Optimal Path-Planning with Random Breakdowns

Marissa Gee[1,2], Alexander Vladimirsky[2,3]

***Abstract*— We derive a model for path planning in the presence of random breakdowns, present a notion of expected cost that incorporates the potential cost of breakdowns and repairs, and use techniques from dynamic programming to generate optimal trajectories. We assume that the risk of breakdown is known throughout the domain, though it may be spatially inhomogenous. When modeling the cost of breakdown and repair we allow two types of breakdowns: total, which halt all movement, and partial, after which movement continues in a damaged state. We use the formalism of piecewise-deterministic Markov processes to derive a system of coupled PDEs governing the optimal behavior in all possible states. We present efficient iterative methods for solving these PDEs numerically that make use of existing efficient methods for deterministic problems. Finally, we perform computational experiments to illustrate the impact of the system parameters on optimal trajectories, and analyze an example based on Martian terrain data.**

I. INTRODUCTION

Robustness is one of the central challenges in path-planning for autonomous vehicles. Even when the dynamics and navigation environment are fully known, there is a question of how to account for the possibility of a partial or total breakdown. Much of the existing literature on planetary rovers models the undesirability of such events heuristically. The speed or cost functions might be modified ad hoc to reflect the risk [1], or terrain obstacles might be classified and essentially excluded from the environment before the actual trajectory planning [2]. A more rigorous approach could be based on multiobjective path-planning [3], recovering the Pareto Frontier to capture all possible tradeoffs between the primary optimization criterion (e.g., the time to target when fully functional) and the risk of a breakdown along the way. This, however, still ignores the possibility of repeated breakdowns. More importantly, all of these approaches ignore the fact that the actual consequences for the mission depend not just on the fact of a random breakdown but also on its *type* and *location*. We propose a model to address these limitations rigorously and systematically in the framework of piecewise-deterministic Markov processes (PDMPs) [4].

We consider an autonomous robot attempting to reach a target while subject to random breakdowns, whose (location-dependent) probability is known in advance. After a *total breakdown*, the robot cannot move, and must pay for an in-place repair. After a *partial breakdown*, it can continue traveling, but must pass through a repair depot on its way to the target. Until a depot is reached, the damaged robot is essentially solving a different optimization problem since it might have a reduced speed or control authority, a restricted set of directions of motion, or a lower energy efficiency. We model these different modes (fully functional and damaged) using PDMPs. A PDMP is a stochastic model where at any point in time the system is in one of finitely many modes. The system switches between these modes stochastically at known rates, while each mode specifies its own deterministic dynamics and running cost. Recently, PDMPs have been applied to path planning problems with changing environments, where the modes represent environmental states such as changing wind direction [5], [6]. We take a similar approach here, but use the modes to model the status of the robot itself.

The range of path-planning methods used in the robotics and optimal control literature is truly broad (see [7]-[11] for some examples). Our approach is based on dynamic programming in continuous state and time: we obtain globally optimal trajectories by solving Hamilton-Jacobi-Bellman PDEs. This is a popular framework (e.g., [12]-[14]), particularly suitable when the state space is low-dimensional. But we note that our main ideas are also suitable for modifying popular discrete state path-planning methods in higher dimensions [9], [10].

Our model and the computational approach are quite general, but here we describe them for isotropic problems only, primarily to simplify the exposition. We start by describing the problem statement and the structure of the governing PDEs for three path-planning scenarios in Section II. We then present an iterative numerical method for solving these PDEs (Section III) and results of computational experiments (Section IV). Most of our test problems use synthetic data to illustrate the effects of system parameters on optimal trajectories, but the last example is more realistic and is based on terrain data for a region of Mars near Jezero crater [15]. We conclude by discussing future extensions in Section V.

II. GENERAL SETTING

A. Classical Path Planning

We consider a robot that obeys the isotropic dynamics

$$\dot{\mathbf{y}}(s) = \mathbf{a}(s) f(\mathbf{y}(s)),\ \mathbf{y}(0) = \mathbf{x}, \tag{1}$$

throughout some bounded domain Ω, where $\mathbf{a}(s) : \mathbb{R} \to S^1$ is the chosen direction of motion and f is the speed of travel. We seek the policy $\mathbf{a}(\cdot)$ that minimizes the cumulative cost

$$J(\mathbf{x}, \mathbf{a}(\cdot)) = \int_0^T K(\mathbf{y}(s))ds + q(\mathbf{y}(T)). \tag{2}$$

Here K is a running cost and q is a terminal cost. While T could be set a priori, our focus is on exit-time problems: let

*Supported by the NSF DMS (awards 1645643, 1738010, and 2111522).
[1]mag433@cornell.edu
[2]Center for Applied Mathematics, Cornell University
[3]Department of Mathematics, Cornell University

$G \subset \Omega$ be a finite set of points representing the target, then $T = \inf\{s|\mathbf{y}(s) \in G\}$.

The value function $u(\mathbf{x})$ is defined to encode the optimal cost-to-go from each point in Ω:

$$u(\mathbf{x}) = \inf_{\mathbf{a}(\cdot)\in\mathcal{A}} J(\mathbf{x}, \mathbf{a}(\cdot)), \tag{3}$$

where $\mathcal{A}$ is the set of measurable functions from $\mathbb{R}$ to S^1. Classical arguments from control theory [16] show that u must be the viscosity solution of the Eikonal PDE (suppressing the dependence on $\mathbf{x}$ for clarity):

$$f|\nabla u| = K,\ \mathbf{x} \in \Omega \qquad u = q,\ \mathbf{x} \in G, \tag{4}$$

for which there are several well-known efficient numerical solvers; e.g., [17]-[20]. Once u is computed, the optimal policy can be recovered by setting $\mathbf{a}^*(s) = -\nabla u(\mathbf{y}(s))/|\nabla u(\mathbf{y}(s))|$. In the special case that $K = 1$ and $q = 0$, $\mathbf{a}^*(s)$ is the time-optimal control.

B. Simplified Model: Total Breakdowns Only

The first extension introduces the possibility of total breakdowns. Let $R(\mathbf{x})$ be the in-place repair cost the robot must pay before continuing, and let $\lambda(\mathbf{x})$ be the rate at which total breakdowns occur. If R is known, then the *expected cost* associated with a trajectory $\mathbf{y}(s)$ is

$$J(\mathbf{x}, \mathbf{a}(\cdot)) = \int_0^T K(\mathbf{y}(s)) + \lambda(\mathbf{y}(s))R(\mathbf{y}(s))ds \tag{5}$$

(where we now assume $q = 0$ at the target). While we model R as being paid instantaneously, we will see later that we can choose R to capture the time taken for a repair (assuming the robot does not accrue running cost while broken down). In that case, s becomes the time spent *moving*, not the total time *taken*. The structure of equation (5) is identical to that of (2), and thus the same arguments show that the value function u solves

$$f|\nabla u| = K + \lambda R,\ \mathbf{x} \in \Omega \qquad u = 0,\ \mathbf{x} \in G. \tag{6}$$

We now describe a particular model for R. Suppose the domain contains a finite number of repair depots at $D = \{\tilde{\mathbf{x}}_1, ..., \tilde{\mathbf{x}}_M\} \subset \Omega$, and that a repair vehicle must travel from a depot to the broken down robot to repair it. If the repair vehicle has its own speed f_R and running cost K_R and minimizes its own travel cost, we can write

$$R(\mathbf{x}) = u_R(\mathbf{x}) + R_F(\mathbf{x}), \tag{7}$$

$$f_R|\nabla u_R| = K_R,\ \mathbf{x} \in \Omega \qquad u_R = R_D,\ \mathbf{x} \in D \tag{8}$$

where R_D and R_F are depot and breakdown location dependent repair costs, respectively. This structure provides great flexibility in modeling the cost of a total breakdown, while still allowing R to be precomputed throughout the domain. For example, if we set $K_R = 1$, R_F to be the time for an in-place repair, and R_D to be the waiting time at each depot, then R captures the time spent broken down. Alternatively, if the repair vehicle charges a rate C for its time, we can scale R_F by C, K_R by $2C$ (to account for the return trip), and set $R_D = 0$ (assuming we are not charged for waiting time) to model the amount *paid* for a repair.

C. Full Model: Total and Partial Breakdowns

Now, we allow two types of breakdown to occur: total, as defined above, and partial, where the robot can continue moving after the breakdown (possibly with a lower f or higher K or λ). This means that the robot can now move in two distinct modes: mode 1, when it is fully functional, and mode 2, when it is damaged. We model this as a PDMP, where each mode has its own value function and they are coupled due to switching via breakdowns and repairs. If we first assume u_2 (the value function in mode 2) is known, then the expected cost in mode 1 is

$$J_1(\mathbf{x}, \mathbf{a}(\cdot)) = \mathbb{E}\left[\int_0^{T_1} K_1(\mathbf{y}(s)) + \lambda_1(\mathbf{y}(s))R(\mathbf{y}(s))ds + \mathbb{1}_{[T_b<T_G]}u_2(\mathbf{y}(T_b))\right]. \tag{9}$$

where T_b is the time at which the first partial breakdown occurs, $T_G = \inf\{s|\mathbf{y}(s) \in G\}$ is the time the agent would reach the target if no breakdown occured, and $T_1 = \min\{T_b, T_G\}$. If partial breakdowns occur at some known rate $\phi(\mathbf{x})$, then this is an example of a randomly terminated finite horizon control problem, as outlined in [21].

We can similarly define J_2, assuming that u_1 is known. In mode 2, the robot returns to mode 1 by being repaired at a depot or having a total breakdown and paying R. Let T_B be the time of the first total breakdown, $T_D = \inf\{s|\mathbf{y}(s) \in D\}$ be the time the robot would reach the depot if no breakdowns occured, and $T_2 = \min\{T_B, T_D\}$, then

$$J_2(\mathbf{x}, \mathbf{a}(\cdot)) = \mathbb{E}\left[\int_0^{T_2} K_2(\mathbf{y}(s))ds + \mathbb{1}_{[T_B<T_D]}(R(\mathbf{y}(T_B)) + u_1(\mathbf{y}(T_B)) + \mathbb{1}_{[T_B\geq T_D]}(R_D(\mathbf{y}(T_D)) + u_1(\mathbf{y}(T_D))\right]. \tag{10}$$

Following the derivation presented in [6], we arrive at the following system of coupled PDEs for the value functions:

$$f_1|\nabla u_1| = K_1 + \lambda_1 R + \phi(u_2 - u_1) \quad \mathbf{x} \in \Omega \tag{11}$$
$$u_1 = 0 \quad \mathbf{x} \in G$$
$$f_2|\nabla u_2| = K_2 + \lambda_2(R + u_1 - u_2) \quad \mathbf{x} \in \Omega \tag{12}$$
$$u_2 = R_D + u_1 \quad \mathbf{x} \in D.$$

The inclusion of partial breakdowns significantly complicates the model. Previously, it was possible to sequentially solve for R and then u, but due to the coupling of u_1 and u_2 that strategy is no longer possible in general. One exception is when the only depot is located at the target and $\lambda_2 = 0$. In this case, the boundary condition for u_2 is known, since plugging in $u_1 = 0$ on G gives $u_2 = R_D$ on D, and the PDEs become decoupled. We can also recover simpler models as special cases of equations (11) and (12). First, setting $\lambda_1 = \lambda_2 = \phi = 0$ recovers the classical path planning problem without breakdowns. If we instead set $\lambda_2 = \phi = 0$, we obtain the simplified model from the previous subsection.

Finally, letting $\lambda_1 = \lambda_2 = 0$ corresponds to an additional model with only partial breakdowns, which we examine in Examples 3 and 4.

III. NUMERICS

A. Discretized Equations

The main computational challenge in solving (11) and (12) is the coupling between u_1 and u_2: without it, the PDEs could be solved entirely using existing numerical techniques. Still, we will make use of multiple existing methods, so we start by reviewing them and later address solving the coupled systems. We will use the discretization $(x, y) \approx (x^{ij}, y^{ij})$, where we assume for convenience that the distances between gridpoints, Δx and Δy, are constant.

When solving Eikonal PDEs, we use the Fast Marching Method (FMM) [17], which relies on having a causal discretization of the PDE. We use one-sided finite difference approximations of the derivatives, given by

$$D^{ij}_{\pm x} u = \frac{\pm u^{i\pm 1, j} \mp u^{ij}}{\Delta x}, \qquad D^{ij}_{\pm y} u = \frac{\pm u^{i, j\pm 1} \mp u^{ij}}{\Delta y}, \tag{13}$$

to define the upwind difference operator in the x-direction

$$\mathcal{D}^{ij}_x u = \min\{D^{ij}_{+x} u, -D^{ij}_{-x} u, 0\} \tag{14}$$

and its equivalent for the y-direction. We can then write the equations that u must solve at each point in the discretized domain. For the Eikonal equation we have

$$\sqrt{\left(\mathcal{D}^{ij}_x u\right)^2 + \left(\mathcal{D}^{ij}_y u\right)^2} = \frac{K^{ij}}{f^{ij}}. \tag{15}$$

We can solve for u by propagating the boundary values as outlined in [17].

We also require solvers for a class of uncertain horizon problems described in [21], which extends the FMM to problems with random termination and *known* terminal cost. Equations (11) and (12) fit this framework if we assume that either u_2 or u_1, respectively, are known. If that were the case, we would arrive at the following discretized equations:

$$\sqrt{\left(\mathcal{D}^{ij}_x u_1\right)^2 + \left(\mathcal{D}^{ij}_y u_1\right)^2} = \frac{K_1^{ij} + \lambda_1^{ij} R^{ij}}{f_1^{ij}} - \frac{\phi^{ij}}{f_1^{ij}}(u_1^{ij} - u_2^{ij}) \tag{16}$$

$$\sqrt{\left(\mathcal{D}^{ij}_x u_2\right)^2 + \left(\mathcal{D}^{ij}_y u_2\right)^2} = \frac{K_2^{ij}}{f_2^{ij}} - \frac{\lambda_2^{ij}}{f_2^{ij}}(u_2^{ij} - (u_1^{ij} + R^{ij})) \tag{17}$$

and each value function could be found using an existing causal method, assuming the other were known.

B. Value Iterations

To take advantage of the methods presented above, we take an iterative approach similar to standard value iteration. At each iteration, we first freeze the value of u_2 and use an existing method to solve (11) for u_1. We then use the new version of u_1 to solve (12) for u_2 in the same way. We alternate updates in this manner until the change between iterations, δ, falls below a specified tolerance tol.

Algorithm 1: Solve equations (11) and (12) via value-policy iteration.

Input: tol, termination threshold
Input: $\rho < 1$, policy evaluation threshold
Solve *Eq.* (8) *for* u_R *using FMM*
Set $R = u_R + R_F$
Set $n = 0$, $d^{(0)} = tol + 1$, $\delta = d^{(n)}$
Initialize $u_1^{(0)}$ *and* $u_2^{(0)}$ *with overestimates*
while $d^{(n)} > tol$ **do**
  while $d^{(n)} > \rho \cdot \delta$ **do**
    Initialize $u_2^{(n+1)} = u_1^{(n)}$ *on* D
    Solve *Eq.* (12) *for* $u_2^{(n+1)}$ *using modified FMM with* $u_1 = u_1^{(n)}$
    Solve *Eq.* (11) *for* $u_1^{(n+1)}$ *using modified FMM with* $u_2 = u_2^{(n+1)}$
    Set $n = n + 1$, $d_n = \|u_1^{(n)} - u_1^{(n-1)}\|_\infty$
  end
  Solve *Eqs.* (19) *and* (20) *for* r_1 *and* r_2
  Set $\delta = \|u_1^{(n)} - r_1\|_\infty$, $u_1^{(n)} = r_1$, $u_2^{(n)} = r_2$
end

To start the iterative process, we need to initialize u_2 using some overestimates, which we obtain by posing simpler "pessimistic" problems. For all pessimistic problems, we assume that the robot does not leave mode 2 after a total breakdown, even after paying R – thus, the only coupling between u_1 and u_2 is on D. This provides an overestimate if $f_2 \le f_1$, $K_2 \ge K_1$, and $\lambda_2 \ge \lambda_1$. To obtain a pessimistic version of u_1 at every $\tilde{\mathbf{x}}_m \in D$ (and thus an overestimate of the boundary conditions for (12)), we replace f_k, ϕ, and $K_k + \lambda_k R$ by their worst values and assume that the vehicle moves along a straight line from $\tilde{\mathbf{x}}_m$ to the closest point in G, returning to $\tilde{\mathbf{x}}_m$ along the same line in case of a partial breakdown. This makes the problem essentially 1D, making it easy to solve analytically using a system of two coupled linear ODEs. We then initialize u_2 throughout Ω by solving

$$f_2|\nabla u_2| = K_2 + \lambda_2 R, \mathbf{x} \in \Omega \qquad u_2 = R_D + \hat{u}_1, \mathbf{x} \in D \tag{18}$$

where $\hat{u}_1$ is the overestimate produced above.

C. Acceleration via Value-Policy Iteration

One short-coming of value iterations is that δ quickly becomes small, even when the current value function is far from the solution. Thus, pure value iterations are slow to converge, especially for poor initializations of the system. To remedy this, we extend to PDMPs the method of combined value iterations and policy evaluations, outlined in [22] for single mode problems. Policy evaluation is commonly found as a step in policy iteration, a popular dynamic programming technique, and involves fixing the control $\mathbf{a}(s)$ and com-

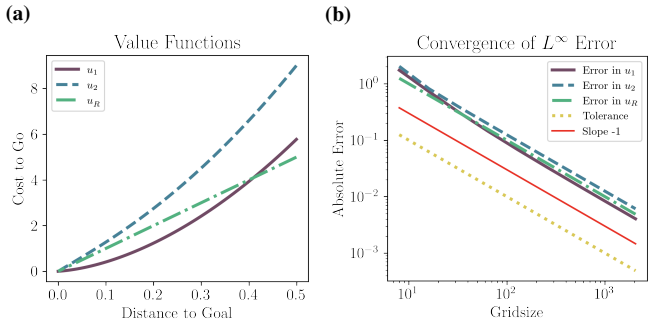

Fig. 1: Example 1. Radially symmetric value functions with $G = D = \{(0.5, 0.5)\}$. **(a)** Value functions for $\phi = 5$, $\lambda_1 = 0.5$, $\lambda_2 = 1.5$, $f_1 = 1$, $f_2 = 0.2$, and $f_R = 0.1$. **(b)** L^∞ error between analytic and numeric solutions, with value iteration tolerance and slope -1 line for reference. Error exhibits first order convergence.

puting the value function exactly for that fixed, suboptimal policy.

Fixing $\mathbf{a}_1(s)$ and $\mathbf{a}_2(s)$ in equations (11) and (12) we get

$$f_1(\mathbf{a}_1 \cdot \nabla r_1) = K_1 + \lambda_1 R + \phi(r_2 - r_1) \quad \mathbf{x} \in \Omega \tag{19}$$
$$r_1 = 0 \quad \mathbf{x} \in G$$
$$f_2(\mathbf{a}_2 \cdot \nabla r_2) = K_2 + \lambda_2(R + r_1 - r_2) \quad \mathbf{x} \in \Omega \tag{20}$$
$$r_2 = R_D + r_1 \quad \mathbf{x} \in D,$$

a system of coupled linear PDEs that can be discretized using finite differences and solved efficiently using any large-scale linear solver. The combined value-policy iteration algorithm for two coupled value functions is outlined in Algorithm 1.

IV. NUMERICAL RESULTS

For all examples except the first, the value functions are represented on a 501×501 grid on $[0, 1] \times [0, 1]$. For all examples, we set $tol = 2/(\Delta x + \Delta y)$, $K_k = 1$ for all modes, $R_F = 1$, $R_D = 0$, and use spatially homogenous environmental parameters f_1, f_2, f_R, λ_1, λ_2, and ϕ unless otherwise specified. [1]

A. Example 1: Convergence of Iterative Scheme

Example 1 (Figure 1) considers a model for which the solution can be computed analytically: one target and one depot, both at the same location. The solution in two dimensions will be radially symmetric about G, and can be obtained by solving a system of ODEs analytically. Figure 1**a** shows the value functions plotted as a function of distance to the target. Figure 1**b** shows the first order convergence of the iterative scheme under grid refinement.

B. Example 2: Inhomogenous Partial Breakdown Rate

We now consider an inhomogenous partial breakdown rate, shown in Figure 2**a**. We set $f_1 = 1$, and use $f_2 = 0.2f_1$ and $f_R = 0.1f_1$, to enforce slower speeds for the damaged robot and the repair vehicle. Figure 2**b** shows the mode 1 value function in the absence of total breakdowns, as well as sample optimal trajectories that clearly avoid areas with high values of ϕ. Without total breakdowns ($\lambda_1 = \lambda_2 = 0$), there is no need to compute u_R and u_2 is radially symmetric about G, so it is optimal to travel in a straight line to the target in mode 2.

This changes once we introduce a chance of total breakdowns. Figures 2**c** and 2**d** show the value functions in mode 2 and mode 1 when we introduce a *constant* chance of total breakdown ($\lambda_1 = 1$, $\lambda_2 = 3$). Now, the mode 2 optimal trajectories are no longer straight lines. Instead, the robot avoids areas where ϕ is high so as not to be caught there if it is repaired in-place after a total breakdown. Thus, the introduction of even a constant value of λ_2 can disrupt the radial symmetry of u_2 about the target when ϕ is inhomogenous.

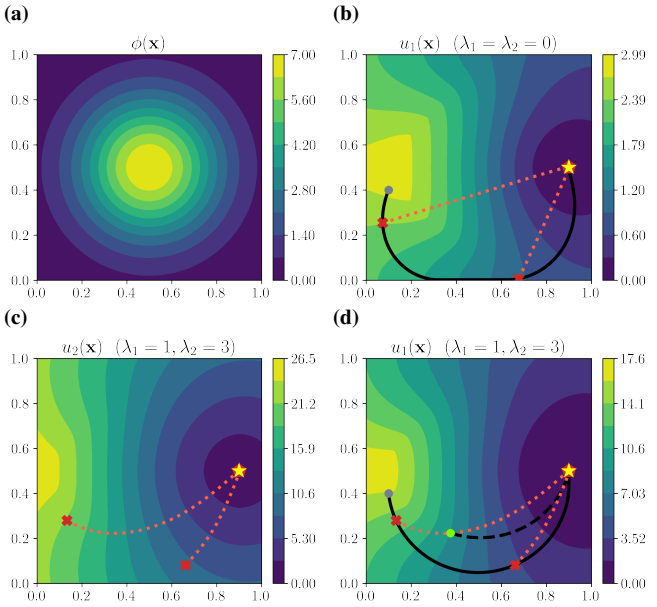

Fig. 2: For all figures: gold stars are targets, red pentagons are depots, grey dots are sample starting locations, red Xs are potential breakdown locations, and green dots are potential in-place repairs. Black trajectories are optimal in mode 1 and red dotted trajectories are optimal in mode 2. Example 2: $G = D = \{(0.9, 0.5)\}$. **(a)** The partial breakdown rate. **(b)** Value function for mode 1 and sample trajectories for modes 1 and 2 when $\lambda_1 = \lambda_2 = 0$. **(c)** Value function and sample trajectories for mode 2, when $\lambda_1 = 1$ and $\lambda_2 = 3$. The optimal trajectories are orthogonal to the level sets of u_2. **(d)** Value function and sample trajectories for mode 1 when $\lambda_1 = 1$ and $\lambda_2 = 3$.

C. Example 3: Changing Partial Breakdown Rate

Example 3 (Figure 3) examines how changes in ϕ lead to changes in optimal paths. We set $\lambda_1 = \lambda_2 = 0$, and use the same speed values as Example 2. Since $G \bigcap D = \emptyset$, a damaged robot *must* first travel to a depot rather than directly to the target. This example illustrates the tradeoffs associated with extending the planned mode 1 trajectories. Longer trajectories can remain closer to D, thus reducing the expected time spent traveling in mode 2, but they also increase the chance that a breakdown occurs in the first place. Figure 3**b** shows the impact of changing ϕ on the optimal trajectories in mode 1. With no chance of breakdowns ($\phi = 0$), the robot would follow the yellow straight line to G. For relatively low values of $\phi > 0$, it is optimal to extend the path

[1] Code for all examples is available at `https://github.com/eikonal-equation/Random_Breakdowns`

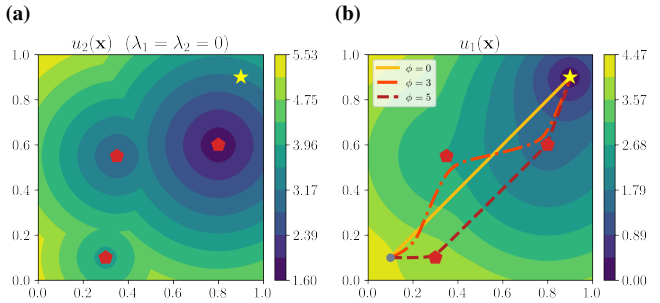

Fig. 3: Example 3, $G = \{(0.9, 0.9\})$ and $D = \{(0.3, 0.1), (0.35, 0.55), (0.8, 0.6)\}$. **(a)** Mode 2 value function with $\phi = 3$ and $\lambda_1 = \lambda_2 = 0$. **(b)** Mode 1 value function for the same parameters. Sample optimal trajectories in mode 1 for $\phi = 0$ (yellow), $\phi = 3$ (orange & orthogonal to the shown level sets of u_1), and $\phi = 5$ (red) starting at $\mathbf{x} = (0.1, 0.1)$.

(orange) in order to stay closer to D and decrease the time spent traveling in mode 2. However, as ϕ increases, multiple breakdowns become more likely, and the robot might have to travel in mode 2, even if after a recent repair. Eventually, it is no longer worth extending the path (red) to be near every repair depot, and it becomes better to prioritize reaching the best depot (i.e., the one with the lowest expected cost to go) as quickly as possible.

D. Example 4: Real-world Terrain

Example 4 (Figure 4) models path planning for a Mars rover, with speed and breakdown rate based on real-world terrain. We examine a region of Mars near Jezero crater, which was the landing site of NASA's Perseverance rover, and thus has extensively characterized terrain. All data was accessed via JMARS, a Mars GIS [15]. We choose to model speed based on the slope of the terrain. Current Mars rovers can navigate slopes of up to approximately $\overline{\sigma} = 20°$, and travel more slowly on steeper slopes due to increased wheel slippage [23]. We measure speed in units of meters/sol (a *sol* is a Martian day). For simplicity we assume that the rover travels at a small but nonzero speed $f_{min} = 1\, m/sol$ for any slope above $\overline{\sigma}$, and following [24] we set $f_{max} = 200\, m/sol$. Therefore, speed is modeled as

$$f_i(\mathbf{x}) = \begin{cases} f_{max} - \frac{f_{max} - f_{min}}{\overline{\sigma}} \sigma(\mathbf{x}) & \sigma \leq \overline{\sigma} \\ f_{min} & \sigma > \overline{\sigma} \end{cases} \tag{21}$$

where $\sigma(\mathbf{x})$ is the slope and we assume that $\overline{\sigma}$ and f_{max} are decreased by a factor of 2 in mode 2. Figures 4**a** and 4**b** show f_i for the region we consider.

We do not allow total breakdowns ($\lambda_1 = \lambda_2 = 0$), since repairs are not feasible for a Mars rover. However, we do allow for partial breakdowns, and set $D = G$, ensuring that the rover will continue to travel to the target in mode 2. The breakdown rate is modeled as proportional to the *terrain roughness*, $\rho(\mathbf{x})$. We compute ρ as the root-mean-square height, a standard measure in geology [25], that has also been used to characterize traversability for Martian rovers in the past [1]. We model the breakdown rate as

$$\phi(\mathbf{x}) = \frac{\rho(\mathbf{x})^2}{5000} \tag{22}$$

and it is shown in Figure 4**c**. The scaling that we choose is arbitrary, and can be modified to capture the sensitivity of the rover being studied.

Figures 4**d** and 4**e** show u_2 and u_1 and sample optimal trajectories for Example 4. In mode 1, the rover is able to traverse the crater's rim by avoiding the most steeply sloped portions. If a breakdown occurs before it has finished climbing, it may become optimal to take a much longer route that avoids the steep slopes of the rim. However, if a breakdown occurs after the rover has navigated most of the treacherous terrain, it is optimal to climb down the slope towards the target, though with an altered path that avoids steeper slopes. Figure 4**f** shows the elevation of the region of interest, a single mode 1 optimal trajectory, and possible breakdown locations and subsequent mode 2 paths.

V. CONCLUSION

We presented a piecewise-deterministic framework for optimal path planning with two types of random breakdown, total and partial, and showed that each leads to structurally different PDE systems for the value functions. We also presented an efficient numerical method for solving these systems via value-policy iterations, and verified the first order of accuracy of our approximate solutions experimentally. Our test problems illustrated the effect that the breakdown type, breakdown rate, and depot/goal placement have on optimal trajectories. Finally, we computed time-optimal trajectories that account for random partial breakdowns in an environment based on Martian terrain data.

One obvious extension of our model is to allow the robot to voluntarily terminate mode 2, paying for the full in-place repair before a total breakdown actually occurs. This would in effect replace equation (12) with a quasi-variational inequality [21]. Another natural extension is to consider anisotropic problems, allowing for control-dependent K, f, ϕ, and λ, and taking advantage of the numerical methods in [18] or [26]. Extensions could also focus on switching rates, either by allowing them to be time-dependent, or by adding more modes, to represent additional levels of damage. In addition, it will be also useful to look beyond the expected cost, maximizing the probability that the cumulative cost falls below any specified threshold [6]. Multiobjective extensions are yet another possibility, with hard or chance constraints, similar to [27], imposed on additional criteria (e.g., directly constraining the probability of a full breakdown on Mars).

ACKNOWLEDGMENT

The authors would like to thank Lars Grüne for his advice on value-policy iteration and Elliot Cartee, whose work on modeling environmental crime inspired this problem statement.

REFERENCES

[1] D. B. Gennery, "Traversability analysis and path planning for a planetary rover," *Autonomous Robots*, vol. 6, no. 2, pp. 131–146, 1999.
[2] H. Seraji, "Traversability index: A new concept for planetary rovers," in *Proceedings 1999 IEEE ICRA (Cat. No. 99CH36288C)*, vol. 3. IEEE, 1999, pp. 2006–2013.

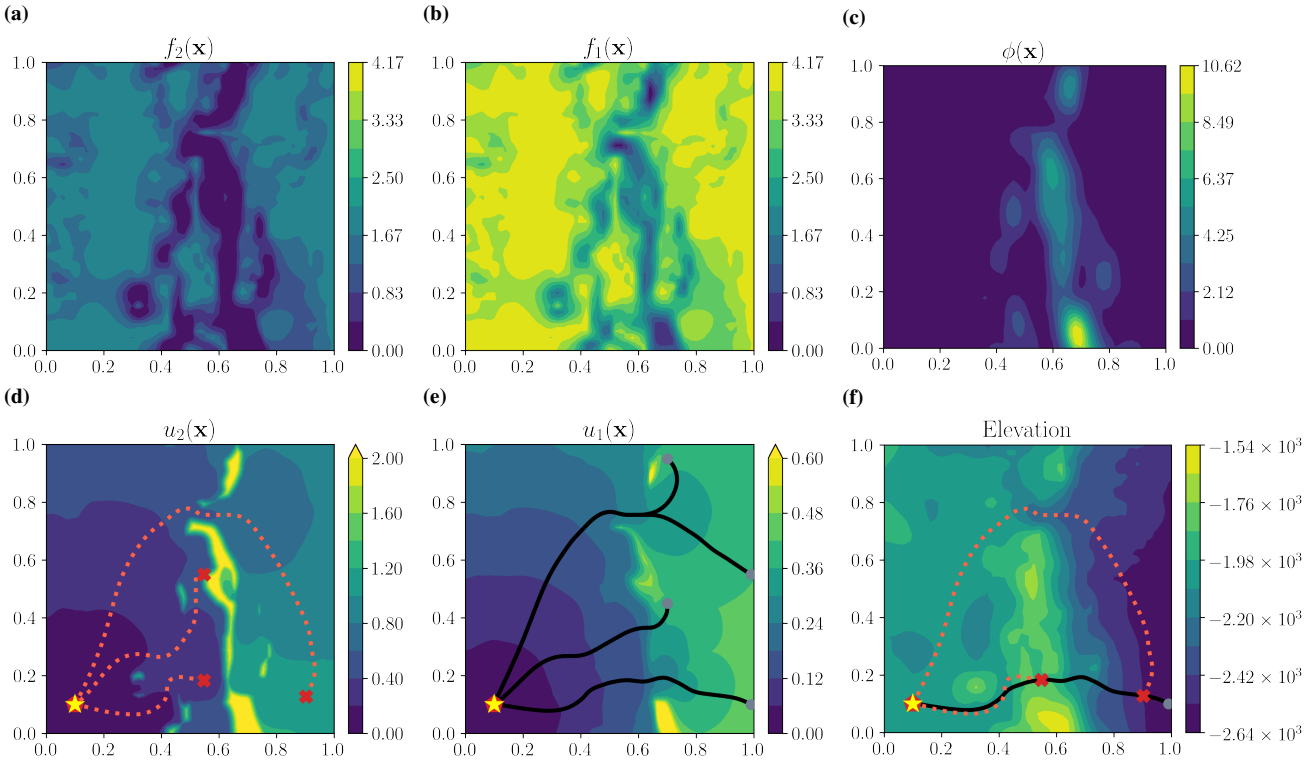

Fig. 4: Example 4, $G = D = \{(0.1, 0.1)\}$. **(a)** Speed of rover in mode 2. **(b)** Speed of rover in mode 1. **(c)** Partial breakdown rate. **(d)** Value function and sample trajectories for mode 2. **(e)** Value function and sample trajectories for mode 1. **(f)** Elevation and sample optimal trajectories in modes 1 and 2.

[3] A. Kumar and A. Vladimirsky, "An efficient method for multiobjective optimal control and optimal control subject to integral constraints," *Journal of Computational Mathematics*, vol. 28, pp. 517–551, 2010.

[4] M. H. Davis, "Piecewise-deterministic Markov processes: A general class of non-diffusion stochastic models," *J. of the Royal Statistical Society: Series B (Methodological)*, vol. 46, no. 3, pp. 353–376, 1984.

[5] Z. Shen and A. Vladimirsky, *Piecewise-Deterministic Optimal Path Planning*. [Online]. Available: http://arxiv.org/abs/1512.08734

[6] E. Cartee, A. Farah, A. Nellis, J. Van Hook, and A. Vladimirsky, "Quantifying and managing uncertainty in piecewise-deterministic Markov processes," *preprint: https://arxiv.org/abs/2008.00555*, 2020.

[7] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[8] B. Chazelle, "Approximation and decomposition of shapes," *Algorithmic and Geometric Aspects of Robotics*, vol. 1, pp. 145–185, 1985.

[9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[10] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[11] K. Alton and I. M. Mitchell, "Optimal path planning under defferent norms in continuous state spaces," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 866–872.

[12] S. Garrido, D. Álvarez, and L. Moreno, "Path planning for Mars rovers using the Fast Marching Method," in *Robot 2015: Second Iberian Robotics Conference*, L. P. Reis, A. P. Moreira, P. U. Lima, L. Montano, and V. Muñoz-Martinez, Eds. Cham: Springer International Publishing, 2016, pp. 93–105.

[13] E. Cartee and A. Vladimirsky, "Control-theoretic models of environmental crime," *SIAM Journal on Applied Mathematics*, vol. 80, no. 3, pp. 1441–1466, 2020.

[14] C. Parkinson, D. Arnold, A. Bertozzi, and S. Osher, "A model for optimal human navigation with stochastic effects," *SIAM Journal on Applied Mathematics*, vol. 80, no. 4, pp. 1862–1881, 2020.

[15] P. Christensen, E. Engle, S. Anwar, S. Dickenshied, D. Noss, N. Gorelick, and M. Weiss-Malik, "JMARS - a planetary GIS," in *AGU Fall Meeting Abstracts*, 2009, pp. IN22A–06.

[16] M. G. Crandall and P.-L. Lions, "Viscosity solutions of Hamilton-Jacobi equations," *Transactions of the American Mathematical Society*, vol. 277, no. 1, pp. 1–42, 1983.

[17] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, February 1996.

[18] Y.-H. R. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao, "Fast sweeping algorithms for a class of Hamilton–Jacobi equations," *SIAM J. Numer. Anal.*, vol. 41, no. 2, pp. 673–694, 2003.

[19] A. Chacon and A. Vladimirsky, "Fast two-scale methods for eikonal equations," *SIAM J Sci Comput*, vol. 34, no. 2, pp. A547–A578, 2012.

[20] ——, "A parallel two-scale method for eikonal equations," *SIAM Journal on Scientific Computing*, vol. 37, no. 1, pp. A156–A180, 2015.

[21] J. Andrews and A. Vladimirsky, "Deterministic control of randomly-terminated processes," *Interfaces and Free Boundaries*, vol. 16, no. 1, pp. 1–40, 2014.

[22] L. Grüne and W. Semmler, "Using dynamic programming with adaptive grid scheme for optimal control problems in economics," *J. Econ. Dyn. Control*, vol. 28, no. 12, pp. 2427 – 2456, 2004.

[23] M. Heverly, J. Matthews, J. Lin, D. Fuller, M. Maimone, J. Biesiadecki, and J. Leichty, "Traverse performance characterization for the Mars Science Laboratory rover," *Journal of Field Robotics*, vol. 30, no. 6, pp. 835–846, 2013.

[24] M. Ono, B. Rothrock, E. Almeida, A. Ansar, R. Otero, A. Huertas, and M. Heverly, "Data-driven surface traversability analysis for Mars 2020 landing site selection," in *2016 IEEE Aerospace Conference*. IEEE, 2016, pp. 1–12.

[25] M. K. Shepard, B. A. Campbell, M. H. Bulmer, T. G. Farr, L. R. Gaddis, and J. J. Plaut, "The roughness of natural terrain: A planetary and remote sensing perspective," *Journal of Geophysical Research: Planets*, vol. 106, no. E12, pp. 32 777–32 795, 2001.

[26] J. A. Sethian and A. Vladimirsky, "Ordered Upwind Methods for Static Hamilton-Jacobi Equations," *Proc. Nat. Acad. Sci.*, vol. 98, no. 20, pp. 11 069–11 074, 2001.

[27] M. Ono, M. Pavone, Y. Kuwata, and J. Balaram, "Chance-constrained dynamic programming with application to risk-aware robotic space exploration," *Autonomous Robots*, vol. 39, no. 4, pp. 555–571, 2015.