12th IFAC/IEEE Workshop on Discrete Event Systems
Cachan, France. May 14-16, 2014

An Integrated CP/OR Method for Optimal Control of Modular Hybrid Systems ⋆

Oskar Wigström * **Bengt Lennartson** *

* *Automation Research Group, Department of Signals and Systems, Chalmers University of Technology, Gothenburg, Sweden (oskar.wigstrom+bengt.lennartson@chalmers.se).*

Abstract: This paper concerns the optimal control of modular hybrid systems synchronized by shared variables. Instead of synchronizing the discrete dynamics of the system into one global mode before optimization, Constraint Programming (CP) is used to model the discrete dynamics of each modular system separately. Integrated in the CP solver are also classic Operations Research (OR) models in the form of Nonlinear Programs (NLPs) approximating the continuous dynamics of the system. Using CP considerably decreases the number of NLPs which must be solved, compared to that of using a traditional mixed integer nonlinear programming approach.

Keywords: Hybrid modes, Optimal control, Mathematical programming, Constraint satisfaction problems, Discrete event systems

1. INTRODUCTION

Hybrid systems are systems which contain both continuous and discrete behaviour. Optimization methods for hybrid systems aim to compute optimal or sub-optimal trajectories for the systems. In this paper, we present an optimal control approach for the case of modular hybrid systems linked by shared variables. The method is specifically suited for problems where the discrete dynamics by themselves may lead to infeasibilities.

There are several approaches to the computation of optimal trajectories for hybrid systems. In some instances, it can be possible to abstract the continuous time dynamics in order to get a convex formulation, Wigström et al. (2013). But unless some simplification or abstraction is made, a general hybrid optimal control problem usually result in a non-convex optimization problem. Methods include Dynamic Programming Hedlund and Rantzer (1999), the Maximum Principle Sussmann (1999) and Mixed Integer Nonlinear Programming (MINLP) Oldenburg and Marquardt (2008). The method in this paper improves upon those in the MINLP category.

In a MINLP approach, boolean variables are used to model the discrete dynamics, and the continuous dynamics approximated by nonlinear constraints. Most often, a MINLP is solved by branch and bound, in one form or another. For systems with complex discrete dynamics, a drawback of the MINLP approach is that infeasibilities caused by discrete choices may not be detected until far down into the branches. Because of this, a lot of nodes are explored unnecessarily, something which is computationally expensive when each node contains an approximation of continuous dynamics.

In Wigström and Lennartson (2013), an algorithm which uses Constraint Programming (CP) integrated with Nonlinear Programming (NLP) was used to solve a scheduling problem with a nonlinear cost function. The key idea is to utilize CP in order to reduce the number of nodes, and in doing so, the number of NLP sub-problems. It was suggested in Wigström and Lennartson (2014) that the algorithm may be extended to include continuous dynamics. In optimal control problems, collocation is a common approach used to approximate continuous dynamics Benson (2005). In this paper, we extend the algorithm to include collocation of continuous dynamics by psuedospectral methods. Also, we add additional functionality to further reduce the number of NLP sub-problems.

To summarize, the main contribution of this paper is the utilization of the modular structure of hybrid systems for optimization including nonlinear cost. CP is used to reduce the number NLP sub-problems, due to CPs ability to early detect infeasible solutions. Note that because the approximation method for the continuous dynamics results in non-convex constraints, only local optimality of the generated trajectories may be guaranteed.

2. PROBLEM FORMULATION

The study of hybrid systems has resulted in a large number of modeling formalisms. In this paper, the Hybrid Automaton (HA) Alur et al. (1992) Cassandras and Lafortune (2006) is used to describe the individual hybrid systems. HAs are generalized finite-state machines with the usual discrete transitions as well as continuous state dynamics which can vary with each mode (discrete state). The continuous states may also influence the discrete transitions.

Shared variables provide a convenient method of synchronization between the individual systems Lennartson et al.

⋆ This work was carried out at the Wingquist Laboratory VINN Excellence Center within the Area of Advance – Production at Chalmers, supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) and the Swedish Research Council. The support is gratefully acknowledged.

(2014). As such, shared variables in the form of integers have been added to the HA. To provide a clearer formulation, these variables are separated from the normal state variables.

2.1 Hybrid Automaton with Shared Variables

We define an individual HA with shared (integer) variables (HASV) as (notation based on Cassandras and Lafortune (2006))

$$G_{HASV} = \langle Q, S, X, U, f, \phi, Inv, guard, \rho, q_0, x_0 \rangle \quad (1)$$

where $Q = \{q_1, ..., q_m\}$ is a set of discrete states or modes, S is the domain of the shared integer variables, $S \subseteq \mathbb{Z}^{n_s}$, X is a continuous state space, $X \subseteq \mathbb{R}^{n_x}$, U is a set of admissible input signals, $U \subseteq \mathbb{R}^{n_u}$, f is a vector field, $f : Q \times X \times U \to X$, ϕ is a discrete state/variable transition function $\phi : Q \times S \times X \to Q \times S$, Inv is a set defining an invariant condition, $Inv \subseteq Q \times X$, $guard$ is a set defining a guard condition, $guard \subseteq Q \times Q \times X$, ρ is a reset function, $\rho : Q \times Q \times X \to X$, q_0 is the initial discrete state and x_0 is the initial continuous state. Note that we have excluded events from the formulation as the synchronization of HA will be based on shared variables.

The vector field f describing the dynamics of X takes the form

$$\dot{x}(t) = f(q(t), x(t), u(t)), \quad (2)$$

for times $t \neq t_k$ and $t_k : \ k = \{1, 2..\}$ are the time instances when transitions occur.

At each discrete transition, the reset condition ρ updates the continuous state vector as

$$x(t_k^+) = \rho(q(t_k^+), q(t_k), x(t_k)) \quad (3)$$

where t_k is an arbitrary time when a discrete transition occurs. The discrete mode and shared integer variables are updated as

$$\begin{bmatrix} q(t_k^+) \\ s(t_k^+) \end{bmatrix} = \phi(q(t_k), s(t_k), x(t_k)). \quad (4)$$

2.2 Synchronization

As previously mentioned, the links between separate individual systems are shared integer variables. These are updated by the discrete state/variable transition function ϕ. For the purposes of this paper, shared variables act much like resources which are incremented at allocation and decremented when deallocated. For example, an allocation of a shared variable s is stated simply $s(t_k^+) = s(t_k) + 1$.

The update conditions on a shared variable will disable a transition if its new value, $s(t_k^+)$, happens to lie outside its domain. That is, a transition may only be executed if $s(t_k^+) \in S$.

Consider for example n water tanks sharing one input flow which can be diverted to only one tank at a time. If each system i, $i \in 1..n$, is defined by tank area A_i, water level $h_i(t)$, input flow $u_i(t)$ and an output flow $v_i(t)$. Then each system can be described by two modes: (i) the initial mode with no input flow is described by $A_i \dot{h}_i(t) = -v_i(t)$; (ii) the second mode with inflow diverted to tank i is $A_i \dot{h}_i(t) = u_i(t) - v_i(t)$. To make sure the input flow is used by only one tank at a time, a shared variable $s_u \in [0, 1]$, $s(0) = 0$, is defined, together with the two transition conditions: (i) from initial to second mode $s_u(t_k^+) = s_u(t_k) + 1$; (ii) from second back to initial mode $s_u(t_k^+) = s_u(t_k) - 1$.

Synchronizing the systems entails finding the global order of transitions including shared variables, such that all shared variables are within their domain.

2.3 Minimization Criteria

In a typical optimal control problem, the cost functional to be minimized usually includes either or both an endpoint cost and an integrated cost. The former describes a cost related to the initial and final state of the system while the latter expresses a cost accumulated during the execution of a trajectory. In this paper, these two costs will be separated for simplicity.

For each HASV there is a cost functional expressing the integrated cost as

$$c = \int_0^{T_f} g(q(t), x(t), u(t))dt, \quad (5)$$

where T_f is the final time of the system and g is a vector field, $g : Q \times X \times U \to \mathbb{R}$.

Similarly to the discrete mode and shared integer variables update ϕ, the endpoint cost Φ is updated as

$$d(t_k^+) = \Phi(q(t_k^+), q(t_k), x(t_k), t_k, x(t_{k-1}^+), t_{k-1}^+) \quad (6)$$

For a set of HASV, the optimal control problem entails finding the mode sequence, transition times and controls which minimize the sum over c and $d(T_f)$ in all systems.

3. PRELIMINARIES

A hybrid optimal control problem described in the previous section can be decomposed into two problems: selecting a mode sequence and finding the continuous controls and transition timings for the specified sequence. The mode sequence selection problem is discrete, while the controls and transition timings are continuous. In our algorithm, we use the complementary strengths of methods suited for discrete scheduling decisions and approximating continuous dynamics respectively.

3.1 Discrete Dynamics

The discrete dynamics of an HASV with shared variables can be modeled as a scheduling problem, either by Mixed Integer Linear Programming (MILP) or CP. In the former case, there are a number of procedures for converting discrete event systems into mixed integer constraints,

depending on the original modeling format: Deterministic Finite Automata, Kobetski and Fabian (2009); Extended Finite Automata, Thorstensson et al. (2011); or a general scheduling model, Wigström and Lennartson (2012).

One key assumption we make here is that the discrete dynamics in each Hybrid Automaton include no looping behaviour, i.e. each mode is visited at most once. Looping behaviour can of course be unfolded a preset number of times before hand to allow optimization.

Optimization of an individual hybrid systems by MINLP and Generalized Disjunctive Programing is suggested in for example Oldenburg and Marquardt (2008). While straight forward, using a mixed integer formulation for the optimal control of hybrid systems, has its drawbacks. Applied to scheduling problems, relaxation of the integer variables do not provide a very tight bounds on the original problem, Hooker and Osorio (1999). As such, infeasibilities which are caused by the discrete dynamics of the systems might not be detected at an early stage. For a standard scheduling problem with a linear cost and no additional constraints, this might not matter as each node is quick to evaluate. However, when continuous dynamics are included, the computational cost at each node increases dramatically, and any possible reduction in the number of evaluated nodes becomes important.

In CP, a branch and bound tree is created much as in classic Operations Research (OR) methods. But while OR takes a global approach to the variables and constraints in each node, CP will process each constraint individually and try to reduce the domain of the variables included in the constraint according to some rule-set, this is called propagation. An introduction to CP can be found in for example Apt (2003). Note that a constraint might be subject to propagation more than once in each node. Also, in contrast to OR methods, CP includes a high number of specialized constraints, some specifically designed for scheduling. For example, the mutual exclusion over time intervals in a unary resource may be modeled by one single constraint. There are also constraints which model alternatives.

Converting the discrete dynamics of a set of HASVs to a CP model is much simpler than to that of a MILP model, see for example Wigström and Lennartson (2012). However, because the constraints available in different CP solvers may differ somewhat, we provide only a general overview of the procedure. In short, for each modular HASV, each mode is instantiated by a start and an end time. The directed graph of the each modular HASV is traversed and any mode with two or more outgoing transitions is treated as an alternative.

Finally, the shared variables are processed. If for example the shared variables are single capacity resources where the allocation and deallocation come in pairs, a mutual exclusion constraint may be posted over the allocation intervals, this is commonly known as a no-overlap or unary resource constraint, Vilím (2004). There are also constraints for the cases where the resources are of multi-capacity type and/or the allocations/deallocations do not appear in pairs Aggoun and Beldiceanu (1993).

3.2 Approximation of Continuous Dynamics

The approximation of the continuous dynamics can be made in a number of ways. In the case of linear dynamics, the whole hybrid optimal control problem could be modeled with a fixed sample length as a Mixed Integer Quadratic Program Karaman et al. (2008). While this would permit proof of global optimality and models with looping behaviour, it would also for each mode and sample, require a boolean variable specifying if the mode is active in the specific sample. For systems with a large number of modes, the number of boolean variables may grow intractable. Instead, we allow the sampling time to be free, dynamics to be nonlinear, and focus on locally optimal solutions.

The lack of analytical solutions for optimal control problems with nonlinear dynamics has lead to a number of numerical methods for solving optimal control problems. These methods can be divided into two categories: indirect methods and direct methods. Indirect methods apply calculus of variations to determine the first-order necessary optimality conditions. Direct methods approximate the continuous time problem by a finite-dimensional NLP. An introduction to indirect and direct methods can be found in Garg (2011).

In this paper direct methods are used to approximate the continuous dynamics of the problem. Within the class of direct methods, there are several approaches available. We have opted for the Gauss pseudospectral method Benson (2005). The following will provide a brief description of how the continuous dynamics in *a single* mode can be approximated.

The Gauss pseudospectral method differs from other pseudospectral methods in that the differential equation is not collocated at the boundary points. Also, the approximating polynomial, $X(t)$, is of degree N, such that $N+1$ points are required to determine its derivative.

Let $L_\ell(t)$, $(\ell = 0, ..., N)$, be a basis of Lagrange polynomials, Davis (1975), on the interval from $[-1, 1]$ given by

$$L_\ell(t) = \prod_{\substack{i=0 \\ i \neq \ell}}^{N} \frac{t - t_i}{t_\ell - t_i}, \qquad \ell = 0, ..., N. \tag{7}$$

We see that

$$L_\ell(t_i) = \begin{cases} 1 & \text{if } \ell = i \\ 0 & \text{if } \ell \neq i \end{cases} \tag{8}$$

The state vector $x(t)$ is approximated by $X(t)$ as

$$x(t) \approx X(t) = \sum_{\ell=0}^{N} x(t_\ell) \cdot L_\ell(t). \tag{9}$$

Recall that the polynomial is defined on the interval [-1,1]. As such, the interpolation points used are the boundary point, -1, and the N Gaussian quadrature points, t_ℓ, $\ell = 1, \cdots, N$. Thus, by (8)

$$X(t_\ell) = x(t_\ell), \quad \ell = 0, ..., N. \tag{10}$$

Differentiation of (9) yields the following expression for the approximated state derivatives $\dot{X}(t_i)$

$$\dot{x}(t_i) \approx \dot{X}(t_i) = x(-1) \cdot \bar{D}_i + \sum_{\ell=1}^{N} x(t_\ell) \cdot D_{i\ell}, \qquad (11)$$

where $D_{i\ell} = \dot{L}_\ell(t_i)$ and $\bar{D}_i = \dot{L}_0(t_i)$.

Now, if an modular HASV occupies a mode mode q_j during $[t_k^+ \ ... \ t_{k+1}]$, then the dynamics (2) during that time interval can be approximated by

$$\frac{2}{(t_{k+1} - t_k^+)} \bar{D}_i \cdot X(t_k^+) + \frac{2}{(t_{k+1} - t_k^+)} \sum_{\ell=1}^{N} D_{i\ell} \cdot X(t_\ell) = f(q_j, X(t_\ell), U(t_\ell), t_\ell), \quad i = 1, \cdots, N. \qquad (12)$$

The terminal states are enforced by quadrature approximation of the dynamics as

$$X(t_f) = X(t_0) + \frac{(t_{k+1} - t_k^+)}{2} \sum_{\ell=1}^{N} w_\ell \cdot f(q_j, X(t_\ell), U(t_\ell), t_\ell). \qquad (13)$$

where $U(t_\ell)$ is an $N-1$ degree approximation of the controls $u(t)$ collocated at the Gauss points; w_ℓ are the gauss weights; $\frac{(t_{k+1} - t_k^+)}{2}$ comes from transforming the problem from $t \in [-1, 1]$ into $[t_k^+, t_{k+1}]$.

The cost (5) can in each mode be approximated using a Gauss quadrature such that

$$c_j = \int_{t_k^+}^{t_{k+1}} g(q_j, x(t), u(t)) dt \approx \frac{(t_{k+1} - t_k^+)}{2} \sum_{\ell=1}^{N} w_\ell \cdot g(q_j, X(t_\ell), U(t_\ell), t_\ell). \qquad (14)$$

The decision variables for each mode q_j are the states, $X(t_\ell)$, controls $U(t_\ell)$, transition times t_k^+ and t_{k+1}.

4. INTEGRATED METHOD

The general execution of the integrated algorithm presented in this paper is stated in Algorithm 1. The following provides a short summary.

First of all, a branch and bound procedure is initiated on the master problem, with the discrete decisions as the branching variables.

In Wigström and Lennartson (2013), where the nonlinear functions treated are convex, a computationally inexpensive linear approximation of the NLP was solved in all but the leaf nodes. Because the NLPs in each node are now much more expensive, measures to reduce the number of NLPs must be taken.

For each node in the master problem, two separate CP models are created, one for each direct child of the current node. These two CP models are solved to determine if there child nodes contain feasible solutions with regard to the discrete dynamics in the hybrid model.

In the case of neither child node containing a feasible solution, the current node is marked as infeasible.

If only one child contains a feasible solution in its sub-tree, there is no reason to create an NLP in the current node. This is because we know the master CP will later remove the infeasible branch and also process to the feasible one. This is unless of course solving an NLP can bound the current branch. But since a lower branch will yield a tighter bound on the objective, it is better to save the NLP generation till the feasible child node is processed by the master problem.

If solutions are available in both child nodes, an NLP containing the continuous dynamics is created. The NLP also contains the discrete decisions already fixed by the parents of the current node.

As mentioned, our integrated algorithm uses several optimization models to solve the problem: (i) the master problem, $\mathrm{CP_M}$ (ii) two feasibility sub-problems, $\mathrm{CP_{F1}}$ and $\mathrm{CP_{F2}}$; (iii) the continuous dynamics sub-problem, $\mathrm{NLP_S}$, which includes the continuous dynamics and precedence constraints. The three latter are generated on the fly each time a full or partial solution is found by $\mathrm{CP_M}$. The following provides a description of the models.

Master problem This is the top level of the model which contains all of the discrete dynamics. At the individual hybrid system level the discrete dynamics are the fixed precedence constraints and yet to be determined alternatives in transitions. At the synchronized system level, it is the mutual exclusion constraints implied by the shared variables.

That is, the discrete search space consists of deciding the executing modes if there are alternatives present, and the global order of transitions affecting any shared variables. The transition timings and duration of modes is also included as decision variables, but the branching is exclusively applied to the discrete decisions.

Additionally, $\mathrm{CP_M}$ includes a special cost constraint which relates the active modes and global transition order to the cost to be minimized. After propagation of all other constraints is halted, the propagation routine for the cost constraint is triggered. The propagation routine will first generate: $\mathrm{CP_{F1}}$ and $_{\mathrm{F2}}$, and second $\mathrm{NLP_S}$. $\mathrm{CP_{F1}}$ and $\mathrm{CP_{F2}}$ will indicate if there exists a feasible solution with regards to the discrete dynamics, $\mathrm{NLP_S}$ will provide: (i) a lower bound on the branch if it is an internal node; (ii) a solution (an upper bound) to the full problem.

Feasability problem The two feasibility problems are CPs with no objective other than finding a feasible solution. In each node of $\mathrm{CP_M}$, two feasibility problems are generated. The discrete decision space of $\mathrm{CP_{F1}}$ and $\mathrm{CP_{F2}}$ are the subtrees of the child nodes of the current node in $\mathrm{CP_M}$. The transition timings and mode durations are also included as decision variables.

If a feasible solution is found in both problems, both branches of the current node in $\mathrm{CP_M}$ contain at least one solution, and $\mathrm{NLP_S}$ may be executed. If only one of the

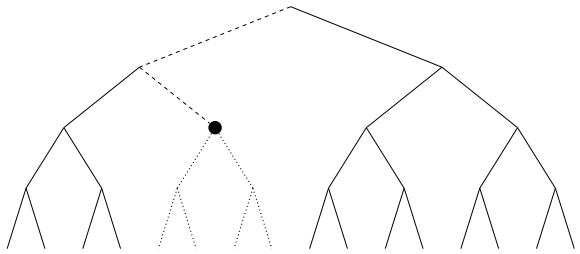

Fig. 1. The whole tree represents the discrete decision space of CP_M. If the current node explored in CP_M is indicated by the black dot, then the following models are generated: CP_{F1} and CP_{F2}, containing the dashed and dotted portions of the tree; NLP_S, containing only the dynamics fixed in the dashed portion

Algorithm 1 Integrated Algorithm

```
UB ← ∞
NodeSet ← Root node
while 0 < |NodeSet| do                    ▷ While any nodes left
    n ← NodeSet                           ▷ Get and remove the next node
    n ← Propagate(n)                      ▷ Use CP for inference
    if n is feasible then
        CP_F1, CP_F2 ← GenerateFeasabilityProblem(n)
        Solve(CP_F1, CP_F2)
        if CP_F1 AND CP_F2 are feasible then
            NLP_S ← GenerateNLP(n)
            cost ← Solve(NLP_S)
            if cost < UB then
                if n is inner node then
                    NodeSet ← Branch(n)
                end if
                if n is terminal node then
                    UB = cost             ▷ New best found
                end if
            end if
        end if
        if CP_F1 XOR CP_F2 are feasible then
            NodeSet ← Branch(n)
        end if
    end if
end while
```

feasibility sub-problems are feasible, the generation of the NLP is skipped, as postponing the NLP to the one feasible child node may only reduce the number of NLPs solved. If both are infeasible, the branch is cut.

Nonlinear Programming sub-problem Once the current CP_M has been verified to contain a feasible solution, the NLP_S is generated. The NLP_S contains only the discrete dynamics already fixed by CP_M. That is, no discrete decisions are considered in the NLP. Because the mode sequence has been fully or partially determined by CP_M, the NLP approximating the continuous dynamics can be regarded as the optimization of controls and switching times for a set of loosely synchronized multiphase systems.

For each mode, the corresponding state approximation, integral cost approximation, invariant conditions and endpoint cost are added. For each transition, the corresponding reset and boundary conditions are added. It is assumed that these can be posed as equalities and inequalities. Finally, timing constraints are added. For each system, equality constraints on the initial and final times of sequenced modes are added. As for the synchronization by shared variables, inequalities are added between systems, corresponding to the partial or full global mode sequence in the current node of CP_M.

Figure 1 contains an illustration of the discrete decision space of the three models. CP_M contains the entire tree. Let the current node in CP_M be indicated by the black dot, then CP_{F1} and CP_{F2} contain the dashed and dotted parts of the discrete search space, and NLP_S only the dashed (already fixed) part.

Note that for the NLP_S there is no reason to include any information on the yet to be fixed discrete constraints further down in the tree. This is because the relaxation on disjunctive constraints is so weak it will not impact the solution.

5. COMPUTATIONAL EXAMPLES

For our implementation, we used CP Optimizer (CPO) in IBM Ilog CPLEX Optimization Studio (v12.51) as CP solver, and Ipopt (3.11) as NLP solver. In addition to applying no-overlap constraints to model the shared variables, reified constraints relating boolean variables to precedences were also added for each mutual exclusion. That is, if the boolean is true then the related constraint must hold, and conversely if the boolean is false then the constraint does not hold. This was the simplest way to make CPO branch on the discrete decisions.

Recall the goal of the algorithm is to reduce the number of solved NLPs. As such simple continuous dynamics have been chosen, each mode consists of one or more double integrators

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix} u. \tag{15}$$

The integrated cost (5) in each HASV is simply

$$\int_0^{t_f} u^T u \; dt. \tag{16}$$

No endpoint cost (6) is considered in the examples.

Each hybrid system consists of a sequence of modes. Each transition has an equality condition guard on the position, x_1. Also, each transition may update one or more variables, i.e. a mode may book one or more zones. If a shared variable is incremented at the start of a mode and decremented at its end, we call this a booking of the shared variable.

Note that as there are no bounds on acceleration or velocity in our examples, the minimum time is simply 0^+. As such, the solutions removed by the CP-algorithms are based on conflicts unrelated to the execution time. If there are bounds on states and accelerations, the computed minimum (and maximum) execution time for each mode may be given to the CP algorithm for increased performance.

Table 1. Shared variables and boundary conditions

Example	Hybrid System	Shared variables booked in mode 1	2	3	4	5	6	Initial and boundary conditions on position state(s) 0	1	2	3	4	5	6
1a	#1	-	0	1	2	6	-	-4.0	-3.5	-2.5	-1.5	-0.5	0.5	1.5
	#2	0	0	1	2	6	-	-3.0	-2.5	-1.5	-0.5	0.5	1.5	2.5
	#3	-	3	4	5	6	-	4.0	3.5	2.5	1.5	0.5	-0.5	-1.5
	#4	3	3	4	5	6	-	3.0	2.5	1.5	0.5	-0.5	-1.5	-2.5
2	#1	-	4	0/4	2/0	-		[4, 0]	[3, 0]	[1, 0]	[-1, 0]	[-3, 0]	[-4, 0]	
	#2	-	1	0/1	3/0	-		[2, 2]	[1, 2]	[0, 1]	[-0, -1]	[0, -3]	[0, -4]	
	#3	-	1	0/1	2/0	-		[-2, 2]	[-1, 2]	[0, 1]	[-1, 0]	[-3, 0]	[-4, 0]	
	#4	-	2	0/2	3/0	-		[-2, -2]	[-2, -1]	[-1, 0]	[0, -1]	[0, -3]	[0, -4]	

5.1 Example 1

In Example 1a, 4 HASV, each with a serial sequence of 6 modes and a single double integrator in each mode. The problem can be regarded as HASVs 1/2 traveling along y-axis and HASVs 3/4 along the x-axis, all systems passing through the origin, booking a number of shared variables on the way. The final time was fixed to 15 time units.

Table 1 shows the initial conditions, transition conditions and which shared variables are booked during which mode. We also present the slightly modified Example 1b, where the booking of variables 0 and 3 are relaxed for systems 2 and 4 in the initial mode increasing the number of feasible solutions by a factor 4.

For Example 1a, the number of NLPs required for the integrated method was 12, and for MINLP 73 NLPs were required. The time to run the CP routines was roughly 1 [s]. In Example 1b which contains a four times as many feasible solutions as Example 1a, the integrated method requires 49 NLPs to be solved while the MINLP needed 212. The required time for CP was about 3 [s] of the MINLP solution time.

Examples 1a and 1b took 12 [s] and 46 [s] to solve using MINLP. Less than 10% of the MINLP solution time used for CP reduces the number of NLPs needed to be solved by more than a factor 4.

5.2 Example 2

In Example 2, 4 HASVs, each with a serial sequence of 5 modes and two double integrators as dynamics. The HASVs share a total of 5 common zones. The position state in the double integrators represents each HASVs position in x/y-space. In contrast to the previous problem, more than one shared variable may be booked in each mode in order to avoid collisions. The final time was fixed to 15 time units.

Solving Example 2 took 191 [s] and required 228 NLPs for the MINLP algorithm. The integrated algorithm took only 49 NLPs, with 7 [s] needed for the CP routines.

6. CONCLUSIONS

This paper presents an algorithm for the optimization of modular hybrid automata with shared variables. The algorithm combines the strength of OR with that of CP. In two computational examples, the algorithm reduces the number of NLPs needed to be solved in a classic mixed integer nonlinear programming approach by more than four times. The required computational time for the added CP routines are less than 10% of the execution time of the MINLP method.

Possible next steps, investigate loops, uncontrollable events, more complex examples reducing energy for multi-robot cells as a challenging example. Also, the modeling of the discrete dynamics should be extended in direction towards the model in Lennartson et al. (2014) which also uses shared variables.

REFERENCES

Aggoun, A. and Beldiceanu, N. (1993). Extending chip in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling*, 17(7), 57 – 73.

Alur, R., Courcoubetis, C., Henzinger, T., and Ho, P.H. (1992). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. 209–229. Springer-Verlag.

Apt, K. (2003). *Principles of constraint programming*. Cambridge University Press.

Benson, D. (2005). *A Gauss pseudospectral transcription for optimal control*. Ph.D. thesis, Massachusetts Institute of Technology.

Cassandras, C. and Lafortune, S. (2006). *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Davis, P. (1975). *Interpolation and approximation*. Dover publications.

Garg, D. (2011). *Advances in global pseudospectral methods for optimal control*. Ph.D. thesis, University of Florida.

Hedlund, S. and Rantzer, A. (1999). Optimal control of hybrid systems. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 4, 3972–3977. IEEE.

Hooker, J.N. and Osorio, M.A. (1999). Mixed logical-linear programming. *Discrete Applied Mathematics*, 96, 395–442.

Karaman, S., Sanfelice, R.G., and Frazzoli, E. (2008). Optimal control of mixed logical dynamical systems with linear temporal logic specifications. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 2117–2122. IEEE.

Kobetski, A. and Fabian, M. (2009). Time-optimal coordination of flexible manufacturing systems using deterministic finite automata and mixed integer linear programming. *Discrete Event Dynamic Systems*, 19(3), 287–315.

Lennartson, B., Basile, F., Miremadi, S., Fei, Z., Hosseini, M.N., Fabian, M., and Åkesson, K. (2014). Supervisory

Control for State-Vector Transition Models - A Unified Approach. *IEEE Transaction on Automation Science and Engineering*, 11(1).

Oldenburg, J. and Marquardt, W. (2008). Disjunctive modeling for optimal control of hybrid systems. *Computers and Chemical Engineering*, 32(10), 2346 – 2364.

Sussmann, H.J. (1999). A maximum principle for hybrid optimal control problems. In *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, volume 1, 425–430. Ieee.

Thorstensson, C., Kanthabhabhajeya, S., Lennartson, B., and Falkman, P. (2011). Optimization of discrete event systems using extended finite automata and mixed-integer nonlinear programming. In *Proceedings of the 18th IFAC World Congress, 2011*, volume 18.

Vilím, P. (2004). O (nlog n) filtering algorithms for unary resource constraint. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 335–347. Springer.

Wigström, O. and Lennartson, B. (2013). Integrated or/cp optimization for discrete event systems with nonlinear cost. In *Decision and Control, 2013 IEEE Conference on, to be published.*

Wigström, O. and Lennartson, B. (2014). Towards integrated or/cp energy optimization for robot cells. In *Robotics and Automation, 2014 IEEE International Conference on, Submitted for possible publication.*

Wigström, O., Lennartson, B., Vergnano, A., and Breitholtz, C. (2013). High level scheduling of energy optimal trajectories. *IEEE Transactions on Automation Science and Engineering.*

Wigström, O. and Lennartson, B. (2012). Scheduling model for systems with complex alternative behaviour. In *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, 587–593. IEEE.