MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

A Comparative Study of MINLP and MPVC Formulations for Solving Complex Nonlinear Decision-Making Problems in Aerospace Applications

Ghezzi, Andrea; Nurkanović, Armin; Weiss, Avishai; Diehl, Moritz; Di Cairano, Stefano

TR2026-024 February 21, 2026

Abstract

High-level decision-making for dynamical systems often involves performance and safety specifications that are activated or deactivated depending on conditions related to the system state and commands. Such decisionmaking problems can be naturally formulated as optimization problems where these conditional activations are regulated by discrete variables. However, solving these problems can be challenging numerically, even on powerful computing platforms, especially when the dynamics are nonlinear. In this work, we consider decision-making for nonlinear systems where certain constraints, as well as possible terms in the cost function, are activated or deactivated depending on the system state and commands. We show that these problems can be formulated either as mixed-integer nonlinear programs (MINLPs) or as mathematical programs with vanishing constraints (MPVCs), where the former formulation involves discrete decision variables, whereas the latter relies on continuous variables subject to structured nonconvex constraints. We discuss the different solution methods available for both formulations and demonstrate them on optimal trajectory planning problems in various aerospace applications. Finally, we compare the strengths and weaknesses of the MINLP and MPVC approaches through a focused case study on powered descent guidance with divert-feasible regions. In our simulations for problems up to medium size, MPVC formulations provide accurate solutions faster than MINLP formulations. However, for larger problems, the MPVC formulation introduces numerous nonconvexities that hinder solver convergence, even when they are relatively simple, making MINLPs the preferred choice in such cases.

Optimal Control Applications and Methods 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

A Comparative Study of MINLP and MPVC Formulations for Solving Complex Nonlinear Decision-Making Problems in Aerospace Applications

Andrea Ghezzi[1] | **Armin Nurkanović**[1] | **Avishai Weiss**[2] | **Moritz Diehl**[3] | **Stefano Di Cairano**[2]

[1]Department of Microsystems Engineering (IMTEK), University of Freiburg, Germany

[2]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA

[3]Department of Microsystems Engineering (IMTEK) and Department of Mathematics, University of Freiburg, Germany

Correspondence
andrea.ghezzi@imtek.uni-freiburg.de

Abstract

High-level decision-making for dynamical systems often involves performance and safety specifications that are activated or deactivated depending on conditions related to the system state and commands. Such decision-making problems can be naturally formulated as optimization problems where these conditional activations are regulated by discrete variables. However, solving these problems can be challenging numerically, even on powerful computing platforms, especially when the dynamics are nonlinear. In this work, we consider decision-making for nonlinear systems where certain constraints, as well as possible terms in the cost function, are activated or deactivated depending on the system state and commands. We show that these problems can be formulated either as mixed-integer nonlinear programs (MINLPs) or as mathematical programs with vanishing constraints (MPVCs), where the former formulation involves discrete decision variables, whereas the latter relies on continuous variables subject to structured nonconvex constraints. We discuss the different solution methods available for both formulations and demonstrate them on optimal trajectory planning problems in various aerospace applications. Finally, we compare the strengths and weaknesses of the MINLP and MPVC approaches through a focused case study on powered descent guidance with divert-feasible regions. In our simulations for problems up to medium size, MPVC formulations provide accurate solutions faster than MINLP formulations. However, for larger problems, the MPVC formulation introduces numerous nonconvexities that hinder solver convergence, even when they are relatively simple, making MINLPs the preferred choice in such cases.

KEYWORDS
decision-making for dynamical systems, optimal control, mixed-integer nonlinear programming, mathematical programming with vanishing constraints

1 | INTRODUCTION

We consider constraint-triggered optimization problems described using logical expressions

$$\min_{z \in \mathbb{R}^{n_z}} \quad f(z) - \sum_{i=1}^{n_\delta} w_i \sigma(H_i(z)) \tag{1a}$$

$$\text{s.t.} \quad g(z) = 0,\ h(z) \leq 0, \tag{1b}$$

$$H_i(z) \geq 0 \Rightarrow G_i(z) \leq 0, \quad i \in \mathbb{Z}_{[1,n_\delta]}, \tag{1c}$$

where σ is the Heaviside step-function $\sigma : \mathbb{R} \to \{0, 1\}$,

$$\sigma(y) \coloneqq \begin{cases} 1, & \text{if } y \geq 0, \\ 0, & \text{if } y < 0. \end{cases} \tag{2}$$

In (1), $w_i \geq 0$ are weight coefficients for the second term in the objective function, and functions $f : \mathbb{R}^{n_z} \to \mathbb{R}$, $g : \mathbb{R}^{n_z} \to \mathbb{R}^{n_g}$, $h : \mathbb{R}^{n_z} \to \mathbb{R}^{n_h}$, $H_i : \mathbb{R}^{n_z} \to \mathbb{R}$, $G_i : \mathbb{R}^{n_z} \to \mathbb{R}^{n_{G_i}}$ are assumed to be twice continuously differentiable. Based on the chosen representation of the Heaviside step-function in the cost (1a), and of the logical expression in the constraint (1c), the mathematical program (1) can be formulated as a mixed-integer nonlinear program (MINLP) or a mathematical program with vanishing constraints (MPVC). The convention $\sigma(0) = 1$ makes (2) an upper semi-continuous function, thus the minimization problem (1) is well-defined [1, §27-28]. In some applications (cf. Sec. 2.2), we consider the left-hand-side of constraint (1c) to hold with strict inequality "$H_i(z) > 0$". Strict inequalities define an open feasible set which might cause some issues in view of the existence of solutions of problem (1). However, the MINLP and MPVC formulations of (1) presented in Sec. 3 will lead to well-defined problems. The distinctive feature of (1) lies in the maximization of the Heaviside step-function within the cost. By tuning the coefficients w_i, one trade-offs between finding optimal solutions to the nonlinear program (NLP) with cost function f and constraints (1b), and the enforcement of additional indicator constraints expressed by functions H_i, that in turn imply constraints expressed by functions G_i.

A special case yet important case of (1) is when indicator constraints $H_i(z) \geq 0$ corresponds to indicator variables. Consider the optimization variable $\tilde{z} = (z, \delta) \in \mathbb{R}^{n_z+n_\delta}$ such that $H_i(\tilde{z}) = \delta_i$ and $\delta_i \in [0, 1]$, for all $i \in \mathbb{Z}_{[1,n_\delta]}$, thus (1) can be written as

$$\min_{(z,\,\delta) \in \mathbb{R}^{n_z+n_\delta}} \quad f(z) - \sum_{i=1}^{n_\delta} w_i \delta_i \tag{3a}$$

$$\text{s.t.} \quad g(z) = 0,\ h(z) \leq 0, \tag{3b}$$

$$\delta_i > 0 \Rightarrow G_i(z) \leq 0, \quad i \in \mathbb{Z}_{[1,n_\delta]}, \tag{3c}$$

$$0 \leq \delta_i \leq 1, \quad i \in \mathbb{Z}_{[1,n_\delta]}, \tag{3d}$$

Problem (3) does not contain the Heaviside step-function from (3a), making (3) easier to treat than (1). A similar remark to one above can be made for the strict inequality in (3c). Additionally, we will show that optimal solution of both MINLP and MPVC reformulation of (3) are characterized by $\delta_i \in \{0, 1\}$, for all $i \in \mathbb{Z}_{[1,n_\delta]}$.

In this work, we introduce optimal trajectory planning problems for aerospace applications that can be modeled by (1) and (3), and demonstrate how the resulting optimization problems are solved.

Related work – Mathematical programs with indicator constraints, or "on/off" constraints, such as (1c) have been thoroughly studied in the operation research literature [2, 3, 4], and it is well-known that logical implications can be represented either via disjunctive programming [5, 6] or complementarity and vanishing constraints [7, 8]. Among the disjunctive programming approaches, the most common and easiest to implement is the *big-M* method [9]. However, the *big-M* method leads to weak relaxations [3]. Other approaches for representing disjunctions are based on perspective formulations [10, 11] and lead to tailored algorithms such as branch-and-cut [12, 13]. Nevertheless, approaches based on the perspective formulation require constructing the convex hull of the disjunction, which can be computed analytically only for specific sets [14, 2, 3]. If the analytical expression is not available, the convex hull might be constructed iteratively via cutting planes as in [12, 13], thus the resulting cutting planes have to be integrated in a branch-and-cut algorithm. The iterative procedure to obtain cutting planes for the convex hull might require similar computation time to that of the big-M formulation. The approach of using complementarity constraints for representing logical implications requires the introduction of structured nonconvexities that violate standard constraint qualifications, such that even robust NLP solvers might fail to converge to a solution [15, 16], [17, §9.3]. In practice, the use of complementarity constraints is advised for problems that possess a connected feasible set. Despite the numerical issues related to the solution of problems with complementarity constraints, a major advantage is that they require the solution of NLPs only, thus there is no reliance on mixed-integer programming (MIP) solvers.

In the context of this paper, (1) arises from the time-discretization of an optimal control problem (OCP) via a direct method, e.g., direct multiple shooting [18] or direct collocation [19]. This type of optimization problem aims at finding the control inputs to a dynamic system that produce the best possible performance over a specified time horizon, subject to constraints on the system state and controls. Similar to this work, in [20] the authors present a MINLP and a MPVC formulation of a mixed-integer OCP whose discrete part is not limited to the representation of logical expressions but also to system states and controls. The focus

of [20] relies in the comparison of the relaxations obtained from the two alternative formulations. In this work, we focus on OCPs that are continuous except for the indicator variables, and we aim at comparing computational methods and quality of the final solutions for both MINLP and MPVC formulations. In Section 2, we specify (1) as an OCP to provide a reference formulation.

The main case study in this paper is the powered descent phase in Mars landing [21, p. 87]. In powered descent guidance (PDG), optimal control and numerical optimization have been largely adopted for computing time- and fuel-optimal trajectories [22]. The method proposed in [22] formulates PDG as a convex problem that can be solved in milliseconds to global optimality [23]. The convexification procedure has been further extended over the years to allow for more flexible models; for an up-to-date list of references, we refer the reader to the tutorial article [21]. The convexification adopts a 3 degrees of freedom (DoF) point mass model, which is generally considered accurate enough for this application. Despite the numerous enhancements to the convex reformulation, at the current status it is not possible to obtain a "lossless convexification" in the case of continuous activation of constraints [21], i.e., if an optimal solution has active path constraints in consecutive time steps. Also, if the objective of the problem is modified, such that it no longer exclusively seeks fuel- and time-optimal trajectories, a new tailored convexification must be developed which is not guaranteed to exist. To avoid the pitfalls of the convexification approach, recent works have considered directly working with a nonlinear 6 DoF model of the spacecraft. In [24], the authors show that a sequential convexification approach can efficiently solve the PDG problem while incorporating additional complexities such as discrete actuation. Moreover, in [25], the authors propose a method based on convex-concave decomposition that enables the direct handling of nonlinear equality constraints, i.e., the 6 DoF dynamic equations, within an otherwise convex problem. The approach in [25] can also embed discrete actuation and, unlike [24], does not require smoothing of the discrete variables or the use of a homotopy loop. In this work, we address a PDG problem augmented with divert-feasible regions. The spacecraft must approach the landing site while traversing the maximum number of these regions, areas from which alternative landing sites are reachable, thereby maximizing the "divert possibility". Such problem can be formulated as (1) which, to the authors' knowledge, cannot be losslessly convexified, and thus must be addressed as is, either as a MINLP or a MPVC.

Contribution – The main contribution of this paper is the development of a tractable modeling and solution framework for a class of constraint-triggered optimization problem as (1), with a focus on optimal control problems involving state-dependent logical constraints and cost terms. While building on established techniques such as the big-M technique and vanishing constraints, we apply them, to the best of our knowledge, for the first time in this context to make formulation (1) numerically tractable as a MINLP or a MPVC. A key technical contribution is the treatment of the Heaviside step function, enabling its incorporation in problems solvable by Newton-based optimization methods. Moreover, we show that several aerospace-relevant problems can be expressed using (1), demonstrating the practical relevance of the proposed formulation. A particularly comprehensive case study is the PDG with divert-feasible regions for Mars landing, a problem that is currently the subject of active research due to the renewed interest in Mars exploration, and for which we propose a formulation enabled by nonlinear modeling. Extensive numerical simulations illustrate the effectiveness of the approach and offer guidance on when a MINLP or a MPVC formulation is more appropriate.

Outline – In Section 2, we specialize formulation (1) for OCPs, and introduce relevant aerospace case studies that can be modeled as (1) and (3). Section 3 shows how to obtain a numerically tractable expression for the logical implication (1c) and for the Heaviside-step function (2). Moreover, we present two alternative formulations of both (1) and (3): one resulting in a MINLP and one in a MPVC. In Section 4, we survey some solution methods for MINLPs and MPVCs, and compare them for a tutorial example. Section 5 describes a case study of the PDG problem constrained with divert-feasible regions, obtains both the MINLP and MPVC formulations, and solves them with the presented solution methods. Finally, Section 6 contains concluding remarks and possible future research.

Notation – The set set of positive real numbers is denoted as $\mathbb{R}_+$. With $\mathbb{Z}_{[a,b]}$, $a < b$, we denote the interval of integers $\{a, a+1, \dots, b\}$. Vector inequalities are intended component-wise. The symbol $\Rightarrow$ denotes the logical implication, σ the Heaviside step-function, and $\|\cdot\|_2$ the vector Euclidean norm. Given two variables a, b, we write a complementarity condition using the symbol $\perp$ as $0 \leq a \perp b \geq 0$, i.e., it must hold either $a \geq 0, b = 0$, or $a = 0, b \geq 0$.

2 | DECISION-MAKING VIA OPTIMAL CONTROL

In this section, we present trajectory planning case studies relevant in the aerospace domain that can be formulated as (1). Each trajectory planning task is formulated as an optimal control problem (OCP) that is solved via a direct method. First, we introduce how to specialize the generic formulation (1) to an OCP formulation. Then, we present the trajectory planning problems.

2.1 | Discrete-time optimal control problem

We consider dynamical systems modeled via ordinary differential equations (ODEs)

$$\dot{x}(t) = f(x(t), u(t)),$$

where t denotes the time, $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ the system state and control vectors, respectively. To optimally control the system, we formulate an OCP over a time interval $t \in [0, t_\mathrm{f}]$ that encodes performance goals via the cost function, and safety specifications via constraints. In order to numerically solve such OCP, we adopt direct multiple shooting [18]. By means of a suitable integration method, e.g., a Runge-Kutta integrator, we discretize the ODEs, the cost function, and the constraints over a uniform grid with N intervals where $0 = t_0 < t_1 < ... < t_N = t_\mathrm{f}$. The discretization step is denoted by t_d and is defined as $t_\mathrm{d} \coloneqq t_\mathrm{f}/N$. The final time t_f may be included as optimization variable to formulate time-optimal problems. After the discretization step, we obtain a mathematical program as (1) but with a specific OCP structure,

$$\min_{t_\mathrm{f}, \mathbf{x}, \mathbf{u}} \quad E(x_N, t_\mathrm{f}) + \sum_{k=0}^{N-1} \left(L(x_k, u_k, \tfrac{t_\mathrm{f}}{N}) - \frac{t_\mathrm{f}}{N} \sum_{i=1}^{n_\delta} w_{i,k} \sigma(r_i(x_k, u_k)) \right) \tag{4a}$$

$$\text{s.t.} \quad x_0 = \bar{x}_0, \tag{4b}$$

$$x_{k+1} = F(x_k, u_k, \tfrac{t_\mathrm{f}}{N}), \qquad k \in \mathbb{Z}_{[0,N-1]}, \tag{4c}$$

$$c(x_k, u_k) \leq 0, \qquad k \in \mathbb{Z}_{[0,N-1]}, \tag{4d}$$

$$r_i(x_k, u_k) \geq 0 \Rightarrow s_i(x_k, u_k) \leq 0, \quad i \in \mathbb{Z}_{[1,n_\delta]}, k \in \mathbb{Z}_{[0,N-1]}, \tag{4e}$$

$$c_N(x_N) \leq 0, \tag{4f}$$

where $\mathbf{x} \coloneqq (x_0, \ldots, x_N)$ and $\mathbf{u} \coloneqq (u_0, \ldots, u_{N-1})$ are the stage-wise concatenation of the state and control variables, respectively. For the simplicity of notation, we do not consider implication constraints involving the terminal state, but the extension to handle this case is straightforward. The cost function (4a) includes a terminal cost $E : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}$ and a stage cost with two distinct terms. The first term $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}$ depends only on state, control, and discretization step. The second term is a weighted sum of the Heaviside step-functions. Function $F : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is obtained from a suitable integration method that discretizes the corresponding ODE, e.g., a Runge-Kutta integrator. Function $c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_c}$ defines nonlinear path constraints, enforced at grid nodes, and $c_N : \mathbb{R}^{n_x} \to \mathbb{R}^{n_{c_N}}$ expresses a terminal constraint. Functions $s_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_s}$ construct constraints that are enforced only when the corresponding triggering function $r_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is satisfied. Similarly to (3), it is straightforward to obtain a special case for (4) when the indicator constraints $r_i(x_k, u_k) \geq 0$ corresponds to indicator variables $\delta_i \in [0, 1]$. We consider formulation (4) as a template for the problems presented in the following subsections.

2.2 | Aerospace trajectory planning case studies

Next, we describe four trajectory planning problems. The first two correspond to the special case of (4), as they lack indicator constraints defined by functions r_i and instead use indicator variables $\delta_i \in [0, 1]$.

2.2.1 | Powered descent guidance with divert-feasible trajectories

We aim at computing landing trajectories for a spacecraft that in addition to minimizing flight time and fuel usage, approach the primary landing site by traversing for as much time as possible divert-feasible regions, i.e., regions from which the spacecraft can divert to alternative landing sites. These divert-feasible regions may be represented by polytopes obtained by reachability analysis [26]. We define $p_k \in \mathbb{R}^3$, $v_k \in \mathbb{R}^3$, where $k \in \mathbb{Z}_{[0,N-1]}$, as the 3D position and velocity of the spacecraft in an inertial reference frame with origin in the primary landing site. Additionally, we define $\xi \coloneqq (p, v)$ as the concatenation of position and velocity. At each point of the time grid, t_k, with $k \in \mathbb{Z}_{[0,N-1]}$, we consider n_p polytopes in the half-space representation: $A_{k,i}\xi_k + b_{k,i} \leq 0$, $i \in \mathbb{Z}_{[1,n_\mathrm{p}]}$, where $A_{k,i} \in \mathbb{R}^{n_i \times 6}$, $b_{k,i} \in \mathbb{R}^{n_i}$, and n_i is the number of halfspaces in the i-th polytope. We introduce one indicator variable $\delta_{k,i} \in [0, 1]$ to model the membership of the state ξ_k to the i-th polytope at the k-th time instant,

$$\delta_{k,i} > 0 \Rightarrow A_{k,i}\xi_k + b_{k,i} \leq 0, \quad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}. \tag{5}$$

Finally, to obtain trajectories that traverse for the longest possible time multiple divert-feasible regions, we specify the second term of the stage cost in (4a) as

$$\frac{t_\mathrm{f}}{N}\sum_{i=1}^{n_\mathrm{p}} w_{k,i}\delta_{k,i}, \quad k \in \mathbb{Z}_{[0,N-1]}, \tag{6}$$

where $w_{k,i} \geq 0$ are weighting coefficients. In Section 5, we describe in detail the PDG constrained with divert-feasible regions and present numerical simulations to compare the MINLP approach against the MPVC approach.

2.2.2 | Coordination of unmanned ground and aerial vehicles

We consider the coordination problem between an unmanned ground vehicle (UGV) and several unmanned aerial vehicles (UAVs) described in [27]. A set of monitoring targets must be visited by UAVs. These are carried by a UGV which serves as a mobile docking site for recharging. We aim to plan a trajectory for the UGV such that the UAVs can visit all the targets.

For each monitoring target it is possible to construct a reachable set for the UAVs. The set expresses the region of space where the UAV can leave the UGV, accomplish the monitoring task, and land again on the UGV with a prescribed minimum state of charge of the battery. We consider inner convexifications of the reachable sets. These convexifications may be more conservative but are easier to handle in optimization problems. Therefore, it is possible to plan the trajectory of the UGV from a start to an end point that encompasses each reachable set at least once. Also, the trajectory has to maximize the time spent by the UGV in the reachable sets, and further constraints can be imposed, e.g., actuation limits, obstacle avoidance.

We denote the position of the UGV by $p_k \in \mathbb{R}^2, k \in \mathbb{Z}_{[0,N-1]}$, and we consider n_p reachable sets, one for each target, described by polytopes as $A_i p_k + b_i \leq 0, i = \mathbb{Z}_{[1,n_\mathrm{p}]}$. Differently from the landing problem, we assume the reachable set to be independent from time. We model the membership of the UGV position in the reachable sets of the targets by introducing an indicator variable $\delta_{k,i} \in [0, 1]$,

$$\delta_{k,i} > 0 \Rightarrow A_i p_k + b_i \leq 0, \quad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}. \tag{7}$$

To ensure that each reachable set is visited at least once along the UGV's trajectory we enforce

$$\sum_{k=0}^{N-1} \delta_{k,i} \geq 1, \quad i \in \mathbb{Z}_{[1,n_\mathrm{p}]}. \tag{8}$$

To maximize the time spent by the UGV in the reachable sets, the second term of the stage cost in (4) is identical to (6). Through functions c and c_N we can impose additional constraints, and through functions L and E additional stage and terminal objectives, respectively.

2.2.3 | Soft docking

We consider a spacecraft docking where the goal is to compute a fuel- and time-optimal trajectory that successfully docks a chaser spacecraft to a target spacecraft. Specifically, we consider the formulation presented in [24] where we substitute the so-called silent thruster constraint with a general soft-docking constraint [28]. The latter ensures a reduction of the maximum velocity of the chaser spacecraft as the target spacecraft is approached. The other important constraint in spacecraft docking is the line-of-sight constraint, which enforces a specific range for the approach angle of the spacecraft with respect to the docking port to ensure proper sensing. To avoid over-conservative trajectories the two docking-specific constraints, i.e., soft-docking and line-of-sight, are imposed only in the proximity of the docking site.

We denote by $p_k \in \mathbb{R}^3, v_k \in \mathbb{R}^3, k \in \mathbb{Z}_{[0,N-1]}$ the 3D position and velocity of the spacecraft in a global inertial reference frame. The soft-docking constraint is formulated as

$$\|p_k - p_\mathrm{f}\|_2 \leq r \Rightarrow \|v_k\|_2 \leq \alpha \|p_k - p_\mathrm{f}\|_2, \quad k \in \mathbb{Z}_{[0,N-1]}, \tag{9}$$

where p_f is the position of the docking site, $r \in \mathbb{R}_+$ is a prescribed distance from the docking site, and $\alpha \in \mathbb{R}_+$ is a parameter for the relationship between the maximum velocity and the distance from the docking site. Similarly the line-of-sight constraint is

$$\|p_k - p_\mathrm{f}\|_2 \leq r \Rightarrow \|p_k - p_\mathrm{f}\| \cos(\theta_{\max}) \leq (p_k - p_\mathrm{f})^\top e_\mathrm{f}, \quad k \in \mathbb{Z}_{[0,N-1]}, \tag{10}$$

where $\theta_{\max} \in (0, \frac{\pi}{2})$ is the approach-cone half angle and $e_f \in \mathbb{R}^3$ is the docking site axis orientation.

For this application the cost term involving indicator variables in (4a) is omitted, since these are not performance objectives but rather safety specifications.

In the aerospace engineering literature, constraints (9), (10) are often called "state-triggered constraints" [29], as the constraints are only enforced if the current system satisfies a specific "trigger" condition which depends on the system state. However, in [29] the constraints on the right side of the logical implications hold with equality. We remark that this problem could also be modelled as a multi-phase OCP by choosing a priori when the spacecraft has to be in the proximity of the docking site. The transition can be imposed via equality constraints on the system state at a specific time instant. Hence, we obtain an OCP without logical expressions which results in a standard NLP after time discretization. But, the multi-phase OCP has reduced flexibility, and thus, in general, lower performance.

2.2.4 | Abort-safe spacecraft rendezvous

By following NASA's convention for a rendezvous of a spacecraft ("deputy") with the International Space Station ("chief"), we consider two rendezvous phases [30]. In the first phase the deputy has to be *passively* safe, meaning that in case of a complete loss of all thrusters, the deputy will not collide with the chief. In the second phase, once the deputy is close enough to the chief, often it is not possible to ensure the existence of *passively* safe trajectories. Thus, we aim at computing deputy trajectories that are *actively* safe, i.e., even in case of partial thrust failure they can avoid collision with the chief by actuating the remaining thrusters. The maximum number of thrust failures allowed is fixed when the second phase starts. To determine which regions of the space are abort-safe it is possible to construct backward reachable sets [31]. The sets are constructed in different ways for both passive and active safety, but here we assume for simplicity that they are given and represented by polytopes. As usual, the planned trajectory of the deputy, in addition to being abort-safe, should also be fuel- and time-optimal.

We model the trajectory planning for a safe rendezvous as follows. First, we introduce a logical implication to determine whether the deputy should be passively or actively safe based on the distance from the chief. For passive safety, the deputy trajectory in each time step must lie in one of the passive reachable sets. Conversely, for active safety, the deputy trajectory must lie in the active reachable set corresponding to the prescribed level of acceptable thrust failure. In addition, it might be possible to enhance robustness against failures by promoting trajectories that, when possible, lie also in other active reachable sets corresponding to higher levels of thrust failures.

Mathematically, we state the above as follows. First, we define by $\xi_k \coloneqq (p_k, v_k), k \in \mathbb{Z}_{[0,N-1]}$ the concatenation of position and velocity of the deputy, respectively, and by $\mathcal{P}_i, i \in \mathbb{Z}_{[1,n_p]}$ the reachable sets corresponding to passively safe regions. For detecting and enforcing passive safety we impose

$$\|p_k - p_c\|_2 > r \Rightarrow \xi_k \in \mathcal{P}_i, \quad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_p]}, \tag{11}$$

where $p_c \in \mathbb{R}^3$ is the position of chief, and r is the distance that divides the region requiring passive safety from the one requiring active safety. Conversely, to detect and enforce active safety we impose

$$\|p_k - p_c\|_2 \leq r \Rightarrow \xi_k \in \mathcal{A}_k, \quad k \in \mathbb{Z}_{[0,N-1]}, \tag{12}$$

where $\mathcal{A}_k$ corresponds to the active-safe set that must be satisfied. Often, $\mathcal{A}_k$ is formed by multiple regions as $\mathcal{A}_k = \bigcup_{i=1}^{n_a} \hat{\mathcal{A}}_{k,i}$, hence (12) can be further specified as

$$\|p_k - p_c\|_2 \leq r \Rightarrow \xi_k \in \hat{\mathcal{A}}_k, \delta_{k,i} \geq 0, \quad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_a]}, \tag{13}$$

where $\delta_{k,i} \in [0, 1]$ are indicator variables that denote the membership in the active-safe sets $\hat{\mathcal{A}}_{k,i}, i \in \mathbb{Z}_{[1,n_a]}$. Specifically, the indicator variables enforce the constraints

$$\delta_{k,i} > 0 \Rightarrow \xi_k \in \hat{\mathcal{A}}_{k,i}, \quad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_a]}, \tag{14}$$

and we guarantee active safety by requiring

$$\sum_{i=1}^{n_a} \delta_{i,k} \geq 1, \quad k \in \mathbb{Z}_{[0,N-1]}. \tag{15}$$

Trajectories that traverse the intersection of multiple active sets $\hat{\mathcal{A}}_{k,i}$ are considered safer as intersections may represent reachable sets for higher levels of thrust failures. In order to encourage the optimizer to find such trajectories, we express the second term of the objective in (1) as

$$\frac{t_\mathrm{f}}{N}\sum_{i=1}^{n_\mathrm{a}} w_{k,i}\delta_{k,i}, \quad k \in \mathbb{Z}_{[0,N-1]}, \tag{16}$$

where $w_{k,i} \geq 0$ are weighting coefficients. Thus, this problem combines logical constraints regulated by indicator variables and indicator constraints.

3 | OBTAINING COMPUTATIONALLY TRACTABLE LOGICAL EXPRESSIONS

In this section, we demonstrate how to translate the logical implications contained in (1) into an optimization problem suitable for numerical solvers. We start by presenting formulations of (3) where the constraints involving the logical implications (1c) are governed solely by indicator variables. The general formulation (1) is addressed at the end of this section since the presence of indicator constraints and the Heaviside step-function in the cost require the introduction of approximations.

There are numerous strategies to handle logical expressions in the literature, here we consider two. The first one relies on the generalized disjunctive programming (GDP) framework introduced by Balas [5] for mixed-integer linear programs (MILPs), and extended by Grossmann and coworkers to MINLPs [6, 32]. In GDP logical expressions can be stated via the so-called "big-M" constraints or via perspective functions based on the convex hull formulation. In this work, we employ big-M constraints since their use is straightforward and avoids the introduction of copies of variables, differently from convex hull formulations [33]. This is particularly important as we consider problems with a large number of decision variables and aim to solve them quickly. A potential drawback of big-M formulations is the assumption that the problem's constraints have an upper (or lower) bound. However, this issue is generally not significant for control problems, where variables and constraints are naturally bounded by physical limits. A second disadvantage of big-M formulations is the low quality of their relaxations, which often result in excessively optimistic lower bounds. Loose relaxations can directly increase solver time. However, OCPs typically admit tight bounds on states and controls, which also lead to tight bounds on other constraints, making big-M formulations more attractive. Furthermore, when state-of-the-art MIP solvers are used, the presolve routines can preprocess the given formulation yielding tighter relaxations [34].

Let functions G_i in (3c) admit an upper bound $M \in \mathbb{R}_+$ such that, for every admissible value z, $G_i(z) \leq M$, $i \in \mathbb{Z}_{[1,n_\delta]}$. Then, (3c) is reformulated as

$$G_i(z) \leq M(1-\delta_i), \quad i \in \mathbb{Z}_{[1,n_\delta]}. \tag{17}$$

When $\delta_i = 1$, the constraint $G_i(z) \leq 0$ is enforced. When $\delta_i = 0$, the constraint becomes $G_i(z) \leq M$, which is trivially satisfied by the definition of M. Problem (3) can be formulated as the MINLP

$$\begin{aligned}
\min_{z \in \mathbb{R}^{n_z},\, \delta \in \{0,1\}^{n_\delta}} \quad & f(z) - \sum_{i=1}^{n_\delta} w_i\delta_i && \text{(18a)}\\
\text{s.t.} \quad & g(z) = 0,\ h(z) \leq 0, && \text{(18b)}\\
& G_i(z) \leq M(1-\delta_i), \quad i \in \mathbb{Z}_{[1,n_\delta]}. && \text{(18c)}
\end{aligned}$$

General MINLPs are NP-hard problem, even undecidable if the respective problem is unbounded [35]. For the special case of *convex* MINLPs, where relaxing the integer variables results in a convex NLP, existing solvers can often compute the global optimum reasonably fast, if a feasible solution exists. However, the complexity of *convex* MINLPs remains NP-hard. In Section 4, we present an overview of methods for solving MINLPs with a special focus on a recently proposed method [36], which has demonstrated to work directly and effectively on nonconvex MINLPs when other solvers fail.

The second strategy we consider for modeling the logical expression is by introducing complementarity or vanishing constraints [7, 8]. The logical implication considered in (3c) only requires vanishing constraints [8]. Specifically, the implication is substituted by the nonconvex constraints

$$\delta_i G_i(z) \leq 0,\ \delta_i \in [0,1], \quad i \in \mathbb{Z}_{[1,n_\delta]}. \tag{19}$$

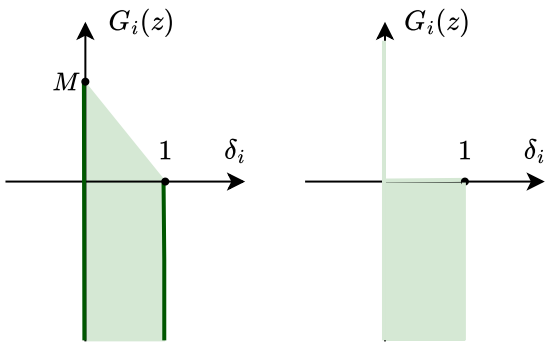

FIGURE 1 Left: in dark green the feasible set of constraint (18c) with $\delta_i \in \{0, 1\}$, and in light green its relaxation. i.e., $\delta_i \in [0, 1]$. Right: in light green the feasible set of constraint (20c).

When $\delta_i = 0$, the constraint is trivially satisfied, but when $\delta_i > 0$ it must hold that $G_i(z) \leq 0$. By means of (19), we can formulate (3) as the MPVC

$$\min_{z \in \mathbb{R}^{n_z},\, \delta \in [0,1]^{n_\delta}} \quad f(z) - \sum_{i=1}^{n_\delta} w_i \delta_i \tag{20a}$$

$$\text{s.t.} \quad g(z) = 0,\ h(z) \leq 0, \tag{20b}$$

$$\delta_i G_i(z) \leq 0, \qquad i \in \mathbb{Z}_{[1,n_\delta]}. \tag{20c}$$

Even though each constituent function is twice continuously differentiable, possibly convex, the vanishing constraint (20c) renders (20) nonconvex and nonsmooth. The nonsmoothness might be mitigated by using complementarity functions, denoted as "C-functions" or "NCP-functions" (cf. [37] for an overview). However, MPVCs are hard to solve even for powerful Newton-based NLP solvers because they violate constraint qualifications [8]. In Section 4, we outline an effective numerical method to solve MPVCs based on relaxation and homotopy.

3.1 | Comparison of the reformulations

The two constraint reformulations that we have described lead to different feasible sets for the corresponding relaxations. Figure 1 depicts the feasible set of constraints (18c) and (20c), respectively. First, consider the feasible set expressed by (18c) for the MINLP. In Figure 1, we see that the feasible set is characterized by two disjoint lines, and an ambiguity in the value of the indicator variable might arise when $G_i(z) = 0$. In this case, both the alternatives

$$G_i(z) \leq M,\ \delta_i = 0, \qquad G_i(z) \leq 0,\ \delta_i = 1, \tag{21}$$

are feasible for $G_i(z) = 0$. The ambiguity is resolved by the cost function, since an optimal solution is characterized by $\delta_i = 1$ (cf. Lemma 1).

A similar reasoning can be applied to the MPVC reformulation which has the connected feasible set shown in Figure 1. When $G_i(z) \leq 0$, the minimization of the cost causes $\delta_i = 1$, resolving the possible ambiguity. Following this intuition we can formally state this property regarding the optimal solutions of (20).

Lemma 1. *Let $(z^\star, \delta^\star)$ be a locally optimal solution of the relaxed MINLP* (18)*, i.e., with $\delta \in [0, 1]^{n_\delta}$, and let $\mathcal{I} \subseteq \mathbb{Z}_{[1,n_\delta]}$ be the set of indices such that $G_i(z^\star) \leq 0$ for all $i \in \mathcal{I}$, then $\delta_i^\star = 1$ for all $i \in \mathcal{I}$.*

Proof. By contradiction, suppose that $(z^\star, \delta^\star)$ is a locally optimal solution where for all $i \in \mathcal{I} \subseteq \mathbb{Z}_{[1,n_\delta]}, G_i(z^\star) \leq 0$, and $0 < \delta_i^\star < 1$. There exists a feasible direction that does not change $z^\star$ but acts on $\delta^\star$ improving the objective value. Therefore, $\delta^\star$ with $0 < \delta_i^\star < 1, i \in \mathcal{I}$ is not optimal, but $\delta^\star$ with $\delta_i^\star = 1$, for all $i \in \mathcal{I}$ is a locally optimal solution. □

Lemma 1 also holds for the optimal solution of the MPVC formulation (20). The proof follows the same argument. Additionally, for the MPVC we can demonstrate that a locally optimal solution does not admit fractional indicator variables.

Theorem 1. *Let* $(z^\star, \delta^\star)$ *be a locally optimal solution of* (20) *then* $\delta^\star \in \{0,1\}^{n_\delta}$.

Proof. In (20), the indicator variables $\delta \in [0,1]^{n_\delta}$ enter only constraint (20c), and only three alternatives are possible

1. $G_i(z) < 0$, then constraint (20c) is satisfied for $\delta_i \in [0,1]$. An optimal solution is characterized by $\delta_i = 1$ (cf. Lemma 1).
2. $G_i(z) = 0$ the same reasoning applies.
3. $G_i(z) > 0$, then constraint (20c) is satisfied only for $\delta_i = 0$, since $\delta_i \in [0,1]$.

Therefore, we can conclude that a locally optimal solution of (20) admits only integer indicator variables, i.e., $\delta \in \{0,1\}^{n_\delta}$. □

A different situation applies when we consider the relaxation of (18), i.e., $\hat{\delta} \in [0,1]^{n_\delta}$. As depicted in Figure 1, the big-M method creates a larger feasible set which admits cases with $G_i(z) > 0$ and $\hat{\delta}_i > 0$. Therefore, the optimal solution of the relaxation might exploit the enlarged feasible set, producing a solution with fractional indicator variables. However, it is possible to postprocess such relaxed solution, and recover a feasible solution for the original MINLP problem (18).

Lemma 2. *Given an optimal solution of the relaxed MINLP* (18)*, i.e., with* $\hat{\delta} \in [0,1]^{n_\delta}$*, the following holds*

$$\hat{\delta}_i = 1 \Leftrightarrow G_i(z) \leq 0. \tag{22}$$

Lemma 3. *Given an optimal solution of the relaxed MINLP* (18)*, i.e., with* $\hat{\delta} \in [0,1]^{n_\delta}$*, a feasible solution of the original problem* (18) *is obtained as*

$$\delta_i = \begin{cases} 0, & \text{if } \hat{\delta}_i < 1, \\ 1, & \text{otherwise.} \end{cases} \tag{23}$$

Lemma 3 gives a way to obtain feasible solutions for (20) which may be only suboptimal.

3.2 | Explicit formulations for problems with Heaviside step-function and indicator constraints

Next, we derive a MINLP and MPVC formulation that can be readily treated by a numerical solver for the general mathematical program introduced in (1). The main difficulty resides in the representation of the Heaviside step-function in the cost (1a), and of the logical implication with general indicator constraints (1c). Here, for completeness, we consider the left-hand-side of constraint (1c) to hold with strict inequality. Again, we propose a formulation with big-M constraints and one with vanishing constraints. For the former, for $i \in \mathbb{Z}_{[1,n_\delta]}$ and for every admissible value of z, we assume $-m \leq H_i(z) \leq M$ with $m, M \geq 0$, and introduce binary variables $\delta_i \in \{0,1\}$ to state the logical relation

$$H_i(z) > 0 \Rightarrow \delta_i = 1 \Rightarrow G_i(z) \leq 0. \tag{24}$$

Expression (24) can be enforced by the following inequality constraints

$$\begin{cases} H_i(z) \leq M\delta_i \\ G_i(z) \leq M(1-\delta_i). \end{cases} \tag{25}$$

Conversely, to represent (1c) with vanishing constraints it is not enough to enforce the constraint

$$H_i(z)G_i(z) \leq 0, \quad i \in \mathbb{Z}_{[1,n_\delta]}, \tag{26}$$

because the product operation does not correspond to a logical implication. To correctly represent the logical implication we must require both (26) and

$$H_i(z) \geq 0, \quad i \in \mathbb{Z}_{[1,n_\delta]}. \tag{27}$$

However, imposing (27) unless it is already present in the original formulation, reduces the feasible set and possibly harms the quality of the solution. The reduction of the feasible set is illustrated in Figure 2 where the big-M formulation (25) represents the logical implication correctly without requiring (27). Thus, adding (27) to (25) introduces an unnecessary restriction.

A correct way to represent the logical implication via structured nonconvexities without restricting the feasible set involves auxiliary variables and complementarity constraints. First, we introduce the complementarity behavior, consider the variables

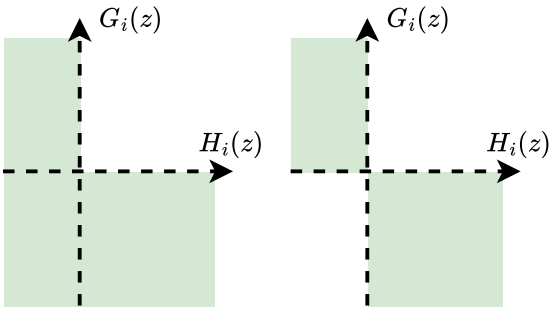

FIGURE 2 Left: in light green the feasible set of constraint (25) (the indicator variable δ_i is projected out). Right: in light green the feasible set of constraint (26). In both plots the dashed lines are part of the feasible set. The MPVC constraints (26) do not correctly represent the logical implication. By considering (27), we obtain two identical feasible sets and a correct representation of the logical implication. However, this introduces unnecessary restrictions for the MINLP formulation.

x, y, such that $0 \leq x \perp y \geq 0$. The orthogonality symbol specifies a complementarity behavior which can be expressed in different equivalent ways: using a bilinear formulation $x, y \geq 0, xy = 0$, or a min-function formulation where $\min(x, y) = 0$, or the Fischer-Burmeister function $\sqrt{x^2 + y^2} - (x + y) = 0$. The bilinear formulation introduces nonconvexity through the product term $xy = 0$ and is commonly used thanks to its simplicity and compatibility with standard optimization software, though it often leads to disjunctive programming or mixed-integer formulations. The min-function formulation is non-differentiable along the line $x = y$, requiring specialized nonsmooth optimization techniques. The Fischer-Burmeister formulation is differentiable everywhere except at (0,0), making it suitable for Newton-type solution methods that rely on gradient information. Each of these three representations leads to different yet degenerate NLPs, since they all violate standard constraint qualification. For a detailed discussion on the properties and computational handling of these formulations, we refer the reader to [37]. Let $y_i \in \mathbb{R}, i \in \mathbb{Z}_{[1,n_\delta]}$ be auxiliary variables , the feasible set represented by (25) can also be obtained as

$$\begin{cases} 0 \leq y_i \perp y_i - H_i(z) \geq 0, \\ y_i G_i(z) \leq 0. \end{cases} \tag{28}$$

Thus, the resulting problem contains both complementarity and vanishing constraints, and it can be classified as a mathematical program with complementarity constraints (MPCC), since every vanishing constraint can be reformulated as a complementarity one. Now, we have obtained two ways to reformulate the logical implication (1c). The representation of (1c) as (28) has been proposed in [29], where the constraints expressed by function $G_i(z)$ are holding with equality.

Next, we consider the representation of the Heaviside step-function in the cost (1a). For the big-M formulation, one can substitute σ simply by the auxiliary indicator variables δ_i. However, when we consider (1c) to hold with strict inequality, i.e., "$H_i(z) > 0$", optimal solutions would have an issue for $H_i(z) = 0$, as both the following cases obtained from (25) are feasible

$$\delta_i = 0 : \begin{cases} H_i(z) \leq 0, \\ G_i(z) \leq M, \end{cases} \qquad \delta_i = 1 : \begin{cases} H_i(z) \leq M, \\ G_i(z) \leq 0. \end{cases} \tag{29}$$

Based on the weighting coefficient w_i in the cost (1a) one of two cases is optimal. Indeed, the optimizer would seek a solution with $\delta_i = 1$ which imposes $G_i(z) \leq 0$, even if it is not required since $H_i(z) = 0$. Moreover, $G_i(z) \leq 0$ is harder to satisfy than $G_i(z) \leq M$. Therefore, if the optimizer sets $\delta_i = 1$, the improvement in the objective value overcomes the possible benefit of a larger feasible set. If we want to avoid the ambiguity for $H_i(z) = 0$, we can strengthen the relation between H_i and δ_i

$$H_i(z) > 0 \Leftrightarrow \delta_i = 1 \Rightarrow G_i(z) \leq 0, \tag{30}$$

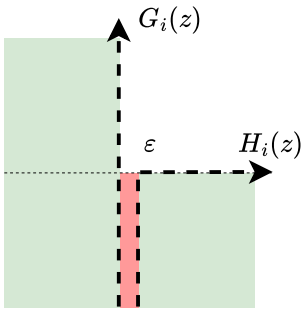

FIGURE 3 Feasible set of constraints (31). The dashed lines are part of the feasible set, the red rectangle defined from zero to ε is the portion of the feasible set removed.

which is translated via the big-M formulation into the inequalities

$$\begin{cases} H_i(z) \leq M\delta_i \\ H_i(z) \geq -m(1-\delta_i) + \varepsilon \\ G_i(z) \leq M(1-\delta_i), \end{cases} \tag{31}$$

where we modified the second inequality by adding a small positive scalar $\varepsilon \in \mathbb{R}_+$, in order to obtain

$$\delta_i = 0 : H_i(z) \geq -m + \varepsilon, \quad \delta_i = 1 : H_i(z) \geq \varepsilon. \tag{32}$$

It is evident that with this modification the solution $\delta_i = 1$ is not feasible when $H_i(z) = 0$. Therefore, the cost term (2) can be replaced by δ_i. Unfortunately, this modification has also reduced the feasible set of constraint (1c) as illustrated in Figure 3. However, such change may be arbitrarily small based on ε.

We now turn our attention to the formulation with vanishing constraints. As stated above, the logical implication can be represented by imposing (26), (27). The unresolved issue is how to obtain the function σ in the cost, with a representation more amenable for computations. An option is to use a surrogate continuous function as

$$\tilde{\sigma}^{\text{SIG}}(H_i(z)) \coloneqq \frac{1}{1 + e^{-\beta H_i(z)}}, \quad i \in \mathbb{Z}_{[1,n_\delta]}, \tag{33}$$

where the coefficient $\beta \in \mathbb{R}_+$ can be tuned to obtain a steeper transition between 0 and 1. Alternatively, it is possible to represent (2) by the Karush-Kuhn-Tucker (KKT) conditions of the linear program (LP)

$$\min_{\delta_i \in \mathbb{R}} \delta_i H_i(z) \;\text{ s.t. } 0 \leq \delta_i \leq 1 \Leftrightarrow \begin{cases} 0 \leq \delta_i \perp \lambda_{1,i} \geq 0, \\ 0 \leq 1 - \delta_i \perp \lambda_{2,i} \geq 0, \\ H_i(z) - \lambda_{1,i} + \lambda_{2,i} = 0, \end{cases} \quad i \in \mathbb{Z}_{[1,n_\delta]}, \tag{34}$$

where λ_1, λ_2 are Lagrangian multipliers associated with the constraints $0 \leq \delta_i$ and $\delta_i \leq 1$, respectively. In this case,

$$\tilde{\sigma}^{\text{KKT}}(H_i(z)) \coloneqq \begin{cases} \{0\}, & \text{if } H_i(z) < 0, \\ [0,1], & \text{if } H_i(z) = 0, \\ \{1\}, & \text{if } H_i(z) > 0. \end{cases} \tag{35}$$

The two different formulations of the Heaviside step-function are shown in Figure 4. Both representations are approximations of the ideal behaviour (2), since they cannot represent exactly the discountinuity at $H_i(z) = 0$. While (35) seems closer to the desired behavior, this may come at the price of more difficult computations.

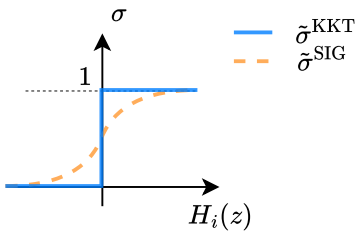

FIGURE 4 Reformulations of the Heaviside step-function: in dashed orange the representation via the sigmoid function (33), in solid blue the representation via KKT conditions (34).

Finally, the overall MINLP formulation is

$$\begin{aligned}
\min_{\substack{z\in\mathbb{R}^{n_z},\\ \delta\in\{0,1\}^{n_\delta}}} \quad & f(z) - \sum_{i=1}^{n_\delta} w_i\delta_i && && \text{(36a)}\\
\text{s.t.} \quad & g(z) = 0,\ h(z) \le 0, && && \text{(36b)}\\
& H_i(z) \le M\delta_i, && i \in \mathbb{Z}_{[1,n_\delta]}, && \text{(36c)}\\
& H_i(z) \ge -m(1-\delta_i) + \varepsilon, && i \in \mathbb{Z}_{[1,n_\delta]}, && \text{(36d)}\\
& G_i(z) \le M(1-\delta_i), && i \in \mathbb{Z}_{[1,n_\delta]}. && \text{(36e)}
\end{aligned}$$

Regarding the MPVC formulations, the feasible set of the vanishing constraints correctly represented by (28) leads to formulate MPCCs. Specifically, when the Heaviside step-function in the cost is represented by (34), the formulation is

$$\begin{aligned}
\min_{\substack{z\in\mathbb{R}^{n_z},\\ y_i\in[0,1]^{n_\delta},\\ \delta,\lambda_1,\lambda_2\in\mathbb{R}^{n_\delta}}} \quad & f(z) - \sum_{i=1}^{n_\delta} w_i\delta_i && && \text{(37a)}\\
\text{s.t.} \quad & g(z) = 0,\ h(z) \le 0, && && \text{(37b)}\\
& y_i G_i(z) \le 0, && i \in \mathbb{Z}_{[1,n_\delta]}, && \text{(37c)}\\
& 0 \le y_i \perp y_i - H_i(z) \ge 0, && i \in \mathbb{Z}_{[1,n_\delta]}, && \text{(37d)}\\
& 0 \le \delta_i \perp \lambda_{1,i} \ge 0, && i \in \mathbb{Z}_{[1,n_\delta]}, && \text{(37e)}\\
& 0 \le 1-\delta_i \perp \lambda_{2,i} \ge 0, && i \in \mathbb{Z}_{[1,n_\delta]}, && \text{(37f)}\\
& H_i(z) - \lambda_{1,i} + \lambda_{2,i} = 0, && i \in \mathbb{Z}_{[1,n_\delta]}. && \text{(37g)}
\end{aligned}$$

Although (37) does not involve integer variables, it includes complementarity constraints (37d)-(37f). Complementarity constraints are akin to vanishing constraints since they violate standard constraint qualifications making the resulting NLP hard to solve and requiring special numerical solvers. The MPCC reformulation with a surrogate of the Heaviside step-function, e.g., (33), is

$$\begin{aligned}
\min_{\substack{z\in\mathbb{R}^{n_z},\\ y_i\in[0,1]^{n_\delta}}} \quad & f(z) - \sum_{i=1}^{n_\delta} w_i\tilde{\sigma}^{\text{SIG}}(H_i(z)) && && \text{(38a)}\\
\text{s.t.} \quad & g(z) = 0,\ h(z) \le 0, && && \text{(38b)}\\
& y_i G_i(z) \le 0, && i \in \mathbb{Z}_{[1,n_\delta]}, && \text{(38c)}\\
& 0 \le y_i \perp y_i - H_i(z) \ge 0, && i \in \mathbb{Z}_{[1,n_\delta]}. && \text{(38d)}
\end{aligned}$$

4 | METHODS FOR SOLVING MINLP AND MPVC

In this section, we review some of the methods for solving MINLPs and MPVCs. The methods presented have been selected because they are general methods for their respective problem classes, they rely on existing software packages and adopt stable solvers in their subroutines, their implementations is open-source, any dependence on closed-source solvers is optional, and they have a user-friendly interface suitable for OCPs. At the end of the section, we compare the MINLP and the MPVC formulations on a tutorial example based on the UGV/UAVs coordination problem.

4.1 | Solving MINLPs

For the solution of MINLPs we briefly present two solution methods: nonlinear branch-and-bound (NBB)[38, 39] and sequential Benders-based mixed-integer quadratic programming (S-B-MIQP) [36]. NBB is a standard and well-studied solution approach that has multiple open-source and commercial implementations. Instead, S-B-MIQP is a recently proposed algorithm that has proven to be suitable for nonconvex MINLP arising from the time discretization of mixed-integer OCP.

4.1.1 | Nonlinear branch-and-bound

Nonlinear branch-and-bound (NBB) is the direct extension of the branch-and-bound method for solving MILP, introduced by [40], first presented for MINLPs in [38], and further studied in [39] for convex MINLPs. For convex MINLPs, NBB returns the global optimum, if one exists, while for nonconvex MINLPs it only finds feasible solutions. NBB is often taken as baseline method for solving MINLPs since it is a general method and there exists a well-interfaced open-source implementation in the Bonmin software package [41].

NBB solves the MINLP by relaxing the integer variables and solving the continuous (convex) NLP relaxations. The search is typically represented by a tree in which each node is a continuous NLP to solve. If a feasible solution of the NLP relaxation has all integer variables taking integer values, then it is also feasible (potentially globally optimal) for the MINLP. Each continuous relaxation with some real valued integer variables is branched into two new NLP subproblems, where a new fractional integer variable is fixed to its lower and upper bound, respectively. The solution of the node problem provide a valid lower bound that can be exploited for branching, and in case the solution is integer feasible it also provides a valid upper bound, which is useful for pruning. In fact, if the solution of a node has an objective higher than the current upper bound, the node can be pruned from the tree. Many ingredients are necessary for obtaining an efficient NBB algorithm such as tight continuous relaxation, cuts to strengthen the relaxations, efficient integration of the NLP solver, and branching strategies. For more details on NBB see, e.g., [42, 43]

4.1.2 | Sequential Benders-based MIQP

This method tackles the MINLP solution from the point of view of decomposition methods, where the aim is to solve separately the continuous and the integer part. The method presented in [36], which builds on ideas in [44, 45], is developed to directly address MINLPs arising from the time discretization of OCPs, and it has shown to be competitive with state of the art solvers on existing benchmarks. Moreover, an open-source implementation is available in the software package `CAMINO` [46]. S-B-MIQP is based on a three-step procedure. First, the problem is linearized at the current best solution ("linearization point"), which corresponds the feasible solution of the original MINLP with the lowest objective among all points visited by the algorithm. Second, a mixed-integer quadratic program (MIQP) with positive semidefinite Hessian approximation is constructed at the linearization point and solved. Third, the integer solution obtained from the MIQP is fixed into the original MINLP resulting in a continuous NLP, which is then solved. If the NLP is feasible, its solution together with the fixed integer variables is a feasible solution of the original MINLP. If the NLP is not feasible, the algorithm switches to a feasibility NLP similarly to the outer approximation scheme proposed in [47]. The solutions are used to construct new cutting planes that restrict the integer search space of the MIQP to be solved in the next algorithm iteration. This cutting planes are similar to the ones derived in the generalized Benders decomposition (GBD) [48], upgraded with a regularization method proposed in [49]. Solving exclusively MIQPs in a decomposition scheme does not guarantee termination with a global optimum in case of convex MINLP, as already shown in [47], because the solution of a MIQP does not provide a valid lower bound for the solution of the original MINLP.

Therefore, S-B-MIQP solves a tailored MILP whenever the solution of the MIQP stagnates during the S-B-MIQP iterations. The MILP constructed in S-B-MIQP is similar to the one adopted in GBD with additional linear constraints resulting from the linearization of the constraints in the original MINLP about the current best point.

4.2 | Solving MPVCs

MPVCs can be reformulated equivalently as MPCCs, and therefore, share similar solution techniques. Crucially, a solver method for MPVCs needs a custom way to deal with the structured nonconvexity. As already shown in [16] and [17, §9.3], adopting a generic NLP solver for MPVC would often result in convergence issues due to the lack of constraint qualifications. Two distinct methods exist to treat nonconvexity, one is based on an active set method and one on a relaxation method. To the authors' knowledge an active set-based solver for MPVC is implemented and tested only in [17], but no public implementation is available. Conversely, relaxation methods rely on generic NLP solvers and on a homotopy loop which can be quickly implemented. In [37], it is shown that even simple relaxation methods are effective for solving a large set of OCP with complementarity constraints. The relaxation approach adopted here has been introduced by Scholtes [50] for solving MPCCs, and later presented for MPVCs by [51, §10]. An algorithm similar to the one utilized in this work has been adopted in [20, 52]. We introduce a vector $\boldsymbol{\tau} \in \mathbb{R}^{n_{G_i}}$ such that $\boldsymbol{\tau} \coloneqq (\tau, \dots, \tau)$ that relaxes the vanishing constraint into

$$G_i(z)H_i(z) \leq \boldsymbol{\tau}, \quad i \in \mathbb{Z}_{[1,n_\delta]}. \tag{39}$$

Then, we solve (20) with the relaxed constraint (39) within a homotopy loop as presented in Algorithm 1. In the numerical simulations the parameters of Algorithm 1 are set as follows: $\tau_0 = 10^2$, $\varepsilon_0 = 0.6$, $\tau_{\min} = 10^{-3}$, $\kappa_0 = 1.6$, $\kappa_1 = 1.2$.

Algorithm 1 Homotopy method for the solution of MPVCs

Require: Initial guess $z^\star$, $\tau = \tau_0 > 1$, $\varepsilon = \varepsilon_0 < 1$, $\tau_{\min} < 1$, $\kappa_0 > 1$, $\kappa_1 > 1$
while $\tau^\star > \tau_{\min}$ **do**
  Set $\tau = \varepsilon \cdot \tau^\star$
  Solve problem with corresponding τ starting from last solution $z^\star$
  if problem is locally infeasible **then:**
    $\varepsilon = \kappa_0 \cdot \varepsilon$
  else
    Store solution as $z^\star$, $\tau^\star = \tau$, $\varepsilon = \varepsilon/\kappa_1$
  end if
end while
return z^*

4.3 | Tutorial example: UGV/UAVs coordination problem

In the following, we present an example illustrating how to formulate and solve the MINLP and the MPVC versions of the UGV/UAVs coordination problem introduced in Section 2. For modeling the UGV, we adopt a standard single-track kinematic model where the state is $x(t) = (p_x(t), p_y(t), \theta(t), v(t), \phi(t))$, and the control $u(t) = (a(t), \psi(t))$. Position of the center of the rear axle along the axes x, y is denoted with p_x, p_y, respectively, θ is the heading angle, v is the velocity, ϕ is the steering angle, a is

the acceleration, and ψ is the steering angular rate. The dynamics are described by the ODE

$$\dot{x} = \begin{pmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\theta} \\ \dot{v} \\ \dot{\phi} \end{pmatrix} = \begin{cases} v\cos(\theta), \\ v\sin(\theta), \\ \dfrac{v\tan(\phi)}{L}, \\ a, \\ \psi. \end{cases} \tag{40}$$

State and control vectors are subject to constraints,

$$\mathcal{X} = \{x \in \mathbb{R}^5 \mid |\theta| \leq \theta_{\max}, v \in [v_{\min}, v_{\max}], |\phi| \leq \phi_{\max}\}, \tag{41}$$

$$\mathcal{U} = \{u \in \mathbb{R}^2 \mid |a| \leq a_{\max}, |\psi| \leq \psi_{\max}\}. \tag{42}$$

With a slight abuse of notation we denote with p the concatenation of the position along the two axes, $p = (p_x, p_y)$. For simplicity here, the reachable sets of the monitoring targets are expressed as rectangles in the position space as

$$A_i p + b_i \leq 0, \quad A_i \in \mathbb{R}^{4\times 4}, b_i \in \mathbb{R}^4, \quad i \in \mathbb{Z}_{[1,n_\mathrm{p}]}, \tag{43}$$

and they are depicted in Figure 5 with pink rectangles, and $n_\mathrm{p} = 5$.

As described earlier we aim at planning a time-optimal trajectory for the UGV that goes from the start to the end point while visiting at least once each monitoring targets and fulfilling constraints. To avoid a tuning effort beyond the scope of this illustrative example, we simply fix the final time $t_\mathrm{f} = 76$ seconds. We formulate an OCP to model such trajectory planning problem, and we solve it by using direct multiple shooting [18]. Here, we specialize the OCP template (4) as

$$\min_{\mathbf{x},\mathbf{u},\boldsymbol{\delta}} \quad w\delta_N + \sum_{k=0}^{N-1}\left(\|u_k\|^2 + w\sum_{i=1}^{n_\mathrm{p}}\delta_{k,i}\right) \tag{44a}$$

$$\text{s.t.} \quad x_0 = \bar{x}_0, \tag{44b}$$

$$x_{k+1} = F(x_k, u_k, h), \qquad k \in \mathbb{Z}_{[0,N-1]}, \tag{44c}$$

$$u_k \in \mathcal{U}, \qquad k \in \mathbb{Z}_{[0,N-1]}, \tag{44d}$$

$$x_k \in \mathcal{X}, \qquad k \in \mathbb{Z}_{[0,N]}, \tag{44e}$$

$$\delta_{k,i} = 1 \Rightarrow A_i p_k + b_i \leq 0, \quad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}, \tag{44f}$$

$$\sum_{k=0}^{N}\delta_{k,i} \geq 1, \qquad i \in \mathbb{Z}_{[1,n_\mathrm{p}]}, \tag{44g}$$

$$\delta_{k,i} \in [0,1], \qquad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}, \tag{44h}$$

where $\boldsymbol{\delta} \coloneqq (\delta_{0,1}, \ldots, \delta_{0,n_\mathrm{p}}, \ldots, \delta_{N,1}, \ldots, \delta_{N,n_\mathrm{p}})$. Function F corresponds to the discretization of (40) with a 4-step explicit Runge-Kutta integrator with sampling time $t_\mathrm{d} = t_\mathrm{f}/N$. Also, constraints (41) are discretized and imposed only at the grid nodes. Constraints (7), (8) introduced earlier are included in (44) as (44f), (44g).

For the MINLP formulation, we modify (44h) such that the indicator variables are binaries, thus $\delta_{k,i} \in \{0,1\}, k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}$. Moreover, (44f) is represented via big-M constraints as in (18) resulting in

$$A_i p_k + b_i \leq M(1-\delta_{k,i}), \quad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}. \tag{45}$$

For the MPVC formulation, (44f) is represented as in (20) resulting in

$$\delta_{k,i}(A_i p_k + b_i) \leq 0, \quad k \in \mathbb{Z}_{[0,N-1]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}. \tag{46}$$

The parameters used in the simulation are contained in Table 1. The two approaches share the same initial guess, which is obtained in the following way. Consider $t = kt_\mathrm{d}, k \in \mathbb{Z}_{[0,N]}$, then the initial guess for the position $p_x(kt_\mathrm{d}), p_y(kt_\mathrm{d}), k \in \mathbb{Z}_{[0,N]}$ is obtained as a linear interpolation between the starting point (0, 0) and the end point (10, 10). The initial guess for the heading angle $\theta(kt_\mathrm{d})$ and the steering angle $\phi(kt_\mathrm{d})$ are set to zero for all $k \in \mathbb{Z}_{[0,N]}$. The velocity of the UGV is initialized as $v(kt_\mathrm{d}) = 0.2$

Parameter	Value	Unit
L	0.1	m
$a_{\max}$	0.05	m/s^2
$\psi_{\max}$	0.5	deg/s
$\theta_{\max}$	175	deg
$(v_{\min}, v_{\max})$	(0.1, 0.8)	m/s
$\phi_{\max}$	5	deg
t_{d}	3.8	s
N	20	-
$\bar{x}_0$	(0, 0, 0, 0.15, 0)	-
$\bar{x}_N$	(10, 10, 0, 0.15, 0)	-
w	-38	-
M	12	-
$\tau_{\min}$	10^{-4}	-

TABLE 1 Parameters of (44) used in the simulations.

Performance measure	Formulation	
	MINLP	MPVC
$\sum_{k=0}^{N-1} \|\|u\|\|_2^2$	0.028	0.025
$\sum_{k=0}^{N} \sum_{i=1}^{n_{\mathrm{p}}} \delta_{k,i}$	16	13
Objective	-607.97	-493.98
Runtime [s]	31.00	1.78

TABLE 2 Objective breakdown and runtime of MINLP and MPVC for (44).

for all $k \in \mathbb{Z}_{[0,N]}$. The variables corresponding to control inputs and indicator variables are initialized at zero in each time step. We solve the MINLP using the S-B-MIQP algorithm [36] implemented in `CAMINO` [46], with `Gurobi` v10.0.3 [53] as MIQP solver, and `IPOPT` v3.14.11 [54] as NLP solver. The MPVC is solved with the homotopy method described in Algorithm 1 using `IPOPT` as NLP solver. The example is coded in Python, the homotopy loop is also coded in Python, and `Gurobi` is used with Presolve disabled. We use `CasADi` [55] to model both the MINLP and the MPVC problem, and to interface the required solvers. The algorithms within `CAMINO` adopt the same interface of `CasADi`, therefore using S-B-MIQP does not require additional interfacing work compared to Alg. 1. The code executes as a single thread on a desktop machine running Ubuntu 22.04 with an Intel(R) Core(TM) i9-13900K CPU and 128GB of memory. Figure 5 compares the position trajectories obtained with the two formulations, and Figure 6 compares the values of the indicator variables along the two trajectories for each monitoring targets. Table 2 compares the constituent components of the cost for both formulations. While the two formulations share the same parameters, they converge to two different solutions. Specifically, the MINLP solution spends more time within the targets at the cost of a slightly higher control actuation compared to the MPVC solution. The overall objective achieved by the MINLP solution is lower compared to the objective of the MPVC solution. It is possible to achieve similar objective value with the MPVC approach by tuning the weight w. Table 2 reports also the computation time for the two formulations. For this relative simple problem the runtime for solving the MINLP is much higher than that of the MPVC. However, allowing multithreading computation for the MIQPs can reduce runtime dramatically. On our machine, allowing `Gurobi` to use up to 32 threads reduces the runtime about 6.5 times, achieving a runtime of 4.71 seconds. We highlight that both algorithms – S-B-MIQP for MINLP and homotopy for MPVC – are general purpose methods for their respective problem class. Thus, they work out of the box without extensive tuning.

Remark 1. We want to emphasize some aspects behind our modeling choices. First, in the considered problems while the dynamical system has continuous state and control, the integer part is related only to the indicator variables. Therefore, there is no need of specific reformulations that are typically adopted in mixed-integer optimal control such as the partial outer convexification [56].

Although the big-M formulation adopted for the logical constraints generally produces weaker relaxations compared to a convex hull formulation, it has the advantage of not introducing auxiliary variables and constraints. This is a relevant aspect since OCP over long horizons already have large dimensions. For a similar reason, we do not introduce an auxiliary variable for simplifying the vanishing constraint (46).

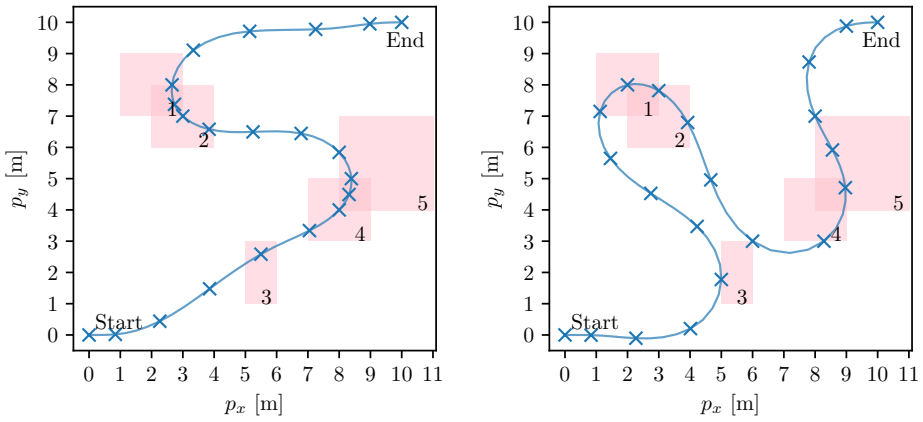

FIGURE 5 Locally optimal UGV position trajectories for MINLP (left) and MPVC (right) for (44).

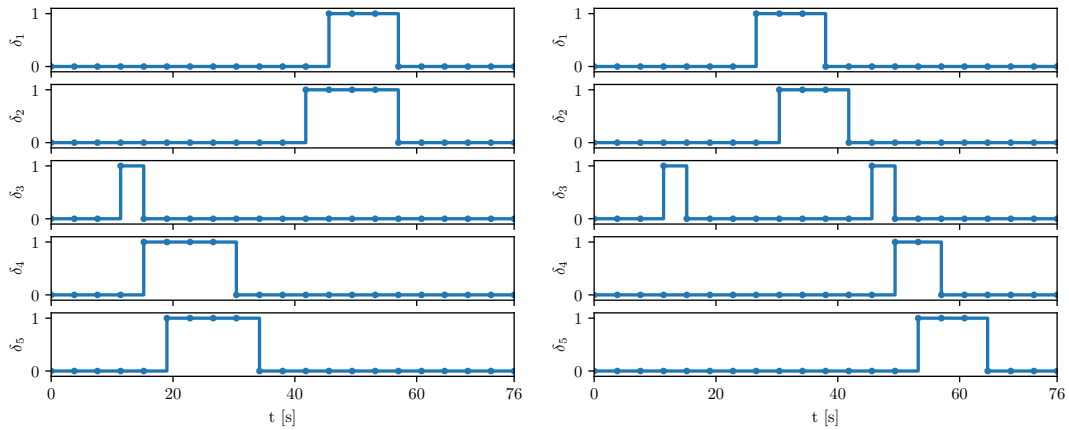

FIGURE 6 Locally optimal trajectories of indicator variables for MINLP (left) and MPVC (right) for (44).

Finally, we consider polytopes represented via halfspaces instead of vertices to avoid equality constraints, which are generally difficult to treat in MINLPs, and to avoid introducing additional variables for representing the coefficients of the vertices' convex combination. As a drawback, with the halfspace representation we have to impose several linear inequalities for each polytope. These inequalities can be tackled easily by a MIP solver, especially if it has an effective presolve routine to eliminate redundant constraints and tighten variable bounds [34]. For a NLP solver, a large amount of linear inequalities does not usually introduce challenges for convergence but rather for memory allocation and time spent executing linear algebra routines, for instance for matrix factorizations.

5 | CASE STUDY: DIVERT-FEASIBLE POWERED DESCENT GUIDANCE

We now analyze the problem of PDG with divert-feasible regions for Mars landing, and present formulations, solution methods and simulations for both the MINLP and MPVC approach. Additionally, by means of a detailed comparison, we demonstrate computation time and objective value efficiency for both methods, when considering a problem instance close to a real application.

Parameter	Value	Unit
γ_{gs}	86	deg
γ_{p}	40	deg
ω	$10^{-3} \cdot [3.5, 0, 2]^\top$	1/s
g_{mars}	-3.71	m/s^2
g_{earth}	9.807	m/s^2
I_{sp}	225	s
$\hat{e}_z$	$[0, 0, 1]^\top$	-
$(\rho_{\text{lb}}, \rho_{\text{ub}})$	(4971, 13258)	N
m_{wet}	1905	kg
m_{dry}	1505	kg

TABLE 3 Model and constraint parameters in spacecraft landing simulations.

5.1 | Modeling

We consider the Mars landing problem described in [21, p. 87]. The lander dynamics corresponds to a double integrator with variable mass, and it is described by the following nonlinear ODE

$$\begin{cases} \dot{r}(t) = v(t), \\ \dot{v}(t) = g_{\text{mars}}\hat{e}_z + \frac{u(t)}{m(t)} - \omega^\times\omega^\times r(t) - 2\omega^\times v(t), \\ \dot{m}(t) = -\frac{\|u(t)\|_2}{g_{\text{earth}} I_{\text{sp}}}, \end{cases} \tag{47}$$

where $g_{\text{mars}} \in \mathbb{R}$ is the constant gravitational acceleration of Mars, $\omega \in \mathbb{R}^3$ is Mars' constant angular velocity, $^\times$ denotes the vector cross-product, $g_{\text{earth}} \in \mathbb{R}$ is the constant gravitational acceleration of Earth, and I_{sp} is the rocket's engine specific impulse. The value of model parameters are reported in Table 3. The state comprises the Cartesian position and velocity along axes xyz, denoted by $r(t)$ and $v(t)$, respectively, and the mass of the lander, denoted by $m(t)$. Additionally, we assume that the lander is moving in a constant gravitational field and is viewed in the planet's rotating frame. Drag forces are neglected as Mars' atmosphere has low density. The ODE can be written compactly as

$$\dot{x}(t) = f(x(t), u(t)), \quad x(t) = (r(t), v(t), m(t)) \in \mathbb{R}^7,\ u(t) \in \mathbb{R}^3.$$

In what follows, we drop the dependency on time for simplicity of notation. The control u is the thrust that can be produced by the lander along axes xyz, respectively. State vector and control vector are constrained to lie in the sets $\mathcal{X}$ and $\mathcal{U}$, respectively,

$$\mathcal{X} = \{x \in \mathbb{R}^7 \mid \hat{e}_z^\top r \geq \|r\|_2 \cos(\gamma_{\text{gs}})\}, \tag{48}$$

$$\mathcal{U} = \{u \in \mathbb{R}^3 \mid 0 < \rho_{\text{lb}} \leq \|u\|_2 \leq \rho_{\text{ub}}, \quad \hat{e}_z^\top u \geq \|u\|_2 \cos(\gamma_{\text{p}})\}. \tag{49}$$

A glide-slope constraint (48) is imposed to limit the approaching angle of the spacecraft with respect to the landing site. The control constraints (49) limit the total thrust and the so-called "pointing angle" of the spacecraft. The left-hand-side of the thrust norm makes constraint (49) nonconvex. For the considered 3-DoF model, the pointing angle of the spacecraft is approximated based on the ratio between the vertical thrust and the total thrust. Limiting the pointing angle is necessary to obtain trajectories that well approximate ones computed with more sophisticated dynamical models. Table 3 contains the values of the model and constraint parameters.

5.2 | OCP formulations

In this subsection, we provide a detailed formulation of the OCP. As introduced in Sec. 2, divert-feasible regions are represented as polytopes. Our goal is to compute trajectories that remain within these polytopes for as long as possible while balancing with the minimization of fuel consumption.

5.2.1 | MINLP formulation

We can formulate the following MINLP via direct multiple shooting [18] by discretizing the spacecraft dynamics and constraints, and by adding divert-feasible regions

$$\min_{t_\mathrm{f},\,\mathbf{u},\,\mathbf{x},\,\boldsymbol{\delta}} \quad -w_0 m_N - w_1 \sum_{i=1}^{n_\mathrm{p}} \delta_{i,k} \tag{50a}$$

$$\text{s.t.} \quad x_0 = (\bar{r}_0, \bar{v}_0, \bar{m}_0), \tag{50b}$$

$$x_{k+1} = F(x_k, u_k, \tfrac{t_\mathrm{f}}{N}), \qquad k \in \mathbb{Z}_{[0,N-1]}, \tag{50c}$$

$$\rho_\mathrm{lb} \le u_k \le \rho_\mathrm{ub}, \qquad k \in \mathbb{Z}_{[0,N-1]}, \tag{50d}$$

$$\rho_\mathrm{lb} \le \|u_k\|_2 \le \rho_\mathrm{ub}, \quad k \in \mathbb{Z}_{[0,N-1]}, \tag{50e}$$

$$\cos(\gamma_\mathrm{p}) \|u_k\|_2 \le \hat{e}_z^\top u_k, \qquad k \in \mathbb{Z}_{[0,N-1]}, \tag{50f}$$

$$\cos(\gamma_\mathrm{gs}) \|r_k\|_2 \le \hat{e}_z^\top r_k, \qquad k \in \mathbb{Z}_{[0,N-1]}, \tag{50g}$$

$$r_N = 0, v_N = 0, m_N \ge m_\mathrm{dry}, \tag{50h}$$

$$A_i \xi_{k,i} - b_i \le M_i(1 - \delta_{k,i}), \qquad k \in \mathbb{Z}_{[0,N]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}, \tag{50i}$$

$$\delta_{k,i} \in \{0, 1\}, \qquad k \in \mathbb{Z}_{[0,N]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}, \tag{50j}$$

where F is the discretization of the associated ODE via a 1-step explicit Runge-Kutta integrator of order 4. The bold letters $\mathbf{x} \coloneqq (x_0, \ldots, x_N)$, $\mathbf{u} = (u_0, \ldots, u_{N-1})$, $\boldsymbol{\delta} = (\delta_{0,0}, \ldots, \delta_{n_\mathrm{p},0}, \ldots, \delta_{0,N}, \ldots, \delta_{n_\mathrm{p},N})$ define the stage-wise concatenation of state, continuous control and binary indicator variables, respectively. Remember that ξ_k in constraint (50i) is defined as $\xi_k \coloneqq (r_k, v_k)$. The scalar $M_i \in \mathbb{R}_+$ is a valid upperbound for the left-hand side of the corresponding constraint.

5.2.2 | MPVC formulation

Again, via direct multiple shooting, we formulate the OCP, this time by introducing vanishing constraints and indicator variables, and obtain the NLP

$$\min_{h,\,\mathbf{u},\,\mathbf{x},\,\boldsymbol{\delta}} \quad -m_N - \frac{t_\mathrm{f}}{N} \sum_{k=0}^{N} \sum_{i=1}^{n_\mathrm{p}} \delta_{i,k} \tag{51a}$$

$$\text{s.t.} \quad \text{(50b), (50c), (50d), (50e), (50f), (50g), (50h),}$$

$$\delta_{k,i}(A_i \xi_{k,i} - b_i) \le \tau, \qquad k \in \mathbb{Z}_{[0,N]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}, \tag{51b}$$

$$\delta_{k,i} \in [0, 1], \quad k \in \mathbb{Z}_{[0,N]}, i \in \mathbb{Z}_{[1,n_\mathrm{p}]}, \tag{51c}$$

where $\tau \ge 0$ is the homotopy parameter. For $\tau = 0$ the MPVC (51) shares the same minimizers of the MINLP (50). Of course, even if (50), (51) share the same parameters and initialization, they might converge to different local optima since the problems are nonconvex, and the methods adopted to solve them perform different operations to find local optima.

5.2.3 | Additional modifications to both formulations

Regarding the discretization of the ODE, we have parameterized the controls with piecewise linear and continuous functions. This is a common choice for PDG problems since it has proven to provide enough numerical accuracy without compromising computation time [57]. Hence, in practice, we augment (47) with an integrator to represent the current thrust and create a new control representing the thrust increment. The augmented ODE is given by

$$\begin{cases} (47), \\ \dot{u}(t) = \mu(t), \end{cases}$$

such that

$$\dot{\tilde{x}}(t) = \tilde{f}(\tilde{x}(t), \mu(t)), \text{ with } \tilde{x}(t) = (r(t), v(t), m(t), u(t)) \in \mathbb{R}^{10}, \mu(t) \in \mathbb{R}^3.$$

Parameter	Value	Unit
$\bar{r}_0$	$[2000, 0, 1500]^\top$	m
$\bar{v}_0$	$1/3.6 \cdot [288, 108, -270]^\top$	m/s
$\bar{m}_0$	m_{wet}	kg
$v_{\max}$	500/3.6	m/s
α	$(g_{\text{earth}} I_{\text{sp}})^{-1}$	s/m
N	50	-
t_{f}	75	s
(w_0, w_1, w_2)	$(10^{-3}, 10^3, 10^{-3})$	-
$\tau_{\min}$	10^{-3}	-

TABLE 4 OCP parameters used in the numerical simulations.

The incremental thrust $\mu(t)$ is unbounded. However, we add a penalization term to the cost function to obtain smoother activation profiles. Therefore, (50a) is updated as follows

$$-w_0 m_N - w_1 \sum_{k=0}^{N} \sum_{i=1}^{n_{\text{p}}} \delta_{i,k} + w_2 \sum_{k=0}^{N-1} \|\mu_k\|_2^2 .$$

Besides the integer variables in (50) and the nonconvex vanishing constraints in (51), both problems have a nonlinear dynamics (50c) and a nonconvex constraint, since the total thrust is lower bounded (50e). Thus, (50) and (51) are challenging to solve. To obtain computationally tractable problems, we consider some further simplifications. First, we do not optimize for final time, and we fix it to 75 seconds. This value corresponds to the minimum time and fuel optimal trajectory that is obtained by solving the standard PDG problem in [21, p. 87]. Second, we add slack variables to soften the terminal state constraints on position and velocity to improve numerical stability, and we penalize the use of slacks by a squared euclidean norm term in the cost function. Constraint (50h) is modified as

$$r_N - s_{r,N} = 0, \; v_N - s_{v,N} = 0, \quad s_{r,N}, s_{v,N} \in \mathbb{R}^3,$$

where $s_{r,N}, s_{v,N}$ slack the final position and velocity, respectively. Also, $s_{r,N}, s_{v,N}$ are bounded such that

$$\begin{bmatrix} -5 \\ -5 \\ 0 \end{bmatrix} \leq s_{p,N} \leq \begin{bmatrix} 5 \\ 5 \\ 5 \end{bmatrix}, \|s_{v,N}\| \leq \begin{bmatrix} 0.01 \\ 0.01 \\ 0.01 \end{bmatrix} .$$

In this way, we guarantee that any feasible solution has a maximal deviation from the prescribed landing site of 5 meters along each coordinate, and a maximal final velocity of 1 cm/s along each coordinate. As a result, the cost functions (50a), (51a) are modified into

$$-w_0 m_N - w_1 \sum_{k=0}^{N} \sum_{i=1}^{n_{\text{p}}} \delta_{i,k} + w_2 \sum_{k=0}^{N-1} \|\mu_k\|_2^2 + \|s_{p,N}\|_2^2 + \|s_{v,N}\|_2^2 .$$

the cost weights adopted in these simulations are reported in Table 4. Finally, the optimization variables in MPVC and MINLP formulation are initialized as follows. The initial guess for the states is obtained via linear interporlation between the initial state $(\bar{r}_0, \bar{v}_0, \bar{m}_0)$ and the end state which is zero for every entry except for the final mass which is equal to m_{dry}. The variables corresponding to control inputs are initialized at 10^{-5}, while both indicator variables and slacks are initialized at zero.

Remark 2. If we disregard divert-feasible regions, (50) is a standard powered descent guidance (PDG) problem [22]. PDG is a fuel- and time-optimal problem for which convex reformulations exist. Specifically, the standard PDG problem can be formulated as a second order cone problem (SOCP) and solved to global optimality in the order of milliseconds.

5.3 | Simulations with simplified divert feasible regions

We consider a landing scenario with simplified divert feasible polytopic regions. Specifically, we consider three polytopes only in the position space, i.e., polyhedrons, with a inverted-pyramid shape.

	MINLP - Bonmin	MINLP - S-B-MIQP	MPVC - Alg. 1	SOCP - IPOPT
Objective	-15290.67	-14565.96	-7812.72	-
Final position (m)	(2.88, -5, 0)	(5, -5, 0)	(-5, 5, 0)	(0, 0, 0)
Final velocity (m/s)	$10^{-3} \cdot (-7, 7, 7)$	$10^{-3} \cdot (-7, 7, 7)$	$10^{-3} \cdot (7, -7, 7)$	(0, 0, 0)
Final mass (kg)	1561.52	1561.79	1560.79	1564.85
$\sum_i \sum_k \delta_{i,k}$	27	28	25	-
Runtime (s)	631.13	263.02	7.33	0.021

TABLE 5 Results for divert-feasible landing with simplified regions, the last column corresponds to the solution of the standard PDG problem.

The polyhedrons are defined as

$$C(r_k - c_i) + d \leq 0, \quad i = 1, 2, 3, k \in \mathbb{Z}_{[0,N]}, \tag{52}$$

where

$$C = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & \cos(\beta) & -\sin(\beta) \\ -\cos(\beta) & 0 & -\sin(\beta) \\ 0 & -\cos(\beta) & -\sin(\beta) \end{bmatrix}, \beta = 70°, d = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$
$$c_1 = (2000, 400, 0), c_2 = (1000, 250, 0), c_3 = (100, -100, 0).$$

One can easily transform (52) into the canonical linear inequality $Ar_k + b_i \leq 0$, $i = 1, 2, 3$, and $k \in \mathbb{Z}_{[0,N]}$ and devise constraints (50i) and (51b). Thus, the MINLP and MPVC formulations can be passed to their respective solvers. Table 4 contains the values of the parameters used in simulation. For Bonmin, we selected the following options that achieved faster runtime `variable selection: osi-simple`, `tree search strategy: top-node`, `node comparison: dynamic`. For S-B-MIQP, we used the default values set in `CAMINO`. For Alg. 1 we chose a $\tau_{\min} = 10^{-3}$ which is sufficient to obtain locally optimal solution with binary values for the indicator variables, the other parameters are unchanged.

Computation time and objective value for each solver are reported in Table 5. In the comparison, we also add the solution from the standard PDG problem, formulated as a SOCP and, here, solved with IPOPT. For the PDG solution we omitted the objective value since it has a different cost function. Also, the reported runtime could be reduced by choosing a tailored solver or by code generation. We compute the optimal trajectory for the PDG problem in order to have a baseline to compare against for the divert-feasible trajectories. Among the solutions to the landing with divert-feasible regions, the lowest objective is achieved by the MINLP formulation solved by Bonmin. The solution exploits the softened terminal constraints on position and velocity. Moreover, the trajectory stays within one or multiple regions for 27 time steps, and employs roughly 3 kg more of fuel compared to the time- and fuel-optimal trajectory of standard PDG. However, computing the solution with Bonmin took more than 10 minutes. By using the S-B-MIQP algorithm, we can solve the MINLP in less than half the time required by Bonmin. We converge to a different local optimum with a slightly higher objective. Finally, the MPVC formulation can be solved very fast compared to the MINLP one, mostly because it only requires the solution of NLPs, but, Alg. 1 converges to a worse local minimum compared to the two MINLP solutions. Indeed, the MPVC trajectory traverses fewer divert-feasible regions, spends more fuel and exploits the slacks on the final position and velocity as the S-B-MIQP solution. We also notice that the MPVC solution is more influenced by the initial guess compared to the MINLP one. It may be possible to improve the MPVC solution by a different tuning of the weights in the cost function. Fine tuning a specific formulation is out of the scope of this work, our main goal is to show that both formulations are solvable and each one has its own advantages and disadvantages. From these simulations we see that the MPVC formulation is faster to solve but requires more tuning to achieve solutions with objective values comparable to the MINLP formulation. Figure 7 shows the position trajectories of the spacecraft for the PDG problem, the MINLP formulation solved with S-B-MIQP and the MPVC formulation. Figure 8 compares the constraint satisfaction for the three formulations, and in the bottom plot of the MINLP and MPVC formulation we report the activation of the indicator variables for the three different divert-feasible regions. The thrust profile of the PDG problem is bang-bang, which is usual for time optimal problem, while for MINLP and MPVC it is smooth. Also, the glide slope constraint is not imposed at the final time step because it might interfere with the slacked terminal state constraints, leading to an infeasible problem. From Figures 7 and 8 we omitted the MINLP trajectories computed by Bonmin because they are similar to the one of S-B-MIQP.

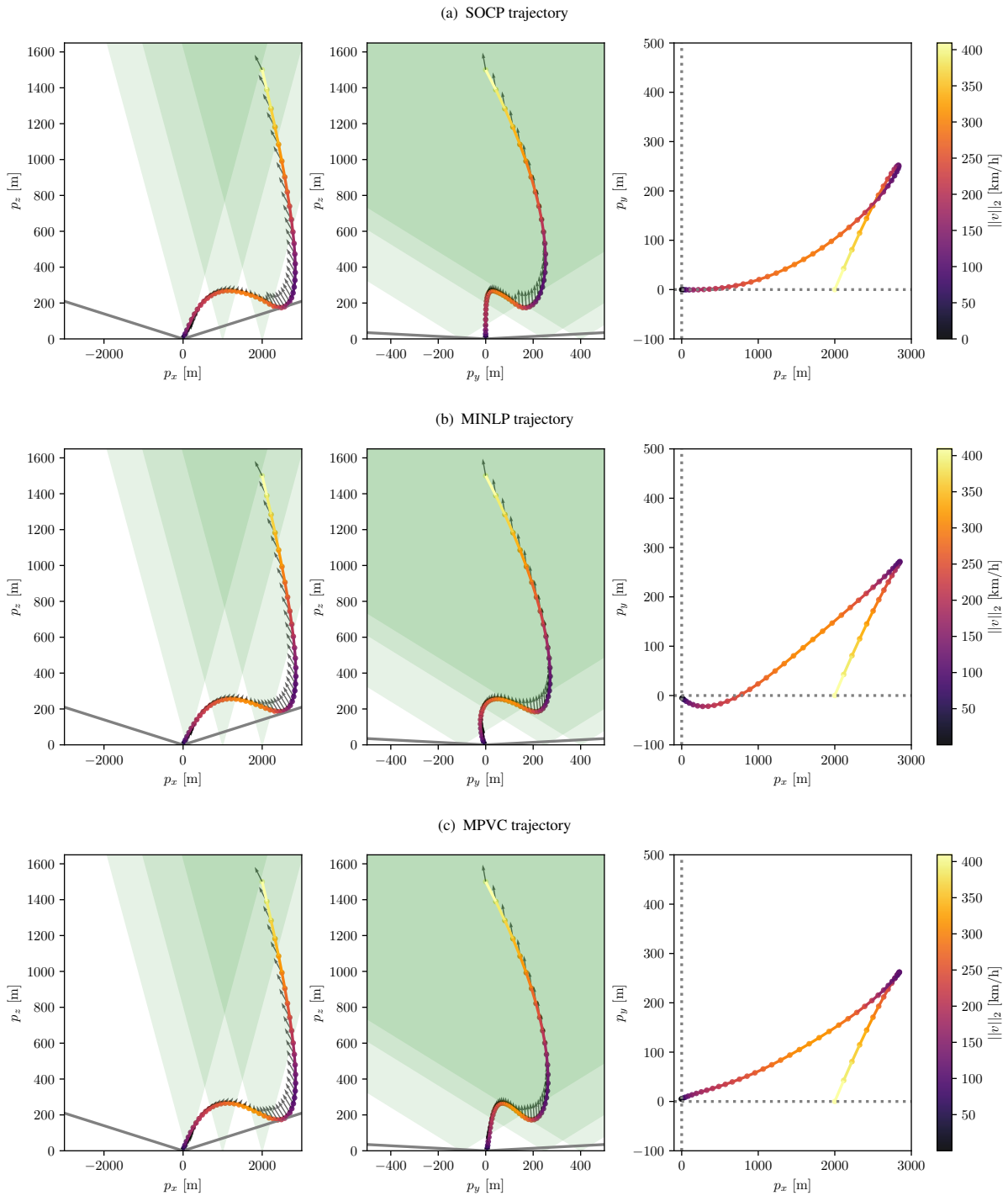

FIGURE 7 Spacecraft trajectories for SOCP (top), MINLP (50) (middle), and MPVC (51) (bottom). For each, the three plots show the trajectories of the position projected onto the *xz*, *yz*, and *xy* plane, respectively. The color used to depict the trajectory shows the Euclidean norm of the lander's velocity. The direction of the grey arrows represents the pointing angle of the lander in every discretization point, and the length of the arrows represents the Euclidean norm of the thrust. The green areas represent the divert-feasible regions. The gray solid lines at the bottom of the left and middle plot represent the glide-slope constraint. The position is constrained to be above such lines.

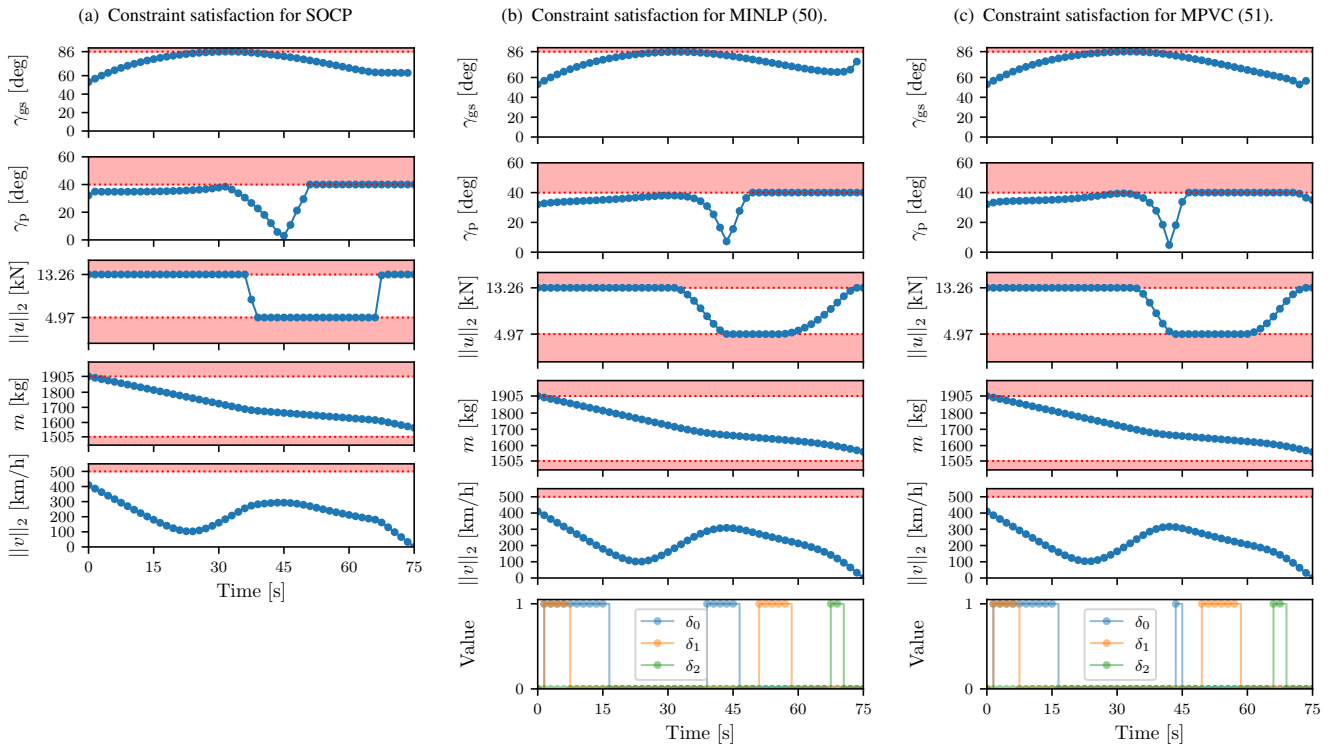

FIGURE 8 Comparison of constraint satisfaction for the time- and fuel-optimal problem (no polyhedral constraints) SOCP (left), against MINLP (50) (middle), and MPVC (51) (right). From top to bottom, the subplots represent the evolution of glide-slope angle, the evolution of the pointing angle, the Euclidean norm of the thrust, the mass depletion, the Euclidean norm of the lander's velocity. The bottom subplot in the middle and right plot represents the trajectory of the indicator variables $\boldsymbol{\delta}$, i.e., the membership in divert-feasible regions.

V-repr.	H-repr. (original)	H-repr. (pruned)
(80, 6)	(3586, 7)	(1672, 7)

TABLE 6 Dimensions of the realistic divert-feasible regions in different polytopic representations.

5.4 | Simulations with realistic regions

Now, we want to show that the two approaches can also deal with the realistic divert-feasible regions. These regions are computed based on reachable set analysis and they are convex polytopes in 6 dimensions, 3D position and 3D velocity. The polytopes are computed according to the method described in [58] which yields polytopes in vertex representation. Here, we transform the vertex representation to the halfspace representation using the `QHull` library from Python SciPy, with the options `QJ Qx C-0.00001 C0.001`. Option `QJ` is used to increase numerical stability, option `Qx` allows merging of coplanar facets and it is controlled by the following two options. `C-`, "pre-merging", allows for merging coplanar facets during the creation of the hull when the centrum of a facet is closer than 0.00001 to the centrum of a neighboring one. `C`, "post-merging", is similar but applies the merging operation after the hull is constructed. In this way, we avoid an explosion in the number of halfspaces required to describe the polytopes by slightly approximating the vertex representation. Since in this work we are interested in showing the computational aspects of the problem, we did not compute tailored divert-feasible regions for the problem at hand. Instead, we constructed our simulation with a polytope taken from [58]. We made three copies and translated them in the position space in order to obtain three scattered regions around the primary landing target located in the origin. The following three translation vectors expressed in meters $(-100, 0, -10)$, $(100, 100, -10)$, $(250, 0, -10)$ are applied in the position space of the polytope represented in Figure 9, in order to obtain three distinct polytopes. Table 6 contains the dimension of the selected polytopes, Figure 9 shows a projection of the polytope onto the 3D position space and onto the xz-position and z-velocity space.

(a) Projection onto 3D position space.

(b) Projection onto XZ-position and Z-velocity space.

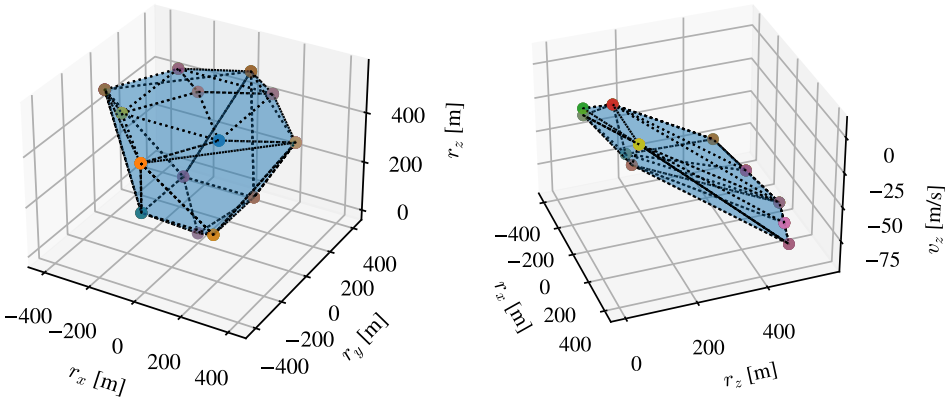

FIGURE 9 Projections of the realistic divert-feasible region.

	MINLP - S-B-MIQP	MPVC with Alg. 1
Objective	-6248.36	-4022.08
Final position (m)	(1.93, -0.24, 0)	(5, 4.88, 0)
Final velocity (m/s)	$10^{-3} \cdot (-7, 7, -7)$	$10^{-3} \cdot (-7, -7, -7)$
Final mass (kg)	1531.40	1533.28
$\sum_i \sum_k \delta_{i,k}$	8	11.12
Runtime (s)	3622.79	19845.02

TABLE 7 Results for divert-feasible landing with realistic regions.

The latter projection results in a very narrow polytope, and it helps to understand why computing trajectories that belong to these polytopes for as long as possible is not an easy task.

In the simulation with simplified regions we used 75 seconds corresponding to the optimal time found by solving the standard PDG problem. To obtain divert-feasible trajectories considering realistic divert-feasible regions, we need to extend the flight duration. Hence, we set the final time $t_f = 110$ s, which is a time budget 50% higher compared to the trajectory computed by the standard PDG. Also, we increased the available fuel mass by 50 kg, thus $m_{wet} = 1955$ kg. By extending the flight time and increasing the fuel mass, we are implicitly giving more freedom to the optimizer to explore divert-feasible trajectories. Finally, to keep a reasonable runtime we shorten the horizon to $N = 30$.

As mentioned in Remark 1, we do not want to deal with the vertex representation because it requires auxiliary variables to construct the convex combination. Also, checking the membership of a point into a polytope in vertex representation requires an equality constraint, which is generally hard to satisfy in the context of MINLP. The total number of auxiliary variables for the considered scenario, cf. Table 6, would correspond to $n_p n_c N = 7200$ where $n_p = 3$ is the number of regions, each one described by $n_c = 80$ vertices, and $N = 30$ is the length of the OCP horizon. Conversely, when using the halfspace representation, the number of variables is unchanged, but we end up having $1672 \cdot n_p N = 150480$ linear inequalities. When using a MINLP solver based on a decomposition scheme like S-B-MIQP, the internal MIP solver usually has powerful presolve routines that can dramatically reduce the number of these inequalities, since many of these are never active. For the NLP solver where presolve routines are not present, the main issue is the memory footprint and the time spent into linear algebra routines. If the problem fits in the available memory, usually the linear inequalities are not challenging in the step computation of an advanced solver like `IPOPT`.

Again, we solved both (50) and (51), the results are collected in Table 7. In this case Bonmin could not solve the problem, therefore we omitted it from Table 7. The MINLP formulation is solved by S-B-MIQP within `CAMINO` which returns a better solution compared to Alg. 1 applied to the MPVC formulation. In Alg. 1, we could not push τ low enough to obtain purely binary values for the indicator variables because of numerical problems. Therefore, we stopped the homotopy loop as soon as we achieved a $\tau < \tau_{min} = 0.1$. We argue that for this instance where we have many constraints describing each divert-feasible region, the MPVC formulation is less efficient because it creates too many nonconvexities which need to be handled by the solver. So, even if the constraints are relaxed in the homotopy loop, it is hard to find a descent direction that reduces τ in every

constraint. On the contrary, for the MINLP formulation the constraints describing the regions are linear inequalities. Specifically, for S-B-MIQP linear inequalities do not create an issue either for the master problems, which are MIPs, or for the auxiliary NLP.

6 | CONCLUSION, DISCUSSION AND OUTLOOK

In this work we presented formulations for modeling decision-making problems with performance objectives and constraints expressed through logical expressions. The approach is particularly relevant in various scenarios, as demonstrated through aerospace case studies. We further showed how the decision-making formulation can be adapted for solution via Newton-based optimization methods, introducing both MINLP and MPVC formulations. Numerical methods for solving these problems were discussed, and the formulations were analyzed and validated through a powered descent guidance (PDG) case study with divert-feasible regions for Mars landing. The numerical experience showed that the MPVC approach is generally faster for computing locally optimal solutions for problems with limited amount of nonconvexities, i.e., of logical implications. However, the performance of the MPVC formulation decreases dramatically when the number of nonconvexities is high, and it might be necessary to stop the homotopy procedure at fractional solutions, as shown in Sec. 5.4. The MINLP formulation solved with the S-B-MIQP algorithm showed to be more reliable than the homotopy method for MPVC even in case of numerous nonconvexities. Also, the MINLP formulation is more intuitive as decisions are directly represented by binary variables rather than by structured nonconvexities. We highlight that the performance of the MINLP approach might change according to the MIP solver adopted in the S-B-MIQP algorithm.

ACKNOWLEDGMENTS

Andrea Ghezzi has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement ELO-X No. 953348. Andrea Ghezzi thanks the financial support of MERL during the internship period when this work was developed. Armin Nurkanović, Moritz Diehl acknowledge fundings from DFG via Research Unit FOR 2401, project 424107692, 504452366 (SPP 2364), and 525018088, from BMWK via 03EI4057A and 03EN3054B, and from the EU via ELO-X 953348.

REFERENCES

1. Rockafellar RT. *Convex Analysis*. 28 of *Princeton Mathematics Series*. Princeton Univ. Press, 1970.
2. Hijazi H, Bonami P, Cornuéjols G, Ouorou A. Mixed-integer nonlinear programs featuring "on/off" constraints. *Computational Optimization and Applications.* 2012;52(2):537–558.
3. Bonami P, Lodi A, Tramontani A, Wiese S. On mathematical programming with indicator constraints. *Mathematical programming.* 2015;151:191–223.
4. Belotti P, Bonami P, Fischetti M, et al. On handling indicator constraints in mixed integer programming. *Computational Optimization and Applications.* 2016;65:545–566.
5. Balas E. Disjunctive programming. *Annals of discrete mathematics.* 1979;5:3–51.
6. Grossmann IE. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and engineering.* 2002;3:227–252.
7. Scholtes S. Nonconvex structures in nonlinear programming. *Operations Research.* 2004;52(3):368–383.
8. Achtziger W, Kanzow C. Mathematical programs with vanishing constraints: optimality conditions and constraint qualifications. *Mathematical Programming.* 2008;114:69–99.
9. Wolsey LA, Nemhauser GL. *Integer and combinatorial optimization*. 55. John Wiley & Sons, 1999.
10. Ceria S, Soares J. Convex programming for disjunctive convex optimization. *Mathematical Programming.* 1999;86:595–614.
11. Grossmann IE, Trespalacios F. Systematic modeling of discrete-continuous optimization models through generalized disjunctive programming. *AIChE Journal.* 2013;59(9):3276–3295.
12. Stubbs RA, Mehrotra S. A branch-and-cut method for 0-1 mixed convex programming. *Mathematical programming.* 1999;86:515–532.
13. Frangioni A, Gentile C. Perspective cuts for a class of convex 0–1 mixed integer programs. *Mathematical Programming.* 2006;106:225–236.
14. Günlük O, Linderoth J. Perspective reformulation and applications. In: , , Springer, 2011:61–89.
15. Fletcher R, Leyffer S, Ralph D, Scholtes S. Local convergence of SQP methods for mathematical programs with equilibrium constraints. *SIAM Journal on Optimization.* 2006;17(1):259–286.
16. Hoheisel T, Kanzow C. Stationary conditions for mathematical programs with vanishing constraints using weak constraint qualifications. *Journal of Mathematical Analysis and Applications.* 2008;337(1):292–310.
17. Kirches C. *Fast numerical methods for mixed-integer nonlinear model-predictive control*. Springer, 2011.
18. Bock HG, Plitt KJ. A multiple shooting algorithm for direct solution of optimal control problems. *IFAC Proceedings Volumes.* 1984;17(2):1603–1608.
19. T. H. Tsang DMH, Edgar TF. Optimal control via collocation and non-linear programming. *International Journal of Control.* 1975;21(5):763–768.
20. Jung MN, Kirches C, Sager S. On perspective functions and vanishing constraints in mixed-integer nonlinear optimal control. In: , , Springer, 2013:387–417.

21. Malyuta D, Reynolds TP, Szmuk M, et al. Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently. *IEEE Control Systems Magazine.* 2022;42(5):40–113.
22. Açıkmeşe B, Ploen SR. Convex programming approach to powered descent guidance for mars landing. *Journal of Guidance, Control, and Dynamics.* 2007;30(5):1353–1366.
23. Açıkmeşe B, Scharf D, Blackmore L, Wolf A. Enhancements on the convex programming based powered descent guidance algorithm for mars landing. *AIAA/AAS astrodynamics specialist conference and exhibit.* 2008:6426.
24. Malyuta D, Reynolds TP, Szmuk M, Açıkmeşe B, Mesbahi M. Fast trajectory optimization via successive convexification for spacecraft rendezvous with integer constraints. *AIAA Scitech 2020 Forum.* 2020:0616.
25. Sagliano M, Seelbinder D, Theil S, Lu P. Six-degree-of-freedom rocket landing optimization via augmented convex–concave decomposition. *Journal of Guidance, Control, and Dynamics.* 2024;47(1):20–35.
26. Srinivas N, Vinod AP, Di Cairano S, Weiss A. Lunar Landing with Feasible Divert using Controllable Sets. *AIAA SCITECH 2024 Forum.* 2024:0324.
27. Kim T, Vinod AP, Di Cairano S. Decoupled Trajectory Planning for Monitoring UAVs and UGV Carrier by Reachable Sets. *2024 American Control Conference (ACC).* 2024:587–593.
28. Di Cairano S, Park H, Kolmanovsky I. Model predictive control approach for guidance of spacecraft rendezvous and proximity maneuvering. *International Journal of Robust and Nonlinear Control.* 2012;22(12):1398–1427.
29. Szmuk M, Reynolds TP, Açıkmeşe B. Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints. *Journal of Guidance, Control, and Dynamics.* 2020;43(8):1399–1413.
30. Fehse W. *Automated rendezvous and docking of spacecraft*. Cambridge university press, 2003.
31. Aguilar-Marsillach D, Di Cairano S, Weiss A. Abort-safe spacecraft rendezvous on elliptic orbits. *IEEE Transactions on Control Systems Technology.* 2022.
32. Grossmann IE, Ruiz JP. Generalized disjunctive programming: A framework for formulation and alternative algorithms for MINLP optimization. *Mixed Integer Nonlinear Programming.* 2012:93–115.
33. Vielma JP. Mixed integer linear programming formulation techniques. *Siam Review.* 2015;57(1):3–57.
34. Achterberg T, Bixby RE, Gu Z, Rothberg E, Weninger D. Presolve reductions in mixed integer programming. *INFORMS Journal on Computing.* 2020;32(2):473–506.
35. Kannan R, Monma CL. On the computational complexity of integer programming problems. *Optimization and Operations Research: Proceedings of a Workshop Held at the University of Bonn, October 2–8, 1977.* 1978:161–172.
36. Ghezzi A, Van Roy W, Sager S, Diehl M. A Sequential Benders-based Mixed-Integer Quadratic Programming Algorithm. *arXiv preprint arXiv:2404.11786.* 2024.
37. Nurkanović A, Pozharskiy A, Diehl M. Solving mathematical programs with complementarity constraints arising in nonsmooth optimal control. *Vietnam Journal of Mathematics.* 2024:1–39.
38. Dakin RJ. A tree-search algorithm for mixed integer programming problems. *The computer journal.* 1965;8(3):250–255.
39. Gupta O, Ravindran A. Branch and Bound experiments in convex nonlinear integer programming. *Management Science.* 1985;31:1533–1546.
40. Land A, Doig A. An Automatic Method of Solving Discrete Programming Problems. *Econometrica.* 1960;28(3):497–520.
41. Bonami P, Biegler L, Conn A, et al. An Algorithmic Framework For Convex Mixed Integer Nonlinear Programs. tech. rep., IBM T. J. Watson Research Center; New York, USA: 2005.
42. Bonami P, Lee J, Leyffer S, Wächter A. On branching rules for convex mixed-integer nonlinear optimization. *Journal of Experimental Algorithmics (JEA).* 2013;18:2–1.
43. Floudas CA. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.
44. Bürger A, Zeile C, Altmann-Dieses A, Sager S, Diehl M. A Gauss–Newton-based decomposition algorithm for Nonlinear Mixed-Integer Optimal Control Problems. *Automatica.* 2023;152:110967.
45. Ghezzi A, Simpson L, Buerger A, Zeile C, Sager S, Diehl M. A Voronoi-Based Mixed-Integer Gauss-Newton Algorithm for MINLP Arising in Optimal Control. *Proceedings of the European Control Conference (ECC).* 2023.
46. Ghezzi A, Van Roy W. CAMINO: Collection of Algorithms for Mixed-Integer Nonlinear Optimization. https://github.com/minlp-toolbox/CAMINO; 2024.
47. Fletcher R, Leyffer S. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming.* 1994;66:327–349.
48. Geoffrion A. Generalized Benders Decomposition. *Journal of Optimization Theory and Applications.* 1972;10:237–260.
49. Kronqvist J, Bernal DE, Grossmann IE. Using regularization and second order information in outer approximation for convex MINLP. *Mathematical Programming.* 2020;180(1):285–310.
50. Scholtes S. Convergence properties of a regularization scheme for mathematical programs with complementarity constraints. *SIAM Journal on Optimization.* 2001;11(4):918–936.
51. Hoheisel T. *Mathematical programs with vanishing constraints*. PhD thesis. Universität Würzburg, Würzburg, Germany; 2009.
52. Bock HG, Kirches C, Meyer A, Potschka A. Numerical solution of optimal control problems with explicit and implicit switches. *Optimization Methods and Software.* 2018;33(3):450–474.
53. Gurobi Optimization, LLC . Gurobi Optimizer Reference Manual. https://www.gurobi.com; 2024. Last accessed: 2024-04-01.
54. Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming.* 2006;106:25–57.
55. Andersson JA, Gillis J, Horn G, Rawlings JB, Diehl M. CasADi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation.* 2019;11:1–36.
56. Sager S, Jung M, Kirches C. Combinatorial integral approximation. *Mathematical Methods of Operations Research.* 2011;73(3):363–380.
57. Scharf DP, Açıkmeşe B, Dueri D, Benito J, Casoliva J. Implementation and experimental demonstration of onboard powered-descent guidance. *Journal of Guidance, Control, and Dynamics.* 2017;40(2):213–229.
58. Lishkova Y, Vinod A, Di Cairano S, Weiss A. Divert-feasible lunar landing under navigational uncertainty. *Proceedings of the Conference on Decision and Control (CDC).* 2024:7497–7503.