Combinatorial Integral Approximation Decompositions for Mixed-Integer Optimal Control

Clemens Zeile · Tobias Weber · Sebastian Sager

Received: date / Accepted: date

Abstract Solving mixed-integer nonlinear programs (MINLPs) is hard in theory and practice. Decomposing the nonlinear and the integer part seems promising from a computational point of view. In general, however, no bounds on the objective value gap can be guaranteed and iterative procedures with potentially many subproblems are necessary. The situation is different for mixed-integer optimal control problems with binary choices that switch over time. Here, a priori bounds were derived for a decomposition into one continuous nonlinear control problem and one mixed-integer linear program, the combinatorial integral approximation (CIA) problem.

We generalize and extend the decomposition idea. First, we derive different decompositions and analyze the implied a priori bounds. Second, we propose several strategies to generate promising candidate solutions for the binary control functions in the original problem. We present the extensions for problems constrained by ordinary differential equations. They are transferable in a straightforward way though to recently suggested variants for certain partial differential equations, for algebraic equations, for additional combinatorial constraints, and for discrete time problems.

All algorithms and subproblems were implemented in AMPL for proof of concept. Numerical results show the improvement compared to standard CIA decomposition with respect to objective function value and compared to general purpose MINLP solvers with respect to runtime.

Keywords Optimal control · Switched Dynamic Systems · Mixed-integer Nonlinear Programming · Mixed-integer Linear Programming · Ordinary Differential Equations · Approximation methods and heuristics

Mathematics Subject Classification (2000) 49K15 · 34H05 · 93C65 · 90C59

C. Zeile
Faculty of Mathematics, Otto-von-Guericke-University Magdeburg, Germany
E-mail: clemens.zeile@ovgu.de

T. Weber, as above, E-mail: tobias.weber@ovgu.de
- S. Sager, as above, E-mail: sager@ovgu.de
1 Introduction

The goal of optimal control is to find control functions and state trajectories that are feasible and optimal. State trajectories are solutions of systems of differential equations for given control functions and boundary conditions. Feasibility refers to constraints on control functions and differential states, optimality refers to an objective functional of controls and states. Mixed-integer optimal control problems (MIOCP) do have additional integrality constraints on some of the control functions. Hence, the differential equations of the underlying system depend on the value of integer control functions, or equivalently, one can switch instantly between several differential equations, [7, 38, 6, 14]. This problem class is ubiquitous in various application areas, e.g., water, gas, traffic, and supply chain networks [5, 28, 18, 9, 17, 16], distributed autonomous systems [1], processes in chemical engineering that involve valves [25, 39], or the choice of gears in automotive control [11, 27]. We chose problems from a web-based MIOCP benchmark collection [33] to evaluate our algorithms.

Several families of methods have been proposed for solving MIOCPs. This comprehend indirect or first-optimize then-discretize approaches [13], Dynamic Programming or Hamilton-Jacobi-Bellman [21], moment relaxations [35], switching time optimization [11, 29, 42, 41, 30], and direct or first-discretize then-optimize approaches [6, 10, 31]. Surveys, references, and comparisons can be found, e.g., in [13, 35, 37, 43].

Our approach is based on previous ideas [4, 31, 36, 24] to decompose the MIOCP into a relaxed continuous optimal control problem (OCP) and a mixed-integer linear program (MILP). In contrast to general MINLP decompositions (see [40] for recent results on error bounds for rounding approaches and [2] for a MINLP survey), the particular setting with dependent (states) and independent (controls) variables allows the derivation of a priori bounds, [26, 32]. As specified in Section 4, the difference between state trajectories $x(\cdot)$ and $y(\cdot)$ that are the unique solutions of the initial value problems for given control functions $\omega$ and $\alpha$, respectively, is bounded for a constant $C$ and all $t \in T$ by

$$||x(t) - y(t)|| \leq C \max_{r \in \mathcal{T}} \left|\int_{t_0}^{t} \alpha(\tau) - \omega(\tau) \, d\tau \right|.$$  

This motivates to solve the relaxed problem (OCP) to obtain $\alpha(\cdot)$, to calculate an integer control function $\omega$ in a second step such that the maximum on the right hand side of (1) is as small as possible, and to obtain $y(\cdot)$ by simulation. Continuity of constraint and objective functions implies “good” behavior with respect to feasibility and objective function value in the original MIOCP. The combinatorial integral approximation (CIA) problem in the second step can be formulated as a MILP [36] and
can be solved either with generic MILP methods, with specifically tailored Branch and Bound methods [24], and in some cases even in linear time with the Sum Up Rounding method [31].

The idea of CIA decomposition has also been used in the context of hyperbolic partial differential equations [20,19], differential-algebraic equation systems [13], with additional combinatorial constraints [36,17], and based on discrete time formulations [22,17]. In this publication we generalize all of these approaches by presenting alternative ways to calculate $\omega$ based on $\alpha$, using different MILPs. We formulate them for the case of initial value problems with ordinary differential equations, although they are applicable to all mentioned variants in a straightforward way.

Our results are also independent of the method that is applied to solve the relaxed problem (OCP). For the numerical results, we are going to use a \textit{first-discretize then-optimize} approach with Radau collocation. \textit{First-discretize} refers a) to approximating the control functions with parameterized basis functions, such as finitely many piecewise constant functions, and b) to relaxing path and control constraints from the domain of a time horizon to finitely many time points. The \textit{then-optimize} refers to solving the resulting finite dimensional optimization problem numerically to optimality. An overview of direct methods for continuous optimal control problems can be found in, e.g., [3,12]. For comparison, we also apply this approach directly to the MIOCP and solve the resulting mixed-integer nonlinear program (MINLP) with the general purpose solver Bonmin.

For a general problem class of MIOCPs the integer gap depends linearly on the control discretization [34]. For many applications this control discretization is not fixed, but can be refined. Thus, the integer gap can be driven to zero (usually at the expense of frequent switching). For practical reasons good or even optimal solutions for a fixed control discretization are of interest, which is our main focus.

\textbf{Contributions.} We derive different versions of the CIA problem, leading to multiple MILPs. Noting that the computational effort to solve one MILP is usually small compared to the solution of the relaxed nonlinear problem and even smaller compared to the deterministic solution of the original MIOCP, we propose to solve several approximation problems. These solutions are candidate solutions themselves, and can be recombined into new switching sequences. We derive theoretical a priori bounds for them, discuss computational (dis)advantages, and show numerically the improvement to existing decomposition approaches with respect to objective function value and to general purpose MINLP solvers with respect to runtime.

The \textit{outline} of the article is as follows. We define the considered MIOCP class and propose a general decomposition framework in Section 2. The algorithm consists of several MILP formulations, which are discussed in Section 3. We discuss theoretical properties in Section 4. In Section 5 we look at strategies that combine and improve existing binary control functions. In Section 6 we provide and discuss numerical results for the MIOCP benchmark library [33].
2 Problem class, definitions, and main algorithm

We denote the considered time horizon by $\mathcal{T} := [t_0, t_1] \subset \mathbb{R}$ and write “for a.e. $t \in \mathcal{T}$” for all $t \in \mathcal{T}$ except on a set of measure zero. $C^k(\mathcal{F}_1, \mathcal{F}_2)$ is the space of $k$-times continuously differentiable functions $f : \mathcal{F}_1 \to \mathcal{F}_2$ where $\mathcal{F}_1, \mathcal{F}_2$ depict subsets of normed spaces. Similarly, we write $L^\infty(\mathcal{F}_1, \mathcal{F}_2)$ for the space of functions $f : \mathcal{F}_1 \to \mathcal{F}_2$ with bounded norm $\|f(x)\|_{\mathcal{F}_2}$ for almost all $x \in \mathcal{F}_1$, and $\mathcal{W}^\infty(\mathcal{F}_1, \mathcal{F}_2)$ for the space of Fréchet differentiable functions with respect to $L^\infty(\mathcal{F}_1, \mathcal{F}_2)$. The null vector is written as $\mathbf{0}$. We are interested in the following class of mixed-integer optimal control problems.

**Definition 1** (MIOCP) We refer to the following control problem (2) as (MIOCP).

$$\min_{\mathbf{x}, \omega} \Phi(\mathbf{x}(t))$$

s.t. $$\dot{\mathbf{x}}(t) = f_0(t, \mathbf{x}(t)) + \sum_{i=1}^{n_\omega} \omega_i(t) f_i(t, \mathbf{x}(t)), \quad \text{for a.e. } t \in \mathcal{T},$$

$$\mathbf{x}(t_0) = \mathbf{x}_0,$$

$$1 = \sum_{i=1}^{n_\omega} \omega_i(t) \quad \text{for a.e. } t \in \mathcal{T},$$

$$\omega(t) \in \{0, 1\}^n \quad \text{for a.e. } t \in \mathcal{T},$$

$$\mathbf{0} \leq c(t, \mathbf{x}(t)) \quad \text{for a.e. } t \in \mathcal{T}.$$  

We minimize a Mayer term $\Phi \in C^1(\mathbb{R}^{n_u}, \mathbb{R})$ over differential states $\mathbf{x} \in \mathcal{W}^\infty(\mathcal{T}, \mathbb{R}^{n_x})$ for binary control functions $\omega \in L^\infty(\mathcal{T}, \{0, 1\}^n)$ with bounded norm $\|f(x)\|_{\mathcal{F}_2}$ for almost all $x \in \mathcal{F}_1$, and $\mathcal{W}^\infty(\mathcal{F}_1, \mathcal{F}_2)$ for the space of Fréchet differentiable functions with respect to $L^\infty(\mathcal{F}_1, \mathcal{F}_2)$. The null vector is written as $\mathbf{0}$. We are interested in the following class of mixed-integer optimal control problems.

**Definition 2** (OCP) We define (OCP) as the canonical relaxation of (MIOCP) with respect to (2e). We substitute hence $\omega \in L^\infty(\mathcal{T}, \{0, 1\}^n)$ by $\alpha \in L^\infty(\mathcal{T}, [0, 1]^n)$. 

Remark 1 See [13, 32, 37] for a discussion of the generality of (MIOCP) and extensions to cope with, e.g., Lagrange functionals, boundary and multi-point constraints, vanishing constraints, free final time, control values and the like. Particularly interesting are continuous control functions $u \in L^\infty(\mathcal{T}, \mathcal{U})$ that often enter (2b,2f) in practical applications. From a theoretical point of view, in the interest of comparability, and for computational speed it is convenient to consider the continuous controls $u(\cdot)$ as fixed to the solution that was obtained by solving the continuous relaxation of (MIOCP) in our approach. It is also possible though, and often makes sense in practice, to improve the objective function value by reoptimizing $u$ when (MIOCP) is evaluated for fixed $\omega$. For notational convenience and without loss of generality, we omit the continuous controls $u$ in the following. An evaluation of (MIOCP) for fixed $\omega$ is then a solution of an initial value problem.

Our algorithm solves continuous relaxations of (MIOCP), defined as follows.

**Definition 2** (OCP) We define (OCP) as the canonical relaxation of (MIOCP) with respect to (2e). We substitute hence $\omega \in L^\infty(\mathcal{T}, \{0, 1\}^n)$ by $\alpha \in L^\infty(\mathcal{T}, [0, 1]^n)$. 

The problem (OCP) in function space can be solved by different approaches, as mentioned above. To be able to work with MILPs to approximate control functions, we map between function space and $[0, 1]^{n_a \times M}$ using a time grid as follows.

**Definition 3** ($\mathcal{G}_\theta, \Delta, \varphi, \varphi^{-1}, \Omega, W$) Let the ordered set $\mathcal{G}_\theta := \{t_0 < \ldots < t_M = T\}$ denote a time grid with $\Delta_j = t_{j+1} - t_j$ and $\Delta_{\max} = \max_j \Delta_j$ for $j = 0 \ldots M - 1$.

We define the mapping

$$\varphi : [0, 1]^{n_a \times M} \rightarrow L^\infty(\mathcal{T}, [0, 1]^{n_a}), \quad \alpha = \varphi(a)$$

using $n_\theta$ piecewise constant functions

$$\alpha_i(t) := a_{i,j}, \quad i \in 1, \ldots, n_\theta, \quad t \in [t_j, t_{j+1}), \quad j = 0 \ldots M - 1, \quad t_j \in \mathcal{G}_\theta.$$ 

A mapping in reverse direction

$$\varphi^{-1} : L^\infty(\mathcal{T}, [0, 1]^{n_a}) \rightarrow [0, 1]^{n_a \times M}, \quad a = \varphi^{-1}(\alpha)$$

is defined by extracting integrals on the grid $\mathcal{G}_\theta$,

$$a_{i,j} := \frac{1}{\Delta_j} \int_{t_j}^{t_{j+1}} \alpha_i(\tau) \, d\tau, \quad i \in 1, \ldots, n_\theta, \quad j = 0 \ldots M - 1, \quad t_j \in \mathcal{G}_\theta.$$ 

Both maps conserve integrality, i.e., $\alpha = \varphi(a) \in L^\infty(\mathcal{T}, \{0, 1\}^{n_a})$ for $\alpha \in \{0, 1\}^{n_a \times M}$, and the other way around. We denote integrality and SOSI using the sets

$$\Omega := \left\{ \omega \in L^\infty(\mathcal{T}, \{0, 1\}^{n_a}) : 1 = \sum_{i=1}^{n_a} \omega_i(t) \text{ for a.e. } t \in \mathcal{T} \right\},$$

$$W := \left\{ w \in \{0, 1\}^{n_a \times M} : 1 = \sum_{i=1}^{n_a} w_{i,j} \text{ for } j = 0 \ldots M - 1 \right\}.$$ 

To scale control variables we are going to need function evaluations and adjoint (dual) variables on the grid $\mathcal{G}_\theta$.

**Definition 4** ($\lambda, \tilde{f}$) Let for a solution of (OCP) $\lambda_{j,k} \in \mathbb{R}, t_j \in \mathcal{G}_\theta, \quad k = 1, \ldots, n_k$ be the discretized and evaluated dual variables and $\{\tilde{f}_{i,j,k}\}_{k \in 1, \ldots, n_k}$ be the entries of

$$\mathbb{R}^{n_k} \ni \tilde{f}_{i,j} := \frac{1}{\Delta_{t_j}} \int_{t_j}^{t_{j+1}} \tilde{f}_i(\tau, x(\tau)) \, d\tau, \quad i \in 1, \ldots, n_\theta, \quad t_j \in \mathcal{G}_\theta.$$ 

We propose to use Algorithm 1 to approximate the solution of (MIOCP) efficiently with a priori bounds. Relaxing (MIOCP) to (OCP) results in state and control trajectories (Line 1). We solve different MILPs to approximate the relaxed controls with binary ones in Lines 2–3. Their performance is evaluated in Line 4 by calculating their corresponding state trajectories and objective values. In Lines 6–7, we use the binary controls in several recombination heuristics in order to create new candidate binary controls, which we evaluate as well (Line 8). Finally, the best performing binary control is selected as solution in Line 10.
Algorithm 1: Decomposition of (MIOCP)

Input: (MIOCP) instance, grid $G$, algorithmic choices in sets $S_{CIA}$ and $S_{REC}$.

1. Solve (OCP) $\to \Phi_{rel}, x, a = \varphi^{-1}(\alpha)$.
2. for milp $\in S_{CIA}$ do
   3. Solve milp for data $a$ with MILP solver $\to w_{milp}$.
   4. Evaluate (MIOCP) with fixed $\omega_{milp} = \varphi(w_{milp}) \to \Phi_{milp}, x$.
3. end
4. for rec $\in S_{REC}$ do
   5. Create $w_{rec}$ using $w_{milp}, \Phi_{milp}$ from all milp $\in S_{CIA}$.
   6. Evaluate (MIOCP) with fixed $\omega_{rec} = \varphi(w_{rec}) \to \Phi_{rec}, x$.
5. end
6. Set $\Phi^* = \min \{ \min_{milp \in S_{CIA}} \Phi_{milp}, \min_{rec \in S_{REC}} \Phi_{rec} \}$, return $\Phi^*, x^*, w^*$ and lower bound $\Phi_{rel}$.

The main idea of Algorithm 1 is to decouple controls and states. We approximate the relaxed control function $\alpha$ that is optimal for (OCP) with binary controls $\omega$ such that a good objective function value is obtained when (MIOCP) is evaluated for $\omega$. Which MILPs and recombination heuristics are used in the algorithm depends on the definition of the sets $S_{CIA}$ and $S_{REC}$, which we discuss in Section 3 and Section 5. A theoretical motivation and error bounds are given in Section 4. Algorithm 1 is a generalization of the decomposition approach in [24,31,36], for which $S_{REC}$ is empty and $S_{CIA}$ contains only one CIA problem formulation.

3 Combinatorial Integral Approximation MILPs

We are going to define the following MILP formulations of combinatorial integral approximation type for $S_{CIA}$ in Algorithm 1

$$S_{CIA} := \{(\text{CIAmax}), (\text{CIA1}), (\text{CIAmaxB}), (\text{CIA1B}), (\lambda\text{CIA1}), (\lambda\text{CIA1B}), (\text{SCIAmax}), (\text{SCIA1}), (\text{SCIAmaxB}), (\text{SCIA1B})\}$$

CIA, $\lambda$CIA, and SCIA refer to different scalings (Section 3.1), 1 and max to different norms $\| \cdot \|$ (Section 3.2), and B to a reversal of time (Section 3.3).

3.1 Combinatorial integral approximation and scaled variants

We consider the following approximation problems in the control function space.

Definition 5 (CIA, SCIA, $\lambda$CIA) Let $x, \alpha$ be optimal for (OCP). Let $f, j(t, x(t))$ be evaluations of the right hand side functions, and $\lambda \in L^\infty(\mathcal{T}, \mathbb{R}^n)$ be the dual variables of the ODE constraint (2b). We define the following (scaled) Combinatorial
Integral Approximation problems and corresponding optimal objective values,

\[ \delta^*_{\text{CIA}} := \min_{\omega(t) \in \Omega} \max_{t \in T} \left| \int_0^t \alpha(\tau) - \omega(\tau) \, d\tau \right|, \quad (\text{CIA}) \]

\[ \delta^*_{\text{SCIA}} := \min_{\omega(t) \in \Omega} \max_{t \in T} \left| \int_0^t \sum_{i=1}^{n_0} (\alpha_i(\tau) - \omega_i(\tau)) f_i(t, x(t)) \, d\tau \right|, \quad (\text{SCIA}) \]

\[ \delta^*_{\lambda,\text{CIA}} := \min_{\omega(t) \in \Omega} \max_{t \in T} \left| \sum_{k=1}^{n_x} \lambda_{t,k} \cdot \int_0^t \sum_{i=1}^{n_0} (\alpha_i(\tau) - \omega_i(\tau)) f_{i,k}(t, x(t)) \, d\tau \right|. \quad (\lambda \text{CIA}) \]

Discretizing with \( a = \varphi^{-1}(\alpha), \tilde{f}, \) and \( \lambda \) from Definition 4 results in

\[
\begin{align*}
\min_{w, t_l \in G_\omega} & \left| \sum_{j=0}^{l-1} (a_{l,j} - w_{l,j}) \cdot \Delta t_j \right|, \\
\min_{w, t_l \in G_\omega} & \left| \sum_{j=0}^{l-1} \sum_{i=1}^{n_0} (a_{i,j} - w_{i,j}) \tilde{f}_{i,j} \cdot \Delta t_j \right|, \\
\min_{w, t_l \in G_\omega} & \left| \sum_{k=1}^{n_x} \lambda_{t,k} \cdot \sum_{j=0}^{l-1} \sum_{i=1}^{n_0} (a_{i,j} - w_{i,j}) \tilde{f}_{i,j,k} \cdot \Delta t_j \right|,
\end{align*}
\]

respectively. To facilitate notation, we now assume an equidistant grid with \( \Delta_j = 1 \).

3.2 Norm dependent MILP formulation

We introduce slack variables to formulate (3–5) with different norms as MILPs. We start with the maximum norm in (3–4).

**Definition 6 (CIAmax, SCIAmax) For given data \( a \) we define (CIAmax) as**

\[
\begin{align*}
\min_{\delta, w, t_l \in G_\omega} & \delta, \\
\text{s.t.} & \quad \delta \geq \pm \sum_{j=0}^{l} a_{l,j} - w_{l,j}, \quad \text{for } i = 1, \ldots, n_\omega, \quad t_l \in G_\omega, \quad (6a) \\
& \delta \geq \pm \sum_{j=0}^{l} \sum_{i=1}^{n_0} (a_{i,j} - w_{i,j}) \tilde{f}_{i,j} \cdot \Delta t_j, \quad (6b)
\end{align*}
\]

and (SCIAmax) as

\[
\begin{align*}
\min_{\delta, w, t_l \in G_\omega} & \delta, \\
\text{s.t.} & \quad \delta \geq \pm \sum_{j=0}^{l} \sum_{i=1}^{n_0} (a_{i,j} - w_{i,j}) \tilde{f}_{i,j,k} \cdot \Delta t_j, \quad \text{for } k = 1, \ldots, n_x, \quad t_l \in G_\omega, \quad (7a) \\
& \delta \geq \pm \sum_{j=0}^{l} \sum_{i=1}^{n_0} (a_{i,j} - w_{i,j}) \tilde{f}_{i,j,k} \cdot \Delta t_j, \quad (7b)
\end{align*}
\]

We now define MILPs based on the Manhattan norm.
Definition 7 (CIA1, SCIA1, λCIA1) For given data \( \mathbf{a} \) and slack variables \( s_{i,t} \) we define (CIA1) as

\[
\min_{\delta, s_{i,t}, \mathbf{w} \in W} \delta
\]

s.t. \( \delta \geq \sum_{i=1}^{n_\omega} s_{i,t}, \quad \text{for } t_t \in G_\omega, \) \hspace{1cm} (8b)

\( s_{i,t} \geq \pm \sum_{j=0}^{l} (a_{i,j} - w_{i,j}), \quad \text{for } i = 1, \ldots, n_\omega, \quad t_t \in G_\omega \) \hspace{1cm} (8c)

and with a different dimension for \( s_{k,t} \) we define (SCIA1) as

\[
\min_{\delta, s_{k,t}, \mathbf{w} \in W} \delta
\]

s.t. \( \delta \geq \sum_{k=1}^{n_x} s_{k,t}, \quad \text{for } t_t \in G_\omega, \) \hspace{1cm} (9b)

\( s_{k,t} \geq \pm \sum_{j=0}^{l} \sum_{i=1}^{n_\omega} (a_{i,j} - w_{i,j}) \cdot \tilde{f}_{i,j,k}, \quad \text{for } k = 1, \ldots, n_x, \quad t_t \in G_\omega. \) \hspace{1cm} (9c)

Motivated by the similarity of (5), we define (λCIA1) by replacing (9c) as

\[
\min_{\delta, s_{k,t}, \mathbf{w} \in W} \delta
\]

s.t. \( \delta \geq \sum_{k=1}^{n_x} s_{k,t}, \quad \text{for } t_t \in G_\omega, \) \hspace{1cm} (10b)

\( s_{k,t} \geq \pm \lambda_{t,k} \cdot \sum_{j=0}^{l} \sum_{i=1}^{n_\omega} (a_{i,j} - w_{i,j}) \cdot \tilde{f}_{i,j,k}, \quad \text{for } k = 1 \ldots n_x, t_t \in G_\omega. \) \hspace{1cm} (10c)

Of course also other norms, such as the Euclidean norm, can be used. We do not consider them here, because of the nonlinearity of the constraints.

3.3 Chronologically ordered constraints

One further possibility to modify MILPs is the chronological order in the constraints for the accumulated difference \( \| \mathbf{a} - \mathbf{w} \| \). Instead of starting from \( t_0 \), we may use an arbitrary ordering of time points. We consider backwards accumulation, starting from the interval \([t_{M-1}, t_M]\):

\[
\delta \geq \pm \sum_{j=1}^{M-1} a_{i,j} - w_{i,j}, \quad \text{for } i = 1, \ldots, n_\omega, \quad t_t \in G_\omega. \] \hspace{1cm} (11)

We denote the problem where (11) replaces (6b) in (CIAmax) by (CIAmaxB). The other defined problems can be modified analogously with backward time accumulation and are named accordingly.
4 A priori bounds for CIA decompositions

We revise results for the a priori bounds resulting from a CIA decomposition [34] and extend them to alternative decompositions that may be used in $S_{\text{CIA}}$ in Algorithm 1.

4.1 Combinatorial integral approximation

This variant of Gronwall’s Lemma is used in the following proofs.

**Lemma 1** Let us first define $\|v(t)\|_{\lambda(t)} := \|\sum_{k=1}^{n_{\alpha}} \lambda_k(t) \cdot v_k(t)\|$, $\lambda(t), v(t) \in L^\infty(\mathcal{T}, \mathbb{R}^n)$. Let $z \in L^\infty(\mathcal{T}, \mathbb{R})$, and for a constant $L_G \in \mathbb{R}_+^+$ it holds

$$\|v(t)\|_{\lambda(t)} \leq z(t) + L_G \int_{t_0}^{t} \|v(\tau)\|_{\lambda(\tau)} d\tau$$

for $t \in \mathcal{T}$ a.e. Then, it also holds

$$\|v(t)\|_{\lambda(t)} \leq \|z(\cdot)\|_{\infty} \cdot e^{L_G(t-t_0)}.$$  

**Proof** See [11], pp. 169-182, for the case with $\lambda(t) = 1, \forall t \in \mathcal{T}$ and without applied absolute value. The proof is applicable to the general case where $v(t) \in L^\infty(\mathcal{T}, \mathbb{R})$ is replaced by $\|v(t)\|_{\lambda(t)}$, which is by the choice of $v(t), \lambda(t)$ also part of $L^\infty(\mathcal{T}, \mathbb{R})$.

The idea behind the following results is to analyze the evolution of two trajectories $x$ and $y$ based on the same ODE system but driven by two different controls $\alpha$ and $\omega$. We want to compute the distance between the two trajectories depending on the distance of the controls.

**Theorem 1** Let $\alpha, \omega, f_0, f_i$ be given as defined previously and $t \in \mathcal{T}$. Let $x(\cdot)$ and $y(\cdot)$ be the unique solutions of the initial value problems

$$x(t) = f_0(t, x(t)) + \sum_{i=1}^{n_{\alpha}} \alpha_i(t) f_i(t, x(t)), \quad x(t_0) = x_0, \quad (14a)$$

$$y(t) = f_0(t, y(t)) + \sum_{i=1}^{n_{\alpha}} \omega_i(t) f_i(t, y(t)), \quad y(t_0) = y_0. \quad (14b)$$

Assume there are positive constants $\delta, C, L_i, B \in \mathbb{R}^+$, $i = 0, \ldots, n_{\alpha}$, together with a vector norm $\|\cdot\|$ and a consistent matrix norm $\|\cdot\|$ such that for all $t \in \mathcal{T}$:

$$\left\| \frac{d}{dt} f_i(t, x(t)) \right\| \leq C, \quad \text{for } i = 1, \ldots, n_{\alpha}, \quad (14c)$$

$$\left\| f_i(t, y(t)) - f_i(t, x(t)) \right\| \leq L_i \left\| y(t) - x(t) \right\|, \quad \text{for } i = 0, \ldots, n_{\alpha}. \quad (14d)$$

Furthermore, let $f_i(\cdot, x(\cdot)), i = 1, \ldots, n_{\alpha}$ be essentially bounded by $B$ on $\mathcal{T}$, and it holds for all $t \in \mathcal{T}$

$$\left\| \int_{t_0}^{t} \alpha(\tau) - \omega(\tau) \, d\tau \right\| \leq \delta. \quad (14e)$$
then it also holds for all \( t \in \mathcal{T} \)

\[
||y(t) - x(t)|| \leq (||y_0 - x_0|| + n_\alpha (B + C(t - t_0)) \delta) e^{L(t - t_0)}.
\]  

(14f)

**Proof** We start with some notations and observations before we approximate the normed difference of \( y \) and \( x \). At first, the FUNDAMENTAL THEOREM OF CALCULUS yields for (14a, 14b) and \( t \in \mathcal{T} \)

\[
x(t) = x_0 + \int_{t_0}^{t} x(\tau) \, d\tau = x_0 + \int_{t_0}^{t} f_0(\tau, x(\tau)) + \sum_{i=1}^{n_\alpha} \alpha_i(\tau) f_i(\tau, x(\tau)) \, d\tau,
\]

\[
y(t) = y_0 + \int_{t_0}^{t} y(\tau) \, d\tau = y_0 + \int_{t_0}^{t} f_0(\tau, y(\tau)) + \sum_{i=1}^{n_\alpha} \alpha_i(\tau) f_i(\tau, y(\tau)) \, d\tau.
\]

From the definition of \( \alpha, \omega \) it is clear that

\[
||\alpha(t)|| \leq 1, \quad ||\omega(t)|| \leq 1.
\]

For brevity we define the auxiliary terms:

\[
L := \sum_{i=0}^{n_\alpha} L_i, \quad \Delta \omega_i(t) := \alpha_i(t) - \omega_i(t), \quad \Delta a_i := \int_{t_0}^{t} \Delta \omega_i(\tau) \, d\tau.
\]

Note that \( \Delta a_i(0) = 0 \) and because of (14e) it holds \( ||\Delta a_i(t)|| \leq \delta \). Using these observations, it follows for all \( t \in \mathcal{T} \)

\[
||y(t) - x(t)|| \leq ||y_0 - x_0|| + \left|\int_{t_0}^{t} f_0(\tau, y(\tau)) + \sum_{i=1}^{n_\alpha} \omega_i(\tau) f_i(\tau, y(\tau)) \right| \left|\alpha_0(\tau) - f_0(\tau, x(\tau)) \right| d\tau
\]

\[
\leq ||y_0 - x_0|| + \left|\int_{t_0}^{t} f_0(\tau, y(\tau)) + \sum_{i=1}^{n_\alpha} \omega_i(\tau) f_i(\tau, y(\tau)) - f_0(\tau, x(\tau)) \right| d\tau
\]

\[
\leq ||y_0 - x_0|| + \left|\int_{t_0}^{t} f_0(\tau, x(\tau)) + \sum_{i=1}^{n_\alpha} \omega_i(\tau) f_i(\tau, x(\tau)) \right| d\tau
\]

\[
\leq ||y_0 - x_0|| + \left|\int_{t_0}^{t} \sum_{i=0}^{n_\alpha} |f_i(\tau, y(\tau)) - f_i(\tau, x(\tau))| \, d\tau \right| + \left|\int_{t_0}^{t} \sum_{i=1}^{n_\alpha} \omega_i(\tau) f_i(\tau, x(\tau)) \right| d\tau
\]

\[
\leq ||y_0 - x_0|| + \left|\int_{t_0}^{t} \sum_{i=0}^{n_\alpha} |f_i(\tau, y(\tau)) - f_i(\tau, x(\tau))| \, d\tau \right|
\]

\[
\leq ||y_0 - x_0|| + \left|\int_{t_0}^{t} \sum_{i=1}^{n_\alpha} \omega_i(\tau) f_i(\tau, x(\tau)) \right| d\tau
\]

\[
\leq ||y_0 - x_0|| + \int_{t_0}^{t} \sum_{i=0}^{n_\alpha} L_i \, ||y(\tau) - x(\tau)|| \, d\tau
\]
Proof From the proof of Theorem 1 follows for all \( t \) which satisfy the assumptions of the lemma, and using the result on the last inequality yields the claim
\[
||y(t) - x(t)|| \leq (||y_0 - x_0|| + n_\omega (B + C(t - t_0)) \delta) e^{L(t-t_0)}.
\]

\( \square \)

**Corollary 1** (Approximation bounds via \((\text{CIA})\))

Assume that \( y_{\text{CIA}}(\cdot), x(\cdot) \) are the solutions of the IVP of the above theorem, where the binary solutions of \((\text{CIA})\) have been applied for \( y \). Then, the state approximation error is bounded for all \( t \in \mathcal{T} \) by
\[
||y_{\text{CIA}}(t) - x(t)|| \leq (||y_0 - x_0|| + (B + C(t - t_0)) \delta_{\text{CIA}}) e^{L(t-t_0)}.
\]

4.2 Scaled combinatorial integral approximation

The following result shows that \((\text{CIA})\) is dominated by \((\text{SCIA})\) in terms of approximation accuracy.

**Corollary 2** (Approximation bounds via \((\text{SCIA})\))

Apply the notations from Theorem 1 and let \( y_{\text{SCIA}} \) be analogously defined as in Corollary 1. It follows for all \( t \in \mathcal{T} \):
\[
||y_{\text{SCIA}}(t) - x(t)|| \leq (||y_0 - x_0|| + \delta_{\text{SCIA}}) e^{L(t-t_0)}
\]
\[
\leq (||y_0 - x_0|| + n_\omega (B + C(t - t_0)) \delta_{\text{SCIA}}) e^{L(t-t_0)}.
\]

**Proof** From the proof of Theorem 1 follows for \( t \in \mathcal{T} \) and any \( \omega \) with corresponding \( y \)
\[
\left| \left| \int_{t_0}^{t} \sum_{i=1}^{n_\omega} \Delta a_i(t) \hat{f}_i(t, x(t)) \, d\tau \right| \right| \leq n_\omega (B + C(t - t_0)) \cdot \left| \left| \int_{t_0}^{t} \alpha(\tau) - \omega(\tau) \, d\tau \right| \right|. 
\]
Taking the minimum yields
\[
\delta^*_{\text{SCIA}} \leq \left\| \int_{t_0}^{t} \sum_{i=1}^{n\omega} (\alpha_i(t) - \omega^{\text{CIA}}(t)) f_i(t, x(t)) \, dt \right\| \leq n\omega (B + C(t - t_0)) \delta^*_{\text{CIA}}.
\]

The corollary states bounds for the normed difference of the trajectories based on relaxed and (SCIA)- binary controls. Moreover, the bounds are compared with those for (CIA)- binary controls from Theorem 1. The scaled bound turns out to be tighter than the existing (CIA)- bound. Thus, it is obvious to consult these alternative binary controls for an approximation study. Ideally, \(\delta^*_{\text{SCIA}} < n\omega (B + C(t - t_0)) \delta^*_{\text{CIA}}\) holds and results in improved objective values. Yet, we confine this hope already with the following remark.

Remark 2 Using (SCIA) for binary control approximation results not necessarily in better state approximation or objective value than using (CIA). The computational results section specifies instances, but anticipate already what can happen:

1. It may hold
   \[|\|y^{\text{CIA}}(t) - x(t)||| < |\|y^{\text{SCIA}}(t) - x(t)||| < \delta^*_{\text{SCIA}}\]
   for \(t \in T\).
2. Even if the above inequality holds reversely, the computed trajectories may fulfill
   \[\Phi(y^{\text{CIA}}(t)) < \Phi(y^{\text{SCIA}}(t))\]
   because of, e.g., non-convexities of the objective.

4.3 \(\lambda\)-combinatorial integral approximation

The \(\lambda\)-approximation stems from the aforementioned theorem of approximating differential states, but with assessing the difference of the soon defined cost-to-go function. We define it in a more general MIOCP setting with a Bolza objective

\[
\Phi(x(t_f)) + \int_{t \in \mathcal{T}} L(x(t), \omega(t)) \, dt,
\]

where \(L \in C^1(\mathbb{R}^{n_x+1} \times \mathbb{R}^{n_\omega}, \mathbb{R})\).

Definition 8 (Cost-to-go function by Hamilton-Jacobi-Bellman)

Let the cost-to-go function \(J \in L^\infty(\mathbb{R}^{n_x} \times \mathcal{T}, \mathbb{R})\) with Bolza objective (15) be implicitly defined as

\[
J(x(t_f), t_f) = \Phi(x(t_f)),
\]

\[
-\frac{\partial J}{\partial t}(x(t), t) = \min_{\omega \in \Omega} L(x(t), \omega(t)) + \frac{\partial J}{\partial x}(x(t), t) \left( f_0(t, x(t)) + \sum_{i=1}^{n\omega} \omega_i(t) f_i(t, x(t)) \right).
\]
We recognize that $\frac{\partial J}{\partial t}$ can be interpreted as Lagrange multiplier or dual variables of the ODE constraints in (MIOCP). Therefore we write in short $\lambda(t)$ instead of $\frac{\partial J}{\partial t}$. With the groundwork above we are ready to deduce bounds:

**Corollary 3** (Approximation bounds via ($\lambda$)CIA)

Apply the notations from Theorem 1 and Definition 8 with state trajectories $y_{\lambda \text{CIA}}(\cdot)$ and $x(\cdot)$ driven by controls $O^{\lambda \text{CIA}}$ and $\alpha$. It follows for all $t \in \mathcal{T}$:

$$|J(y_{\lambda \text{CIA}}(t), t) - J(x(t), t)| \leq \left( \|y_0 - x_0\|_{\lambda(t)} + \delta^*_{\lambda \text{CIA}} \right) e^{\ell(t-t_0)} + \Theta \left( \|y_{\lambda \text{CIA}} - x\|^2 \right).$$

**Proof** We consider the difference of the cost-to-go functions by approximating with a partial first order Taylor expansion around $J(x(t), t)$. The approximation is done in respect of the trajectories $x, y_{\lambda \text{CIA}}$. Hence, for $t \in \mathcal{T}$

$$J(y_{\lambda \text{CIA}}(t), t) - J(x(t), t) = \frac{\partial J}{\partial x}(x(t), t) (y_{\lambda \text{CIA}}(t) - x(t)) + \Theta \left( \|y_{\lambda \text{CIA}} - x\|^2 \right).$$

Rewriting the terms with absolute values, using the notation for $\lambda$ and $|\cdot|_{\lambda(t)}$ yields

$$|J(y_{\lambda \text{CIA}}(t), t) - J(x(t), t)| \leq \|y_{\lambda \text{CIA}} - x\|_{\lambda(t)} + \Theta \left( \|y_{\lambda \text{CIA}} - x\|^2 \right)$$

$$\leq \ldots \text{ (as in proof of Theorem 1) }$$

$$\leq \|y_0 - x_0\|_{\lambda(t)} + L \int_{t_0}^t \|y_{\lambda \text{CIA}}(\tau) - x(\tau)\|_{\lambda(t)} d\tau$$

$$+ \left| \sum_{i=1}^{n_0} \int_{t_0}^t (\omega^i_{\lambda \text{CIA}}(\tau) - \alpha_i(\tau)) \cdot f_i(\tau, x(\tau)) \ d\tau \right|_{\lambda(t)}$$

$$+ \Theta \left( \|y_{\lambda \text{CIA}} - x\|^2 \right).$$

The third addend of the last inequality represents the objective ($\lambda$CIA1). Thus, we apply the adapted Gronwall Lemma 1 to $v = (y_{\lambda \text{CIA}} - x)$ and

$$z = \|y_0 - x_0\|_{\lambda(t)} + \left| \sum_{i=1}^{n_0} \int_{t_0}^t (\omega^i_{\lambda \text{CIA}}(\tau) - \alpha_i(\tau)) \cdot f_i(\tau, x(\tau)) \ d\tau \right|_{\lambda(t)}$$

so that the claim is proven.

**Remark 3** One could modify and extend Corollary 3.

1. For the sake of clearness in ($\lambda$)CIA, we added the Lagrange term to the objective. On the other hand, Mayer and Lagrange terms can be transformed into each other and therefore we have assumed a Mayer term as objective for the rest of the paper.
2. The upper bounds from Corollary 3 resemble in a sense those obtained from (SCIA), i.e., Corollary 2. It is possible to further approximate the upper bound to obtain similar bounds as for the (CIA) case.
3. ($\lambda$CIA) benefits from a first degree Taylor approximation. One might think about higher degree expansions for tighter bounds.
4.4 Backwards accumulating constraints

If we adapt the MIOCP instance with fixed final states \( x_f \in \mathbb{R}^n \) and with a Lagrangian objective type, we can also apply the altered setting to Theorem 1. The following corollary deals with this issue.

**Corollary 4** (Approximation bounds via backwards constraints)

Under the assumptions of Theorem 1, with \( x(\cdot) \) and \( y(\cdot) \) being the solutions not of the initial value problems, but of the problems \( 2b \) together with fixed final states \( x(t_f) = x_f \) and \( y(t_f) = y_f \). Assume it holds for all \( t \in T \), \( \delta \in \mathbb{R}^+ \)

\[
\left\| \int_t^{t_f} \alpha(\tau) - \omega(\tau) \, d\tau \right\| \leq \delta,
\]

(17a)

then it also holds for all \( t \in T \)

\[
\left\| y(t) - x(t) \right\| \leq \left( \left\| y_f - x_f \right\| + n \omega \left( B + C(t_f - t) \right) \delta \right) e^{L(t - t_f)}. \]

(17b)

**Proof** Note that by the Fundamental Theorem of Calculus

\[
x(t) = x_f - \int_t^{t_f} f_0(\tau, x(\tau)) + \sum_{i=1}^{n \omega} \alpha_i(\tau) f_i(\tau, x(\tau)) \, d\tau,
\]

holds and an analogous equality for \( y \). The proof of Theorem 1 can be applied to the altered setting. First, \( \left\| y_0 - x_0 \right\| \) is replaced by \( \left\| y_f - x_f \right\| \). Throughout the proof, we integrate over \([t, t_f]\) instead of \([0, t]\), but the approximation ideas still apply. Finally, a variant of the Gronwall Lemma is used that bounds the normed \( v(t) \) by \( \|v(\cdot)\|_{\infty}, e^{L(t - t_f)} \).

\[\square\]

**Remark 4** Relevance for MILP formulations:

1. Corollary 4 can be used to deduce bounds for the discretized (CIA), (SCIA), and \( (\lambda \text{CIA}) \) problems. For instance in case of (SCIA), the bounds are adapted according to Corollary 2.
2. With the assumption of \( \left\| x(t_f) - y(t_f) \right\| \) being small, the backwards approach is appropriate also for (MIOCP) with given initial values \( x_0, y_0 \) and variable final states.

4.5 Connection to decomposition algorithm and optimization problem

As a last step in this chapter, the previous results are related to (MIOCP) and the decomposition Algorithm 1.

**Remark 5** (Arbitrary close approximation of (OCP) solution)

We could extend Algorithm 1 with an outer loop which checks if \( \Phi_{\text{fin}} \) is sufficiently close to \( \Phi_{\text{rel}} \) and if not we would refine the grid \( G_{\omega} \). In [37], an arbitrary close approximation of the (OCP) solution has been deduced with this procedure and based on (CIA). With the assumption of \( \Phi(\cdot), c(\cdot) \) being continuous and that there exists a
feasible trajectory $\mathbf{x}$ for (OCP), it follows that for any $\varepsilon > 0$ there exists a grid $\mathcal{G}_\omega$, with grid size $\Delta t$ such that there is a feasible trajectory $\{y(t_j)\}_{t_j \in \mathcal{G}_\omega}$ with

$$|\Phi(x(t_f)) - \Phi(y(t_f))| \leq \varepsilon,$$

$$\|c(t_j, x(t_j)) - c(t_j, y(t_j))\| \leq \varepsilon.$$

The proof uses the Sum Up Rounding scheme that derives the binary control approximation with $\delta^{\star}_{\text{CIA}} \leq \text{Const}(n_\omega) \Delta t$ [34,26], where Const($n_\omega$) is a constant depending on $n_\omega$. In case we extend Algorithm 1 with the refinement procedure and if (CIA) or (SCIA) are chosen to be elements of $\mathcal{S}^{\text{CIA}}$, the same approximation result holds.

**Corollary 5** *(Solution accuracy of differential states of Algorithm 1)*

Let $\delta^{\star}_{\text{SCIA,max}}$, $\delta^{\star}_{\text{SCIA,1}}$ be the optimal objective values of (SCIAmax), respectively (SCIA1). Furthermore, let $\mathbf{x}$, $\mathbf{y}$ describe the trajectories obtained by applying the optimal $\mathbf{a}$, $\mathbf{w}$ from Algorithm 1 and $\mathcal{G}_\omega$ the applied grid. It follows for $t_i \in \mathcal{G}_\omega$

$$\|y(t_i) - x(t_i)\|_j \leq (\|y_0 - x_0\|_j + \delta^{\star}_{\text{SCIA,j}}) e^{L(t_i-t_0)}, \quad j \in \{\text{max, 1}\}. \quad (18)$$

**Proof** The claim is a direct result of Corollary 2. □

We have deliberately chosen the tightest bound from the previous corollaries, but could also use others. The received grid specific rounding error bounds aim primarily at approximating the differential states. But with ($\lambda$CIA) or Lipschitz continuity of the objective, there are also tools to discuss the rounding error of the objectives. The so far omitted recombination heuristics work in the area of objective approximation without supporting error bounds.

## 5 Recombination heuristics

We present several recombination heuristics, that recombine different binary controls $\mathbf{w}$ to new candidate solutions with potentially lower objective value. The general framework is open to apply different heuristics such as Genetic algorithms [15], that are not introduced in this article.

### 5.1 GreedyTime

Algorithm 2 establishes a routine to use the MILP solutions in a greedy approach, aiming at constructing solutions $\mathbf{w}$ with improved objective value $\Phi(\varphi(\mathbf{w}))$. Notice the abbreviations $[M] := \{1, \ldots, M\}$ and $\Phi_{m_1} = \Phi(\varphi(\mathbf{w}^{m_1}))$ in the algorithm.

**GreedyTime** iterates over all discretization intervals $j$ in their natural order (Line 1). We check in Line 2 on every interval if there are MILP pairs $(m_1, m_2)$, which differ in their binary solution vectors. For each of these pairs we recombine the $m_1$ solution with the binary vector solution from $m_2$ at interval $j$ to create a temporary solution $\tilde{\mathbf{w}}^{m_1}$ (Line 3). Then, we evaluate the objective of this new solution in Line 4 and overwrite the solution $\mathbf{w}^{m_1}$ with the recombeded solution $\tilde{\mathbf{w}}^{m_1}$ if it results in a better
Algorithm 2: GreedyTime heuristic for finding improved $w$ variables

**Input**: Control grid $\mathcal{G}_0$, binary variable vectors $w^{\text{milp}}$ as solutions of $\text{milp} \in S_{\text{CIA}}$, corresponding objectives $\Phi(\varphi(w^{\text{milp}}))$.

1. for $j \in \mathcal{G}_0$ do
   2. for $(m_1, m_2) \in S_{\text{CIA}} \times S_{\text{CIA}}, m_1 \neq m_2, w^{m_1} \neq w^{m_2}$ do
   3. Set $\tilde{w}_k^{m_1} = w_k^{m_1}, k \neq j, k \in \mathcal{G}_0$ and $\tilde{w}_j^{m_1} = w_j^{m_2}$.
   4. Evaluate (MIOCP) with fixed $\tilde{\omega}^{m_1} = \varphi(\tilde{w}^{m_1}) \rightarrow \tilde{\Phi}_{m_1}$.
   5. if $\tilde{\Phi}_{m_1} \leq \Phi_{m_1}$ then
      6. Set $w_j^{m_1} = \tilde{w}_j^{m_1}$ and $\Phi_{m_1} = \tilde{\Phi}_{m_1}$.
   end
   end
9. end
10. return: $\Phi(\varphi(w^{\text{rec}})) := \min_{\text{milp} \in S_{\text{CIA}}} \Phi(\varphi(w^{\text{milp}}))$.

Objective value (Lines 5–7). In the same way we proceed with the second solution $m_2$ when the (same) pair $(m_2, m_1)$ comes up in the inner loop.

Note that with a high number of calculated MILP, there might be large number of pairs with unequal solutions. Instead of swapping and testing each variation for every $w^{m_1}$, it is advisable to use just the $w^{m_2}$ solution for swapping with the so far lowest objective value. Fig. 1 illustrates an example recombination step for the pairs (CIA,SCIA) and (SCIA,CIA).

**Remark 6** (GreedyTime speed up)

1. The evaluation in Line 4 in iteration $j$ can be accelerated by reusing the evaluation for the same solution from the previous iteration $j-1$ for the interval $[t_0, t_j]$. 

---

**Fig. 1** Visualization of GreedyTime algorithm. Two candidate solutions, here from CIA and SCIA, are used to construct new candidates. An enumeration between 0 and 1 is performed at all times $t_j$ when the input vectors differ. Next, we fix the two candidate solutions $w$ and evaluate (MIOCP) for both vectors. The resulting objective function values are compared with its previous values and the binary $w_j$-values with the lower objective value are fixed in the candidate solutions. We repeat this procedure on the next grid point with unequal candidate solutions.
2. If a MILP solution \( m_1 \) differs from two MILPs \( m_2, m_3 \) with identical binary solutions vectors \( w_j^{m_2} = w_j^{m_3} \), it is sufficient to simulate its recombination with only one of the two.

**Remark 7 (GreedyTime modifications)**

1. The outer loop in Algorithm 2 can be also applied backward in time. We name the backward version \textit{GreedyTimeBackward}.
2. Instead of looping over all intervals, we may consider only singular arcs, i.e., the intervals where \( \varepsilon < a_{i,j} < 1 - \varepsilon \) holds, with a certain threshold \( \varepsilon > 0 \).
3. \textit{Greedy-cost-to-go}: Assume we have calculated the adjoints \( \lambda_{j,k}, k \in [n_x], t_j \in \mathcal{G}_\omega \) of the state equations of (OCP). Then, re-sort \( \mathcal{G}_\omega \) in descending order according to \( \sum_{k \in [n_x]} |\lambda_{j,k}|, t_j \in \mathcal{G}_\omega \). This results in a new ordered grid \( \mathcal{G}_\omega^{\lambda} \) to be applied to Algorithm 2.

5.2 Singular arc recombination

Algorithm 3 aims at recombining singular arcs of the different MILP solutions. Usually, when \( a \) is binary or almost binary for a certain grid point, \( w \) should attain these binary values as well - regardless of the MILP choice. Therefore we partition \( \mathcal{G}_\omega \) into singular and binary arcs.

**Definition 9 \((\mathcal{G}_{\text{arc}}, \mathcal{G}_{\text{bin}})\)** Let the union \( \mathcal{G}_{\text{arc}} \cup \mathcal{G}_{\text{bin}} \) be a partition of \( \mathcal{G}_\omega \) and both sets be of minimum cardinality. Elements \( \mathcal{arc} \in \mathcal{G}_{\text{arc}} \) consist of consecutive time points, i.e. \( \mathcal{arc} = \{t_j, t_{j+1}, \ldots, t_l\}, j, l \in [M] \), for which the relaxed control satisfies \( a_k \in [\varepsilon, 1-\varepsilon]^{n_\omega}, \varepsilon > 0, \forall t_k \in \mathcal{arc} \). Sets of consecutive time points where the relaxed control takes values smaller \( \varepsilon \) or larger than \( 1 - \varepsilon \) are summarized in \( \mathcal{G}_{\text{bin}} \). We write \( n_{\text{arc}} := |\mathcal{G}_{\text{arc}}| \) for the number of singular arcs and \( w_{\text{arc}} = \{w_k\}_{k \in \mathcal{arc}}, \mathcal{arc} \in \mathcal{G}_{\text{arc}} \) and accordingly \( w_{\text{bin}}, \mathcal{bin} \in \mathcal{G}_{\text{bin}} \).

In Algorithm 3 we set the temporary solution \( w^{\text{tmp}} \) on the binary arcs to the rounded relaxed solution for these binary arcs in Line 1. Then we test every possible variation (Line 2) of the different MILP solutions on the singular arcs to fill up the singular arcs of the temporary solution \( w^{\text{tmp}} \) (Line 3) and evaluate its objective value in Line 4. In case a recombination has a lower objective value than the so far best solution, it will be saved as so far best solution in Lines 5–8. Each variation \( \text{var} \in (S_{\text{CIA}})^{n_{\text{arc}}} \) consists of a specific MILP solution choice on each \( \mathcal{arc} \in \mathcal{G}_{\text{arc}} \), which is denoted by \( \text{var}(\mathcal{arc}) = \text{milp} \in S_{\text{CIA}} \).

**Remark 8** For computational effort, one has to take care of \( |S_{\text{CIA}}|^{n_{\text{arc}}} \). Therefore, we choose for our numerical tests a small subset of MILPs out of \( S_{\text{CIA}} \) and instances with few singular arcs. In case of more than 4 singular arcs, Algorithm 3 may be modified to be greedy, applying the idea of Algorithm 2 on arcs instead of single grid points. Furthermore, if two MILP solutions are identical on a singular arc, only one has to be considered for the variations. Our algorithmic implementation checks if there are singular arcs with no unequal binary vectors and excludes these arcs from recombination.
Algorithm 3: Singular arc block recombination heuristic for finding improved $w$ variables

Input: Control grid $G_ω$, optimal relaxed variable vectors $a^{milp}$, binary variable vectors $w^{milp} ∊ S_{CIA}$, corresponding objectives $Φ(ϕ(w^{milp}))$, sets $G_{arc}$, and $G_{bin}$.

1. Set $w_{bin}^{tmp} = \lfloor a_{bin} + ε \rfloor$, ∀ $bin ∈ G_{bin}$ and $Φ_{rec} = w$.
2. for $var ∈ (S_{CIA})_{arc}$ do
3. Set $w_{arc}^{tmp} = w_{var}^{arc}$, ∀ $arc ∈ G_{arc}$.
4. Evaluate (MIOCP) with fixed $ω_{tmp} := ϕ(w_{tmp}) → Φ_{tmp}$.
5. if $Φ_{tmp} < Φ_{rec}$ then
6. Set $Φ_{rec} = Φ_{tmp}$, $w_{rec} = w_{tmp}$.
7. end
8. end
9. return: $w_{rec}$ together with $Φ_{rec}$.

Fig. 2 Visualization of singular arc block recombination heuristic for two MILP solution vectors with three singular arcs. We generate every variation from singular arcs and candidate solutions and evaluate (MIOCP) for each of the found variations. The minimal objective value of all variations represents the heuristic output.

Remark 9 The singular arc recombination supports an objective value that is at least as good as the previous found via the MILPs. However, there is no framework yet to quantify these possible improvements in terms of new rounding errors of the objective.
6 Computational results

6.1 Software implementation

We implemented Algorithm 1 in AMPL [8] using the code `ampl_mintoc`, that is a modeling framework for handling optimal control problems. It features different discretization schemes of ODEs, though we used only a Radau collocation from [3]. The tool is advantageous for our purpose, since it includes automatic differentiation, interfaces to MILP solvers, and its problem formulation stays close to mathematics. Also AMPL provides the dual variables $\lambda$. Throughout the numerical study, we applied Gurobi 7.5.0 as MILP solver and IPOPT 3.12.4 as NLP solver to solve the discretized (OCP). We assumed that the choice of the MILP solver has little influence on solution quality and verified this by testing also with CPLEX 12.7. We tested our algorithms also with CasADi and received similar results as with `ampl_mintoc`. All results were obtained on a workstation with 4 Intel i5-4210U CPUs (1.7 GHz) and 7.7 GB RAM running Ubuntu 14.04 LTS.

6.2 Instances and results

We included the following MIOCPs from the benchmark collection site `mintoc.de` [33] in our numerical study: "F-8 aircraft (AMPL variant)", "Egerstedt standard problem", "Double Tank", "Double Tank multimode", "Lotka Volterra fishing problem", "Lotka Volterra multi-arcs problem", "Lotka Volterra multimode problem", "Van der Pol Oscillator (binary variant)", "D’Onofrio chemotherapy model", "Catalyst Mixing problem".

The problems involving path constraints might result in infeasible solutions after solving the binary approximation problem and solving the MIOCP with fixed binary controls. To this end, we relaxed the path constraints and applied a merit function that penalizes constraint violation as part of the objective with sufficiently high penalty factor. Also, in case of infeasibilities due to terminal state constraints, we dropped these constraints and included them as penalizing term in the Mayer objective. We chose a differential states discretization with $N$ intervals such that the objective value differs only to the 5th decimal place with respect to a finer discretization for constant $M$. Afterwards we varied $M$ with fixed $N$ in order to create several instances. For further details we refer to Appendix 8. We tested all in this article mentioned MILP formulations, but present only the best performing and most outstanding ones in terms of objective values. Our code includes also several variations of the GreedyTime recombination, where the heuristic is applied only on singular arcs. Again, since these heuristics hardly changed the results on average, we skip its presentation.

Figure 3 depicts a performance plot of different algorithms in terms of objective deviation to the relaxed solution in percentage (x-axis) and fraction of instances with less than $x\%$ objective deviation (y-axis). Note that, depending on the selected algorithm, up to 20% of instances had a deviation of more than 100%, which can be explained by highly penalized infeasible solutions of path-constrained problems. The SCIA approaches performed clearly better than CIA on the instances compris-
ing the highest and lowest deviation, whereas CIA was the better choice on other instances. The mean deviation of the scaled approaches (SCIAmax: 18.73%, SCIA1: 28.31%) appeared to be slightly lower than these of CIA (CIAmax: 29.20%, CIA1: 28.31%). However, (CIA) performed better, worse and as good as (SCIA) variants on approximately a third of the instances each. The numerical study revealed several instances, where (SCIAmax) or (SCIA1) run into a binary solution with active controls on some intervals with relaxed values close to zero. Under the assumption that the combinatorial approximation is done mainly on singular arcs, these cases might be called degenerated. We experienced underperforming objective values for SCIA in case of degenerated controls, which explains some of the underperforming instances. For detailed numerical results we refer to the Appendix 8. Throughout the instances, (λCIA1) appeared not to be competitive. MILPs with backward ordered constraints delivered solutions close to those of CIA, but since we solved only initial value problems, the performance curve of (CIAmaxB) appeared mostly to the right of its forward variant.

The depicted recombination heuristics are all based on the five illustrated MILPs together with (CIA1). They performed particularly well on path constrained instances with high objective deviation. Greedy-cost-to-go and GreedyTime outperform the MILP solutions the most, followed by GreedyTimeBackward and Singular arc block

Fig. 3 Performance profile comparing objective deviation from the relaxed solution in percentage and log-scale of several MILP and recombination heuristic solutions. The results are based on instances from the mintoc.de benchmark library. Using the best of several MILP solutions or recombination strategies can improve the CIA performance significantly.
recombination. Note that most instances include only one singular arc and in these cases ArcRecombination represents the minimum over all MILP solutions. Since the ArcRecombination graph is located clearly to the left of all MILP graphs, we conclude that taking only several MILP solutions results already in an enhancement with respect to the single CIA approach. The mean objective difference to the relaxed solution decreased compared to the MILPs to roughly 1% (Greedy-cost-to-go, Greedy-Time), 2% (GreedyTimeBackward) and 4.41% (ArcRecombination). The same tendency holds for the standard deviation of the objective values. Minimizing over all recombination heuristics (Minimum of all algorithms in Figure 3) is the output of Algorithm 1 with full $\bar{S}^{\text{CIA}}$ and $\bar{S}^{\text{REC}}$.

We calculated also the numbers of switches. There were no significant difference between the MILP and recombination solutions in this respect, but slightly lower mean values for $\bar{S}^{\text{CIA}}$ and $\lambda_{\text{CIA}}$ compared to CIA.

Figure 4 shows exemplarily the relationship between runtime and objective function values for the Lotka Volterra multimode problem with $N = 12000$ and varying $M$. We compare (CIAmax) values both with GreedyTime and the solutions obtained by the MINLP solver Bonmin 1.8.6. Elapsed real time from AMPL represents runtime in our computations, since CPU time appeared to be very similar for our Bonmin calculations and Gurobi on the other hand is known to be a multi-threaded solver. First of all, the illustration shows that the objective spread to the relaxed solution vanished with increasing $M$ - regardless of the selected approach. Second, CIA was for some instances already quite close to Bonmin ($M = 25, 50$) in terms of objective quality, so that GreedyTime cannot improve much. For other discretization with a considerable gap between CIA and Bonmin solution, GreedyTime could close most of this gap while being two orders of magnitude faster than Bonmin.

The average runtimes of scaled MILPs and recombination heuristics increased significantly compared to (CIA), but stayed mostly in an acceptable range of a few minutes (see Appendix 8).

6.3 Discussion

Our results show that scaled MILPs improved the solution of (CIA) solution on average, but without individual guarantee. The theoretical error bounds on (SCIA) in Section 4 do not contradict this fact, since they are not sharp and only provide information about approximation in state space, not necessarily about objective functional space. ($\bar{\lambda}^{\text{CIA}}$) attempts to approximate the objective linearly, but fails to provide solutions of high quality, so one might want to think about higher order Taylor approximation. Nevertheless, even less performing MILP solutions are candidates for recombination strategies, which in turn proved to be reliable as candidates differ.

ArcRecombination is relatively inexpensive and offers a solution that is at least as good as the best MILP. The algorithm is most beneficial in case of several singular arcs, in contrast to most applied problem instances where there is only one arc. The greedy algorithm variants are quite expensive (run times of up to 15 minutes), yet providing solutions with objective function values very close to those of the relaxed problem. Recombination heuristics are especially suitable for problems with path
constraints, since infeasible solutions can be reduced or even resolved by the feedback with the original MIOCP.

Our study is limited by model assumptions as we excluded combinatorial constraints regarding the binary controls and control dependent vanishing constraints. Although relevant for many applications, they are challenging to handle and therefore not included in this paper. Another limitation concerns the runtime of the algorithms presented in this study. Gurobi needed on average a few seconds to solve (CIA), whereas solving (SCIA) was computationally intensive with a runtime of up to 30 minutes, which was our limit to stop the solver with the best solution found. The greedy heuristics and the ArcRecombination are to be used cautiously, since an input of many MILPs leads to a high number of recombinations that have to be evaluated.

Note that we are interested in different MILP solutions as a basis for post-processing. Therefore, it is justified to set up a time limit for the MILP solver and work with intermediate solutions. Another way to significantly reduce computation time is to apply Branch and Bound [23] or the Sum Up Rounding scheme [31]. Both methods solve (CIA) under mild assumptions to optimality and might be adapted to the scaled MILP case as part of a future study. We may accelerate the post-processing routines as described in Remark 6. Furthermore, AMPL as modeling framework is a high level modeling language. To this end, low-level languages such as C++ could be used to further reduce the run time of the recombination heuristics. Finally, run times

---

**Fig. 4** Log-plot of runtime and objective deviation from relaxed solution for the Lotka-Volterra multimode problem, $N = 12000$. Numbers indicate the corresponding number of control grid points $M$ and by $\delta(\mathcal{S}^{CIA})$ we mean the minimal objective deviation over all MILP solutions. One observes the convergence of all approaches towards the lower bound provided by the relaxed solution, and a closure of the gap between CIA solution and Bonmin solution for a fixed discretization. GreedyTime is roughly two orders of magnitude slower than CIA, but faster than Bonmin.
of days or even weeks when it comes to MINLP solver cast a positive light on the proposed decomposition algorithm including recombination.

The main added value of this study is a decomposition algorithm that works much faster than Bonmin, but still offers qualitatively similar solutions and performs on average better than the existing (CIA) approach. The framework is open to extensions, both on the MILP and on the post-processing level. One possible recombination extension could be an ArcRecombination version that bisects singular arcs in case of only one or two available singular arcs. Altogether, the beneficial feature of the heuristics is that they automatically find solutions at least as good as the best MILP solution found.

7 Summary and conclusion

We extended the decomposition approach based on combinatorial integral approximation [36], using multiple MILP formulations and recombination heuristics in an outer loop. At the price of additional MILP solutions and MIOCP evaluations, we obtain a potential improvement of the objective function value for every fixed control discretization grid. The integer gap depends linearly on the maximum interval in this control discretization grid.

A numerical study with benchmark problems shows no clear winner among the different MILP formulations with respect to objective function value. They differ in many cases, however, allowing to simply choose the best one for that particular instance. Additionally, they provide candidate solution vectors that can be recombined. Recombinations, especially exploiting the particular structure of singular or path-constrained arcs, resulted in significant improvements over single formulations.

Additional work is necessary to incorporate vanishing and combinatorial constraints and to derive tailored numerical algorithms that generalize Sum Up Rounding and/or tailored Branching algorithms to the various MILP formulations.

8 Appendix: Detailed Numerical Results

8.1 Problem Discretization Details

For generating Fig. 3, we applied Algorithm 1 on the following discretized problems:

"F-8 aircraft (AMPL variant)"
\( N = 6000, M \in \{30, 40, 50, 60, 100, 120, 150, 200, 240, 300, 400, 500\} \),

"Egerstedt standard problem"
\( N = 6000, M \in \{20, 30, 40, 60, 100, 120, 150, 200, 240, 300\} \),

"Double Tank"
\( N = 18000, M \in \{25, 50, 100, 180, 250, 300, 360, 720\} \),
"Double Tank multimode":
\( N = 12000, \ M \in \{20, 25, 50, 100, 200, 250, 300, 400, 600\} \),

"Lotka Volterra fishing problem":
\( N = 12000, \ M \in \{20, 30, 40, 60, 100, 120, 200, 300, 400, 600\} \),

"Lotka Volterra multi-arcs problem":
\( N = 18000, \ M \in \{25, 50, 100, 150, 200, 250, 300, 400, 600\} \),

"Lotka Volterra multimode problem":
\( N = 12000, \ M \in \{25, 50, 100, 150, 200, 250, 300, 400, 800\} \),

"Van der Pol Oscillator (binary variant)":
\( N = 6000, \ M \in \{20, 30, 40, 50, 60, 100, 120, 150, 200, 300\} \),

"D’Onofrio chemotherapy model":
Scenario 1, 2, and 3 with \( N = 6000, \ M \in \{20, 30, 40, 50, 60, 100, 120, 150, 200, 300\} \),
Only \( M \in \{20, 30, 60\} \) for scenario 1, \( M = 100 \) for scenario 2, and \( M \in \{40, 100\} \) for scenario 3 resulted in feasible relaxed solutions and were included.

"Catalyst Mixing problem":
\( N = 3000, \ M \in \{10, 15, 20, 30, 50, 60, 75, 100, 120, 150\} \).

8.2 Average Performance Indicators and Individual Problem Results

<table>
<thead>
<tr>
<th>Approach</th>
<th>Obj. dev [%]</th>
<th>Switches [#]</th>
<th>Runtime [s]</th>
<th>( \sigma ) (Obj. dev)</th>
<th>( \sigma ) (Switches)</th>
<th>( \sigma ) (Runtime)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIAmax</td>
<td>29.20</td>
<td>33.45</td>
<td>9.71</td>
<td>102.91</td>
<td>34.36</td>
<td>54.95</td>
</tr>
<tr>
<td>CIA1</td>
<td>28.31</td>
<td>33.23</td>
<td>80.15</td>
<td>100.32</td>
<td>33.54</td>
<td>313.89</td>
</tr>
<tr>
<td>SCIAmax</td>
<td>18.73</td>
<td>31.58</td>
<td>67.36</td>
<td>47.55</td>
<td>30.84</td>
<td>315.09</td>
</tr>
<tr>
<td>SCIA1</td>
<td>28.31</td>
<td>33.23</td>
<td>78.49</td>
<td>100.32</td>
<td>33.54</td>
<td>309.82</td>
</tr>
<tr>
<td>( \lambda ) CIA1</td>
<td>53.61</td>
<td>29.84</td>
<td>101.54</td>
<td>178.44</td>
<td>49.91</td>
<td>402.48</td>
</tr>
<tr>
<td>CIAmaxB</td>
<td>36.83</td>
<td>33.78</td>
<td>43.55</td>
<td>127.80</td>
<td>34.62</td>
<td>282.78</td>
</tr>
<tr>
<td>GreedyTime</td>
<td>1.19</td>
<td>32.26</td>
<td>83.71</td>
<td>2.82</td>
<td>32.26</td>
<td>118.71</td>
</tr>
<tr>
<td>GreedyTimeB</td>
<td>2.02</td>
<td>31.38</td>
<td>80.69</td>
<td>5.14</td>
<td>29.21</td>
<td>115.28</td>
</tr>
<tr>
<td>Greedy-Cost-to-go</td>
<td>1.16</td>
<td>32.33</td>
<td>97.21</td>
<td>2.72</td>
<td>31.72</td>
<td>162.49</td>
</tr>
<tr>
<td>ArcRecombination</td>
<td>4.41</td>
<td>33.32</td>
<td>11.52</td>
<td>10.99</td>
<td>35.04</td>
<td>35.91</td>
</tr>
</tbody>
</table>
Table 2 Results for the Lotka Volterra multimode problem with $N = 12000$ and varying $M$. The tables list objective values, differences to relaxed objective, number of switches, and runtime in seconds.

<table>
<thead>
<tr>
<th>M</th>
<th>(CIAmax)</th>
<th>(CIA1)</th>
<th>GreedyTime</th>
<th>GreedyTimeBackward</th>
<th>ArcRecombination</th>
<th>Greedy-cost-to-go</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>1.84519</td>
<td>0.00920032</td>
<td>6</td>
<td>0.419997</td>
<td>1.84519</td>
<td>0.00920032</td>
</tr>
<tr>
<td>50</td>
<td>1.83353</td>
<td>0.00189968</td>
<td>9</td>
<td>0.498163</td>
<td>1.83353</td>
<td>0.00189968</td>
</tr>
<tr>
<td>100</td>
<td>1.83458</td>
<td>0.00470921</td>
<td>15</td>
<td>0.564993</td>
<td>1.83458</td>
<td>0.00470921</td>
</tr>
<tr>
<td>150</td>
<td>1.83049</td>
<td>0.00129738</td>
<td>20</td>
<td>0.979946</td>
<td>1.83058</td>
<td>0.00138375</td>
</tr>
<tr>
<td>200</td>
<td>1.8294</td>
<td>0.00041246</td>
<td>23</td>
<td>0.983907</td>
<td>1.8294</td>
<td>0.00041246</td>
</tr>
<tr>
<td>250</td>
<td>1.82887</td>
<td>8.52473e-05</td>
<td>30</td>
<td>2.01582</td>
<td>1.82887</td>
<td>8.52473e-05</td>
</tr>
<tr>
<td>300</td>
<td>1.82884</td>
<td>2.1597e-05</td>
<td>33</td>
<td>1.87382</td>
<td>1.82884</td>
<td>2.1597e-05</td>
</tr>
<tr>
<td>400</td>
<td>1.82875</td>
<td>2.58672e-05</td>
<td>87</td>
<td>66.8739</td>
<td>1.82875</td>
<td>2.58672e-05</td>
</tr>
<tr>
<td></td>
<td>(SCIAmax)</td>
<td>(SCIA1)</td>
<td>GreedyTime</td>
<td>GreedyTimeBackward</td>
<td>ArcRecombination</td>
<td>Greedy-cost-to-go</td>
</tr>
<tr>
<td>25</td>
<td>1.84519</td>
<td>0.00920032</td>
<td>6</td>
<td>0.313487</td>
<td>1.84519</td>
<td>0.00920032</td>
</tr>
<tr>
<td>50</td>
<td>1.83353</td>
<td>0.00235793</td>
<td>8</td>
<td>0.493533</td>
<td>1.83353</td>
<td>0.00189968</td>
</tr>
<tr>
<td>100</td>
<td>1.8834</td>
<td>0.0542079</td>
<td>20</td>
<td>2.84568</td>
<td>1.83458</td>
<td>0.00470921</td>
</tr>
<tr>
<td>150</td>
<td>1.86972</td>
<td>0.0407389</td>
<td>25</td>
<td>7.90383</td>
<td>1.8294</td>
<td>0.00041244</td>
</tr>
<tr>
<td>200</td>
<td>1.82887</td>
<td>8.52473e-05</td>
<td>30</td>
<td>11.6632</td>
<td>1.82887</td>
<td>8.52473e-05</td>
</tr>
<tr>
<td>250</td>
<td>1.82884</td>
<td>4.80446e-05</td>
<td>32</td>
<td>8.75161</td>
<td>1.82884</td>
<td>4.80446e-05</td>
</tr>
<tr>
<td>300</td>
<td>1.82887</td>
<td>1.94567e-05</td>
<td>47</td>
<td>3.42292</td>
<td>1.82887</td>
<td>1.94567e-05</td>
</tr>
<tr>
<td>400</td>
<td>1.82877</td>
<td>1.94567e-05</td>
<td>47</td>
<td>3.42292</td>
<td>1.82877</td>
<td>1.94567e-05</td>
</tr>
<tr>
<td>800</td>
<td>1.82875</td>
<td>2.58672e-05</td>
<td>87</td>
<td>66.8739</td>
<td>1.82875</td>
<td>2.58672e-05</td>
</tr>
</tbody>
</table>
References