Minimizing Conflicts Between Moving Agents over a Set of Non-Homotopic Paths Through Regret Minimization

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

Abstract

This paper considers a game-theoretic framework for motion coordination challenges. The focus of this work is to minimize the number of interactions agents have when moving through an environment. In particular, agents employ a replanning framework and regret minimization over a set of actions, which correspond to different homotopic paths. By associating a cost to each trajectory, a motion coordination game arises. Regret minimization is argued as an appropriate technique, as agents do not have any knowledge of other agents' cost functions. This work outlines a methodology for minimizing the regret of actions in a computationally efficient way. Initial simulation results involving pairs of mobile agents show indications that the proposed framework can improve the choice of non-colliding paths compared to a greedy choice by the agents, without increasing any information requirements.

Introduction

A solution to a motion coordination problem should provide collision free paths for each agent and each agent should move towards its goal in an efficient way. Although centralized solutions can provide these properties, they often require knowing many specific details of every agent, such as its goal and utilities for different actions. Especially for the case that a robot interacts with a human, such information is difficult to come by since the robot's motion planner has little knowledge about the human's future actions. Thus, it becomes difficult to guarantee safety while progressing towards the robot's goal. To avoid collisions with unexpected obstacles or other self-interested agents, the agents can be augmented with reactive collision avoidance techniques, such as a reactive method like Velocity Obstacles [Fiorini and Shiller1998]. While these methods generally provide smooth and natural-looking paths, they are primarily local methods and do not reason at a global level which path an agent should follow. This paper draws inspiration from such methods, such as Reciprocal Velocity Obstacles [van den Berg et al.2011], in that the proposed method aims towards a similar notion of reciprocity between agents, except at a motion planning level.

This paper proposes game theory as an appropriate way to formulate motion coordination challenges. Consider two people walking down a corridor. It is not appropriate for these types of problems to consider a common scalar cost function, as each agent can have individual objectives, which may not be known globally. Coordination can thus be posed as a problem of finding a Pareto optimal solution, which would require a centralized approach. Instead, it is possible to consider the computation of Nash Equilibria, which can be achieved in a decentralized fashion. Even this methodology, however, would still require knowledge of the game (i.e., knowing the goals and payoffs of the other agents). In addition to this, Nash Equilibrium solutions are sometimes counter intuitive [Halpern and Pass2012], but more importantly, computing them has a high computational complexity for a large numbers of agents.

The problem can be posed as follows: a solution is needed that achieves coordination between agents, with minimal information and without centralization, and is an intuitive solution for people. Regret minimization has recently garnered much attention in game theory, as solutions computed using regret minimization have higher average utilities for certain types of games. A nice property of regret minimization is that it does not require knowledge of the other agents' payoffs. The proposed method for motion coordination challenges takes advantage of this property to select paths for each agent. Furthermore, regret minimization can be seen as a learning approach, which accumulates how much regret is associated with each action during each step of the coordination game by observing the choices of the other agents in the same workspace. Overall, the proposed framework brings together motion planning primitives, game theoretic notions and learning tools to provide an algorithmic framework capable of computing acceptable solutions to motion coordination challenges in a decentralized, communicationless way, which can be easily combined with reactive strategies to achieve efficient, collision-free behavior.

The framework is in its early stages of development and evaluation. An application to a prototypical, basic motion coordination challenge is initially presented in this report. Based on this basic challenge, this work provides a framework for dealing with more general motion coordination games. Simulations for two agents negotiating different paths are presented in this report that show some promise.

Background

There is extensive literature on motion coordination and collision avoidance in robotics, as well as on game theory in A.I. This section focuses on some of the most related works.

Reactive Obstacle Avoidance: Reactive obstacle avoidance techniques can be used for local motion coordination. They often require a minimal amount of information and provide generally intuitive, at least from a human viewpoint, solutions. Some of these methods draw inspiration from human social interaction [Shi et al.2008] [Knepper and Rus2012]. Others use the notion of reciprocity between agents [van den Berg, Lin, and Manocha2008]. However, as these are local methods, they do not reason about which homotopic paths agents should select and in the general case result in deadlock/livelock situations.

Intersection of Motion Planning and Game Theory: Problems such as air combat [Lachner, Breitner, and Pesch1995], and adversarial path planning [Vanek et al.2010] lie at the intersection of motion planning and game theory. Differential game theory studies such challenges, including pursuit-evasion but analytical techniques are difficult to apply [Isaacs1965]. There are also numerical approaches for pursuit-evasion [Ehtamo and Raivio2001] and graph-based solutions, which typically consider finite state and action spaces [Isler, Sun, and Sastry2005]. Recent work has shown it is possible to use asymptotically optimal variants of sampling-based planners to address pursuit-evasion [Karaman and Frazzoli2010]. There are methods which compute Pareto optimal strategies for two agents in roadmaps [Lavalle1995, Ghrist, O'Kane, and LaValle2005].

Applications of Game Theory in Motion Selection: There has been some work in formulating coordination problems as game theoretic problems. Coordination can be posed as an extensive-form game and reinforcement learning can then be used to create adaptive, albeit loosely coupled, agents [Kaminka, Erusalimchik, and Kraus2010]. Some approaches qualitatively measure the effectiveness of coordination between agents offline in order 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]. However, all of these approaches require knowledge of the agents' actions. 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. 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 [Nisan et al.2007, Filiot, Le Gall, and Raskin2010], it is possible to solve certain games without knowing the other agent's actions.

Problem Formulation

While motion coordination challenges arise at a variety of applications, this report focuses on navigation problems as they can provide an easy way to describe the different notions of the framework.

Consider a set $\mathbb{N}$ of planar, holonomic agents moving with a bounded velocity $\{v : v_{min} \leq v \leq v_{max}\}$. Agents are capable of instantaneously moving with a velocity vector v_a of magnitude m. The configuration space of an agent is $Q = \mathbb{R}^2$, and is partitioned into two sets, Q_{free} and Q_{obst}. Q_{free} represents the obstacle free part of the space, and Q_{obst} is the part of the space with obstacles.

Definition 1 *(Solution Trajectory):* For some agent $i \in \mathbb{N}$, a trajectory $\tau = \{q | q : [0,1] \rightarrow Q_{free}\}$ is a solution trajectory if $\tau(0) = q_i^{init}$ and $\tau(1) = q_i^{goal}$.

In order to define the motion coordination problem, the notion of homotopic trajectories must be introduced.

Definition 2 *(Homotopic Trajectories):* The trajectories τ_1 and τ_2 are homotopic, or share the same homotopy class, if for some topological space T, there exists a continuous function $f : [0,1] \times [0,1] \rightarrow T$, such that $\forall x \in [0,1]$, $f(x,0) = \tau_1(x)$, $f(x,1) = \tau_2(x)$, and $\forall s \in [0,1]$, $f(0,s) = f(0,0)$, $f(1,s) = f(1,0)$.

For two dimensional problems, trajectories are in the same homotopy class if the area between them does not intersect with any obstacles in the environment. A complete definition for homotopy can be found in [Hatcher2002]. The homotopy class of a trajectory τ is denoted as $H(\tau)$.

An agent's action set is comprised of a finite set of trajectories from different homotopic classes, and is denoted as $\mathbb{A}$. By using different homotopic paths, agents thus reason over their global reachability, as opposed to actions such as move left or move right, which limits the agent to reason locally.

Each agent $i \in \mathbb{N}$ has a geometry that can be approximated by a bounding sphere with radius r_i. Agents are able to sense the position and velocity of other agents in the environment as long as the other agents are within some sensing radius d_i^{sense}. Consider the distance $d(i,j,t)$ between agents i and j at time t. If $d(i,j,t) < r_i + r_j$, then the agents are said to be in collision at time t. Collisions with static obstacles occur when $q_i(t) \in Q_{obst}$.

Definition 3 *(Conflict Minimization):* Given agents i, j, select actions $a_i \in \mathbb{A}_i$ and $a_j \in \mathbb{A}_j$ so that $H(a_i) \neq H(a_j)$ or $\forall k \in [0,1]$, $a_i(k)$ and $a_j(k)$ does not result in a collision.

The objective of the motion coordination problem is for each agent to select an action from their individual action set that minimizes the number of obstructive interactions. Agent interaction is defined as two or more agents traversing the same homotopy class and coming within some threshold distance of one another. An interaction becomes obstructive when it prevents one or more agents from completing their action, such as through a collision. The paper now examines an example of a motion coordination game.

A Prototypical Motion Coordination Game

Consider two agents A and B in the situation illustrated in Figure 1. Aside from sensing the position of each other, neither agent has any additional knowledge of the other agent. Each agent has two possible ways of reaching its goal, either through the top corridor (a_1, b_1) or the bottom one (a_2, b_2). The action sets for agents A and B are therefore $\{a_1, a_2\}$

and $\{b_1, b_2\}$ respectively. A greedy choice that requires no communication is for each agent to select the action which will move it along the shortest path to its goal.

Depending on the width of the corridors, it may be impossible for two agents to pass each other. Thus, if both agents continue to approach each other in the corridor, then at some point at least one agent will have to backtrack out of the corridor, resulting in a much longer solution length for the backtracking agent. The desirable outcome is for agents A and B to move into separate corridors, without having to use communication. This would correspond to a Nash equilibrium of the corresponding game. Since agent B is already an ϵ-distance committed down one of the corridors, the problem is formulated so that the pareto-dominating choice among the two possible Nash equilibria is for agent A to move to its goal using the top, unoccupied corridor.

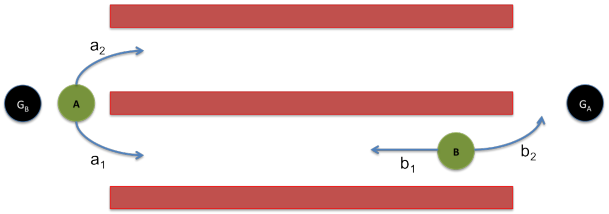

Figure 1: An example motion coordination challenge.

For each agent, define the cost of each action as the length of the corresponding path to the goal denoted as $\{C(a) : a \in \mathbb{A}\}$. Then if $a_i^* = argmin_{\forall a_k} C(a_k)$, $b_j^* = argmin_{\forall b_l} C(b_l)$ and if $i \neq j$, then the greedy choice will lead the agents along different paths. However, if $i = j$, then the greedy choice will force the agents to meet in a corridor. Alternatively, this challenge can be viewed as a game where each combination of actions results in payoffs or costs for both agents. Let I_{ij} represent the interaction cost for action a_i, b_j, where interaction could represent how much closer the two agents will get by executing their actions. Consider the following game formulated in Table 1.

	b_1	b_2
a_1	$C(a_1)+I_{11}$,$C(b_1)+I_{11}$	$C(a_1)+I_{11}$,$C(b_2)+I_{12}$
a_2	$C(a_2)+I_{21}$,$C(b_1)+I_{21}$	$C(a_2)+I_{22}$,$C(b_2)+I_{22}$

Table 1: An example motion coordination game, where each agent has two possible actions leading into two separate corridors.

The cost of the strategy profile $\{a_i, b_j\}$ is defined as: $\{C(a_i) + I_{ij}\}$ for the first agent. For the setup in the example, $I_{12} = I_{21} = 0$ as there will not be an interaction in this case (i.e., the agents chose separate corridors), while $I_{11} = I_{22} = C > 0$ since the agents will end up in the same corridor. Assume $C >> C(a_i)$ and $C >> C(b_j)$. The game can now be rewritten as shown in Table 2, resulting in Nash equilibria for the strategies $\{a_2, b_1\}$ and $\{a_1, b_2\}$. Whether a Pareto dominant strategy profile among the two exists depends on the exact $C(a_i)$ and $C(b_j)$ values.

A problem with the game theoretic reasoning above is that each agent needs to know the actions, goal, and corresponding cost functions of the other agent in order to formulate

	b_1	b_2
a_1	$C(a_1)+C$,$C(b_1)+C$	$C(a_1)$,$C(b_2)$
a_2	$C(a_2)$,$C(b_1)$	$C(a_2)+C$,$C(b_2)+C$

Table 2: An altered version of the game, where interaction costs have been substituted in.

and solve the game. This is typically not available without communication. This is why regret minimization is appealing - it does not require information about the other agent. By applying a learning based methodology for each agent, where by playing repeated games and observing the actions of the opponent along with the corresponding incurred costs of the agent's own actions, it is possible to converge to any of the Nash equilibria present in the example game.

Given that the paths (actions) a_1 and a_2 form a cycle, it is possible to rewrite their costs as follows:

$$\begin{aligned} C(a_1) &= V_A - \epsilon_A \\ C(a_2) &= V_A + \epsilon_A \end{aligned}$$

Without loss of generality, assume that $C(a_1) < C(a_2)$ for any $\epsilon_A > 0$, and that V_A is half the length of the cycle defined by the paths a_1 and a_2. Similarly, for agent B: $C(b_1) = V_B - \epsilon_B, C(b_2) = V_B + \epsilon_B$. Suppose that $\epsilon_B > \epsilon_A$ (in other words, that agent B is closer to its goal than agent A). Next, assume that the desirable strategy profile is $\{a_2, b_1\}$, which must be achieved through the regret minimization framework (i.e., the agent with the shortest path to its goal continues to move along it, while the other agent takes a different, potentially suboptimal, route).

Initially, the only information available to A is $C(a_1)$ and $C(a_2)$, causing it to select the greedy choice of choosing a_1 as $C(a_1) < C(a_2)$. Similarly, agent B will also select b_1. Each agent then simultaneously plays their selected actions and observes what the other agents are playing. The idea is that at each step of the game, each agent observes one another and then computes a loss vector for its corresponding choices. The definition of the loss vector includes a multiplication factor η called the learning rate, which goes beyond penalizing an action purely on how much distance the two agents approached each other by.

Thus, after the first iteration, the loss function for agent A (and similarly for agent B) looks like this: $l^1(a_1) = 2\eta_1\delta$ (since both agents approached each other) and $l^1(a_2) = 0$, where δ is how much the two agents moved between step 0 and step 1. The agents are assumed to be moving on a 1-dimensional circle.

Then after k iterations during which the two agents have not changed strategies, the regret-loss vector (LR) will be:

$$\begin{aligned} LR^k(a_1) &= 2k\eta_1\delta & LR^k(b_1) &= 2k\eta_2\delta \\ LR^k(a_2) &= 2k\delta + 2\epsilon_1 & LR^k(b_2) &= 2k\delta + 2\epsilon_2 \end{aligned} \quad (1)$$

In order for agent A to switch actions through regret minimization and end in the desired strategy profile of $\{a_2, b_1\}$, it is necessary that: $LR^k(a_1) > LR^k(a_2)$ and $LR^k(b_1) < LR^k(b_2)$. Substituting values in and solving for k results in the following inequality:

$$\frac{\epsilon_1}{\delta(\eta_1 - 1)} < k < \frac{\epsilon_2}{\delta(\eta_2 - 1)}$$

By setting the learning rate to be

$$\eta_i = \frac{\delta}{\epsilon_i} + 1 \tag{2}$$

it is then possible to achieve the following time constraint on when the agents will achieve a switch in strategy:

$$\frac{\epsilon_1^2}{\delta^2} < k < \frac{\epsilon_2^2}{\delta^2} \tag{3}$$

Therefore, if $\epsilon_2 - \epsilon_1 > 0$ and $\epsilon_2 > \delta$, then there is an iteration k for which the first agent will switch to a_2, while the second agent will remain on b_1. Then, at iteration $k+1$, the loss vectors for the agents will be:

$$l^{k+1}(a_1) = 2(k+1)\eta_1\delta \qquad l^{k+1}(b_1) = 2k\eta_2\delta$$
$$l^{k+1}(a_2) = 0 \qquad l^{k+1}(b_2) = 2\eta_2\delta$$

Thus agent A will continue to penalize action a_1, while agent B starts penalizing b_2. After this point, it is clear that the agents will keep selecting the actions $\{a_2, b_1\}$.

General Motion Coordination Games

Now that a method for solving a specific motion coordination game using regret minimization has been described, the generalization of this solution is presented. Each of the following sections provides details on extending the example formulation to work in a general path planning format.

Replanning Framework

Replanning has the effect of turning a standard path planning problem into a repeated game, rather than a single game, which thus matches the formulation specified in the prototypical motion coordination example. Each agent is given a small amount of time, dt, which is the agent's planning cycle and represents how long the algorithm has before it must return an action for the agent to select. The planning cycle also represents the frequency with which the agent senses its environment, and consequently how often it obtains information about its neighbors.

Action Set Computation

While it was straightforward to see which two actions an agent had in the example problem, it is not as easy in a general setting, since there is an infinite number of paths an agent could traverse. However, only the trajectories belonging to distinct least-cost homotopy classes need to be considered. There are many methods available that can accomplish homotopic class computations [Jaillet and Simeon2006] [Bhattacharya, Kumar, and Likhachev2010]. In order to construct the agent's action set with these least-cost homotopy classes, the notion of path separation [Green and Kelley2007] is utilized. The resulting algorithm is somewhat heuristic in nature, with regards to not guaranteeing different homotopic paths, however the variety of paths it computes provides agents with a sparse but descriptive action set. An algorithm for computing an agent's action set is given in Algorithm 1.

Algorithm 1: Action Set Computation

Data: A roadmap of the environment: $G(V, E)$,
Agent i's start and goal states: q_i^{init}, q_i^{goal}
Number of candidate actions and final actions: N, M
Result: The agent's action set $\mathbb{A}_i$

1 $\mathbb{A}'_i = \emptyset$
2 **while** $|\mathbb{A}'_i| < N$ **do**
3 $\quad \tau =$ RESOLVE_QUERY$(G, q_i^{init}, q_i^{goal})$
4 $\quad \mathbb{A}'_i = \{\mathbb{A}'_i, \tau\}$
5 $\mathbb{A}_i = \{\mathbb{A}'_i(0)\}$
6 **while** $|\mathbb{A}_i| < M$ **do**
7 $\quad P_{candidate} = \emptyset$
8 $\quad$ **foreach** *Action* $a \in \mathbb{A}_i$ **do**
9 $\quad\quad b =$ MAX_SEPARATION$(a, \mathbb{A}'_i)$
10 $\quad\quad P_{candidate} = \{P_{candidate}, b\}$
11 $\quad \tau_{new} = argmax_\tau(P_{candidate})$
12 $\quad \mathbb{A}_i = \{\mathbb{A}_i, \tau_{new}\}$
13 Return $\mathbb{A}_i$

Initially, Algorithm 1 is given a roadmap $G(V, E)$ along with the start and goal configurations of the agent as input parameters. Although any roadmap generating method will work, a $PRM*$ [Karaman and Frazzoli2011] was used as it provided a dense roadmap which asymptotically can provide optimal solutions. The other two input parameters, N and M, determine the following: N gives a bound on how many paths from the roadmap are added to the candidate action set, while M is the number of actions in the final action set.

Next, the roadmap is repeatedly queried with the agent's start and goal positions for N iterations (Lines 2- 4). After each query, the resulting shortest path is returned and stored into $\mathbb{A}'_i$, while the edges of the roadmap along this resulting path has its corresponding edge weight set to a very large value (Line 3). Setting these edges to a large value heuristically guarantees that successive queries on the roadmap will generate different paths. Next, the first computed path in the candidate action set is removed and inserted into the agent's final action set. Until the final action set has M actions, continue with the following: for each action a in $\mathbb{A}_i$, find the path b that has the largest separation to a (Lines 8-10). Separation is computed by summing the Euclidean distance between each time-paired configuration (i.e., configurations at time 0, configurations at time 1, etc.) in a and b. b is then added to the candidate path set $P_{candidate}$. The trajectory with the largest separation value is found and appended onto the final action set (Lines 11-12).

Regret Minimization with Learning

Once an agent has computed their action set, the regret minimization approach is applied in order to compute the agent's action. As in the motion coordination example, for the first step $t = 1$, agents will choose their greedy action (the action with the lowest cost path to their goal). Then, $\forall t > 1$, the agents use the method shown in Algorithm 2.

In order to update the learning vector, for each trajectory $\tau_a : a \in \mathbb{A}_i$, apply τ_a to the agent's previous configura-

Algorithm 2: Learned Regret Minimization

Data: The agent's action set $\mathbb{A}_i$
Result: The minimized learned regret action: a_{min}
1 UPDATE_LEARNING_VECTOR(l_i)
2 UPDATE_ACTIONS($\mathbb{A}_i$)
3 **foreach** *Action* $a \in \mathbb{A}_i$ **do**
4 | Action Cost $C(a) =$ `length(a)`
5 Best Cost $C(a^*) = min\ C(a)$
6 **foreach** *Action* $a \in \mathbb{A}_i$ **do**
7 | Learned Regret $LR(a) = C(a) - C(a^*) + l_i(a)$
8 $a_{min} =$ argmin$_a$ $LR(a)$.

tion $q_i(t-1)$ to obtain a new configuration $q_i^{new}(t)$. δ is measured as the distance between $q_i^{new}(t)$ and each of the neighboring agents' configurations (Line 1). As agents move through the environment, the paths in the agent's action set may no longer be applicable due to avoiding unexpected obstacles. Thus, a reconnection strategy, which updates the homotopic paths, must be employed. For each path τ_a corresponding to action a, a portion of the beginning of the trajectory is removed, and a path connecting the current configuration to τ_a is appended (Line 2). The cost of each action is estimated by computing the length of the corresponding paths. Standard regret minimization is then utilized to compute the selected action (Lines 3-8) .

Simulations

The approach was implemented on a simulation software platform used to develop motion planning algorithms, called `PRACSYS` [Kimmel et al.2012]. The approach was evaluated for two agents in the corridor environment from the prototypical motion coordination example, as well as a random environment populated with varying sized circular obstacles.

The experiments were run on a single computer with a 3.1 GHz Intel i3 processor and 8GB of RAM. The proposed framework was compared against the greedy choice, where the agents move along the shortest path on the roadmap computed by $PRM*$. Both methods were evaluated in performance with regards to the following metrics: average solution time (each agent has reached their goal) in seconds S_{avg}, total number of obstructive interactions with agents during the entire experiment C_{tot}, and average computation time (in seconds) per planning cycle T_{avg}.

Either method can use a reactive method for collision avoidance. Since this choice directly affects the solution time of the agents, collision avoidance was removed from both methods, and instead the number of obstructive interactions was measured. Thus, the metrics computed are unbiased.

Corridor Experiments

The purpose of the corridor experiments was to validate the proposed approach in the prototypical example. A mock up of the environment is shown in Figure 2. In the experiment, the agents are given two actions, one for each corridor. Three different cases of the corridor environment were tested. Case 1: One agent is ϵ committed to a corridor. Case 2: Both agents are ϵ committed to the same corridor. Case 3: Neither agent is ϵ committed to a corridor. For each case, ten experiments were run and averaged, with agents placed in random configuration but within constraints of the case. The results are shown in Table 3.

Figure 2: The environment from the prototypical motion coordination example. It contains two least-cost homotopic classes of paths, each corresponding to a different corridor.

In nearly all of the experiments, the agents moving without coordination ended up in an obstructive interaction. Although acting without coordination required no additional computation time, the additional overhead from the proposed framework was relatively small (agents were given 0.5 seconds per planning cycle). As for the path costs, coordination performed better in case 1, but did not perform so well in case 3. The reason for this is that both agents are not ϵ committed to a particular corridor, thus some oscillations occur until one agent continues down the same corridor.

	Without Coordination			**With Coordination**		
	S_{avg}	C_{tot}	T_{avg}	S_{avg}	C_{tot}	T_{avg}
Case 1	20.04	7	0	19.74	0	0.27
Case 2	20.36	10	0	21.99	0	0.27
Case 3	18.47	5	0	23.82	0	0.27

Table 3: Experimental results for two agents in three different cases of the corridor environment.

Random Obstacle Experiments

The purpose of the random obstacle experiments was to test the proposed approach in a different setting. A mock up of the environment is shown in Figure 3. In the experiment, the agents are given five different actions, which covered different least-cost homotopic classes. The two different cases correspond to different arrangements of obstacles. For each case, ten experiments were run and averaged, with the results shown in Table 4.

	Without Coordination			**With Coordination**		
	S_{avg}	C_{tot}	T_{avg}	S_{avg}	C_{tot}	T_{avg}
Case 1	20.93	5	0	27.28	0	0.60
Case 2	22.04	10	0	23.01	0	0.56

Table 4: Experimental results for two agents in two different randomly generated environments

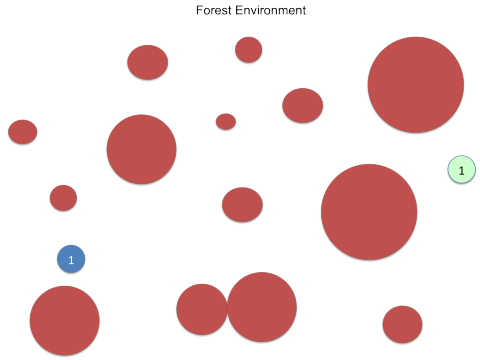

Figure 3: This environment contains randomly placed, varying sized circular obstacles. It contains several least-cost homotopic classes of paths.

As in the previous set of experiments, the coordinating agents were once again able to avoid all obstructive interactions. Although the proposed motion coordination framework ended up producing paths that were longer, and also ended up taking more time (0.6 seconds out of a 0.8 second planning cycle) due to the increased number of actions and obstacles, it successfully kept agents from obstructing one another.

Discussion

As this is a work in-progress, there are many aspects of the framework that can be improved upon. Rather than using the heuristic method of generating different homotopic paths during action set generation, the framework could benefit from approaches that actually do compute homotopy classes [Bhattacharya, Kumar, and Likhachev2010]. For large numbers of actions, another possible avenue of extension is through the use of a regret-based probability distribution (i.e. mixed strategy, rather than pure strategy) over the agent's actions, which could help achieve a better overall utility [Nisan et al.2007].

References

Bhattacharya, S.; Kumar, V.; and Likhachev, M. 2010. Search-based path planning with homotopy class constraints. In *Third Annual Symposium on Combinatorial Search*.

Ehtamo, H., and Raivio, T. 2001. On applied nonlinear and bilevel programming for pursuit-evasion games. *Journal of Optimization Theory and Applications* 108:65–96.

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).

Ghrist, R.; O'Kane, J. M.; and LaValle, S. M. 2005. Computing pareto optimal coordinations on roadmaps. *The International Journal of Robotics Research* 24(11):997–1010.

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

Halpern, J. Y., and Pass, R. 2012. Iterated regret minimization: A new solution concept. *Games and Economic Behavior* 74(1):184–207.

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

Isaacs, R. 1965. *Differential Games*. Wiley.

Isler, V.; Sun, D.; and Sastry, S. 2005. Roadmap-based pursuit-evasion and collision avoidance. In *Robotics: Science and Systems*.

Jaillet, L., and Simeon, T. 2006. Path Deformation Roadmaps. In *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. 2010. Incremental sampling-based algorithms for a class of pursuit evasion games. In *Workhop on the Algorithmic Foundations of Robotics (WAFR)*.

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

Kimmel, A.; Dobson, A.; Littlefield, Z.; Krontiris, A.; Marble, J.; and Bekris, K. E. 2012. Pracsys: An extensible architecture for composing motion controllers and planners. In *Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*.

Knepper, R. A., and Rus, D. 2012. Pedestrian-inspired sampling-based multi-robot collision avoidance. In *RO-MAN, 2012 IEEE*, 94–100. IEEE.

Lachner, R.; Breitner, M. H.; and Pesch, H. J. 1995. Three-dimensional air combat: Numerical solution of complete differential games. In Olsder, G. J., ed., *New Trends in Dynamic Games and Applications*. Birkhauser.

Lavalle, S. M. 1995. *A game-theoretic framework for robot motion planning*. Ph.D. Dissertation, University of Illinois.

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

Shi, D.; Collins, E.; Donate, A.; Liu, X.; Goldiez, B.; and Dunlap, D. 2008. Human-aware robot motion planning with velocity constraints. In *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, 490–497. IEEE.

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.

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)*.

Vanek, O.; Bosansky, B.; Jacob, M.; and Pechoucek, M. 2010. Transiting areas patrolled by a mobile adversary. In *Proc. IEEE Conf. on Computational Intelligent and Games*.

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.