Fast discovery of influential outcomes for risk-aware MPDM

Dhanvin Mehta Gonzalo Ferrer Edwin Olson

***Abstract*— In the Multi-Policy Decision Making (MPDM) framework, a robot's policy is elected by sampling from the distribution of current states, predicting future outcomes through forward simulation, and selecting the policy with the best expected performance. Electing the best plan depends on sampling initial conditions with influential (very high costs) outcomes. Discovering these configurations through random sampling may require drawing many samples, which becomes a performance bottleneck.**

In this paper, we describe a risk-aware approach which augments this sampling with an optimization process that helps discover those influential outcomes. We describe how we overcome several practical difficulties with this approach, and demonstrate significant performance improvements on a real robot platform navigating a semi-crowded, highly dynamic environment.

I. Introduction

Multi-Policy Decision Making (MPDM) [1] provides a powerful framework for autonomous navigation among uncertain dynamic agents by choosing from amongst a set of closed loop policies - $\{$*Go-Solo, Follow, Stop*$\}$. Each candidate policy is evaluated based on forward simulations of samples drawn from the estimated distribution of the agents' states (Monte-Carlo sampling). These forward simulations and thereby the cost function, capture agent-agent interactions as well as agent-robot interactions which depend on the policy being evaluated.

Collisions are profoundly serious, but near misses are mundane. Sampling randomly is likely to miss high-cost events, even if they are individually reasonably probable (high probability density) because of the scarcity of such configurations in the state space (low total probability mass of high-cost outcomes). Discovering these configurations through random sampling may require drawing many samples, which becomes a performance bottleneck.

Without enough samples to find influential outcomes, the quality of planning suffers. Addressing this issue is crucial for reliable systems in applications such as autonomous cars, navigation in social environments, etc.

The key challenges arise from the uncertainty associated with the inferred state of the agents (people) and the complex multi-agent interactions which make the forward-simulated trajectories (Fig. 1) sensitive to the initial configurations sampled. The dimensionality of the space of all possible initial configurations is very large. However, typically, only the joint configuration of a small subset of the observed agents affect the cost function significantly.

The authors are associated with the University of Michigan, Ann Arbor. {dhanvinm,gferrerm,ebolson}@umich.edu.

This project was supported by the ONR ProbCog grant F033232.

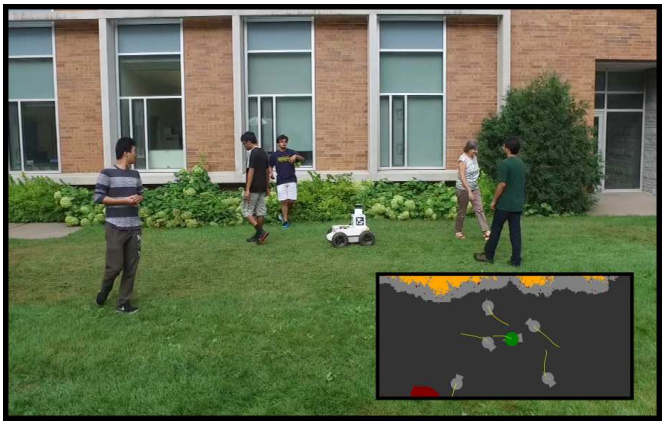

Fig. 1. *Top*: The MAGIC robot navigating among a group of pedestrians. *Bottom*: GUI of the system with observed agents are depicted by gray circles. The yellow lines show the predicted trajectories for a single initial configuration of the observed agents. The robot elects a policy by forward-simulating the interactions between itself and other agents.

In our application, it is especially difficult to sample bad outcomes because we assume that all agents follow policies that tend to avoid collisions and dangerous scenarios in the first place. The bad outcomes arise from a few initial configurations whose neighborhoods may be fairly uninteresting. In other words, the cost function tends to be tightly peaked around these configurations. This problem is more prevalent in complex multi-agent scenarios.

The key idea in this paper is that we explicitly search for influential outcomes (those that have high cost and probability) because they influence our decision making process the most. Our contributions are as follows:

- We formulate a risk-aware objective for evaluating policies to bias the sampling of initial configurations towards likely, high-cost outcomes. This is in contrast to the conventional approach which tried to minimize the expected cost.
- We propose a search-based approach that quickly computes an ordered list of promising perturbations. Furthermore, it is anytime, finding increasingly influential configurations at every iteration.
- We demonstrate the efficiency of this algorithm through extensive simulation experiments (Sec. VI) and show that using only 50 samples, our proposed method can perform comparably to sampling with 500 samples.
- Finally, we incorporate this idea into MPDM and demonstrate significant performance improvements on a real robot platform navigating in a semi-crowded highly dynamic environment (Sec. VII).

II. Related Work

Several approaches to autonomous navigation in dynamic environments restrict the state of the system exclusively to the robot's state. Reactive approaches provide a powerful technique for dealing with multiple dynamic agents [2], [3] by centering a potential field on them, but they suffer from local minima problems. Obstacle avoidance techniques [4], [5], provide a valid path avoiding possible obstacles or other agents. Augmenting the dimensionality of the system by considering other agents helps us model agent-agent interactions, required to deal with more complex situations. More recent approaches [6]–[8] account for the probabilistic future locations of dynamic obstacles and have been effective for safe navigation.

Inverse Reinforcement Learning approaches capture relevant features that explain the interactions taking place between dynamic agents [9], [10] although they may be limited by the training datasets used.

Visual tracking of human motion faces similar challenges in providing good solutions to these entangled environments. Bera and Manocha [11] extract the trajectories of people, improving their approach by a hybrid motion model of pedestrians. Large et al. [12] clustered recorded trajectories of pedestrians and used an informed Velocity Obstacle model for planning. More recently, Vasquez [13] proposed a prediction system based on a joint planning problem, assuming some optimal objectives to be sought by the pedestrians.

The work of Fulgenzi et at. [14] addressed a *full* state of the system by predicting the agents' trajectories, over a time horizon. However, this approach does not contemplate changes to the current predictions of the scene according to the robot's plans. An example of reciprocal evaluation of future trajectories and robot's plans is the work of Trautman et al. [15] using Gaussian Process to perform regression on the agents' trajectories and estimate the intentions of dynamic agents or the work of [16], considering temporal constraints and optimizing over several objectives. This paper follows a similar approach of conditioning predictions on the robot's policy and vice versa.

In addition to a large state, we must deal with uncertainty. POMDPs describe a complete theoretical framework to deal with uncertainty, although they quickly become intractable. Recent methods based on scenario sampling and forward simulation have recently been applied to autonomous navigation [17] and mapping [18], using similar approximations to those considered in MPDM [1].

In this work, we show the benefits of biasing sampling towards efficiently discovering high-cost outcomes for multi-policy decision making (MPDM). This same idea has been applied in Grisetti et al. [19] in the context of SLAM, where they introduced an informative search over their cost function.

III. Problem formulation

Our model of the environment consists of static obstacles (e.g. walls) and a set of freely moving dynamic agents, assumed to be pedestrians.

The robot maintains an estimate of the state of each observed agent i, which consists of its position and velocity in two dimensions and an inferred goal point. The collective state $\boldsymbol{x}_t$ consists of the robot state plus all the agents visible to the robot at time t. Throughout the paper, we will refer to $\boldsymbol{x}_0$ as the collective state at the current time. The robot maintains an estimate $P(\boldsymbol{x}_0)$ based on observations of the pedestrians' positions $\boldsymbol{z}$.

An initial configuration $\boldsymbol{x}_0$ is forward simulated until s time-steps, by iteratively applying the transition operator T_π, which yields the *trajectory*

$$\boldsymbol{X}(\pi, \boldsymbol{x}_0) = \{\boldsymbol{x}_0, T_\pi(\boldsymbol{x}_0), T_\pi^2(\boldsymbol{x}_0), \ldots, T_\pi^s(\boldsymbol{x}_0)\}. \quad (1)$$

The transition operator T_π can be viewed as a "black-box" – it can compute a likely future state $\boldsymbol{X}$ given an initial configuration $\boldsymbol{x}_0$ and the robot's proposed policy π. However, other types of analysis (e.g. inversion) are not possible. In practice, T_π is a simulator that proposes the forward dynamics of the agents. With a suitably small step size, these future predictions reflect the closed-loop interactions of agents (see [20] for details). Note that these simulations are performed online.

IV. Multi-Policy Decision Making Approaches

In MPDM, the robot dynamically switches from amongst a set of closed-loop policies adapting to different situations. A robot may *Follow* a person through a cluttered environment, or *Stop* in case of a commotion or high perceptual uncertainty.

Our original implementation of MPDM [20] chooses the policy with the lowest expected cost:

$$\pi^* = \arg\min_\pi E_{\boldsymbol{x}_0}\{C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)\}, \quad (2)$$

where the cost $C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)$ is associated with the current state $\boldsymbol{x}_0$ upon choosing a policy π.

Note that the cost function C is not only a function of the final state, but also all of the intermediate states through the transition function T_π. Therefore, $C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)$ is a highly nonlinear function of robot policy π, and an initial configuration $\boldsymbol{x}_0$ whose evaluation involves a time consuming forward propagation of the system. It might compute, for example, high costs for trajectories that lead to near-collisions.

The robot's behavior reflects not only the mean state estimates of the other agents, but also the uncertainty associated with those estimates. Estimation uncertainty and measurement noise affect the quality of sampled future trajectories and thereby system performance.

MPDM relies on quick re-planning in order to deal with uncertainty, which constrains the number of forward propagations that can be evaluated per candidate policy.

A. Cost Function

We consider the application where the robot is in a social, dynamic environment. The robot is assigned a goal point and it tries to reach the goal without inconveniencing people.

For the application domain described above, we developed a cost function with two components: *Blame* which captures the potential disturbance that the robot causes in the environment and *Progress* which indicates progress made towards the goal.

Blame: We use the distance to the closest agent as a proxy for the potential disturbance caused to the environment by the robot.

$$B\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big) = \sum_{k=0}^{s} \max_{j\neq r} \mathbf{u}(\|v_r\| - \epsilon) e^{-d_{r,j}(k)/\sigma} \qquad (3)$$

where $d_{r,j}(k)$ is the distance between the robot and agent j and $\|v_r(k)\|$ is the speed of the robot at time-step k. $\mathbf{u}$ is the step function which is 1 when the argument is ≥ 0 and 0 otherwise. If the robot is in motion ($\|v_r\| > \epsilon$), the decay rate σ determines how much proximity to other agents is penalized.

Progress: We reward the robot for the distance-made-good during the planning horizon.

$$PG\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big) = \big(p_r(s) - p_r(0)\big) \cdot \boldsymbol{e}_{p_r \to g_r}, \qquad (4)$$

where $p_r(k)$ is the position of the robot at time-step k and $\boldsymbol{e}_{p_r \to g_r}$ is the unit vector from the current position of the robot to the goal g_r .

The resultant cost function is a linear combination of both

$$C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big) = -\alpha PG\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big) + B\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big), \qquad (5)$$

where α is a weighting factor.

B. Sampling-based MPDM

In our previous work [20], we used Monte Carlo sampling from the estimator's posterior distribution $P(\boldsymbol{x}_0)$ to approximate the expected cost

$$E_{\boldsymbol{x}_0}\{C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)\} \sim \frac{1}{N}\sum_{n=1}^{N} C\big(\boldsymbol{X}(\pi,\boldsymbol{x}^n)\big), \qquad (6)$$

where $\{\boldsymbol{x}^1,\ldots,\boldsymbol{x}^N\}$ is the set of samples drawn from the distribution $P(\boldsymbol{x}_0)$.

The dimensionality of the space of all possible initial configurations is very large. Sampling is likely to miss high-cost events, even if they are individually reasonably probable (high probability density) because of the scarcity of such outcomes in the state space (low total probability mass of high-cost outcomes). In other words, collisions are profoundly serious, but near misses are mundane. Good planning performance, under these conditions, can require a prohibitive number of samples.

C. Proposed Approach

The key idea in this paper is that we explicitly search for influential outcomes (those that have high cost and probability) because they influence our decision making process the most. For instance, we may perturb the state elements of $\boldsymbol{x}_0$ while sampling in order to find high $C(\boldsymbol{x}_0)P(\boldsymbol{x}_0)$ outcomes. However, the samples used to search are no longer random samples from the posterior distribution and cannot be used to approximate the expectation according to Eq. 6.

We propose an anytime algorithm for discovering influential configurations through optimization of the following objective function

$$\max_{\boldsymbol{x}_0}\{P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)\}. \qquad (7)$$

This objective allows us to use optimization techniques; unlike sampling, it does not matter *how* we find the maximizing configuration.

One could set aside some samples for the optimization while the remaining samples could be used to calculate a more informed estimate of expectation by constructing a proposal distribution around the likely and dangerous configurations (importance sampling [21]). In this way, our method can be used to augment sampling. We do not explore this option in our experiments. Instead, we change our decision making rule as follows

$$\pi^* = \arg\min_{\pi} \big[\max_{\boldsymbol{x}_0}\{P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)\}\big]. \qquad (8)$$

Naturally, changing the objective function from Eq. 2 to Eq. 8 changes the emergent behavior of the planner, and the design of the cost function may need to be adjusted. In practice, though, we have had no difficulty adapting our systems to the proposed objective function. For safety-critical systems in which it makes sense to be risk-aware, using Eq. 8 may even be a more natural choice than minimizing the expected cost of an outcome.

V. Optimization

We now turn our attention to the problem of constructing informative outcomes (maximizing $P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)$) given a budget on the number of samples or forward simulations.

In our application, it is especially difficult to sample bad outcomes because we assume that all agents follow policies that tend to avoid collisions and dangerous scenarios in the first place ("unstable" points in some sense). The bad outcomes arise from a few initial configurations whose neighborhoods may be fairly uninteresting. In other words, the cost function tends to be tightly peaked around these configurations.

Unfortunately, we cannot rely on an analytical expression for the gradient of $P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi,\boldsymbol{x}_0)\big)$ since the transition function and thereby the cost function is non-differentiable. Also, the dimensionality of the space of all possible initial configurations is very large. Hence, gradient ascent requires many forward simulations (one for every dimension) just to compute the gradient numerically, enabling just one actual gradient step.

Using gradient-free local optimization techniques such as coordinate ascent is an alternative. Coordinate ascent randomly explores dimensions restrictively, perturbing them one at a time while keeping the other dimensions fixed (in contrast with random sampling which tries to change all the dimensions simultaneously). When a significant gradient is

found, it is exploited repeatedly. This restrictive exploration becomes problematic when the cost function depends on only a small subset of the observed agents. In these cases, coordinate ascent wastes many samples perturbing uninteresting dimensions during exploration before finding an improvement. In short, coordinate ascent is efficient when a large fraction of the dimensions are interesting. Empirically, we observe that coordinate ascent fails to outperform Monte-Carlo sampling over $P(\boldsymbol{x}_0)$, as we show in Sec. VI-A.

Rather than testing random perturbations to the initial configuration, we use information available in the predicted trajectories to find promising perturbations. We describe our approach below.

A. Heuristic Cost Function

The cost function $C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)$ captures the complex interactions between agents. This function is highly non-linear and non-differentiable with respect to the initial configuration $\boldsymbol{x}_0$. However, since collisions lead to high costs, we can use domain knowledge to propose a heuristic cost function

$$\tilde{C}(\boldsymbol{x}_0) = \sum_{i \in A} \sum_{k=0}^{s} \underbrace{v_{ir}(k) \cdot \hat{p}_{ir}(k)}_{\tilde{C}_i(k)} \tag{9}$$

where A is the set of agents, p_{ir} and v_{ir} are the velocity and position of agent i with respect to the robot. The operator $\hat{}$ indicates a unitary vector. The value of Eq. 9 is large when the trajectories of the agents are consistently aligned for a collision with the robot (see Fig. 2).

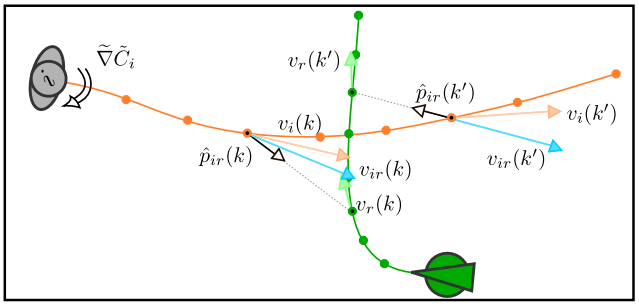

Fig. 2. Intuition for the heuristic cost function $\tilde{C}(\boldsymbol{x}_0)$. At time-step k, the robot is more likely to collide with the agent if the relative velocity $v_{ir}(k)$ aligns with the relative position $p_{ir}(k)$. The gradient of the initial velocity of agent i turns out to be $\sum_{k=0}^{s} \hat{p}_{ir}(k)$. Intuitively, this gradient promotes collision with the robot.

The gradient $\nabla\tilde{C}$ with respect to the initial conditions $\boldsymbol{x}_0$ corresponds to

$$\nabla\tilde{C} = \frac{\partial \tilde{C}}{\partial \boldsymbol{x}_0} = \sum_{i \in A} \sum_{k=0}^{s} \frac{\partial \tilde{C}_i(k)}{\partial \boldsymbol{x}_k} \frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_0}. \tag{10}$$

We approximate the gradient by ignoring the non-linearities making the assumption that perturbations in the initial state $\boldsymbol{x}_0$ reflect directly into $\boldsymbol{x}_k$ i.e. $\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{x}_0} \approx 1$. Thereby,

$$\nabla\tilde{C} = \frac{\partial \tilde{C}}{\partial \boldsymbol{x}_0} \simeq \sum_{i \in A} \underbrace{\sum_{k=0}^{s} \frac{\partial \tilde{C}_i(k)}{\partial \boldsymbol{x}_k}}_{\widetilde{\nabla}\tilde{C}_i} = \widetilde{\nabla}\tilde{C}, \tag{11}$$

where all states of the trajectory are contributing to modify equally the initial conditions $\boldsymbol{x}_0$ as if in a voting scheme. The main advantage of this approximation is that the gradient $\widetilde{\nabla}\tilde{C}$ can be computed directly from the trajectory.

B. Stochastic Gradient Ascent

With the series of approximations made in the previous section, we must remember that $\widetilde{\nabla}\tilde{C}(\boldsymbol{x}_0) \neq \nabla P(\boldsymbol{x}_0) C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)$. In order to limit divergence, we update only one agent at a time and limit the step size η of every update. In short, updating the configuration of all the agents at once according to $\widetilde{\nabla}\tilde{C}(\boldsymbol{x}_0)$ may even decrease the value of $P(\boldsymbol{x}_0) C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)$.

Algorithm 1 Stochastic Gradient Ascent

```
1:  function HillClimb(x_0, π)
2:      x ← x_0
3:  again:
4:      best, gradients ← Evaluate(x, π)
5:      while ∇C̃_i* =gradients.pop() do
6:          x' ← x + η∇C̃_i*
7:          probcost, — ← Evaluate(x', π)
8:          if probcost > best then
9:              x ← x'
10:             goto again
11:         end if
12:     end while
13:     return x
14: end function
15:
16: function Evaluate(x, π)
17:     X ← ForwardSimulate(x, π)
18:     utility ← P(x)C(X)
19:     gradients ← OrderedGradients(X)
20:     return utility, gradients
21: end function
```

Stochastic gradient ascent [22] is a popular technique to maximize complex objective functions consisting in summations, such as the one presented in Eq. 9. Consequently, $\widetilde{\nabla}\tilde{C}$ is composed of gradients for each agent $\widetilde{\nabla}\tilde{C}_i$ (Eq. 11). We view these components as promising modular perturbations. $\widetilde{\nabla}\tilde{C}$ is decomposed into an ordered list of stochastic gradients based on the proximity of the agent i to the robot along their trajectories

$$d_i^{min}(\boldsymbol{X}) = \min_{0 \leq k \leq s} \|p_{ir}(k)\|. \tag{12}$$

The sorted election (Alg. 1 line 5) is motivated by the fact that the nearest agents potentially have a higher impact on the resulting cost $C(\boldsymbol{x}_0)$.

We propose an iterative approach (Alg. 1) where we evaluate a perturbed solution according to $\widetilde{\nabla}\tilde{C}_{i^*}$ where i^* is the most promising agent (in the ordered list) whose gradient is unexplored. If it improves the objective function, the perturbation is retained and the ordered list is re-evaluated based on the new sample. If the update fails, the next gradient

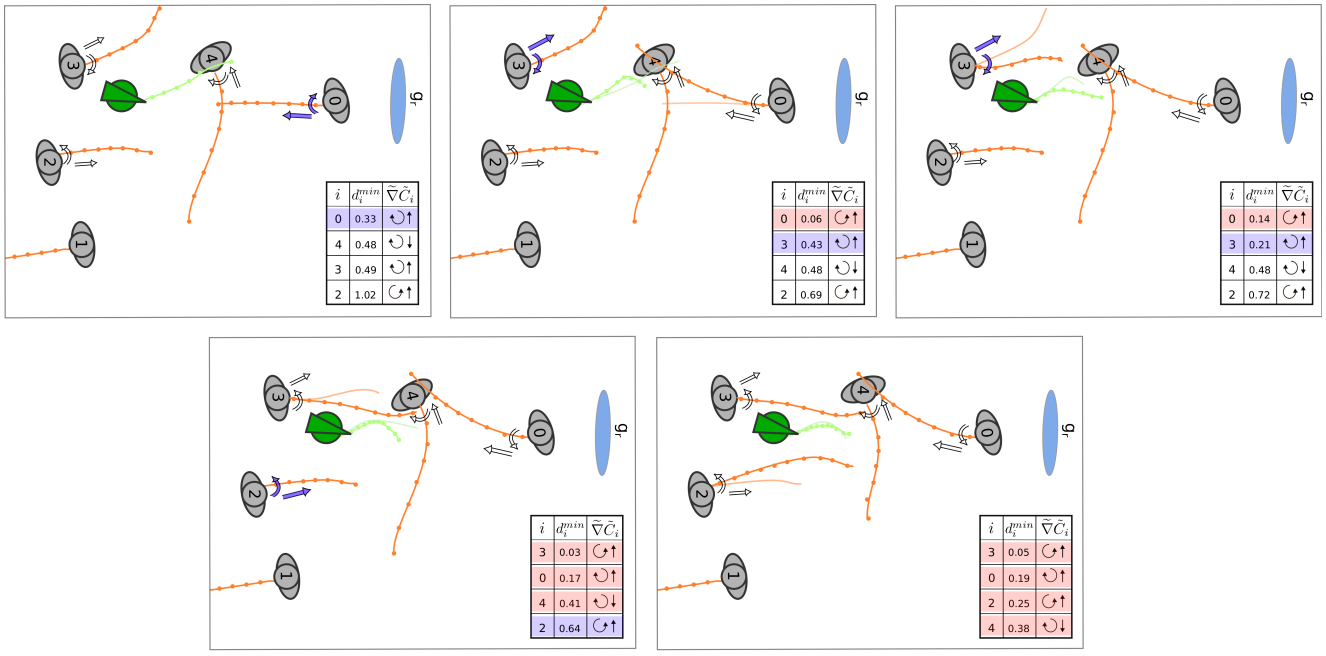

Fig. 3. An example demonstrating hill climbing (Alg. 1) for a given initial sample. *Top-Left*: Initial Sample. *Table*: Blue means that the gradient was evaluated and an improvement was observed. Red means that the gradient was evaluated but no improvement was observed. The above hill climbing takes 13 forward propagations or samples. Here we see that the search algorithm can capture important multi-agent configurations that are likely to arise. The arrows near the agents indicate the stochastic gradients $\widetilde{\nabla}\tilde{C}_i$. Note that these perturbations promote collision with the robot. The blue arrows indicate $\widetilde{\nabla}\tilde{C}_{i^*}$, the most promising gradient. In this example, the robot is following the *Go-Solo* policy towards its goal g_r.

is evaluated (line 5). The algorithm keeps performing updates until the sample budget is consumed or until all the gradients on the list have failed to update a new solution.

Figure 3 illustrates our procedure to order gradients, update trajectories, and evaluate improvements.

VI. Simulation Experiments

Our operating environment is an open space, freely traversed by a set of agents while the robot tries to reach a goal. The unconstrained nature of this domain makes the trajectories more dependent on initial configurations. Agents can randomly slow down or come to a stop. We set to $\alpha = 5$ using a procedure similar to [20] so that both *Blame* and *Progress* have more or less equal impact on the cost function. One simulation 'epoch' consists of a random initialization of agent states followed by a 5 minute simulated run at a granularity $\Delta t = 0.1$s. MPDM is carried out at 3Hz to match the frequency of the sensing pipeline for state estimation in the real-world experiment. The planning horizon is $3s$ into the future.

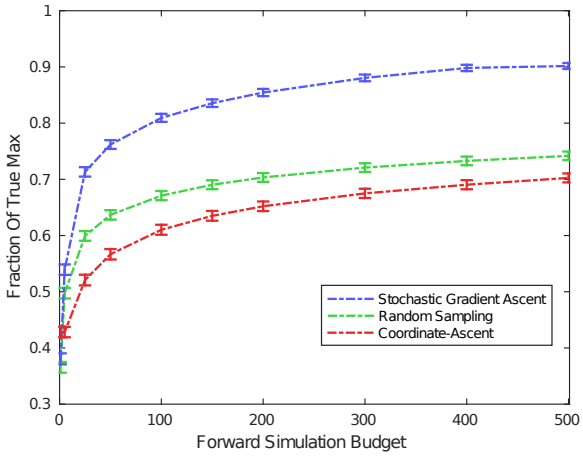

Fig. 4. Efficacy of search in estimating $\max P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi, \boldsymbol{x}_0)\big)$. Varying the number of samples available, we show here the mean and standard error of the fraction of the true maximum attained. The statistics are computed using 2k randomly generated scenarios from the atrium domain. Each scenario required at least one agent to be within 2m of the robot. The true maximum for a scenario was estimated from 50k random initializations.

A. Efficiency of Search

We tested the efficacy of the proposed algorithm in evaluating $\max P(\boldsymbol{x}_0)C(\boldsymbol{x}_0)$ on 2k random scenarios from the simulated domain. The scenarios were generated by running the simulation and capturing the state of the system if an agent is within 2m of the robot. To ensure independence, the scenarios were spaced at least 10 seconds apart.

For each scenario, we sample 50k initial configurations. Additionally, each of the local search algorithms (coordinate ascent and stochastic gradient ascent) were run 1k times and the objective function values for all the intermediate configurations are stored.

Varying the sample budget, the above datasets are bootstrap sampled (with replacement) 1k times in order to obtain the plot in Fig. 4. The 'true maximum' is the maximum function value over all the random samples, and all the local search samples. In short, each search algorithm is run 1k

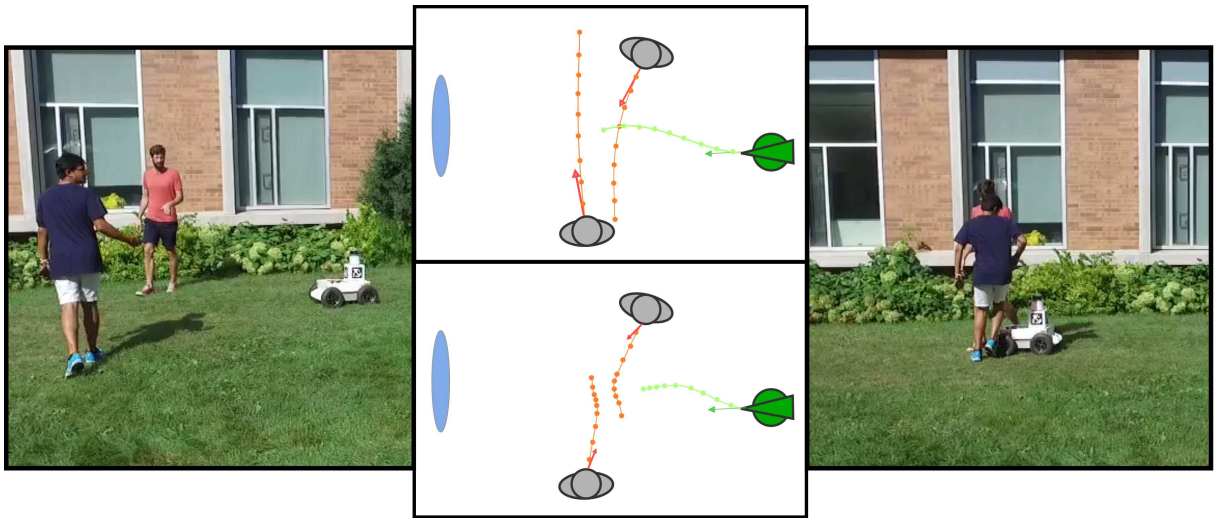

Fig. 5. Random sampling often results in poor decision making. If the two humans move away from each other (*Middle-Top*), the robot would be able to pass through without causing inconvenience. However, if the agents move too close to each other and interact in the path of the robot (*Middle-Bottom*), *Go-Solo* is no longer a good policy for the robot. Initial configurations that result in such interactions should ideally be accounted for while evaluating the utility of a policy (Go-Solo, in this case). Such configurations have high probability density but low probability mass and hence, Monte-Carlo sampling from a cost-function agnostic posterior distribution often fails to capture these situations. *Left*: A real world scenario where two agents are walking towards each other. *Middle-Top*: At their current speed and orientation, the agents would pass the robot and not obstruct it. This is mostly the case. *Middle-Bottom*: However, the adverse situation where the two agents head straight at each other is also probable but is unlikely to be sampled (it has high probability density but low probability mass). *Right*: The scenario 0.7s later. The robot decides to stop at the very last moment when the probability mass for the imminent adverse scenario is large.

times on each of the 2k scenarios and the mean and standard error of their performance is recorded.

Fig. 4 shows that random sampling fails to capture influential outcomes for complex scenarios and even increasing the sample budget does not help much. In short, we just cannot sample enough to guarantee that we find the bad outcomes.

Coordinate ascent explores the dimensions, but perturbs them one at a time while keeping the other dimensions fixed. In contrast, random sampling tries to change all the dimensions simultaneously. Coordinate ascent wastes a lot of samples perturbing non influential dimensions during exploration, when only a small subset of observed agents are truly affecting the cost, before finding an improvement. Due to wasteful sampling, empirically, coordinate ascent performs worse than random sampling for our domain.

We observe that the stochastic gradient search outperforms random sampling and can consistently improve performance with a bigger sample budget. Using only 50 samples, our proposed method performs comparably to random sampling with 500 samples. In this sense, we are about 10 times more efficient than random sampling in this domain.

This shows that it is possible to efficiently search for high-probability, high-cost configurations (influential outcomes).

VII. Real-World Experiments

We implemented our system on the MAGIC robot [23], a differential drive platform equipped with a Velodyne VLP-16 laser scanner used for tracking and localization. [1]

[1]We encourage the reader to see our video (`https://www.youtube.com/playlist?list=PLbPJN-se3-QiwIITl5cNsUV4-SRIyl9OM`) demonstrating the advantages of risk-aware MPDM.

An LED grid mounted on the head of the robot has been used to visually indicate the policy chosen at any time.

We use a laptop with an Intel i7 processor and 8GB RAM for our forward simulations and LCM [24] for inter-process communication. Every 333ms (policy election cycle), MPDM chooses a policy. This constrains the robot to a budget of $N = 50$ forward simulations per core used (we use only one core for the planning algorithm, although the task is readily parallelizable). Although the policy election is slow, the robot is responsive as the policies themselves run at over 100Hz.

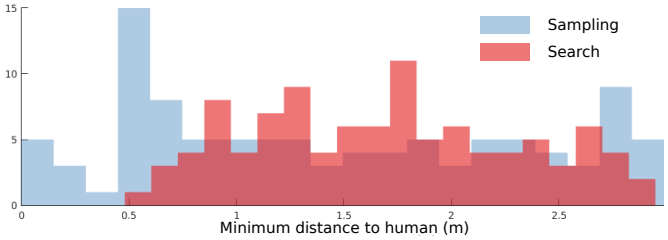

Fig. 6. Real-world results. The scenario described in Fig. 5 was repeated 20 times for each algorithm. The normalized histograms for the distance to the closest human the robot is moving towards. We notice that close calls and collisions are avoided by searching for likely high-cost outcomes. The sampling based approach collided with agent on 3 runs out of the 20 while our proposed approach never collided with another agent.

We ran two types of real-world experiments - one in which volunteers were asked to repeat the scenario described in Fig. 5 and the second in which six volunteers were asked to randomly move around in the open space for 30 minutes (Fig. 1). In both experiments, the planning algorithm being used was unknown to the participants by randomly switching

between the two approaches. Fig. 6 shows the normalized histograms for the distance to the closest human while the robot is navigating. While the Monte-Carlo sampling failed to consistently capture influential configurations resulting in an optimistic utility for the *Go-Solo* policy, our approach can consistently capture such configurations, making the robot stop when required.

VIII. Conclusions

We have presented a method that efficiently searches for likely configurations that have high cost and hence, influence on the decision making. We formulate a new objective function that captures this notion and propose an efficient anytime optimization algorithm.

We use stochastic gradient ascent on a heuristic cost function capturing collisions between the robot and agents to provide promising perturbations. At every update of the optimization process, we verify if the real cost function has been improved. Our proposed search technique outperforms other optimization methods such as Monte Carlo simulation or coordinate ascent.

We have also demonstrated that incorporating this new objective directly into MPDM improves performance of navigation significantly in simulated and real scenarios. We show that the system is more reliable, since it is able to systematically capture potentially dangerous outcomes that our former method was not able to consistently detect.

References

[1] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *Proc. IEEE Int. Conf. Robot. and Automation, Seattle, WA, USA*, 2015.

[2] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[3] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *European Conference on Mobile Robotics*, 2013, pp. 331–336.

[4] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Proceedings of the International Conference on Robotics and Automation*. IEEE, 1996, pp. 3375–3382.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[6] T. Schouwenaars, "Safe trajectory planning of autonomous vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[7] A. Bautin, L. Martinez-Gomez, and T. Fraichard, "Inevitable collision states: A probabilistic perspective," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4022–4027.

[8] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2012.

[9] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3931–3936.

[10] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, 2016.

[11] A. Bera and D. Manocha, "REACH-Realtime crowd tracking using a hybrid motion model," in *IEEE International Conference on Robotics and Automation*. IEEE, 2015, pp. 740–747.

[12] F. Large, D. Vasquez, T. Fraichard, and C. Laugier, "Avoiding cars and pedestrians using velocity obstacles and motion prediction," in *Intelligent Vehicles Symposium, 2004 IEEE*. IEEE, 2004, pp. 375–379.

[13] D. Vasquez, "Novel planning-based algorithms for human motion prediction," in *IEEE Conference on Robotics and Automation*, 2016.

[14] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Probabilistic motion planning among moving obstacles following typical motion patterns," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 4027–4033.

[15] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[16] G. Ferrer and A. Sanfeliu, "Multi-objective cost-to-go functions on robot navigation in dynamic environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3824–3829.

[17] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 454–460.

[18] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robotics and Autonomous Systems*, vol. 83, pp. 15–31, 2016.

[19] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[20] D. Mehta, G. Ferrer, and E. Olson, "Autonomous navigation in dynamic social environments using multi-policy decision making," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, p. to appear.

[21] S. M. Ross, *A course in simulation*. Prentice Hall PTR, 1990.

[22] S. Amari, "A theory of adaptive pattern classifiers," *IEEE Transactions on Electronic Computers*, no. 3, pp. 299–307, 1967.

[23] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and B. Marinier, "Progress toward multi-robot reconnaissance and the magic 2010 competition," *Journal of Field Robotics*, vol. 29, no. 5, pp. 762–792, 2012.

[24] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on*. IEEE, 2010, pp. 4057–4062.