Computation Time Reduction Techniques for Model Predictive Control of Hybrid Systems

Koichi Kobayashi

Japan Advanced Institute of Science and Technology

Outline of This Presentation

1. Hybrid Systems
2. Model Predictive Control
3. Technical Difficulty and Our Approach
4. Proposed Method
5. Further Extension

Outline of This Presentation

1. Hybrid Systems
2. Model Predictive Control
3. Technical Difficulty and Our Approach
4. Proposed Method
5. Further Extension

What are Hybrid Systems?

Hybrid Systems: Dynamical Systems with Continuous Variables and Discrete Variables

【Example 1】 Automobile

Accelerator pedal (Torque)

Continuous

Gear 1/Gear 2/Gear 3･･･

Discrete

- Biped walking robot
- Aircraft autopilot modes
- Air and ground transportation
- Chemical plants

…

What are Hybrid Systems?

【Example 1】 Automobile

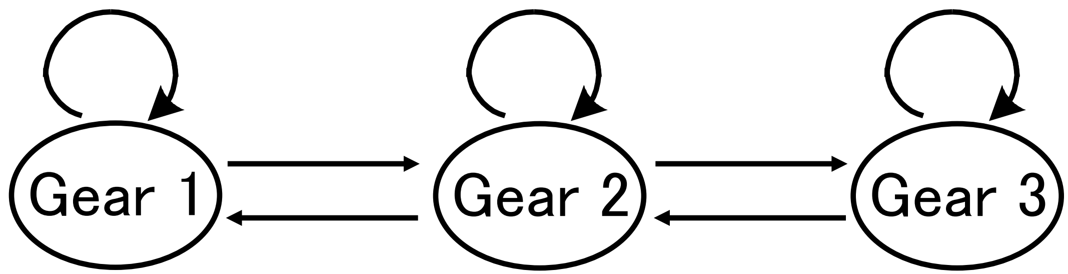
Accelerator pedal

Continuous

Gear 1/Gear 2/Gear 3･･･

Discrete

Finite Automaton:



Dynamics of Automobile in Gear *i* :

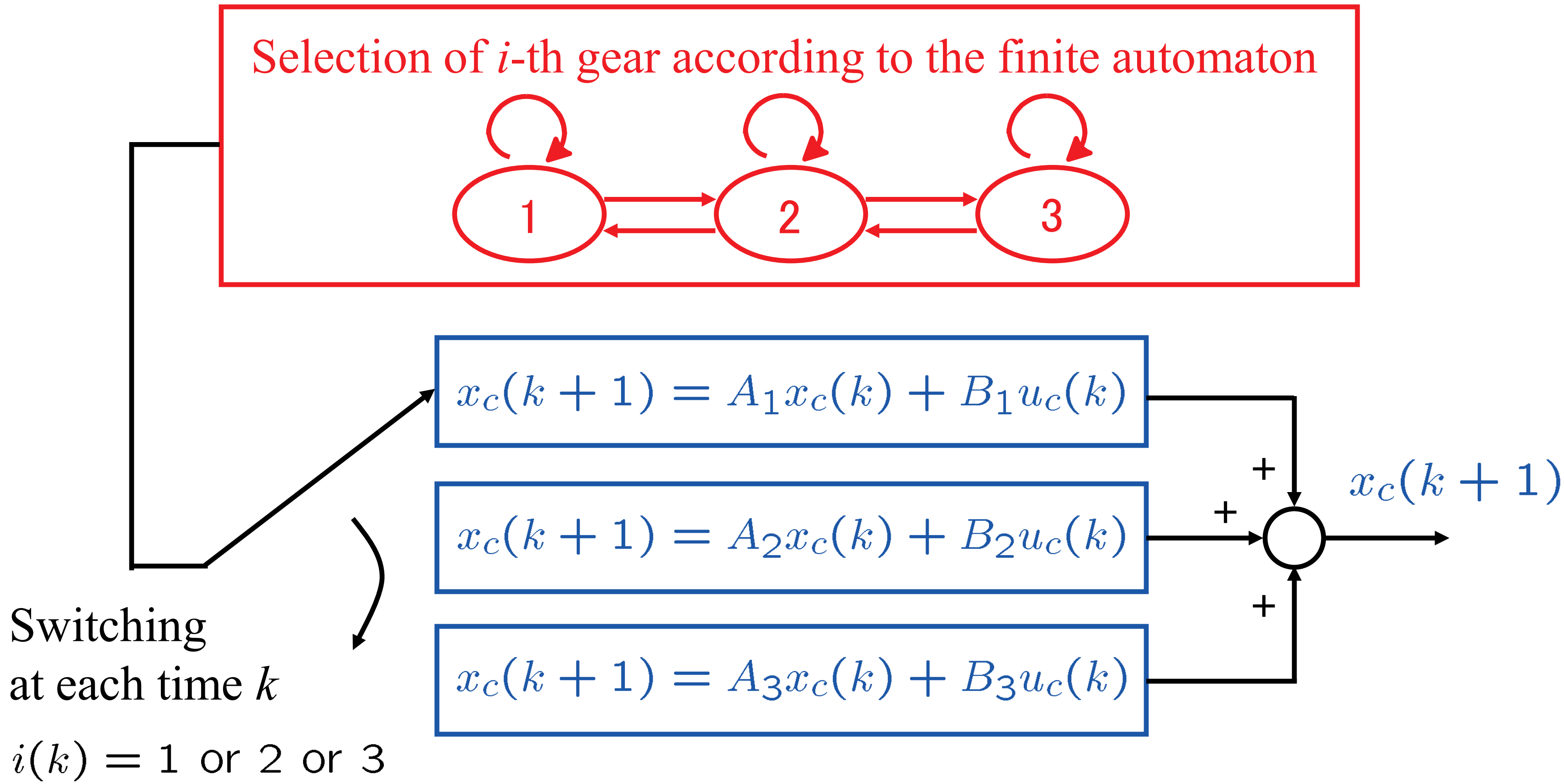
(Discrete-time state equation)

$$x_c(k+1) = A_{i(k)} x_c(k) + B_{i(k)} u_c(k)$$

$$x_c(k) = \begin{bmatrix} \text{Position} \\ \text{Velocity} \end{bmatrix} \qquad u_c(k) = \text{Torque}$$

$$i(k) = 1 \text{ or } 2 \text{ or } 3 \qquad k = 0,\ 1,\ 2,\ \cdots \ \text{Time}$$

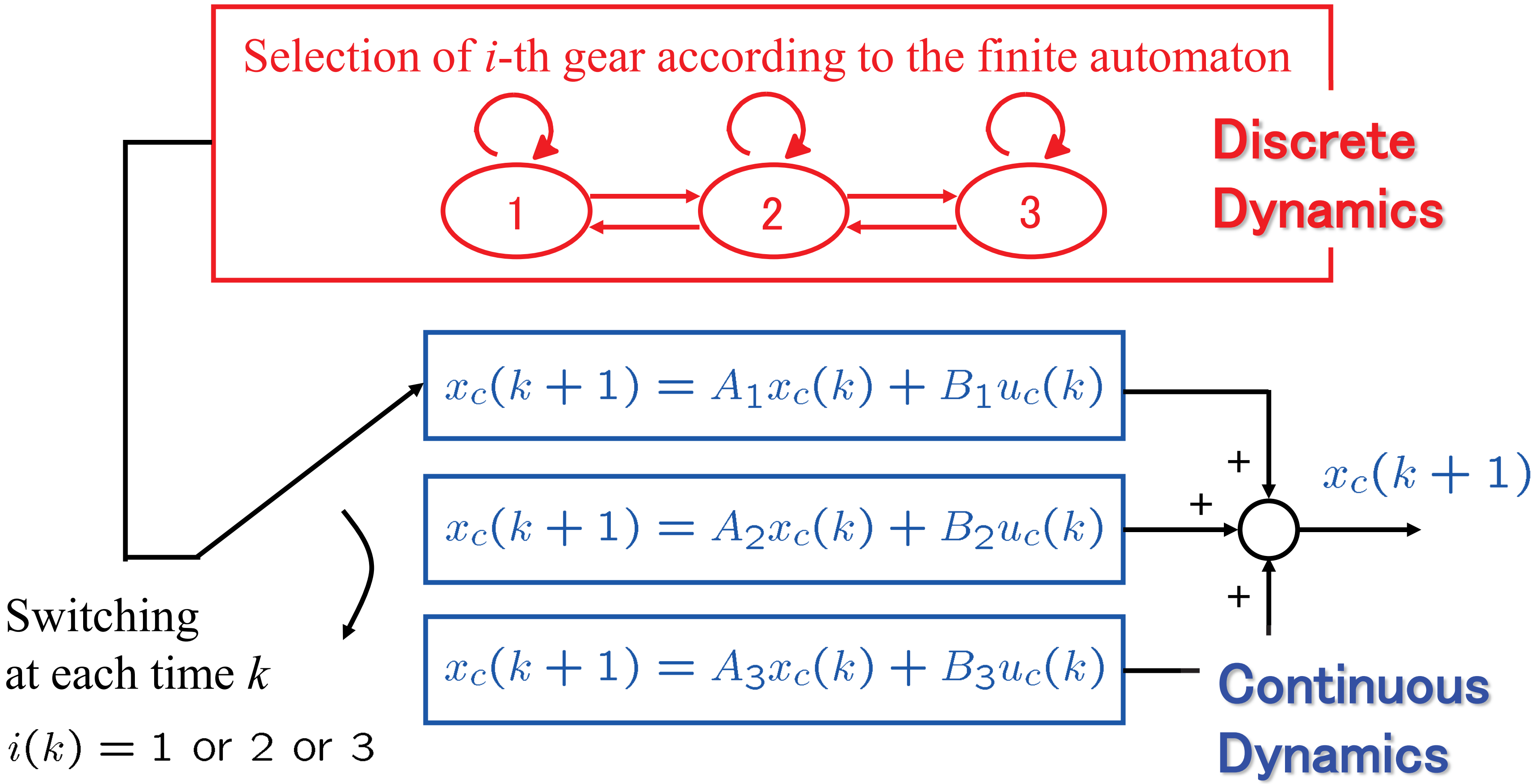
What are Hybrid Systems?



By determining $x_c(k),\ u_c(k),\ i(k)$ at time k ,

$x_c(k+1)$ at next time $k+1$ is uniquely determined.

What are Hybrid Systems?



<u>In general,</u>

(1) Gear 1, Gear 2, Gear 3 $\rightarrow$ mode 1, mode 2, mode 3

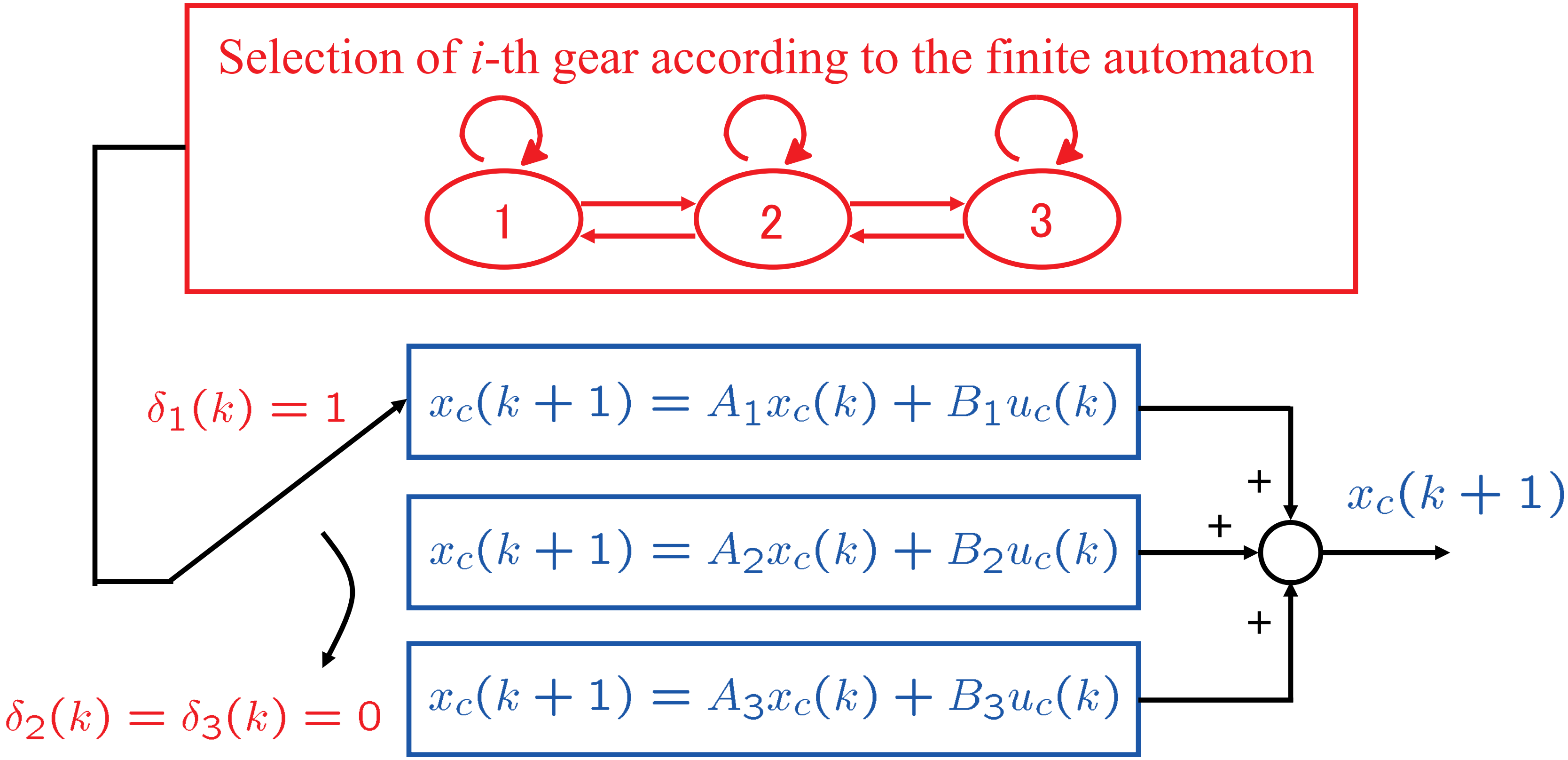
(2) Mode transition constraints depend on $x_c(k)$, $u_c(k)$.

(Piecewise Affine Systems)

Outline of This Presentation

1. Hybrid Systems
2. Model Predictive Control
3. Technical Difficulty and Our Approach
4. Proposed Method
5. Further Extension

Modeling of Hybrid Systems



Binary variables $\delta_1,\ \delta_2,\ \delta_3$ are used.

$$\begin{cases} \delta_i(k) = 1 & \text{if Gear } i \\ \delta_i(k) = 0 & \text{otherwise} \end{cases}$$

$$\Rightarrow \quad x_c(k+1) = \sum_{i=1}^{3} \delta_i(k)\Big(A_i x_c(k) + B_i u_c(k)\Big)$$

+ Model of the finite automaton

Modeling of Hybrid Systems

Model of Hybrid System:

$$x_c(k+1) = \sum_{i=1}^{3} \delta_i(k)\Big(A_i x_c(k) + B_i u_c(k)\Big)$$

+ Model of the finite automaton

$$\Downarrow$$

Mixed Logical Dynamical (MLD) Model:

[Bemporad and Morari 1999]

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ Cx(k) + Du(k) \leq E \end{cases} \qquad \begin{array}{l} x(k) \in \mathbf{R}^{n_c} \times \{0,1\}^{n_d} \\ u(k) \in \mathbf{R}^{m_c} \times \{0,1\}^{m_d} \end{array}$$

$$x(k) = \begin{bmatrix} x_c(k) \\ x_d(k) \end{bmatrix}, \quad x_c(k) \in \mathbf{R}^{n_c}, \quad x_d(k) \in \{0,1\}^{n_d}$$

$$u(k) = \begin{bmatrix} u_c(k) \\ u_d(k) \end{bmatrix}, \quad u_c(k) \in \mathbf{R}^{m_c}, \quad u_d(k) \in \{0,1\}^{m_d}$$

Continuous Dynamics **Discrete Dynamics**

Finite-time Optimal Control Problem

$$\begin{aligned}
\text{find} \quad & u(k) \in \mathbf{R}^{m_c} \times \{0,1\}^{m_d}, \quad k = T,\ T+1,\ \cdots, T+N-1 \\
\text{min} \quad & J = \sum_{i=T}^{T+N-1} \left\{ x'(i)Qx(i) + u'(i)Ru(i) \right\} + x'(N)Q_f x(N) \\
\text{subject to} \quad & \begin{cases} x(k+1) = Ax(k) + Bu(k) \\ Cx(k) + Du(k) \le E \end{cases}
\end{aligned}$$

T : Current time

$\Downarrow$

MIQP Problem (Mixed Integer Quadratic Programming Problem)

$$\begin{aligned}
\text{find} \quad & \bar{u} = \left[u^T(T) \quad u^T(T+1) \quad \cdots \quad u^T(T+N-1) \right]^T \\
\text{min} \quad & \bar{u}^T M_1 \bar{u} + \bar{u}^T M_2 x(T) \\
\text{subject to} \quad & L_1 \bar{u} \le L_2 x(T) + L_3
\end{aligned}$$

The dim. of continuous variables: $m_c N$

The dim. of binary variables: $m_d N$

The computation time is too long for practical plants.

Model Predictive Control

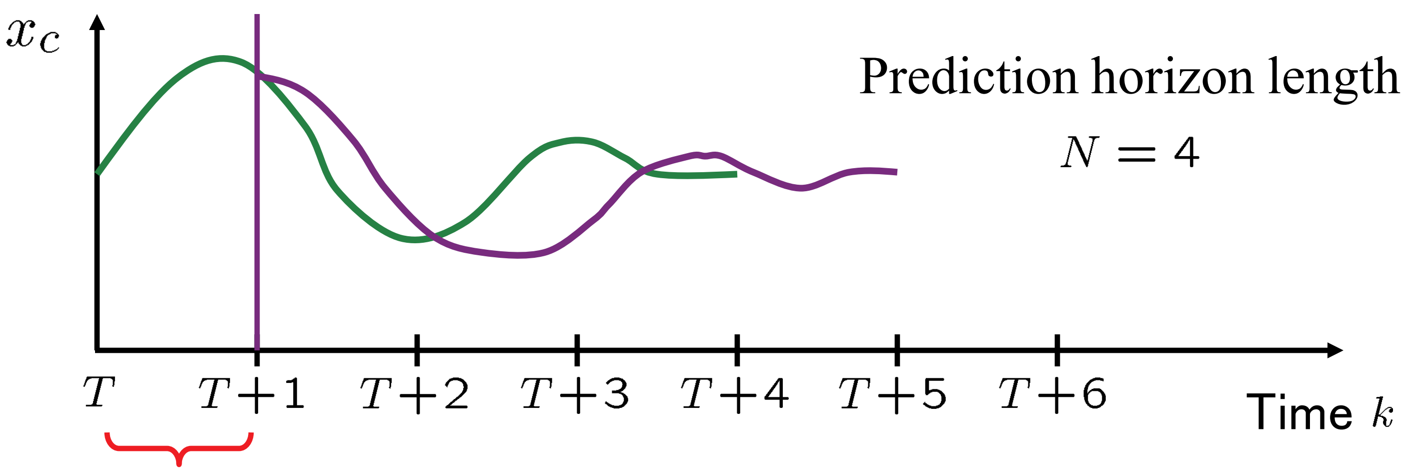
Model Predictive Control …

The finite-time optimal control problem is solved at each time.

Step 1: Derive $\bar{u} = \left[u'(T) \;\; u'(T+1) \;\; \cdots \;\; u'(T+N-1) \right]'$ by solving the MIQP problem.

Step 2: Apply only $u(T)$ to the plant.

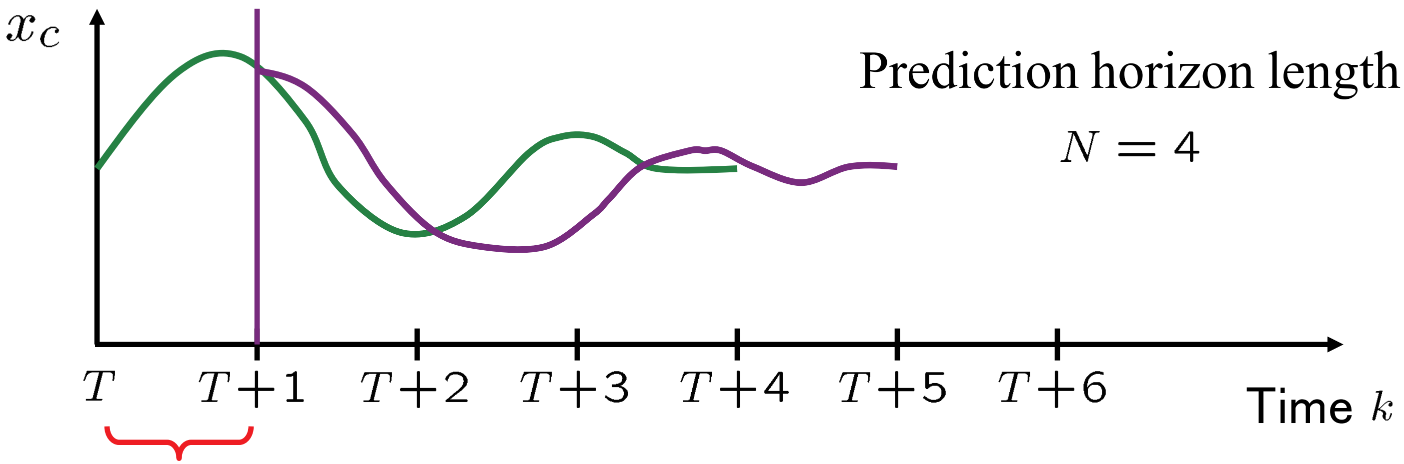
Step 3: Update $T \to T+1$, and go to Step 1.



The MIQP problem must be solved within the sampling period.

'On-line Optimization'

Model Predictive Control



The MIQP problem must be solved within the sampling period.

'On-line Optimization'

Another Approach… **Off-line Optimization**

The multi-parametric MIQP problem is solved.
Then the explicit controller $u^*(k) = f(x(k))$ is derived.

Practical applications are restricted.
Small size systems, Slow dynamical systems, …

Outline of This Presentation

1. Hybrid Systems?
2. Model Predictive Control
3. Technical Difficulty and Our Approach
4. Proposed Method
5. Further Extension

Summary of Our Approach

Technical Difficulty:

The computation time to solve the MIQP problem is too long.

$\Downarrow$ To overcome this difficulty

(1) Improvement of the MIQP solver

(2) Improvement of the modeling of discrete dynamics

→ The computation time depends on the modeling method.

However, few results from the modeling viewpoint

Summary of Our Approach

(2) Improvement of the modeling of discrete dynamics

→ The computation time depends on the modeling method.

The Standard Method: Inequality based modeling

[Bemporad and Morari 1999]

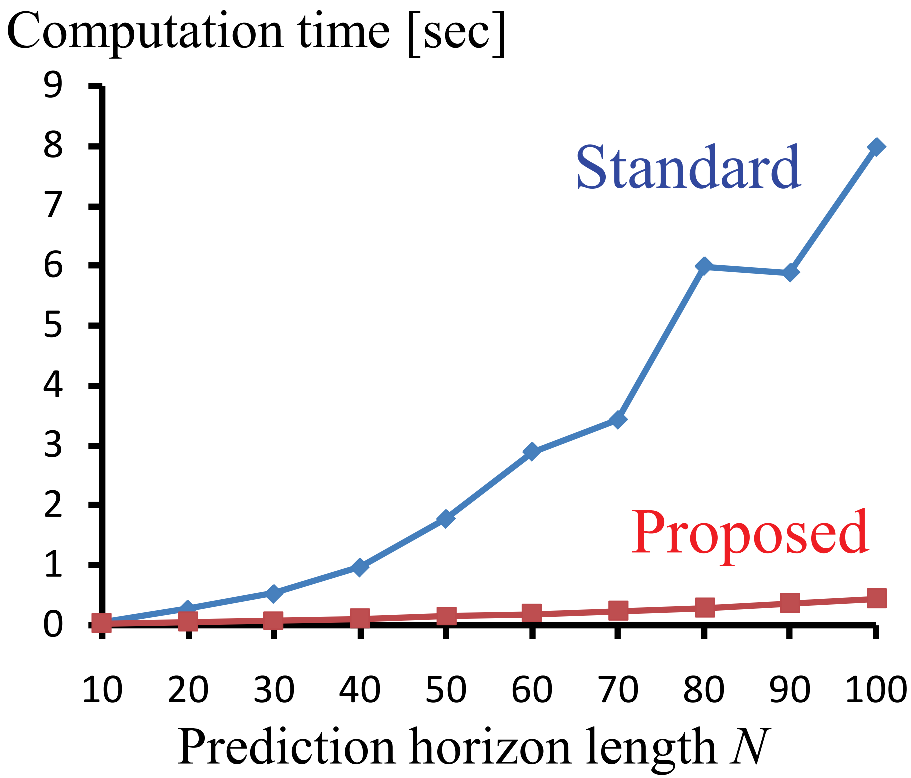
The Proposed Method:

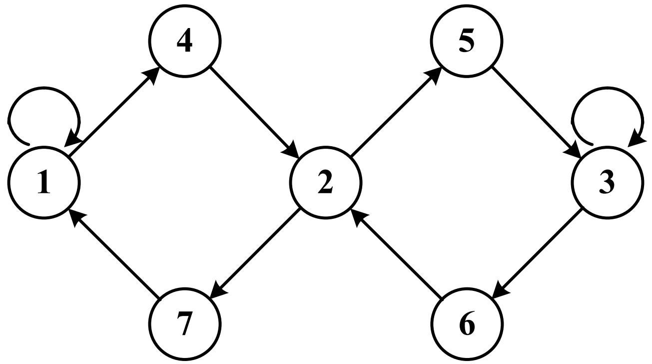
- State-equation based modeling [Kobayashi, Imura 2006, 2007]
- Time-sequence based modeling [Kobayashi, Imura 2007]
- Graph-switching based modeling

Effectiveness of The Proposed Method

Computation time of the finite-time optimal control problem:

- Plant: 2^{nd}-order system with 7-mode switching





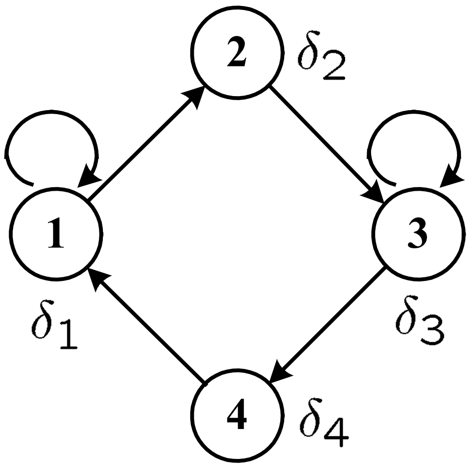
CPU: Intel Core2 Duo 3.0GHz
Memory: 2GB
MIQP solver: CPLEX 11.0

Reduction of the computational time is achieved only by changing from the standard to the proposed representation.

Outline of This Presentation

1. Hybrid Systems?
2. Model Predictive Control
3. Technical Difficulty and Our Approach
4. Proposed Method
5. Further Extension

The Standard Method: Inequality Based Modeling



$\delta_i \in \{0,1\},$

$\delta_1+\delta_2+\delta_3+\delta_4=1$

$$\begin{cases} \delta_i(k)=1 & \text{if mode } i \\ \delta_i(k)=0 & \text{otherwise} \end{cases}$$

Propositional logic:

$$[\delta_1(k)=1] \rightarrow [\delta_1(k+1)=1] \vee [\delta_2(k+1)=1] \qquad \text{(mode 1)}$$

$$\vdots$$

$$\Downarrow$$

Linear inequalities: $\underbrace{\delta_1(k)}_{x_{d1}(k)} \leq \underbrace{\delta_1(k+1)}_{u_{d1}(k)} + \underbrace{\delta_2(k+1)}_{u_{d2}(k)}$

$$\vdots$$

$$\Downarrow$$

A combination of state equation and inequality:

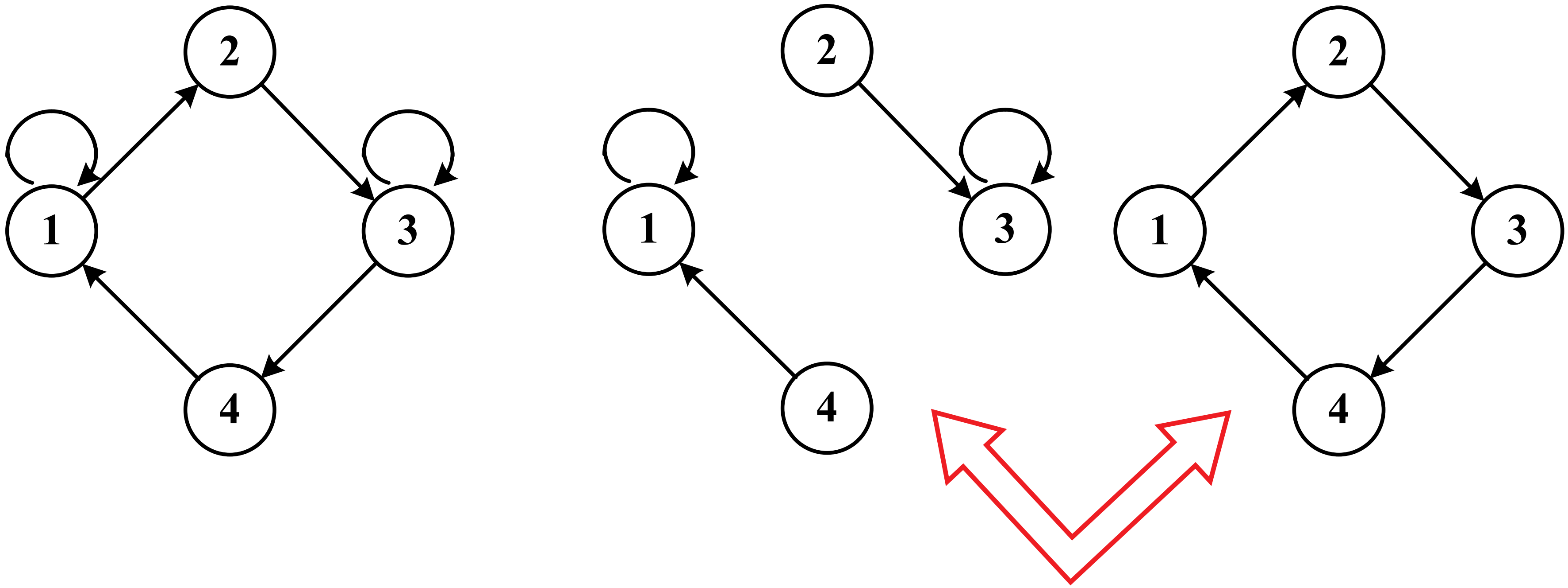
$$\begin{cases} x_d(k+1)=u_d(k) \\ C_d x_d(k)+D_d u_d(k) \leq E_d \end{cases}$$

MLD model

$$\begin{cases} x(k+1)=Ax(k)+Bv(k) & x \in \mathbf{R}^{n_c} \times \{0,1\}^{n_d} \\ Cx(k)+Dv(k) \leq E & v \in \mathbf{R}^{m_c} \times \{0,1\}^{m_d} \end{cases}$$

The Key Idea of The Proposed Method

Finite Automaton ⇨ **Decomposed to two finite automata**

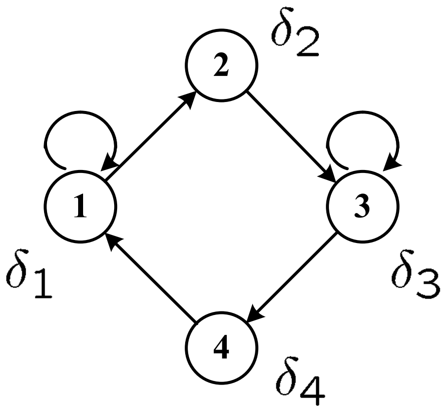


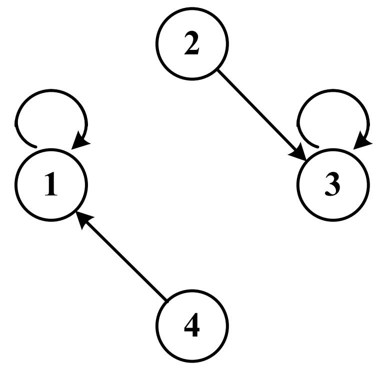
For a given initial mode, by selecting one of finite automata, the mode at the next time is uniquely determined.

⇨ The branch and bound algorithm works effectively.

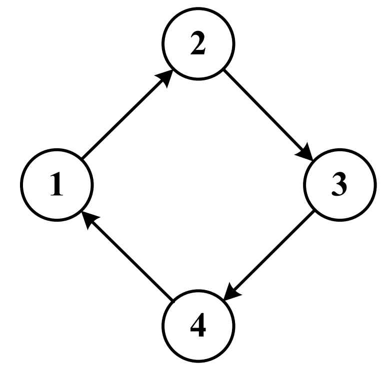
Procedure For Deriving The Proposed Model

Finite Automaton $\Longrightarrow$ **Decomposed to two finite automata**





$\delta(k) = 1$



$\delta(k) = 0$

<u>Model based on the adjacency matrix</u>

$$\begin{bmatrix} \delta_1(k+1) \\ \delta_2(k+1) \\ \delta_3(k+1) \\ \delta_4(k+1) \end{bmatrix} = \begin{bmatrix} \delta(k) & 0 & 0 & 1 \\ 1-\delta(k) & 0 & 0 & 0 \\ 0 & 1 & \delta(k) & 0 \\ 0 & 0 & 1-\delta(k) & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix}$$

$$\delta_1(k) + \delta_2(k) + \delta_3(k) + \delta_4(k) = 1$$

Procedure For Deriving The Proposed Model

Model based on the adjacency matrix

$$\begin{bmatrix} \delta_1(k+1) \\ \delta_2(k+1) \\ \delta_3(k+1) \\ \delta_4(k+1) \end{bmatrix} = \begin{bmatrix} \delta(k) & 0 & 0 & 1 \\ 1-\delta(k) & 0 & 0 & 0 \\ 0 & 1 & \delta(k) & 0 \\ 0 & 0 & 1-\delta(k) & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix}$$

$$\delta_1(k) + \delta_2(k) + \delta_3(k) + \delta_4(k) = 1$$

By defining $u_1(k) := \delta(k)\delta_1(k),\ u_2(k) := \delta(k)\delta_3(k)$,
we obtain

$$\begin{bmatrix} \delta_1(k+1) \\ \delta_2(k+1) \\ \delta_3(k+1) \\ \delta_4(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}.$$

Procedure For Deriving The Proposed Model

Nonlinear polynomial terms can be transformed into linear inequalities as follows.

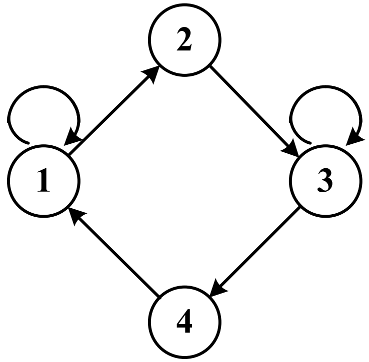
$$u_1(k) = \delta(k)\delta_1(k) \Leftrightarrow \begin{cases} \delta_1(k) - u_1(k) + \delta(k) \leq 1 \\ -\delta_1(k) + 2u_1(k) - \delta(k) \leq 0 \end{cases}$$

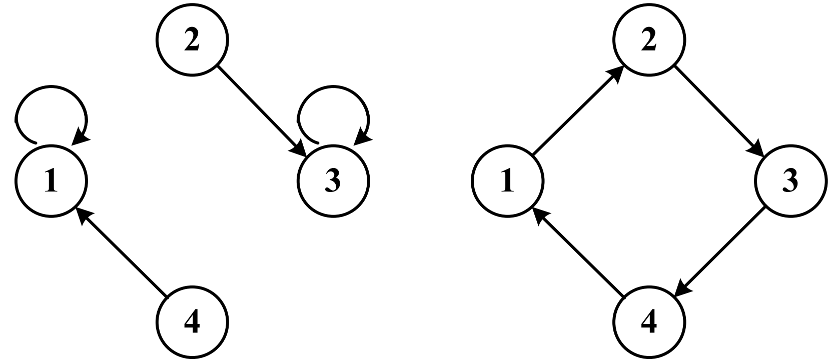
$$u_2(k) = \delta(k)\delta_3(k) \Leftrightarrow \begin{cases} \delta_3(k) - u_2(k) + \delta(k) \leq 1 \\ -\delta_3(k) + 2u_2(k) - \delta(k) \leq 0 \end{cases}$$

[Cavalier *et al.* 1990]

Procedure For Deriving The Proposed Model

Finite Automaton $\Longrightarrow$ **Decomposed to two finite automata**





$$\begin{bmatrix} \delta_1(k+1) \\ \delta_2(k+1) \\ \delta_3(k+1) \\ \delta_4(k+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta_1(k) \\ \delta_2(k) \\ \delta_3(k) \\ \delta_4(k) \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}$$

$$\begin{cases} \delta_1(k) - u_1(k) + \delta(k) \leq 1 \\ -\delta_1(k) + 2u_1(k) - \delta(k) \leq 0, \end{cases} \quad \begin{cases} \delta_3(k) - u_2(k) + \delta(k) \leq 1 \\ -\delta_3(k) + 2u_2(k) - \delta(k) \leq 0 \end{cases}$$

$$\delta_1(0), \delta_2(0), \delta_3(0), \delta_4(0) \in \{0,1\}, \quad \sum_{i=1}^{4} \delta_i(0) = 1$$

$$\delta_1(k), \delta_2(k), \delta_3(k), \delta_4(k) \in \mathbf{R}, \; u_1(k), u_2(k), \delta(k) \in \{0,1\}$$

Modeling of Hybrid Systems

Model of Hybrid System:

$$x_c(k+1) = \sum_{i=1}^{3} \delta_i(k)\Big(A_i x_c(k) + B_i u_c(k)\Big)$$

+ Model of the finite automaton

$$\Downarrow$$

Mixed Logical Dynamical (MLD) Model:

[Bemporad and Morari 1999]

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ Cx(k) + Du(k) \leq E \end{cases} \qquad \begin{array}{l} x(k) \in \mathbf{R}^{n_c} \times \{0,1\}^{n_d} \\ u(k) \in \mathbf{R}^{m_c} \times \{0,1\}^{m_d} \end{array}$$

$$x(k) = \begin{bmatrix} x_c(k) \\ x_d(k) \end{bmatrix}, \quad \boxed{x_c(k) \in \mathbf{R}^{n_c},} \quad \boxed{x_d(k) \in \{0,1\}^{n_d}}$$

$$u(k) = \begin{bmatrix} u_c(k) \\ u_d(k) \end{bmatrix}, \quad \boxed{u_c(k) \in \mathbf{R}^{m_c},} \quad \boxed{u_d(k) \in \{0,1\}^{m_d}}$$

Continuous Dynamics **Discrete Dynamics**

Outline of This Presentation

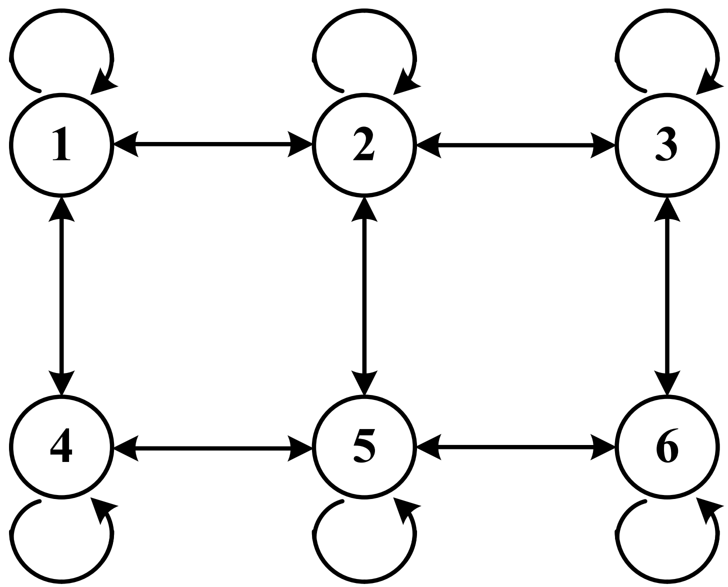
1. Hybrid Systems?
2. Model Predictive Control
3. Technical Difficulty and Our Approach
4. Proposed Method
5. Further Extension

Further Extension

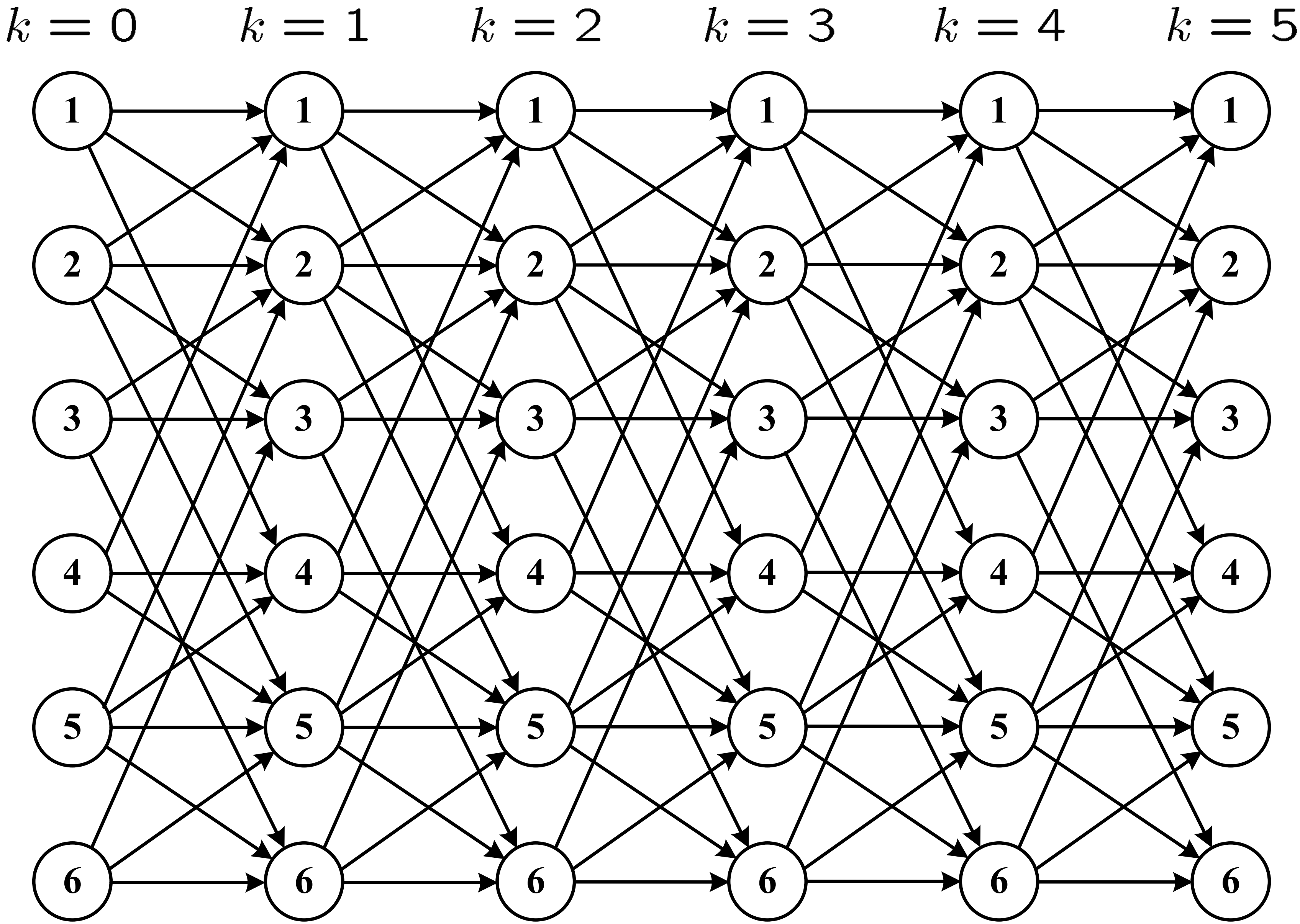
As a further approach for reducing the computation time,

let us approximate a time sequence of finite automata.

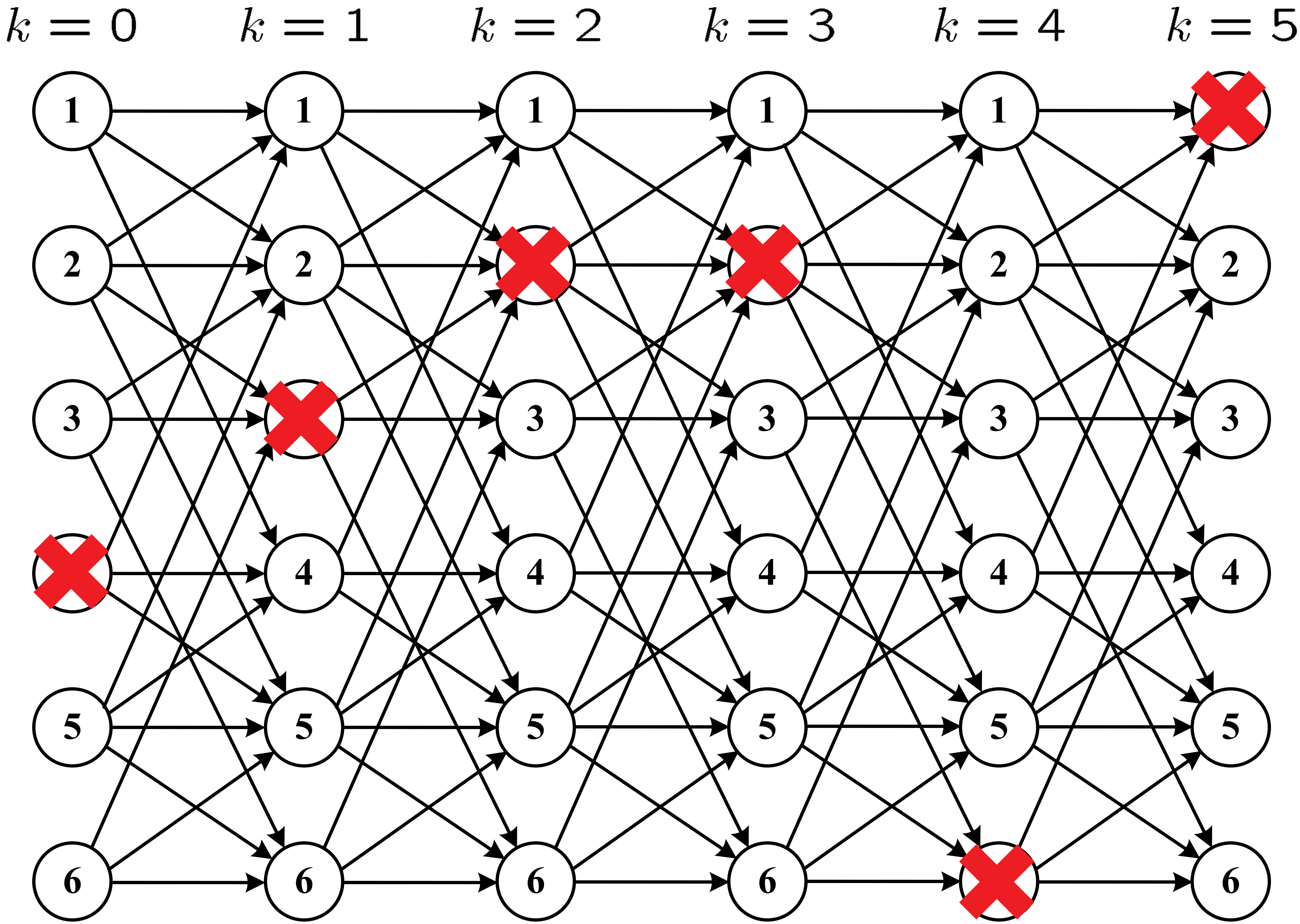
Example:



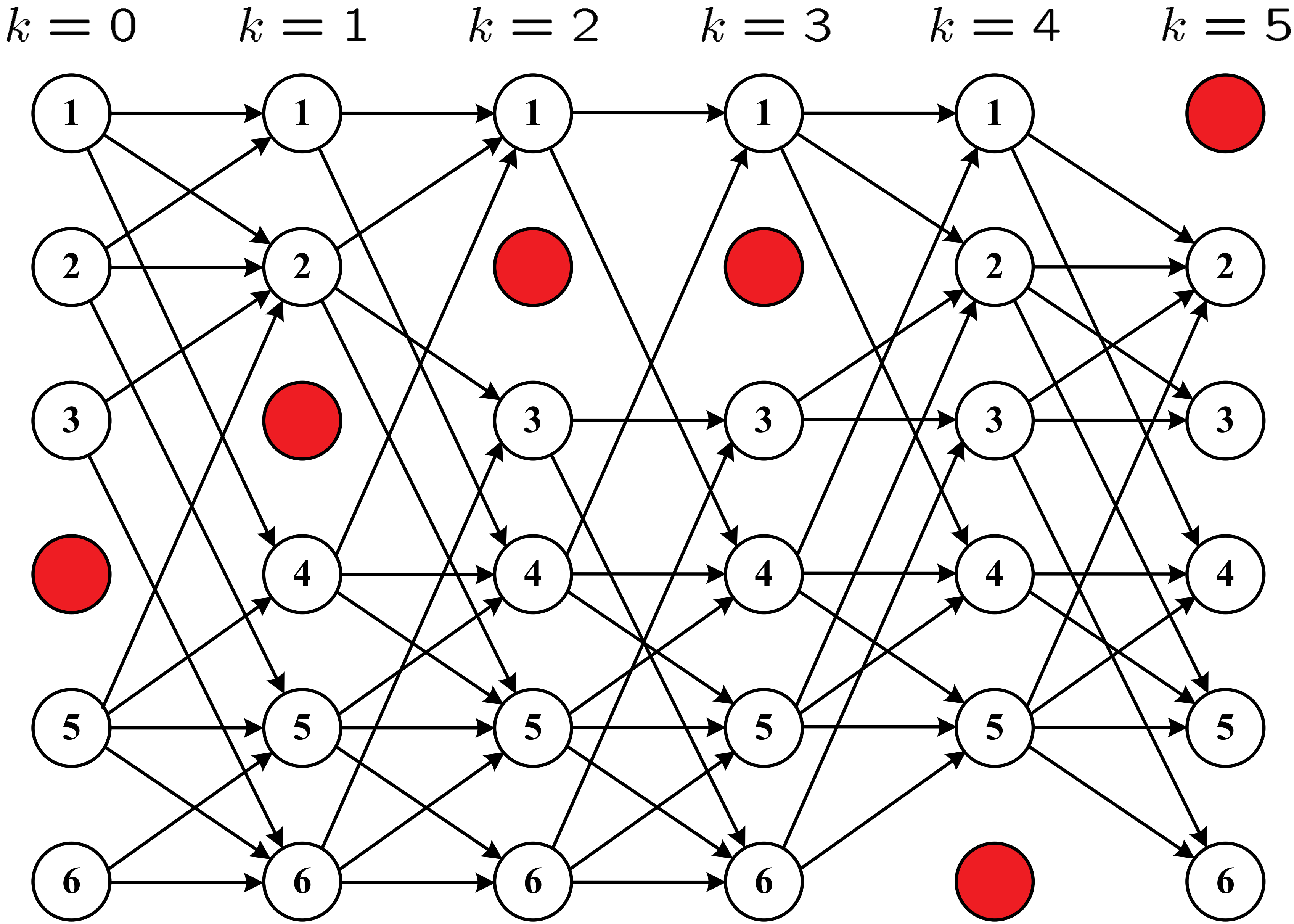
Time Sequence of Finite Automaton



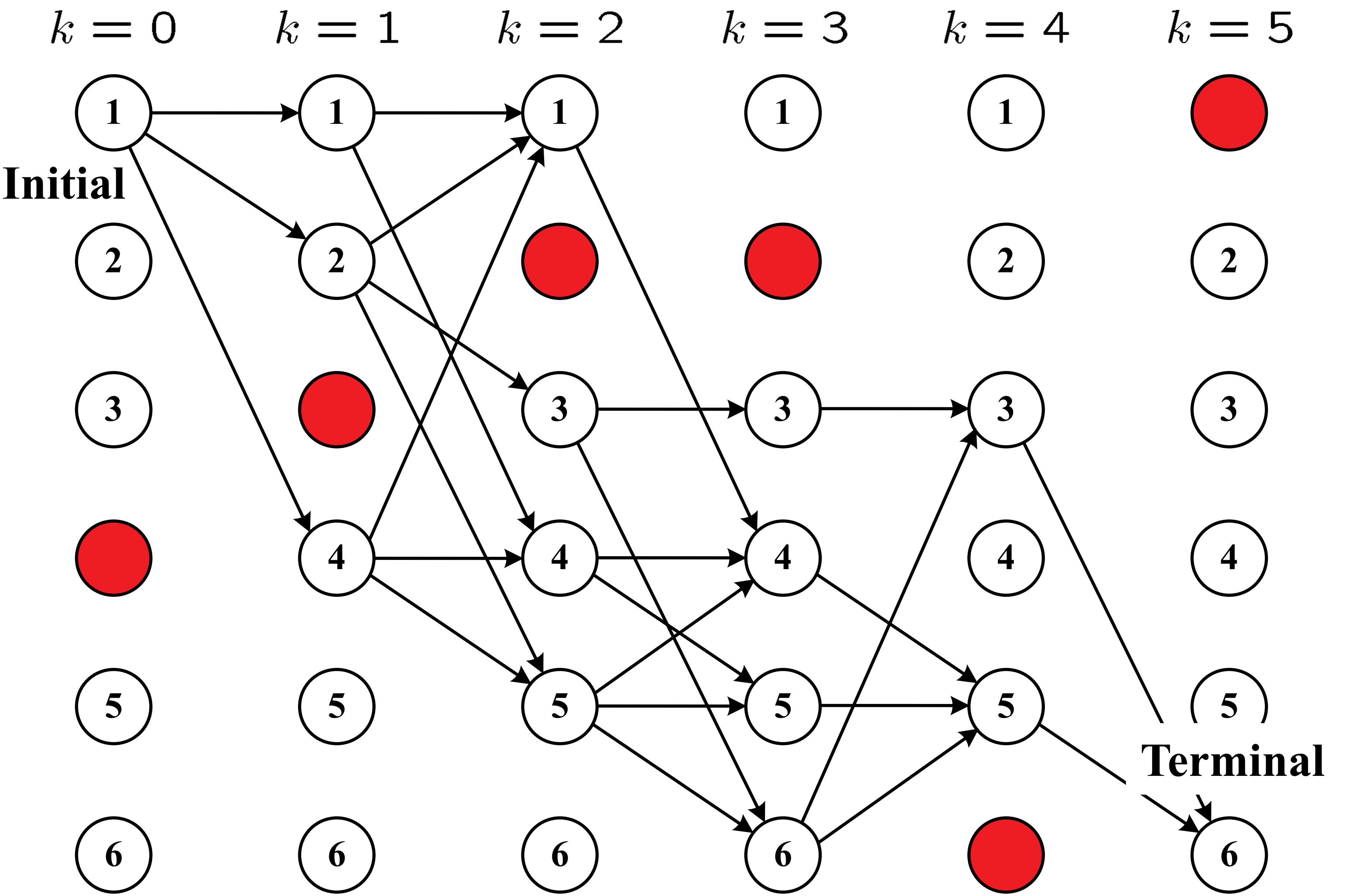
If there are temporal logic constraints,



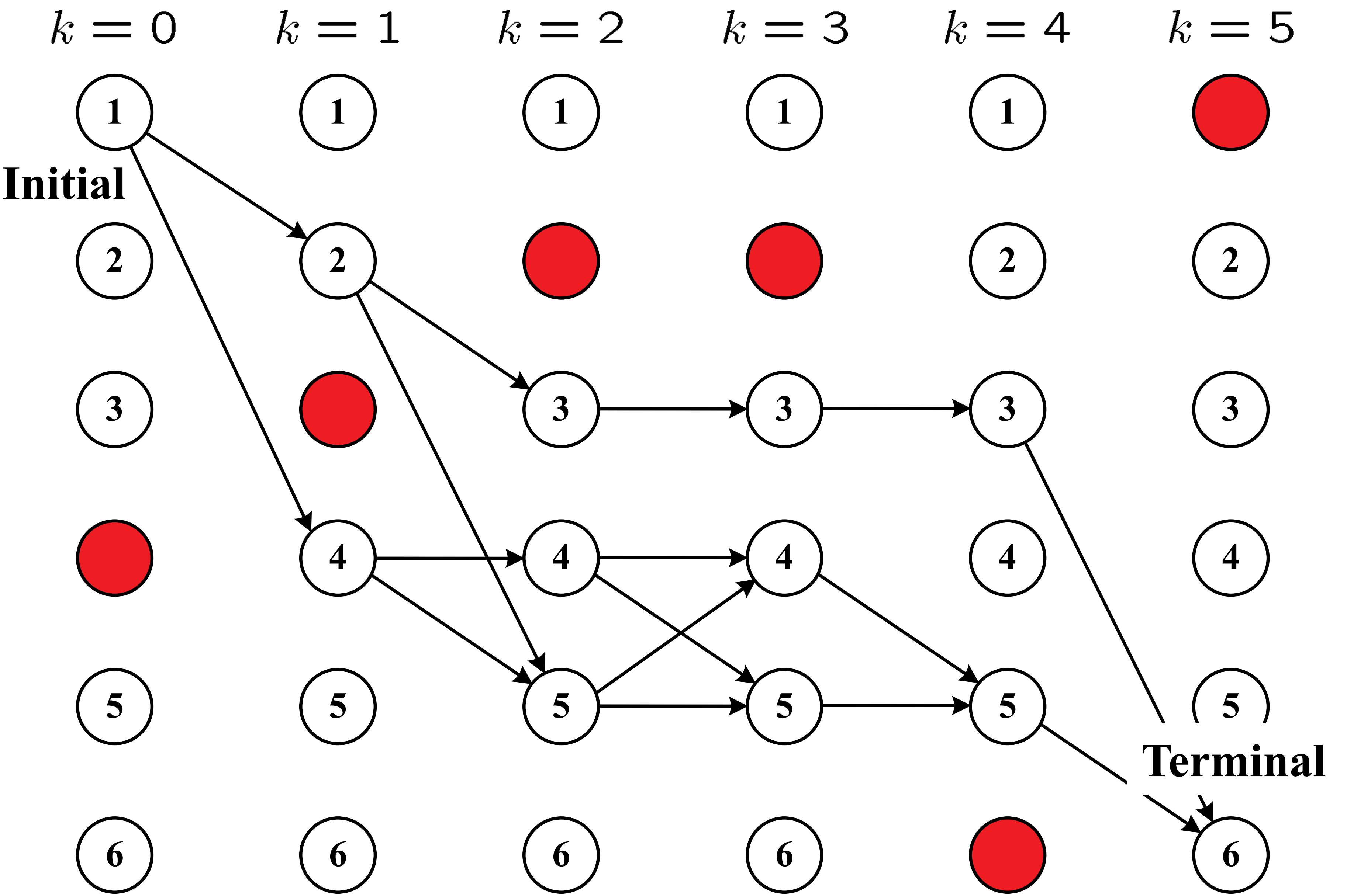
then the time sequence is limited.



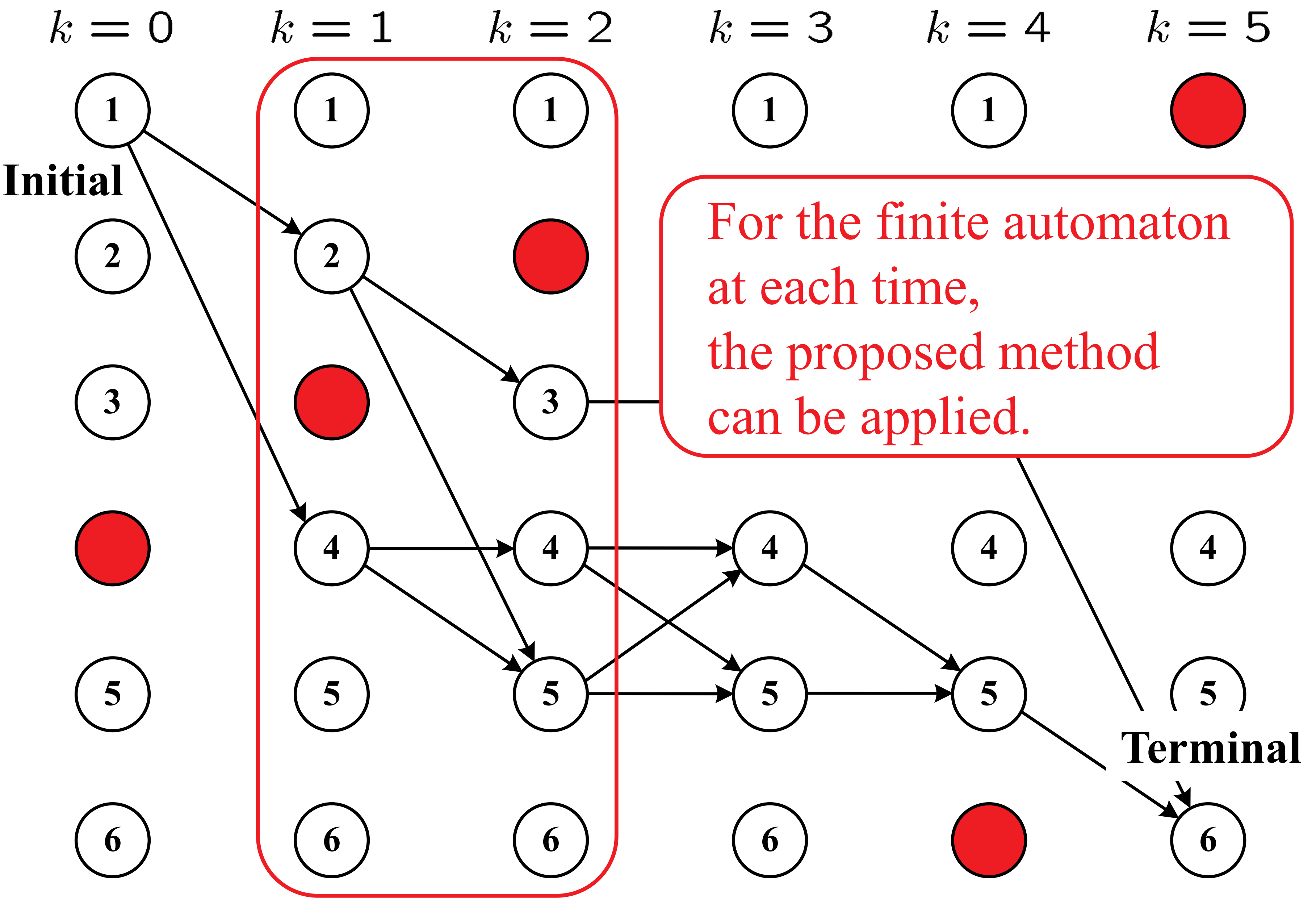
If initial and terminal modes are given, …



Approximate Time Sequence → Suitable??



Approximate Time Sequence → Suitable??



Conclusion

【 Contents of This Presentation 】

（1） Difficulty of hybrid systems control

（2） Introduction to our framework:
Modeling of Finite Automata
for Computation Time Reduction

（3） Further approach: focus on time sequences

【 Future Work 】

（1） Improved algorithm based on
both the on-line and the off-line optimizations