Optimal Path Planning of a Rover over Complex Topography

Joe Mockler*
University of Maryland, College Park, MD, 20742

A global path is optimized knowing complete information about the local terrain and minimizing the end-to-end time for a mission. The vehicle's equations of motion are derived from first principles and framed such that the total time to traverse the path is calculable. Then, five characteristically different optimization schemes are compared for their ability to return an optimum in a computationally efficient manner; their potential use cases for different vehicle models or topography are commented on. Ultimately, all five schemes converge to a similar optimum, which shortens the mission time by 23 seconds compared to the straight-line connection over a 5x2.5 km map. Of these, the unconstrained scheme is most efficient with a computation time of about half that of the constrained schemes and is best suited for the smooth, convex problem framed here.

I. Nomenclature

R	=	0.5, Wheel radius [m]
m	=	170, mass of mars rover [kg]
g_{Mars}	=	3.71, gravity on mars [m/s^2]
C_D	=	0.33, coefficient of drag past a Mars rover [-]
μ_{RR}	=	0.6, coefficient of rolling resistance [-]
A_x	=	3, Cross-sectional area [m^2]
ρ_{mars}	=	0.0145, density of Mar's atmosphere [kg/m^3]
$\vec{x_f}$	=	coordinate of final position [m]
$\vec{x_0}$	=	coordinate of initial position [m]
v	=	velocity [m/s]
T_{in}	=	input motor torque [N]
H	=	height contour across an x-y plane [m]
$\vec{P}$	=	vector of the path through 3D space with N selected design points creating the path [m]
N	=	discrete number of potential design way-points in the path, P
r	=	used as a subscript, indicates the Rover's local coordinates
G	=	used as a subscript, indicates global coordinates

II. Introduction

This final project focuses on calculating the shortest mission time for a 4-wheeled vehicle over an arbitrary hilly contour. While the designed application is for planetary exploration, the presented framework can find applicability in search-and-rescue operations, transportation problems, or military and defense maneuvers. The problem requirements will be first formulated as high-level objectives to write out the minimization problem.

A. Problem Statement

Suppose a rover is driving over an arbitrary terrain has some timed requirement to complete a mission going from one point to another. If information about the topography is locally know, the qualitative question becomes: what is the fastest route to the end-point? The following assumptions about the rover and the terrain are made:

1) The rover is a simple 4-wheeled rover with reinforced polymer tires of radius R with a one-torque, τ, operation drive. The rover is driven by all 4 wheels and is rigid.

*Ph.D Student, Univ. of Maryland, Dept. of Aerospace Engineering

2) The rover has started with some initial velocity and will not change gears, torque, or perform other complex motion control during its mission.
3) The rover is traversing mars and the operation occurs over consistent terrain conditions. The problem may be framed on another planet or terrain by simply changing the environmental constants.
4) The rover is able to make sharp maneuvers (relative to the global mission) with ease while traversing the 3D environment.

With this, the problem can be described as the minimization of the total time to complete the mission, where the rover's path represents the continuous design space; shown later, the path is discretized into N design variables, each with a degree of freedom to optimize. The mission time is a function of the path $\vec{P}$, computed as the path integral along $\vec{P}$ with an N dimensional design vector of moveable points along the path. The rover cannot exit the known map, representing a constraint that bounds the path.

$$\min_{\vec{P}\in\mathbb{R}^N} t_m = \min_{\vec{P}\in\mathbb{R}^N} \int_{\vec{x_0}}^{\vec{x_f}} dt \tag{1}$$

$$\text{Subject to } \vec{P} \in [x_{lb}, x_{ub}] \text{ x } [y_{lb}, y_{ub}] \tag{2}$$

III. Methodology

A. Model Derivation

The equations of motion are derived from the simple rover model in 3D space. Typical vehicle models are more complex and constraining on the vehicle's motion, but in this problem, we assume the relative scales of the mission are sufficiently larger than the vehicle itself, and these complex constraints become negligible. To begin deriving the equations of motion, a free body diagram of the rover is drawn; see Fig. 1. Two coordinate systems are used: the rover's local coordinates and the global coordinates.

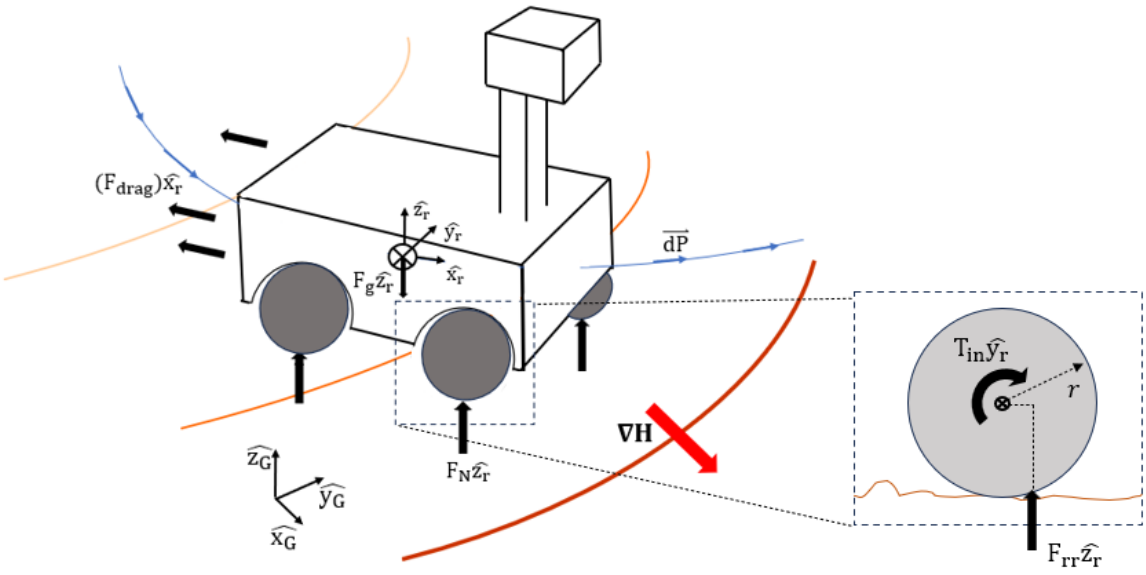

Fig. 1 Constructed free-body diagram of the rover to guide the model derivation. The forces are shown with black arrows, the red-colored lines are the planet's contour, and the blue line is the path the rover takes.

The vehicle mass and geometry are inspired from the Mars opportunity rover mission [1]. The primary acting forces are the DC motor torque, rolling resistance, aerodynamic drag, gravity, and the normal reaction from ground. Motion is constrained along $\hat{x_r}$ and the other degrees of freedom (y, z, roll, pitch, yaw) are considered negligible when planning the mission path. The DC motor torque is considered to be acting on all 4 wheels and applied to the wheel's rotating

axis. The force of gravity is applied to the body's center of mass and acts in $\hat{z_G}$; correspondingly, the normal reactions are applied on all 4 wheels that act in $\hat{z_r}$. Note that the contributions of gravity and the normal can be summed to write $-mg_{mars}\nabla H \cdot dP$, which is the sum of those two the forces acting in $\hat{x_r}$. If the gradient is negative (the rover's path traverses down a hill), the term adds to the rover's acceleration and vice versa. Aerodynamic drag acts oppositely along the vehicle path $\hat{x_r}$ in the local coordinates [2] [3]. Finally, rolling resistance is considered by offsetting the normal slightly in the direction of motion, accounting for the deformation of a tire as it rolls [4]. By rewriting the moment, this becomes a force acting negatively as $\mu_{RR}mg$ in the direction of motion. While this model has ignored more complicated forms of wheel-ground interaction (time-varying tire deformation, slip, nonlinear behaviors, complex drag phenomena), a more detailed treatment may eventually be incorporated into the optimization model [5].

$$m\vec{v} = \Sigma\vec{F} = (4T_{in}r - 4\mu_{RR}mg - 0.5\rho_{mars}C_D v_x^2 - g_{mars}\nabla H_G \cdot dP_G)\hat{x_r} + 0\hat{y_r} + 0\hat{z_r} \tag{3}$$

To write the optimization problem in an appropriate form for minimization, the time along the path must be solved. Because the forces in the $\hat{y_r}$ and $\hat{z_r}$ directions are 0, the vector problem is simplified to just one dimension. To ultimately write the total time along the path, velocity is first integrated, then a substitution is made for dt with the constraint $dt = dx/v$.

$$dv = \frac{dx}{vm}(4T_{in}r - 4\mu_{RR}mg - 0.5\rho_{mars}C_D v_x^2 - g_{mars}\nabla H_G \cdot dP_G) \tag{4}$$

To simplify, V on the RHS is approximated as V_0, which upon inspection of the numerical solution to the equation, is generally held within 10%. This approximation allows equation to be easily integrated along the path P, and thus, makes the assumption reasonable for this exercise in optimization. Upon integration, the final formula for the rovers velocity along $\hat{x_r}$ is produced and may be transformed into the global coordinates for a final path integration.

$$v(\vec{x}) = \int_{\vec{x_0}}^{\vec{x_f}} \frac{1}{v_0 m}(4T_{in}r - 4\mu_{RR}mg - 0.5\rho_{mars}C_D v_0^2 - g_{mars}\nabla H \cdot dP_G)d\vec{x} + v_0 \tag{5}$$

To express the total travel time as an path integral, $v = dx/dt$ is modified to compute the path integral using the calculated $v(\vec{x})$ from above. The final minimization problem is written below, where the velocity is integrated along the 3D path through space. The problem is constrained to placing design points on the map represented by Cartesian lower and upper bounds on the path, $\vec{P}$; see Eq. (2).

$$\min_{\vec{P}\in\mathbb{R}^N} t_{mission} = \min_{\vec{P}\in\mathbb{R}^N} \int_{\vec{x_0}}^{\vec{x_f}} dt = \min_{\vec{P}\in\mathbb{R}^N} \int_{\vec{x_0}}^{\vec{x_f}} \frac{d\vec{x}}{v(\vec{x})} \tag{6}$$

$$\text{Subject to } \; y_{lb} \leq \vec{P}_y \leq y_{ub} \quad x_{lb} \leq \vec{P}_x \leq x_{ub} \tag{7}$$

B. Numerics

This section will briefly describe the methods of calculating the path integral numerically; all work is performed in MATLAB. First, a contour is randomly generated and smoothed over a user-defined map (here, longitude, x by latitude, y is 5000 m by 2500 m). The path is then established along the 2D grid by placing points equally spaced in the x direction and letting the y direction vary. This is used to constrain the optimizer's allowable degrees of freedom, and preventing the solution from quickly exploding, but still lets it take an arbitrary shape in the space. Thus, the optimizer has N dimensions to explore, where N is the number of design points in $\vec{P}$, rather than N^2. A linear connection is then drawn between the points to create the complete "path" of the rover with 1000 equally-spaced points interpolated along the path. Differentiating then creates the vector dP along the path. To compute the terms in Eq. (5), the created vector can be dotted with the interpolated gradient information along the path to get the gravitational effects. The other terms are essentially scalars (by approximating $v_{local} \approx v_0$) that can dotted along the path. Finally, the expression $d\vec{x}/v(\vec{x})$ is numerically integrated along the entire path by dotting the reciprocal of the velocity vector along the path with the dP of the path itself and summing over the 1000 interpolated points; this final mission time is the calculated objective.

C. Optimization Schemes and Selected Parameters

In this optimization problem, the routine must solve an N-dimensional, nonlinear problem, where N represents the number of design variables from which the path is interpolated. The optimization will return the y-positions of the

N points along the map, recalling that the x dimension is constrained to reduce the computational complexity. The objective is a smooth function because the primary contribution is the interaction of the path with the contour, and consequently, the gradient and hessian of the object may be numerically approximated and used during the optimization. While the problem is technically constrained to the known local map, these constraints are unlikely to be active because running to the edge of the map would unfavorably increase the mission time.

An unconstrained algorithm will likely be the most efficient optimizer for this problem, but a constrained problem may be applicable if constrains were active (e.g. an area of the map must be avoided). A gradient free method might be advantageous if the vehicle model allowed for discontinuities or non-linearities, making the objective not smooth. Considering the range of potential cases of vehicle motion or topography, a few different optimization techniques are studied here.

The three constrained method studied are: interior points, sequential quadratic programming, and active-set; the unconstrained method employs a quasi-Newton algorithm; and the gradient-free method tested is simulated annealing. Ultimately, simulation time and mission time will be compared for the different schemes when the number of path points is varied. A more comprehensive description of the specific algorithms and parameters are in the appendix. This project uses the MATLAB implementation of each.

In the unconstrained case, the optimality condition is written as the $||\nabla f||_\infty$, which is numerically approximated with a finite difference. The constrained cases use KKT optimality conditions, written as the $||\nabla_x L(x,\lambda)||_\infty$. In both of these cases, stopping occurs if the tolerance dips below 1e-4 or 500*N function evaluations are reached. The simulated annealing will similarly stop below 1e-4 normalized changed in the objective function or if 3000*N function evaluations are reached. Elaborated descriptions of each algorithm's conditions are found in the appendix.

IV. Results

The following parameters were initialized for a first-pass minimization: $N = 6, V_0 = 10$[m/s], map = 5000 x 2500 [m x m], average contour height = 150 [m]. Each optimization algorithm is first initialized with a straight-line between the start and end point. This straight path is typically the shortest time path between two points in space and serves as the baseline of comparison. Then, the optimization routine is run and timed; the results are plotted below in Fig. 2 and the calculated mission time and the optimization computation time are summarized in Table 2. Each scheme shows a **21 sec improvement** over the 540 sec baseline and all follow a nearly identical trajectory.

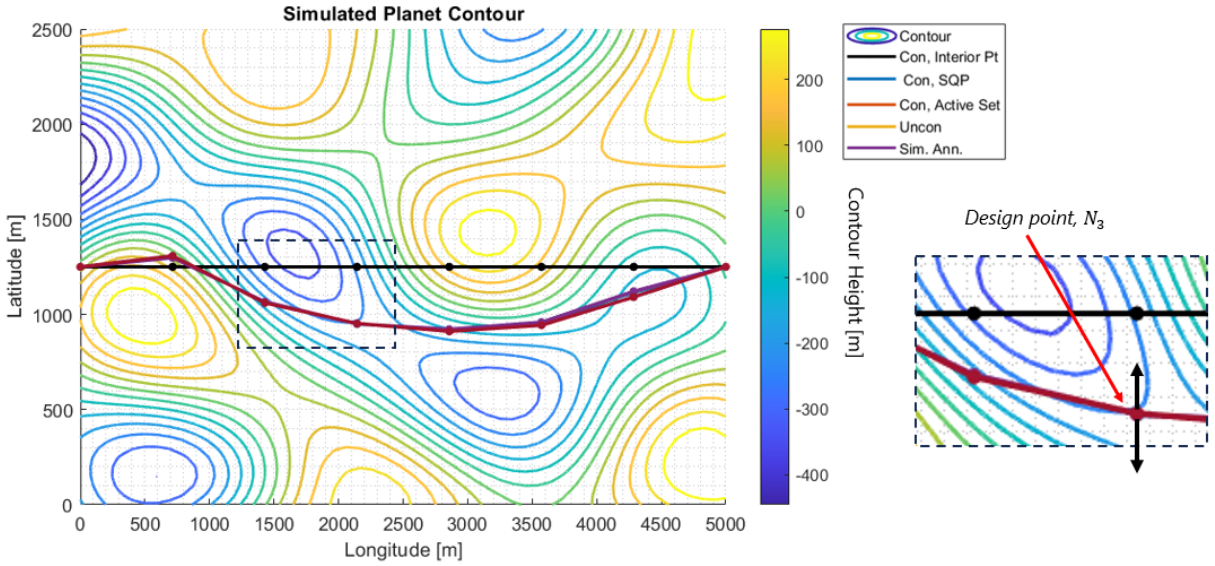

Fig. 2 The optimal points for N=6 variables, which excludes the endpoints at x = 0 and x = 5000, are shown with dots while the interpolated path is the connected line. The detailed cutout (boxed, right) shows the 3rd design point and its degree of freedom as an example. Each optimization method converged on nearly the exact same optimal path and are overlapping in the figure.

But are N=6 points sufficient to create the most optimal path? N = 3,4,...,15 design points are optimized and the mission time is calculated for each algorithm; V_0 is held at 10 [m/s]. All five methods converged to similar mission times and all showed improvements to the mission time. The improvement time increased with N, but showed diminishing returns and stabilizing to 23-24 sec at N = 10 or 11. These plots are found in the appendix. For more physical intuition in the optimizer's strategy, some intermediate points of the unconstrained optimizer for N = 10 points are plotted. See Figure 3 for details.

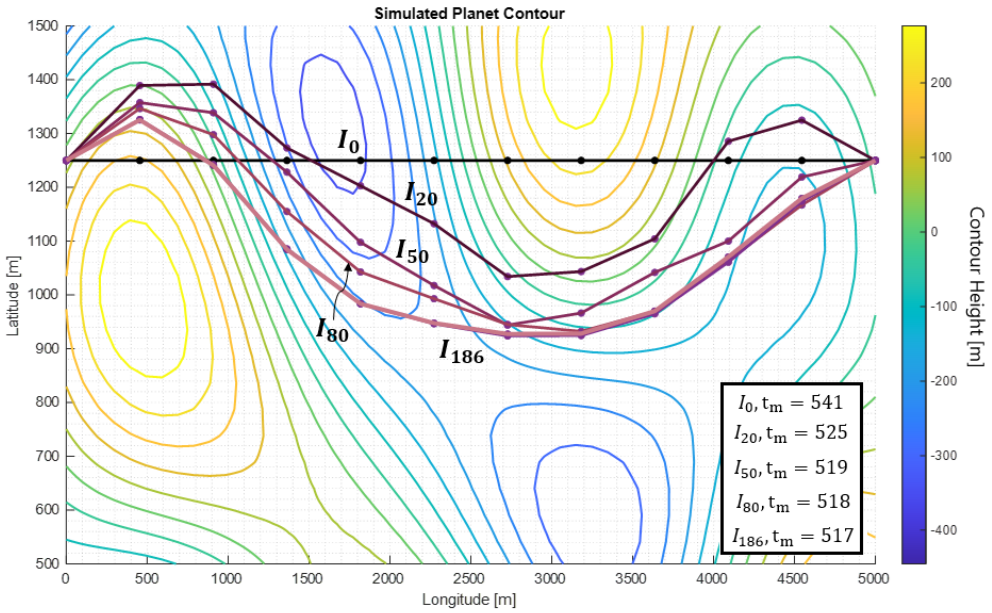

Fig. 3 The unconstrained optimizer is run with N = 10 points. Some intermediate paths during the optimization process are plotted to show the physical path the optimizer takes to arrive at the shortest time path (I_m represents the mth iteration). The mission times at the plotted iterations are listed in the bottom, right.

V. Conclusions

The optimization is clearly shortening the mission time, which provides a critical advantage in time-sensitive missions like search-and-rescue or planetary exploration. This minimization scheme shortens a 5 km mission time by 25 seconds for a rover travelling nominally at 10 [m/s].

As the optimizer runs, the path begins to snake through the contours and finds the "flattest" path between the hills and valleys; if the rover must traverse a gradient, it does along the normal. Inspection of the objective generally corroborates this, where of all the terms in the objective Eq. (5) and Eq. (6)), the minimization is largely driven by the $d\vec{x}$ term and the gradient term. Thus, the minimization will balance these two effects and find a path that maintains a roughly flat profile, minimizing $d\vec{x}$, while also leveraging decreasing gradients where applicable. Compare this to Fig. 2 and 3. The straight line traverses two major hills, adding substantial time to the mission, while the optimal path avoids large gradients.

By running five different optimization algorithms, we gain confidence in achieving global optimality with each method. While the unconstrained method is best-suited here, as the map constraints were never active, an analyst may use a constrained routine if areas of the environment are inaccessible (e.g. soft soil, snow, poorly-characterized) by constraining specific regions. If the rover is designed for more complicated dynamics like gear changes, variable power, time-varying coefficients, then a gradient-free solver would be best suited to accept the non-smooth objective. Of course, these come at a high computational cost, and should be considered by an analyst. This paper demonstrates the success of a path-planning formulation that minimizes travel time.

Appendix

A. Description of the Optimization Algorithms

1) Interior Points - For a convex, constrained optimization problem, the problem is written as an unconstrained optimization problem with barrier functions at the constraints. This algorithm breaks this down into a sub-optimization problem, where the KKT equations are satisified and the algorithm pushes the solution into a feasible range via a barrier [6]. The method is solved with using MATLAB's `fmincon()` [7].
2) Sequential Quadratic Programming - Similar to Newton's method, where a the problem is broken into quadratic sub-problems that are optimized; these quadratic sub-problems incorporate the constraint into a merit function that is optimized with less strict constraints. The process is iterated until the KKT equations are satisfied [8]. The method is solved with using MATLAB's `fmincon()` [7].
3) Active Set - The algorithm identifies active constrains and converts these inequalities into equality constraints, transforming into an easier problem. The method tries to satisfy the KKT equations. [9]. The method is solved with using MATLAB's `fmincon()` [7].
4) Quasi-Newton Unconstrained - A classic optimization scheme where the gradient and hessian are computed and used to select and move in a search direction towards the optimum. The algorithm uses a BFGS Quasi-Newton method with a cubic line search procedure to advance the search. The method is solved with using MATLAB's `fminunc()` [10].
5) Simulated Annealing (derivative-free method) - this gradient-free technique uses a probability estimate, in conjunction with the function at existing points, a new point and slowly work towards the global optimum. The method is solved with using MATLAB's `simulannealbnd()` [11]

Table 1 Relevant MATLAB Optimization Parameter Options. * Denotes a non-default setting

Parameter	Interior Point	SQP	Active Set	Uncon. Quasi-Newton	Sim. Annealing
`Max Fcn Evals*`	500N	500N	500N	500N	3000N
`Max Iters`	500	500	500	500	Inf
`Optim Tol*`	1e-4	1e-4	1e-4	1e-4	1e-4
`Step Tol`	1e-10	1e-10	1e-10	1e-10	1e-10
Gradient Formulation	Finite Fwd	Finite Fwd	Finite Fwd	Finite Fwd	-
Hessian Formulation	BFGS	quasi-Newton	quasi-Newton	Finite Difference	-
Barrier Update	Monotone Scheme	-	-	-	-
Cooling Schedule	-	-	-	-	$T = T_0 * 0.95^k$

B. Optimization Results Supporting Tables and Figures

Table 2 A summary of mission times and computation times across the tested methods when N=6

Optimization Method	Mission Time [s]	Improvement over Straight Path [s]	Computation Time [s]
Constrained, Interior Pt	519.24	21.47	114.39
Constrained, SQP	519.24	21.47	146.64
Constrained, Active-Set	519.24	21.47	84.08
Unconstrained, Quasi-Newton	519.24	21.47	60.82
Simulated Annealing	519.37	21.34	730.99

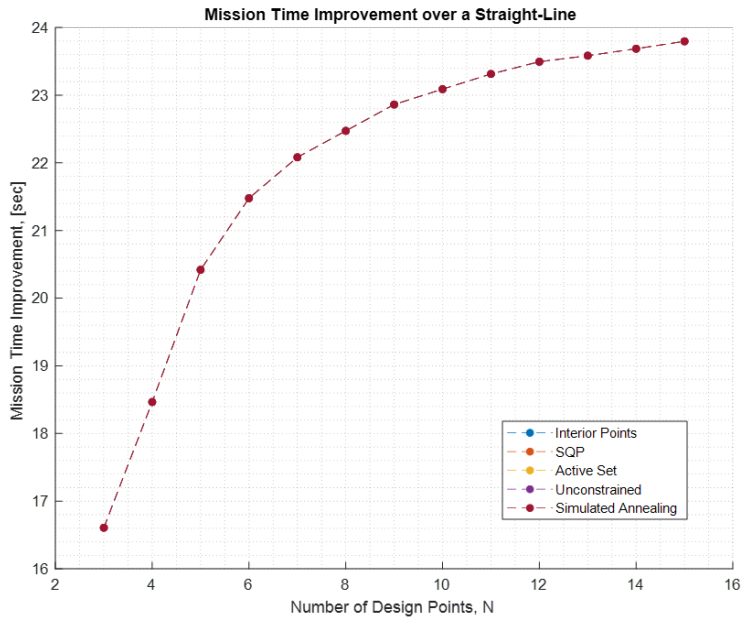

Fig. 4 The optimization routine was run for all 5 algorithms where the number of design variables, N, was varied. The plateauing behavior shows that increased discretization approaches the continuous ideal path. All algorithms converged to a nearly identical optimal path

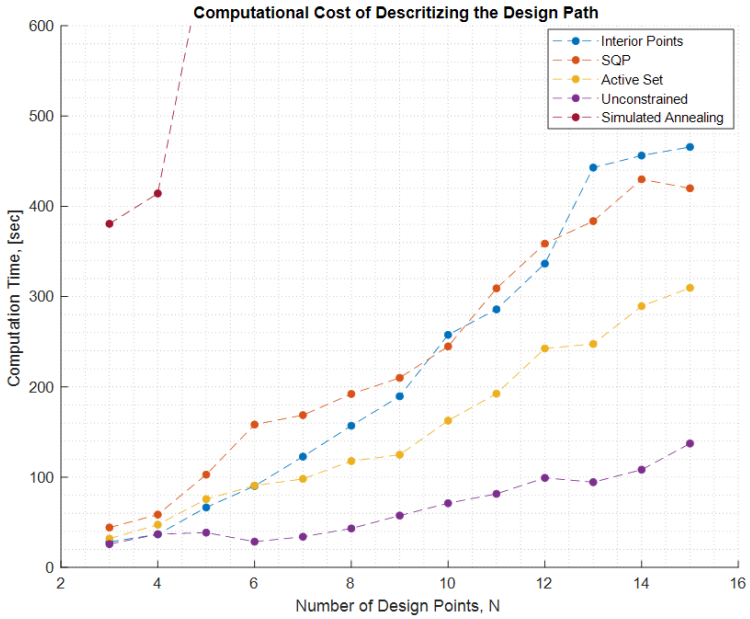

Fig. 5 The optimization routine was run for all 5 algorithms where the number of design variables, N, was varied. While Fig. 4 shows mission time improvement with increased N, the computational cost also rises. Simulated annealing was costly, with some minimizations taking thousands of seconds. The plot is trimmed to show detail of the other algorithms. Some noise is from MATLAB allocating CPU resource ad hoc.

References

[1] Squyres, S., "Mars Exploratory Rover Project," *NASA/JPL Joint Project Report*, 2001.

[2] S. Chaligne, R. T., and Gaylard., A., "The Aerodynamics Development of the New Land Rover Discovery," 2017.

[3] T. Liu, A. O., and Fujii, K., "Scaling Analysis of Propeller-Driven Aircraft for Mars Exploration," *Journal of Aircraft*, Vol. 50, No. 5, 2013, pp. 1593–1604.

[4] N. Costes, J. F., and Geoerge, E., "MOBILITY PERFORMANCE OF THE LUNAR ROVING VEHICLE: TERRESTRIAL STUDIES - APOLLO 15 RESULTS," *NASA Technical Report N. 401*, 1972.

[5] Villella, M. G., "Nonlinear Modeling and Control of Automobiles with Dynamic Wheel-Road Friction and Wheel Torque Inputs," *M.S Dissertation*, 2004.

[6] Nocedal, F. , Jorge, and Waltz, R. A., "An Interior Point Method for Nonlinear Programming with Infeasibility Detection Capabilities," *Optimization Methods Software*, Vol. 29, No. 4, 2014, pp. 837–854.

[7] MathWorks, "MATLAB Optimization Toolbox Documentation: fmincon()," 2023. URL `https://www.mathworks.com/help/optim/ug/fmincon.html#busog7r-2`.

[8] Nocedal, J., and Wright, S. J., "Numerical Optimization," *Springer Series in Operations Research*, 2006.

[9] Nocedal, J., and Wright, S., "Numerical Optimization," 2006.

[10] MathWorks, "MATLAB Optimization Toolbox Documentation: fminunc()," 2023. URL `https://www.mathworks.com/help/optim/ug/fminunc.html`.

[11] MathWorks, "MATLAB Optimization Toolbox Documentation: simulannealbd()," 2023. URL `https://www.mathworks.com/help/gads/simulated-annealing-options.html`.