Proceedings of the 4th IFAC Conference on Analysis and Design of Hybrid Systems (ADHS 12)
June 6-8, 2012. Eindhoven, The Netherlands

Optimization of Hybrid Systems with Known Paths ⋆

Oskar Wigström * **Nina Sundström** * **Bengt Lennartson** *

* *Automation Research Group,Department of Signals and Systems, Chalmers University of Technology, SE-412 96, Göteborg, Sweden, (e-mail: oskar.wigstrom@chalmers.se)*

Abstract: In this paper we study a subset of hybrid systems and present a generalized method for their optimization. We outline Hybrid Cost Automata (HCA), an extension to Hybrid Automata, where discrete and continuous cost expressions are added. The class of hybrid systems with known spatial paths is defined in the context of HCA. This type of system is common in industry where for example AGVs transport goods from one location to another, or manipulators move between joint coordinates. The optimization is performed using Dynamic Programming as a preprocessing step, whereafter Mixed Integer Nonlinear Programming is used for scheduling. A case study of a four robot cell is presented with energy consumption used as a minimization criterion.

Keywords: Optimal control; Trajectory; Hybrid automata

1. INTRODUCTION

Much work has been put into analysis and optimization of general hybrid systems, see for example Johansson and Rantzer [1998], Hedlund and Rantzer [1999] or for an extensive summary, Barton and Lee [2002]. In this paper we present a slightly more specialized method geared towards a subset of hybrid systems, those with known paths and time invariant cost functions. In manufacturing, many systems are comprised of robots and other moving devices. These devices most often move between waypoints and the paths between these can be assumed to be known, or more trivial to compute. First, we outline a useful extension for Hybrid Automata (HA), Hybrid Cost Automata (HCA), where continuous and discrete cost expressions have been added. Based on the HCA we define the class of hybrid systems with known paths. We present an optimization method that efficiently solves instances of this problem class, exploiting its special structure. The continuous time dynamics are abstracted using local cost functions and the resulting problem is that of scheduling discrete transitions based on nonlinear cost functions.

One particularly important design driver for manufacturing systems is energy efficiency. This type of criteria will result in nonlinear but most often convex cost functions, much like the general problem of hybrid systems with known paths considered in this paper. In the energy minimization case, optimization of mechatronic devices is well investigated in Saidur [2010], Visinka [2002], Yang et al. [2009], Hirzinger et al. [2002]. Energy optimal trajectories for robot applications is a big research field itself, see e.g. Diken [1994], Park [1996] and Sergaki et al. [2002]. From a system design perspective, a selection and matching of efficient design solutions for pre-defined operations is studied in Maimon et al. [1991], Roos et al. [2006], Izumi et al. [2009]. The optimization of the scheduling supervisor of the overall system is a promising area. Two approaches are presented in Kobetski and Fabian [2008], where idle time between the operations is used to reduce velocities and accelerations, without concern to the energy consumption. In Vergnano et al. [2010], a method based on the reduction of velocities and accelerations with concern to energy was presented. Based on a dynamic scaling of trajectories, we presented an optimization method in Wigström and Lennartson [2011] Wigström et al. [2012], which serves as the foundation for the generalization in this paper.

HA can be used to model hybrid systems. They do however lack a formal definition of costs, both in the continuous and discrete sense. Similar as Timed Automata have been extended with linear costs into Priced Timed Automata Behrmann et al. [2001], we suggest an extension of HA into HCA. Observe that, HCA can be regarded as a generalization of Priced Time Automata in the same way as HA can be regarded as a a generalization of Timed Automata. Normally, cost expressions are included into the continuous state equations, e.g. Barton and Lee [2002]. Using HCA allows system dynamics to be modeled seperate from costs expressions in a self-contained way, clarifying the underlying system structure. Note that what differentiates continuous and discrete costs from states and modes (discrete states), are that the two former influence neither dynamics, transitions, guard conditions nor invariant conditions.

Our optimization method can be divided into two parts. First, for each mode (discrete state) in the system, enumerate all the combinations of initial and final states. These combinations are assumed to have a known path as well as some other properties presented in the next section.

⋆ 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.

Dynamic programming can be applied as in Wigström et al. [2012] to generate the local optimal cost for each path combination as a function of time. This will abstract the continuous time dynamics, i.e. flatten the HCA into a Timed Automata with nonlinear costs. The second part of the optimization, is scheduling the discrete transitions of the flattened system. The resulting scheduling problem is solved using Mixed Integer Nonlinear Programming as in Wigström et al. [2012] to generate the optimal schedule.

Note that, if the local cost functions are non-convex, they should be convexified before step two, e.g. divided into convex intervals. Also, the optimization formulation scales well for larger problems as the complexity for adding local costs increases linearly in the first step. The exponential complexity of the second step has so far been shown to constitute the smaller part of the computation time for our case study example. For a larger problem instance, partitioning might be necessary for a tractable formulation.

The paper is structured as follows. Section 2 will give an outline of Hybrid Cost Automata and explain what conditions are necessary for the optimization method. Section 3 contains the first part of the optimization method, the trajectory planning problem formulation and its optimization model. Section 4 briefly covers a Mixed Integer Nonlinear Programming scheduling model. Section 5 presents the results from a four robot test case and finally in Section 6 conclusions are drawn along with a short discussion.

2. HYBRID SYSTEMS

A hybrid system is a dynamical system, which behavior is described by both discrete and continuous dynamics. One modeling framework for hybrid systems is the Hybrid Automata (HA) Alur et al. [1992] Cassandras and Lafortune [2006], a generalized finite-state machine. In addition to the usual discrete transitions, there are also continuous states with dynamics that can vary for each mode (discrete state). The continuous states can also influence the discrete transitions. Amongst other things, HA are used for model checking, simulation and optimization. With respect to optimization, we argue that the HA framework is somewhat lacking. In this paper, we outline an extension of the HA with cost function expressions into Hybrid Cost Automata (HCA).

HCA include both continuous as well as discrete instantaneous costs. With this modeling formalism a hybrid system, including dynamics and relevant optimization criteria or constraint parameters can be expressed on a self contained form. The following subsections will provide a definition of HCA and also show how under certain conditions, the criteria of HCA can be optimized using our method.

Note that what distinguishes cost functions from continuous states is that the former influence neither state transitions, guard conditions nor invariant conditions. Also, the HCA does not infer how optimization should be performed or how criteria should be weighted. It is, and should only be considered as a model of a hybrid system and its costs. An optimization model should only include a relevant subset of the information contained in the HAC.

2.1 Hybrid Cost Automata

If the HA is augmented with continuous and discrete cost criteria, the resulting HCA can be defined by the following tuple (notations based on Cassandras and Lafortune [2006])

$$G_{HCA} = \langle Q, X, E, U, C, D, f, \phi, g, \delta, Inv, guard, \rho, q_0, x_0 \rangle \quad (1)$$

where $Q = \{q_1, ..., q_m\}$ is a set of discrete states or modes, X is a continuous state space, $X \subseteq \mathbb{R}^{n_x}$, E is a set of events, U is a set of admissible input signals, $U \subseteq \mathbb{R}^{n_u}$, C is a set of continuous costs, $C \subseteq \mathbb{R}^{n_c}$, D is a set of incrementally updated costs, $D \subseteq \mathbb{R}^{n_d}$, f is a vector field, $f : Q \times X \times U \to X$, ϕ is a discrete state transition function $\phi : Q \times X \times E \to Q$, g is a vector field, $g : Q \times X \times U \to C$, δ is a set of discrete costs, $\delta : Q \times X \times E \to D$, 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, $\rho : Q \times Q \times X \times E \to X$, q_0 is the initial discrete state and x_0 is the initial continuous state.

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

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

for times $t \neq t_k$ and $t_k : \ k = \{1, 2..\}$ are the time instances when transitions occur. In the same way, for $t \neq t_k$, g describes the evolution of the continuous cost vector c as

$$\dot{c} = g(q, x, u), \quad (3)$$

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

$$x(t_k^+) = \rho(q(t_k^+), q(t_k), x(t_k), e(t_k)) \quad (4)$$

where t_k is an arbitrary time when a discrete transition occurs. The discrete mode is updated as

$$q(t_k^+) = \phi(q(t_k), x(t_k), e(t_k)) \quad (5)$$

In much the same way, the discrete cost evolves as

$$d(t_k^+) = d(t_k) + \delta(q(t_k), x(t_k), e(t_k)) \quad (6)$$

where $d(t_k) = d(t_{k-1}^+)$, as d_k has not been updated since $d(t_{k-1}^+)$. This also holds for $q(t_k)$ and $q(t_{k-1}^+)$.

Because $\dot{c}$ in (3) is independent of c, we can apply integration to form the following expression for the vector cost c.

$$c = \int_0^t g(q, x, u) dt, \quad (7)$$

We can also rewrite (6) as

$$d(t_k^+) = \sum_{i=1}^{k} \delta(q(t_i), x(t_i), e(t_i)), \quad (8)$$

the sum of the discrete cost over all transitions. How the two set of criteria, (7) and (8), are ultimately used is application specific. In general, the final optimization problem can be posed as

$$\begin{aligned} \min \quad & J(c, d) \\ \text{subject to} \quad & h(c, d) = 0 \\ & p(c, d) \leq 0 \end{aligned} \tag{9}$$

where $J(c, d)$ is an objective function, $h(c, d)$ is an equality constraint and $p(c, d)$ an inequality constraint. The objective function is of course implicitly subject to the HCA's dynamics. Observe that c, d, p and h are generally vectors.

The HCA formulation is similar to that of Priced Timed Automata (PTA) Behrmann et al. [2001], an extension of Timed Automata. PTA have added linear costs on modes and discrete costs on transitions, utilizing the existing clock framework of time automata. Linear costs in our case would imply $\dot{c} = g(q)$, where $g(q)$ is constant in each mode q. Just as HA can be thought of as a generalization of Timed Automata including general state equation models (2), HCA generalizes the linear costs from PTA into continuous state and discrete mode dependent nonlinear costs.

2.2 Optimization under special conditions

When scheduling industrial manufacturing systems, the paths of moving systems are most often known. For example, an AGV is to move from one position to another via a number of adjacent nodes. The path between each pair of nodes is most often trivial to define. The paths of industrial robots are not quite as simple, but a multitude of algorithms exists that can generate a path. Some background on path planning can be found in Sciavicco et al. [2000].

We have developed a two stage method for optimizing this type of common industrial system. First, the continuous dynamics of each possible path in each mode is locally optimized as to yield the optimal cost as a function of time. By performing this step, J in (9) is locally optimized and the continuous time dynamics are abstracted. The resulting problem is in essence a TA with nonlinear costs. This abstraction is possible due to the special characteristics of the problem. The second stage consists of scheduling the discrete transitions, subject to h and p in (9). The constraints in (9) are thus only considered on a global level. The local optimization in stage one can of course still be constrained by Inv, the invariant conditions.

The following conditions need to be fulfilled for the method to be applicable. The continuous states should consist of only coordinates (position, angle, etc.) and their time derivatives. The paths of these coordinates should be known, the speed and acceleration are however not necessary. Also, for each mode, the possible initial values of the continuous states should be a well defined discrete set. In terms of HCA, this means that all transitions leading into a mode must either have an equality guard condition or the continuous states must be reset. It is also assumed that there exists an input u such that any invariant conditions are upheld.

In the next section, we show how a trajectory planning method can, for a given path, generate the optimal cost (and trajectories) as a function of time. If the given paths from the initial state in every mode to any outgoing transitions are subjected to this optimization method, then the optimal continuous state evolution in every mode is known. After this first optimization step, we flatten the HCA into a Timed Automaton with nonlinear costs.

3. TRAJECTORY PLANNING

A trajectory planning problem can be described as generating the set of control inputs that will move an object along a predefined geometric path without violating any dynamic or kinematic constraints. Trajectory planning has been an area of research since the early 1970s Kahn and Roth [1971], an excellent overview of the last three decades can be found in Gasparetto and Zanotto [2008].

This paper uses dynamic programming as in Wigström et al. [2012] in order to solve the trajectory planning problem for a range of execution times simultaneously. The grid required is as small as two dimensions and yields an optimal trajectory with discontinuous acceleration. If the grid is of high enough resolution and the size of the acceleration discontinuities are constrained, taking jerk into further consideration should not be necessary. Also note that since the optimization is based on a single scaling parameter, the dimensionality will be unchanged for the number of spatial dimensions as well as more intricate cost expressions. By numerical means, the solution has been shown to converge as grid resolution is increased, cf. Wigström et al. [2012].

3.1 Problem formulation

Solving a trajectory planning problem entails finding the control signal required to move a manipulator or other moving device along a predefined geometric path, while upholding its dynamical constraints. Note that there are no mode transitions during each individual trajectory traversal. Also, the optimal control signal u^* is given implicitly by the optimal trajectory. As such, the vector field g (3), describing the cost c can be written as a function of the position, speed and acceleration vectors.

$$\dot{c} = g(x, \dot{x}, \ddot{x}) \tag{10}$$

Let the geometric path be defined by a function $x_p(\tau)$, a parameterized curve dependent on one single variable $\tau(t)$. The time optimal trajectory can be used to define x_p and its derivatives. This implies that τ is the time scale for the time optimal trajectory, x_p. For example, defining $\tau = t$ would result in the time optimal trajectory. The relationship between x and x_p can therefore be expressed as

$$x(t) = x_p(\tau(t)), \qquad 0 \leq \tau \leq \tau_f, \tag{11}$$

where $\tau(t)$ is a monotonically increasing function with a starting value of 0 and final value τ_f, where τ_f in our case corresponds to the time optimal execution time. If $\tau(t_f) = \tau_f$, then t_f is the new final execution time of the dynamically scaled trajectory. The derivatives of x_p with respect to τ are the same as those of the time optimal trajectory with respect to time. Differentiating (11) with regard to time yields expressions for speed and acceleration which are needed for computing the cost function and upholding constraints,

$$\dot{x}(t) = \frac{dx_p(\tau)}{d\tau}\dot{\tau} \tag{12}$$

$$\ddot{x}(t) = \frac{dx_p(\tau)}{d\tau}\ddot{\tau} + \frac{d^2x_p(\tau)}{d\tau^2}\dot{\tau}^2 \tag{13}$$

Further, combining (12) and (13) with (10) results in an new expression for g as a function of τ and $x_p(\tau)$,

$$\dot{c} = g(x_p, \frac{dx_p}{d\tau}, \frac{d^2x_p}{d\tau^2}, \tau, \dot{\tau}, \ddot{\tau}) \tag{14}$$

The optimization procedure is also subject to the invariant conditions and set definitions in (1), e.g. limits on acceleration and speed. This can be implemented using a barrier function, or if the original trajectory is assumed to be time optimal, by adding the constraint $\dot{\tau} \leq 1$.

Since $x_p(\tau)$ and $\tau(0)$ are known we can now express the local cost function c, the integral over (14), as

$$c(t_f) = \int_0^{t_f} g(\dot{\tau}(0), \ddot{\tau})\ dt \tag{15}$$

With this, the vector of local costs for a trajectory $c(t_f)$, is a functional of $\ddot{\tau}$ and $\dot{\tau}(0)$. Since the total local cost for each trajectory $J(t_f)$ (based on (9)) is an arbitrary function of (15), minimizing J is also a matter of finding these, while minimizing the cost and satisfying the constraints.

3.2 Optimization model

As mentioned, solving the trajectory planning problem entails finding the τ that minimizes a given cost function. Define the second derivative of τ as

$$\ddot{\tau}(t) = u(t), \tag{16}$$

where $u(t)$ is a control input. Also, introduce a time-varying sampling time h_k that affects the time updates as

$$t_{k+1} = t_k + h_k \tag{17}$$

and let the input variable be piecewise constant during the sampling intervals, i.e.

$$u(t) = u(t_k), \qquad t_k \leq t < t_{k+1}, \tag{18}$$

The decision to use a piece-wise constant $\ddot{\tau}$ is an abstraction that will restrict the dimensionality of the problem to two. Even though it will introduce small discontinuities in the acceleration through (13), these minor artifacts can be considered marginal. One could instead choose to define the third derivative of τ as constant, and instead achieve only discontinuous jerk, but at the cost of dimensionality and complexity. Discretization of (16), with a sampling period h_k and constant control input as in (17) and (18), gives the discrete state space model

$$\begin{bmatrix} \tau_{k+1} \\ \nu_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & h_k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_k \\ \nu_k \end{bmatrix} + \begin{bmatrix} h_k^2/2 \\ h_k \end{bmatrix} u(t_k) \tag{19}$$

where $\nu = \dot{\tau}$ and for simplicity, we introduce $\tau_{k+1} = \tau(t_{k+1})$, $\tau_k = \tau(t_k)$ and $\nu_{k+1} = \nu(t_{k+1})$, $\nu_k = \nu(t_k)$. The minimization of (15), including this discrete time model of the time function $\tau(t)$ can be solved with dynamic programming, but for computational reasons discussed later, it is convenient to reformulate the problem. Since τ is monotonically increasing it is possible, instead of taking steps along the t-axis in each iteration, to take steps along the τ-axis and let (17) act as a discrete state equation. Define

$$h_k\nu_k + h_k^2 u(t_k)/2 = \Delta_k, \tag{20}$$

where Δ_k can be regarded as a user defined sampling period or gridding of τ. If (20) is inserted into (19), then in every step k, τ will be updated as

$$\tau_{k+1} = \tau_k + \Delta_k \tag{21}$$

Equation (20) can also be manipulated into an expression for the control signal, $u(t_k) = 2(\Delta_k - h_k\nu_k)/h_k^2$. Inserting this expression for $u(t_k)$ into the bottom equation of (19) gives a new state equation for ν. Regarding the sampling time, h_k as the new control signal and letting (17) act as a state space equation lead us to the reformulated discrete state space model

$$\begin{bmatrix} t_{k+1} \\ \nu_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} t_k \\ \nu_k \end{bmatrix} + \begin{bmatrix} h_k \\ 2\Delta_k/h_k \end{bmatrix} \tag{22}$$

The relation between (21) and (17) can be seen as a mapping of τ onto t. From here, dynamic programming can be applied to solve the discrete time optimal control problem. For the implementation, see Wigström and Lennartson [2011]. A detailed account of the theory behind dynamic programming applied to discrete optimal control problems can be found in for example Lewis and Syrmos [1995] or Naidu [2003].

4. SCHEDULING

The 'local cost J' for a trajectory is defined as the contribution of that trajectory to the 'total cost J' in (9). Applying the presented Dynamic Programming algorithm to a path will yeild an optimal local cost $J^*(t_f)$ as a function of the execution time t_f. If the algorithm is applied to all combinations of initial and final states in each mode, the locally optimal continuous state evolution for all paths will be known. This removes the continuous time dynamics from the HCA and flattens it into a Timed Automaton with nonlinear costs.

We would like to schedule the discrete transitions resulting from this flattened HCA. If the system consists of several HCA, this step can be thought of as synchronization, where besides scheduling, forbidden states are identified and removed by mutual exclusion. The constraints governing the scheduling problem can be expressed with linear constraints consisting of real and binary variables. For a thorough account on mixed integer constraint modeling, see for example Pochet and Wolsey [2006] or Williams [1999].

In short, the possible trajectories of all HCAs are enumerated. The decision variables for the problem are the real valued starting and stopping times for these trajectories as well as binary variables representing mutual exclusion. Mutual exclusion stems from for example the synchronization of two or more HCA that utilize the same resources.

Note that in practice, each instance of the Dynamic Programming algorithm will generate a sampled data set of the optimal local cost $J^*(t_f)$. This data set is then approximated as a polynomial. The polynomials are checked for convexity and that the relative error related to the original data is within appropriate bounds. The cost function for the scheduling problem can now be formed using the polynomials. These polynomials make it easy to specify an analytical gradient and hessian for the solver. As such, the problem class is that of a convex Mixed Integer Nonlinear Linear Program (MINLP).

The MINLP master problem can be divided into a number of subproblems, one for each possible schedule of the discrete transitions. These are enumerated and each subproblem is treated as a linearly constrained problem with a nonlinear cost function. Not all of these choices are feasible, depending on the final time specified for the model. After checking for feasibility, the valid subproblems are solved using MATLAB's optimization toolbox. The algorithm employed uses an interior point method combined with a barrier function for the constraints. The interior point method is described in Byrd et al. [2000], Byrd et al. [1997] and Waltz et al. [2006]. There are of course more efficient approaches to the master problem than explicit enumeration. However, as the primary focus of this paper is that of optimization of hybrid systems, implementing advanced scheduling techniques has not been our concern. Also, as mentioned, the exponential complexity of the scheduling problem has so far been shown to constitute the smaller part of the computation time even for problems of industrial size. There are of course many modern techniques for solving convex MINLP including among others: Branch-and-Bound, Outer-Approximation, LP/NLP-based branch and bound Fernandes et al. [2009], Leyffer et al. [2009].

4.1 Constraint modeling

Let the global starting and finishing time for the j:th trajectory executed by the i:th HCA be denoted t^s_{ij} and t^f_{ij}. A sequence of two trajectories is simply expressed by defining the finishing time of the preceding trajectory as smaller than the starting time of the following trajectory,

$$t^s_{ij} \geq t^f_{lm} + \epsilon, \tag{23}$$

where l and m is the HCA and trajectory index of the first trajectory, i and j that of the second and ϵ a significantly small positive constant. It is also required to limit the minimum execution time of trajectories. If a trajectory j in an HCA i has a minimum execution time of $T_{0,ij}$, then the execution time can be constrained by

$$t^f_{ij} \geq t^s_{ij} + T_{0,ij} \tag{24}$$

Shared resources can be expressed in the following way. If two trajectories, ij and lm, each belonging to different local HCAs, share the same resource, then define $\sigma_{ij,lm}$ as a boolean variable representing trajectory ij being performed after lm. Also $\sigma_{lm,ij}$ is a boolean variable representing the negation of the previous statement, i.e. trajectory lm is performed after ij. The resulting constraints for this example are

$$\begin{aligned} t^s_{ij} &\geq t^f_{lm} - M(1-\sigma_{ij,lm}) \\ t^s_{lm} &\geq t^f_{ij} - M(1-\sigma_{lm,ij}) \\ \sigma_{ij,lm} &+ \sigma_{lm,ij} = 1 \end{aligned} \tag{25}$$

In other words, a boolean variable or expression is used to negate constraints when false. As such, the constant M needs to be sufficiently large for this negation to be valid. The same principle can be used for one HCA having an alternative order of execution for its trajectories.

For constraining the cycle time, an additional variable is added. This variable is constrained to be larger than the stopping time of the last trajectory for each HCA. Adding an upper bound for the new variable will now constrain the complete cycle time. With the scheduling problem described by these mixed integer linear constraints and the cost function by the convex polynomials, the optimization model can now be solved using any standard MINLP solver.

5. CASE STUDY

For the case study, an example with four six-joint industrial robots is considered. The joint torques of the manipulators can be expressed by a Lagrange formulation, see Sciavicco et al. [2000], pp. 131-140. The torque, T_i acting on the i:th joint can be expressed as

$$T_i = J_{ij}\ddot{x}^j + C_{ijk}\dot{x}^j\dot{x}^k + F_{ij}\dot{x}^j + G_i \tag{26}$$

where J is the inertia matrix, C the tensor of centrifugal and Coriolis coefficients, F the viscous friction matrix, G the gravitational vector and x^i the angular position of joint i. Note that J, C and G are all functions of x. Considering the trajectory execution times relevant to this paper, as in Park [1996] and Ohm [(2006, Feb. 3], the energy consumption of an AC permanently excited synchronous motor can be expressed by the following simplified voltage and current models

$$\begin{aligned} V_i(t) &= R_i I_i(t) + K_{V,i}K_{R,i}\dot{x}_i(t) \\ I_i(t) &= T_i(t)/(K_{T,i}K_{R,i}) \end{aligned} \tag{27}$$

where $I_i(t)$ and $V_i(t)$ are the equivalent DC current and voltage of the i:th rotor, R_i is the stator resistance, $K_{V,i}$ is the electrical (back emf) constant, $K_{R,i}$ is the transmission gear ratio and $K_{T,i}$ is the equivalent torque constant. With (27), we can define g in (3), as the power of each motor.

$$g_i(t) = V_i(t)I_i(t) = \frac{R_i}{K^2_{T,i}K^2_{R,i}}T^2_i(t) + \frac{K_{V,i}}{K_{T,i}}T_i(t)\dot{x}_i(t) \tag{28}$$

With g defined as above, the continuous cost vector c in (7) becomes a vector expressing the energy consumption of each individual joint

$$c_i = \int_0^t g_i(t)dt = \int_0^t V_i(t)I_i(t)dt \tag{29}$$

For our case study, the objective function for the synchronized system is the energy consumption of all robots. This implies that the minimization criteria for each local trajectory should be the energy consumption of each robot moving along each specific trajectory. In other words, the 'local cost' for each trajectory, the cost function that is minimized using Dynamic Programming, is the sum over the energy consumption of each robot joint. The global criteria is the sum over each 'local cost'.

The four robots work together on a single work piece located in between the robots. Each robot has in the range of 4-12 modes and 1-2 transitions from each mode. The cell includes three common zones in which only one robot can work at a time. The robot cell was modeled with ABB RobotStudio [2011, Nov. 1] from where path/trajectory information for each operation was extracted. All optimization was run on a Windows 7 64bit system with a 2.66 [Ghz] Intel Core2 Quad CPU and 4 [GB] of RAM. The final time feasibility check for each subproblem, which also generates a starting point, took < 0.1 [s] in 98% of the cases. Most subproblems were solved in less than $10[s]$. It should be noted that for a few instances the initial barrier function weighting had to be varied in order for MATLAB to produce an optimal solution. In total, the scheduling took around 5 [min]. As for the trajectory planning problem, a total of 40 instances were solved, each instance taking close to 40 [s]. All the operations scaled resulted in convex functions. This was to be expected as regenerative breaking was not considered.

In Fig. 1, the total energy consumption for all four robots is shown, running an energy optimal schedule. The dashed line shows the result of trajectories based on a minimum time policy, i.e. minimum energy scheduling is performed but trajectories are set to time optimal execution. The solid curve represents our method, dynamic scaling of trajectories. As a reference, the dotted curve is the resulting energy cost of using linear scaling Vergnano et al. [2010], i.e. the parameter $\dot{\tau}$ is a constant. While upholding a

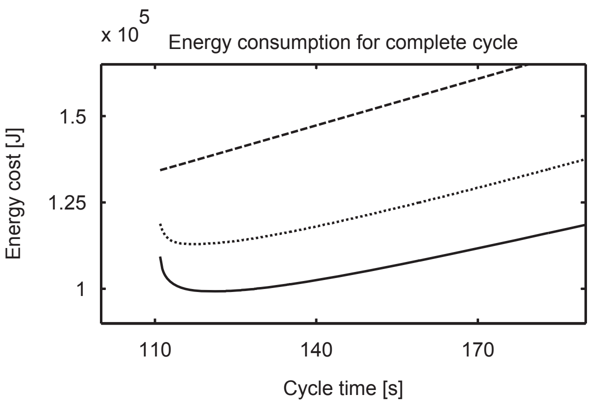

Fig. 1. Overall minimum energy consumption for the case study as a function of cycle time. The dashed line shows scheduling where no scaling is allowed, the dotted and solid curves represent scheduling based on linearly and dynamically scaled operations.

time optimal cycle time, linear scaling reduces the energy usage by 11%, and dynamic scaling a total of 18%. If the cycle time is allowed to be extended by 10% (10[s]), linear scaling will reduce energy cost by 18%, and dynamic scaling as much as 28%.

6. DISCUSSION AND CONCLUSION

In this paper, we presented an optimization method which can be applied to hybrid systems with known paths. We also presented the outline for an extension of Hybrid Automata into Hybrid Cost Automata. This is proposed by adding expressions for continuous and discrete time costs. The optimization method uses a preprocessing step consisting of dynamic programming in which the trajectory planning problem is solved for multiple execution times. This first step actually flattens the Hybrid Cost Automata into a Timed Automata with nonlinear costs. Optimizing the system is then a matter of scheduling the discrete transitions. This is solved using Mixed Integer Nonlinear Programming. The results from the case study are promising and show how for example energy usage can be significantly lowered. The class of moving systems considered in this paper are common in industry and as such, the method presented is directly applicable to many problems of industrial size.

REFERENCES

ABB RobotStudio. `http://www.robotstudio.com`, 2011, Nov. 1.

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

P.I. Barton and C. K Lee. Modeling, simulation, sensitivity analysis, and optimization of hybrid systems. *ACM Trans. Model. Comput. Simul.*, 12:256–289, October 2002.

G Behrmann, A Fehnker, Th Hune, K Larsen, P Pettersson, J Romijn, and F Vaandrager. Minimum-cost reachability for priced time automata. In Maria Di Benedetto and Alberto Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer Berlin / Heidelberg, 2001.

R.H. Byrd, M.E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9:877–900, 1997.

R.H. Byrd, J.C. Gilbert, and J Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89:149–185, 2000. ISSN 0025-5610.

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

H. Diken. Energy efficient sinusoidal path planning of robot manipulators. *Mechanism and Machine Theory*, 29(6):785 – 792, 1994.

F.P. Fernandes, M.F.P. Costa, and E.M.G.P. Fernandes. Overview on mixed integer nonlinear programming problems. *AIP Conference Proceedings*, 1168(1):1374–1377, 2009.

A. Gasparetto and V. Zanotto. A technique for time-jerk optimal planning of robot trajectories. *Robotics and Computer-Integrated Manufacturing*, 24(3):415 – 426, 2008.

S. Hedlund and A. Rantzer. Optimal control of hybrid systems. In *In proceedings of the 38th IEEE Conference on Decision and Control*, pages 3972–3977, 1999.

G. Hirzinger, N. Sporer, A. Albu-Schaffer, M. Hahnle, R. Krenn, A. Pascucci, and M. Schedl. Dlr's torque-controlled light weight robot iii-are we reaching the technological limits now? In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1710 – 1716 vol.2, 2002.

T. Izumi, H. Zhou, and Z. Li. Optimal design of gear ratios and offset for energy conservation of an articulated manipulator. *Automation Science and Engineering, IEEE Transactions on*, 6(3):551 –557, Jul. 2009.

M. Johansson and A. Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. *Automatic Control, IEEE Transactions on*, 43(4):555 –559, apr 1998.

M. E. Kahn and B. Roth. The near-minimum-time control of open-loop articulated kinematic chains. *Journal of Dynamic Systems, Measurement, and Control*, 93(3): 164–172, 1971.

A. Kobetski and M. Fabian. Velocity balancing in flexible manufacturing systems. In *Discrete Event Systems, 2008. WODES 2008. 9th International Workshop on*, pages 358 –363, may 2008.

F.L. Lewis and V.L. Syrmos. *Optimal control.* A Wiley-Interscience publication. J. Wiley, 1995.

S. Leyffer, J. Linderoth, J. Luedtke, A. Miller, and T. Munson. Applications and algorithms for mixed integer nonlinear programming. *Journal of Physics: Conference Series*, 180(1):012014, 2009.

O. Maimon, E. Profeta, and S. Singer. Energy analysis of robot task motions. *Journal of Intelligent and Robotic Systems*, 4:175–198, 1991.

D.S. Naidu. *Optimal control systems.* Electrical engineering textbook series. CRC Press, 2003.

D.Y. Ohm. *Selection of servo motors and drives (rev.2).* `http://www.drivetechinc.com/`, (2006, Feb. 3).

J. S. Park. Motion profile planning of repetitive point-to-point control for maximum energy conversion efficiency under acceleration conditions. *Mechatronics*, 6(6):649 – 663, 1996.

Y. Pochet and L.A. Wolsey. *Production Planning by Mixed Integer Programming (Springer Series in Operations Research and Financial Engineering).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

F. Roos, H. Johansson, and J. Wikander. Optimal selection of motor and gearhead in mechatronic applications. *Mechatronics*, 16(1):63 – 72, 2006.

R. Saidur. A review on electrical motors energy use and energy savings. *Renewable and Sustainable Energy Reviews*, 14(3):877 – 898, 2010.

L. Sciavicco, B. Siciliano, and B. Sciavicco. *Modelling and Control of Robot Manipulators.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2nd edition, 2000.

E.S. Sergaki, G.S. Stavrakakis, and A.D. Pouliezos. Optimal robot speed trajectory by minimization of the actuator motor electromechanical losses. *J. Intell. Robotics Syst.*, 33:187–207, Feb. 2002. ISSN 0921-0296.

A. Vergnano, C. Thorstensson, B. Lennartson, P. Falkman, M. Pellicciari, Chengyin Yuan, S. Biller, and F. Leali. Embedding detailed robot energy optimization into high-level scheduling. In *Automation Science and Engineering (CASE), 2010 IEEE Conference on*, pages 386 –392, 2010.

R. Visinka. *Ch. 2 - Energy Efcient Three-Phase AC Motor Drives for Appliance and Industrial Applications.* Goldberg and Middleton, 2002.

R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Math. Program.*, 107:391–408, Jul. 2006. ISSN 0025-5610.

O. Wigström and B Lennartson. Energy optimization of trajectories for high level scheduling. In *Automation Science and Engineering (CASE), 2011 IEEE Conference on*, 2011.

O. Wigström, B. Lennartson, A. Vergnano, and C. Breitholtz. *High level scheduling of energy optimal trajectories.* Conditionally accepted for publiction in Transactions on Automation Science, 2012.

H. P. Williams. *Model Building in Mathematical Programming, 4th Edition.* Wiley, 4 edition, October 1999.

A. Yang, J. Pu, C.B. Wong, and P. Moore. By-pass valve control to improve energy efficiency of pneumatic drive system. *Control Engineering Practice*, 17(6):623 – 628, 2009.