2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Addressing Discrete Dynamic Optimization via a Logic-Based Discrete-Steepest Descent Algorithm

Zedong Peng, Albert Lee, David E. Bernal Neira

Abstract— **Dynamic optimization problems involving discrete decisions have several applications, yet lead to challenging optimization problems that must be addressed efficiently. Combining discrete variables with potentially nonlinear constraints stemming from dynamics within an optimization model results in mathematical programs for which off-the-shelf techniques might be insufficient. This work uses a novel approach, the Logic-based Discrete-Steepest Descent Algorithm (LD-SDA), to solve Discrete Dynamic Optimization problems. The problems are formulated using Boolean variables that enforce differential systems of constraints and encode logic constraints that the optimization problem needs to satisfy. By posing the problem as a generalized disjunctive program with dynamic equations within the disjunctions, the LD-SDA takes advantage of the problem's inherent structure to efficiently explore the combinatorial space of the Boolean variables and selectively include relevant differential equations to mitigate the computational complexity inherent in dynamic optimization scenarios. We rigorously evaluate the LD-SDA with benchmark problems from the literature that include dynamic transitioning modes and find it to outperform traditional methods, i.e., mixed-integer nonlinear and generalized disjunctive programming solvers, in terms of efficiency and capability to handle dynamic scenarios. This work presents a systematic method and provides an open-source software implementation to address these discrete dynamic optimization problems by harnessing the information within its logical-differential structure.**

I. Introduction

Control tasks with high-level discrete or logical decisions, such as integrated process design and control [1], trajectory optimization [2], and energy management in hybrid electric vehicles [3], can be modeled using Mixed-Integer Dynamic Optimization (MIDO). However, MIDO problems are computationally challenging to solve, and standard optimization solvers cannot simultaneously handle discrete variables and differential algebraic equations (DAE). The standard routine for solving MIDO problems is to discretize differential equations, resulting in a mixed-integer nonlinear programming (MINLP) problem [1], [4], [5]. However, the resulting MINLP problem is usually nonconvex, as it involves nonlinear equality constraints, and existing MINLP solvers are insufficient to solve these problems [6], particularly in online settings relevant to MIDO.

Since the discrete behaviors in mixed-integer control problems can sometimes be modeled as logic variables and constraints, an alternative to MIDO is Differential-Algebraic Generalized Disjunctive Programming (DAGDP). In DAGDP problems, logical decisions, represented as Boolean variables indicating the choices via True or False values, interact with the dynamic systems depicted by differential equations. Compared to the MIDO formulation, the disjunctions in DAGDP problems allow for a more compact representation of the logic-induced control problem, where certain operations or configurations are only feasible or relevant under specific discrete conditions. Moreover, another benefit of the DAGDP formulation is that it facilitates the use and development of alternative solution strategies in addition to the MINLP reformulation, which more effectively leverages the structure of the optimization model. The use of dynamic models within the framework of DAGDP offers a promising avenue to model control problems associated with processes, and this paper introduces an innovative approach to effectively tackling their complexities.

Zedong Peng, Albert Lee, and David E. Bernal Neira are with the Davidson School of Chemical Engineering, Purdue University, West Lafayette, IN, 47907 USA {peng372,lee4382,dbernaln}@purdue.edu

A. Related Work

The approaches to solving DAGDP problems primarily derive from and combine established ideas from both generalized disjunctive programming (GDP) and dynamic optimization. Typically, solution strategies for dynamic optimization problems include single shooting, multiple shooting, direct transcription, and numerical integration [5]. In the literature, direct transcription, e.g., finite difference or orthogonal collocation, is first applied to discretize the DAE system. Through this, the DAGDP problem will be reformulated into a GDP model, which will be solved by reformulation into MINLP (BigM or Hull) or logic-based methods [1], [7]. Although this approach is simple and intuitive, it has a significant drawback, that the reformulated MINLP problem is usually nonconvex.

In addition to MINLP reformulation, an alternative method to solve MIDO problems is to apply complete discretization techniques based on complementarity constraints, also known as the Mathematical Program with Equilibrium Constraints (MPEC) [8]. The complementarity constraint avoids the use of discrete variables. However, MPEC reformulations yield a highly nonconvex nonlinear programming (NLP) problem, usually challenging to solve.

Other DAGDP solution approaches in the literature handle the discrete variables first and then solve a series of continuous dynamic optimization subproblems. This strategy does not necessarily depend on a particular discretization technique and provides more freedom to tackle the DAE system. For example, the MIDO problems can be decomposed into a primal dynamic optimization and mixed-integer linear programming (MILP) sub-problems. These can be solved later using a MINLP decomposition algorithm, such as the

979-8-3503-1632-2/24/$31.00 ©2024 IEEE

outer-approximation and generalized bender decomposition methods [9]. Chachuat et al. [10] address MIDO problems by adapting the outer-approximation algorithm and applying the branch-and-bound method to solve dynamic optimization subproblems to global optimality. Deriving valid linear inequalities to construct the MILP subproblems is a challenging task for MIDO problems, which usually have nonconvex nonlinear constraints. Inspired by this approach, we propose decomposition methods that decouple the dynamic optimization from the discrete choices. We aim to tackle the discrete variable optimization without defining a MILP.

Another technique used to optimize discrete problems is through a discrete steepest descent algorithm (D-SDA), where the search space is mapped to a lattice representing each realization of the discrete variables and its corresponding subproblem over the remaining continuous decision, followed by a series of steepest descent steps to find improving solutions [11]. This lattice can be smaller than the original combinatorially large one defined by all combinations. The D-SDA is designed to address MINLP problems and has been used to efficiently solve dynamic optimization problems that integrate design and nonlinear model predictive control (NMPC) [12] and consider cases even under uncertainty [13]. This approach effectively avoids the pitfalls of nonconvexities and suboptimal solutions inherent in other optimization methods. In this particular case, the dynamic model is not directly enforced by the discrete choices in the problem. While the D-SDA demonstrates robustness in managing complex decision variables and nonconvex problems, as illustrated in a distillation column case study, the authors acknowledge the computational intensity of the subproblems as a challenge and point towards future enhancements to improve efficiency and applicability to broader systems.

Another recent extension of the D-SDA is to consider the case where discrete choices imply sets of (potentially nonlinear) constraints, usually expressed as GDP problems [14]. This approach was named logic-based D-SDA (LD-SDA). Using the LD-SDA shows that leveraging the logical structure in these disjunctive programs leads to an improved solution of highly nonlinear problems, such as a catalytic distillation column, compared to using the D-SDA over a reformulated MINLP. The constraints implied by the Boolean variables addressed by the LD-SDA were algebraic, and no previous work has addressed the case where a system of differential equations implies these constraints.

B. Contributions

This work's contributions can be summarized as follows:

- We extend the LD-SDA to solve DAGDP problems. The LD-SDA utilizes a logic-based search strategy to explore the search space given by the discrete choices, here Boolean variables, involved in disjunctions, and it is independent of the type of method used to solve the dynamic optimization subproblems. This flexibility allows a variety of dynamic optimization approaches to be applied, such as finite difference methods, orthogonal collocation, and even numerical integrators.
- This work integrated the model transformations in the open-source algebraic modeling language Pyomo [15], Pyomo.DAE and Pyomo.GDP, for DAGDP modeling. Several numerical instances of mode transition are provided. The implementation is available at github.com/SECQUOIA/LD-SDA-Dynamic.
- An open-source and general implementation of LD-SDA is provided as an option for the GDPOpt solver [16] for users to solve GDP problems, potentially with DAE systems in the disjunctions.

II. Background

A. Discrete-Continuous Dynamic Optimization Problems

In this work, we focus on DAGDP problems with Boolean variables, i.e., $Y \in \{False, True\} = \{\bot, \top\}$, and differential-algebraic equations (DAE) in the disjunctions. The problem formulation can be written as follows.

$$\min_{\substack{x(t),y(t),\\ u(t),p,\\ Y(t)\in\{\bot,\top\}}} \quad \psi(x(t),y(t),u(t),p,t) \tag{1a}$$

$$\text{s.t. } \xi(x(t),\dot{x}(t),y(t),u(t),p,t) = 0 \tag{1b}$$

$$\bigvee_{r\in D_k} \begin{bmatrix} Y_{kr} \\ \varphi_{kr}(x(t),\dot{x}(t),y(t),u(t),p,t) = 0 \end{bmatrix} \tag{1c}$$

$$\veebar_{r\in D_k}(Y_{kr}) = True = \top \quad \forall k \in K \tag{1d}$$

$$\Omega(Y_{kr}) = True = \top \tag{1e}$$

$$x(0) = x_0 \tag{1f}$$

$$x^L \le x(t) \le x^U, y^L \le y(t) \le y^U \tag{1g}$$

$$u^L \le u(t) \le u^U, p^L \le p \le p^U. \tag{1h}$$

where $x(t)$ represents the time-dependent state vectors, with time derivatives indicated by $\dot{x}(t)$. $y(t)$ denotes the algebraic variable states, and $u(t)$ the control actions. The vector p encompasses the decision variables of the system not dependent on time t. Eq. (1b) represents the DAE system or the purely algebraic equations that are consistently enforced. For each disjunction $k \in K$, a selection is made from a set of options defined by D_k. For every disjunct $r \in D_k$, a Boolean variable Y_{kr}, differential algebraic equations, representing system dynamics, are specified as Eq. (1c). The objective function $\psi(\cdot)$ and functions $\xi(\cdot)$ and $\varphi_{kr}(\cdot)$ that define constraints are potentially nonlinear. The Boolean variables follow an exclusive OR ($\veebar$) constraint as Eq. (1d), ensuring that within each disjunction k, exactly one option r is selected. If $Y_{k,r}(t)$ is set to $True = \top$, the corresponding disjunct and the constraints inside will be active. Otherwise, the disjunct can be discarded. Eq. (1e) is the logical propositions that represent the relationships of the logical variables. The initial state and the bounds are provided in Eqs. (1f) - (1h).

B. Reformulations and logic-based methods for GDP

There are different strategies to solve GDP problems. One approach is to reformulate the GDP problem into MINLP problems via BigM and Hull reformulations. The BigM reformulation introduces a sufficiently large scalar that makes the particular constraint redundant when its indicator variable

is not selected. The Hull reformulation lifts the model to a higher-dimensional space by introducing copies of the continuous variables and constraints inside disjunctions. The Hull reformulation always yields tighter relaxations than the BigM reformulation at the expense of larger model sizes. For more details, see [17].

In addition to MINLP reformulation, one can exploit explicit logical propositions in GDP problems via logic-based methods, such as the logic-based outer approximation (LOA) and logic-based branch and bound (LBB) [16]. Contrary to the MINLP reformulation, logic-based methods formulate specific subproblems corresponding to the values of the logical variables while solving the problems. These subproblems only include constraints activated by the logical variables within each evaluated assignment of $True/False$ to each Boolean variable, or configuration. For instance, if the specific logical configuration $\hat{Y}$ is given, the disjunctions can be fixed, and the subproblem becomes

$$\min_{x(t),y(t),u(t),p} \quad \psi(x(t),y(t),u(t),p,t) \tag{2a}$$
$$\text{s.t.} \quad \xi(x(t),\dot{x}(t),y(t),u(t),p,t) = 0 \tag{2b}$$
$$\varphi_{kr}(x(t),\dot{x}(t),y(t),u(t),p,t) = 0 \ \forall Y_{kr} = \top \tag{2c}$$
$$x(0) = x_0 \tag{2d}$$
$$x^L \le x(t) \le x^U, y^L \le y(t) \le y^U \tag{2e}$$
$$u^L \le u(t) \le u^U, p^L \le p \le p^U. \tag{2f}$$

The subproblem represents the optimization problem under constraints with a fixed logical configuration. This problem might avoid evaluating numerically challenging nonlinear equations whenever their corresponding logical variables are irrelevant. Since the subproblem satisfies the logical proposition (1e), the algorithm avoids solving subproblems from infeasible logical configurations.

Solving the subproblems (2), which result from exploration of the space defined by discrete variables, can result in convergence to the optimal solution of (1). The methods for choosing the series of subproblems lead to different logic-based methods, among them LOA and LBB. LOA uses gradient-based linearization of the nonlinear constraints at the optimal solution of Eq. (2) to approximate the feasible region of the original problem. The additional constraint would be added to a mixed-integer programming problem, denoted as the main problem. The optimal solution to the main problem returns a configuration of Boolean variables. On the other hand, LBB systematically solves GDP by exploring the values of Boolean variables in the search tree, where each node represents the partial fixation of the Boolean variables. The solutions in the node provide bounds to the optimal solution. Both methods are designed to efficiently find the optimal configuration of Boolean variables [16].

C. Discretization for Dynamic Optimization

DAGDP problems contain differential and algebraic constraints in their disjunctions. An approach to obtain a problem with only algebraic constraints that can be solved as a GDP, as described in § II-B, is to use the transcription approach [5], which transforms sets of differential equations into algebraic equations through an orthogonal collocation within finite elements. The transformed GDP becomes

$$\min_{\substack{x_{ij},y_{ij},u_{ij},p,\\ Y_{kr}\in\{\bot,\top\}}} \quad \psi(x_{ij},y_{ij},u_{ij},p) \tag{3a}$$
$$\text{s.t.} \quad F(x_{ij},y_{ij},u_{ij},p) \le 0 \tag{3b}$$
$$\bigvee_{r\in D_k} \begin{bmatrix} Y_{kr} \\ f_{kr}(x_{ij},y_{ij},u_{ij},p) = 0 \end{bmatrix} \tag{3c}$$
$$\veebar_{r\in D_k}(Y_{kr}) = True = \top \quad \forall k \in K \tag{3d}$$
$$\Omega(Y_{kr}) = True = \top \tag{3e}$$
$$x(0) = x_0 \tag{3f}$$
$$x^L \le x_{ij} \le x^U, y^L \le y_{ij} \le y^U \tag{3g}$$
$$u^L \le u_{ij} \le u^U, p^L \le p \le p^U \tag{3h}$$
$$i \in N_e,\ j \in N_c,\ k \in K,\ r \in D_k, \tag{3i}$$

where N_e is the number of finite elements and N_c is the number of internal collocation points used for properly discretizing the DAE. When comparing the DAGDP formulation with GDP, the differential equations in Eqs. (1b) and (1c) are mapped into algebraic equations in Eqs. (3b) and (3c).

When discretizing the DAGDP into GDP, the entire time horizon is separated into several finite elements. The dynamic behavior of the process is captured in each stage using a series of points given by an orthogonal collocation. The smoothness of the dynamic response determines the right number of these elements. The size of a given finite element i represents the particular length of the independent variable. The orthogonal collocation within each finite element facilitates the precise determination of the internal location points, ensuring accurate modeling of the process dynamics.

When applying both transformations, the subproblems become the following NLP problems

$$\min_{x_{ij},y_{ij},u_{ij},p} \quad \psi(x_{ij},y_{ij},u_{ij},p) \tag{4a}$$
$$\text{s.t.} \quad F(x_{ij},y_{ij},u_{ij},p) \le 0 \tag{4b}$$
$$f_{kr}(x_{ij},y_{ij},u_{ij},p) = 0 \quad \forall Y_{kr} = \top \tag{4c}$$
$$x(0) = x_0 \tag{4d}$$
$$x^L \le x_{ij} \le x^U, y^L \le y_{ij} \le y^U \tag{4e}$$
$$u^L \le u_{ij} \le u^U, p^L \le p \le p^U \tag{4f}$$
$$i \in N_e,\ j \in N_c,\ k \in K,\ r \in D_k. \tag{4g}$$

D. Convergence Analysis

The motivation for the LD-SDA is based on discrete convex analysis [18], where the convergence to global optimal points of various convex functions (M-convex and L-convex) over lattices is characterized by local optimality over integral neighborhoods. The value of the function corresponds to the optimal solution of a dynamic optimization problem given a fixed set of discrete choices. However, we cannot guarantee that the solutions to the DAGDP models over the discrete choices define a function that is either M- or L-convex. Therefore, the LD-SDA serves as a heuristic method, and the convergence of global optimality cannot be guaranteed.

III. Logic-Based Discrete-Steepest Descent Algorithm

In this section, we explain how to apply the Logic-based Discrete-Steepest Descent Algorithm (LD-SDA) to tackle DAGDP problems with ordered Boolean variables.

As mentioned in §I-A, the key of LD-SDA is to first map the ordered Boolean variables into a lattice with each point representing a particular realization of the disjunctions. Then, LD-SDA performs the neighbor search and the line search to find improving solutions by solving the corresponding continuous dynamic optimization subproblems.

A. Reformulation

In the DAGDP model, ordered Boolean variables $\{Y_{kr} \mid r \in D_k\}$ can be reformulated into a set of integer variables referred to as *external variables*. For example, in each exclusive OR constraint (1d), the Boolean variables $Y_{1k}, Y_{2k}, \ldots, Y_{|D_k|k}$ can be represented by discrete values $z_k \in \{1, 2, \ldots, |D_k|\}$ according to an ordered sequence they follow. After the reformulation, the feasible region of the boolean variables is mapped into a $|D_k|$-dimensional integer lattice.

B. Algorithm Description

In the lattice, each point corresponds to a continuous dynamic optimization subproblem by fixing the disjunctions, which can be solved by arbitrary dynamic optimization methods. The LD-SDA starts from the given initial point and obtains the initial primal bound (PB) by solving the subproblem. Then, a neighbor search over the external variable lattice is performed to find the steepest descent direction. Two types of integral neighborhoods, i.e., search directions, are supported, defined by the L_2 and L_∞ norms. If no better solution is found in the neighbor search, a locally optimal solution is reached, and the algorithm terminates. Otherwise, there exists at least one improving direction. In this case, the algorithm will move to the best neighbor, and a line search will be performed in the improving direction. If a worse solution is detected during the line search, a new neighbor search and line search will be repeated at the incumbent best-found point until a local optimum is found. After each neighbor and line search, the explored points will be added to the set G of explored points, used to avoid exploring the same point twice. The detailed steps are described in Algorithm 1. For simplicity, we use $\mathbf{z}$ to denote the external variables and $\mathbf{x}$ for all remaining variables in the DAGDP problem.

Algorithm 1: Logic-based Discrete-Steepest Descent Algorithm (LD-SDA) for DAGDP problems

Input: An external variable feasible solution $\mathbf{z_0}$; Integral neighborhood $\in \{L_2, L_\infty\}$.

Initialize: $k \leftarrow 0$, $G \leftarrow \{\mathbf{z_0}\}$
Generate the search directions $d \in D$.
Solve the initial DO subproblem with the given $\mathbf{z_0}$
$(\mathbf{x_0}, \mathrm{PB}_0) \leftarrow \mathbf{SolveDO}(\mathbf{z_0})$
while *True* **do**
  Generate neighbors
  $N_k = \{n : n = \mathbf{z_k} + d \ \forall d \in D\} \setminus G$
  Perform **Neighbor Search**
  $(\mathbf{z_{k+1}}, \mathbf{x_{k+1}}, \mathbf{d}^*, G, \mathrm{PB}_{k+1}) \leftarrow \mathbf{NS}(\mathbf{z_k}, N_k, \mathrm{PB}_k)$
  $k \leftarrow k + 1$
  if $PB_k > PB_{k-1}$ **then**
    while $PB_k > PB_{k-1}$ **do**
      Perform **Line Search**
      $(\mathbf{z_{k+1}}, \mathbf{x_{k+1}}, G, \mathrm{PB}_{k+1}) \leftarrow \mathbf{LS}(\mathbf{z_k}, \mathbf{d}^*, \mathrm{PB}_k)$
      $k \leftarrow k + 1$
  else
    $\mathbf{z}^* \leftarrow \mathbf{z_k}$, $\mathbf{x}^* \leftarrow \mathbf{x_k}$
    return Best found feasible solution $\mathbf{z}^*$ and $\mathbf{x}^*$

IV. Computational experiments

A. Implementation Details

This section shows the effectiveness of the LD-SDA through several computational experiments. The DAGDP models are written using the DAE and GDP modules in Pyomo. We provide an implementation of LD-SDA in GDPOpt in Pyomo. We further benchmark LD-SDA against solving the problem as a reformulated MINLP problem, and other logic-based methods, such as LOA, GLOA, and logic-based enumeration. KNITRO and BARON are used as (MI)NLP solvers. For KNITRO, `mip_multistart` is set to 1 to enable a mixed-integer multi-start heuristic and improve the chances of finding the global solution. All tests ran on a Linux cluster with 48 AMD EPYC 7643 2.3GHz CPUs and 1 TB RAM, restricted to using only a single thread.

B. Three-stage Dynamic Model Switching

Consider the optimization of a system with two dynamic modes and three stages [7]. At each stage s, only one of the two modes can be enforced. This problem aims to compute the dynamic model and the optimal control actions that apply in each stage by maximizing the square of the state variable over the time horizon, and is formulated as

$$\min_{\substack{x(t),u(t),\\ Y\in\{\bot,\top\}}} V(x) = -\int_{t_0}^{t_s} x^2(t) \tag{5a}$$

$$\text{s.t.} \begin{bmatrix} Y_{s,1} \\ \frac{dx}{dt} = -xe^{x-1} + u \end{bmatrix} \veebar \begin{bmatrix} Y_{s,2} \\ \frac{dx}{dt} = \frac{0.5x^3+u}{20} \end{bmatrix}, t \in [t_{s-1}, t_s] \tag{5b}$$

$$Y_{s,2} \Rightarrow \vee_{s'<s} Y_{s',1} \tag{5c}$$

$$Y_{s,2} \Rightarrow \neg \vee_{s'>s} Y_{s',1} \tag{5d}$$

$$t_0 = 0,\ t_s = s \qquad \forall s = \{1,2,3\} \tag{5e}$$

$$x(0) = 1,\ u(0) = 4, \tag{5f}$$

where x is the state variable and u is the control variable. To solve this problem, we first use orthogonal collocation to discretize the differential equations in each disjunction with 30 finite elements and three collocation points at each stage. The discretized model has 553 variables, 819 constraints, and three disjunctions. To apply the LD-SDA to this problem, we first reformulate the disjunctions using external variable

TABLE I

COMPUTATIONAL RESULTS OF PROBLEM (5)

Strategy	Solver	Obj	Time [s]	Status
MINLP BigM	KNITRO	-	0.05	Infeasible
	BARON	-9.18	900+	maxTimeLimit
MINLP Hull	KNITRO	-	9.75	Infeasible
	BARON	-12.74	900+	maxTimeLimit
L-Enumerate	KNITRO	-12.74	1.56	Optimal
	BARON	-9.18	900+	maxTimeLimit
LOA	KNITRO	-12.74	4.63	Optimal
	BARON	-12.74	900+	maxTimeLimit
GLOA	KNITRO	-12.74	5.03	Optimal
	BARON	-12.74	900+	maxTimeLimit
LD-SDA L_2	KNITRO	-12.74	1.50	Optimal
	BARON	-9.18	900+	maxTimeLimit
LD-SDA L_∞	KNITRO	-12.74	1.47	Optimal
	BARON	-9.18	900+	maxTimeLimit

$z_s \in \{1,2\}$. $z_s = 1$ represents when mode 1 is active and $z_s = 2$ when mode 2 is active at stage s.

The computational results for (5) are presented in Table I, and the time limit is set at 900 seconds. All the methods using BARON reach the maximum time limit and cannot prove global optimality within the time limit. KNITRO fails to solve the BigM and Hull reformulation and returns the infeasible termination condition, while all logic-based methods using KNITRO find the optimal solution within 5 seconds. In this problem, both LD-SDA L_2 and L_∞ explore all feasible disjunctions similarly to an enumeration over the logic space. LOA and GLOA terminate after around 5 seconds, slower than logic-based enumeration and D-SDA. These computational results show the advantage of the logic-based method over MINLP reformulations for this problem.

C. Multi-stage Dynamic Model Switching

Consider the following DAGDP with three dynamic modes and S stages and sequencing constraints:

$$\min_{\substack{x(t),u(t),\\ Y\in\{\bot,\top\}}} V(x) = -\int_{t_0}^{t_S} x^2(t) \tag{6a}$$

$$\text{s.t.} \begin{bmatrix} Y_{s,1} \\ \frac{dx}{dt} = \frac{-x}{e^{1-x}} + u \end{bmatrix} \veebar \begin{bmatrix} Y_{s,2} \\ \frac{dx}{dt} = \frac{0.5x^3+u}{20} \end{bmatrix} \veebar \begin{bmatrix} Y_{s,3} \\ \frac{dx}{dt} = \frac{x^2+u}{t+20} \end{bmatrix},$$

$$t \in [t_{s-1}, t_s] \tag{6b}$$

$$Y_{s,2} \Rightarrow \vee_{s'<s} Y_{s',1} \tag{6c}$$

$$Y_{s,2} \Rightarrow \neg \vee_{s'>s} Y_{s',1} \tag{6d}$$

$$Y_{s,3} \Rightarrow \vee_{s'<s} Y_{s',2} \tag{6e}$$

$$Y_{s,3} \Rightarrow \neg \vee_{s'>s} Y_{s',2} \tag{6f}$$

$$t_0 = 0, t_s = s \qquad \forall s = \{1,...,S\} \tag{6g}$$

$$x(t_0) = 1, u_t \in [-4,4], x \in [0,10]. \tag{6h}$$

Eqs. (6c)-(6f) are mode sequence constraints which denote that mode 1 should be performed before mode 2 and mode 2 before mode 3. We discretize the differential equations with 30 finite elements and three collocation points at each stage using orthogonal collocation. The statistics of the discretized models are presented in Table II.

Using LD-SDA to solve this problem requires a reformulation based on external integer variables. A straightforward reformulation is to define a variable $z_s \in \{1,2,3\}$ to represent the mode choice at each stage. However, this would result in a search space as high-dimensional as the original Boolean variables space. This space is limited to three or fewer points in each dimension, limiting the possibility of line search and resulting in slow convergence depending on the initial point. Therefore, we propose another reformulation based on the mode transition, which can also apply to other problems with sequence constraints, such as in scheduling and planning. Instead of using the integer variable to denote the mode choice at each stage, we use it to indicate whether the mode transition happens at each stage. Due to the sequence constraint, only two transitions are allowed. i.e., $\{A: 1 \to 2, B: 2 \to 3\}$, defining external variables $z_{s,A}, z_{s,B}$. Compared to the first reformulation, the high-dimensional space is reduced into a two-dimensional space. The relationship between mode transition and selection is expressed in Eqs. (6i)-(6j). Moreover, since mode transition might not occur within the stage horizon, we define extra variables in Eqs. (6k)-(6l) to capture this case and force one of the z_s variables of each transition to be *True* in Eqs. (6m)-(6n).

$$z_{s,A} \Leftrightarrow Y_{s-1,1} \wedge Y_{s,2} \qquad \forall s = \{2,...,S\} \tag{6i}$$

$$z_{s,B} \Leftrightarrow Y_{s-1,2} \wedge Y_{s,3} \qquad \forall s = \{3,...,S\} \tag{6j}$$

$$z_{S+1,A} \Leftrightarrow \neg \vee_{s\in\{2,...,S\}} Y_{s,2} \tag{6k}$$

$$z_{S+1,B} \Leftrightarrow \neg \vee_{s\in\{3,...,S\}} Y_{s,3} \tag{6l}$$

$$\veebar_{s\in\{2,...,S+1\}} z_{s,A} = True = \top \tag{6m}$$

$$\veebar_{s\in\{3,...,S+1\}} z_{s,B} = True = \top \tag{6n}$$

The computational results of solving problem (6) with $S = \{4,\ldots,9\}$ using LD-SDA, MINLP reformulation, and other logic-based methods are presented in Fig. 1. For the 4-stage problems, all the methods using BARON, except logic-based enumeration, can find the optimal solution. However, with increasing number of stages, none of these methods could guarantee to find the global optimal solution by closing the optimality gap within the time limit of up to 1 hour. Both LD-SDA L_2 and L_∞ using KNITRO as NLP solver converge to the optimal solution for all problems and outperform the other logic-based and MINLP reformulation methods.

The search path of LD-SDA L_2 and L_∞ for the nine-stage problem (6) is presented in Fig. 2. Both algorithms start from point $(1,2)$, which means that the $A: 1 \to 2$ mode transition occurs at stage 2, and the $B: 2 \to 3$ mode transition

TABLE II

STATISTIC OF THE MULTI-STAGE DYNAMIC SWITCHING PROBLEM (6)

# of stages	# of variables	# of constraints	# of disjunctions
4	741	1183	4
5	927	1547	5
6	1113	1911	6
7	1299	2275	7
8	1485	2639	8
9	1671	3003	9

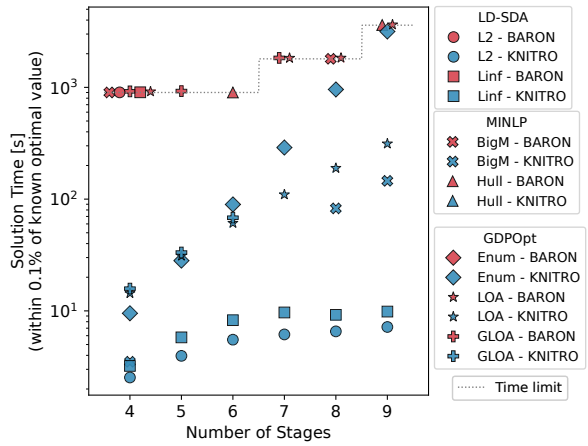

Fig. 1. Computational results of MINLP reformulation, LD-SDA, and other logic-based methods for varying S in problem (6)

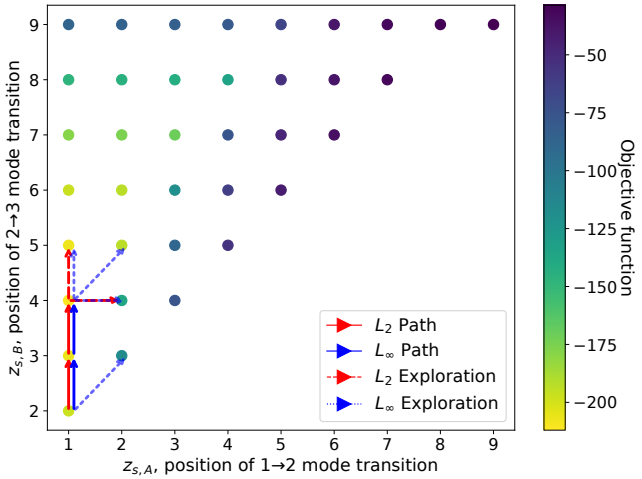

Fig. 2. Search path of the LD-SDA for problem (6) with $S = 9$

occurs at stage 3. After one round of neighbor search and a following line search, both LD-SDA L_2 and L_∞ converge to the optimal point $(1,4)$. Since LD-SDA L_∞ allows more search directions, it explores two more points than LD-SDA L_2 before termination. However, neither method converging to the global optimum is guaranteed. If the optimal point is not in the search directions of the starting or intermediate points, the LD-SDA could be trapped in a local optimum.

V. Conclusions and Future Work

This work presents the LD-SDA as an optimization tool for discrete dynamic optimization modeled via Differential Algebraic Generalized Disjunctive Programming (DAGDP). The LD-SDA exploits the logical structure of the problem and allows more flexibility in handling the DAE systems. Two examples based on dynamic mode selection are provided to evaluate the performance of LD-SDA. The computational results demonstrate that LD-SDA outperforms the equivalent MINLP formulation and other logic-based methods when DAEs are discretized via direct transcription. Moreover, an open-source implementation of the LD-SDA is provided, and this work integrates the model transformations in Pyomo.DAE and Pyomo.GDP for DAGDP modeling.

Future research directions include the integration of the LD-SDA with decomposition-based methods and exploring the effects of cutting planes on the LD-SDA.

References

[1] A. Flores-Tlacuahuac and L. T. Biegler, "Simultaneous mixed-integer dynamic optimization for integrated design and control," *Computers & chemical engineering*, vol. 31, no. 5-6, pp. 588–600, 2007.

[2] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1469–1475.

[3] N. Robuschi, C. Zeile, S. Sager, and F. Braghin, "Multiphase mixed-integer nonlinear optimal control of hybrid electric vehicles," *Automatica*, vol. 123, p. 109325, 2021.

[4] V. Bansal, J. D. Perkins, and E. N. Pistikopoulos, "A case study in simultaneous design and control using rigorous, mixed-integer dynamic optimization models," *Industrial & Engineering Chemistry Research*, vol. 41, no. 4, pp. 760–778, 2002.

[5] L. T. Biegler, *Nonlinear programming: concepts, algorithms, and applications to chemical processes*. SIAM, 2010.

[6] J. Kronqvist, D. E. Bernal, A. Lundell, and I. E. Grossmann, "A review and comparison of solvers for convex MINLP," *Optimization and Engineering*, vol. 20, pp. 397–455, 2019.

[7] R. Ruiz-Femenia, A. Flores-Tlacuahuac, and I. E. Grossmann, "Logic-Based Outer-Approximation Algorithm for Solving Discrete-Continuous Dynamic Optimization Problems," *Industrial & Engineering Chemistry Research*, vol. 53, no. 13, pp. 5067–5080, 2014.

[8] B. Baumrucker, J. G. Renfro, and L. T. Biegler, "MPEC problem formulations and solution strategies with chemical engineering applications," *Computers & Chemical Engineering*, vol. 32, no. 12, pp. 2903–2913, 2008.

[9] B. Burnak, N. A. Diangelakis, and E. N. Pistikopoulos, "Mixed-Integer Dynamic Optimization for Simultaneous Process Design and Control," in *Integrated Process Design and Operational Optimization via Multiparametric Programming*. Springer, 2020, pp. 21–46.

[10] B. Chachuat, A. B. Singer, and P. I. Barton, "Global methods for dynamic optimization and mixed-integer dynamic optimization," *Industrial & Engineering Chemistry Research*, vol. 45, no. 25, pp. 8373–8392, 2006.

[11] D. A. Linán, D. E. Bernal, L. A. Ricardez-Sandoval, and J. M. Gómez, "Optimal design of superstructures for placing units and streams with multiple and ordered available locations. Part I: A new mathematical framework," *Computers & Chemical Engineering*, vol. 137, p. 106794, 2020.

[12] D. A. Liñán and L. A. Ricardez-Sandoval, "A Discrete-Steepest Descent Framework for the Simultaneous Process and Control Design of Multigrade Reactive Distillation Columns," *IFAC-PapersOnLine*, vol. 55, no. 7, pp. 370–375, 2022.

[13] O. Palma-Flores, L. A. Ricardez-Sandoval, and L. T. Biegler, "Simultaneous design and NMPC control under uncertainty and structural decisions: A discrete-steepest descent algorithm," *AIChE Journal*, vol. 69, no. 11, p. e18188, 2023.

[14] D. E. Bernal, D. Ovalle, D. A. Liñán, L. A. Ricardez-Sandoval, J. M. Gómez, and I. E. Grossmann, "Process Superstructure Optimization through Discrete Steepest Descent Optimization: a GDP Analysis and Applications in Process Intensification," in *Computer Aided Chemical Engineering*. Elsevier, 2022, vol. 49, pp. 1279–1284.

[15] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, D. L. Woodruff, *et al.*, *Pyomo-optimization modeling in python*. Springer, 2021, vol. 67.

[16] Q. Chen, E. S. Johnson, D. E. Bernal, R. Valentin, S. Kale, J. Bates, J. D. Siirola, and I. E. Grossmann, "Pyomo. GDP: an ecosystem for logic based modeling and optimization development," *Optimization and Engineering*, vol. 23, no. 1, pp. 607–642, 2022.

[17] I. E. Grossmann, *Advanced optimization for process systems engineering*. Cambridge University Press, 2021.

[18] K. Murota, "Discrete convex analysis," *Mathematical Programming*, vol. 83, no. 1, pp. 313–371, 1998.