Scalable Scheduling of Energy Control Systems

Truong X. Nghiem
Automatic Control Laboratory
École Polytechnique Fédérale de Lausanne
Lausanne, Switzerland
xuan.nghiem@epfl.ch

Rahul Mangharam
Dept. of Electrical & Systems Engineering
University of Pennsylvania
Philadelphia, Pennsylvania, USA
rahulm@seas.upenn.edu

ABSTRACT

Peak power consumption is a universal problem across energy control systems in electrical grids, buildings, and industrial automation where the uncoordinated operation of multiple controllers result in temporally correlated electricity demand surges (or peaks). While there exist several different approaches to balance power consumption by load shifting and load shedding, they operate on coarse grained time scales and do not help in de-correlating energy sinks. The Energy System Scheduling Problem is particularly hard due to its binary control variables. Its complexity grows exponentially with the scale of the system, making it impossible to handle systems with more than a few variables.

We developed a scalable approach for fine-grained scheduling of energy control systems that novelly combines techniques from control theory and computer science. The original system with binary control variables are *approximated by an averaged system* whose inputs are the *utilization values* of the binary inputs within a given period. The error between the two systems can be bounded, which allows us to derive a safety constraint for the averaged system so that the original system's safety is guaranteed. To further reduce the complexity of the scheduling problem, we *abstract the averaged system by a simple single-state single-input dynamical system* whose control input is the upper-bound of the total demand of the system. This model abstraction is achieved by extending the concept of simulation relations between transition systems to allow for input constraints between the systems. We developed conditions to test for simulation relations as well as algorithms to compute such a model abstraction. As a consequence, we only need to solve a small linear program to compute an optimal bound of the total demand. The total demand is then broken down, by solving a linear program much smaller than the original program, to individual utilization values of the subsystems, whose actual schedule is then obtained by a low-level scheduling algorithm. Numerical simulations in Matlab show the effectiveness and scalability of our approach.

Categories and Subject Descriptors

C.3 [**Computer Systems Organization**]: Special-purpose and Application-based Systems—*Real-time and embedded systems*; J.2 [**Computer Applications**]: Physical Sciences and Engineering—*Engineering*; J.7 [**Computer Applications**]: Computers in Other Systems

Keywords

embedded control systems, peak power management, cyber-physical systems

.

1. INTRODUCTION

Peak power consumption is a universal problem across energy control systems in electrical grids, buildings, and industrial automation where the uncoordinated operation of multiple controllers result in temporally correlated electricity demand surges. For example, in the case of the electrical grid, when the popular UK TV soap Eastenders comes to an end five times a week, the grid has to deal with around 1.75 million kettles requiring power at the same time (to prepare tea). That is an additional *3 gigawatts* of power for the roughly 3-5 minutes it takes each kettle to boil. So big is the surge, caused by correlated human behavior, that backup power stations have to go on standby across the country, and there is even additional power made available in France just in case the UK grid cannot cope [1].

In the case of building systems, heating, ventilating, air conditioning and refrigeration (HVAC&R) systems, chiller systems, and lighting systems operate independently of each other and frequently trigger concurrently, resulting in peak power demand. Most commercial buildings are subject to peak demand pricing which can be 200-400 times that of the nominal power rate [2]. High peak loads also lead to a higher cost of production and distribution of electricity and lower reliability. Therefore, peaks in electricity usage are inefficient and expensive for both suppliers and customers.

While there exist several different approaches to balance power consumption by load shifting and load shedding, they operate on coarse grained time scales and do not help in de-correlating energy sinks. Several approaches, e.g., [3, 4], employed mixed integer optimization to find optimal schedules for electric loads. Because mixed integer programming suffers a scalability issue when the number of integer variables is large, these approaches often worked on coarse grained time scales to reduce the computational burden. However, their run-time implementations still required powerful computer hardware and commercial optimization software.

While traditional real-time scheduling algorithms [5] may be applied to such resource sharing problems, they impose stringent constraints on the task model. Generally, real-time scheduling is restricted to tasks whose worst case execution times are fixed and known a priori [6]. While this simplifies the runtime complexity, it does not effectively capture a control system's behavior whose operation is dependent on the plant dynamics and environmental conditions.

In [7], a *Green Scheduling* approach was developed for fine-grained scheduling of energy control systems within an aggregate peak power envelop while ensuring the individual systems are maintained within the desired states. The approach was scalable but could only handle simple plant models and small uncertainties in the disturbances. [8] improved Green Scheduling to remove these limitations but the algorithm was not scalable. Moreover, [7, 8] could only bound the number of actuators activated simultaneously, rather than minimizing a more meaningful cost function such as the total electricity cost. The focus of this paper is on a scalable scheduling approach for energy control systems which has none of the above limitations. Our main contributions are:

1. We developed a model abstraction method based on the concept of simulation relations between transition systems [9, 10]. The method allows us to reduce the original multi-state multi-binary-input model to a single-state single-real-input model with input bound tracking and safety guarantees.

2. We developed a scalable hierarchical control structure based on the derived model abstraction. The run-time controller involves only two small-scale linear programs and has a low computational requirement that it is potentially suitable for embedded platforms.

3. Using simulations, we showed that while the mixed integer programming approach [3, 4] may not work beyond a trivially small-scale system, our approach can handle efficiently a much larger system with a fast sampling rate and a long control horizon.

1.1 The Energy System Scheduling Problem

We motivate our scheduling problem by an example of a room heating system. Consider an energy system consisting of n rooms and a heater in each room, where the heaters have fixed heating powers and can only be switched on and off. Let $x_i \in \mathbb{R}$ denote the air temperature of room i and $\rho_i \geqslant 0$ the power of the heater in that room. Thermal comfort specifications require that x_i should always be in a comfort range $[l_i, h_i]$. The system is subject to disturbances which are the ambient air temperature T_a and the internal heat gains Q_i in each room, e.g., from its occupants and electrical appliances. The law of conservation of energy gives us the following heat balance equation for room i:

$$C_i \dot{x}_i(t) = K_i \left(T_a(t) - x_i(t)\right) + \textstyle\sum_{j\neq i} K_{ij} \left(x_j(t) - x_i(t)\right) + \rho_i u_i(t) + Q_i(t)$$

in which $C_i > 0$ is the thermal capacity of the room, $K_i > 0$ the thermal conductance between the ambient air and the room, $K_{ij} \geq 0$ the thermal conductance between room i and room $j \neq i$, and $u_i \in \{0,1\}$ the on/off control input of heater i. Collect all the states in $x = [x_1, \dots, x_n]^T$, the comfort ranges in a bounded subset X of $\mathbb{R}^n$, the control inputs in $u = [u_1, \dots, u_n]^T$, and the disturbances in $w = [T_a, Q_1, \dots, Q_n]^T$. The dynamics of x is a linear system with control input u and disturbance input w (see (1)).

The energy demand and consumption of the system are determined based on time intervals of a given sampling time $T > 0$. In each interval $[kT, (k+1)T]$ the averaged total demand is calculated as $d_k = \frac{E_k}{T}$ where E_k is the total energy consumption during the interval. This is a practical way to determine energy demands, e.g., in electricity bills and in demand response (DR) programs. The peak demand, for example to be used in calculating the demand charge, is the maximum interval demand over a given billing period. Our goal is to schedule the subsystems to minimize the energy cost while maintaining a safe operation.

Generally we consider a continuous-time linear system Σ_b

$$(\Sigma_b) \qquad \dot{x} = Ax + Bu + Ew \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \{0,1\}^m$ are the binary control inputs that represent the schedules, and $w \in \mathbb{R}^p$ is the disturbance vector. The safety condition requires that the state $x(t)$ stays in a safe set $\mathcal{X} \subset \mathbb{R}^n$ at all time. The disturbances are constrained in a known set $w(t) \in \mathcal{W}$ for all t. Both $\mathcal{X}$ and $\mathcal{W}$ are bounded polyhedral sets.

At any time, the total demand is a linear combination of the binary control inputs: $d(t) = \rho^T u(t)$ where ρ is the vector of the individual power demand of each subsystem. During each time interval $[kT, (k+1)T]$ the total energy consumption is $E_k = \int_{kT}^{(k+1)T} d(t)\mathrm{d}t$. In this scheduling problem, we aim to schedule the subsystems, i.e., to compute the control inputs u, so that the energy cost over a finite horizon is minimized. The energy cost consists of a charge for the peak demand and possibly a charge for the energy consumption. Specifically, we want to minimize the following cost function defined over a horizon of N time intervals:

$$\text{cost} = c_d \cdot \max_{0\leqslant k\leqslant N-1} d_k + \textstyle\sum_{0\leqslant k\leqslant N-1} c_{e,k} \cdot E_k \tag{2}$$

where c_d is the fixed price for the peak demand and $c_{e,k}$ is the time-varying price for the interval energy consumption. Typically $c_d \gg c_{e,k}$ which gives customers great incentive to reduce their peak demands [2].

1.1.1 *Fundamental scalability issues*

The scheduling problem can be formulated as minimizing the cost function (2) subject to, $\forall t \geq 0, \forall k = 0, \dots, N-1$

$$\text{Equation (1)}, \quad x(t) \in \mathcal{X}, \quad u(t) \in \{0,1\}^m, \quad w(t) \in \mathcal{W}$$
$$E_k = \int_{kT}^{(k+1)T} \rho^T u(\tau)\mathrm{d}\tau, \quad d_k = \frac{E_k}{T}.$$

This optimization is intractable because $u(\cdot)$ is infinite dimensional. If we discretize Σ_b with sampling time T, the optimization becomes:

$$\begin{aligned} \underset{u_0,\dots,u_{N-1}}{\text{minimize}} \quad & c_d \cdot \max_{0\leqslant k\leqslant N-1} d_k + \textstyle\sum_{0\leqslant k\leqslant N-1} c_{e,k} \cdot E_k \qquad (3)\\ \text{subject to} \quad & x_{k+1} = A_T x_k + B_T u_k + E_T w_k \\ & d_k = \rho^T u_k, \quad E_k = T d_k \\ & x_k \in \mathcal{X}, \quad u_k \in \{0,1\}^m, \quad w_k \in \mathcal{W} \end{aligned}$$

where the constraints hold for all $k = 0, \dots, N-1$. Here, the subscript k denotes the value of a variable at time step k, and matrices A_T, B_T, and E_T are of the discrete-time dynamical model. Note that the safety condition, that x_k stays inside $\mathcal{X}$ at all k, must be robust to the unknown but bounded disturbances w, and thereby results in conservative control inputs trading off performance for safety. In practice, disturbance forecasts are usually available, for example in the forms of weather forecast and occupancy schedules, which should be exploited to obtain more accurate predictions of future states and hence less conservative control inputs and better performance. Therefore, the disturbances are modeled as $w_k = \tilde{w}_k + \delta_k$ where $\tilde{w}_k \in \mathcal{W}$ is the forecast and δ_k is the forecast uncertainty. The forecast accuracy, as a bounded set of δ_k, is assumed to be known and certainly smaller than $\mathcal{W}$. The optimization (3) is a mixed-integer linear program (MILP) and can be solved by an MILP solver.

However, except for small-scale systems with only a few control inputs and a short horizon N, the number of binary variables can be prohibitively large and the MILP is difficult to solve. For example, if Σ_b has $m = 20$ control inputs with sampling time $T = 5$min for a horizon of $N = 288$ steps in 24 hours, the MILP will have 5,760 binary variables, not to mention the continuous state and disturbance variables. Solving such large MILPs often requires powerful computers with commercial optimization solvers. In our simulation study (Section 7), with only two subsystems ($m = 2$), an 8-core MacPro with 64Gb RAM running the solver Gurobi 6.0 did not finish the MILP optimization after the time limit of 15 minutes. Moreover, due to the uncertainty caused by the disturbances, the control decisions need to be adjusted regularly by solving the optimization repeatedly at every time step (model predictive control (MPC)). Therefore, for any practical size of the system, implementing this approach can be highly demanding, if even possible, in terms of run-time hardware and software requirements. Although there are techniques in MPC to reduce the complexity, e.g., move blocking, an MILP solver is still required. Certainly the controller cannot be implemented on an embedded processor with limited processing power and memory.

1.1.2 Goal

Our goal in this paper is to develop a scalable scheduling algorithm that can potentially run on embedded processors, even for systems of moderate practical size. We consider a slightly more general problem than (3):

$$\begin{aligned}
\text{minimize}_{u(\cdot)} \quad & c_d \cdot \max_{0\leqslant k\leqslant N-1} d_k + \textstyle\sum_{0\leqslant k\leqslant N-1} c_{e,k}\cdot E_k \qquad (4)\\
\text{subject to} \quad & \dot{x}(t) = Ax(t) + Bu(t) + Ew(t) \quad \forall t \geq 0\\
& u(t) \in \{0,1\}^m \quad \forall t \geq 0\\
& x_k = x(kT) \in \mathcal{X}\\
& w(t) = w_k \in \mathcal{W} \quad \forall t \in [kT,(k+1)T)\\
& E_k = \int_{kT}^{(k+1)T} \rho^T u(\tau)\mathrm{d}\tau, \quad d_k = \frac{E_k}{T}
\end{aligned}$$

where the constraints hold for all $k = 0,\dots,N-1$. Problem (4) is more general than (3) because we consider a continuous-time schedule $u(\cdot)$, which is not necessarily constant during each time interval.

2. OVERVIEW OF THE APPROACH

In this work, we develop a hierarchical design using *model approximation* followed by *model abstraction*, combining techniques from control theory and computer science to solve the energy scheduling problem. This is complemented by a run-time implementation approach for the above scheduling problem. An overview of the overall idea and results is described in this section and in Fig. 1, while the technical details will be presented in the subsequent sections.

The complexity of the scheduling problem (4) comes mainly from the *binary* control inputs u. To alleviate this issue, we approximate the original model Σ_b by the averaged system Σ_a with state $\bar{x}$ for each time interval $t \in [kT,(k+1)T)$:

$$(\Sigma_a) \qquad \dot{\bar{x}}(t) = A\bar{x}(t) + B\eta_k + Ew(t)$$

where $\eta_k = \frac{1}{T}\int_{kT}^{(k+1)T} u(t)\mathrm{d}t$ is the *average*, also called the *utilization*, of $u(t)$ during the interval. Clearly, η_k is a vector of real numbers between 0 and 1: $\eta_k \in [0,1]^m$. Because η_k is continuous, if we use Σ_a in place of Σ_b, the complexity of the optimization is greatly reduced to that of an LP. However, to ensure the safety constraint, we need to bound the deviation of $\bar{x}(kT)$ from $x(kT)$ for all possible continuous-time binary input signals $u(t)$ that satisfy the utilization equation. In Section 3, we will derive a tight bound on the error between x and $\bar{x}$. As we will see, the error can be unbounded or become too large as k increases, leading to overly conservative control inputs to ensure safety, or even infeasibility. To keep this error as small as possible, we reset the state $\bar{x}$ of Σ_a to the measured state x of Σ_b at each kT, so that their error is reset to 0. Since the input η_k is constant in each interval, Σ_a

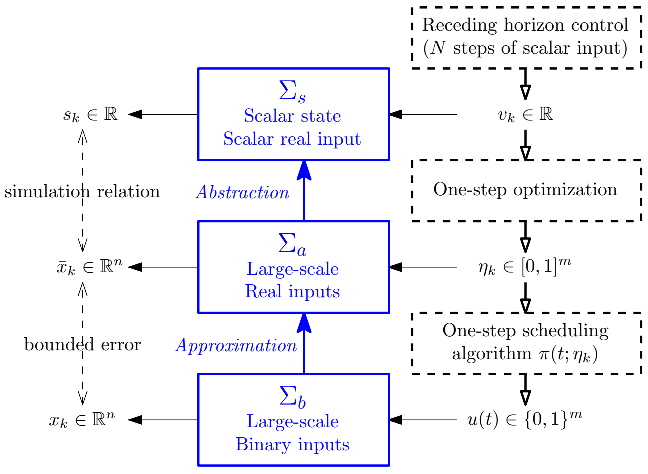

Figure 1: Overview of the scheduling approach. The solid blue boxes represent the different models at different scales resulted from the offline design process: the original large-scale binary-input model Σ_b is reduced to the single-state single-input model Σ_s. The dashed boxes represent the run-time components, which include two lightweight optimizations and one simple scheduling algorithm.

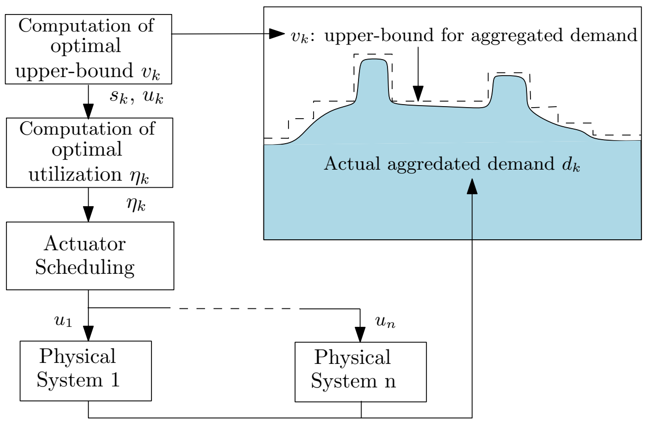
Computation of optimal upper-bound v_k
v_k: upper-bound for aggregated demand
s_k, u_k
Computation of optimal utilization η_k
Actual aggredated demand d_k
η_k
Actuator Scheduling
u_1
u_n
Physical System 1
Physical System n

Figure 2: Hierarchical control for minimizing aggregated energy cost across multiple subsystems.

can be discretized with sampling time T. With the averaged system in place, we can reformulate the optimization (4) as minimizing the cost function (2) subject to

$$\begin{aligned}
&\dot{x}(t) = Ax(t) + B\pi(t;\eta_k) + Ew_k \quad \forall t \in [kT,(k+1)T) && (5a)\\
&A_T\bar{x}_k + B_T\eta_k + E_Tw_k + e_{k+1} \in \mathcal{X} \quad \forall e_{k+1} \in \mathcal{E}(\eta_k) && (5b)\\
&\bar{x}_k = x(kT) && (5c)\\
&\eta_k \in [0,1]^m, \quad w_k \in \mathcal{W}, \quad E_k = Td_k, \quad d_k = \rho^T\eta_k && (5d)
\end{aligned}$$

where the constraints hold for all $k = 0,\dots,N-1$. In the continuous-time dynamics (5a), $\pi(t;\eta_k)$ is a scheduling algorithm that calculates the continuous-time schedule $u(t)$ during the k-th interval so that its utilization is η_k. At each instant kT the state $\bar{x}_k$ of the averaged system is reset to the actual state $x(kT)$ (constraint (5c)). Let e_{k+1} be the error between $x((k+1)T)$ and $\bar{x}_k$, which is unknown but bounded in the η_k-dependent set $\mathcal{E}(\eta_k)$ (cf. Section 3). To ensure that $x(kT) \in \mathcal{X}$, constraint (5b) robustly restricts the next state in $\mathcal{X}$ for all possible errors e_{k+1}.

In problem (5), the number of optimization variables η_k over horizon N is mN, which can be large for a large-scale system over a long horizon. For instance, with $m = 20$ and $T = 5$min over 24 hours, there are 5,760 optimization variables. Constraint (5b) involves high-dimensional dynamical equations and set operations over a long horizon, resulting in many constraints. More importantly, constraint (5a) requires execution of a continuous-time dynamical model. For these reasons, problem (5) is still difficult to solve, and its computational burden in run-time is still prohibitive for systems of practical size over a long horizon[1]. An embedded implementation of the controller is therefore unlikely realizable. To reduce the complexity even further, we employ a *model abstraction* technique based on the concept of simulation relations between transition systems [9, 10].

A transition system is a generalization of a dynamical system, whose state can transition into a new state either autonomously or under the influence of exogenous inputs. Intuitively speaking, a transition system $\mathcal{T}_2$ is said to *simulate* another transition system $\mathcal{T}_1$ if every move of $\mathcal{T}_1$ can be tracked by $\mathcal{T}_2$ with respect to a symbolic relation between their states [10]. The original definitions of simulation relations in the literature do not take into account any relation between the two systems' inputs. In Section 4, we extend this concept so that $\mathcal{T}_2$ simulates $\mathcal{T}_1$ while satisfying not only its state and input constraints but also a constraint between its input and that of $\mathcal{T}_1$.

Suppose that there exists a scalar discrete-time system:

$$\begin{aligned}(\Sigma_s)\quad & s_{k+1} = \alpha s_k + v_k + \beta^T w_k \\ & s_k \in \mathcal{S} \subseteq \mathbb{R}, v_k \in \mathcal{V} \subseteq \mathbb{R}, (s_k, v_k) \in \Omega \subseteq \mathcal{S} \times \mathcal{V}\end{aligned}$$

and a relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{S}$ such that Σ_a simulates Σ_s subject to the constraint $T\rho^T \eta_k \leqslant v_k$ between their inputs. The intuition is that s_k represents an aggregated state of all the energy subsystems at time step k, and v_k is the total energy input to the system (more precisely, an upper-bound thereof). Effectively, the scalar system Σ_s abstracts the higher-dimensional system Σ_a by aggregating all the energy states and inputs. The constraints $\mathcal{S}$, $\mathcal{V}$, and Ω are to ensure compatibility with the state and input constraints of Σ_a. We will show that any admissible trajectory of Σ_s can be tracked by an admissible trajectory of Σ_a with equal or less energy cost. Therefore, instead of solving the large-scale optimization (5), we can optimize the sequence of $\{v_0, \ldots, v_{N-1}\}$ for the scalar system Σ_s, then recover the input η_k by solving a simple optimization for each time step k. The benefit of our approach is that all the involved optimization programs have much fewer variables and much less complexity, hence it is scalable. This advantage is more significant when a receding horizon control approach is employed to compute η_k one step at a time. In that case, we only solve two small-scale programs at each step, compared to a large-scale and complex program if we solve (5) directly. Details will be presented in the subsequent sections.

Our approach is best illustrated by Fig. 1. There are two processes involved:

- In the **offline** controller design process (blue boxes and text, left side, from bottom up) we start with the original system model Σ_b and approximate it with the averaged system model Σ_a. The error bounds between the states of the two are also calculated as functions of η_k. Then Σ_a is abstracted by a scalar system Σ_s, and a simulation relation $\mathcal{R}$ between them is constructed.

- In the **online** controller execution process (dashed boxes, from top down) a receding horizon control framework is used with the abstract model Σ_s to compute an optimal sequence of aggregate energy demands. Each energy demand v_k is disaggregated into utilization values η_k for the individual control inputs by a one-step optimization. A simple one-step scheduing algorithm π schedules the subsystems so that each achieves the corresponding utilization. This hierarchical control structure is illustrated in Fig. 2.

[1]In our targeted energy control applications, the horizon is typically a day, even a week or longer.

3. APPROXIMATION BY AVERAGING

Consider the averaged system Σ_a of Σ_b for each time interval $t \in [kT, (k+1)T)$: $\dot{\bar{x}}(t) = A\bar{x}(t) + B\eta_k + Ew(t)$. Recall that the *utilization* $\eta_k = \frac{1}{T}\int_{kT}^{(k+1)T} u(\tau)\mathrm{d}\tau$ is the average value of $u(t)$ in the interval, and is a vector of real numbers in $[0, 1]$. Initially $\bar{x}_0 = \bar{x}(0) = x(0)$. Generally, as k increases, $\bar{x}_k$ may deviate further and further from $x(kT)$, which is undesirable since the state constraint $\mathcal{X}$ needs to be maintained at all time. For this reason, as mentioned earlier, we reset the value of $\bar{x}$ to the measured state x at each time instant kT, to keep the deviation within a tight bound. The deviation $e(t) = x(t) - \bar{x}(t)$ follows the dynamics $\dot{e}(t) = Ae(t) + B(u(t) - \eta_k)$ for all $t \in [kT, (k+1)T)$ and $e(kT) = 0$. At the next instant $(k+1)T$, the error is

$$e_{k+1} = e((k+1)T) = \int_{kT}^{(k+1)T} \mathrm{e}^{A((k+1)T-s)} B(u(s) - \eta_k)\mathrm{d}s.$$

A tight bound on e_{k+1} can be obtained, which is dependent on the value of η_k. In this section, we make a practical assumption that the state matrix A is diagonalizable with real eigenvalues, that is $VA = DV$ where D is a diagonal matrix of the eigenvalues of A and V is a non-singular matrix whose columns are the corresponding right eigenvectors. We then define a new state $z = Vx$, whose dynamics are

$$\dot{z} = V\dot{x} = VAx + VBu + VEw = Dz + VBu + VEw.$$

This new model is in the form of (Σ_b) with a diagonal state matrix. Let λ_i be the i-th diagonal element of D, for $i = 1, \ldots, n$. The following theorem provides bounds on e_{k+1}.

THEOREM 1. *Let b_i be the i^{th} row of VB, $\bar{b}_i \geqslant 0$ the sum of all positive elements of b_i, and $\underline{b}_i \leqslant 0$ the sum of all negative elements of b_i. For each $i = 1, \ldots, n$, define*

$$\xi_i = \begin{cases} 0 & \text{if } \lambda_i = 0 \\ \frac{1}{\lambda_i}(\mathrm{e}^{\lambda_i T} - 1) - T & \text{if } \lambda_i > 0 \\ \frac{1}{\lambda_i}(\mathrm{e}^{\lambda_i T} - 1) - T\mathrm{e}^{\lambda_i T} & \text{if } \lambda_i < 0 \end{cases}$$

and $\bar{\varepsilon}_i = \xi_i \bar{b}_i$, $\underline{\varepsilon}_i = \xi_i \underline{b}_i$. Define $\tilde{B} = V^{-1}\Xi B$ where Ξ is the diagonal matrix of all ξ_i. Then the error e_{k+1} is bounded by

$$\underline{\varepsilon} \leqslant V(e_{k+1} + \tilde{B}\eta_k) \leqslant \bar{\varepsilon} \quad \textit{(element-wise)}$$

where $\bar{\varepsilon}$ and $\underline{\varepsilon}$ are the vectors of all $\bar{\varepsilon}_i$ and $\underline{\varepsilon}_i$ respectively. Equivalently, $e_{k+1} = V^{-1}\varepsilon - \tilde{B}\eta_k$ for some ε bounded element-wise by $\underline{\varepsilon} \leqslant \varepsilon \leqslant \bar{\varepsilon}$.

We note that the utilization η_k, i.e., the control input of Σ_a, enters affinely in the bounds on e_{k+1}.

REMARK 1. *We can still use our approach when A is not diagonalizable, however the error bounds obtained in Theorem 1 must be generalized. For instance, we can use the Jordan normal form to transform A into a block diagonal matrix, and obtain bounds in a way similar to that in Theorem 1. The made assumption simplifies the presentation of our results but does not limit the practicality of our approach.*

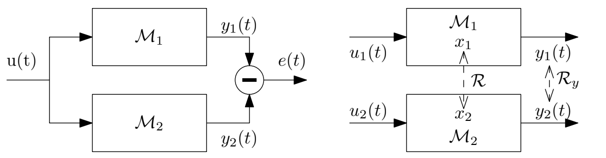

Figure 3: Model abstraction (right) is different from model order reduction (left): $\mathcal{M}_1$ and $\mathcal{M}_2$ do not share their inputs and outputs, but maintain a relation between their states along their evolutions.

4. MODEL ABSTRACTRION WITH SIMULATION RELATIONS

As we discussed in Section 2, to further reduce the optimization (5), we employ the notion of model abstraction and simulation relations. We first distinguish model abstraction and model order reduction. Model order reduction is a common technique in control theory to reduce the order of a large-scale system so that it becomes manageable. The reduced model, however, still has the same input and output as the original model. Under the same input signal, it should produce an output signal which approximates that of the original model, within some error bound. The left diagram in Fig. 3 illustrates this concept, where $\mathcal{M}_2$ is a reduced-order model of $\mathcal{M}_1$. On the other hand, model abstraction derives a new model, usually of a lower order, with completely different input and output than the original model. However, there exists a relation between their states that can be maintained along their evolutions. In other words, one model can track the behavior of the other under this relation. The right diagram in Fig. 3 depicts two models with a symbolic state relation $\mathcal{R}$. If for any admissible input $u_2(\cdot)$ to $\mathcal{M}_2$ there exists an admissible input $u_1(\cdot)$ to $\mathcal{M}_1$ so that the state relation is always maintained along their traces, we have a model abstraction. Furthermore, we may require that a symbolic relation $\mathcal{R}_y$ between their outputs (observations) is also maintained. A nice property of model abstraction is that we can design a controller for $\mathcal{M}_1$ by first designing a supposedly simpler controller for $\mathcal{M}_2$ and then refining it for $\mathcal{M}_1$ using the symbolic state relation.

4.1 Simulation Relations

In this section we review the concept of simulation relations between transition systems. The readers are referred to [9, 10] for more thorough treatments of the subject.

Transition systems allow us to unify the modeling of discrete and continuous, deterministic and non-deterministic dynamical systems. As we do not concern with the observable outputs in this paper, we remove the observation aspect from the definition of transition systems in [10]:

Definition 1 (Transition System, [10]). *A transition system is a tuple $T = (Q, \Sigma, \to, Q^0)$ that consists of*

- *a possibly infinite set Q of states;*
- *a possibly infinite set Σ of labels;*
- *a transition relation $\to \subseteq Q \times \Sigma \times Q$;*
- *a possibly infinite set $Q^0 \subseteq Q$ of initial states.*

A transition from state q to state q^+ under the label σ, i.e., $(q, \sigma, q^+) \in \to$, is denoted $q \xrightarrow{\sigma} q^+$. We assume that the transition systems are *nonblocking*, meaning that for all $q \in Q$, there exists at least one transition starting from q. If for any $q \in Q$ and any $\sigma \in \Sigma$, there is at most one transition $q \xrightarrow{\sigma} q^+$ of T and Q^0 is a singleton, then T is called *deterministic*. Otherwise, it is called *nondeterministic*.

Example 1. *Transition systems generalize dynamical systems. As an example, consider a nonlinear discrete-time dynamical system $x^+ = f(x,u)$ where $x \in \mathbb{R}^n$ is the state and $u \in \mathbb{R}^m$ is the input, with initial state x_0. This system can be represented by a deterministic transition system T with $Q = \mathbb{R}^n$, $\Sigma = \mathbb{R}^m$, $\to = \{(x,u,x^+) | x \in Q, u \in \Sigma, x^+ = f(x,u) \in Q\}$, and $Q^0 = \{x_0\}$. If the system is subject to disturbance: $x^+ = f(x,u,w)$ where $w \in \mathcal{W} \subseteq \mathbb{R}^p$ is the disturbance, then T becomes nondeterministic where $\to = \{(x,u,x^+) | x \in Q, u \in \Sigma, w \in \mathcal{W}, x^+ = f(x,u,w) \in Q\}$.*

A simulation relation between two transition systems $T_1 = (Q_1, \Sigma_1, \to_1, Q_1^0)$ and $T_2 = (Q_2, \Sigma_2, \to_2, Q_2^0)$ is a stronger notion of system refinement, which allows one system to "track" the other system while maintaining a certain relation between their states.

Definition 2 (Simulation). *A relation $\mathcal{R} \subseteq Q_1 \times Q_2$ is called a simulation relation of T_1 by T_2 if and only if for all $(q_1, q_2) \in \mathcal{R}$ and for all transitions $q_1 \xrightarrow{\sigma_1}_1 q_1^+$, there exists a transition $q_2 \xrightarrow{\sigma_2}_2 q_2^+$ such that $(q_1^+, q_2^+) \in \mathcal{R}$.*

Note that in [10], T_1 and T_2 have the same label sets and $\sigma_1 = \sigma_2 = \sigma$. However in Definition 2 we relax this requirement, i.e., Σ_1 and Σ_2, hence σ_1 and σ_2, can be different. T_2 is said to simulate T_1, denoted $T_1 \preceq T_2$, if there exists a simulation relation $\mathcal{R}$ of T_1 by T_2 such that for all $q_1 \in Q_1^0$, there exists $q_2 \in Q_2^0$ such that $(q_1, q_2) \in \mathcal{R}$. Intuitively, if $T_1 \preceq T_2$ then every state trajectory of T_1 can be "tracked" by T_2 with respect to the state relation $\mathcal{R}$. Simulation relations and their variants are a powerful tool for safety verification and hierarchical controller design [10, 11, 12, 13].

4.2 Input-Constrained Simulation Relations

Conventionally, simulation relations as defined in Definition 2 do not explicitly take into account disturbances. Moreover, there is no relation between the inputs (or labels) to the two systems as required for the total demand bound in our problem. Therefore, this section extends the concept of simulation relations to account for these ingredients.

Remark 2. *For nondeterministic systems, the simulation relation as defined in Definition 2 is not robust to the nondeterminism. In particular, it is implicitly assumed that the transition from any q_1 under any σ_1, although nondeterministic, is detectable so that an appropriate transition $q_2 \xrightarrow{\sigma_2} q_2^+$ can be selected. T_2 is not nondeterministic anymore because here $q_2 \xrightarrow{\sigma_2} q_2^+$ can be chosen to satisfy $(q_1^+, q_2^+) \in \mathcal{R}$. Furthermore, it assumes no correlation between the nondeterminism of the two systems, which in reality may exist, e.g., if they are subject to the same disturbances in different ways. Our extended definition accounts for both robustness to disturbances and nondeterminism correlation.*

Remark 3. *A relation between the labels of T_1 and T_2 generalizes the simulation relation definition in [10], where σ_1 and σ_2 are related by $\sigma_1 = \sigma_2$ (provided that $\Sigma_1 = \Sigma_2$).*

We generalize the label σ as a pair $\sigma = (\upsilon, \delta)$ of a control label $\upsilon \in \mathcal{U}$ and a disturbance label $\delta \in \mathcal{D}$. Here, $\mathcal{U}$ and $\mathcal{D}$ are respectively the (possibly infinite) sets of control labels and disturbance labels. The definition of the transition relation is modified accordingly as $\to \subseteq Q \times \mathcal{U} \times \mathcal{D} \times Q$, and a transition $(q, \upsilon, \delta, q^+) \in \to$ is denoted $q \xrightarrow{(\upsilon,\delta)} q^+$. Given two transition systems T_1 and T_2, we impose a constraint between their control labels υ_1 and υ_2 as a relation

$\mathcal{R}_u \subseteq \mathcal{U}_1 \times \mathcal{U}_2$, and a constraint between their disturbance labels $\mathcal{R}_d \subseteq \mathcal{D}_1 \times \mathcal{D}_2$. Note that the constraint $\mathcal{R}_u$ is desirable but not required for the execution of the transition systems, i.e., an admissible execution of the transition systems may violate the constraint. The relation $\mathcal{R}_d$ takes effect whenever T_1 and T_2 are executed simultaneously in the same environment. For system T_i, the set of all admissible successor states from a state q_i under label $\sigma_i = (\upsilon_i, \delta_i)$ is denoted by $\mathrm{succ}_i(q_i, \upsilon_i, \delta_i) = \{q_i^+ \mid (q_i, \upsilon_i, \delta_i, q_i^+) \in \rightarrow_i\}$.

Definition 3 (Input-constrained Simulation). *A relation $\mathcal{R} \subseteq Q_1^S \times Q_2^S$ is an input-constrained simulation relation of T_1 by T_2 if and only if for all $(q_1, q_2) \in \mathcal{R}$ and all $(\upsilon_1, \delta_1) \in \mathcal{U}_1 \times \mathcal{D}_1$ such that $\mathrm{succ}_1(q_1, \upsilon_1, \delta_1) \neq \emptyset$, there exists $\upsilon_2 \in \mathcal{U}_2$ such that*

1. *$(\upsilon_1, \upsilon_2) \in \mathcal{R}_u$; and*
2. *for all $\delta_2 \in \mathcal{D}_2$ such that $(\delta_1, \delta_2) \in \mathcal{R}_d$, $\mathrm{succ}_2(q_2, \upsilon_2, \delta_2) \neq \emptyset$ and $(q_1^+, q_2^+) \in \mathcal{R}$ $\forall (q_1^+, q_2^+) \in \mathrm{succ}_1(q_1, \upsilon_1, \delta_1) \times \mathrm{succ}_2(q_2, \upsilon_2, \delta_2)$.*

The existence of an input-constrained simulation relation $\mathcal{R}$ of T_1 by T_2 allows T_2 to track any state sequence of T_1 with respect to $\mathcal{R}$, as long as their initial states are in $\mathcal{R}$. We define a trajectory of T_i as a (potentially infinite) sequence of admissible states, labels, and transitions: $q_i^0 \xrightarrow{\upsilon_i^0, \delta_i^0}_i q_i^1 \xrightarrow{\upsilon_i^1, \delta_i^1}_i q_i^2 \cdots$. The next lemma follows directly from Definition 3. It essentially says that if an input-constrained simulation relation $\mathcal{R}$ of T_1 by T_2 exists then T_2 can be controlled to track any admissible trajectory of T_1, with respect to $\mathcal{R}$, while keeping the input constraint $\mathcal{R}_u$.

Lemma 1. *Suppose $\mathcal{R}$ is an input-constrained simulation relation of T_1 by T_2, and $(q_1^0, q_2^0) \in \mathcal{R}$. Let $\kappa_2 : Q_1 \times \mathcal{U}_1 \times \mathcal{D}_1 \times Q_2 \to \mathcal{U}_2$ be any feedback law such that for all admissible quadruples $(q_1, \upsilon_1, \delta_1, q_2) \in Q_1 \times \mathcal{U}_1 \times \mathcal{D}_1 \times Q_2$, $\upsilon_2 = \kappa_2(q_1, \upsilon_1, \delta_1, q_2)$ satisfies the conditions in Definition 3. Such κ_2 always exists. Then for any trajectory of T_1, any corresponding trajectory of T_2 with $\upsilon_2^k = \kappa_2(q_1^k, \upsilon_1^k, \delta_1^k, q_2^k)$ satisfies $(q_1^k, q_2^k) \in \mathcal{R}$ and $(\upsilon_1^k, \upsilon_2^k) \in \mathcal{R}_u$ for all k.*

5. MODEL ABSTRACTION FOR ENERGY SYSTEM SCHEDULING

Going back to the energy system scheduling problem, we apply the framework of input-constrained simulation relations to abstract the high-dimensional model Σ_a by a low-dimensional model Σ_s. We then derive a feedback control law that allows Σ_a to track the state of Σ_s while maintaining their simulation relation, all their state and input constraints, as well as the input upper-bounds set by Σ_s. In particular, we consider the scalar discrete-time system Σ_s:

$$(\Sigma_s) \quad s_{k+1} = \alpha s_k + v_k + \beta^T w_k \tag{6}$$
$$s_k \in \mathcal{S} \subseteq \mathbb{R}, v_k \in \mathcal{V} \subseteq \mathbb{R}, (s_k, v_k) \in \Omega \subseteq \mathcal{S} \times \mathcal{V}$$

The intuition is that s_k represents an aggregated state of all the subsystems at time step k and v_k specifies an upper-bound of the total energy demand. For example, consider the room heating system in Section 1.1. At time k, s_k can represent the total enthalpy of all rooms, while v_k bounds the total heating energy of all heaters during that interval. Because the system is subject to heat loss and disturbances, the dynamics of s have the form of (6).

The abstraction of Σ_a, i.e., the input-constrained simulation relation $\mathcal{R}$, depends on how the control input v_k is computed and how the disturbances are represented in Σ_s. In practice, forecasts of the disturbances, denoted $\tilde{w}_k$, with known bounded accuracies are often available and used in computing the control inputs for both Σ_s and Σ_a. The forecast accuracies are represented by a vector $\zeta \geq 0$ in $\mathbb{R}^p$ such that $-\zeta \leqslant w_k - \tilde{w}_k \leqslant \zeta$ element-wise for all k.

We consider a feedback control approach in which, at each time step k, v_k is decided using the disturbance forecast $\tilde{w}_k$ and the actual state s_k, computed from the observed actual state $\bar{x}_k$. The control input η_k is calculated from the actual states s_k and $\bar{x}_k$, and v_k and $\tilde{w}_k$. Hence, both the control laws for Σ_a and Σ_s are feedback.

We derive the abstraction Σ_s of Σ_a in three steps:

1. Σ_s and Σ_a are formulated as transition systems;
2. Assuming a certain form of $\mathcal{R}$, we derive conditions for $\mathcal{R}$ to be an input-constrained simulation relation;
3. Based on the conditions, we develop algorithms to compute the parameters of Σ_s, its joint constraint Ω, and the simulation relation $\mathcal{R}$.

5.1 Transition Systems

The evolution of Σ_s is influenced by the actual disturbances, which are unknown but within known accuracies from their forecasts. To model this phenomenon, we will include the disturbance forecasts in the control labels of both transition systems, while their disturbance labels are the errors between the forecasts and the actual disturbances. Specifically, we define T_1 of Σ_s as follows:

- State $q_1 \equiv s$ and state set $Q_1 = \mathcal{S}$.
- Control label $\upsilon_1 \equiv (v, \tilde{w}_1)$ with $\mathcal{U}_1 = \mathcal{V} \times \mathcal{W}$.
- Disturbance label δ_1 is the disturbance forecast error $\delta_1 = w - \tilde{w}_1$; $\mathcal{D}_1 = \{\delta_1 \in \mathbb{R}^p \mid -\zeta \leqslant \delta_1 \leqslant \zeta\}$.
- Transitions $\rightarrow_1 = \{(s, v, \tilde{w}_1, \delta_1, s^+) \mid (s, v) \in \Omega, \tilde{w}_1 \in \mathcal{W}, \delta_1 \in \mathcal{D}_1, \tilde{w}_1 + \delta_1 \in \mathcal{W}, s^+ = \alpha s + v + \beta^T(\tilde{w}_1 + \delta_1) \in \mathcal{S}\}$.

The transition system T_2 of Σ_a is defined as:

- State $q_2 \equiv \bar{x}$; $Q_2 = \mathcal{X}$.
- Control label $\upsilon_2 \equiv (\eta, \tilde{w}_2)$ with $\mathcal{U}_2 = [0,1]^m \times \mathcal{W}$.
- Disturbance label δ_2 is also the forecast error; $\mathcal{D}_2 = \{\delta_2 \in \mathbb{R}^p \mid -\zeta \leqslant \delta_2 \leqslant \zeta\}$.
- Transition set: we again model the state reset as a nondeterministic but bounded perturbation of $\bar{x}$. Define $\rightarrow_2 = \{(\bar{x}, \eta, \tilde{w}_2, \delta_2, \bar{x}^+) \mid \bar{x} \in \mathcal{X}, \eta \in \mathcal{U}_2, \tilde{w}_2 \in \mathcal{W}, \delta_2 \in \mathcal{D}_2, \tilde{w}_2 + \delta_2 \in \mathcal{W}, \bar{x}^+ = A_T\bar{x} + B_T\eta + E_T(\tilde{w}_2 + \delta_2) + e \in \mathcal{X}, \underline{\varepsilon} \leqslant V(e + \tilde{B}\eta) \leqslant \bar{\varepsilon}\}$.

The joint input constraint $\mathcal{R}_u$ must specify that their disturbance forecasts are the same, hence $\mathcal{R}_u = \{((v, \tilde{w}_1), (\eta, \tilde{w}_2)) \in \mathcal{U}_1 \times \mathcal{U}_2 \mid T\rho^T\eta \leq v, \tilde{w}_1 = \tilde{w}_2\}$. Finally, because T_1 and T_2 are subject to the same actual disturbances, we define the disturbance relation $\mathcal{R}_d = \{(\delta_1, \delta_2) \in \mathcal{D}_1 \times \mathcal{D}_2 \mid \delta_1 = \delta_2\}$.

5.2 Conditions for $\mathcal{R}$

To derive $\mathcal{R}$, we need to be able to certify if any given relation $\mathcal{R}$ of s and $\bar{x}$ is an input-constrained simulation relation of T_1 by T_2. The following theorem provides such a necessary and sufficient condition for $\mathcal{R}$. Here, the product between a matrix $M \in \mathbb{R}^{m \times n}$ and a set $\mathcal{P} \subseteq \mathbb{R}^n$ is defined as the image of $\mathcal{P}$ under the linear map M, i.e., $M\mathcal{P} = \{y \in$

$\mathbb{R}^m \mid y = Mx, x \in \mathcal{P}\}$. $\mathbb{I}$ denotes the identity matrix and $\mathbf{0}$ is a zero vector or matrix, whose dimensions are often omitted when they are obvious from the context. The symbol $\ominus$ denotes the Pontryagin difference between two sets.

THEOREM 2. *A set $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{X}$ is an input-constrained simulation relation of T_1 by T_2 if and only if*

$$M_l \mathcal{P}_l \subseteq M_r \mathcal{P}_r \tag{7}$$

where

$$M_l = \begin{bmatrix} 0 & \mathbf{0} & 1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \alpha & \mathbf{0} & 1 & \beta^T & \beta^T & \mathbf{0} \\ 0 & A_T & 0 & E_T & E_T & \mathbb{I} \end{bmatrix}, \quad M_r = \begin{bmatrix} 0 & \mathbf{0} & 1 & \mathbf{0} \\ 1 & \mathbf{0} & 0 & \mathbf{0} \\ 0 & \mathbb{I} & 0 & \tilde{B} - B_T \end{bmatrix}$$

and

$$\mathcal{P}_l = \big\{(s,\bar{x},v,\delta,\tilde{w},\varepsilon) \big| (s,\bar{x}) \in \mathcal{R}, (s,v) \in \Omega, -\zeta \leqslant \delta \leqslant \zeta, \tilde{w} \in \mathcal{W},$$
$$\tilde{w} + \delta \in \mathcal{W}, \alpha s + v + \beta^T \tilde{w} \in \mathcal{S} \ominus \beta^T \mathcal{D}_1, \underline{\varepsilon} \leqslant V\varepsilon \leqslant \bar{\varepsilon}\big\}$$
$$\mathcal{P}_r = \big\{(s^+,\bar{x}^+,\bar{v},\eta) \big| \bar{x}^+ \in \mathcal{X}, (s^+,\bar{x}^+) \in \mathcal{R}, \eta \in [0,1]^m, T\rho^T \eta \leqslant \bar{v}\big\}$$

Although Theorem 2 is useful for verifying whether a given $\mathcal{R}$ is an input-constrained simulation relation, it does not allow us to directly compute $\mathcal{R}$ since $\mathcal{R}$ appears on both sides of (7). Furthermore, computing the image of a set under a linear map and checking for a set inclusion are generally difficult. Therefore, we will impose a certain structure on the relation $\mathcal{R}$ that results in polyhedral sets $\mathcal{P}_l$ and $\mathcal{P}_r$. We will also derive a simpler condition for $\mathcal{R}$, which can be verified by checking the feasibility of a linear program.

As mentioned earlier, s represents an aggregated state of the subsystems. Thus, we can choose s to be approximately a linear combination of all the states of the subsystems. Let $s \approx c^T \bar{x}$, where c is a constant vector in $\mathbb{R}^n$. The set $\mathcal{R}$ is chosen to represent this relationship between s and $\bar{x}$ as

$$\mathcal{R} = \{(s,\bar{x}) \in \mathcal{S} \times \mathcal{X} \mid |c^T \bar{x} - s| \leqslant \gamma\}, \tag{8}$$

where $\gamma \geqslant 0$ is a design parameter. Intuitively, in an energy system, the set of $\bar{x} \in \mathcal{X}$ such that $(s,\bar{x}) \in \mathcal{R}$, for any $s \in \mathcal{S}$, is the set of all states that have roughly the same aggregated energy level s (up to an error of γ).

We have assumed in Section 1.1 that $\mathcal{X}$ and $\mathcal{W}$ are polyhedra. Suppose also that Ω is a polyhedron. Let their hyperplane representations be

$$\mathcal{X} = \{x \in \mathbb{R}^n \mid H_x x \leqslant k_x\}, \qquad \mathcal{W} = \{w \in \mathbb{R}^p \mid H_w x \leqslant k_w\}$$
$$\Omega = \{(s,v) \in \mathcal{S} \times \mathcal{V} \mid H_\Omega^s s + H_\Omega^v v \leqslant k_\Omega\}$$

where H_x, k_x, H_w, k_w, H_Ω^s, H_Ω^v, k_Ω are matrices and vectors of appropriate dimensions. Because s and v are scalars, we can bound them as $\underline{s} \leqslant s \leqslant \bar{s}$ and $\underline{v} \leqslant v \leqslant \bar{v}$, where $\underline{s}$, $\bar{s}$, $\underline{v}$, $\bar{v}$ are respectively the lower- and upper-bounds of s and v. Note that Ω is a subset of $\mathcal{S} \times \mathcal{V} = [\underline{s}, \bar{s}] \times [\underline{v}, \bar{v}]$. With these assumptions, $\mathcal{P}_l$ and $\mathcal{P}_r$ become polyhedral sets. We can now state a sufficient condition for $\mathcal{R}$ in the next theorem.

THEOREM 3. *A set $\mathcal{R}$ defined in (8) is an input-constrained simulation relation of T_1 by T_2 if there exist a real matrix Z of non-negative elements, a real matrix Q, and a real vector q such that*

$$M_r Q = \mathbb{I},\ M_r q = \mathbf{0},\ Z H_l = H_r Q M_l,\ Z k_l \leqslant k_r - H_r q \tag{9}$$

in which

$$H_r = \begin{bmatrix} & H_x & & \\ -1 & c^T & & \\ 1 & -c^T & & \\ & & -1 & T\rho^T \\ & & & \mathbb{I} \\ & & & -\mathbb{I} \end{bmatrix}, \quad k_r = \begin{bmatrix} k_x \\ \gamma \\ \gamma \\ 0 \\ \mathbf{1} \\ \mathbf{0} \end{bmatrix}$$

$$H_l = \begin{bmatrix} & H_x & & & & \\ -1 & c^T & & & & \\ 1 & -c^T & & & & \\ H_\Omega^s & & H_\Omega^v & & & \\ & & & & H_w & \\ & & & H_w & H_w & \\ & & & \mathbb{I} & & \\ & & & -\mathbb{I} & & \\ \alpha & & 1 & & \beta^T & \\ -\alpha & & -1 & & -\beta^T & \\ & & & & & V \\ & & & & & -V \end{bmatrix}, k_l = \begin{bmatrix} k_x \\ \gamma \\ \gamma \\ k_\Omega \\ k_w \\ k_w \\ \zeta \\ \zeta \\ \bar{s} - \max_{\delta \in [-\zeta,\zeta]} \beta^T \delta \\ -\underline{s} + \min_{\delta \in [-\zeta,\zeta]} \beta^T \delta \\ \bar{\varepsilon} \\ -\underline{\varepsilon} \end{bmatrix}.$$

Here, the blank spaces are zero blocks and $\mathbf{1}$ denotes a vector of 1's. Although Theorem 3 is only sufficient, it reduces the condition to checking the feasibility of the linear program (9), which can be solved rather efficiently by any LP solver such as Gurobi, MOSEK, CPLEX, CLP.

The design parameters (α, β, c, γ, H_Ω^s, H_Ω^v, k_Ω) are multiplied by the elements of Z in (9). Consequently, finding them by solving (9) leads to a bilinear program, which is intractable to solve for any significant size. We will discuss a way to compute these parameters in the next section.

5.3 Computation of Design Parameters

As the design parameters (α, β, c, γ, H_Ω^s, H_Ω^v, k_Ω) enter the sufficient condition (Eq. (9)) in bilinear terms, it is intractable to compute them (optimally) using this condition. In this section, we present a method to design these parameters in a suboptimal but tractable way.

5.3.1 *Computing* α, β, c

We compute α, β, c based on the intuition that s is the aggregated state of all the subsystems: $s \approx c^T \bar{x}$. From the discrete-time dynamics of $\bar{x}$ we have

$$s_{k+1} \approx c^T \bar{x}_{k+1} = c^T A_T \bar{x}_k + c^T B_T \eta_k + c^T E_T w_k$$

while at the same time

$$s_{k+1} = \alpha s_k + v_k + \beta^T w_k \approx \alpha c^T \bar{x}_k + T\rho^T \eta_k + \beta^T w_k.$$

Matching the two equations, we aim to find α and c so that $c^T B_T = T\rho^T$ while minimizing $\|c^T A_T - \alpha c^T\|$, where $\|\cdot\|$ denotes some vector norm. If B_T is full row rank, c can be uniquely computed by solving a linear equation, and the minimization becomes a simple unconstrained program. If B_T is not full row rank, c and α can be found by solving the above nonlinear optimization, which can be made unconstrained by rewriting c in terms of the basis vectors of the kernel of B_T^T. We then calculate $\beta = E_T^T c$.

5.3.2 *Computing* γ *and* Ω

Once α, β, and c are chosen, we can compute the parameter γ of the relation $\mathcal{R}$ and the constraint set Ω of (s, v). We first choose the ranges $[\underline{s}, \bar{s}]$ and $[\underline{v}, \bar{v}]$ of s and v that satisfy

$$\underline{s} \geqslant \min\nolimits_{x \in \mathcal{X}} c^T x, \quad \bar{s} \leqslant \max\nolimits_{x \in \mathcal{X}} c^T x$$
$$\underline{v} \geqslant \min\nolimits_{\mathbf{0} \leqslant \eta \leqslant \mathbf{1}} T\rho^T \eta, \quad \bar{v} \leqslant \max\nolimits_{\mathbf{0} \leqslant \eta \leqslant \mathbf{1}} T\rho^T \eta$$

based on the relationships between s and $\bar{x}$, v and η. Because X is bounded, $\underline{s}$ and $\bar{s}$ are finite; and so are $\underline{v}$ and $\bar{v}$.

Choosing γ and Ω is difficult because γ appears in both k_l and k_r, and Ω is a set. It is unnecessary for γ to be larger than $\bar{s} - \underline{s}$ due to $|c^T \bar{x} - s| \leqslant \gamma$. Therefore we can select any value between 0 and $\bar{s} - \underline{s}$ for γ, with the apparent intuition that as γ gets smaller, the constraint between s and $\bar{x}$ becomes stricter, keeping $(s, \bar{x})$ in $\mathcal{R}$ is more difficult, hence $\mathcal{R}$ is less likely an input-constrained simulation relation. For Ω, which is a subset of $[\underline{s}, \bar{s}] \times [\underline{v}, \bar{v}]$, the following result allows us to construct Ω from smaller sets.

LEMMA 2. *Suppose that all the parameters are fixed except for Ω. Let Ω_1 and Ω_2 are two subsets of $[\underline{s}, \bar{s}] \times [\underline{v}, \bar{v}]$ and suppose that they satisfy the sufficient condition in Theorem 3. Then their union $\Omega_1 \cup \Omega_2$ satisfies the necessary and sufficient condition in Theorem 2.*

This lemma can be proved by noting that Ω_1 and Ω_2 satisfy Theorem 2, and Ω appears only in the left-hand side of Eq. (7). Using this result, we can check the sufficient condition in Theorem 3 for small sets, then construct the final Ω set from their union. This algorithm is outlined below.

1. Grid the finite intervals of s and v as $\underline{s} = s^0 < s^1 < \cdots < s^{l_s} = \bar{s}$ and $\underline{v} = v^0 < v^1 < \cdots < v^{l_v} = \bar{v}$, i.e., grid the box $[\underline{s}, \bar{s}] \times [\underline{v}, \bar{v}]$ into $l_s l_v$ small cells.

2. For each cell $\Omega_{i,j} = [s^i, s^{i+1}] \times [v^j, v^{j+1}]$, for $0 \leqslant i \leqslant l_s - 1$ and $0 \leqslant j \leqslant l_v - 1$, check if it satisfies Theorem 3.

3. Construct Ω as the union of all cells $\Omega_{i,j}$ that satisfy Theorem 3. If the projection of Ω onto s covers $[\underline{s}, \bar{s}]$ then take Ω. This is to ensure that we can always find an admissible control input v for any valid value of s. We can simplify Ω by a convex polyhedron contained in the union, from which H^s_Ω, H^v_Ω and k_Ω are extracted.

REMARK 4. *The constructed set Ω may not satisfy Theorem 3 because the condition is only sufficient. However, Ω is valid because it satisfies the stronger necessary and sufficient condition in Theorem 2, due to Lemma 2.*

6. SCALABLE SCHEDULING: DESIGN AND ALGORITHMS

This section summarizes the results developed in the previous sections and discusses the design process and the run-time implementation of our approach.

6.1 Design Process of Scalable Scheduling

Given the scheduling problem (4):

1. Discretize Σ_b (Eq. (1)) with sampling time T.

2. Compute $\underline{\varepsilon}$, $\bar{\varepsilon}$, and $\tilde{B}$ using Theorem 1.

3. Compute α, β, c of the scalar model Σ_s (Section 5.3.1).

4. Compute parameters γ and Ω of the relation $\mathcal{R}$ (Section 5.3.2). The construction of Ω requires checking the feasibility of the LP (9) for many times. The computational burden for this step can be high. However, the design process is performed offline, hence the run-time performance of our approach is not affected.

The design is complete when all parameters are obtained.

6.2 Run-time Implementation

The run-time implementation consists of three hierarchical components, as illustrated in Fig. 1.

Top level – optimal upperbounds

At the top level is an optimization to compute optimal inputs v_k for Σ_s, which are also the upper-bounds of the aggregated demand of the system. The optimization is solved repeatedly at each time step, taking into account the current state of the system and the robustness to errors in disturbance forecasts. Let the current step be $i \leqslant N - 1$; the optimization is formulated below.

$$\begin{aligned}
\underset{v_i,\ldots,v_{N-1}}{\text{minimize}} \quad & c_d \cdot \max_{i\leqslant k\leqslant N-1} v_k/T + \textstyle\sum_{i\leqslant k\leqslant N-1} c_{e,k} \cdot v_k \\
\text{subject to} \quad & s_{k+1} = \alpha s_k + v_k + \beta^T \tilde{w}_k \\
& -\gamma \leqslant c^T x_i - s_i \leqslant \gamma \\
& \underline{v} \leqslant v_k \leqslant \bar{v},\ \underline{s} \leqslant s_k \leqslant \bar{s},\ H^s_\Omega s_k + H^v_\Omega v_k \leqslant K_\Omega \\
& \underline{s} - \min_{-\zeta\leqslant\delta\leqslant\zeta} \beta^T \delta \leqslant s_{i+1} \leqslant \bar{s} - \max_{-\zeta\leqslant\delta\leqslant\zeta} \beta^T \delta
\end{aligned}$$

for all $k = i, \ldots, N - 1$. Only the first step of the solution, i.e., s_i and v_i, will be used in the lower levels because at the next time step, the optimization will be repeated.

The above optimization is an LP with the scalar dynamics Σ_s and at most N variables $v_0, \ldots, v_{N-1}$, which can be solved very efficiently [14], even on embedded computers using optimization libraries or code-generation tools such as FORCES, ECOS, and FiOrdOs[2].

6.2.1 *Middle level – optimal utilizations*

Given an optimal (s_k, v_k) from the top level, the middle level computes an optimal utilization η_k that satisfies the conditions of the input-constrained simulation relation $\mathcal{R}$. By Lemma 1, such η_k always exists and maintains the relation $\mathcal{R}$ between the states of Σ_s and Σ_a. At each step k, we minimize a cost function $J(\eta_k)$ subject to the constraints:

$$\begin{gathered}
\eta_k \in [0,1]^m,\ T\rho^T \eta_k \leqslant v_k \\
A_T x_k + (B_T - \tilde{B})\eta_k + E_T \tilde{w}_k \in X \ominus E_T \mathcal{Z} \ominus \mathcal{E} \\
-\gamma - \min\nolimits_{\varepsilon\in\mathcal{E}} c^T \varepsilon \leqslant c^T (A_T x_k + (B_T - \tilde{B})\eta_k + E_T \tilde{w}_k) - \\
(\alpha s_k + v_k + \beta^T \tilde{w}_k) \leqslant \gamma - \max\nolimits_{\varepsilon\in\mathcal{E}} c^T \varepsilon
\end{gathered}$$

where $\mathcal{E} = \{\varepsilon \,|\, \underline{\varepsilon} \leqslant V\varepsilon \leqslant \bar{\varepsilon}\}$. The last two constraints are to ensure that the next state is robustly safe and robustly maintains the relation $\mathcal{R}$. The cost function $J(\eta_k)$ can be chosen to minimize the energy demand:

$$J(\eta_k) = \rho^T \eta_k$$

or to minimize the difference with a given (ideal) state $x^\star$:

$$J(\eta_k) = \|A_T x_k + B_T \eta_k + E_T \tilde{w}_k - x^\star\|_2.$$

Note that **the middle-level optimization is solved for a single time step** and has only m variables (η_k) with linear constraints. Therefore it can be solved efficiently if J is convex in η_k [14]. Because the constraint $T\rho^T \eta_k \leqslant v_k$ is always maintained, the actual energy cost will never exceed the optimal cost computed by the top-level optimization.

6.2.2 *Low level scheduling*

The solution η_k of the middle level is passed to the last component at the lowest level, which schedules the binary control inputs u to achieve the desired utilization η_k. In this paper, we use a simple and fast scheduling algorithm, which switches each control input on for a fraction of the duration T equal to its utilization value. Better scheduling algorithms for this level is a topic for future research.

7. CASE STUDY: ROOM HEATING SYSTEM

We validated our scheduling approach using simulation of the room heating system example, described at the beginning of Section 1.1. In this case study, we consider 20 rooms ($n = 20$) with 20 heaters ($m = 20$), one in each room. The system parameters were randomly generated with room's

[2]`http://www.embotech.com` and `http://fiordos.ethz.ch`

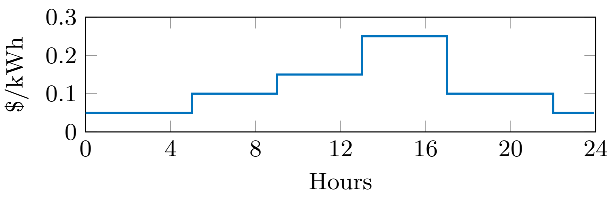

Figure 4: Time-varying price of energy.

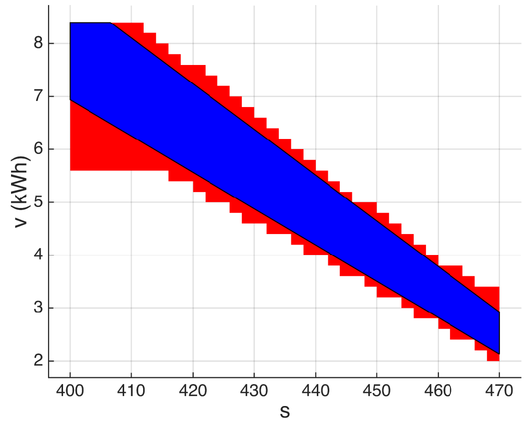

Figure 5: Set Ω constructed by gridding s and v.

thermal capacity $C_i \in [2500, 4500]$ (kJ/K) and thermal conductance K_i chosen proportionately from $[0.48, 0.72]$ (kW/K). The heaters' powers were chosen based on the size of the rooms (C_i values) $\rho_i \in \{4.5, 7.5\}$ (kW). We randomly assigned rooms which can thermally interact with each other and the value of their inter-room thermal conductance $K_{i,j}$ was chosen from $[0.1, 0.2]$ (kW/K). Room temperatures were required to be kept in the comfort range between 20.3 and 24.5 Celsius degrees. The ambient air temperature was varying but bounded between 6°C and 16°C. The internal heat gain in each room was between 0.1 and 1.0 (kW). Forecasts of the ambient air temperature are provided with an accuracy of 0.2°C, and forecasts of the internal heat gains are provided by occupancy and operation schedules, with an accuracy of 0.1 (kW). The daily time-varying energy price (Fig. 4) is 5 times more expensive during the on-peak hours ($0.25/kWh from 1-5 PM) than during the off-peak hours ($0.05/kWh at night). The peak demand price is $2/kW.

7.1 Complexity of the MILP

We first tried solving the MILP (3) for a trivially small scale system of only two rooms ($n = m = 2$) with a horizon of $N = 288$ (24 hours with 5-minute sampling time), on an 8-core MacPro with 64Gb RAM running the state-of-the-art commercial solver Gurobi 6.0. It did not finish after 15 minutes (three times the sampling time) and was therefore interrupted. Although there exist techniques to improve the performance of the optimization, for example move-blocking and warm-start, we believe that it would not scale beyond a very small number of subsystems running in real-time on a control computer with limited processing power.

7.2 Scalable Scheduling Design

We performed the design process outlined in Section 6.1 and computed the parameters of the model abstraction. The sampling time is $T = 5$min. Taking the most time was the construction of the set Ω on a grid of $50 \times 50 = 2{,}500$ cells. Reported in red color in Fig. 5 is the union of the cells resulting from this computation, for $\gamma = 20$. To simplify Ω, we took the blue polyhedron inside this union as Ω.

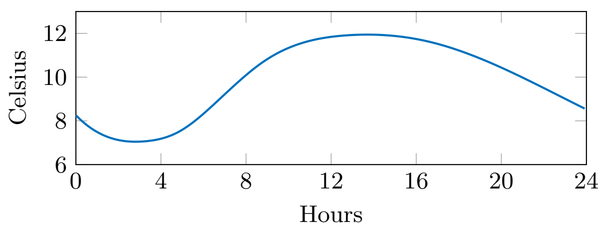

Figure 6: Ambient air temperature profile.

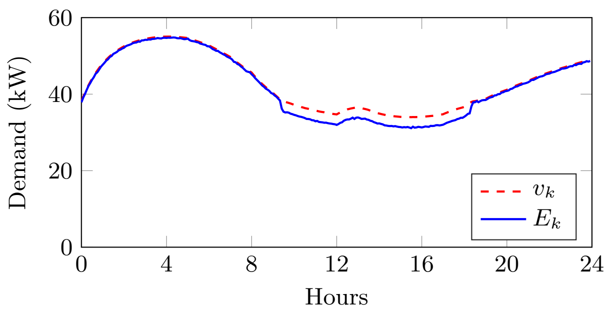

Figure 7: Aggregated energy demand upper-bounds v_k and the actual demands E_k.

7.3 Simulation Results

The ambient air temperature profile for a day is plotted in Fig. 6. The internal heat gain profiles were generated following a typical pattern in office buildings. The disturbance forecasts were generated from the actual profiles with random errors within the given accuracies. We simulated the run-time implementation of our approach (Section 6.2) in Matlab with Gurobi 6.0 as the optimization solver. Each of the top-level and middle-level optimizations took less than 15ms. Each iteration of our scheduling algorithm took less than 30ms. Therefore our run-time scheduling algorithm is very scalable compared to the MILP approach.

Figure 7 plots the aggregated energy demand upper-bound v_k (red, dashed), computed by the top-level optimization, and the actual total energy demand E_k (blue, solid). Obviously, E_k never exceeded v_k. Also observe that the total demands, both the upper-bounds and the actual values, were lowered during the on-peak hours to reduce the cost. The total cost for energy, including the usage charge and the peak demand charge, was $218.60 with 1004.38 kWh of consumption and 54.815 kW peak demand. About half of the total cost was the peak demand charge.

Figure 8 reports the air temperatures of the first 4 rooms, which were always maintained inside the comfort range.

8. CONCLUSIONS

We developed an approach which combines techniques from control theory and computer science to solve the energy scheduling problem for large-scale systems with fast sampling time. The approach uses a model abstraction method based on the concept of simulation relations between tran-

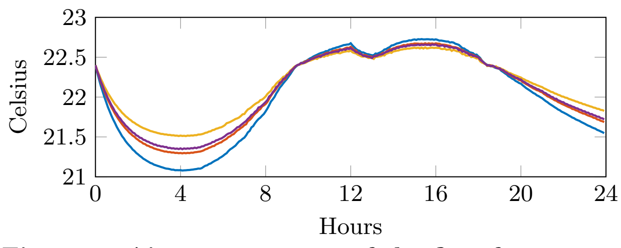

Figure 8: Air temperatures of the first four rooms.

sition systems, which allows us to reduce the original multi-state multi-binary-input model to a single-state single-real-input model with input bound tracking and safety guarantees. Unlike the mixed integer programming approach, which has computational issues for any system of practical size, our approach is much more scalable. While the offline design process may require significant computing power, the run-time implementation is lightweight that it can potentially be implemented on embedded computers. We validated our approach using Matlab simulations of a room heating system. Our numerical simulations showed that while the mixed integer programming approach may not work beyond a trivially small-scale system, our approach can handle efficiently a much larger system with a fast sampling rate and a long control horizon.

A future extension of this work is to test the control algorithms on an embedded platform to verify its run-time scalability. Another direction is to develop better scheduling algorithms for the low-level actuator scheduler, such that not only the averaged interval demand but also the instantaneous demand are reduced. We also aim to apply the approach to more practical systems other than the room heating system.

9. REFERENCES

[1] A. Marr. *Tea Time in Britain, Britain from Above*, 2008 (accessed March 30, 2015).

[2] TRFund Energy Study. Understanding PECO's general service tariff, 2007.

[3] Kin Cheong Sou, James Weimer, Henrik Sandberg, and Karl Henrik Johansson. Scheduling smart home appliances using mixed integer linear programming. In *Decision and Control and European Control Conference (CDC-ECC)*, pages 5144–5149, 2011.

[4] Di Zhang, Lazaros G Papageorgiou, Nouri J Samsatli, and Nilay Shah. Optimal scheduling of smart homes energy consumption with microgrid. In *ENERGY 2011, The First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies*, pages 70–75, 2011.

[5] Jane W. S. Liu. *Real-time Systems.* Prentice Hall, 2000.

[6] T. Facchinetti, E. Bini, and M. Bertogna. Reducing the peak power through real-time scheduling techniques in cyber-physical energy systems. In *International Workshop on Energy Aware Design and Analysis of Cyber Physical Systems*, 2010.

[7] Truong X. Nghiem, Madhur Behl, Rahul Mangharam, and George J. Pappas. Scalable scheduling of building control systems for peak demand reduction. In *American Control Conference (ACC)*, 2012.

[8] Truong X. Nghiem, George J. Pappas, and Rahul Mangharam. Event-based green scheduling of radiant systems in buildings. In *Proceedings of the American Control Conference (ACC)*, 2013.

[9] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proc. IEEE*, 88(7):971–984, 2000.

[10] A. Girard and G. J. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Trans. Autom. Control*, 52(5):782–798, 2007.

[11] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking.* MIT press, 1999.

[12] G. J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Trans. Autom. Control*, 45(6):1144–1160, 2000.

[13] A. Girard and G. J. Pappas. Verification using simulation. In *Hybrid Systems: Computation and Control*, pages 272–286. Springer, 2006.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2006.

APPENDIX

Proof of Theorem 1. We first suppose that A is diagonal with λ_i being its diagonal elements. In this case, each element $e_i((k+1)T)$ can be calculated independently

$$e_i((k+1)T) = \int_{kT}^{(k+1)T} \mathrm{e}^{\lambda_i((k+1)T-s)} b_i(u(s)-\eta_k)\mathrm{d}s.$$

If $\lambda_i = 0$ then obviously $e_i((k+1)T) = 0$. Consider the case $\lambda_i \neq 0$. Observe that $\int_{kT}^{(k+1)T} \mu_i b_i(u(s)-\eta_k)\mathrm{d}s = 0$ for any constant μ_i. Consequently

$$e_i((k+1)T) = \int_{kT}^{(k+1)T} (\mathrm{e}^{\lambda_i((k+1)T-s)} - \mu_i) b_i(u(s)-\eta_k)\mathrm{d}s$$

where $\mu_i = 1$ if $\lambda_i > 0$ and $\mu_i = \mathrm{e}^{\lambda_i T}$ if $\lambda_i < 0$. It is straightforward to show that $\mathrm{e}^{\lambda_i((k+1)T-s)} - \mu_i \geqslant 0$ for all $s \in [kT, (k+1)T]$. Because $u(s) \in \{0,1\}^m$ for all s, $\underline{b}_i \leqslant b_i u(s) \leqslant \bar{b}_i$ with $\underline{b}_i$ and $\bar{b}_i$ defined as above. Therefore $(\mathrm{e}^{\lambda_i((k+1)T-s)} - \mu_i)(\underline{b}_i - b_i\eta_k) \leqslant (\mathrm{e}^{\lambda_i((k+1)T-s)} - \mu_i) b_i(u(s) - \eta_k) \leqslant (\mathrm{e}^{\lambda_i((k+1)T-s)} - \mu_i)(\bar{b}_i - b_i\eta_k)$. It follows that

$$\underline{\varepsilon}_i - \xi_i b_i \eta_k \leqslant e_i((k+1)T) \leqslant \bar{\varepsilon}_i - \xi_i b_i \eta_k$$

where $\underline{\varepsilon}_i = \xi_i \underline{b}_i$, $\bar{\varepsilon}_i = \xi_i \bar{b}_i$, and $\xi_i = \int_{kT}^{(k+1)T} (\mathrm{e}^{\lambda_i((k+1)T-s)} - \mu_i)\mathrm{d}s = \frac{1}{\lambda_i}(\mathrm{e}^{\lambda_i T} - 1) - \mu_i T$. The result is proved for when A is diagonal. If A is diagonalizable by V then by applying the above result to $D = VA$ and VB, the theorem is proved.

Proof sketch of Theorem 2. To prove this theorem, we need the following lemma, which can be verified to be true by inspection:

LEMMA 3. *Consider the matrices $M_l \in \mathbb{R}^{n\times m}$ and $M_r \in \mathbb{R}^{n\times p}$, and the sets $\mathcal{P}_l \subset \mathbb{R}^m$ and $\mathcal{P}_r \subset \mathbb{R}^p$. The following statements are equivalent*

- *$\forall x \in \mathcal{P}_l$, $\exists u \in \mathcal{P}_r$ such that $M_l x = M_r u$*
- *$M_l \mathcal{P}_l \subseteq M_r \mathcal{P}_r$.*

From Definition 3 and all the constraints and dynamical equations of Σ_a and Σ_s, we have that $\mathcal{R}$ is an input-constrained simulation relation of T_1 by T_2 if and only if

$$\forall x \in \mathcal{P}_l,\ \exists u \in \mathcal{P}_r \text{ such that } M_l x = M_r u$$

where M_l, $\mathcal{P}_l$, M_r, and $\mathcal{P}_r$ are defined as in the theorem. The result then follows directly from the above lemma.

Proof sketch of Theorem 3. It is straightforward to see that the matrices H_l, H_r and the vectors k_l, k_r define the polyhedral sets $\mathcal{P}_l$ and $\mathcal{P}_r$ in Theorem 2. We only need to show that condition (9) is sufficient for $M_l\mathcal{P}_l \subseteq M_r\mathcal{P}_r$. Define $\mathcal{P} = \{x \mid H_r Q M_l x \leqslant k_r - H_r q\}$. It follows from (9) and the Farkas' lemma that $\mathcal{P}_l \subseteq \mathcal{P}$, which leads to $M_l\mathcal{P}_l \subseteq M_l\mathcal{P} = \{y \mid H_r Q y \leqslant k_r - H_r q\}$. For all $y \in M_l\mathcal{P}_l$, $H_r(Qy + q) \leqslant k_r \Rightarrow Qy + q \in \mathcal{P}_r \Rightarrow M_r(Qy+q) = y \in M_r\mathcal{P}_r$, where we use the fact that $M_r Q = \mathbb{I}$ and $M_r q = \mathbf{0}$. Thus $M_l\mathcal{P} \subseteq M_r\mathcal{P}_r$. Therefore $M_l\mathcal{P}_l \subseteq M_r\mathcal{P}_r$. The proof is complete.