2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Madrid, Spain, October 1-5, 2018

C-MPDM: Continuously-parameterized risk-aware MPDM by quickly discovering contextual policies

Dhanvin Mehta[1] Gonzalo Ferrer[2] Edwin Olson[1]

***Abstract*— Risk-aware Multi-Policy Decision Making (MPDM) is a powerful framework for reliable navigation in a dynamic social environment where rather than evaluating individual trajectories, a "library" of policies (reactive controllers) is evaluated by anticipating potentially dangerous future outcomes using an on-line forward roll-out process. There is a core tension in Multi-Policy Decision Making (MPDM) systems - it is desirable to add more policies to the system for flexibility in finding good policies, however, this increases computational cost. As a result, MPDM was limited to small (perhaps 5-10) discrete policies — a significant performance bottleneck.**

In this paper, we radically enhance the expressivity of MPDM by allowing policies to have continuous-valued parameters, while simultaneously satisfying real-time constraints by quickly discovering promising policy parameters through a novel iterative gradient-based algorithm. Our evaluation includes results from extensive simulation and real-world experiments in semi-crowded environments.

I. Introduction

Rather than evaluate individual trajectories, in Multi-Policy Decision Making (MPDM) [1], the robot navigates dynamic social environments by switching between a set of candidate policies (closed-loop reactive controllers) to adapt to different situations. For example, the robot may 'Follow' a pedestrian through a crowd or come to a 'Stop' in the case of mounting uncertainty. Quick decision making allows the robot to stay reactive to sudden and unexpected changes in the environment.

The robot must consider numerous possible future outcomes arising from the uncertainty associated with the inferred state of the agents (people) due to tracking errors and sensor noise, as well as the complex agent-agent interactions. Risk-aware MPDM evaluates each candidate policy (*ego-policy*) by anticipating potentially dangerous outcomes through an on-line optimization process based on forward roll-outs. While it is desirable to add more policies to increase the expressivity of the system, the aforementioned optimization is computationally expensive and only a handful of policies can be evaluated reliably in real-time. As a result, earlier MPDM systems were limited to a small finite set of candidate policies, which was a significant performance bottleneck.

C-MPDM radically enhances the expressivity of MPDM without increasing computational complexity. By allowing policies to have *continuous*-valued parameters, and then efficiently computing good values of those continuous parameters (Fig. 1), C-MPDM enables the robot to choose from an infinite number of policies in real-time.

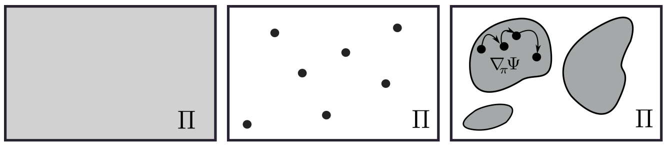

Fig. 1. Expressible policy spaces. *Left*: In principle, we would like to find the best policy from the entire space of policies Π; this is generally intractable. *Middle*: Earlier MPDM systems constrained the search to a finite number apriori-known discrete policies, whose size (perhaps 5-10) depends on the computational budget. *Right*: Continually-parameterized MPDM (C-MPDM) can represent much larger volumes within the policy space. By quickly generating promising context-derived candidate policies using "risk-aware policy-gradients" $\nabla_\pi\Psi$, C-MPDM increases expressivity of the actions available to the robot without increasing computational complexity.

Bilevel optimization is a well-studied class of mathematical programs encountered in various fields ranging from management [2], to optimal control [3] where there are two levels of optimization tasks, one nested within the other. In risk-aware MPDM, the upper-level optimizer (the ego-robot) chooses the policy with the most benign (low-cost) evaluation, while lower-level optimization (risk-aware policy evaluation of an ego-policy) involves finding the most potentially dangerous (likely, high-cost) outcome from all possible pedestrian configurations. In this way, risk-aware MPDM can be viewed as a bilevel optimization, but we will now consider an infinite number of ego-policies for the upper-level optimization.

Unfortunately, merely checking strict or local optimality in bilevel optimization problems is NP-hard and most academic research has been focused on simple bilevel programs with convex objective functions [4]. In our domain, even evaluating the cost function involves a time-consuming forward simulation. Furthermore, the high-dimensional search-space of possible robot policies and pedestrian configurations and the complex multi-agent interactions make the forward-simulated trajectories and thereby the cost function sensitive to the decision variables. Conflicting objectives and real-time requirements make this problem harder.

Our proposed algorithmic approach fundamentally rethinks the MPDM algorithm, replacing a core component with a method that more closely resembles a deep learning algorithm than a motion planner (though it is neither!). The key idea in this paper is to quickly generate promising context-derived candidate policies using a local iterative gradient-based search procedure (Fig. 1) where the necessary gradients are efficiently computed using backpropagation.

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

[2]G. Ferrer is with the CDISE department, Skolkovo Institute of Science and Technology, Moscow, Russia. g.ferrer@skoltech.ru

Promising policy candidates are then extensively evaluated. Our contributions include the following:

- We formulate C-MPDM as a bilevel optimization and allow policies to have continuous-valued parameters, which improve the expressivity and flexibility of the decision making process (Sec. III).
- We provide an effective real-time solution to the bilevel program. Our novel anytime algorithm finds increasingly desirable contextual ego-policy parameters (Sec. IV).
- Through extensive experiments in simulation and on a real robot platform, we demonstrate the benefits of C-MPDM over evaluating a fixed set of hand-crafted policies or evaluating randomly sampled policies.

II. Related Work

We first discuss previous approaches in the area of autonomous navigation in dynamic environments. In this work, a new set of tools– well-developed in other disciplines– is brought to bear on behavioral planning. We discuss other applications of bilevel optimization and gradient-based methods that have motivated our approach.

Recent motion-planning approaches for autonomous navigation in dynamic environments [5]–[7] augment the state of the system with the inferred states of other agents to model agent-agent interactions during planning which is required to deal with more complex situations. Learning-based approaches [8]–[10] use relevant features that might explain the interactions taking place between dynamic agents, but may be limited by the training datasets used.

POMDPs provide a powerful formulation for incorporating uncertainty into planning, but quickly become intractable. Recently, approximate POMDP methods based on scenario sampling and forward simulation have been applied to navigation [11] and mapping [12]. MPDM [1] is a constrained POMDP solver, in which the space of policies that can be evaluated is constrained by design and the ego-robot autonomously navigates by dynamically switching between these candidate policies. In this paper, we extend MPDM to continuous-valued policies, allowing the robot to choose from an infinite number of ego-policies in real-time.

With its roots tracing back to Stackelberg's game-theoretic modeling [13], bilevel optimization commonly appears in several practical applications such as environmental economics [14], chemical engineering [15] and operations research [16]. Risk-aware MPDM can be viewed as a bilevel optimization although previous MPDM systems [17] only considered apriori known discrete ego-policies, which reduced the bilevel problem to a series of single level optimizations. Unfortunately, bilevel programming is known to be strongly NP-hard [18]. Most practical methods find satisfactory, rather than optimal solutions by solving multiple simpler single level optimizations [19]. The idea behind C-MPDM is similar — promising candidates are quickly generated based on the current context and then evaluated extensively.

By representing a forward simulation as a deep neural network, we compute accurate gradients efficiently using backpropagation [20] and use an iterative gradient descent algorithm (gradient-descent in the first variable and gradient-ascent in the second one) similar to Ratliff et al. [21]. These iterative gradient-based methods, first introduced by Arrow, Hurwicz and Uzawa [22], have also been applied to distributed linear programming [23] and power networks [24].

III. Problem Formulation

For each observed agent i, the robot maintains a probabilistic estimate of its state - i.e. its position, velocity and inferred policy based on past observations of the pedestrians' positions[1]. The collective state $\boldsymbol{x}_t \in \mathcal{X}$ consists of the state of the robot and all observed agents at time t. Throughout the paper, we will refer to $\boldsymbol{x}_0$ as the collective state of all agents and the robot's state at the beginning of the planning cycle.

The robot's policy π_r (which we refer to as *ego-policy*) is an instantiation of a continuously-parameterized policy $\pi(v_{pref}, \psi_{g_r}) \in \Pi$ (similar to Chen et. al. [10]). The robot executing $\pi_r = \pi(v_{pref}, \psi_{g_r})$ tries to move at a preferred speed v_{pref}, while avoiding obstacles according to the Social Force Model [25]. Rather than heading straight towards its goal g_r, the robot tries to move towards a point to the left or right of the goal, as determined by the direction offset parameter ψ_{g_r}. For example, $\pi_r = \pi(0.5, 30°)$ implies a policy where the robot tries to move at 0.5m/s at an orientation 30° to the right of the goal. For the sake of clarity, we do not explicitly refer to parameters of an ego-policy π_r. Gradients are expressed w. r. t. π_r, although we are implicitly referring to its parameters.

For an ego-policy π_r, an initial sampled configuration $\boldsymbol{x}_0$ is forward simulated H time-steps (through $t = 1, \ldots, H$), by recursively applying the transition operator $T : \mathcal{X} \to \mathcal{X}$ to yield a trajectory

$$\begin{aligned}\boldsymbol{X}(\boldsymbol{x}_0, \pi_r) &= \{\boldsymbol{x}_0, T(\boldsymbol{x}_0), T^2(\boldsymbol{x}_0), \ldots, T^H(\boldsymbol{x}_0)\} \\ &= \{\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_H\},\end{aligned}$$

where $\boldsymbol{x}_t \in \mathcal{X}$ is the collective state comprising of the state of the robot plus all the agents at time t of the forward simulation. The operator $T()$ captures the policy that each agent is executing while at the same time considering the interactions with all other agents. The cost function $C\big(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r)\big)$ assigns a scalar value to the outcome of a forward simulation.

Like our earlier work [17], our transition function is governed by the Social Force Model [26] capturing agent-agent interactions and agent kinematic models. Our cost function penalizes the inconvenience the robot causes to other agents in the environment (*Blame*) along the predicted trajectory and rewards the robot's progress towards its goal (*Progress*).

[1]Several methods can be used for obtaining this posterior. We use a Kalman Filter to infer position and velocity and a Naive Bayes Classifier to infer an agent's policy parameters.

Note that the MPDM framework is largely agnostic to the chosen transition function and cost function.

In Risk-aware MPDM, each ego-policy π_r is evaluated based on the most influential (likely high-cost) forward simulated configuration $\Psi(\pi_r)$, which is discovered by optimizing a probabilistic cost surface -

$$\Psi(\pi_r) = \underset{\boldsymbol{x}_0 \in \mathcal{X}}{\arg\max}\{P(\boldsymbol{x}_0)C(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0))\}. \tag{1}$$

During each planning cycle, the ego-policy with the most benign influential outcome is chosen and executed. Thus, policy election for risk-aware MPDM can be modeled through the following bilevel program -

$$\begin{aligned} &\min_{\pi_r \in \Pi} P(\boldsymbol{x}_0)C(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)) \\ &\quad \text{s.t. } \boldsymbol{x}_0 = \underset{\boldsymbol{x}_0 \in \mathcal{X}}{\arg\max}\{P(\boldsymbol{x}_0)C(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0))\} \\ &\iff \boldsymbol{x}_0 = \Psi(\pi_r). \end{aligned} \tag{2}$$

Algorithm 1 Policy Election with Handcrafted Policies

```
1:  function POLICY-ELECTION LOOP($P(\boldsymbol{x}), \Pi_d, N_\pi$)
2:      for $\boxed{\pi \in \Pi_d}$ do
3:          Initialize $U_\pi, n \leftarrow 0$
4:          while $n < \frac{N_\pi}{|\Pi_d|}$ do
5:              Sample $\boldsymbol{x}_0 \sim P(\boldsymbol{x})$
6:              $U^*, n_{opt} \leftarrow$ Optimize-Env($\boldsymbol{x}_0, \pi$)
7:              $n \leftarrow n + n_{opt}$
8:              $U_\pi \leftarrow \max\{U^*, U_\pi\}$
9:          end while
10:     end for
11:     $\pi^* \leftarrow \arg\min_\pi U_\pi$
12: end function
```

Algorithm 1 describes earlier approaches to policy-election. Provided with a probability distribution over initial configurations, $P(\boldsymbol{x}_0)$, an apriori-known discrete set of candidate policies, Π_d, and a forward simulation budget, N_π, each candidate policy evaluated (scored) independently (Line 2) according to the most influential (worst-case) outcome discovered within the computational budget and the policy with the most benign influential outcome is elected.

Unfortunately, only a handful of ego-policies can be reliably evaluated in real-time, limiting Alg. 1 to a small number of discrete policies — a significant performance bottleneck. In the next section, we extend the risk-aware MPDM framework by allowing policies to have continuous-valued parameters.

IV. Continuous Risk-Aware MPDM

In our bilevel program (Eqn. 2), even the lower level optimization (computing $\Psi(\pi_r)$ exactly for fixed ego-policy) is computationally infeasible due to the large space of possible initial pedestrian configurations [17]. The conflicting objectives, the real-time requirements and the lack of a closed-form expression for the objective function make this problem harder.

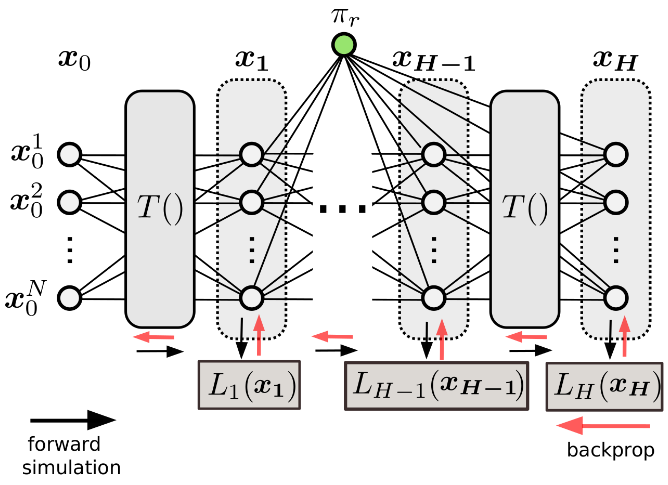

Fig. 2. A deep network representation for our cost function. For an ego-policy π_r, the initial configuration $\boldsymbol{x}_0$ is propagated through several layers of the transition function T (detailed in [17]). The output of layer t determines a cost for a single time-step $L_t(\boldsymbol{x}_t)$. Our cost function $C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r))$ accumulates costs calculated at each time-step along the forward simulated trajectory. Our representation enables quick computation of accurate gradients for the bilevel optimization.

We propose a novel approach where accurate gradients are computed efficiently through backpropagation and these gradients are used to dynamically generate promising contextual candidate policies in real-time. Our anytime candidate generation algorithm produces increasingly desirable policies for the robot to execute.

A. Accurate Gradients for discovering influential configurations $\boldsymbol{x}_0$

Deep neural networks chain (compose) relatively simple functions such as convolutions or sigmoids to model complex functions. In the same way, a forward simulation for H time-steps captures the complex dynamics of the system using simple one-step transition functions T. We want to penalize trajectories that have bad interactions at any point during the forward roll-out, not just those that *end* in a bad interaction. Therefore, unlike most deep networks, the cost function $C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r))$ is not a simple loss computed at the final output, but rather costs accumulated at each time step $L_t(\boldsymbol{x}_t)$.

We define a partial cost function $\mathcal{L}(t, \boldsymbol{X})$ that accumulates costs along the trajectory from time $\tau = t \dots H$ (until the end of the roll-out).

$$\mathcal{L}(t, \boldsymbol{X}) = \sum_{\tau=t}^{H} L_\tau(\boldsymbol{x}_\tau). \tag{3}$$

Accurate gradients of the cost function can be computed efficiently by backpropagation through deep network, which leverages the following recurrence relation -

$$\begin{aligned} \nabla_{\boldsymbol{x}_t}\mathcal{L}(t, \boldsymbol{X}) &= \frac{\partial \mathcal{L}(t, \boldsymbol{X})}{\partial \boldsymbol{x}_t} = \frac{\partial\{\mathcal{L}(t+1, \boldsymbol{X}) + L_t(\boldsymbol{x}_t)\}}{\partial \boldsymbol{x}_t} \\ &= \frac{\partial \mathcal{L}(t+1, \boldsymbol{X})}{\partial \boldsymbol{x}_{t+1}} \frac{\partial T(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} + \frac{\partial L_t(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t}. \end{aligned} \tag{4}$$

The gradient of the cost function with respect to the agent configurations $\nabla_{\boldsymbol{x}_0} C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r)) = \nabla_{\boldsymbol{x}_0}\mathcal{L}(0, \boldsymbol{X})$ is used during gradient-ascent to simultaneously perturb the configurations of all pedestrians towards increasingly likely and high-cost outcomes until convergence as follows -

$$\boldsymbol{x}_0 = \boldsymbol{x}_0 + \eta_1\big(\frac{\nabla_{\boldsymbol{x}_0} C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r))}{C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r))} + \frac{\nabla_{\boldsymbol{x}_0} P(\boldsymbol{x}_0)}{P(\boldsymbol{x}_0)}\big). \quad (5)$$

Each agent's update rate is determined using line-search along the gradient direction. Note that Eqn. 5 locally optimizes $\log\big(P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi_r, \boldsymbol{x}_0)\big)\big)$. We refer to this procedure as Optimize-Env.

B. Bilevel optimization by generating contextual risk-aware policies $\boldsymbol{\pi}_r$

This section details the main contribution of this paper — a technique for quickly discovering effective contextual ego-policy parameters (boxed in Alg. 2). A key insight is that in addition to discovering influential outcomes, we can use the same backpropagation machinery to compute the gradient of the cost function with respect to the ego-policy parameters -

$$\nabla_{\pi_r} C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r)) = \sum_{t=1}^{H} \frac{\partial \mathcal{L}(t, \boldsymbol{X})}{\partial \boldsymbol{x}_t}\frac{\partial \boldsymbol{x}_t}{\partial \pi_r}.$$

We perform gradient-descent, perturbing the ego-policy π_r towards increasingly benign parameters until convergence as follows -

$$\pi_r = \pi_r - \eta_2 \frac{\nabla_{\pi_r} C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r))}{C(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r))}. \quad (6)$$

This gradient-descent procedure, which we call Optimize-Robot, locally optimizes $\log\big(C\big(\boldsymbol{X}(\boldsymbol{x}_0, \pi_r)\big)\big)$.

Since finding the global optimum for the bilevel program (Eqn. 2) is computationally intractable, we first generate promising contextual candidate policies using a greedy local search procedure (POLICY-GENERATION) and then evaluate these candidates extensively (POLICY-EVALUATION).

Algorithm 2 summarizes the overall policy election cycle. C-MPDM elects an ego-policy, taking as input a probability distribution over initial pedestrian configurations, $P(\boldsymbol{x}_0)$, a continuous ego-policy space, Π, and forward simulation budgets for candidate generation (N_{cg}) and policy evaluation (N_π). A pedestrian configuration $\boldsymbol{x}_0$ sampled from $P(\boldsymbol{x}_0)$ and an ego-policy π_r sampled from Π (Line 3), seed the POLICY-GENERATION procedure (Line 4) which discovers promising policy parameters in the current navigation context. Promising policies π_r^* are extensively evaluated (Line 8) and the policy with the most benign influential outcome is executed (Line 10).

POLICY-GENERATION is an iterative gradient-based process that converges at a saddle point where the robot's policy and the pedestrian configuration are both locally optimal as illustrated in Fig. 3. First, Optimize-Env (Line 15) perturbs the seed pedestrian configuration $\boldsymbol{x}_0$ towards the locally most influential outcome which determines the score U_0 of the seed ego-policy π_r. Then, the seed ego-policy is repeatedly perturbed and evaluated locally - at each iteration,

Algorithm 2 Policy Election for Continuous Policy Spaces

```
 1: function C-MPDM($P(\boldsymbol{x}), \Pi, N_{cg}, N_\pi$)
 2:     while Time Remaining do
 3:         Sample $\boldsymbol{x}_0 \sim P(\boldsymbol{x}),\ \pi_r \sim \Pi$
 4:         $\pi_r^*, U_\pi, U_0 \leftarrow$ Policy-Generation($\pi_r, \boldsymbol{x}_0, N_{cg}$)
 5:         if $U_\pi - U_0 \le \delta$ then
 6:             continue
 7:         end if
 8:         $U_\pi \leftarrow$ Policy-Evaluation($\pi_r, U_\pi, N_\pi$)
 9:     end while
10:     return $\arg\min_\pi U_\pi$
11: end function
12:
13: function $\boxed{\text{POLICY-GENERATION}}$ ($\pi_r, U_\pi, N_{cg}$)
14:     Initialize $U_\pi, n \leftarrow 0$
15:     $\boldsymbol{x}_0, n_{opt} \leftarrow$ Optimize-Env($\boldsymbol{x}_0, \pi_r$)
16:     $n \leftarrow n + n_{opt}$
17:     $U_0 \leftarrow P(\boldsymbol{x}_0)C(\boldsymbol{x}_0, \pi_r)$
18:     while $n < N_{cg}$ do
19:         $\tilde{\pi}_r, n_{opt} \leftarrow$ Optimize-Robot($\boldsymbol{x}_0, \pi_r$)
20:         $n \leftarrow n + n_{opt}$
21:         $\boldsymbol{x}_0, n_{opt} \leftarrow$ Optimize-Env($\boldsymbol{x}_0, \tilde{\pi}_r$)
22:         $n \leftarrow n + n_{opt}$
23:         $U_i \leftarrow P(\boldsymbol{x}_0)C(\boldsymbol{x}_0, \tilde{\pi}_r)$
24:         if $U_i \le U_{i-1} - \epsilon$ then
25:             $\pi_r \leftarrow \tilde{\pi}_r$
26:             $U_\pi \leftarrow \mathcal{U}_i$
27:         else
28:             break
29:         end if
30:     end while
31:     return $\pi_r, U_\pi, U_0$
32: end function
33:
34: function POLICY-EVALUATION($\pi_r, U_\pi, N_\pi$)
35:     while $n < N_\pi$ do
36:         Sample $\boldsymbol{x}_0 \sim P(\boldsymbol{x})$
37:         $U^*, n_{opt} \leftarrow$ Optimize-Env($\boldsymbol{x}_0, \pi_r$)
38:         $n \leftarrow n + n_{opt}$
39:         $U_\pi \leftarrow \max\{U^*, U_\pi\}$
40:     end while
41:     return $U_\pi$
42: end function
```

Optimize-Robot (Line 19) perturbs the ego-policy π_r towards more benign parameters using n_{opt} gradient-descent steps (Eqn. 6) and the perturbed policy is evaluated according to the locally most influential outcome (Line 21). If the forward simulation budget for candidate generation N_{cg} is exceeded (Line 18) or if ego-policy perturbation does not improve the worst-case outcome by ϵ (Line 24), the search terminates. If perturbations in the ego-policy result in a significantly more benign (locally) worst-case outcome than the seed ego-policy (Line 5), we assume that π_r^* is likely to be promising.

The POLICY-EVALUATION function evaluates π_r^* more

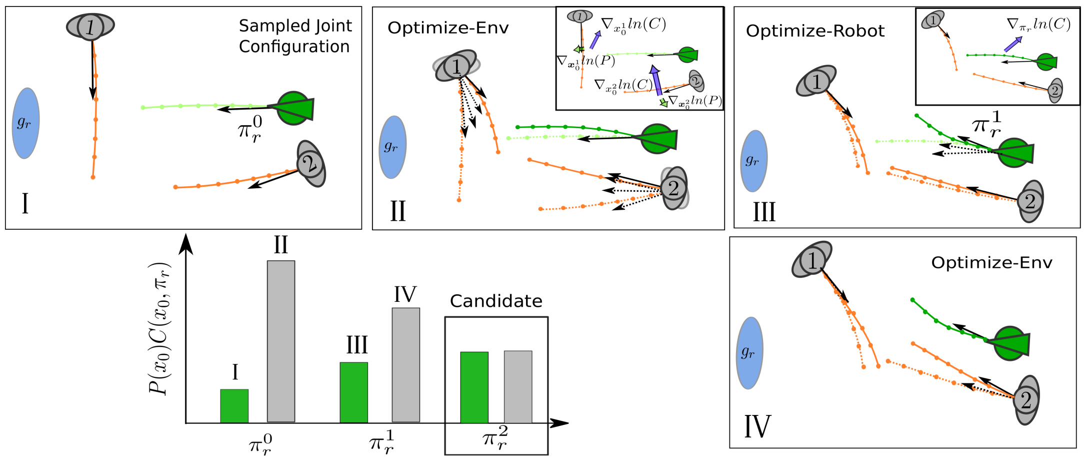

Fig. 3. Candidate ego-policy generation through iterative gradient-based optimization (Alg. 2 - POLICY-GENERATION). The forward propagated outcome of a randomly sampled joint configuration (tile I) is not discouraging for the robot as its ego-policy π_r^0 does not inconvenience either agent. For agent $i \in \{1,2\}$, the gradients computed using backpropagation $\nabla_{\boldsymbol{x}_0^i} ln\big(C(\boldsymbol{X})\big)$ (Blue) drives the agent towards configurations where the robot would cause inconvenience under its current policy while $\nabla_{\boldsymbol{x}_0^i} ln\big(P(\boldsymbol{x}_0)\big)$ (Green) drives it towards a more likely configuration (tile II inset). The agent configurations are simultaneously updated (dotted black arrows) according to Eqn. 5 until convergence, resulting in a likely high-cost outcome (tile II) that is used to evaluate π_r^0. The bar plot (*Bottom-Right*) shows the corresponding rise in the objective function $P(\boldsymbol{x}_0)C\big(\boldsymbol{X}(\pi_r^0,\boldsymbol{x}_0)\big)$. Once π_r^0 is evaluated, $-\nabla_{\pi_r} ln\big(C(\boldsymbol{X})\big)$ is computed (tile III inset) through backpropagation (Blue), and the ego-policy is updated according to Eqn. 6 until convergence, resulting in a more benign robot policy π_r^1 with a lower objective function value. The newly discovered ego-policy π_r^1 is then evaluated once again perturbing the agent configurations to discover the locally most influential outcome for π_r^1 (tile IV). The bar plot (*Bottom-Right*) shows that even though the outcome in tile III for π_r^1 has higher cost than the outcome (tile I) for π_r^0, the worst-case outcome for π_r^1 (tile IV) is more benign than that for π_r^0 (tile II). After one more iteration, our approach converges to a saddle point, yielding a candidate ego-policy π_r^2.

extensively by performing local searches for influential outcomes from different seed pedestrian configurations (Line 36) using at most N_π forward simulations. Otherwise, a new joint configuration is sampled and the procedure repeats.

V. EXPERIMENTS

Our operating environment consists of an open area that is freely traversed by a set of pedestrians that randomly change speed and direction without signaling while the robot tries to reach its goal. The unconstrained nature of this domain makes the trajectories more sensitive to the initial pedestrian configurations as well as the robot's ego-policy. By re-planning quickly (every 300ms), the robot is able to react to sudden and unexpected changes in the environment.

For each observed agent i, the robot maintains a probabilistic estimate of its state. A sudden change in speed is accounted for by assuming a distribution over the preferred speed v_{pref} of each agent that is a mixture of two truncated Gaussians - one centered around the estimated most-likely current speed with a $\sigma = 0.4m/s$ to account for speeding up or slowing down and a truncated half Gaussian with a peak at 0 and $\sigma = 0.2m/s$ to account for coming to a sudden stop. Pedestrian i's sub-goal g_i is inferred from a set of salient points in the environment using a Naive Bayes classifier. However, the pedestrian can suddenly change direction without signaling which is captured by assuming a Gaussian distribution for the pedestrian's short-term heading ψ_{g_i} that is centered around the agent's estimated most-likely orientation and $\sigma = 30°$. All truncated Gaussians are restricted to $\mu \pm 1.5\sigma$.

High-cost outcomes correspond to those where the robot inconveniences other agents by driving too close to them, thus accumulating high *Blame*. The robot is also rewarded according to the *Progress* it makes towards the goal. These cost-metrics are defined in greater detail in previous MPDM systems [27].

A. Simulation Experiments

Our simulation experiments are run on an Intel i7 processor and 8GB RAM to mimic the computational capabilities of our robot platform. We have used the same simulation environment that was used in our previous work [17]. We demonstrate that our proposed approach discovers more desirable ego-policies than random policy sampling. Moreover, we show that C-MPDM outperforms previous approaches in finding safe, yet effective policies in real-time.

1) Efficiency of Candidate Generation: We generated a dataset consisting of 4k randomly chosen simulated scenarios, each of which has at least one agent present within 5m of the robot. For each scenario, we estimate $\max\limits_{\pi_r \in \Pi} \Psi(\pi_r)$, the optimal solution to the bilevel optimization (Eqn. 2) by evaluating a large number of policies $\tilde{\Pi}$, consisting of 1k randomly sampled ego-policies as well as 500 context-aware policies generated using our proposed approach (Alg. 2). For each policy, $\Psi(\pi_r)$ was estimated using the POLICY-EVALUATION function with a forward simulation budget $N_\pi = 100$. This brute force estimation of the "globally optimal" ego-policy takes about 5 minutes for each scenario (three orders of magnitude slower than our real-time requirements).

Given real-time constraints, only a handful of ego-policies can be evaluated reliably. For a particular scenario in the dataset, we define the *Performance Ratio* of N candidate ego-policies $\Pi_{cand} = \{\pi_r^i\}_{i=1}^N$ the ratio of "globally optimal" ego-policy's utility to the utility of the most benign ego-

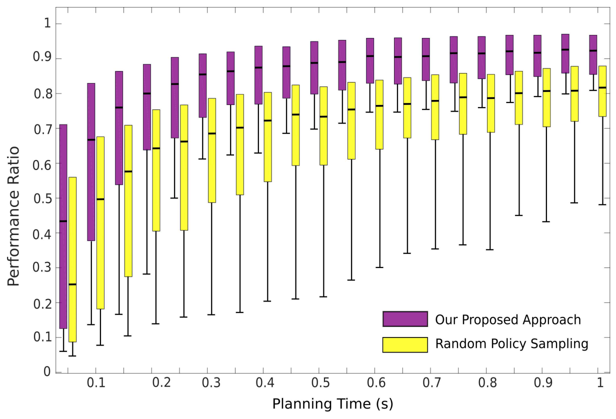

Fig. 4. Varying the planning time, we evaluate the efficacy of our proposed method in estimating $\min_{\pi_r \in \Pi} \Psi(\pi_r)$ (a bilevel optimization). We compare the distribution of the *Performance Ratio* of 400k candidate sets generated using our approach with those generated by random policy sampling. The bars mark the median and quartiles of the data-points while the lower whisker represents the 10^{th} percentile. A *Performance Ratio* of 1 indicates optimality, while candidates with lower *Performance Ratios* would cause the robot to stop unnecessarily or pick a sub-optimal policy. Random policy sampling often fails to find desirable ego-policy parameters even with impractical planning times of $1s$. Our method significantly outperforms random policy sampling over the entire range of *Planning Time*.

policy in the generated candidate set -

$$\text{Performance Ratio}\,(\Pi_{cand}) = \frac{\min\limits_{\pi_r \in \tilde{\Pi}} \Psi(\pi_r)}{\min\limits_{\pi_r \in \Pi_{cand}} \Psi(\pi_r^i)}.$$

A *Performance Ratio* of 1 is ideal and smaller ratios imply poorer candidates.

Varying the time available for planning t_p from $50ms$ to $1s$, we compare the *Performance Ratio* of 400k candidate sets of randomly sampled ego-policies with context-derived candidate ego-policies generated using our proposed approach. As more planning time is available, more candidate policies can be evaluated and in general, the *Performance Ratio* increases. However, in order to stay reactive to sudden changes in the environment, t_p should not exceed $400ms$ (based on experiments on our physical robot platform).

For each scenario, candidates are bootstrap sampled from the dataset and their *Performance Ratio* is represented by box-plots in Fig. 4. The boxes represent the quartiles while the bar represents the 10^{th} percentile. We observe that our method significantly outperforms random policy sampling over the entire range of *Planning Time*. Within real-time constraints ($300 - 400ms$), while random policy sampling is highly unreliable (long 10^{th} percentile bars), our iterative gradient-based approach the elected policy is almost always within a factor of two of the optimal ego-policy (99.9% of the times) which is better than random policy sampling with $1s$ of (impractical) planning time.

2) C-MPDM system validation: Through 10 hours of autonomous navigation in our simulated environment, we demonstrate that the enhanced expressivity provided by C-MPDM results in significant performance improvements. Each simulation 'epoch' consists of a random initialization of agent states followed by a 5 minute simulated run at a

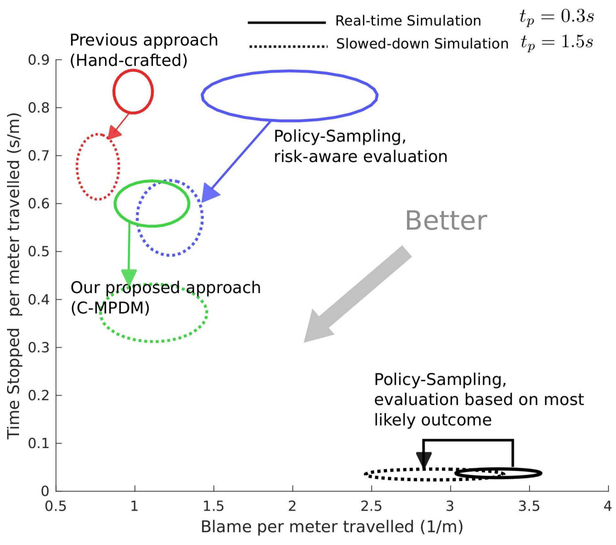

Fig. 5. Our proposed method, C-MPDM can find good policies more often, even other where methods cannot. We compare the performance of various algorithms on 10 hours of navigation in our simulated environment. We measure the *Time Stopped* for every goal reached as well as the *Blame per meter traveled* by the robot. For each algorithm, we use bootstrap sampling to estimate the mean and standard error for these metrics, represented by the axes of an ellipse. Smaller values of *Blame* and *Time Stopped* are better. Note that in practice, a planning time $t_p < 0.4s$ is required for responsive behavior $t_p = 1.5s$ is impractical for our application. Without risk-aware evaluation, even densely sampling the policy space fails to anticipate potentially dangerous outcomes and incurs high *Blame*. Upon running the simulator in real-time with a planning time $t_p = 0.3s$ (solid ellipses), we observe that our proposed method (C-MPDM) outperforms both, risk-aware policy evaluation with random samples and with hand-crafted ego-policy parameters. Upon slowing down the simulator in order to allow an impractical planning time $t_p = 1.5s$ (dashed ellipses), all the algorithms perform better. Still, C-MPDM outperforms both policy-sampling and Hand-crafted policies.

granularity $\Delta t = 0.15$s. During each policy-election cycle, the simulator was perturbed to account for sensor noise and tracking uncertainty.

We record the *Time Stopped* as well as the *Blame* normalized by the distance to goal. *Time Stopped* indicates the failure of the planner to find a safe policy. With a larger policy set, the robot is more likely to find a safe policy, and *Stops* less often. However, if the robot cannot evaluate its policy set quickly enough, it is unable to react to sudden changes in the environment and accumulates *Blame*. Ideally we would like a robot to navigate safely (low *Blame*), with minimal Stop-and-Go motion.

We run the simulator both in real-time ($t_p = 0.3s$), as well as slowed-down to allow an unrealistic planning time ($t_p = 1.5s$). We compare the performance of our proposed approach C-MPDM, which allows for continuous-valued parameterized policies with the following alternatives for generating discrete policies -

1) *Hand-crafted Candidates (HC)* - A set of hand-crafted candidate policies $\Pi_{hc} = \{$*(Fast, Medium, Slow)*$\times$*(Straight, Left, Right), Stop, Follow-other*$\}$ - used in previous MPDM-systems [17] is provided. Each candidate is independently evaluated as in Alg. 1. Rather than going straight towards the goal at maximum speed ($1.6m/s$), the robot may also choose to

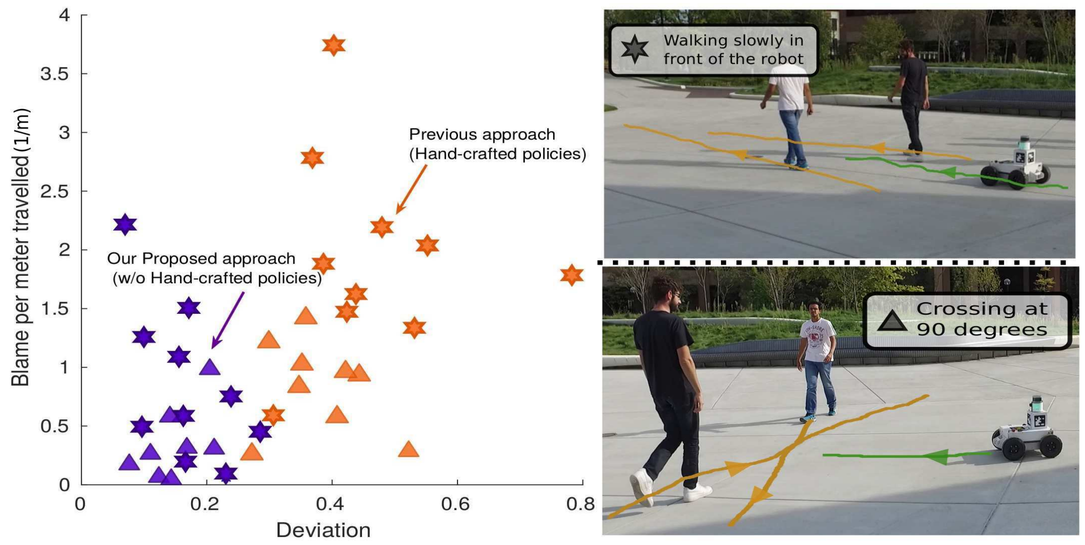

Fig. 6. Repeated real-world experiments. Data is collected from two repeatable experiments represented by different symbols. 1) Pedestrians crossing the robot's trajectory orthogonally (Δ) and 2) Pedestrians walking slowly in front of the robot (star). We measure the trajectory *Deviation* as well as the *Blame per meter traveled* by the robot. Lower the *Deviation* and *Blame*, the better. Our proposed approach C-MPDM allows the robot to modulate policy parameters appropriately so as to achieve low *Deviation* while avoiding close encounters.

go at *Medium* speed ($0.9m/s$) or *Slowly* ($0.2m/s$). Simultaneously, the robot can also choose to create a sub-goal to the *Left* or *Right* of the goal. Additionally, the robot can choose to follow any nearby agent. Due to the open nature of our domain, the *Follow-other* policies have the lowest priority (they are evaluated only if time permits after evaluating the others).

2) *Policy Sampling with Risk-Aware evaluation (PS-RA)* - Given a continuous policy space Π, (without a set of hand-crafted policies), a set of policies can be randomly sampled from Π and evaluated in a risk-aware fashion.
3) *Policy Sampling without Risk-Aware evaluation (PS-ML)* - Since risk-aware policy evaluation is computationally expensive, an alternate approach is to densely sample the policy space Π and evaluate policies very quickly only based on the most-likely scenario. In our experiments, around 150-200 policies could be evaluated for a planning time of $0.3s$.

For each of the four planning algorithms, Fig. 5 shows the mean and standard error for the *Time Stopped* and the *Blame per meter traveled* by the robot. Under real-time constraints, we observe that our proposed method (C-MPDM) is able to discover effective ego-policy parameters more often than both, risk-aware policy evaluation with randomly sampled (PS-RA) as well as with hand-crafted (HC) ego-policy parameters and *Stops* less frequently. On the other hand, dense policy sampling (PS-ML) does not account for the uncertainty while evaluating a policy and accumulates much higher *Blame* as it fails to anticipate potentially dangerous outcomes and *Stops* only when collision is imminent. Upon slowing down the simulator to 5x slower than real-time constraints would allow, sampling performs much better. The hand-crafted *Follow-other* policies can be evaluated with $t_p = 1.5s$ and hence, with more candidates, HC *Stops* less often. However, C-MPDM still offers much more flexibility to the decision making process and outperforms both HC and PS-RA.

B. Real-World Experiments

We implemented our system on a differential drive robot equipped with a Velodyne VLP-16 laser scanner used for tracking pedestrians as well as for localization. Every 300ms, MPDM evaluates a set of policies and chooses the least risky one. Although the policy election is slow, the robot is responsive as the policies themselves run at 50Hz. Fig. 6 shows data from 75 minutes of repeatable real-world experiments where volunteers were asked to repeat fixed scenarios while the robot made its way towards its goal about $25m$ away.

We compared C-MPDM with hand-crafted discrete policies (HC) based on the *Blame per meter traveled* as well as the trajectory *Deviation* which we define as the ratio of the extra distance traveled by the robot to the minimum (straight line) distance that the robot would travel if there were no pedestrians. Using the 10 hand-crafted policies (the planning time was insufficient to evaluate the *Follow-other* policies), the robot often finds sub-optimal policies resulting in larger trajectory *Deviation*. Our proposed method C-MPDM (purple) is able to adjust its speed and direction at a much higher resolution and as a result, finds policies that result in lower *Blame* and *Deviation*.

In another real-world experiment, seven volunteers were asked to move between marked points around an open space for 45 minutes. On several occasions, the volunteers were adversarial towards the robot, trying to test its capabilities by suddenly changing direction, blocking its path or jumping in front of it. We encourage the reader to see our attached video demonstrating emergent behavior using C-MPDM. (`https://goo.gl/WgXW55`)

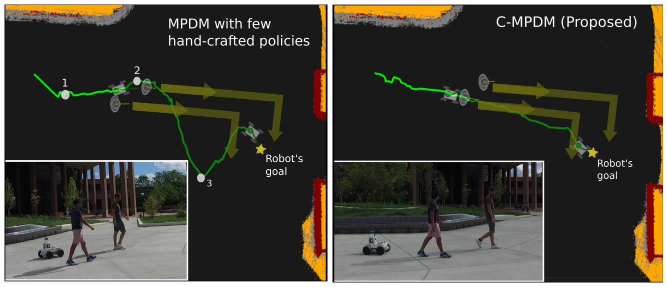

Fig. 7. Robot trajectories from a real-world experiment. Two agents walk closely in front of the robot as it navigates towards its goal. The agents arbitrarily change speed and suddenly turn acutely near the robot's goal (star). The arrows mark the general trajectory of the pedestrians. Salient locations are marked by circles. The undesirable behavior from MPDM with hand-crafted policies (Π_{hc}) stems from switching between very distinct policies (*Left*). At location 1, the MPDM chooses to overtake the agents at a High speed towards the Left of the goal (see Sec. V-A.2). However, the pedestrians also speed up slightly and the robot still lags behind as it approaches location 2 at which point, in order to make more *Progress*, the robot switches strategies and attempts to overtake from the other side (towards the Right of the goal). As the robot approaches location 3, the agents start turning acutely. and in order to reduce *Blame*, rather than going straight towards the goal, the robot tries to swerve around the agent from the Left of its goal. This extreme switch causes the robot to deviate once again. By allowing smoother transitions in nominal speed and heading, C-MPDM avoids unnecessary deviation (*Right*).

Discussion: Undesirable behavior arising from discrete ego-policies

Fig. 7 shows the resultant trajectories from an experiment where two pedestrians walk closely in front of the robot, but not directly towards the robot's goal. The pedestrians arbitrarily change speed along their path and suddenly turn acutely near the robot's goal. With a small set of discrete hand-crafted policies, MPDM is forced to make extreme choices such as overtaking from the right or left, or going very slowly. Switching between these policies can result in undesirable trajectories (Fig. 7). C-MPDM alleviates this problem through continuous-valued parameterized policies, allowing the robot to modulate its speed and heading. The attached video shows the dynamics of this scenario.

VI. Conclusions

In this paper, we have extended MPDM, allowing ego-policies to have continuous-valued parameters and reducing the need for carefully hand-engineered policies. C-MPDM radically improves the flexibility of MPDM while simultaneously satisfying real-time constraints by quickly finding promising parameters through an iterative gradient-based optimization. As a result, we can generate a continuum of risk-aware policies allowing the robot to adapt better to the dynamic environment which is critical for real-time risk-aware navigation, as demonstrated through our experimental results and the attached video.

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] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard, "A bilevel model for toll optimization on a multicommodity transportation network," *Transportation Science*, vol. 35, no. 4, pp. 345–358, 2001.

[3] C. Jones and M. Morari, "Approximate explicit mpc using bilevel optimization," in *European Control Conference*. IEEE, 2009, pp. 2396–2401.

[4] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals of operations research*, vol. 153, no. 1, pp. 235–256, 2007.

[5] 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.

[6] 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*, 2009, pp. 4027–4033.

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

[8] 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.

[9] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[10] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *arXiv preprint arXiv:1703.08862*, 2017.

[11] N. Ye, A. Somani, D. Hsu, and W. Lee, "DESPOT: Online POMDP planning with regularization," *Journal of Artificial Intelligence Research*, vol. 58, pp. 231–266, 2017.

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

[13] H. von Stackelberg, *The Theory of the Market Economy*. Oxford University Press, 1952.

[14] A. Sinha, P. Malo, A. Frantsev, and K. Deb, "Multi-objective stackelberg game between a regulating authority and a mining company: A case study in environmental economics," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 478–485.

[15] A. U. Raghunathan and L. T. Biegler, "Mathematical programs with equilibrium constraints (mpecs) in process engineering," *Computers & chemical engineering*, vol. 27, no. 10, pp. 1381–1392, 2003.

[16] H. I. Calvete, C. Galé, and M.-J. Oliveros, "Bilevel model for production–distribution planning solved by using ant colony optimization," *Computers & operations research*, vol. 38, pp. 320–327, 2011.

[17] D. Mehta, G. Ferrer, and E. Olson, "Backprop-MPDM: Faster risk-aware policy evaluation through efficient gradient optimization," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2018.

[18] P. Hansen, B. Jaumard, and G. Savard, "New branch-and-bound rules for linear bilevel programming," *SIAM Journal on scientific and Statistical Computing*, vol. 13, no. 5, pp. 1194–1217, 1992.

[19] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: from classical to evolutionary approaches and applications," *IEEE Transactions on Evolutionary Computation*, 2017.

[20] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[21] L. J. Ratliff, S. A. Burden, and S. S. Sastry, "Characterization and computation of local nash equilibria in continuous games," in *Communication, Control, and Computing (Allerton), 2013 51st Annual Allerton Conference on*. IEEE, 2013, pp. 917–924.

[22] K. J. Arrow, L. Hurwicz, H. Uzawa, and H. B. Chenery, "Studies in linear and non-linear programming," 1958.

[23] D. Richert and J. Cortés, "Robust distributed linear programming," *IEEE Transactions on Automatic Control*, vol. 60, no. 10, pp. 2567–2582, 2015.

[24] X. Zhang and A. Papachristodoulou, "A real-time control framework for smart power networks with star topology," in *American Control Conference (ACC), 2013*. IEEE, 2013, pp. 5062–5067.

[25] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

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

[27] D. Mehta, G. Ferrer, and E. Olson, "Fast discovery of influential outcomes for risk-aware MPDM," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017, pp. 6210–6216.