. The notion of path diversity has been shown to be helpful in many different challenges, but frequently corresponds to a local concept [7, 18]. One way to compute a diverse set of global paths is to consider different homotopy classes using search-based primitives [2, 3] over an underlying roadmap. This work provides a method for selecting a minimally conflicting path out of this set by defining an "interaction cost" for each path given the other agents' current positions. This process is designed so as to address the issue in the prototypical example provided of Fig. 1.

Computing a large number of diverse paths is computationally challenging in complex scenes. If, instead, only a small set of k shortest paths in distinct homotopy classes is considered deadlocks can still arise. An alternative is to compute the "minimum-conflict" homotopy class, which minimizes interactions with other agents. This notion is related to the "minimum constraint displacement" problem, which has recently attracted attention [10]. Here, constraints on the minimum-conflict homotopy class correspond to the observed locations of other agents and can be discovered in a computationally efficient manner using an underlying roadmap. The experimental evaluation shows that when the moving agents follow their "minimum-conflict" paths, they can avoid deadlocks even in complex scenes.

While minimum conflict paths achieve deadlock avoidance experimentally, they can be inefficient, forcing agents to follow long paths to reach their goals. Considering both the minimum-conflict path and a set of k shortest paths in distinct homotopy classes typically results in improved performance in terms of path length. This work considers two approaches for choosing between (a) the conservative alternative of following the "minimum-conflict" path and (b) the greedy choice corresponding to one of the k shortest paths. The first approach is a deterministic strategy that selects the path among the $k + 1$ choices, which minimizes the "interaction cost".

The second approach is based on the notion of regret minimization and corresponds to the Polynomial Weights `PW` algorithm [21, 22]. Regret minimization is a favorable way to reason among unpredictable agents, without knowing their goals, utilities, intents, or beliefs, as in the setup of this work. The application of the `PW` algorithm here accumulates regret for the "minimum-conflict" strategy and the "greedy" choice strategy by observing the choices of the other agents in the same workspace and assigning a loss to each strategy in hindsight. A probability is then assigned for selecting each strategy based on the accumulated losses.

Both the deterministic and the learning-based solution result in deadlock avoidance in the simulated benchmarks considered in this work. The learning approach can also adapt to different behaviors by neighboring agents.

Contribution and Overview of Results: The key observations from this work can be summarized as the following points:

(a) Using "interaction cost" to select a path among multiple choices assists in minimizing the occurrence of deadlocks. In a prototypical benchmark with few homotopy classes and agents, it can guarantee deadlock avoidance.
(b) Computing minimum-conflict paths is experimentally shown to be critical for avoiding deadlocks even in more complex scenes with many homotopy classes and is interesting to further analyze the properties of this strategy.
(c) Considering multiple paths in distinct homotopy classes together with the minimum-conflict path results in improved path quality and execution time.
(d) Regret minimization is a computationally efficient way for robots to react to the behavior of other agents in the scene without explicit communication.

2 Problem Setup

Path deconfliction problems can be defined in general configuration spaces but this discussion will focus on holonomic planar 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 $\{a_1, \ldots, a_m\}$ that move with bounded velocity $v \in [0, v_{max}]$ in the same workspace $\mathbb{W}$. The configuration space of an agent is $\mathbb{Q} = \mathbb{R}^2$, where $\mathbb{Q}_{free}$ represents the obstacle free subset given static obstacles. Given a configuration $q_i \in \mathbb{Q}$, the expression $a(q_i)$ corresponds to the collision volume of agent a_i in $\mathbb{W}$.

A path $\pi_i = \{q_i | q_i : [0, 1] \rightarrow Q_{free}\}$ for agent a_i corresponds to a continuous curve in Q_{free}. Given a time scaling function $\sigma_i : \mathbb{R}^{\geq 0} \rightarrow [0, 1]$ it is also possible to define the sequence of configuration $\tau_i = \pi_i \circ \sigma_i$ that the agent visits at each point in time.

The problem formulation assumes that each agent a_i wants to reach a desired goal $q_i^{\mathbb{G}} \in \mathbb{Q}$ without conflicts. The objective then is for the agents to select the sequence of configurations $\{\tau_1, \ldots, \tau_m\}$ they will follow in a decentralized manner, such that in finite time T: $\forall\, i \in \{1, \ldots, m\} : \tau_i[T] = q_i^{\mathbb{G}}$. Collisions between agents must be avoided, unless one of the agents has reached its goal, i.e.,

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

The above description implies a version of the so called "garage" assumption. When agents reach their goal, they are removed from the workspace and are not considered for collisions when other agents pass through that goal.

Agents are never aware of the goal of any other agent or the path selected by another agent. At any point in time an agent can only observe the positions of other agents as long as their configurations are within a certain sensing radius.

Agents are assumed to have access to a collision avoidance method (e.g., Reciprocal Velocity Obstacles [30] are used in the accompanying experiments), which is used to follow their selected path while still avoiding collisions with other agents. This means that the planned path may not be executed perfectly due to the influence of neighboring agents and the use of the obstacle avoidance method.

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 setup involving unpredictable neighboring agents motivates a replanning framework for recomputing paths given the latest observed configurations of agents. This replanning approach forms the basis of the overall methodology that is described in the following section.

3 Methods

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

Replanning Framework: During execution of an action, a robot can deviate from its corresponding planned path due to the reactive collision avoidance executing a safety control. Naïvely following the original planned path is therefore not sufficient, as the robot will most likely not be able to reach its goal as intended. A replanning framework [9, 23] as illustrated in Fig. 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 [29]. The sampling-based planner used in this work is PRM* [14]. The path computed for time $t-1$ to t will not be executed perfectly, as shown in Fig. 2; however, the framework updates the predicted state of the robot accordingly. By using such a replanning

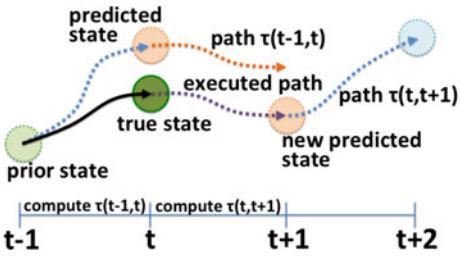

Fig. 2 The path computed $[t-1, t]$ is executed during time $[t, t+1]$. The state at time t can deviate from the predicted initial state for the computed plan, so planning for cycle $[t+1, t+2]$ must start from an updated predicted state

framework, where the agent updates its new predicted state given the perceived differences between its true state and the previous predicted state, the agent becomes robust to perturbations in the solution path caused by the use of reactive methods. The most straightforward path selection process corresponds to selecting 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, when the shortest paths of two agents conflict.

K-Best Paths from Different Homotopy 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 and select one that reduces possible interactions with other agents. In an environment with many obstacles and narrow corridors which cause conflicts between agents that cannot be resolved by reactive obstacle avoidance, it makes sense to consider paths that belong to different homotopy classes [3]. Homology classes [4] can also be used to compute a diverse set of paths, however this work does not utilize homologies since they do not differentiate between paths that symmetrically loop around obstacles. When considering 2D problems, paths are in different homotopy classes when the area between them contains an obstacle. A complete definition for homotopy can be found in the related literature [8].

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 Fig. 3.

Minimizing Interaction Cost The question that arises is how should agents differentiate among the available paths in order to select motions that will allow them to make progress to their goals. To describe the proposed process, consider the situation depicted in Fig. 1, where agent A can follow action a_1 to move through the lower corridor and action a_2 to move through the upper corridor towards its goal G_A. These actions correspond to two solutions returned from the homotopy class computation described in the previous section, regardless of the current configuration of the robot q_A. Similarly, agent B has choices b_1 and b_2.

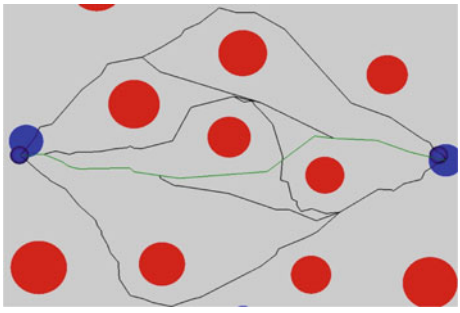

Fig. 3 Selecting the path with the lowest interaction cost from k different homotopy classes is the "k-best" strategy

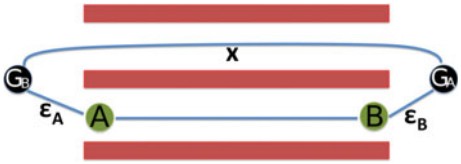

Fig. 4 Backtracking increases path length by ε_A and ε_B for robots A and B respectively, while remaining in the current corridor reduces path length by ε_A and ε_B respectively

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. The most conflicted situation occurs when both robots are already following 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 Fig. 4. If the corridors are too narrow, then one of 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 (the lower corridor). Then the lengths of the shortest paths for the robots to reach their goals via the corresponding homotopic paths can be computed as follows:

$$P_1^A = x - \varepsilon_A,\ P_2^A = x + \varepsilon_A, \qquad P_1^B = x - \varepsilon_B,\ P_2^B = x + \varepsilon_B,$$

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:

$$I_1^A = \frac{\varepsilon_B}{x - \varepsilon_A},\ I_2^A = 0, \qquad I_1^B = \frac{\varepsilon_A}{x - \varepsilon_B},\ I_2^B = 0.$$

Then the proposed cost function for actions is $C_i^X = P_i^X(1 + 2 \cdot I_i^X)$, which translates to the following costs in the above scenario:

$$C_1^A = x - \varepsilon_A + 2 \cdot \varepsilon_B,\ C_2^A = x + \varepsilon_A, \qquad C_1^B = x - \varepsilon_B + 2 \cdot \varepsilon_A,\ C_2^B = x + \varepsilon_B.$$

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

$$C_1^A = x - \varepsilon_A + 2 \cdot \varepsilon_B < x + \varepsilon_A = C_2^A \Rightarrow \quad A \text{ selects corridor 1 iff: } \varepsilon_B < \varepsilon_A \quad (2)$$

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

$$C_1^B = x - \varepsilon_B + 2 \cdot \varepsilon_A < x + \varepsilon_B = C_2^A \Rightarrow \quad B \text{ selects corridor 1 iff: } \varepsilon_A < \varepsilon_B \quad (3)$$

From Eqs. 2 and 3 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 homotopy class.

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

In practice, using the value $\alpha = 2$ as was done 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 used. First, compute the k-shortest paths that belong to k different homotopy classes. Then, for each one of these paths, compute their costs according to the definition of C_i^X, where the interaction cost is computed according to Eq. 1. 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 a robot to consider multiple alternative choices as well as interactions with neighbors. In this manner, it provides a resolution to the basic "corridor" challenge under the assumption that the goals of the two agents are symmetric.

Minimum Conflict (`MC`) **Path**: The "k-best" strategy was able to avoid deadlocks as long as the total number of simple homotopy classes did not significantly exceed k, where simple homotopy classes correspond to those that do not include loops. When this property is true, then the strategy described in the previous section results in the selection of paths which allow the team to make progress overall. Even in relatively simplistic scenes, however, the number of homotopy classes can quickly become large. This introduces a computational challenge, since the $k + 1$th homotopy class corresponds to an increasingly longer path, which translates to a longer search time on the underlying roadmap. To keep the proposed method computationally

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 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 a_i marks edges in the roadmap that intersect q_j. Edges that are marked then have their weights inflated by a large amount, 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 in the accompanying simulations.

Even so, the resulting paths may not be as desirable when all the agents follow their minimum conflict action. As shown in Fig. 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: first, a homotopy class computation algorithm is used to extract the k-shortest paths that belong to k different homotopy classes. Next, the "minimum-conflict" action is computed. For each one of the above $k + 1$ paths, their costs are computed according to the definition of C_i^X, where the interaction cost is from Eq. 1. Finally, the action with the minimum cost is returned.

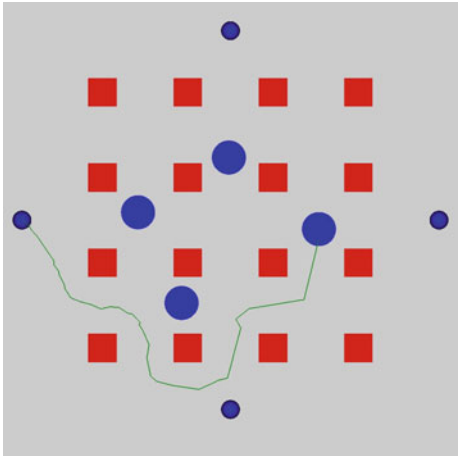

Fig. 5 A "minimum-conflict" path computed for the right-most agent for a goal to the left. The shortest path without conflicts is returned. For a large distribution of agents, the "minimum-conflict" path will typically intersect some agents

This "deterministic" approach for combining the agent's greedy choices, i.e., k-shortest paths, and the safe/conservative choice, i.e., the minimum conflict path, takes advantage of the process for evaluating interaction costs. It allows agents to sometimes 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 early on.

A Probabilistic Selection Strategy To allow some adaptability to the choices of other agents, this work considers an online learning method to select the appropriate strategy out of the following: (a) the "minimum conflict" (the shortest path with the least amount of agent interaction), and (b) a greedy strategy, where this work considers two possible alternatives for the greedy strategy—returning the shortest path ignoring other agents or, returning the action selected by the "k-best" strategy. The objective of this approach is to learn during the execution of a path whether it is better to play the conservative/"minimum conflict"strategy or the greedy alternative, given the cost that it experiences for the outcomes of these strategies over time.

The learning approach corresponds to the Polynomial Weights method, which applies regret minimization [21, 22]. 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 has low regret and its weight is not reduced, while the worse performing strategy incurs regret, and thus receives a lowered weight. The implementation of Polynomial Weights in the context of this challenge implies a loss computation as follows:

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

The term C_i again corresponds to the weighted interaction cost. 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 [21, 22]. 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.

4 Simulations

Each of the strategies presented in the previous sections was evaluated experimentally using appropriate simulation software [17]: shortest path (`Greedy`), K-best (`KBest`), Minimum-conflict (`Min Conf`), Deterministic combination of `Min Conf` and `KBest` (`Determ`), and Polynomial-Weights with (`Greedy`) (`PWGreedy`) and with `KBest` (`PWBest`).

The metrics used were average completion time in seconds for all agents and 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.

Each agent had a physical radius of 18 cm and a sensing radius of 200 cm. A visualization of the environments used in the experiments is shown in Fig. 6. In the Grid environment, the block obstacles measured 30 $\times$ 30 cm, and were placed so as to create corridors with width 40 cm. In the Random Obstacle environment, the obstacles were cylinders with variable radius from 10 to 40 cm.

A centralized optimal path planner is not compared against in the performance section, as it would remove the "minimum information" requirement while only providing a comparison point for small numbers of agents. Scenes without enough agents in them would not introduce many conflicts, so the proposed approaches do not provide benefit over using existing reactive obstacle avoidance.

Evaluating Validity: The current work can only provide guarantees on conflict avoidance in scenarios such as the one presented in Sect. 3: Minimizing Interaction Cost. Accordingly, the experiments begin with a simple corridor setup, with two agents having symmetrical goals and attempting to reach opposite sides of the corridor. The purpose of such a simplistic setup is to find whether the proposed methods,

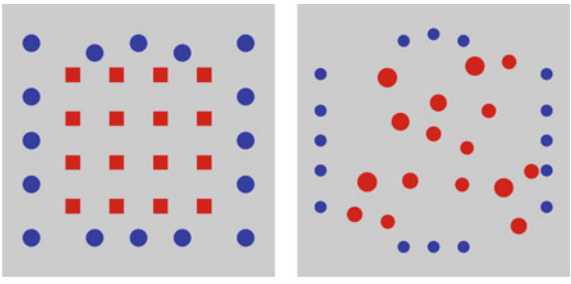

Fig. 6 The environments used to evaluate the proposed methods. The *blue* disks are the agent's initial positions, when they are not randomized

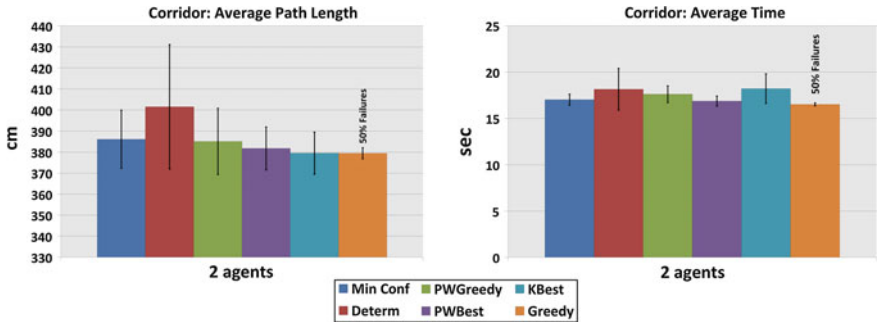

Fig. 7 Validation results in the corridor scenario of Fig. 1

including the `Greedy` approach, are able to solve simple congestion problems as well as testing the validity of the framework in simulation. The results are averaged over 10 runs, with the average path lengths and solution times shown in Fig. 7.

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 random obstacle 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 random 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 for the grid environment are averaged over 10 runs and presented in Fig. 8.

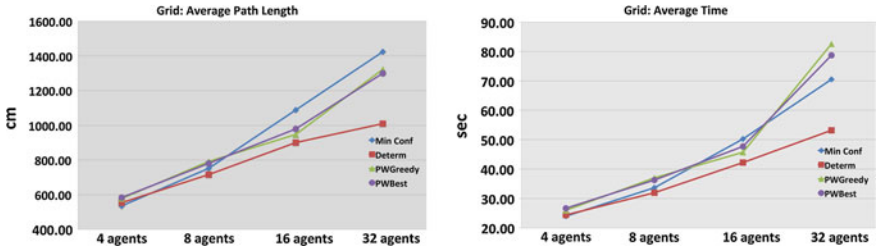

Fig. 8 Results for randomly selected starting positions in the grid environment

The deterministic approach, `Determ`, which considers 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. This makes sense since these experiments were homogeneous (all agents used the same approach), so the adaptive strategies could not take advantage of any differences in strategies (such as learning to follow a greedy strategy against agents that play conservatively and follow a minimum-conflict strategy).

A set of experiments was conducted in the random obstacle scene, however, the results showed that all the approaches performed at roughly the same level. The explanation for this is that the random placement of obstacles, combined with a larger workspace, did not cause a constrained enough environment, hence there were not frequent conflicts between agents. This allowed agents to consider a larger set of possible actions that are conflict-free, so each of the strategies presented were able to provide an equivalent quality solution.

Evaluating Performance Without Reciprocal Velocity Obstacles A requirement of the Reciprocal Velocity Obstacle (RVO) method is that agents are able to sense the velocities of neighboring agents. In order to study the viability of the proposed methods without relying on sensing velocities, a "lite" version of the methods are evaluated, where the RVO method is replaced with a simplistic, position-based approach (agents stop moving in a particular direction if this direction brings it too close to another agent).

To get a better idea of how well the proposed methods perform, and to serve as a comparison point, a "straw-man" type algorithm was implemented: when an agent finds its currently selected path results in a conflict, it randomly selects a different, potentially viable path. This algorithm is presented in the results as the "KRand" algorithm. The results are presented in Fig. 9.

All of the methods, except for KRand, were able to solve the problems a 100 % of the time. The comparison method, KRand, provided competitive solution path lengths, at the expense of much longer solution times, as well as a 40 % failure rate in the 32 agent scenario. Although `Min Conf` always solved the problems, both `Determ` and `PWBest` outperformed it in solution time and solution length.

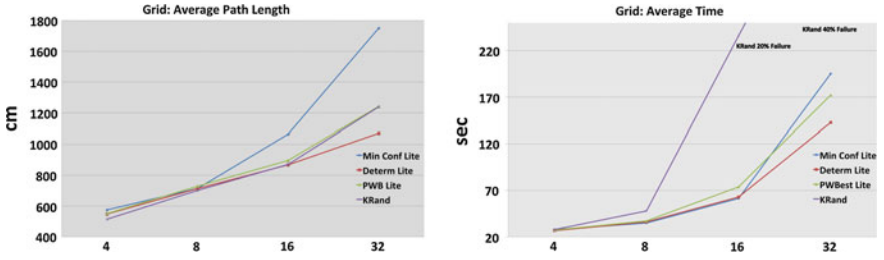

Fig. 9 Results for randomized grid using the "lite" versions of the algorithms. The Reciprocal Velocity Obstacle collision avoidance method is replaced with a simple, position-based method

Fig. 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`)

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 Fig. 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.

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 in the general case, since their path length is equivalent in homogeneous setups. A video providing a qualitative description of the performed experiments can be found at: http://www.cs.rutgers.edu/~kb572/dars2014.mp4.

5 Discussion

The proposed framework brings together path planning primitives, such as search-based methods for computing paths in different homotopy classes [3] and sampling-based motion planners for computing roadmaps [14], reactive obstacle avoidance methods [27, 30] as well as game theoretic and learning tools [22] to provide an algorithmic framework capable of computing acceptable solutions to motion coordination challenges in a decentralized, communication-less way.

There are many interesting future directions for this line of research. This includes the evaluation of approaches on dynamical systems as well as more complex environments and 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. Additional experimentation of the adaptive, learning methods in a larger set of heterogeneous setups is interesting, as well as imposing a stricter sensing range on the agents. Furthermore, it is important to analyze the conditions under which the current framework, in particular the consideration of the "minimum-conflict" path, is able to guarantee that the robots are free of deadlocks and livelocks, using tools that have been developed towards this direction [20].

References

1. Bekris, K.E., Grady, D.K., Moll, M., Kavraki, L.E.: Safe distributed motion coordination for second-order systems with different planning cycles. IJRR **31**(2) (2012)
2. Bhattacharya, S., Kumar, V., Likhachev, M.: Search-based path planning with homotopy class constraints. In: Third Annual Symposium on Combinatorial Search (2010)
3. Bhattacharya, S., Likhachev, M., Kumar, V.: Identification and representation of homotopy classes of trajectories for search-based path planning in 3D. In: RSS (2011)
4. Bhattacharya, S., Likhachev, M., Kumar, V.: Topological constraints in search-based robot path planning. Auton. Robots **33**(3), 273–290 (2012)
5. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. IJRR 17(7) (1998)
6. Fraichard, T., Delsart, V.: Navigating dynamic environments with trajectory deformation. J. Comput. Inf. Technol. **17**(1) (2009)
7. Green, C., Kelly, A.: Toward optimal sampling in the space of paths. In: ISRR (2007)
8. Hatcher, A.: Algebraic Topology. Cambridge University Press (2002)
9. Hauser, K.: Adaptive time stepping in real-time motion planning. In: Algorithmic Foundations of Robotics IX, pp. 139–155. Springer (2011)
10. Hauser, K.: Minimum constraint displacement motion planning. In: RSS (2013)
11. Henry, P., Vollmer, C., Ferris, B., Fox, D.: Learning to navigate through crowded environments. In: ICRA. Anchorage, AK (2010)
12. Jaillet, L., Siméon, T.: Path deformation roadmaps. In: Workhop on the Algorithmic Foundations of Robotics (WAFR) (2006)
13. Kaminka, G., Erusalimchik, D., Kraus, S.: Adaptive multi-robot coordination: a game-theoretic perspective. In: ICRA (2010)
14. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. In: IJRR (2011)

15. Karamouzas, I., Geraerts, R., Overmars, M.: Indicative routes for path planning and crowd simulation. In: Foundations of Digital Games (FDG), pp. 113–120 (2009)
16. Kimmel, A., Dobson, A., Bekris, K.E.: Maintaining team coherence under the velocity obstacle framework. In: Autonomous Agents and Multiagent Systems (AAMAS) (2012)
17. Kimmel, A., Dobson, A., Littlefield, Z., Krontiris, A., Marble, J., Bekris, K.E.: Pracsys: an extensible architecture for composing motion controllers and planners. In: SIMPAR (2012)
18. Knepper, R.A., Mason, M.T.: Path diversity is only part of the problem. In: ICRA (2009)
19. Knepper, R.A., Rus, D.: Pedestrian-inspired sampling-based multi-Robot collision avoidance. In: IEEE RO-MAN, pp. 94–100. IEEE, Paris, France (2012)
20. Knepper, R.A., Rus, D.: On the completeness of ensembles of motion planners for decentralized planning. In: ICRA. Karlsruhe, Germany (2013)
21. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Inf. Comput. **108**, 212–261 (1994)
22. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press (2007)
23. Petti, S., Fraichard, T.: Partial motion planning framework for reactive planning within dynamic environments. In: ICINCO, pp. 199–204 (2005)
24. Qutub, S., Alami, R., Ingrand, F.: How to solve deadlock situations within the plan-merging paradigm for multi-Robot cooperation. IROS **3**, 1610–1615 (1997)
25. Shi, D., Collins, E.G., Donate, A., Liu, X., Goldiez, B., Dunlap, D.: Human-aware Robot motion planning with velocity constraints. In: IEEE International Symposium on Collaborative Technologies and Systems, pp. 490–497 (2008)
26. Sisbot, E.A., Marin-Urias, L.F., Alami, R., Siméon, T.: A human-aware mobile Robot motion planner. IEEE Trans. Robot. **23**(5), 874–883 (2007)
27. Snape, J., van Den Berg, J., Guy, S., Manocha, D.: The hybrid reciprocal velocity obstacle. IEEE Trans. Robot. **27**(4), 696–706 (2011)
28. Thompson, S., Horiuchi, T., Kagami, S.: A probabilistic model of human motion and navigation intent for mobile Robot path planning. In: ICARA (2009)
29. van Den Berg, J., Patil, S., Sewall, J., Manocha, D., Lin, M.: Interactive navigation of individual agents in crowded environments. In: I3D (2008)
30. van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2008)
31. van den Berg, J., Snape, J., Guy, S., Manocha, D.: Reciprocal collision avoidance with acceleration-velocity obstacles. In: ICRA (2011)
32. Ziebart, B.D., Ratliff, N., Gallagher, G., Mertz, C., Peterson, K., Bagnell, J.A., Hebert, M., Dey, A., Srinivasa, S.: Planning-based prediction for pedestrians. In: IROS (2009)