# Autonomous Decision Making for Planetary Rovers Using Diagnostic and Prognostic Information

**Sriram Narasimhan** [*] **Edward Balaban** [**] **Matthew Daigle** [**]
**Indranil Roychoudhury** [***] **Adam Sweet** [**] **Jose Celaya** [***] **Kai Goebel** [**]

[*] *University of California, Santa Cruz, NASA Ames Research Center, Moffett Field, CA, 94035, USA*
*E-mail: sriram.narasimhan-1@nasa.gov*
[**] *NASA Ames Research Center, Moffett Field, CA, 94035, USA*
[***] *SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*

**Abstract:** Rover missions typically involve visiting a set of predetermined waypoints to perform science functions, such as sample collection. Given the communication delay between Earth and the rover, and the possible occurrence of faults, an autonomous decision making system is essential to ensure that the rover maximizes the scientific operations performed without damaging itself further or stalling. This paper presents a modular software architecture for autonomous decision making for rover operations that uses diagnostic and prognostic information to influence mission planning and decision making to maximize the completion of mission objectives. The decision making system consists of separate modules that perform the functions of control, diagnosis, prognosis, and decision making. We demonstrate our implementation of this architecture on a simulated rover testbed.

*Keywords:* autonomous decision making, diagnosis, prognosis, planetary rover.

## 1. INTRODUCTION

Planetary rovers are designed to perform science investigation on the surface of celestial objects. Typically, rovers receive a new set of instructions at the beginning of each day. Sent from the scientists and engineers on Earth, the command sequence tells the rover which waypoints to go to and which science experiments to perform. Given the communication time delay between Earth and the rover (e.g., about 20 minutes for Mars), it is not possible to respond to rover events quickly . When an unexpected situation is encountered, e.g., a fault, the rover goes into safing mode, stopping and waiting for instructions from Earth. The ground engineers have to then gather sensor data from the rover to determine its current state and provide a new set of instructions. This whole process might take up to a few days while the rover is idle, costing time and money and losing mission value.

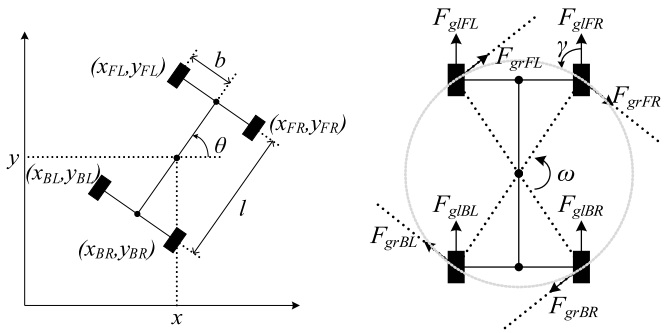
Therefore, a decision making (DM) system that takes a set of waypoints and autonomously decides the order of waypoint traversal while taking into account the health of the different components of the rover can be of immense value to rover operations. In this paper, we develop an integrated DM architecture that incorporates low-level control, diagnosis, prognosis, and decision making algorithms within a single framework for autonomous DM. The diagnosis and prognosis algorithms are not only necessary to maintain vehicle safety and performance by determining whether a vehicle component is failing and predicting how long it will take for it to fail completely; but also to use that information to suggest actions to optimize vehicle maintenance, ensure mission safety, and extend mission duration. Faults and damage progression may require changes to controller gains or switching of control laws, changes to the usage of the system or its components, and/or changes to the mission plan.

We plan to use the K11 rover at NASA Ames Research Center that was originally slated to serve as a robotic technologies test vehicle in the Antarctic (Lachat et al., 2006) to test, verify and validate our architecture (Balaban et al., 2011). In this paper, however, we focus on the overall approach, and demonstrate it instead on a simulation testbed of the rover, describing some interesting fault scenarios. The software simulator allows for rapid validation of autonomy algorithms. The main contribution of this work is the integrated architecture and protocols for how the different components of the architecture interact. Hence, we select some algorithms which have been developed in previous work and only briefly describe how the algorithms work and how they are applied to the rover.

The paper is organized as follows. Section 2 describes the rover hardware and the simulation testbed. Section 3 describes the integrated DM architecture and the selected technologies for the architecture, including control, diagnosis, prognosis, and decision making. Section 4 provides some example scenarios demonstrating the integration of diagnosis, prognosis, and decision making on the simulation testbed. Section 5 concludes the paper.

## 2. TESTBED

The K11 rover is about $1.4$ m long by $1.1$ m wide by $0.63$ m tall. Each wheel is driven by an independent $250$ W graphite-brush motor, connected through a bearing and gearhead system, with its motor controlled by a single-axis digital motion controller. An onboard laptop executes the motor control software, performs data acquisition, and runs the reasoning algorithms. The sensors include a GPS receiver, gyroscope, a still and video camera, and a compass, as well as current, temperature, and wheel speed sensors. The rover software includes the navigation software, middleware, and telemetry software. Further




(a) Generalized rover coordinates. (b) Rover forces.

Fig. 1. Rover coordinates and forces.

details about the K11 rover testbed are described in (Balaban et al., 2011).

The simulation testbed is developed from a physics-based model of the hardware testbed. The model assumes that the rover consists of a symmetric rigid frame with four independently-driven wheels. Generalized rover coordinates are shown in Fig. 1(a). The subscripts $F$, $B$, $L$, and $R$ subscripts stand for "front", "back", "left" and "right", respectively. The rover pose is given by $(x, y, \theta)$. The independent dynamic variables describing the motion include the body longitudinal velocity $v$, the body rotational velocity $\omega$, and the wheel rotational velocities $\omega_{FL}$, $\omega_{FR}$, $\omega_{BL}$, and $\omega_{BR}$. Note that the body velocities and wheel velocities are independent due to the presence of slip, as explained below.

The rover experiences forces opposing movement in both longitudinal and rotational directions (see Fig. 1(b)). For a wheel $w$, when the longitudinal velocity of the rover is equal to that of a wheel, then there is no slip and no force. Otherwise, some amount of slip is present and the difference in the relative velocities produces a ground force $F_{glw}$ that pushes the wheel $w$ along the ground. These forces are transmitted to the rover body, moving it in the longitudinal direction. The $F_{glw}$ forces produce torques on the rover body, producing a rotation that is opposed by additional friction forces $F_{grw}$. The friction forces are defined as $F_{glw} = \mu_{gl}(v_w - v)$ and $F_{grw} = \mu_r \omega$, where $\mu_{gl}$ and $\mu_r$ are friction coefficients for the translational and rotational movements of the rover, respectively. Note that $\mu_{gl}$ and $\mu_r$ are not in the same units. The $F_{grw}$ forces, opposing the rotation, act at a right angle from the diagonal going from the robot center to the wheel, and in the direction that opposes the rotation. The forward component of this force affects the forward velocity of the rover, just as the component of a $F_{glw}$ force perpendicular to the diagonal affects the rotational velocity. The angle $\gamma$ is given by $\gamma = \arctan \frac{l}{2b}$, where $l$ is the length of the rover, and $b$ is half the width of the rover.

For a given wheel $w$ on the left side, the rotational velocity is described by

$$\dot{\omega}_w = \frac{1}{J_w} \left( \tau_{mw} - \tau_{fw} - r_w F_{glw} + r_w F_{gr} \cos\gamma \right),$$

and on the right side as

$$\dot{\omega}_w = \frac{1}{J_w} \left( \tau_{mw} - \tau_{fw} - r_w F_{glw} - r_w F_{gr} \cos\gamma \right).$$

A friction torque existing between a wheel $w$ and the axle and motor produces the counter-torque $\tau_{fw} = \mu_w \omega$.

The forward velocity is described by

$$\dot{v} = \frac{1}{m} \left( F_{glFL} + F_{glFR} + F_{glBL} + F_{glBR} \right),$$

assuming that $\mu_r$ is the same for all wheels. The rotational velocity is described by

$$\dot{\omega} = \frac{1}{J}(d\cos\gamma F_{glFR} + d\cos\gamma F_{glBR} - d\cos\gamma F_{glFL} - d\cos\gamma F_{glBL} - 4dF_{gr}).$$

Note that the $F_{gl}$ forces are at distance $d$ from the rover center with the perpendicular component at angle $\gamma$. The $\cos\gamma$ factor projects the force onto the tangent of the rotation.

The wheel motors are DC motors with PID control is used. The DC motor model for wheel $w$ is given by

$$\frac{di_{mw}}{dt} = \frac{1}{L}(V_w - i_m R - k_\omega \omega_w),$$

where $V_w$ is the motor voltage provided by the controller, $L$ is the motor inductance, $R$ is the motor resistance, and $k_\omega$ is an energy transformation term. The motor torque, $\tau_{mw}$, is described by the relationship $\tau_{mw} = k_\tau i_{mw}$, where $k_\tau$ is an energy transformation gain.

The motors windings are designed to withstand temperatures up to a certain point, after which, the insulation breaks down, the windings short, and the motor fails. This defines an EOL criterion for the motors. The motor temperatures must remain below this critical temperature. The motor thermocouple is located on the motor surface. The surface loses heat to the environment and is heated indirectly by the windings, which, in turn, are heated up by the current passing through them. The temperature of the windings is given by

$$\dot{T}_w = 1/J_w \left( i^2 R - h_w (T_w - T_m) \right),$$

where $J_w$ is the thermal inertia of the windings, $h_w$ is a heat transfer coefficient, and $T_m$ is the motor surface temperature (Balaban et al., 2011). It is assumed that heat is lost only to the motor surface, and that winding resistance $R$ is approximately constant for the temperature range considered. The surface temperature is given by

$$\dot{T}_m = 1/J_s (h_w (T_w - T_m) - h_a (T_m - T_a))$$

where $J_s$ is the thermal inertia of the motor surface, $h_a$ is a heat transfer coefficient, and $T_a$ is the ambient temperature. Heat is transferred from the windings to the surface and lost to the environment.

The batteries are modeled using an electric circuit equivalent model that includes a large capacitance in parallel with a resistance, together in series with another resistance. The batteries are at $48$ V fully charged, and end-of-discharge (which determines an end-of-life criterion for the rover) occurs at $38.4$ V.

We incorporate several faults into the model. A motor friction fault in wheel $w$ is represented by an increase in the motor friction coefficient $\mu_w$. A parasitic load is also considered that drains an additional current from the batteries. Bias and drift faults in sensors are considered as well, and these are represented as additive terms on the sensor equations.

## 3. INTEGRATED DECISION MAKING ARCHITECTURE

A typical rover mission consists of visiting and performing desired scientific operations at a set of predetermined waypoints $\{(x_i, y_i), r_i\}_{i=1}^N$, where $r_i$ is the reward associated with waypoint $(x_i, y_i)$. An autonomous DM system for the rover determines the order to visit the waypoints and the speed $v_i$ to travel

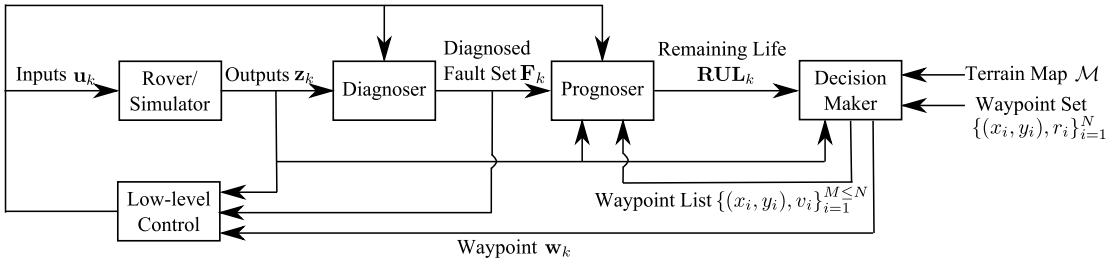



Fig. 2. Integrated decision making architecture, where $k$ denotes the time index.

between them, so as to maximize the reward while minimizing the power used and health deterioration. When diagnostic and prognostic information is available, the decision making system may alter the waypoint list (reduce and/or reorder) to minimize the impact of the faults and their progression over time. Fig. 2 presents the integrated decision making architecture, which consists of four main components, namely (*i*) low-level control (LLC), (*ii*) diagnosis (DX), (*iii*) prognosis (PX), and (*iv*) decision making (DM).

The LLC is responsible for guiding the rover to the current waypoint, $\mathbf{w}_k = \{(x_k, y_k), v_k\}$, by commanding individual wheel velocities to be $v_k$ at time $k$. The rover is a skid-steered vehicle, i.e., its wheels cannot be steered and the rover is rotated by commanding the wheel speeds on the left and right sides to different values. The low-level controller needs to be robust to drive-system faults, therefore, it receives the current diagnosis $\mathbf{F}_k$ from the DX algorithm.

The DX module takes in the inputs $\mathbf{u}_k$ and sensor data $\mathbf{z}_k$, and reasons about faults in the system. It runs continuously, trying to detect when a fault occurs, isolate the underlying faults or failures, and identify the magnitudes of the faults. When a diagnosis $\mathbf{F}_k$ is available, it is passed on to the LLC, for mitigation, and the PX module, which uses this information as a starting point to predict how the fault will progress.

The PX module continuously estimates the health state of the rover and obtains remaining useful life (RUL) predictions, $\mathbf{RUL}_k$. RUL is the time until the system violates functional or performance criteria. The RUL prediction is conditional on the future usage of the rover, determined by the waypoint list.

The DM module is responsible for planning under nominal and faulty conditions. At the mission start, the DM gets a list of $N$ desired waypoints along with associated rewards for visiting them $\{(x_i, y_i), r_i\}_{i=1}^{N}$. The DM also has a terrain map $\mathcal{M}$ that describes the geographical layout of the terrain in which the waypoints are located. The DM then, based on the predicted power usage and RUL predictions received from the PX, determines an ordered list of waypoints. Many different protocols may be implemented for when the DM is called. For example, the DM may be run periodically to adjust for any errors, or it may be called after each waypoint is reached.

In our implementation each of the constituent modules of our integrated DM architecture utilizes one or more types of reasoning algorithms. These algorithms are listed and briefly described in the rest of this section.

### *3.1 Low-level Control*

*Proportional Control* Given the next waypoint and the desired cruise speed, the controller sets the average wheel velocity to achieve the desired cruise speed. It then implements a proportional control based on the difference between the current and desired heading, and adjusts the desired speed of the left and right sides to turn toward the waypoint:

$$v_L^* = v^* - P_\theta e_\theta,$$
$$v_R^* = v^* + P_\theta e_\theta,$$

where the superscript $^*$ indicates a commanded value, $P_\theta$ is the proportional gain, and $e_\theta$ is the heading error.

*Differential Speed* Normally, the desired velocity of the front and back wheels is equal to the desired speed for that side of the rover. However, if a motor friction fault is present on that side, then it is more efficient overall to increase the desired velocity of the good wheel and decrease the velocity of the faulty wheel. The amount of change depends on the magnitude of the friction fault. For a given side of the rover $S$, the good and faulty wheel speeds are determined as:

$$v_{FS}^* = v_S^* 2 \Big/ \left(1 + \tfrac{\beta}{\alpha}\right),$$
$$v_{BS}^* = v_S^* 2 \Big/ \left(1 + \tfrac{\alpha}{\beta}\right),$$

where $\alpha = 1/(1+\mu'_{wF})$, $\beta = 1/(1+\mu'_{wB})$, $v_S^* = (v_F^* + v_B^*)/2$, and $\mu'$ indicates the difference between the faulty value of $\mu$ and the nominal value. These equations assume that at most one motor is faulty on each side. We refer to this as *robust control*.

### *3.2 Diagnosis*

*QED* The Qualitative Event-based Diagnosis (QED) algorithm, described in (Daigle and Roychoudhury, 2010), utilizes a qualitative diagnosis methodology that isolates faults based on the transients they cause in system behavior, manifesting as deviations in residual values (Mosterman and Biswas, 1999). Transients produced by faults are abstracted using qualitative + (increase), − (decrease), and 0 (no change) values to form *fault signatures*. Fault signatures represent these measurement deviations from nominal behavior as the change in magnitude and the first nonzero derivative change. These symbols are computed from the residuals using symbol generation. In addition to signatures, QED captures the temporal order of measurement deviations, termed *relative measurement orderings*. The predicted fault signatures and measurement orderings can be computed manually or automatically from a system model. The predicted signatures and orderings are compared with observed signatures and orderings in order to isolate faults. The combination of signatures and orderings establishes an event-based fault isolation framework.

QED uses the model of the rover described in Section 2to derive fault signatures and measurement orderings. Algebraic functions computing fault magnitudes are derived and used for fault identification.

*HyDE* The Hybrid Diagnosis Engine (HyDE) is a consistency based diagnosis engine that uses hybrid and stochastic models and reasoning (Narasimhan and Browston, 2007). Users build models of constituent components of the system being diagnosed and then compose the system model by defining the connections between the various components. Component models include the discrete modes the components could be in and the behavior of the component in these different modes. Faults are modeled as additional discrete modes of operation of the components, and transitions from the nominal mode to one of these fault modes indicates the occurrence of the fault. At any point in time, HyDE maintains a set of candidates that offer alternate possible states of the system that are consistent with the sensor observations seen so far. When additional observations are available, this candidate set is updated by pruning inconsistent candidates and adding new candidates. Several parameters are available to adjust the performance of the diagnosis engine including heuristics to determine the ranking of different candidates and the kind of simulation to use.

The rover HyDE model was developed directly from the simulation model, with components and equations in one-to-one correspondence with it. Fault modes were added for all sensors, motor friction, and battery parasitic loads.

### *3.3 Prognosis*

*Model-based Prognosis* We use a model-based prognosis paradigm in which prognosis consists of two steps: (*i*) health state estimation, which computes a joint state-parameter estimate $p(\mathbf{x}(k), \boldsymbol{\theta}(k)|\mathbf{z}(0{:}k))$, where $\mathbf{x}$ is the state vector and $\boldsymbol{\theta}$ is the unknown parameter vector, and (*ii*) prediction, which simulates the model forward from a given health state out to the end-of-life (EOL) threshold, based on hypothesized future inputs to the system (Daigle and Goebel, 2011). The prediction module is invoked at time $k_P$ to predict RUL of the system (or, alternatively, end of life (EOL)). Specifically, using the current joint state-parameter estimate, $p(\mathbf{x}(k_P), \boldsymbol{\theta}(k_P)|\mathbf{z}(0{:}k_P))$, which represents the most up-to-date knowledge of the system at time $k_P$, the goal is to compute $p(\text{RUL}(k_P)|\mathbf{z}(0{:}k_P)$. The general approach to solving the prediction problem is through simulation. The state-parameter distribution is sampled, and each sample is simulated forward to EOL to obtain the complete EOL distribution. Note that we need to hypothesize future inputs of the system for prediction, since fault progression is dependent on the operational conditions of the system. The choice of expected future inputs depends on the knowledge of expected operational settings. In the case of the rover, the future inputs are determined by the ordered waypoint list provided by the DM module.

### *3.4 Decision Making*

Currently, the sample problem for the rover application is formulated as the following:
Given:

| | |
|---|---|
| $\boldsymbol{g}(\pi) = \{g_h(\pi), g_e(\pi)\}$ | Inequality constraints on available energy and health |
| $\boldsymbol{f}(\pi) = \{f_r(\pi), f_h(\pi), f_e(\pi)\}$ | Objective functions on cumulative reward, health, and energy |
| $\boldsymbol{v} = \{v_r, v_h, v_e\}, (v_r, v_h, v_e \in [0, 1])$ | Optimization preferences vector |
| $W = \{w_1, w_2, ..., w_L\}$ | Waypoints to be visited |
| $\pi = \{a_1, a_2, ..., a_M\}$ | Policy is defined as a set of actions |
| $a = \{n_i, n_j\}, i \in [1, 2, ...L-1], j \in [1, 2, ...L-1]$ | An action constitutes a move between a pair of waypoints (start and finish) |
| $a_1 = \{n_1, n_1\}$ | The first action is a special case - start traversal at node 1 |
| $a_m = \{n_j, n_k\} \vert (a_{m-1} = \{n_i, n_j\}), (i, j, k \in [1, 2, ...L]), (m \in [2, 3, ...M])$ | Any action after the first one needs to start on the node where the previous one finished |

Find:

| | |
|---|---|
| $\Pi^*$ | A compromise (Pareto) set of policy solutions (see, for instance, Deb (2001)) |

Two PDM approaches are currently implemented: one based on dynamic programming principles and the other one on probabilistic principles inspired by Probability Collectives work (Wolpert, 2006). An exhaustive search method (which is substantially more computationally expensive) is also implemented for verification of the other two algorithms.

*Dynamic Programming* The algorithm uses forward propagation, evaluating the best solution for transitioning from stage to stage, while assuming optimality of the previously made decisions. An *efficiency index* $e_{a_b} = r_b/(\sum \boldsymbol{c_{ab}})$ is used for guiding the stage-to-stage decision process, where $a$ and $b$ are the starting and the ending waypoints, respectively; $r_b$ is the reward value associated with waypoint $b$; and $c_{ab}$ is a transition cost between $a$ and $b$. If either health or energy values become less or equal to zero (or all the nodes are visited), the forward propagation phase is stopped. The solution path is then 'backed out' by traversing the stages in the opposite (right-to-left) direction, starting with the node associated with the highest accumulated reward.

*Probabilistic Decision Making* The probabilistic algorithm operates on a distribution associated with a set of policy roots $\Pi'$ (a policy root, $\pi'$, is an initial segment of a policy $\pi$). The algorithm starts with shorter roots, shaping the probability distribution associated with them as it picks sample roots to extend to full policies and evaluate their fitness. The roots are then extended by one action. To extend the roots, sets of possible follow-on actions are determined first. In the sample problem described, each waypoint should be visited once at the most. Thus, if five waypoints maximum are to be visited, root $\pi' = \{a_1, a_2\}$ has $A_{\pi'} = \{a_3, a_4, a_5\}$ as the set of possible follow-on actions. The valid one-action extensions are then $\{a_1, a_2, a_3\}$, $\{a_1, a_2, a_4\}$, and $\{a_1, a_2, a_5\}$. These *offspring* policy roots replace the *parent* root ($\pi'$) in $\Pi'$ and split its probability value evenly. Once the roots reach the maximum specified length, a Monte-Carlo simulation is ran on the resulting distribution $p(\Pi')$ and the best-performing full-

length policy (according to the objective vector $\boldsymbol{f}$) is selected as the solution.

## 4. CASE STUDY

This section illustrates a practical implementation of the integrated architecture. The case study presented is that of the rover starting its mission at a waypoint $w_0$ and attempting to visit, at an average speed of $0.5$ m/s, a set of $10$ waypoints, accomplishing a scientific objective at each waypoint. As mentioned previously, a reward value is associated with every waypoint and the overall objective is to maximize the cumulative reward. In the absence of system faults, the rover has enough energy stored in the batteries to visit all of the waypoints. In addition to this nominal scenario, three fault scenarios are considered: increased motor friction, a parasitic load in the power distribution system, and a battery voltage sensor fault. In the fault scenarios, the diagnostic system is expected to detect and identify the fault mode during the $w_0 \rightarrow w_1$ transition. The decision making system is then expected to modify the waypoint traversal plan taking into account prognostic estimates of future energy consumption and fault magnitude progression. An *a posteriori* estimate of change in these two quantities during the $w_0 \rightarrow w_1$ transition, given the fault diagnosis, is made as well.

### *4.1 Battery Parasitic Load Fault*

As a first scenario, we consider an abrupt parasitic load fault that draws an additional amount of current from the batteries. A parasitic current of $0.1$ A is injected starting at $50$ s. As a result, the net current increases and the battery voltages decrease in response to the increased current. No other effects appear, but the batteries will drain faster and some decision making will have to be performed to determine if the objectives can still be met with this increased current draw.

QED detects the fault at $50.4$ s with an increase in $i$, the current drawn from the batteries. The candidate set reduces to a failure and friction faults of the motors, the parasitic load fault, and a bias or drift in the current sensor. At $51.2$ s, the change is determined to be abrupt, eliminating the friction faults and the drift fault. At $76.15$ s, a decrease in the voltage of battery $4$ is detected. This eliminates the current sensor fault, since it would not affect another sensor, and eliminates the motor failure faults, as these would cause a change in individual motor currents before a change in voltage (i.e., there is a relative measurement ordering for these faults expressing this constraint), thus uniquely isolating the parasitic load fault. The parasitic current is computed as the difference between the battery current and the sum of the motor currents, averaged from the time of detection to the present time. At $51.2$ s, the value of the parasitic current is estimated at $0.127$ A. By $100$ s, the estimate is at $0.103$ A. HyDE detects the fault at $50.05$ and isolates the fault to either a parasitic load or one of the motors being in an unknown mode. [1] In addition, HyDE was tested on a double fault scenario where there is a battery parasitic load and the battery current sensor is faulty. HyDE is able to use other sensors to determine this condition. The response is the same as with the parasitic load fault since sensor faults do not affect the DM.

[1] In the unknown mode, the behavior of the motor is undefined and hence it could be drawing more current. However, unknown faults are catch-all modes that should be considered only when no other explanations are available.

Prognosis begins once the fault is identified, determining when the batteries will reach end-of-discharge with the current mission plan. The prognosis algorithm reports that the batteries will reach this point in $3.83$ h (as opposed to $4.54$ h without the fault). Continuing at an average speed of $0.5$ m/s, the rover can cover about $6.9$ km with the fault ($8.2$ without the fault). Since the rover must cover over $6.9$ km to cover all waypoints, mission optimization is required. The DM module estimates that there will not be enough energy to visit all of the waypoints and eliminates waypoints $9$ and $4$ from the plan, reconfiguring the path to be $q_s = \{1, 6, 3, 5, 2, 8, 10, 7\}$ (see Fig. 3).

### *4.2 Motor Friction Fault*

As a second scenario, we consider a motor friction fault in the back-left motor. The friction coefficient value is increased by a factor of $10$ at $50$ s. As a result, the motor current increases because its PID controller is still trying to maintain the wheel speed at the same value. This corresponds to an increase in the total current drawn from the batteries and an accelerated rate of discharge (and, thus, decreased RUL). The motor temperature also rises due to the increased current draw, and this can lead to an EOL condition of the motor due to the overheating. As a result, decision making will have to optimize RUL with respect to battery life and motor health.

QED detects the fault at $50.15$ s with an increase in $v_{BL}$, the velocity of the back-left wheel. The candidate set reduces to a failure of the back-left motor, increased friction in the back-left motor, and a bias or drift in the $v_{BL}$ sensor. At $50.25$ s, an increase in both $i$ and $i_{BL}$ are detected, eliminating the motor failure fault (which would have decreased the current) and the faults in the $v_{BL}$ sensor (since they cannot affect any other sensors), leaving the motor friction fault as the sole candidate. The friction value is computed using the steady-state wheel speed equation, averaged from the time of detection to the present time. At $50.25$ the friction value is estimated at $14.7$ times its nominal value. By $100$ s, the estimate is at $10.1$ times its nominal value. HyDE is also able to diagnose this fault at $50.2$. However, HyDE does not support fault identification.

Prognosis begins once the fault is identified. Here, multiple possible future input trajectories may be assumed that will help with decision making. The increase in current due to the fault will cause the batteries to discharge earlier than expected, and the increase in motor temperature may bring the motor to its EOL due to overheating. According to the prognosis algorithm, the overheating event occurs in $1423$ s if the rover continues to travel at the same speed and at $4084$ s at $80\%$ speed ($0.4$ m/s). At $60\%$ speed ($0.3$ m/s), the steady-state value of the temperature is below the motor temperature limit, so the rover can travel indefinitely without overheating the motor, and EOL is determined solely by battery end-of-discharge, which ends up being $6313$ s. However, if the differential speed controller is used, then the overall current draw decreases and motor overheating can be prevented. Furthermore, less overall current is drawn from the batteries for the same forward speed of the rover, resulting in improved RULs of $4283$ s at normal speed ($0.5$ m/s), $6004$ s at $0.4$ m/s, and $8897$ s at $0.3$ m/s.

If the non-robust LLC is used, the DM module considers traverses with different speeds ($0.3$, $0.4$, and $0.5$ m/s) in order to arrive at the optimal payoff. Traveling at $0.5$ m/s does not allow the rover to visit any of the other waypoints due to high energy draw. Traveling at $0.4$ m/s gets the rover to waypoints

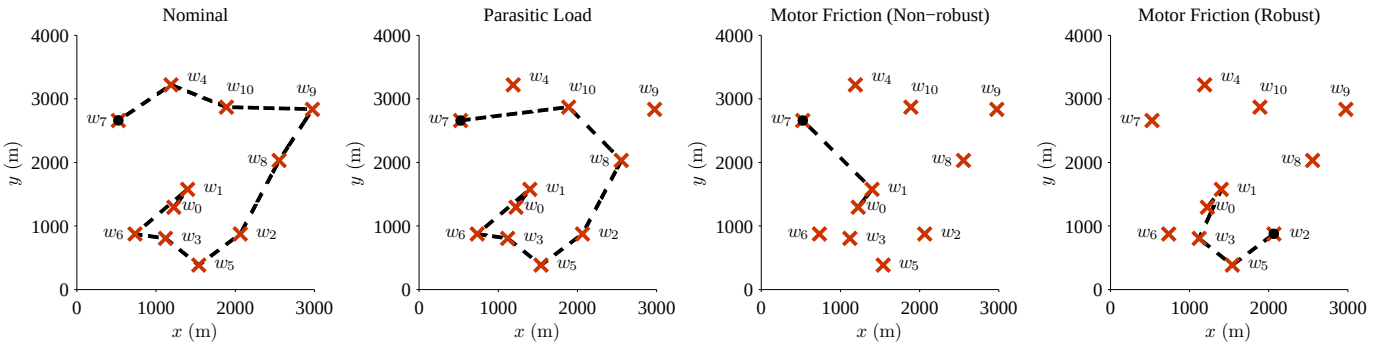



Fig. 3. Decision making results. Here, $r_1 = 4$, $r_2 = 5$, $r_3 = 3$, $r_4 = 2$, $r_5 = 6$, $r_6 = 4$, $r_7 = 10$, $r_8 = 5$, $r_9 = 3$, and $r_{10} = 6$.

3 and 6, with $R(q_s)$=11. Reducing the speed further to 0.3 m/s allows the rover to get to a higher-reward (but more difficult to get to) waypoint 7 instead ($R(q_s)$=14). Consequently, the rover speed is reduced to 0.3 m/s. Introducing the friction-robust LLC allows to keep the motor temperature within safe limits at any of the speeds considered and, due to the reduced power draw, makes it possible to accomplish longer traverses: $q_s = \{1, 2, 5\}$ at 0.5 m/s ($R(q_s)$=15); $q_s = \{1, 6, 3, 5\}$ at 0.4 m/s ($R(q_s)$=17); and $q_s = \{1, 3, 5, 2\}$ at 0.3 m/s ($R(q_s)$=18). Since travel time is not one of the optimization constraints, the lower speed of 0.3 m/s is therefore selected.

### *4.3 Voltage Sensor Fault*

As a third scenario, we consider a bias fault in the voltage sensor of battery 1. The bias has a value of 0.5 V and is injected at 50 s. QED detects the fault at 50.3 s, and, since no other faults can produce an increase in voltage, only bias and drift faults in the voltage sensor are valid candidates. At 50.9 s, the change is determined to be abrupt, isolating the bias fault as the candidate. The bias value is estimated to be 0.51 V. HyDE is able to detect a voltage sensor fault on Battery 1 at 50.1. However the HyDE model does not try to further isolate the fault as a bias or drift. Here, no prognosis is required, and the rover can still operate in the presence of the fault once it is identified. Therefore, the DM module does not need to modify the mission plan and the execution proceeds as planned.

In all of the above scenarios, the DM algorithms (dynamic programming, probabilistic, and exhaustive search) produced the same results. Both dynamic programming and probabilistic algorithms work well on relatively uncomplicated problems, however, some of their limitations become evident in more complex scenarios. A dynamic programming formulation becomes more challenging if multi-objective problems are posed, unless the objective functions lend themselves to aggregation into a single one. The probabilistic method, on the other hand, does not guarantee convergence to a global optimum in a finite amount of time and its ability to get close to the optimum is obviously dependent on the size of the search space. Nevertheless, it appears to be suited for multi-objective formulations.

## 5. CONCLUSIONS

During rover operations, when a fault occurs, it may require changes to low-level control, system usage, or the overall mission plan. In this paper, we described an integrated architecture for autonomous decision making, using diagnostic and prognostic information. The architecture is modular and allows different technologies for the LLC, DX, PX and DM modules. We demonstrated the approach on several scenarios in simulation.

In future work, we plan to further develop each module of the integrated architecture, and evaluate which algorithms perform best in each module of the architecture, according to a set of criteria which will include computation time and robustness to input noise, among others. PX and DM algorithms will be extended to better accommodate uncertainty in system modeling and prognostic predictions, allow handling of multiple degrading components, and incorporate system parameter/constraint optimization. The hardware testbed will also be further developed in order to demonstrate and validate our architecture in complex real-world scenarios.

## ACKNOWLEDGEMENTS

The funding for this research is provided by NASA ARMD System-wide Safety & Assurance Technology (SSAT) project.

## REFERENCES

Balaban, E., Narasimhan, S., Daigle, M., Celaya, J., Roychoudhury, I., Saha, B., Saha, S., and Goebel, K. (2011). A mobile robot testbed for prognostics-enabled autonomous decision making. In *Annual Conference of the Prognostics and Health Management Society 2011*, 15–30.

Daigle, M. and Goebel, K. (2011). Multiple damage progression paths in model-based prognostics. In *Proceedings of the 2011 IEEE Aerospace Conference*.

Daigle, M. and Roychoudhury, I. (2010). Qualitative event-based diagnosis: Case study on the second international diagnostic competition. In *Proc. of the 21st International Workshop on Principles of Diagnosis*, 371–378.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA.

Lachat, D., Krebs, A., Thueer, T., and Siegwart, R. (2006). Antarctica rover design and optimization for limited power consumption. In *4th IFAC Symp. on Mechatronic Systems*.

Mosterman, P.J. and Biswas, G. (1999). Diagnosis of continuous valued systems in transient operating regions. *IEEE Trans. on Sys., Man and Cybernetics, A*, 29(6), 554–565.

Narasimhan, S. and Browston, L. (2007). Hyde - a general framework for stochastic and hybrid modelbased diagnosis. In *Proc. DX07,*, 162–169.

Wolpert, D. (2006). Information theory–the bridge connecting bounded rational game theory and statistical physics. *Complex Engineered Systems*, 262–290.