Self-Triggered Optimal Control Based on Path Search Algorithm

Yoshiki NAGATANI *, Kenji SAWADA **, and Seiichi SHIN *

Abstract: This paper proposes a path search formulation and its solution method to a finite optimal control problem on self-triggered control systems. Previous methods to the problem have high computational complexity. This paper focuses on the formulation and its data structure for reduction of calculation time and reformulates the optimal control problem to a path search problem. We also consider a data structure of a graph for the path search problem. The key point is the sharing of vertices. The sharing leads to the reduction of calculation time. We compare the calculation time of the path search algorithm and the mixed-logical dynamical method.

Key Words: self-triggered control systems, finite optimal control problem, path search, dynamic programming.

1. Introduction

On networked control systems, which consist of continuous dynamics and discrete dynamics, the self-triggered control law is a strategy for the reduction of communication frequency. This paper considers a finite optimal control problem on self-triggered control systems. There are three approaches to the problem. The first approach is based on continuous dynamics [1]–[3]. This approach can consider the stability of systems explicitly, whereas the optimality of events such as communication frequency is considered implicitly. The second approach is based on discrete dynamics, for example, the limited lookahead policy (LLP) control [4]. This approach can consider the optimality of events explicitly, whereas the stability of systems is considered implicitly. The third approach is based on both continuous and discrete characters [5],[6]. This approach considers the performance of systems and the optimality of events. We deal with this approach.

Hybrid systems can consider both dynamics. One method constructs a mixed-logical dynamical (MLD) [7] model and solves a mixed-integer quadratic programming (MIQP) problem obtained from the MLD model. However, there are two issues of MLD model. The first issue is that the construction of the MLD model is complex because the addition of auxiliary variables is needed. The second issue is that MIQP is known to be NP-hard and solving MIQP takes a long time.

This paper proposes a formulation and solution method of a finite optimal control problem on self-triggered control for the reduction of calculation time. We consider the optimization of a switching pattern on a switched system, in which the controller switches the sampling time. The state of the discrete dynamical model follows an automaton

\[ \delta \in \Sigma \times I \times \{0, 1\}, \]

where \( \Sigma = \{ \Sigma_1, \Sigma_2, \ldots, \Sigma_n \} \) is the set of symbols, \( I = \{0, 1\} \) is the set of inputs, and \( \delta \) is the state transition function. The controller observes the state \( x \) at time \( t_0, t_1, \ldots, t_k, \ldots \), and the holder keeps the control input value \( u \) between each sampling interval as follows:

\[ u(t) = -Kx(t), \forall t \in [t_k, t_{k+1}), \]

where \( K \in \mathbb{R}^{\Sigma \times I} \) is a state feedback gain.

We focus on the discrete dynamics of the controller that switches the sampling time. The state of the discrete dynamics is called a mode in this paper. The controller switches the mode following an automaton \( G \) shown in Fig. 2. That is, the relation between \( t_k \) and \( t_{k+1} \) follows the automaton \( G \) given by

\[ G = (\Sigma, I, \delta, s_{init}). \]

To solve this issue, we formulate the optimal control problem as a path search problem based on LLP. We use the tree in LLP, in which the tree is used for the representation of event sequences. The method enumerates all switching patterns with the tree and searches the optimal path on the tree. The second issue is what is the best data structure of the tree. The expression with the tree makes the number of vertices enormous, which prolongs the path search. To solve this issue, we consider the sharing of vertices. The sharing is based on zero-suppressed binary decision diagram (ZDD) [8], which is an efficient expression of binary trees. The sharing makes an adaption of dynamic programming easy and leads to the reduction of calculation time.

In the experiments, we inspect calculation time and optimality of the path search algorithm, comparing with those of the MLD method.

Notation: \( \mathbb{R} \) is the set of real numbers, \( \mathbb{R}_n \) is the set of non-negative real numbers and \( \mathbb{Z}_n \) is the set of non-negative integers. We write a positive definite matrix \( P \in \mathbb{R}^{n \times n} \) as \( P > 0 \).

2. Problem Settings

This paper considers a self-triggered switched system shown in Fig. 1. The plant \( P \) is a continuous time linear system given by

\[ \dot{x}(t) = A_c x(t) + B_c u(t) \]

where \( x(t) \in \mathbb{R}^n \) is a state vector, \( u(t) \in \mathbb{R}^m \) is a control input vector, \( A_c \in \mathbb{R}^{n \times n} \) and \( B_c \in \mathbb{R}^{n \times m} \) are constant matrices.

The samplers observe the state \( x \) at time \( t_0, t_1, \ldots, t_k, \ldots \), and the holder keeps the control input value \( u \) between each sampling interval as follows:

\[ u(t) = -Kx(t), \forall t \in [t_k, t_{k+1}), \]

where \( K \in \mathbb{R}^{n \times n} \) is a state feedback gain.

We focus on the discrete dynamics of the controller that switches the sampling time. The state of the discrete dynamics is called a mode in this paper. The controller switches the mode following an automaton \( G \) shown in Fig. 2. That is, the relation between \( t_k \) and \( t_{k+1} \) follows the automaton \( G \) given by

\[ G = (\Sigma, I, \delta, s_{init}). \]
where $\Sigma = \{\sigma_1, \ldots, \sigma_{i_m}\}$ is the set of events, $I = \{i_1, i_2, \ldots, i_M\}$ is the set of modes, $\delta : \Sigma \times I \to I$ is the state transition function, and $i_{init} \in I$ is the initial mode. An event $\sigma_i \in \Sigma \ (i \in I)$ means a switching to mode $i$. The controller can switch from any mode to any mode. That is, $\delta$ satisfies

$$\delta(\sigma, i) = i' \forall i \in I \forall i' \in I. \quad (4)$$

A sampled time $t_k$ depends on $\sigma \in \Sigma$. Define $h_\sigma \in \mathbb{R} \ (\forall \sigma \in \Sigma)$ as a sampling time of each mode. When the event $\sigma(k) = \sigma_{i_m} \in \Sigma$ is invoked at $t_k$, the controller switches the mode to $i_{init} = I(t_k)$, and then the relation between $t_k$ and $t_{k+1}$ is

$$t_{k+1} = t_k + h_{\sigma(k)}. \quad (5)$$

For the stability of the system, we assume that $h_\sigma (\forall \sigma \in \Sigma)$ satisfies the condition that there exists a matrix $P \in \mathbb{R}^{n \times n}$ satisfying

$$P > O, \quad P - \Phi(h_\sigma)\Phi(h_\sigma)^T > O, \quad (6)$$

where $\Phi(h_\sigma) = e^{A h_\sigma} - I$. This is a quadratic stability condition, and the switched system is stable exponentially if (6) is satisfied.

We consider a finite optimal control problem on the automaton $G$. In the problem, we find an optimal event sequence $s = \sigma(0) \sigma(1) \cdots \sigma(N-1) \in L(G)$, where $L(G)$ is the set of event sequences generated from $G$. In the self-triggered system, the controller switches the mode such that it minimizes the cost function $f$ from the initial time $t_0$ to the continuous terminal time $t_0 + H$:

$$f(s, x(t_0)) = \sum_{k=0}^{N-1} \left\{ \int_{t_k}^{t_{k+1}} x(t)^T Q x(t) dt + u(t_k)^T R u(t_k) + w_{\sigma(k)} \right\}$$

subject to $\sum_{k=0}^{N-1} h_{\sigma(k)} = H, \quad (7)$

where $N \in \mathbb{Z}_+$ is a discrete terminal time.

Equation (8) is a terminal time condition and $N$ which satisfies (8) depends on an event sequence. Therefore, $N$ cannot be given beforehand. The first, second, and third terms of (7) are costs to the state, the control input, and the mode, respectively. The state and control input cost (the first and second terms) includes the responses for each sampling interval. These terms become $\int_{t_0}^{t_0 + H} x(t)^T Q x(t) + u(t)^T R u(t) dt$ via (8).

The cost to the mode is for adjusting the balance between the control performance and the sampling frequency. When $w_{\sigma}$ is large compared with $Q$ and $R$, the sampling frequency is reduced while the control performance becomes worse. Conversely, the control performance is preceded when $w_{\sigma}$ is small compared with $Q$ and $R$. We have not considered the design method of the weights yet. That is future work.

Problem 1 can be solved with a mixed-integer quadratic programming (MIQP) obtained from a mixed-integer dynamical (MLD) model (cf. Appendix A). However, MIQP has high computational complexity. To solve this problem, we propose a path search formulation and its solution method in the next section.

### 3. Path Search Formulation

Our key idea is the tree representation of event sequences similar to limited lookahead policy (LLP control) [4]. The basic process of the proposed algorithm constructs a tree $G_t(G, H)$ which represents the set of event sequences $L(G)$ satisfying the terminal time condition (8) and searches an optimal event sequence. Figure 3 shows the structure of $G_t$ against Fig. 2. The tree $G_t$ is obtained from $G$ by tree search and consists of five elements:

$$G_t(G, N) = (\Sigma, Q_t, \delta_t, q_{init}^t, Qm^t), \quad (9)$$

where $\Sigma = \{\sigma_1, \ldots, \sigma_{i_m}\}$ is the set of events in $G$, $Q_t$ is a set of state, $\delta_t : \Sigma \times Q_t \to Q_t$ is a state transition function, $q_{init}^t \in Q_t$ is an initial state and $Qm^t \subseteq Q_t$ is a set of accept states. The state $q \in Q$ has information of time and mode. For the simplicity of notations, this paper introduces a structure representation for $q$.

The symbol $q.t \in \mathbb{R}$ means a time and $q.t \in I$ means a mode. The initial state $q_{init}$ and the set of accept states $Qm^t$ satisfy the conditions:

$$q_{init}^t.t = t_0, \quad q_{init}^t.t \in I_{init}, \quad q.t = H \forall q \in Qm^t. \quad (10)$$

The symbol $L_t(G)$ is the set of event sequences generated from $G_t$ and $Lw(G_t) \subseteq L_t(G)$ is a set of accepted event sequences. Each event sequence of $Lw(G_t)$ satisfies the terminal condition. Problem 1 is converted to Problem 2, in which we find an optimal event sequence $s \in Lw(G_t)$.
Problem 2 Suppose that weight matrices \( \overrightarrow{Q}_r \in \mathbb{R}^{n \times n} \), \( R \in \mathbb{R}^{m \times n} \), a scalar weight \( w_r \in \mathbb{R} \) for each \( \sigma \in \Sigma \), an initial state \( x(t_0) \), an automaton \( G_t(G, H) \) are given. Find an event sequence \( s = \sigma(0) \ldots \sigma(N - 1) \in L_m(G_t) \) which minimizes the cost function

\[
f(s, x(t_0)) = \sum_{k=0}^{N-1} C(x(t_k), \sigma(k)),
\]

where

\[
C(x(t_k), \sigma(k)) := x(t_k)^T \overrightarrow{Q}_r x(t_k) + u(t_k)^T R t(t_k) h_{r(\sigma(k))} + w_{r(\sigma(k))}.
\]

The weight matrix \( \overrightarrow{Q}_r \) is given by

\[
\overrightarrow{Q}_r = \int_0^{h_{0r}} \Phi(t)^T \overrightarrow{Q} \Phi(t) dt. \tag{11}
\]

The first term \( x(t_k)^T \overrightarrow{Q}_r x(t_k) \) in the cost \( C(x(t_k), \sigma(k)) \) is a discrete representation of the cost function \( f(x, x(t_0)) \) is a sum of the weights on a path. Then, the problem is a shortest path search problem in which the weights of edges are dynamical.

To solve Problem 2, we need to consider the structure of \( G_t \) and the path search algorithm.

### 3.1 Data Structure of Solution Space

The representation of the set of event sequences \( L_m(G_t) \) becomes a solution space of Problem 2. That means Problem 2 is a path search problem on \( G_t \), whereas Problem 1 is a path search problem on \( G \). In Problem 2, the cost function \( C(x(t_k), \sigma(k)) \) is a weight of edge and the cost function \( f(x, x(t_0)) \) is a sum of the weights on a path. Then, the problem is a shortest path search problem in which the weights of edges are dynamical.

To solve Problem 2, we need to consider the structure of \( G_t \) and the path search algorithm.
at the same time. Lines 10–13 describe the sharing. Line 10 decides whether a new state $q'$ should be added to $Q_s$. If $q'$ already exists in $Q_s$, line 14 adds only an edge to $q'$.

The graph $G_s$ is a directed acyclic graph (DAG) against the tree $G_t$. A DAG allows us to adopt dynamic programming to the path search algorithm easily.

### 3.2 Path Search Algorithm

Our path search algorithm based on dynamic programming solves subproblems repeatedly. We show the subproblem on the proposed algorithm. Define $G_{q_j}$, $q_j \in Q_s$, as an automaton whose set of accepted event sequences $L_m(G_{q_j})$ includes all the event sequences from $q_j^{init}$ to $q_j$. Each $q_j$ in $G_{q_j}$ is an accept state. Define $x_{q_j}(s) \in \mathbb{R}^s$, $s \in L_m(G_{q_j})$, as a state vector at $q_j$. The subproblem in this paper is the optimal path search problem from $q_j^{init}$ to each $q_j$:

$$\min_{s \in L_m(G_{q_j})} f(s, x(t_0)) = \min_{s \in L_m(G_{q_j})} \sum_{k=0}^{n} C(x(t_k), \sigma(k)).$$

(16)

where $s \in L_m(G_{q_j})$ is an event sequence to the state $q_j$ and $n$ is a discrete terminal time.

In Fig. 6, there are $j$ edges connected to $q_0$. The subproblem to $q_0$ can be divided into $j$ problems:

$$\min_{s \in L_m(G_{q_j})} f(s, x(t_0)) = \min_{s \in L_m(G_{q_j})} \left\{ \min_{s \in L_m(G_{q_j})} f(s, x(t_0)), \ldots, \min_{s \in L_m(G_{q_j})} f(s, x(t_0)) \right\}.$$  

(17)

Each term of the right side of (17) can be divided into

$$\min_{s \in L_m(G_{q_j})} f(s, x(t_0))$$

(18)

For adaption of dynamic programming, we need to isolate the right side of (18) into

$$\min_{s \in L_m(G_{q_j})} f(s, x(t_0)) = \min_{s \in L_m(G_{q_j})} \left( f(s, x(t_0)) + C(x_{q_j}(s), \sigma_j) \right).$$

(19)

However, $x_{q_j}(s)$ in the term $C(x_{q_j}(s), \sigma_j)$ in (18) depends on $s_j$ and disturbs the isolation.

For the isolation, we introduce an assumption

$$\min_{s \in L_m(G_{q_j})} C(x_{q_j}(s), \sigma_j) = C(x_{q_j}(\sigma_j), \sigma_j).$$

(20)

where $\sigma_j$ is a solution of $\min_{s \in L_m(G_{q_j})} f(s, x(t_0))$. The weight of edge $C$ from $q_j$ to $q_0$ depends on the path from $q_j^{init}$ to $q_j$, and the assumption means the optimal path from $q_j^{init}$ to $q_j$ also optimizes the weight $C(x_{q_j}(s), \sigma_j)$ from $q_j$ to $q_0$. An introduction of the assumption (20) makes (17) into

$$\min_{s \in L_m(G_{q_j})} f(s, x(t_0)) = \min_{s \in L_m(G_{q_j})} \left\{ \min_{s \in L_m(G_{q_j})} f(s, x(t_0)) + C(x_{q_j}(\sigma_j), \sigma_j) \right\}.$$  

(21)

Equation (21) is a recurrence relation about vertices. Our path search algorithm implements (21). The proposed algorithm based on (21) finds a suboptimal solution efficiently.

We show the concrete process of the algorithm. The search is based on a breadth-first search. For the management of a path in the algorithm, define $p := (s, f, q, x)$ where $s$ is an event sequence, $f \in \mathbb{R}$ is a cost function value of the path, $q \in Q_s$ is a final state of the path and $x$ is a state at $q$.

Algorithm 2 is a pseudo code of the path search algorithm. We explain the basic process of Algorithm 2. Figure 7 shows each process of the algorithm. The first step is moving to the next vertices and the calculation of the cost function and next vertices. The first step is moving to the next vertices and the calculation of the cost function and next vertices. The first step is moving to the next vertices and the calculation of the cost function and next vertices. The first step is moving to the next vertices and the calculation of the cost function and next vertices.
Algorithm 2 Search of an Optimal Event Sequence

**Inputs:** \( G_s = (\Sigma, Q_s, \delta_s, q_0^s, Q_f^s), x(t_0) \)

**Outputs:** A optimal path \( p_{\text{min}} \)

1: \( X_2 \leftarrow \text{a queue} \)
2: \( P(q_i) \leftarrow \text{a set of } p \) (\( q_i \in Q_s \))
3: \( P_{\text{final}} \leftarrow \text{a set of } p \)
4: Enqueue \( p_0 = (q_0, x(t_0)) \) to \( X_2 \)
5: while \( X_2 \) is not empty do
6: \( p = (s, f, q_i, x) \) dequeue from \( X_2 \)
7: for all \( \sigma \in \Sigma \) do
8: \( s_{\text{next}} \leftarrow s \sigma \)
9: \( f_{\text{next}} \leftarrow f + C(x, \sigma) \)
10: \( q_{\text{next}} \leftarrow \delta_s(\sigma, q_i) \)
11: \( x_{\text{next}} \leftarrow \Phi(h_\sigma, x) \)
12: Add \( \left(s_{\text{next}}, f_{\text{next}}, q_{\text{next}}, x_{\text{next}}\right) \) to \( P(q_i) \)
13: end for
14: if \( |P(q_i)| = \text{indegree of } q_i \) then
15: \( \text{Enqueue } p_{\text{next}} \) to \( P_{\text{final}} \)
16: end if
17: end while
18: return \( p_{\text{min}} \in P_{\text{final}} \) whose \( f \) is the minimum

Fig. 7 The process of path search algorithm.

one. Therefore, the solid line is regarded as the solution of the subproblem to the vertex. Line 15 is the implementation of (21). The snippet of lines 14–21 is the dynamic programming in this paper. The algorithm searches to the accept states and finally outputs an optimal path.

This algorithm searches all edges of \( G_s \) just only one time. Therefore, the time complexity is \( O(E) \) (\( E \) is the number of edges).

The assumption (20) can make an influence on the optimality. We inspect the optimality in the experiment.

4. Numerical Experiments

4.1 Sharing of Vertices

We show the effectiveness of the sharing. We consider the automaton \( G \) shown in Fig. 2, which includes three modes. The sampling time of each mode is \( h_{r_1} = 0.2 \) s, \( h_{r_2} = 0.1 \) s, \( h_{r_3} = 0.05 \) s. The terminal time \( H \) is 0.5 s. Figures 8 and 9 show \( G_s \) and \( G_{s'} \) obtained from \( G \). The number of the vertices reduce from 392 to 27. This reduction will make the path search fast.

In the next section, we show the calculation time of the path search algorithm running on \( G_s \).

4.2 Calculation Time

This section compares the calculation time of the MLD method and the path search algorithm. We consider two types of experiments. The first type is to increase the terminal time \( H \) and the second type is to increase the number of modes.

We consider a linear system (1) where \( A_c = \begin{bmatrix} 0 & 1 \\ -2 & 3 \end{bmatrix} \) and \( B_c = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \) and give the state feedback \( K \) with the pole placement method. The pole \( p \) is \(-2, -3\). The other parameters are \( x(t_0) = \begin{bmatrix} 3 \\ 3 \end{bmatrix} \), \( Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \), \( R = 0.01 \), \( w_{r_1} = w_{r_2} = w_{r_3} = 2.5 \).

In the first experiment, we consider the 3 modes system. The automaton \( G \) and the sampling time of each mode is the same as Section 4.1. The terminal time \( H \) is 0.2 s, 0.4 s, ... , 1.6 s. The MIQP problem obtained from the MLD method is solved with Gurobi 8.00 and the path search algorithm is implemented with MEX on MATLAB R2018a. The CPU is Intel(R) Xeon(R)
CPU E5-2603 v4 @ 1.70 GHz (two processors) and the capacity of the memory is 32.0 GB.

Figure 10 shows the result. In Fig. 10, the calculation time of the MLD method is a runtime of Gurobi and the calculation time of the path algorithm is a runtime of Algorithm 2. From the result, the time of the MLD method increases exponentially with the terminal time $H$. On the other hand, the time of path search increases linearly. Figure 11 shows the number of edges on each problem. The number of edges increases linearly. The time complexity of the proposed path search algorithm is proportional to the number of edges. Then, the calculation time also increases linearly.

In the second experiment, the numbers of modes are $3, 4, \ldots, 10$. The mode follows the complete graph. For the simplicity of the second experiment, the sampling time of all mode is $0.01$ s. The weight $w_\sigma (\forall \sigma \in \Sigma)$ is 50. The terminal time $H$ is 0.1 s. The other parameters are equal to the first experiment.

Figure 12 shows the result. The calculation time of the path search algorithm increases exponentially. The number of edges in Fig. 13 also increases. This is because the combination of modes increases exponentially.

From the two experiments, the proposed algorithm is efficient especially to the increase of the terminal time.

### 4.3 Optimality

To evaluate the optimality, we change the imaginary part of the pole $p$ and compare the solutions with the MLD method. The plant $P$ is the same as the plant in Section 4.2. The graph $G$ and the sampling time of each mode are the same as Section 4.1. The continuous terminal time $H$ equals 2 s. The parameters $Q, R$, and $w_\sigma$ of the cost function are the same as Section 4.2. The sets of poles are shown in Table 1. The symbol $f_{\text{MLD}}$ and $f_{\text{Path}}$ in Table 1 are the minimum cost function values by the two methods and $f_{\text{Path}} - f_{\text{MLD}}$ is an error rate of the two cost functions. From the result, there are differences of solutions by the two methods and the solution with the path search algorithm is worse than that of the MLD method if the system response is vibrational. Figures 14, 15, and 16 show the response $x$, the control input $u$, and the mode transition of each method ($p = -3 + 8j, -3 - 8j$). From Fig. 14, both of the control performance are similar. The convergence of $x$ by the path search algorithm is a little late than that of the MLD method.

To clarify the difference between the two methods, we focus on the three points (a), (b), and (c) on the mode transitions in Fig. 16. Figure 16 shows the mode transitions of the two methods. Both solutions of the two methods pass at the point (a) in Fig. 16. Each function value of the MLD method and the path search algorithm at (a) is about 167 and 164. The path search algorithm is better at this moment. However, each value at the terminal time, (b) and (c) is about 229 and 234. The MLD method is better finally. For the adaption of dynamic programming, we forcibly divide the problem with time (cf. (21)). The path search algorithm (Algorithm 2) based on dynamic programming finds an optimal event sequence which satisfies (21).

![Fig. 10 Calculation time and terminal time $H$.](image1)

![Fig. 11 Number of edges and terminal time $H$.](image2)

| $p_1, p_2$ | $f_{\text{MLD}}$ | $f_{\text{Path}}$ | $\% \Delta f_{\text{MLD}}/f_{\text{MLD}}$ |
|------------|------------------|------------------|-----------------------------------|
| $-3, -3.1$ | 44.0             | 44.0             | 0%                                |
| $-3 + 2j, -3 - 2j$ | 48.9             | 48.9             | 0%                                |
| $-3 + 4j, -3 - 4j$ | 71.6             | 71.6             | 0%                                |
| $-3 + 6j, -3 - 6j$ | 119.5            | 119.8            | 0.19%                             |
| $-3 + 8j, -3 - 8j$ | 229.0            | 234.3            | 2.3%                              |
| $-3 + 10j, -3 - 10j$ | 463.7            | 524.8            | 13%                               |

![Fig. 12 Calculation time and number of modes $M$.](image3)

![Fig. 13 Number of edges and number of modes $M$.](image4)
Fig. 14 The response $x$.

Fig. 15 The control input $u$.

Fig. 16 Difference of mode transitions.

Fig. 17 Error transitions.

Note that the optimal event sequence may not satisfy the original problem (17) and is a suboptimal solution of Problem 2.

Figure 17 shows the error transition of the cost function values. From the figure, the error is unstable. Vibrational systems often cause this phenomenon. Therefore, solutions of subproblems may not be a part of the whole problem, especially in vibrational systems. That is, the path search algorithm provides suboptimal solutions.

5. Conclusion

This paper proposed the path search formulation and its solution method of the finite optimal control problem on self-triggered control systems. From the results of experiments, the proposed algorithm calculates a suboptimal solution efficiently, especially to the increase of terminal time. The experiments show the calculation time of the path search algorithm is faster than that of the MLD method.

We will consider simultaneous optimization of the control input and the sampling pattern.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Number 17H06293, 15K06134, and 26420429.

References

[1] W.P.M.H. Heemels, K.H. Johansson, and P. Tabuada: An introduction to event-triggered and self-triggered control, 51st IEEE Conference on Decision and Control, pp. 3270–3285, 2012.

[2] G.S. Deaecto, M. Souza, and J.C. Geromel: Discrete-time switched linear systems state feedback design with application to networked control, IEEE Transactions on Automatic Control, Vol. 60, No. 3, pp. 877–881, 2015.

[3] T. Gommans, D. Antunes, T. Donkers, P. Tabuada, and M. Heemels: Self-triggered linear quadratic control, Automatica, Vol. 50, No. 4, pp. 1279–1287, 2014.

[4] S.-L. Chung, S. Lafortune, and F. Lin: Limited lookahead policies in supervisory control of discrete event systems, IEEE Transactions on Automatic Control, Vol. 37, No. 12, pp. 1921–1935, 1992.

[5] K. Kobayashi and K. Hiraishi: Event-triggered and self-triggered control for networked control systems using online optimization, IEICE Transactions on Fundamentals of Electronic Communications and Computer Sciences, Vol. E99-A, No. 2, pp. 468–474, 2016.

[6] S. Nakano and T. Ushio: Self-triggered predictive control with time-dependent activation costs of mixed logical dynamical systems, IEICE Transactions on Fundamentals of Electronic Communications and Computer Sciences, Vol. E97-A, No. 2, pp. 476–483, 2014.

[7] A. Bemporad and M. Morari: Control of systems integrating logic, dynamics, and constraints, Automatica, Vol. 35, No. 3, pp. 407–427, 1999.

[8] S. Minato: Zero-suppressed BDDs for set manipulation in combinatorial problems, Proceedings of the 30th International Design Automation Conference, pp. 272–277, 1993.

[9] P.J. Ramadge and W.M. Wonham: Supervisory control of a class of discrete event processes, SIAM Journal on Control and Optimization, Vol. 25, No. 1, pp. 206–230, 1987.

[10] J.P. Hespanha, P. Naghshtabrizi, and Y. Xu: A survey of recent results in networked control systems, Proceedings of the IEEE, Vol. 95, No. 1, pp. 138–162, 2007.

Appendix A MLD Model of the System

For the simplicity of notations, we introduce discrete time representation for each symbol,

$$x[k] = x(t_k), \quad u[k] = u(t_k).$$

In this appendix, $O_{even}$ is an $n \times m$ zero matrix, and $E_n$ is an $n \times n$ identity matrix. The MLD model [7] of the self-triggered switched system is derived by defining the next two symbols:

$$\delta_i[k] \in \{0, 1\} \quad \forall i \in I, \quad (A.1)$$
The binary variable $\delta_i$ represents a mode. If the mode $i \in I$ is (non-) active at $k$, $\delta_i[k] = 1(0)$. The auxiliary variable $z_i$ is a state vector of each mode. Equation (A.2) is represented by the following linear inequalities:

$$x_{\text{min}} \leq z_i[k] \leq x_{\text{max}}, \quad (A.3)$$

$$x[k] - x_{\text{min}}(1 - \delta_i[k]) \leq z_i[k] \leq x[k] - x_{\text{min}}(1 - \delta_i[k]), \quad (A.4)$$

where $x_{\text{min}} \in \mathbb{R}^{\Omega_0}$ and $x_{\text{max}} \in \mathbb{R}^{\Omega_0}$ are a lower and upper bound of the state region, respectively. The switched system is given by

$$\Phi(t)x[k] = \begin{bmatrix} e^{A_1} & \cdots & e^{A_m} \\ \vdots & \ddots & \vdots \\ e^{A_M} & \cdots & e^{A_1} \end{bmatrix} x[k]. \quad (A.7)$$

The MIQP problem is obtained from the MLD model (A.8)-(A.10). Discretization of the cost function is needed (cf. Appendix B). Details are omitted in this paper. Note that $N$ in Problem 1 cannot be obtained beforehand. To solve the issue, we set sufficiently large discrete terminal time $N'$ beforehand. The discrete terminal time $N'$ is given by

$$N' = \left\lfloor \frac{H}{\min(h_1, \ldots, h_M)} \right\rfloor. \quad (A.11)$$

because $N$ is less than $N'$ of (A.11) at least.

Problem 3 Suppose that the initial state $x[0] \in \mathbb{R}^{\Omega_0}$ and the continuous terminal time $H \in \mathbb{R}$, are given. The discrete terminal time $N' \in \mathbb{Z}_+$ is given by (A.11). Find a mode transition $\delta[0], \delta[1], \ldots, \delta[N' - 1]$ which minimizes the cost function

$$f((z[k], \delta[k])_{k=0,1,\ldots,N'-1}) = \sum_{k=0}^{N'-1} [z[k]^T(Q' + R')z[k] + W\delta[k]] \quad (A.12)$$

subject to

$$\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} \delta[0] = 1, \quad (A.9)$$

$$Cz[k] + D_1\delta[k] + D_2\delta[k] \leq D_3, \quad (A.10)$$

where

$$z[k] := [z_1^T[k] \cdots z_{M_1}^T[k]]^T, \quad \delta[k] := [\delta_1[k] \cdots \delta_M[k]]^T,$$

$$B := [\Phi(h_1) \cdots \Phi(h_M)]^T, \quad M := |I|,$$

$$C := \begin{bmatrix} O_{2n0} \\ E_n \\ -E_n \\ \vdots \end{bmatrix}, \quad D_1 := \begin{bmatrix} D_1' & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_1' \end{bmatrix},$$

$$D_2 := \begin{bmatrix} -E_n \\ E_n \\ -E_n \end{bmatrix}, \quad D_3 := \begin{bmatrix} -x_{\text{min}} & -x_{\text{max}} & -x_{\text{min}} \\ x_{\text{max}} & x_{\text{max}} & x_{\text{max}} \\ -x_{\text{min}} & -x_{\text{min}} & -x_{\text{min}} \end{bmatrix},$$

$$O_{2n0} \in \mathbb{R}^{2n_0}, \quad E_n \in \mathbb{R}^{n_0 \times M}, \quad -E_n \in \mathbb{R}^{n_0 \times M},$$

$$\begin{bmatrix} h_1R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & h_MR \end{bmatrix} \begin{bmatrix} -K & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -K \end{bmatrix} = H,$$

$$\begin{bmatrix} h_1R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & h_MR \end{bmatrix} \begin{bmatrix} -K & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -K \end{bmatrix} = -K.$$
From the solution of Problem 3, \( N \) is obtained by
\[
N = \sum_{k=0}^{N-1} \sum_{i=1}^{M} \delta_i[k]. \quad \text{(A. 14)}
\]
We use \( \delta[0], \ldots, \delta[N-1], \ N \leq N' \) as a solution to Problem 1.

### Appendix B Discretization of Cost Function

The solution of the system (1) is given as the following:
\[
x(t) = e^{A(t-t_k)} x(t_k) + \int_{t_k}^{t} e^{A(t-\tau)} B \Phi(t-\tau) Q x(t_k) d\tau. \quad \text{(B. 1)}
\]
The control input is given by (2). Substituting (2) to (B. 1), we have
\[
x(t) = e^{A(t-t_k)} x(t_k) - \int_{t_k}^{t} e^{A(t-\tau)} B \Phi(t-\tau) Q x(t_k) d\tau
\]
\[
= \Phi(t-t_k) x(t_k), \quad \text{(B. 2)}
\]
where \( \Phi(t) := \int_0^t e^{A(t-\tau)} B \Phi(t-\tau) d\tau \). Discrete time expression of \( \int_{t_k}^{t} Q x(t) dt \) is obtained by substituting (B. 2) into the cost function:
\[
\int_{t_k}^{t_k+h_k} x(t) Q x(t) dt
\]
\[
= \int_{t_k}^{t_k+h_k} x(t_k)^T \Phi(t-t_k) Q \Phi(t-t_k) x(t_k) dt
\]
\[
= x(t_k)^T \int_{t_k}^{t_k+h_k} \Phi(t-t_k) Q \Phi(t-t_k) dx(t_k)
\]
\[
= x(t_k)^T \Phi_{tr} x(t_k) \quad \text{(B. 3)}
\]
where \( \Phi_{tr} \) is given by (11).

---

**Yoshiki Nagatani**

He received the B.E. degree in mechanical engineering from the University of Electro-Communications, Tokyo, Japan, in 2017. He is currently pursuing the M.E. degree at the University of Electro-Communications. His research interests include theory of networked control especially self-triggered and event-triggered control.

**Kenji Sawada (Member)**

He received his Ph.D. degree in engineering in 2009 from Osaka University. He is an Associate Professor in Info-Powered Energy System Research Center, the University of Electro-Communications, Japan. He is also an advisor of Control System Security Center since 2016. He received Outstanding Paper Awards from FA foundation and Fluid Power Technology Promotion Foundation in 2015 and 2018. His research interests include control theory of cyber-physical system and control system security. He is a member of ISCIE, IEICE, IEEJ, IEEE.

**Seiichi Shin (Member, Fellow)**

He received D.E. in 1987 from the University of Tokyo. Since 2006, He is a professor of the University of Electro-Communications through the University of Tokyo and the University of Tsukuba. He is also a president of Control System Security Center since 2012. He received Outstanding Paper Awards from SICE in 1991, 1992, 1993 and 1998 including the Takeda Prize. He received a Minister of Economy, Trade and Industry Award in 2018. He is interested in control theory and its application.