EMPIRICAL COMPLEXITY ANALYSIS OF A MILP-APPROACH FOR OPTIMIZATION OF HYBRID SYSTEMS

Jochen Till, Sebastian Engell, Sebastian Panek, and Olaf Stursberg

Process Control Lab (CT-AST), University of Dortmund, 44221 Dortmund, Germany. E-mail: {j.till|s.engell|s.panek|o.stursberg}@ct.uni-dortmund.de

Abstract: Many techniques for designing controllers for hybrid systems suffer inherently from the complexity of computation, such that the applicability is limited to relatively small problems. It is often not obvious, however, which particular part of a problem formulation has a dominant impact on the increase of complexity with the problem size. This paper describes a thorough empirical investigation of the sources of complexity for an approach to optimal control of hybrid systems. This approach transforms the control task into a mixed-integer programming problem that is solved by Branch&Bound techniques. The influence of various parameters on the computational costs are investigated for a scalable technical example.

Keywords: Complexity, Empirical Analysis, Hybrid Systems, Mixed Integer Programming, Optimal Control.

1. INTRODUCTION

In comparison to designing controllers for purely discrete or purely continuous systems, the design of controllers for hybrid systems is a particularly challenging task: even in phases of the evolution of a hybrid system in which the structure of the dynamic equations remains constant, the evolution can be governed by continuous and discrete inputs. Design techniques have to account for the interaction of both degrees of freedom, with possibly opposing effects. If, in addition, the model structure changes discretely at the occurrence of specific internal or externally triggered events, the design has to consider further discontinuities. Nevertheless, various algorithmic approaches to controller design for different types of hybrid systems have emerged in recent years, notably those for algorithmic synthesis and verification as can be found, e.g., in the proceedings of the HSCC-workshops (Tomlin and Greenstreet, 2002). A common property of at least those techniques that consider nonlinear dynamics in the continuous part of hybrid systems is that the computational costs increase drastically with the problem size (see, e.g., (Silva et al., 2001)).

A first step in making existing techniques applicable to real-world problems is to investigate which degrees of freedom or which particular part of a problem formulation are dominating sources of complexity. The intention of this contribution is to identify those effects that prohibit the application to large systems. The investigation is carried out exemplarily for an approach to compute optimal control strategies for hybrid systems with nonlinear continuous dynamics and continuous as well as discrete inputs. The method is based on a transformation into a mixed-integer linear programming (MILP) problem that is solved
by a Branch&Bound algorithm (Stursberg and Panek, 2002; Stursberg et al., 2002). After sketching the approach in Sec. 2, we employ two different methods to assess the complexity, as described in (Nemhauser et al., 1994): In Sec. 3, a worst-case analysis provides guaranteed lower and upper bounds for the complexity as functions of key parameters of the problem formulation. Since these bounds are often overly conservative, additional information is obtained from an empirical analysis (Sec. 4). In this step, the computation time is measured for different variations of the problem formulation, using a scalable example.

Of course, the results presented are specific for the considered approach, but we nevertheless think that they point to typical problems arising in algorithms for hybrid systems and are thus helpful in assessing the complexity of other approaches as well.

2. AN OPTIMAL CONTROL PROBLEM FOR HYBRID SYSTEMS

We consider the task of determining a control strategy to drive a given hybrid system from an initial state $x_0$ to a target region $R_{\text{tar}}$ with minimal cost. Such problems typically arise in start-up procedures of processing systems where the main goal is to reach a new desired operating region as quickly as possible while state and input constraints have to be met.

The solution approach described in (Stursberg and Panek, 2002; Stursberg et al., 2002) consists of three steps: The starting point is to model the given system as a hybrid automaton with possibly nonlinear continuous-time dynamics and continuous and discrete control inputs. The state space $X \subset \mathbb{R}^n$ of dimension $n_x$ is partitioned by a set of $n_E$ hyperplanes such that $n_R$ polyhedral regions $R_r$ result: $\bigcup_r R_r = X$. For each region $R_r$ the dynamics is defined by a function $\dot{x} = f_r(x, u, v)$, where $x$ is the current state, and $u$ and $v$ are continuous and discrete inputs. The number of all possible discrete input values is denoted by $n_v$. The dynamics switches autonomously from $f_r$ to $f_{r'}$ if the state trajectory $x(t)$ crosses the hyperplane that separates the regions $R_r$ and $R_{r'}$.

The second step comprises the following: (1) Time is discretized by transforming the ODEs from step one into a set of difference equations defined for a set $T$ of $n_T$ discrete points of time. (2) All functions $f_r$ are replaced by linear approximations for appropriate linearization points of $x^L_r$ (in each region $R_r$) and $u^L_r$. In order to improve this approximation, it can be necessary to introduce additional hyperplanes that lead to a finer partition of the state space, and hence to a larger number $n_R \geq \tilde{n}_R$ of regions. The result is a set of linear discrete time functions $x_{r+1} = f_{r,v}(x_k, u_k) = A_{r,v}x_k + B_{r,v}u_k + L_{r,v}$ for each region $R_r$ and each discrete input value $v$. The time index is $k$.

In the third step, the optimization problem is formulated using the dynamic model as constraints: The linear dynamics define restrictions on the continuous variables $x_k$ and $u_k$, i.e., they relate the states $x_k, x_{k+1}$ at successive points of time. Since only one linearized dynamics $f_{r,v}(x_k, u_k)$ can be valid for each $t_k \in T$, all other dynamic equations must be disabled. For this purpose, a set of binary variables $b_{r,r',v}$ is defined, each of which enables or disables the corresponding function $f_{r,v}(x_k, u_k)$ at $k$. The choice of the dynamics for a specific region $R_r$ and discrete input $v_k$ is formulated by:

$$x_{k+1} = \sum_{r,v} b_{k,r,v} \cdot (A_{r,v}x_k + B_{r,v}u_k + L_{r,v}).$$

A transition between different dynamics hence corresponds to a change of values for exactly two binary variables.

Since the objective is to formulate a linear optimization problem, the products of variables are linearized by introducing continuous auxiliary variables $x_{k,r,v} := b_{k,r,v} \cdot x_k, u_{k,r,v} := b_{k,r,v} \cdot u_k$ and special sets of inequalities. This construction was proposed in (Stursberg and Panek, 2002) as disjunctive formulation. It is appropriate to define the relationship between continuous and binary variables without requiring large numbers of binary auxiliary variables. The formulation of the dynamics results in:

$$x_{k+1} = \sum_{r,v} A_{r,v}x_{k,r,v} + B_{r,v}u_{k,r,v} + b_{k,r,v}L_{r,v},$$

and it must hold that $\sum_{r,v} b_{k,r,v} = 1$ for all $k \in T$.

The set of constraints additionally contains inequalities that specify the regions $R_r$ and the check whether $x_k \in R_r$. For regions specified by $C, x \leq d, \text{the latter is written as } b_{k,r}(C, x - d) \leq 0, \sum_{r,v} b_{k,r} = 1$ with additional binary variables $b_{k,r} := \sum_{r,v} b_{k,r,v}$. Also here, the product of variables can be replaced by the disjunctive formulation.

As described in (Stursberg et al., 2002), different possibilities exist to specify appropriate objective functions in the given context. We consider here a version that formulates the distance to a given rectangular target region $R_{\text{tar}} \subset X$ in each step $k$: $J = \sum_k ||x_k - z||_1, z \in R_{\text{tar}}$. Since used in a minimization $\min_{u,v} J$, those trajectories of continuous $(u_k)$ and discrete inputs $(v_k)$ are selected, which drive the system as close as possible to $R_{\text{tar}}$ in each step. The objective function together with the listed constraints represents a mixed integer linear programming problem (MILP) that can be solved by standard techniques such as Branch&Bound algorithms.
3. THEORETICAL COMPLEXITY ANALYSIS

For the optimal control problem stated in Sec. 2, the objective of the complexity studies is to investigate the computational costs as a function of the problem size. To determine the latter, the following three measures are employed: \( M_1 \) – the size of the original problem measured by the values of the characteristic parameters \( n_x, n_R, n_v, \) and \( n_T \); \( M_2 \) – the size of the programming problem specified by the overall number of variables \( n \), the number of binary variables \( d \) (contained in \( n \)), and the overall number of constraints \( m \); \( M_3 \) – the computational cost to solve the MILP measured as CPU-time. This section describes the dependencies between the three measures prior to empirical studies.

3.1 Relation between \( M_1 \) and \( M_2 \)

The number of variables \( n \) and the number of equations \( n \) grow linearly with \( n_x, n_R, \) and \( n_T \). While \( d \) is not affected by \( n_x \), it depends according to \( \lceil \log_2(n_R) \rceil \) and \( \lceil \log_2(n_v) \rceil \) on the number of regions and discrete input combinations due to the binary encoding of the discrete variables \( n_R \) and \( n_v \). The number of regions itself is bounded by \( (n_E+1) \leq n_R \leq (2^{n_E}) \). In addition, all parameters related to the measure \( M_2 \) grow linearly with the considered number \( n_T \) of time points. The complexity of \( M_2 \) depending on \( M_1 \) is \( O(n_T n_x n_v n_R) \) for \( m \) and \( O(n_T \log_2 (n_v + n_R)) \) for \( d \).

3.2 Relation between \( M_2 \) and \( M_3 \)

This section aims at assessing the relation between \( (n, d, m) \) and the computation time by reviewing published literature on the complexity of MILP problems. By Branch&Bound algorithms, MILPs are decomposed into an integer program (IP) and a series of subordinated linear programs (LP).

The size of an LP is defined by its \( n \) variables and \( m \) constraints. A standard solver for LPs is the Simplex Algorithm, which searches along the vertices of the feasible region in direction of a decreasing value of the objective function. In the worst case, all vertices have to be investigated what leads to \( \psi = \binom{m}{3} \) iterations. The relation between the problem size of an LP and the average complexity of the solution is still an open issue, but it has been found experimentally that the effort is often considerably smaller than in the worst-case: (Edgar and Himmelblau, 1988) mention that the number of iterations usually depends much more on \( m \) than on \( n \), and is often between \( m \) and \( 3 \cdot m \). This range is confirmed in (Nemhauser et al., 1994) for problems in which \( n \) is a small multiple of \( m \), but for large-scale problems it is reported that a more realistic estimation is \( O(m) \) for the number of iterations and \( O(m \cdot n) \) for the number of arithmetic operations in each iteration. In (Hocks, 1995) it is stated that the complexity of the simplex algorithm grows from linear to polynominal with increasing problem size (Table 1), whereas barrier methods promise lower complexity for large scale problems.

<table>
<thead>
<tr>
<th>Scale</th>
<th>Dimension</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>( m &lt; 100 )</td>
<td>( O(m) )</td>
</tr>
<tr>
<td>medium</td>
<td>( 100 \leq m &lt; 10000 )</td>
<td>( O(m^2) )</td>
</tr>
<tr>
<td>large</td>
<td>( m \geq 10000 )</td>
<td>( O(m^3 \ldots m^5) )</td>
</tr>
<tr>
<td>worst case</td>
<td></td>
<td>( O(2^n) )</td>
</tr>
</tbody>
</table>

The classical approach for solving an IP (with \( d \) binary variables) is the tree search by a Branch&Bound algorithm with linear programming relaxation, which generally has an exponential complexity (NP-hard) (Floudas and Pardalos, 2001). In the best case, the complexity of the Branch&Bound algorithm is linear in the number of binary variables \( d \), and in the worst case the full tree has to be searched. Heuristics may be used to generate a first feasible solution and upper bounds, which reduce the size of the relaxed problem and also the tree size. An exact relation between the number of binary variables and the average complexity of practical problems is still unknown, but the average complexity varies from \( O(d) \) over \( O(d^2) \) to \( O(2^d) \) (Zhang, 1999).

4. EMPIRICAL ANALYSIS

Since especially the relation between \( M_2 \) and \( M_3 \) cannot be established precisely from theoretical considerations, we now empirically analyze the performance of the approach. We use a scalable multi-tank system to investigate directly the relation between \( M_1 \) and \( M_3 \). Scaling means in this context that one of the parameters determining \( M_1 \) is varied (while the others are kept constant) and the change of \( M_3 \) is observed.

The scalable system under consideration is shown in Fig. 1: It consists of a series of tanks connected by pipes. The feed into the first tank is located above this tank, while all other feed tubes are located at the medium height \( (H) \) of the subsequent tank. The feed and the outflow pipes are equipped with valves to adjust the flow rates. Autonomous changes of the dynamics occur, if the liquid level \( h \) in a tank exceeds the height of the feed tube. A corresponding hyperplane and two distinct dynamics account for this in the hybrid model. The fact, that a reversed flow from a tank into a preceding one is assumed to be not possible,
is modeled by a 'forbidden' region in the state space (see (Stursberg and Engell, 2002) for details on specifying forbidden regions).

The system as shown in Fig. 1 acts as a reference model, i.e., it is the basis for investigating variations of one of the parameters \( n_x, n_R, n_v, n_T \). The reference model is defined by three tanks \((n_x = 3)\) with only continuous inputs, thus only one discrete input combination is considered \((n_v = 1)\). The continuous valves \( V \) represent bounded continuous inputs. Modelling of the system leads to \( n_E = 4 \) separating hyperplanes which define \( n_R = 7 \) regions. The optimization goal is to minimize the objective function as specified in Sec. 2 for the transition from the initial state \( x_0 = (0.1, 0.1, 0.1)^T \) to the target region \( R_{\text{tar}} = [0.7, 0.8] \times [0.7, 0.8] \times [0.7, 0.8] \) within \( n_T = 10 \) time steps.

### 4.1 Analysis of the Computation Time

The MILPs are solved by using the package GAMS/CPLEX 7.5 which is based on a Branch&Bound algorithm with cuts and heuristics, and a simplex algorithm to solve the LP subproblems. Upper bounds for the number of iterations and the maximum resource usage are set to \( 10^6 \), and \( 10^9 s (= 27.7h) \) respectively. The relative gap to optimality is set to \( \epsilon \leq 0.01 \). All computations were performed on a Workstation Sun Ultra-Sparc with 300 MHz. In all subsequent figures, we use a scaled CPU-time \( \tau_c = t/t_{\text{ref}} \) where \( t_{\text{ref}} = 19.74s \) is the computing time for the reference system.

#### 4.1.1. Time steps:  
To observe the effect of altering the number of time steps, the time horizon is kept constant as \( t_{\text{ref}} = 50s \) while \( n_T \) is varied between 5 and 11. As shown in Figure 2(a), the CPU-time grows exponentially.

#### 4.1.2. Cont. State Space: 
The dimension \( n_x \) of \( \mathbf{X} \) is increased by adding additional tanks, with

Fig. 2. Analysis of computing time: scaled time \( \tau_c \) vs. problem size \( M_1 \)

feeding pipes located at the top (as for the first tank). The results for \( n_x \) between 3 and 8 are shown in Fig. 2(b).

#### 4.1.3. Regions:  
In order to study the impact of \( n_R \), hyperplanes are successively removed from the model (by setting the height of the inflow pipes to a maximum value of the level), or added respectively. For example, by adding one additional separating hyperplane to the first tank of the references model (i.e., \( n_E = 5 \)), the number of regions is increased to \( n_R = 5 \). The effect on the CPU-time is shown in Fig. 2(c).

#### 4.1.4. Discrete Input Combinations:  
Finally, the effects of a changed number of input combinations is investigated by allowing different settings for the two valves \( V_1 \) and \( V_2 \). Figure 2(d) summarizes the results for \( n_v \) between 1 and 16.

### 4.2 Summary of the Experimental Results

As shown in Figures 2(a)-2(d), the following dependencies of \( M_3 \) from \( M_1 \) were observed: \( O(2^{2\tau_c}) \), \( O(n_x^2) \), \( O(n_R^2) \), and \( O(n_v^2) \). The most critical effect is the exponential growth by an increase of \( n_T \). If \( n_x, n_R, \) and \( n_v \) are increased (independently), the complexity grows only polynomially.

From about one hundred optimization studies, it was found that problems with 5 to 10 continuous state variables and a relatively small number of time steps \( n_T \approx 5 \) are solvable for up to 8 discrete input combinations and less than 10 regions within the given resource bounds.
4.3 Analysis of the Algorithm

To get further insight into the dependencies, the complexity of the solution algorithm is studied by observing the number of nodes $N_{IP}$ explored by the Branch&Bound algorithm, and the average time required to solve an LP. The time required for one iteration during the LP solution varies due to the number of necessary algebraic operations. Therefore, the computing time of the LP solution is computed by the average number of iterations $I_{LP}$ per LP times the average CPU-time per iteration $T_{IT}$.

This analysis includes the configurations of Sec. 4.1 (series A) and a larger set of about 70 experiments with changes of more than one system parameter at a time (series B). We assume that the LP model size is characterized by $m$ and the IP size by $d$. In the following figures, the parameters $N_{IP}$, $I_{LP}$, and $T_{IT}$ are plotted against $m$ or $d$, respectively.

The number of iterations per LP $I_{LP}$ is almost linear in $m$ for series A (Fig. 3(a)) with a few exceptions. For series B, Fig. 4(a) shows that especially for more than 5000 equations the spread of points is rather large.

The computation times per LP iteration $T_{IT}$ of series A are shown in Fig. 3(b) and of series B in Fig. 4(b). In both figures $T_{IT}$ appears linear in $m$.

It is more difficult to estimate the dependency between $N_{IP}$ and $d$: Series A in (Fig. 3(c)) shows similar results as series B (Fig. 4(c)). Both figures do not show a linear or quadratic relation clearly.

4.4 Interpretation of the Results

The aim of the empirical analysis is to define the gap between the worst-case theoretical complexity and the estimated complexity of practical problems. The expected computing time of a MILP instance can be related to $m$ and $d$ by: $T_{MILP} = K_I m K_T m K_N d$. The parameters $K$ are obtained from the empirical series by: $K_I = I_{LP} / m$, $K_T = T_{IT} / m$, and $K_N = N_{IP} / d$. Now the empirical upper and lower bound of the total computing time can be estimated by taking the minimum and the maximum value of each $K$ (Tab. 2) and relate $m$ and $d$ to the respective values of the MILP instance. Figure 5 shows the computing time of the problems of series B and the empirical upper and lower bounds. We assume a constant value of the computing time per iteration $T_{IT}$ for that. Compared to the empirical results, the theoretical worst case computing time can be more than $10^{40}$ times higher.

If we compare the results of Table 2 with the considerations of Section 3, we can conclude that
Table 2. Empirical results of $K$

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>min</th>
<th>mean</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_T(B)$</td>
<td>0.014</td>
<td>0.066</td>
<td>0.25</td>
</tr>
<tr>
<td>$K_T(B)$ (ms)</td>
<td>0.37 · 10^{-6}</td>
<td>0.89 · 10^{-6}</td>
<td>1.71 · 10^{-6}</td>
</tr>
<tr>
<td>$K_N(B)$</td>
<td>0.14</td>
<td>3.42</td>
<td>11.93</td>
</tr>
<tr>
<td>$K_TK_TK_N$ (ms)</td>
<td>7.25 · 10^{-10}</td>
<td>2.01 · 10^{-7}</td>
<td>5.10 · 10^{-6}</td>
</tr>
</tbody>
</table>

Fig. 5. Worst case and empirical bounds for the series of MILP problems was solved very efficiently with respect to their problem size.

To obtain a more general result, we roughly estimate the complexity from Figures 3(a) - 4(c). We find $T_{LP}$ to be between $O(m)$ and $O(m^2)$. For $T_{IT}$ it is $O(m)$ and for $N_{LP}$ we assess a range between $O(d)$ and $O(d^2)$. As a result we can state the computational complexity $M_4$ of practical problems depending on $M_2$ to be $O(m^2d)$ to $O(m^3d^2)$. If we combine this with the results of Section 3.1, we obtain the relation between $M_1$ and $M_3$ to be bounded by $O((n_Tn_xn_vn_R)^2n_T\log_2(n_v+n_R))$ and $O((n_Tn_xn_vn_R)^3(n_T\log_2(n_v+n_R))^2)$.

5. CONCLUSIONS

The investigation shows (not surprisingly) that the time horizon ($n_T$) has the dominant influence on the solution performance: The number of equations and variables grows linearly with $n_T$ what leads to an exponential growth of the number of degrees of freedom. Two modifications to weaken this effect have been introduced: An obvious idea ist the use of a MPC-strategy that solves the optimization problem on short and moving horizons (Stursberg et al., 2002). A further reduction is obtained by using variable time steps what allows to cover larger periods of time with constant inputs (Stursberg and Engell, 2002).

But even with these extensions, large scale problems remain unsolvable by the considered approach. This calls for reducing the size of the problem formulation for each point of time. Considering the dependency of the complexity on $n_x$, $n_R$, and $n_v$, their impact seems to be similar, i.e., there is no clear candidate to be tackled first. Hence, a reasonable strategy for enhancing the applicability of this or similar approaches seems to be to introduce the notion of modularity. If the problem can be split into several modules (each with small values for $n_x$, $n_R$, and $n_v$), the solution of the optimal control problems should be efficiently possible for each of them. The challenging task is to construct a solution for the complete system from the composition of the single solutions.

REFERENCES


