
HOKKAIDO UNIVERSITY

Title	Computationally Efficient Model Predictive Control for Multi-Agent Surveillance Systems
Author(s)	KOBAYASHI, Koichi; KIDO, Mifuyu; YAMASHITA, Yuh
Citation	IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E102.A(2), 372-378 https://doi.org/10.1587/transfun.E102.A.372
Issue Date	2019-02-01
Doc URL	http://hdl.handle.net/2115/90182
Rights	copyright©2019 IEICE
Type	article
File Information	Computationally Efficient Model Predictive Control for Multi-Agent Surveillance Systems.pdf


Instructions for use

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

IEICE TRANS. FUNDAMENTALS, VOL.E102–A, NO.2 FEBRUARY 2019

PAPER Special Section on Mathematical Systems Science and its Applications

Computationally Efficient Model Predictive Control for Multi-Agent Surveillance Systems

Koichi KOBAYASHI[†a)], ***Member***, **Mifuyu KIDO**[†], ***Nonmember***, ***and*** **Yuh YAMASHITA**[†], ***Member***

SUMMARY In this paper, a surveillance system by multiple agents, which is called a multi-agent surveillance system, is studied. A surveillance area is given by an undirected connected graph. Then, the optimal control problem for multi-agent surveillance systems (the optimal surveillance problem) is to find trajectories of multiple agents that travel each node as evenly as possible. In our previous work, this problem is reduced to a mixed integer linear programming problem. However, the computation time for solving it exponentially grows with the number of agents. To overcome this technical issue, a new model predictive control method for multi-agent surveillance systems is proposed. First, a procedure of individual optimization, which is a kind of approximate solution methods, is proposed. Next, a method to improve the control performance is proposed. In addition, an event-triggering condition is also proposed. The effectiveness of the proposed method is presented by a numerical example.

key words: *mixed integer programming, model predictive control, multiple agents, persistent surveillance*

1. Introduction

The persistent surveillance problem that computes trajectories of agents to patrol a given area has many applications such as city safety management and disaster rescue. In particular, persistent surveillance by multiple agents is important. In this paper, a surveillance system by multiple agents is called a *multi-agent surveillance system*. The control problem of multi-agent surveillance systems has been studied in e.g., [1], [5], [6], [8], [15].

In this paper, a multi-agent surveillance system over an undirected connected graph is studied. In control of complex systems, it is appropriate to approximately solve the problem by using discrete abstraction techniques (see, e.g., [2], [16], [17]). In the surveillance problem, modeling a surveillance area as a graph is one of the abstraction methods. The surveillance problem over a graph has been studied in e.g., [1], [8]. Our previously proposed method in [8] is based on the MLD (mixed logical dynamical) framework [3] and model predictive control (MPC) [4], [14]. For each node in the surveillance area, a time-varying penalty is given. Penalties and agents are modeled by an MLD system model consisting of a linear state equation and a linear inequality constraint with respect to continuous and binary variables. Using the MLD system model, we can easily impose several constraints such as fuel constraints. The finite-time surveillance problem is reduced to a mixed integer linear programming (MILP) problem. In the conventional MPC method, the control input is generated by solving the finite-time optimal control problem at each discrete time. When MPC is applied to the surveillance problem, the next location of each agent can be obtained by solving the MILP problem at each discrete time. However, this method has the following two weaknesses: i) the computation time for solving the MILP problem exponentially grows with respect to the number of agents, ii) the MILP problem must be solved at each discrete time.

In this paper, a new MPC method for multi-agent surveillance systems is proposed as an improved version of the method in [8]. First, we propose a procedure of individual optimization. In this method, the MILP problem for m agents is decomposed to m MILP problems for a single agent. Since the MILP problem for a single agent can be solved fast, reducing the overall computation time can be achieved. However, in this method, m MILP problems must be solved sequentially, and it is necessary to determine the order of agents.

Next, we propose a method for determining the priority of agents. In this method, we focus on the neighborhood of each agent. Based on penalties of nodes in the neighborhood, the priority of agents is determined. Using this method, the approximate solution by individual optimization is improved.

Finally, we introduce an event-triggering condition on penalties of nodes. Event-triggered control is a method that the control input is updated only when an event-triggering condition is satisfied (see, e.g., [7]). In event-triggered MPC, the finite-time optimal control is solved (see, e.g., [13]). In this paper, the finite-time surveillance problem is solved only when the sum of penalties at the next time is equal to or greater than a certain threshold. By choosing the threshold appropriately, we can consider the trade-off between the control performance and the number of times that the problem is solved. The effectiveness of the proposed method is presented by a numerical example based on several situations.

Notation: Let $\mathcal{R}$ denote the set of real numbers. Let $\{0, 1\}^n$ denote the set of n-dimensional vectors, which consists of elements 0 and 1. Let 1_n denote the n-dimensional vector whose elements are all one. Let I_n and $0_{m\times n}$ denote the $n \times n$ identity matrix and the $m \times n$ zero matrix, respectively. For simplicity of notation, we sometimes use the symbol 0 instead of $0_{m\times n}$, and the symbol I instead of I_n. For the matrix M, let $M^\top$ denote the transpose of M.

Manuscript received April 15, 2018.
Manuscript revised August 10, 2018.

[†]The authors are with the Graduate School of Information Science and Technology, Hokkaido University, Sapporo-shi, 060-0814 Japan.
a) E-mail: k-kobaya@ssi.ist.hokudai.ac.jp

DOI: 10.1587/transfun.E102.A.372

Copyright © 2019 The Institute of Electronics, Information and Communication Engineers

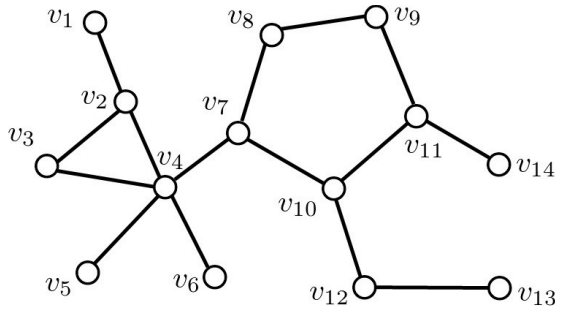

Fig. 1 Example of undirected connected graphs, where self-loops in all nodes are omitted.

2. Optimal Surveillance Problem and Model Predictive Control

In this section, we formulate the optimal surveillance problem (see, e.g., [8]) and the model predictive control (MPC) method (see, e.g., [4], [14]).

A surveillance area is given by an undirected connected graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, and $E \subseteq V \times V$ is the set of arcs. We assume that an agent can move according to a given graph, and behavior of an agent is expressed by a discrete-time system. The number of agents is given by m. As an example, consider the undirected connected graph in Fig. 1, where self-loops in all nodes are omitted. Suppose that the initial location of an agent is given by v_4. Then, the candidates of the locations at the next time are constrained to the set $\{v_2, v_3, v_4, v_5, v_6, v_7\}$. Thus, a complicated surveillance area can be modeled by an undirected connected graph. In this paper, the system that multiple agents move on a surveillance area given by an undirected connected graph is called a *multi-agent surveillance system*.

For each node, the penalty $x_i(k) \in \mathcal{R}$ is defined as follows:

$$x_i(k+1) = \begin{cases} 0 & \text{if the agent is located on } v_i \text{ at } k, \\ x_i(k) + 1 & \text{otherwise,} \end{cases} \tag{1}$$

where $k \in \{0, 1, 2, \ldots\}$ is discrete time.

Then, the optimal surveillance problem is formulated as follows.

Problem 1: For the undirected connected graph $G = (V, E)$ and the time evolution (1) of the penalty, suppose that the current locations of agents, the current penalty $x_i(t)$ (t is the current time), and the prediction horizon N are given. Then, find trajectories of m agents minimizing the following cost function:

$$J = \sum_{k=t}^{t+N} \sum_{i=1}^{n} q_i x_i(k), \tag{2}$$

where $q_i \geq 0$ is a given weight.

If there exists an unimportant node, then the corresponding q_i is given by $q_i = 0$. We may impose a constraint such as $x_i(k) \leq \sigma$, where $\sigma > 0$ is a given scalar. Temporal logic constraints can also be imposed for Problem 1 (see e.g., [9], [10], [12]). In this paper, for simplicity of discussion, we consider the case where no constraints are imposed.

Next, we present the MPC method for multi-agent surveillance systems.

MPC for Multi-Agent Surveillance Systems:

Step 1: Set $t = 0$, and give $x_i(0)$ (the initial penalty for each node) and the initial location for each agent.

Step 2: Solve Problem 1.

Step 3: Move agents based on the computation result.

Step 4: Update $t := t + 1$, and return to Step 2.

Consider the single agent case as a simple example. Suppose that a surveillance area is given by Fig. 1. Suppose also that q_i in the cost function (2), the initial location of an agent, and the initial penalty $x_i(0)$ are given by $q_i = 1$, v_4, and $x_i(0) = 0$, respectively. At time 0, $x_i(1)$ can be obtained as $x_4(1) = 0$ and $x_i(1) = 1$, $i = 1, 2, 3, 5, 6, \ldots, 14$. We assume that the next location is obtained as v_7 by solving Problem 1 at time 0. At time 1, $x_i(2)$ can be obtained as $x_7(2) = 0$, $x_4(2) = 1$, and $x_i(2) = 2$, $i = 1, 2, 3, 5, 6, 8, 9, \ldots, 14$. By solving Problem 1, the location at time 2 can be obtained. By repeating this procedure, the optimal trajectory can be obtained.

3. Reduction of Problem 1 to an MILP problem

In this section, we summarize the reduction of Problem 1 to an MILP problem. See [8] for further details.

Consider m agents that move on the undirected connected graph $G = (V, E)$. The binary variable $\delta_{i,j}(k)$ is defined by

$$\delta_{i,j}(k) = \begin{cases} 1 & \text{if the agent } j \text{ is located on the node } v_i, \\ 0 & \text{otherwise,} \end{cases}$$

where the following constraint:

$$\sum_{i=1}^{n} \delta_{i,j}(k) = 1 \tag{3}$$

must be imposed. In addition, we define

$$\bar{\delta}_j(k) = [\delta_{1,j}(k)\ \delta_{2,j}(k)\ \cdots\ \delta_{n,j}(k)]^\top.$$

Let Φ denote the adjacency matrix of G. Then, the time evolution of agent j can be modeled by

$$\bar{\delta}_j(k) - \Phi \bar{\delta}_j(k+1) \leq 0. \tag{4}$$

See [11] for further details.

Next, consider the dynamics of the penalty defined by (1). Here, we define

$$\delta_i(k) = \begin{cases} 1 & \text{if at least one agent is located on } v_i \\ & \text{at } k, \\ 0 & \text{otherwise.} \end{cases}$$

We also define $\delta(k) := [\delta_1(k)\ \delta_2(k)\ \cdots\ \delta_n(k)]^\top$. The relation between $\delta_{i,j}(k)$ and $\delta_i(k)$ is given by the following linear inequalities:

$$\delta_{i,1}(k) \le \delta_i(k) \le \sum_{j=1}^{m} \delta_{i,j}(k), \tag{5}$$

$$\delta_{i,2}(k) \le \delta_i(k) \le \sum_{j=1}^{m} \delta_{i,j}(k), \tag{6}$$

$$\vdots$$

$$\delta_{i,m}(k) \le \delta_i(k) \le \sum_{j=1}^{m} \delta_{i,j}(k). \tag{7}$$

Using $\delta_i(k)$, the penalty $x_i(k)$ is modeled by

$$x_i(k+1) = (1-\delta_i(k))(x_i(k)+1). \tag{8}$$

The lower bound of $x_i(k)$ is 0, and the upper bound of $x_i(k)$ is given by $x_{\max} < \infty$ ($x_{\max}$ can be determined from a given graph). Then, $z_i(k) := \delta_i(k)x_i(k) - 1$ is equivalent to the following linear inequalities:

$$\begin{cases} -1 \le z_i(k) \le x_{\max}\delta_i(k) - 1, \\ x_i(k) - x_{\max}(1-\delta_i(k)) - 1 \le z_i(k) \le x_i(k) - 1. \end{cases} \tag{9}$$

See [3] for further details.

From the above results, the time evolutions of agents and penalties can be modeled by (3)–(9), i.e., the following mixed logical dynamical (MLD) system model [3]:

$$\begin{cases} x(k+1) = Ax(k) + Bv(k), \\ Cx(k) + Dv(k) \le E, \end{cases} \tag{10}$$

where

$$\begin{aligned} x(k) &= [x_1(k)\ x_2(k)\ \cdots\ x_n(k) \\ &\qquad \bar{\delta}_1^\top(k)\ \bar{\delta}_2^\top(k)\ \cdots\ \bar{\delta}_m^\top(k)\ ]^\top \in \mathcal{R}^n \times \{0,1\}^{nm}, \\ v(k) &= [z_1(k)\ z_2(k)\ \cdots\ z_n(k) \\ &\qquad \delta^\top(k)\ \bar{\delta}_1^\top(k+1)\ \bar{\delta}_2^\top(k+1)\ \cdots\ \bar{\delta}_m^\top(k+1)\Big]^\top \\ &\qquad \in \mathcal{R}^n \times \{0,1\}^{n+nm}, \\ A &= \begin{bmatrix} I_n & 0_{n\times nm} \\ 0_{nm\times n} & 0_{nm\times nm} \end{bmatrix}, \\ B &= \begin{bmatrix} -I_n & -I_n & 0_{n\times nm} \\ 0_{nm\times n} & 0_{nm\times n} & I_{nm} \end{bmatrix}, \\ C &= \left[\begin{array}{cc} 0_{nm\times n} & I_{nm} \\ \hline 0_{n\times n} & 0_{n\times nm} \\ 0_{n\times n} & 0_{n\times nm} \\ I_n & 0_{n\times nm} \\ -I_n & 0_{n\times nm} \\ \hline 0_{m\times n} & \text{block-diag}(1_n^\top, \ldots, 1_n^\top) \\ 0_{m\times n} & -\text{block-diag}(1_n^\top, \ldots, 1_n^\top) \\ \hline 0_{nm\times n} & I_{nm} \\ 0_{n\times n} & -I_n\ \cdots\ -I_n \end{array}\right], \end{aligned}$$

$$D = \left[\begin{array}{ccc} 0_{nm\times n} & 0_{nm\times n} & -\text{block-diag}(\Phi, \ldots, \Phi) \\ \hline -I_n & 0_{n\times n} & 0_{n\times nm} \\ I_n & -x_{\max}I_n & 0_{n\times nm} \\ -I_n & x_{\max}I_n & 0_{n\times nm} \\ I_n & 0_{n\times n} & n\times nm \\ \hline 0_{m\times n} & 0_{m\times n} & 0_{m\times nm} \\ 0_{m\times n} & 0_{m\times n} & 0_{m\times nm} \\ \hline & -I_n & \\ 0_{nm\times n} & \vdots & 0_{nm\times nm} \\ & -I_n & \\ 0_{n\times n} & I_n & 0_{n\times nm} \end{array}\right],$$

$$E = \left[\begin{array}{c} 0_{nm\times 1} \\ \hline 1_n \\ -1_n \\ (x_{\max}+1)1_n \\ -1_n \\ \hline 1_m \\ -1_m \\ \hline 0_{nm\times 1} \\ 0_{n\times 1} \end{array}\right].$$

Using (10), Problem 1 can be equivalently transformed into the following MILP problem.

Problem 2: Suppose that the initial state $x(0)$ is given by $x(0) = x_0$. Then, find $\bar{v} := [v^\top(0)\ v^\top(1)\ \cdots\ v^\top(N-1)]^\top \in (\{0,1\}^{nm+n} \times \mathcal{R}^n)^N$ minimizing the following linear cost function:

$$J = \bar{Q}\bar{B}\bar{v} + \bar{Q}\bar{A}x_0$$

subject to the following linear constraint:

$$(\bar{C}\bar{B} + \bar{D})\bar{v} \le \bar{E} - \bar{C}\bar{A}x_0,$$

where

$$\begin{aligned} \bar{Q} &= [Q\ Q\ \cdots\ Q], \\ Q &= [q_1\ q_2\ \cdots\ q_n]\,[I_n\ 0_{n\times nm}], \\ \bar{A} &= \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ AB & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A^{N-1}B & \cdots & AB & B \end{bmatrix}, \\ \bar{C} &= \begin{bmatrix} \text{block-diag}(C, \ldots, C) & 0 \end{bmatrix}, \\ \bar{D} &= \text{block-diag}(D, \ldots, D), \\ \bar{E} &= \begin{bmatrix} E^\top & \cdots & E^\top \end{bmatrix}^\top. \end{aligned}$$

4. Proposed Method

In Sect. 2, we formulated the optimal surveillance problem, and explained the MPC method. In Sect. 3, we reduced it to

Problem 2, i.e., the MILP problem. The optimal trajectories of multiple agent can be obtained by solving the MILP problem at each discrete time. However, there are two technical issues as follows.

i) The computation time for solving Problem 2 exponentially grows with the number of agents.

ii) According to the policy of MPC, Problem 2 must be solved at each discrete time.

For the above technical issue i), we propose the following two methods.

i) Individual optimization for each agent.

ii) Automatic selection of the priority for each agent.

For the above technical issue ii), we propose the following method.

iii) Introduction of an event-triggering condition.

In this section, we explain these three methods. Finally, we present the overall procedure.

4.1 Individual Optimization for Each Agent

In this method, we consider solving Problem 2 for a single agent under the assumption that trajectories of other agents are given. Let $\bar{\delta}_j(k|t)$ denote $\bar{\delta}_j(k)$ obtained by solving Problem 2 at time t. In addition, we define

$$\bar{\delta}_j(0:N-1|t) := \left[\bar{\delta}_j(0|t)\ \ \bar{\delta}_j(1|t)\ \ \cdots\ \ \bar{\delta}_j(N-1|t)\right].$$

The procedure is given as follows, where we assume that the priority of agents is given by $1 \to 2 \to \cdots \to m$.

Procedure of Individual Optimization at time t:

Step 0: Assume that there exists the computation result $\bar{\delta}_j(0:N-1|t-1)$ at time $t-1$.

Step 1: Set $j = 1$.

Step 2: Set $\bar{\delta}_{\hat{j}}(0:N-1|t)$, $\hat{j} \in \{1,2,\ldots,m\} \setminus \{j\}$ as follows:

$$\bar{\delta}_{\hat{j}}(0:N-1|t) = \left[\bar{\delta}_{\hat{j}}(0|t)\ \ \bar{\delta}_{\hat{j}}(1|t)\ \ \cdots\ \ \bar{\delta}_{\hat{j}}(N-1|t)\right] \quad \text{if } \hat{j} < j,$$

$$\bar{\delta}_{\hat{j}}(0:N-1|t) = \left[\bar{\delta}_{\hat{j}}(1|t-1)\ \ \bar{\delta}_{\hat{j}}(2|t-1)\ \ \cdots\ \ \bar{\delta}_{\hat{j}}(N-1|t-1)\ \ \bar{\delta}_{\hat{j}}(N-1|t-1)\right] \quad \text{if } \hat{j} > j.$$

Step 3: Solve Problem 2 under the assumption that $\bar{\delta}_{\hat{j}}(0:N-1|t)$, $\hat{j} \in \{1,2,\ldots,m\} \setminus \{j\}$ is given.

Step 4: If $j = m$, then the procedure stops. Otherwise, update $j := j+1$, and return to Step 2.

At time 0, Problem 2 for m agents must be solved. However, it may be solved before control starts (i.e., $t = 0$). Hence, a long computation time is allowed. At time $t \geq 1$, the computation time of Step 3 is that for solving Problem 2 for a single agent. In the above procedure, Problem 2 for a single agent is solved m times. In general, the computation time for solving an MILP problem exponentially grows with the number of binary variables. From this fact, the computation time to solve the m MILP problems for a single agent is shorter than the computation time to solve the MILP problem for m agents. Thus, the online computation time can be decreased by using the above procedure.

4.2 Automatic Selection of the Priority for Each Agent

In the procedure of the previous subsection, the priority of agents is given by $1 \to 2 \to \cdots \to m$. However, it is desirable to change the priority of agents depending on the situation. In this subsection, a method to decide the priority of agents is proposed. Here, we focus on penalties of nodes in the neighborhood of each agent.

First, we consider deriving the set of nodes that the agent $j \in \{1,2,\ldots,m\}$ can move within L discrete time, where L is given in advance. Such nodes can be characterized by the following row vector:

$$\gamma_j(k) := \bar{\delta}_j^\top(k)\Phi^L$$

(Φ is the adjacency matrix of a given graph). If the i-th element of $\gamma_j(k)$ is greater than 1, then the agent can move to the node v_i within L discrete time. Let $\hat{\gamma}_j(k)$ denote the row vector obtained by replacing an element which is greater than 1 with 1. Then, consider

$$\kappa_j(k) := \frac{\hat{\gamma}_j(k)\left[q_1x_1(k)\ \ q_2x_2(k)\ \ \cdots\ \ q_nx_n(k)\right]^\top}{\hat{\gamma}_j(k)1_n},$$

which implies the average of penalties of nodes that the agent j can move within L discrete time. For each j, we can easily calculate $\kappa_j(k)$. We can obtain the priority of agents by arranging $\kappa_j(k)$, $j \in \{1,2,\ldots,m\}$ in descending order.

4.3 Introduction of an Event-Triggering Condition

In the conventional MPC method, the optimal control problem such as Problem 1 must be solved at each discrete time. To overcome this technical issue, we introduce a simple event-triggering condition.

The proposed condition consists of two conditions. First, we evaluate the sum of penalties of all nodes. That is, we consider the following condition:

$$\hat{J}(t) := \sum_{i=1}^{n} q_i x_i(t+1) \geq \alpha,$$

where t is the current time, and α is a certain threshold given in advance.

Next, we focus on the fact that the trajectories with the length N can be obtained by solving Problem 1. For example, consider the case of $N = 5$. If Problem 1 is solved at $t = 0$, then there exists the next location at $t = 1,2,3,4$. At $t = 5(= N)$, Problem 1 must be re-computed. Let t_l denote

the last time that Problem 1 was solved. Then, we consider the following condition:

$$t \geq t_l + N.$$

From the above discussion, Problem 1 is solved at time t, only when either $\hat{J}(t) \geq \alpha$ or $t \geq t_l + N$ is satisfied.

4.4 Proposed Procedure

Finally, we propose a procedure of MPC based on the above three methods.

Computationally Efficient MPC for Multi-Agent Surveillance Systems:

Step 1: Set $t = 0$, and give $x_i(0)$ (the initial penalty for each node) and the initial location for each agent.

Step 2: Solve Problem 2 for m agents. Go to Step 7.

Step 3: Check if either $\hat{J}(t) \geq \alpha$ or $t \geq t_l + N$ is satisfied. If this condition holds, then go to Step 4. Otherwise, go to Step 6.

Step 4: Determine the priority of agents based on the method in Section 4.2.

Step 5: Solve Problem 2 based on the procedure of individual optimization in Section 4.1.

Step 6: Set $\bar{\delta}_j(t+1|t) := \bar{\delta}_j(t+1|t-1)$.

Step 7: Move agents based on the obtained $\bar{\delta}_j(t+1|t)$.

Step 8: Update $t := t + 1$, and return to Step 3.

Using this procedure, the number of times that Problem 2 is solved can be decreased. Even if Problem 2 is solved at time $t \geq 1$, it can be solved fast using the procedure in Section 4.1. Thus, the proposed MPC method is efficient from the computational viewpoint.

Remark 1: In [8], we have proposed a method for reducing the computation time using a graph decomposition. In this method, each agent is assigned to each subgraph. Since Problem 2 is decomposed to small MILP problems, the computation time is reduced. For the large-scale complex surveillance problem such that there are many nodes and many agents, we must consider the case that agents stop due to failures. In such a case, the method in [8] has a weakness. That is, for each case on failures of agents, a graph decomposition must be re-calculated. In the proposed method, we do not need to consider these cases individually. Behaviors of agents are automatically changed by imposing the constraint that faulty agents stop.

5. Numerical Example

In this section, we present a numerical example to show the effectiveness of the proposed method. In this numerical example, we used the computer with CPU: Intel Core i7-4770K processor, Memory: 32 GB, and used IBM ILOG CPLEX Optimizer 12.7.1 as an MILP solver.

Consider the undirected connected graph shown in Fig. 1 as a surveillance area. Suppose that the number of agents is given by three ($m = 3$). Suppose also that the initial penalty, the initial location, the prediction horizon N, and the weight q_i are given by $x_i(0) = 0$, v_1 for the agent 1, v_7 for the agent 2, v_{14} for the agent 3, $N = 10$, and $q_i = 1$, respectively.

Table 1 Computation results (automatic selection of the priority for each agent is not applied).

Case	1	2	3	4	Optimal
α	0	20	30	40	0
J_s	5187	5161	5329	5524	5012
T_a [sec]	0.3	0.4	0.4	0.4	37.6
T_w [sec]	0.4	0.4	0.4	0.4	107
s	150	149	141	47	150

Table 2 Computation results (automatic selection of the priority for each agent is applied).

Case	5	6	7
α	20	30	40
J_s	5157	5141	5467
T_a [sec]	0.3	0.4	0.4
T_w [sec]	0.4	0.4	0.4
s	149	146	42

We present the computation results. Here, we evaluate the following four points:

i) J_s: the sum of penalties in $[0, 150]$, i.e., $J_s := \sum_{k=0}^{150} \sum_{i=1}^{n} q_i x_i(k)$,

ii) T_a: the average computation time for solving Problem 1 in $[1, 149]$,

iii) T_w: the worst computation time for solving Problem 1 in $[1, 149]$,

iv) s: the number of times that Problem 1 is solved in $[0, 149]$.

First, we present the computation result using the existing method in [8], where Problem 2 for three agents is solved at each time. Then, we can obtain

$$J_s = 5012, \quad T_a = 37.6 \text{ [sec]}, \quad T_w = 107 \text{ [sec]}, \quad s = 150.$$

We remark that $J_s = 5012$ is optimal.

Next, we present the computation results using the proposed method. Table 1 shows the computation result, where automatic selection of the priority for each agent in Sect. 4.2 is not applied. From this table, we see that the computation time is improved by using individual optimization in Sect. 4.1. In Case 4, the control performance J_s is degraded about 10 [%]. Comparing Case 1 with Case 2, J_s in Case 2 is better than that in Case 1. This seems to be strange. However, recomputation of Problem 2 may not necessarily produce a better computation result. This is because the initial solution is obtained by solving Problem 2 for m agents. When Problem 2 is solved at time 1, the procedure of individual optimization in Sect. 4.1 is applied, and the solution may be degraded. This is one of the interesting points of the

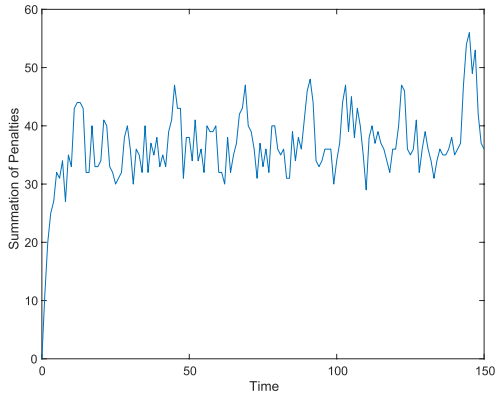

Fig. 2 Time response of the sum of penalties in Case 4.

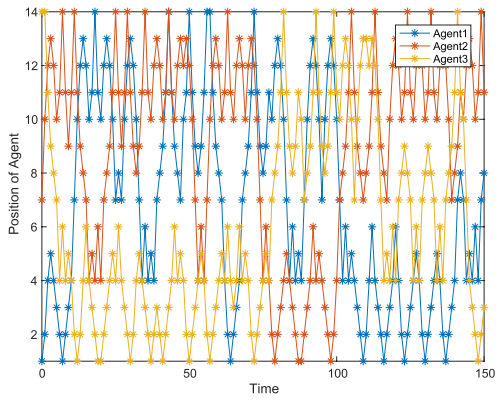

Fig. 3 Time response of agents in Case 4.

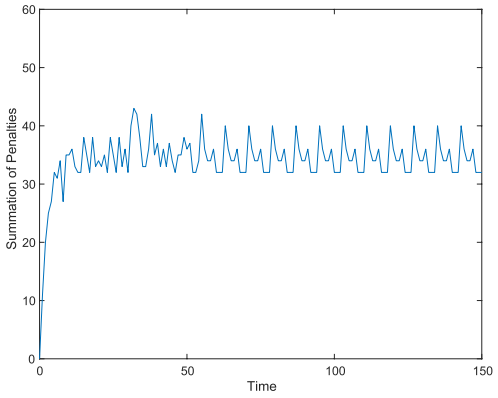

Fig. 4 Time response of the sum of penalties in Case 6.

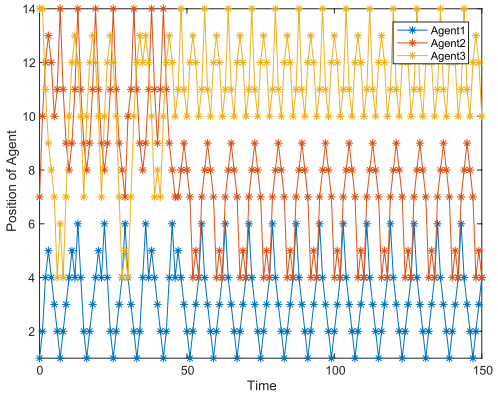

Fig. 5 Time response of agents in Case 6.

proposed MPC method.

Third, automatic selection of the priority for each agent in Sect. 4.2 is also applied, where L is set as $L = 3$. Then, we can obtain Table 2. From a comparison between Case 2 and Case 5, a comparison between Case 3 and Case 6, and a comparison between Case 4 and Case 7, we see that the control performance J_s is improved.

Finally, we present time responses of agents and the sum of penalties. Figures 2 and 3 show time responses of agents and the sum of penalties in Case 4, respectively. Figures 4 and 5 show time responses of agents and the sum of penalties in Case 6, respectively. From Fig. 2 and Fig. 4, we see that as an overall trend, the sum of penalties in Fig. 4 is less than that in Fig. 2. In addition, from Fig. 5, we see that in Case 6, the periodic trajectories of agents are obtained.

6. Conclusion

In this paper, we proposed a new MPC method for multi-agent surveillance systems. The proposed method is an improved version of our previous proposed method in [8], but several new ideas are included. First, the procedure of individual optimization was proposed. This method is a kind of approximation algorithms to solve Problem 2. Next, to improve the control performance, a method for selecting the priority of agents was proposed. Finally, an event-triggered condition was proposed. The effectiveness of the proposed method was presented by numerical simulations.

In future work, we will consider a large-scale graph with practical constraints such as fuel constraints. In addition, it is important to analyze an accuracy of the solution obtained by the proposed method. To further reduce the computation cost, it is also important to develop a parallel and distributed algorithm.

This work was partly supported by the Telecommunications Advancement Foundation and JSPS KAKENHI Grant Numbers JP17K06486, JP16H04380.

References

[1] D. Aksaray, K. Leahy, and C. Belta, "Distributed multi-agent persistent surveillance under temporal logic constraints," Proc. 5th IFAC Workshop on Distributed Estimation and Control in Networked Systems, pp.174–179, 2015.
[2] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas, "Discrete abstraction of hybrid systems," Proc. IEEE, vol.88, no.7, pp.971–984, 2000.
[3] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," Automatica, vol.35, no.3, pp.407–427, 1999.
[4] E.F. Camacho and C.B. Alba, Model Predictive Control, Springer, 2008.
[5] C.G. Cassandras, X. Lin, and X. Ding, "An optimal control approach to the multi-agent persistent monitoring problem," IEEE Trans. Autom. Control, vol.58, no.4, pp.947–961, 2013.

[6] J.G.M. Fu and M.H. Ang, Jr., "Probabilistic ants (PAnts) in multi-agent patrolling," Proc. 2009 IEEE/ASME Int'l Conf. on Advanced Intelligent Mechatronics, pp.1371–1376, 2009.
[7] W.P.M.H. Heemels, K.H. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," Proc. 51st IEEE Conf. on Decision and Control, pp.3270–3285, 2012.
[8] M. Kido, K. Kobayashi, and Y. Yamashita, "MPC-based surveillance over graphs by multiple agents," SICE Journal of Control, Measurement, and System Integration, vol.10, no.3, pp.253–258, 2017.
[9] S. Karaman, R.G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," Proc. 47th IEEE Conf. on Decision and Control, pp.2117–2122, 2008.
[10] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-UAV mission planning," Int J. Robust Nonlinear Control, vol.21, no.12, pp.1372–1395, 2011.
[11] K. Kobayashi and J. Imura, "Deterministic finite automata representation for model predictive control of hybrid systems," J. Process Contr., vol.22, no.9, pp.1670–1680, 2012.
[12] K. Kobayashi, T. Nagami, and K. Hiraishi, "Optimal control of multi-vehicle systems with temporal logic constraints," IEICE Trans. Fundamentals, vol.E98-A, no.2, pp.626–634, Feb. 2015.
[13] D. Lehmann, E. Henriksson, and K.H. Johansson, "Event-triggered model predictive control of discrete-time linear systems subject to disturbances," Proc. 2013 European Control Conf., pp.1156–1161, 2013.
[14] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O. M. Scokaert, "Constrained model predictive control: Stability and optimality," Automatica, vol.36, no.6, pp.789–814, 2000.
[15] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms," IEEE Trans. Robot., vol.28, no.3, pp.592–606, 2012.
[16] P. Tabuada, Verification and Control of Hybrid Systems, Springer, 2009.
[17] Y. Tazaki and J. Imura, "Discrete abstractions of nonlinear systems based on error propagation analysis," IEEE Trans. Autom. Control, vol.57, no.3, pp.550–564, 2012.

Koichi Kobayashi received the B.E. degree in 1998 and the M.E. degree in 2000 from Hosei University, and the D.E. degree in 2007 from Tokyo Institute of Technology. From 2000 to 2004, he worked at Nippon Steel Corporation. From 2007 to 2015, he was an Assistant Professor at Japan Advanced Institute of Science and Technology. Since 2015, he has been an Associate Professor at the Graduate School of Information Science and Technology, Hokkaido University. His research interests include analysis and control of discrete event and hybrid systems. He is a member of IEEE, IEEJ, IEICE, ISCIE, and SICE.

Mifuyu Kido received the master degree from Hokkaido University in 2018. Her research interests include design of surveillance systems.

Yuh Yamashita received his B.E., M.E., and Ph.D. degrees from Hokkaido University, Japan, in 1984, 1986, and 1993, respectively. In 1988, he joined the faculty of Hokkaido University. From 1996 to 2004, he was an Associate Professor at the Nara Institute of Science and Technology, Japan. Since 2004, he has been a Professor of the Graduate School of Information Science and Technology, Hokkaido University. His research interests include nonlinear control and nonlinear dynamical systems. He is a member of SICE, ISCIE, RSJ, and IEEE.