**2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA**

Anytime Navigation with Progressive Hindsight Optimization

Julio Godoy, Ioannis Karamouzas, Stephen J. Guy and Maria Gini

Abstract— **In multi-robot systems, efficiently navigating in a a partially-known environment is an ubiquitous but challenging task, as each robot must account for the uncertainty introduced, for example, by other moving robots. This uncertainty makes pre-computed plans not always applicable, and often hinders the desired efficient use of the robot's resources. In this work, we present a local anytime approach for robot motion planning that accounts for the uncertainty of the environment by generating 'snapshots' of possible future scenarios. Our approach adapts the Hindsight optimization technique to allow robots to plan their immediate motion based on long-term efficiency. We validate our approach by comparing the efficiency on the paths executed against a state-of-the art navigation technique in a variety of scenarios, and show that by accounting for the uncertainty in the environment, agents can improve their time- and energy-efficient motions.**

I. INTRODUCTION

The task of safely steering an agent amidst static and dynamic obstacles has many applications in robotics, from vacuum cleaning using Roombas to delivering parts for packaging in warehouses [1]. Often, it is important for the robot to navigate in an efficient manner, for example, to save resources (i.e., battery life) or reach critical locations (e.g., in search and rescue applications). However, when multiple agents or dynamic obstacles are simultaneously moving in the environment, navigation becomes much more challenging; potential conflicts between the agents' paths result in uncertainty in their future trajectories. This uncertainty prevents robots from following a pre-computed ideal navigation plan.

A common way to address this problem is by computing optimal paths for the robots in a composite configuration space. However, this requires a centralized planner with perfect knowledge of the environment, an assumption that may not always hold. An alternative approach is to decouple global planning from local collision avoidance and periodically replan a robot's path when significant changes are observed. However, these decentralized solutions do not typically account for uncertainty induced by the presence of the dynamic entities sharing the environment.

In this paper, we propose an *anytime local* approach to plan the motions of multiple agents in a decentralized manner. We adapt the Hindsight optimization technique [2] from AI to account for uncertainty in the multi-agent navigation problem and introduce the concept of *Progressive Hindsight optimization* (PHOP), which allows each robot to plan ahead and compute its immediate actions based on possible future situations that it may encounter. The resulting algorithm is anytime, enabling robots to follow in real-time the best solution found so far and improve their computed action plans when more time is available. As compared to state-of-the-art local navigation techniques, our method leads to more time- and energy-efficient paths due to the ability of the robots to anticipate and respond proactively.

*All authors are with the Department of Computer Science and Engineering, University of Minnesota, USA

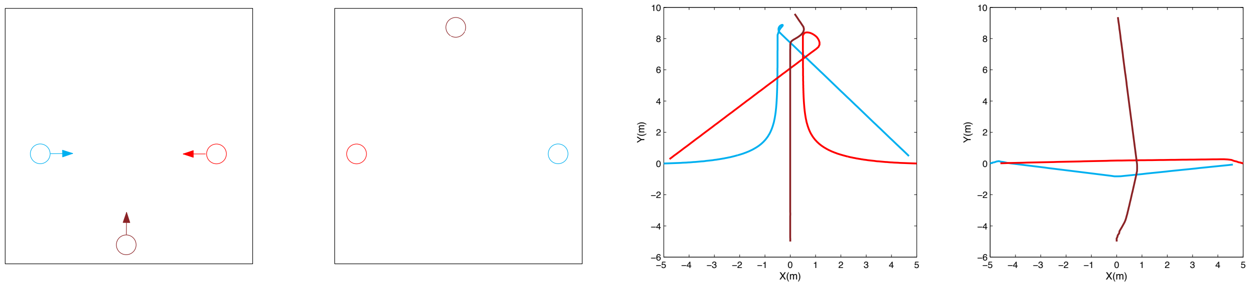
The rest of this paper is organized as follows. We give a brief overview of prior work in Section II. In Section III, we highlight the Hindsight optimization technique and discuss how it can be used for multi-agent navigation. Section IV presents our progressive Hindsight optimization approach. In Section V, we evaluate the efficiency of our planner on several benchmarks. Finally, some conclusions and plans for future research are presented in Section VI.

II. RELATED WORK

Computing a collision-free motion from a given start to a given goal configuration has been extensively studied in robotics (see [3] for an overview). When multiple agents need to simultaneously move in an environment, the problem is typically handled by centralized planners and decoupled methods. Centralized planners combine the configuration spaces of the agents into a composite one (e.g., [4]), whereas decoupled methods, such as in [5], plan a path for each robot independently and coordinate the resulting paths. Space-time planning algorithms have also been proposed that add the time dimension to the agent's configuration space and plan in a joint state-time space [6], [7]. However, all these aforementioned approaches are quite slow and are not suited for real-time navigation of multiple agents.

An alternative approach is to decompose global planning from local collision avoidance. Typically, a roadmap is used to direct the global motion of each agent, whereas collisions with other agents and the static part of the environment are resolved by using a local planner. Over the past twenty years, numerous local collision avoidance techniques have been proposed based on potential fields [8], social forces [9] and Velocity Obstacles variants [10], [11].

Different global planning approaches have also been proposed using PRMs [12], RRTs [13] and navigation meshes [14]. Approaches have also been introduced that dynamically update the weights of a roadmap [15] and periodically replan paths in an anytime, deterministic fashion [16], [17]. However, due to the decentralized nature of these planners, an agent cannot typically account for the uncertainty induced by the presence of the other agents in the environment. In addition, it is not always obvious when an agent needs



(a) Start positions (b) Goal positions (c) ORCA (d) PHOP

Fig. 1: **Crossing (3 Agents)**. Three agents cross paths. (a) Initial positions of the agents. (b) Goal positions of the agents. (c) When navigating with ORCA, the agents run into and push each other resulting in inefficient paths. (d) When using our PHOP approach the agents plan around potential upcoming collisions, resulting in more efficient paths. In particular, the agent moving vertically adjusts to avoid the horizontal agents.

to replan its path. To address these issues, we propose an anytime local approach based on the Hindsight optimization technique from AI and combine it with the ORCA navigation framework proposed in [11] to guarantee collision-free motions. Our approach can be seen as a form of Receding Horizon Control [18], although it does not require communication between the agents and allows planning for longer time horizons.

III. Overview

We address the problem of goal-oriented navigation for multiple agents in real world environments, where uncertainty caused by dynamic elements may reduce the efficiency in their navigation. Consider for example the three agents in Figure 1a. Their goal is to reach the opposite side of a room (Figure 1b). If any of the agents does not consider the potential future paths of the other two agents, it may easily get stuck in a local minimum and its motion becomes very inefficient (Figure 1c), as it may need to wait until one of the other agents reaches the goal before continuing its path. If, instead, the agent considers the future movement of its neighbors, it could predict the local minimum and plan accordingly to avoid it (Figure 1d), improving the efficiency of its motion. In this work, we propose an anytime planning approach, based on Hindsight optimization [2], to increase the agent's awareness of potential future states of the environment and account for the likely actions of its nearby neighbors.

A. Problem Description

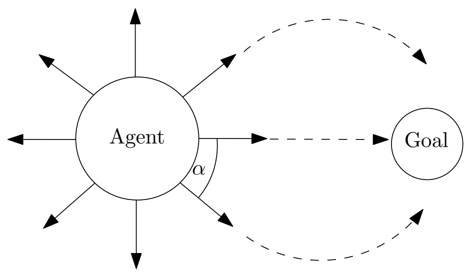
We assume we are given a set of n heterogeneous agents, each one with a unique start and goal position, $\mathbf{g}$, specified in $\mathbb{R}^2$. The task, then, is to *safely* and *efficiently* steer the agents to their goals in a decentralized, collision-free manner. We assume that an agent located at position $\mathbf{p}$ can advance by selecting an action a from a set of actions A (see Figure 2a). In order to ensure collision- free motion, these actions are not taken directly, but rather are used as a preferred velocity $\mathbf{v}_{\text{pref}}$ in a collision avoidance subroutine. Consequently, during each simulation cycle, an agent has to select an action ($\mathbf{v}_{\text{pref}}$) that leads to efficient goal-oriented navigation, and this action will result in a new current velocity $\mathbf{v}$ as mediated by the collision avoidance method. We assume that the agent has perfect sensing within a fixed range of 15 meters.

B. ORCA Collision Avoidance

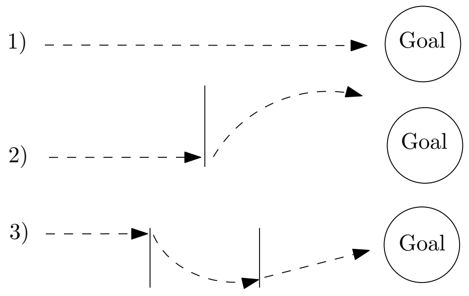
Selecting collision-free velocities as close as possible to $\mathbf{v}_{\text{pref}}$ is a challenging problem in and of itself. Here, we use the ORCA (Optimal Reciprocal Collision Avoidance) navigation framework proposed by Berg et al. [11], a geometric approach which provides a computationally efficient solution to this problem. While ORCA provides provably optimal motion in some limited scenarios, it is, in general, not a sufficient method to generate efficient paths for multiple agents navigating in complex, shared environments. In particular, ORCA can lead agents to choose velocities that lead to local minima such as standing still, rather than selecting velocities with longer term benefits like, for example, moving sideways to avoid congestion. Our approach builds on ORCA while allowing agents to change their preferred velocity each time step to improve their individual motion efficiency.

C. Uncertainty

A fundamental problem for an agent navigating in an environment shared with other agents is the uncertainty in the future motion of those other agents, and the uncertainty in how these agents will respond to any new agent approaching them. Our methods proposes to account for this uncertainty by having each agent account for many different potential future states for each action before executing any action. In this way, an agent can, in 'hindsight', evaluate and select an action that in the long term seems more efficient, even if it may be less efficient locally. In our approach, each agent extrapolates a potential deterministic future based on the motions of other agents in its neighborhood, and evaluates the impact of many different actions sequences. This approach is analogous to the Hindsight optimization technique commonly used in the probabilistic planning domain.



(a) Action space



(b) Example Trajectories

Fig. 2: **Actions**. a) The nine available actions correspond to moving at 1.5 m/s with different angles with respect to the goal: 0°, α, $-\alpha$, 90°, $-90°$, 180°, $180° + \alpha$, $180° - \alpha$ and complete stop. In our implementation, $\alpha = 20°$. b) Example of progressive agent trajectories with 1, 2 and 3 partitions respectively.

D. Hindsight Optimization

Hindsight optimization (HOP) is a technique for online action selection used to compute an upper bound on the expected reward of an action by allowing the agent to 'peek' on potential future scenarios and plan accordingly. Although the expected reward is not a tight upper bound on the real value of the action, it is often indicative enough for action comparison purposes [19]. HOP has been applied to many domains [2], [20]. An agent using HOP formulates a series of *T-Horizon* futures of length *T*, computes a policy for each time $t \in T$ and obtains rewards for executing that policy from $t = 0$ until $t = T$, which is associated with the policy at time $t = 0$ (i.e. the first action executed). By generating multiple deterministic future scenarios, and averaging the results under the same initial action, the agent can estimate, in hindsight, the best action to perform in the present. For a more detailed theoretical description of Hindsight optimization, we refer the reader to [20].

E. Anytime Approaches

Although using HOP can result in large computational gains [20], the planning time must be balanced with the need to quickly adapt to new scenarios. This is critical assuming that agents are navigating through a partially-known environment, continuously sensing and updating information about the environment that may invalidate previously generated plans. Real time constraints may require aborting the planning process for an action to be performed, or when new relevant information is gathered. Our anytime planner computes paths in a progressive manner to ensure that when the planning process must be stopped, the current best plan can be executed. Because of this anytime feature, we call our approach Progressive Hindsight optimization (PHOP).

IV. APPROACH

The Progressive Hindsight optimization approach works by allowing each agent to simulate possible plans of actions for a given time horizon, and to analyze in 'hindsight' the consequences of each plan. It accounts for the uncertainty in the environment by creating potential future scenarios, in order to obtain an estimate of the value of the agent's immediate actions. We assume nine available actions for the agent, each corresponding to a different preferred velocity value $\mathbf{v}_{\text{pref}}$. See Figure 2a for details.

Our approach is executed at each simulation cycle and finishes when the agent is at its goal position. At each timestep, the agent follows the sense-plan-act cycle.

A. Reward Function

We evaluate an action based on the agent's local information (position and velocities of neighbors). We decompose the reward function into two components, that measure the *progress* of the agent towards its goal, and the *energy* that the agent is expected to spend.

The progress component is defined as the scalar product of the agent's velocity $\mathbf{v}$ with the normalized vector pointing from the position $\mathbf{p}$ of the agent to its goal $\mathbf{g}$:

$$Progress(a) = \mathbf{v} \cdot \frac{\mathbf{g} - \mathbf{p}}{\|\mathbf{g} - \mathbf{p}\|}. \tag{1}$$

As such, it highly rewards actions that move the agent closer to its goal.

The energy component approximates the energy that is expected to be consumed (per unit time) by the agent while it is moving as follows:

$$Energy(a) = b + c \cdot \|\mathbf{v}_{\text{pref}}\|^2. \tag{2}$$

Here, the first term, b, corresponds to the energy consumed by the agent due its processing and sensing capabilities, while the second term, $c \cdot \|\mathbf{v}_{\text{pref}}\|^2$, corresponds to the energy consumed by the agent while moving and is proportional to the square of the agent's preferred velocity. In our implementation, we use 1 as the value for both b and c. Note that by using the preferred velocity instead of the actual collision-free velocity $\mathbf{v}$, we are comparing the action's intended effect (Eqn. 2) with its actual outcome (Eqn. 1). In combination, our reward function encourages actions that maximize goal oriented motions while using the least amount of energy:

$$R(a) = \frac{Progress(a)}{Energy(a)}. \tag{3}$$

B. Plan generation and execution

The agent performs the plan generation process based on Hindsight optimization. This process consists of simulating multiple plans of actions in the future. For each plan, the agent internally simulates the execution of a sequence of actions for a specified number of timesteps (*T-Horizon*) in the future. During each timestep of the simulation, the agent updates its potential position and velocity, based on the prediction of its relative position to the static obstacles and to its neighbors' positions and velocities. The agent also predicts its neighbors' motion based on their observed velocities assuming that they are using ORCA to avoid potential collisions in these predictions. Based on each predicted timestep, the agent computes its potential reward using Eqn. 3. After the *T-Horizon* of the projected plan is reached, the agent averages its reward across the T simulated timesteps, and associates the averaged reward with the initial action of the plan. The planning continues until a specified time limit is reached. This allows the agent to have a broad estimate of the value of each of its actions when time is very limited, and to get a more accurate estimate of the expected reward for each action when more time is available. Finally, the agent performs the action that maximizes the expected reward based on the plans generated.

In order to account for unforeseen changes in the environment (for example, other agents changing their motions in an unexpected manner), it becomes necessary to replan at every timestep, allowing the agent to adapt to the new positions and velocities of the other agents. Hence, our planner is executed at every timestep to increase the agent's adaptability.

Compared to the original Hindsight optimization technique, our approach has two main differences. First, instead of generating different futures independent of the agent's actions, we consider the different futures that could occur as a consequence of the agent taking a particular sequence of actions. This is because the agent and its neighbors influence each other with their actions. Second, to make the approach anytime, we need to ensure that the quality of the obtained solution increases as more time is given to the algorithm. To do this, we increasingly partition the *T-Horizon* across the different available actions, computing a more detailed plan as the number of partitions increases. Fig. 2b shows three example action plans with 1, 2 and 3 partitions respectively.

C. Algorithmic Description

Algorithm 1 presents the pseudocode for a simulation loop using PHOP. It takes as input the *T-Horizon* desired for the plan ($timeHorizon$) and a time limit for the planning process ($timeLimit$).

The function $generatePlans(partitions)$ returns a set of plans ($PlanSet$) for the agent to simulate. We adopted a top-down approach for plan generation: we first simulate the agent following each of its individual actions for the entire *T-Horizon*, and then incrementally partition the *T-Horizon* (according to the value of $partitions$) to combine the simulation of multiple actions. Instead of populating vectors for each plan we adopted a memory efficient approach that dynamically generates each action, based on the current number of partitions desired. The function $hindsight(plan, timeHorizon)$ takes as input a single plan

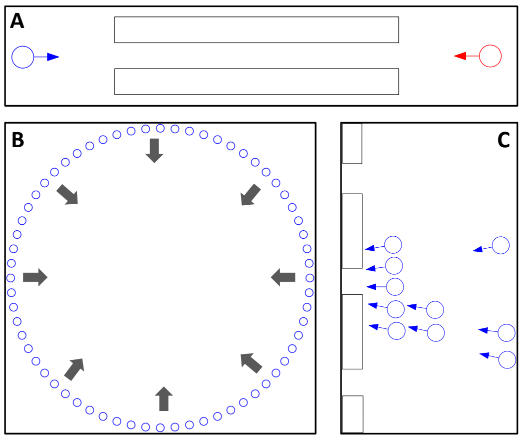


Fig. 3: **Simulated scenarios**. a) *Corridor*: Two agents travel to reach the opposite side of the narrow corridor, and must select their path before entering it. b) *Circle*: 64 agents move to their antipodal points in a circle, but congestion forms in the middle. c) *3-Exit*: Ten agents must exit the room through one of the exits, where the center exit is on the shortest path but congestion develops.

Algorithm 1: PHOP step for an agent

```
1: Input: $timeHorizon \in \mathbb{Z}$, $timeLimit \in \mathbb{R}^+$
2: initialize simulation, $\mathbf{p} = (x_{start}, y_{start})$
3: while not at the goal do
4:     $partitions \leftarrow 1$
5:     $time \leftarrow 0$
6:     while $time \leq timeLimit$ do
7:         $PlanSet \leftarrow generatePlans(partitions)$
8:         for all $plan \in PlanSet$ do
9:             $PlanR \leftarrow hindsight(plan, timeHorizon)$
10:            $a \leftarrow initialAction(plan)$
11:            $Count(a) \leftarrow Count(a) + 1$
12:            $AvgR(a) \leftarrow \frac{AvgR(a)*(Count(a)-1)+PlanR}{Count(a)}$
13:            $a^* \leftarrow \arg\max_{a \in A} AvgR(a)$
14:            $checkTime(timeLimit)$
15:        end for
16:        $partitions \leftarrow partitions + 1$
17:    end while
18:    $PerformAction(a^*)$
19: end while
```

and the *T-Horizon* specified. In this function, the agent performs the plan generation process described in the previous section and returns its reward value. The function $intialAction(plan)$ returns the first action in the sequence specified by $plan$, and it is used to relate the reward of the plan to that initial action.

The planning process finishes when the time limit has been reached, as checked both in line 6 and the $checkTime(timeLimit)$ function. If this is the case, the algorithm returns the currently best evaluated action (a^*).

This action serves as an input preferred velocity to the ORCA framework in the $PerformAction(a^*)$ function, which in turn computes a new collision-free velocity for the the agent and the sense-plan-act cycle is repeated.

V. Experimental Results

To better understand the performance of our proposed approach we compared it to the performance of ORCA in a variety of scenarios. Below we briefly describe each scenarios:

- **Crossing**: Three agents cross paths while moving towards their goals. (Fig 1a)
- **Corridor**: Two agents start at opposite sides of a long, narrow corridor. The two agents cannot both fit simultaneously. (Fig 3a)
- **Circle**: 64 agents walk to antipodal points on a circle. (Fig 3b)
- **3-Exit**: Ten agents exit a room with three exits. The central exit is closest to most agents, but the congestion will slow down the agents if all of them go through the same exit. (Fig 3c)

In all scenarios, agents were given a planning horizon (*T-Horizon*) of 60 seconds, and each agent was given 100ms to run Algorithm 1 unless otherwise noted. These parameter values were chosen as they allowed agents to differentiate between plans, and to keep real-time performance of our approach, respectively. Each experimental result is the average of 10 simulations. Small random perturbations were added to the preferred velocities of the agents to prevent symmetry problems. Results were gathered on an Intel Core 2 Duo at 2.4 GHz.

A. Performance Results

One way to measure the performance of the navigation method is by computing the maximum time taken for the agents to reach their goals. However, we can get a better context for the performance of our approach by comparing it to the theoretically best performance possible. In general, the true optimal performance is difficult to compute exactly, but we can compute a theoretical lower bound of this value by dividing the length of shortest path to the goal by the maximum agent speed. Fig. 4 shows the corresponding results. In all cases, the time taken for agents to reach their goals when navigating using PHOP is shorter than when using ORCA, often significantly shorter. The smallest relative performance difference comes in the *Circle* scenario, this is because for most of the time agents are moving in linear, uncontested paths. Our method consistently outperforms ORCA with improvements ranging from 45% (*Circle*) to 99% (*Corridor*, *Crossing*) with respect to the lower bound of the optimal travel time.

We can also measure an agent's performance in terms of energy used to travel to the goal. As described in Eqn. 3, our agents seek to maximize progress to the goal, per unit of expected energy expended. We would therefore expect improved performance both in terms of time and energy efficiency. To test the latter, we compute the average energy expended by each agent on its path to the goal. Recall, our measure of energy includes both the energy cost of being on and responsive and the energy cost of moving. Again, we see that in all cases, the average energy taken by agents navigating with our method is much lower than when using ORCA alone (see Table I).

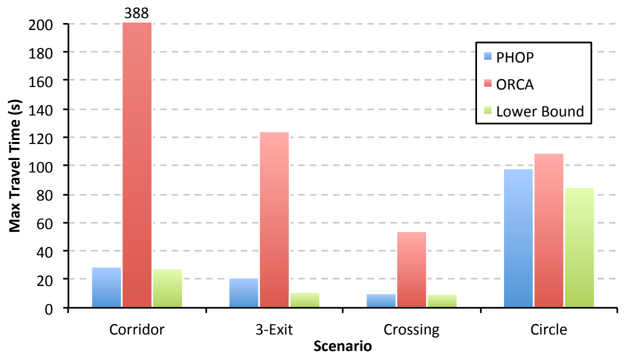


Fig. 4: **Performance (Time)**. A comparison of the maximum travel time across four scenarios using PHOP, ORCA and a lower bound estimate of the optimal travel time.

TABLE I: **Performance (Energy)**. A comparison of the average energy (in $J \cdot Kg^{-1} \cdot s^{-1}$) expended by the agents across four scenarios using PHOP and ORCA.

Method	Corridor	3 Exit	Crossing	Circle
ORCA	4,321	741	662	2,842
PHOP	893	317	254	2,790

B. Analysis

Agents employing our navigation approach display a variety of emergent behaviors which contribute to their improved efficiency. For example, agents in the *Crossing* scenario (Fig 1d), display an implicit form of coordination to resolve the potentially upcoming congestion: as the two horizontally moving agents reciprocate to avoid the upcoming collision, the vertically moving agent moves far to the right to avoid getting caught-up in the avoidance maneuver of the other two agents. This is in contrast to using a simple ORCA approach where the three agents move directly into a conflict and spend significant time to resolve it. A similar type of coordination is seen in the *Corridor* scenario (Fig 3a) where after one agent chooses to go into the corridor, the other plans a path that avoids the potential conflict created by entering the already occupied corridor. Again, this is in contrast to ORCA agents who meet in the middle of the corridor, slowing down and pushing each other until one agent has completely backtracked out.

Importantly, in both of the above scenarios, the coordination shown between agents is completely implicit with each agent choosing their own locally optimal actions without any communication. The lack of explicit communication allows our approach to scale to large numbers of agents. In the *Circle* scenario, over sixty agents must resolve mutual collisions to successfully reach their goals. When navigating

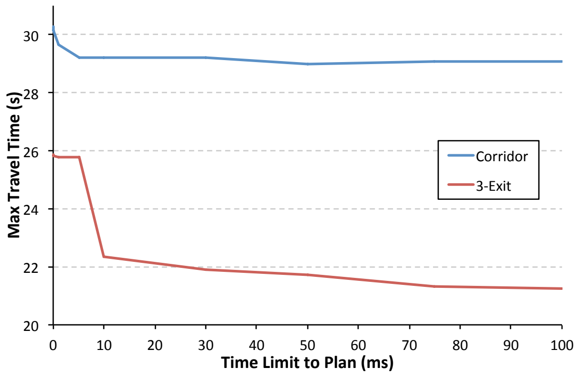


Fig. 5: **Anytime Behavior**. As each agent is given more time to plan, the overall performance improves. Even with limited planning time, our approach shows a significant improvement over ORCA (as shown in Fig. 4).

using PHOP, the agents plan future actions that resolve the heavy congestion in the middle of the circle very quickly.

Another emergent behavior seen by our agents is that agents will take longer, less direct paths when these paths are optimally more efficient than the direct path. The effect of this behavior is most clearly seen in the *3-Exit* scenario. Here, when using ORCA, all agents gather around the single closest exit leading to significant congestion around the exit. In contrast, with our approach, several agents choose to egress through one of the side exits, avoiding the congestion they could have experienced and reducing the total congestion experienced by all agents.

C. Computational Cost

As our approach is decentralized and each agent plans independently, its computational cost scales quadratically with respect to the number of neighbors, but is invariant to the total number of agents in the environment. PHOP is computationally more expensive than ORCA, as each agent solves a planning problem for its entire neighborhood, during a time period of (*T-Horizon*) timesteps. However, as described in Algorithm 1, our approach is anytime, and allows an agent to plan for as long as is appropriate and then take its chosen action. This is important if the approach is to be used as part of a sense-plan-act loop on a robot navigating online with an unknown amount of planning time. Figure 5 shows the performance of PHOP in the *3-Exit* and *Corridor* scenarios as a function of the planning time given to the agent. Even with very little time to plan, our approach results in significantly faster performance than ORCA. As planning time increases, the performance of the agents also increases, reaching a plateau at around 20ms of planning time.

VI. CONCLUSIONS

In this paper, we have introduced the concept of Progressive Hindsight optimization, an anytime algorithm for local navigation of multiple agents in dynamic environments. Combined with the ORCA framework, our solution can plan collision-free paths in real-time and in an anytime fashion. We validated our approach experimentally in a variety of environments. In all of our simulations, agents account for the uncertainty induced by the dynamic elements of the environment and intelligently plan a sequence of actions that lead to time- and energy-efficient motions.

Our approach combines elements from the fields of AI and robotics. We believe that the union of these two fields can be beneficiary to a wide range of planning problems. Our current solution, for example, is not applicable for navigation in unknown environments, as it assumes that the agents have at least partial knowledge of the environment. As such, we are currently investigating techniques that will allow the agents to adapt their motions in an online manner, as they navigate through the environment. Another possibility is to integrate PHOP with a framework that accounts for uncertainty in the sensors and actuators [21].

REFERENCES

[1] E. Guizzo, "Three engineers, hundreds of robots, one warehouse," *IEEE Spectrum*, vol. 45, no. 7, pp. 26–34, 2008.
[2] E. Chong *et al.*, "A framework for simulation-based network control via Hindsight optimization," in *IEEE Conf. on Decision and Control*, vol. 2, 2000, pp. 1433–1438.
[3] S. LaValle, *Planning algorithms*. Cambridge University Press, 2006.
[4] T.-Y. Li and H.-C. Chou, "Motion planning for a crowd of robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003, pp. 4215–4221.
[5] T. Simeon *et al.*, "Path coordination for multiple mobile robots: A resolution-complete algorithm," *IEEE Trans. Robot. Autom.*, vol. 18, no. 1, pp. 42–49, 2002.
[6] T. Fraichard, "Trajectory planning in a dynamic workspace: A 'state-time' approach," *Advanced Robotics*, vol. 13, pp. 75–94, 1999.
[7] J. van den Berg and M. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Trans. Robot. Autom.*, vol. 21, pp. 885–897, 2005.
[8] J. Reif and H. Wang, "Social potential fields: A distributed behavioral control for autonomous robots," *Robot. Auton. Syst.*, vol. 27, no. 3, pp. 171–194, 1999.
[9] D. Helbing *et al.*, "Simulating dynamical features of escape panic," *Nature*, vol. 407, no. 6803, pp. 487–490, 2000.
[10] J. Snape *et al.*, "Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 5917–5922.
[11] J. van den Berg *et al.*, "Reciprocal n-body collision avoidance," in *Int. Symp. of Robotics Research*, ser. Springer Tracts in Advanced Robotics, vol. 70. Springer, 2011, pp. 3–19.
[12] L. Kavraki *et al.*, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.
[13] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Int. Conf. on Robotics and Automation*, 2000, pp. 995–1001.
[14] R. Geraerts, "Planning short paths with clearance using explicit corridors," in *IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 1997–2004.
[15] W. van Toll *et al.*, "Real-time density-based crowd simulation," *Computer Animation and Virtual Worlds*, vol. 23, pp. 59–69, 2012.
[16] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Int. Joint Conf. on Artificial Intelligence*, vol. 14, 1995, pp. 1652–1659.
[17] M. Likhachev *et al.*, "Anytime dynamic A*: An anytime, replanning algorithm," in *Int. Conf on on Automated Planning and Scheduling*, 2005, pp. 262–271.
[18] J. Mattingley *et al.*, "Receding horizon control," *IEEE Control Systems*, vol. 31, no. 3, pp. 52–65, 2011.
[19] C. Browne *et al.*, "A survey of monte carlo tree search methods," *IEEE Trans. Comput. Intell. and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.
[20] S. W. Yoon *et al.*, "Probabilistic planning via determinization in Hindsight." in *AAAI*, 2008, pp. 1010–1016.
[21] S. Patil *et al.*, "Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty," in *IEEE Int. Conf. on Robotics and Automation*, 2012, pp. 3238–3244.