Proceedings of the SICE Annual Conference 2020
September 23-26, 2020, Chiang Mai, Thailand

Dynamic Programming for Discrete-Event and Hybrid Systems

——— With respect to the 100th anniversary of Richard-Ernest-Bellman's birth ———

Yoshifumi Okuyama

Humanitech Laboratory Co., Ltd. & Tottori University (Professor Emeritus)
Nakatsuura 115-7, Hachiman-cho, Tokushima, 770-8072 Japan
(Tel: +81-88-625-2545; E-mail: oku@humanitech-lab.jp)

Abstract: This year is the 100th anniversary of Richard-Bellman's birth. Although his research results are 'great' even now, we should reconsider them suitable for the present technology (i.e., the digital society). In this paper, the concept of dynamic programming for discrete-event and hybrid systems is considered and expanded. In addition, the multistage decision processes are discussed in transferred vectors (and connective (0,1)- matrices) representation. Especially, in this paper, the 'comparability' of systems performance (i.e., recurrent inequalities expression) will be emphasized rather than the 'optimality' based on the usual min/max notation. Lastly, numerical examples are shown in some application areas including a Bellman's result.

Keywords: Discrete-event dynamic systems; transferred vectors; permutation matrices; recurrence inequalities; packet-switching networks

1. INTRODUCTION

Richard Ernest Bellman (1920-1984): This year becomes his 100th birthday. His research results are 'great' even now [1-8]. At present, there are many event-driven types of discrete control systems in practice, e.g., manufacturing systems, industrial and welfare robots, and even packet-switching communication networks for IoT [9, 10]. Therefore, in this paper, the concept of dynamic programming for discrete-event and hybrid systems is reconsidered. In addition, the multistage decision processes are discussed in transferred vectors (and connective (0,1)-matrices) representation. Especially, in this paper, the 'comparability' of systems performance (i.e., recurrent inequalities expression) will be emphasized rather than the 'optimality' based on the usual min/max notation.

2. ALLOCATION PROCESSES

In general, finite-state and discrete-event dynamic systems(DEDS) can be expressed as the following multistage processes [1]:

$$\begin{aligned} &\boldsymbol{x}(t_{k+1}) = \boldsymbol{f}(\boldsymbol{x}(t_k), \boldsymbol{e}(t_k)) \\ &k \in \mathbb{N} := \{0, 1, 2, \ldots, N\}, \end{aligned} \tag{1}$$

where $\boldsymbol{x}(\cdot)$, $\boldsymbol{e}(\cdot)$, and $\boldsymbol{f}(\cdot,\cdot)$ are states, event-signals, and a transition function, respectively, as written below:

$$\begin{aligned} &\boldsymbol{x}(t_k) \in \mathbb{Z}^n, \ \ \boldsymbol{e}(t_k) \in \mathbb{Z}^m, \\ &\boldsymbol{f} : \ \mathbb{Z}^n \times \mathbb{Z}^m \to \mathbb{Z}^n, \end{aligned}$$

where $\mathbb{Z}$ is considered as a finite set of integers. Of course, the states and events may be non-numerical (qualitative) situations in practice. However, they can be considered as ordered sets, and thus the ordered sets will be replaced by (positive) integer numbers [11, 12].

† Yoshifumi Okuyama is the presenter of this paper.

[1] Multistage processes and systems concept are clearly defined in [6].

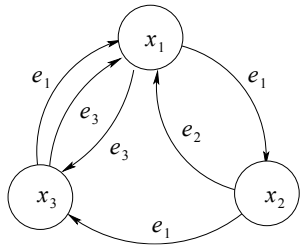

Fig. 1 A transition graph with three states.

These processes will be defined and categorized in various ways: finite, infinite; bounded, unbounded; discrete and continunous [6].

In regard to simple discrete-event systems, (1) can also be written by using a permutation matrix (connective (0,1)-matrix)[11] as

$$x(t_{k+1}) = \boldsymbol{\mathcal{P}}(\boldsymbol{x}(t_k), \boldsymbol{e}(t_k)) \cdot x(t_k), \tag{2}$$

where [2]

$$\begin{aligned} &\boldsymbol{\mathcal{P}}(\cdot,\cdot) = \boldsymbol{P}(t_k) \in \mathbb{I}_+^{n\times n}, \\ &\mathbb{I}_+ : \ \text{permutation } (0,1) - \text{ matrix}. \end{aligned}$$

Therefore, (2) can be expanded as follows:

$$\boldsymbol{x}(t_k) = \left(\prod_{j=0}^{k-1} \boldsymbol{P}(t_j) \right) \boldsymbol{x}(t_0). \tag{3}$$

First, we consider a state-transition system with three states as shown in Fig. 1 [13]. When the states start from[3]

$$\begin{cases} x_1(t_0) = \hat{x}_1 \\ x_2(t_0) = \hat{x}_2 \\ x_3(t_0) = \hat{x}_3. \end{cases} \tag{4}$$

[2] In the following, we write simply $\boldsymbol{\mathcal{P}}(t_k) = \boldsymbol{\mathcal{P}}(\boldsymbol{x}(t_k), \boldsymbol{e}(t_k))$.

[3] Here, coordinates (places) x_i $(i = 1, 2, 3)$ and values (transitions) $\hat{x}_j$ $(j = 1, 2, 3)$ are distinguished.

The state-transitions are written as the following transferred vectors [4]:

$$\begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix} \overset{e_1}{\to} \begin{bmatrix}\hat{x}_3\\ \hat{x}_1\\ \hat{x}_2\end{bmatrix} \overset{e_1}{\to} \begin{bmatrix}\hat{x}_2\\ \hat{x}_3\\ \hat{x}_1\end{bmatrix} \overset{e_1}{\to} \begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix}, \tag{5}$$

$$\begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix} \overset{e_1}{\to} \begin{bmatrix}\hat{x}_3\\ \hat{x}_1\\ \hat{x}_2\end{bmatrix} \overset{e_2}{\to} \begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix} \overset{e_\phi}{\to} \begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix},$$

$$\begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix} \overset{e_3}{\to} \begin{bmatrix}\hat{x}_3\\ \hat{x}_2\\ \hat{x}_1\end{bmatrix} \overset{e_3}{\to} \begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix} \overset{e_\phi}{\to} \begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\end{bmatrix}.$$

When using a matrix multiplication, the following expressions (i.e., permutation matrices) can be given:

$$\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}_{e_1}\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}_{e_1}\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}_{e_1}, \tag{6}$$

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}_{e_\phi}\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}_{e_2}\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}_{e_1},$$

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}_{e_\phi}\begin{bmatrix}0&0&1\\0&1&0\\1&0&0\end{bmatrix}_{e_3}\begin{bmatrix}0&0&1\\0&1&0\\1&0&0\end{bmatrix}_{e_3}.$$

The above expressions are as for event-driven discrete dynamical systems. However, actual discrete systems will accompany with continuous (analog) sub-systems. Thus, discere-continous (hybrid-type) systems should be considered.

2.1. Considerations for Hybrid Sstems

When there are some continuous sub-systems, hybrid systems will be written as follows[14, 15] [5]:

$$\dot{\boldsymbol{\xi}}_i(\tau) = \varphi_i(\boldsymbol{\xi}_i(\tau), x_i(t_k)),\ \ t_k \le \tau < t_{k+1}, \tag{7}$$

$$x_j(t_{k+1}) = \mathbb{Q}(\boldsymbol{\xi}_i(t_{k+1})), \tag{8}$$
$$i, j = 1, 2, \cdots, n,$$

where $\boldsymbol{\xi} \in \mathbb{R}^n$, $\varphi : \mathbb{R}^n \times \mathbb{Z} \to \mathbb{R}^n$, and $\mathbb{Q} : \mathbb{R}^\kappa \to \mathbb{Z}$ (i.e., a kind of quantization). Moreover, (7) can be expressed as follows:

$$\boldsymbol{\xi}_i(t_k+\Delta\tau) = \boldsymbol{\xi}_i(t_k) + \int_{t_k}^{t_k+\Delta\tau} \varphi(\boldsymbol{\xi}_i(\tau), x_i(t_k))\,\mathrm{d}\tau. \tag{9}$$

However, at the present day, it should be considered that there are no artificial continuous (i.e., analog) subsystems. Thus, we should be considered the following discrete sub-system:

$$\boldsymbol{\xi}(\tau_{h+1}) = \boldsymbol{\xi}(\tau_h) + \boldsymbol{\eta}(\boldsymbol{\xi}(\tau_h), x_i(\tau_h)), \tag{10}$$
$$t_k \le \tau_h < t_{k+1}.$$

[4] Of course, there is the case where event e_2 is repeated (change occurs when considering a vending machine). Here, e_ϕ means an *empty event* (i.e., no action).

[5] Those systems are also called as a hybrid automaton[16].

In these equations, $x_i(t_k)$ $(i = 1, 2, \cdots)$, i.e.,

$$\boldsymbol{x}(t_k) = \begin{bmatrix}x_1(t_k) & x_2(t_k) & \cdots & x_n(t_k)\end{bmatrix}^T$$

are state-transitions. Therefore, the behavior of hybrid systems will be considered simple discrete systems (i.e. mltistage processes in [6]),

$$\boldsymbol{x}(t_k) = \boldsymbol{\phi}(\boldsymbol{x}(t_{k-1}), e(t_k)),\quad k = 1, 2, \cdots, N \tag{11}$$

in a modified expression of (1) and (2). Here, $\boldsymbol{\phi} : \mathbb{Z}^n \times \mathbb{Z}_+ \to \mathbb{Z}^n$.

3. MULTISTAGE DECISION PROCESSES

3.1. The principle of Optimality

Richard Bellman once proposed the following optimal policy with regard to the design of dynamical systems [2, 5, 6]. The principle of optimality is written as follows.

An optimal policy has the property that whatever the initial state and initial dicision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

3.2. Minimization of Cost

Consider the following cost-minimization problem:

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N) \tag{12}$$
$$= \psi(\boldsymbol{x}_0, \boldsymbol{x}_1) + \psi(\boldsymbol{x}_1, \boldsymbol{x}_2) + \cdots + \psi(\boldsymbol{x}_{N-1}, \boldsymbol{x}_N),$$

where $\psi(\boldsymbol{x}_{k-1}, \boldsymbol{x}_k),\ k = 1, 2, \cdots, N$ are state-transition costs based on the event sequences[6]:

$$\boldsymbol{\mathcal{E}}_N = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_N\}. \tag{13}$$

When considering dynamical systems, the state-transition in regard to multistage and hybrid processes will be written as shown in (11), i.e.,

$$\boldsymbol{x}(t_k) = \boldsymbol{\phi}(\boldsymbol{x}(t_{k-1}), \boldsymbol{e}(t_k)),\ \ k = 1, 2, \cdots, N. \tag{14}$$

where $\boldsymbol{x}_0 := \boldsymbol{x}(t_0) = \boldsymbol{x}(0)$, $\boldsymbol{x}_k := \boldsymbol{x}(t_k)$, and $\boldsymbol{e}_k := \boldsymbol{e}(t_k)$, $k = 1, 2, \cdots, N$. The minimization of costs should be given as:

$$\Psi_N(\boldsymbol{x}_N) = \min_{\boldsymbol{\mathcal{E}}_N} C(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N). \tag{15}$$

Therefore, we obtain the following functional equation based on *Dynamic Programing*:

$$\Psi_N(\boldsymbol{x}_N) = \min_{\boldsymbol{\mathcal{E}}_N} \left[\psi(\boldsymbol{x}_{N-1}, \boldsymbol{x}_N) + \Psi_{N-1}(\boldsymbol{x}_{N-1})\right]. \tag{16}$$

The above can be written as follows:

$$\begin{cases}\Psi_1(\boldsymbol{x}_1) = \min\limits_{\{\boldsymbol{e}_1\}} \psi(\boldsymbol{x}_0, \boldsymbol{x}_1)\\ \Psi_2(\boldsymbol{x}_2) = \min\limits_{\{\boldsymbol{e}_1, \boldsymbol{e}_2\}} \left[\psi(\boldsymbol{x}_1, \boldsymbol{x}_2) + \Psi_1(\boldsymbol{x}_1)\right]\\ \ \ \vdots\\ \Psi_N(\boldsymbol{x}_N) = \min\limits_{\boldsymbol{\mathcal{E}}_N} \left[\psi(\boldsymbol{x}_{N-1}, \boldsymbol{x}_N) + \Psi_{N-1}(\boldsymbol{x}_{N-1})\right].\end{cases} \tag{17}$$

[6] For example, e_1, e_2, e_3 in (5) are coorrespond to these event sequences.

3.3. Minimization of Distance

When dealing with only distance at the state-transition, the costs will be given as:

$$D(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N) \quad (18)$$
$$= \|\boldsymbol{x}_0 - \boldsymbol{x}_1\| + \|\boldsymbol{x}_1 - \boldsymbol{x}_2\| + \cdots + \|\boldsymbol{x}_{N-1} - \boldsymbol{x}_N\|,$$

where $\|\boldsymbol{x}\| := \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$. In this case,

$$\Psi_N(\boldsymbol{x}_N) = \min_{\boldsymbol{\mathcal{E}}_N} D(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N). \quad (19)$$

Therefore, the recurrence equation is

$$\Psi_N(\boldsymbol{x}_N) = \min_{\boldsymbol{\mathcal{E}}_N} \left[\|\boldsymbol{x}_{N-1} - \boldsymbol{x}_N\| + \Psi_{N-1}(\boldsymbol{x}_{N-1})\right]. \quad (20)$$

3.4. Maximization of Gain

When teating the N-stage gain of the process,

$$G(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N) \quad (21)$$
$$= g(\boldsymbol{x}_0, \boldsymbol{x}_1) + g(\boldsymbol{x}_1, \boldsymbol{x}_2) + \cdots + g(\boldsymbol{x}_{N-1}, \boldsymbol{x}_N).$$

In this case,

$$\Gamma_N(\boldsymbol{x}_N) = \max_{\boldsymbol{\mathcal{E}}_N} G(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N). \quad (22)$$

Therefore, we obtain the following functional equation:

$$\Gamma_N(\boldsymbol{x}_N) = \max_{\boldsymbol{\mathcal{E}}_N} \left[g(\boldsymbol{x}_{N-1}, \boldsymbol{x}_N) + \Gamma_{N-1}(\boldsymbol{x}_{N-1})\right]. \quad (23)$$

The above can be written as follows:

$$\begin{cases} \Gamma_1(\boldsymbol{x}_1) = \max\limits_{\{\boldsymbol{e}_1\}} g(\boldsymbol{x}_0, \boldsymbol{x}_1) \\ \Gamma_2(\boldsymbol{x}_2) = \max\limits_{\{\boldsymbol{e}_1, \boldsymbol{e}_2\}} \left[g(\boldsymbol{x}_1, \boldsymbol{x}_2) + \Gamma_1(\boldsymbol{x}_1)\right] \\ \quad \vdots \\ \Gamma_N(\boldsymbol{x}_N) = \max\limits_{\boldsymbol{\mathcal{E}}_N} \left[g(\boldsymbol{x}_{N-1}, \boldsymbol{x}_N) + \Gamma_{N-1}(\boldsymbol{x}_{N-1})\right]. \end{cases}$$

4. COMPARABILITY AND RECURRENCE INEQUALITIES

Systems and Control Theory has been developed based on the principle of optimality. However, the optimality of control system may be an illusion of mathematician. The theorists of systems and control theory are very particular about continuous mathematics, i.e., differential and integral calculus, differental equation, and linear algebra, especially quadratic form. However, what is 'continuous'. We think that the technique of differentiation/integration and the concept of limitation and infinity (e.g.,$\lim_{\Delta x \to 0}$, $y \to \infty$) would not be necessary [7]. Every computer-simulated systems are 'finite' and 'discrete'.

In the preface of book[8] (i.e., *Algorithms Graphs and Computers*, Academic Press, 1970), Richard says that "The development of the electronic computer has profoundly and irerevocably changed the scientific world".

At present day, we are exactly in the digital-computer society. Therefore, we think that comparable viewpoints

[7] The author also thinks that the infinity dimensional space and moreover the fuction space, e.g., L_p, H_p, and really H_∞ spaces may be a 'dream world' for the theorists.

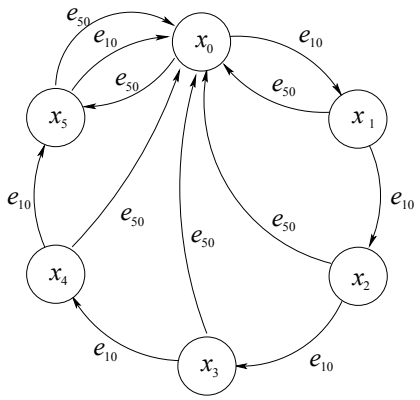

Fig. 2 A vending machine for canned drinks.

of system-performance, (profit/usefulness on the contrary, loss/uselessness) are rather important. For example, with regard to non-numerical (qualitative) problems, the following expression may be written:, e.g.,

$$\Psi_k(x_k) \preceq \psi(x_{k-1}, x_k) \cup \Psi_{k-1}(x_{k-1}),\ k = 1, 2, \cdots, N.$$

However, in any case the comparability of performance will be given by 'recurrence inequalities'. That is, they are nothing else 'if then $\sim$ else $\sim$' rules in the computer program.

As is obvious, (16) can be written by the following recurrence inequalities,

$$\begin{cases} \Psi_1(\boldsymbol{x}_1) \le \psi(\boldsymbol{x}_0, \boldsymbol{x}_1), \qquad \forall\{\boldsymbol{e}_1\} \\ \Psi_2(\boldsymbol{x}_2) \le \psi(\boldsymbol{x}_1, \boldsymbol{x}_2) + \Psi_1(\boldsymbol{x}_1), \quad \forall\{\boldsymbol{e}_1, \boldsymbol{e}_2\} \\ \quad \vdots \\ \Psi_{N-1}(\boldsymbol{x}_{N-1}) \le \psi(\boldsymbol{x}_{N-2}, \boldsymbol{x}_{N-1}) + \Psi_{N-2}(\boldsymbol{x}_{N-2}), \\ \Psi_N(\boldsymbol{x}_N) \le \psi(\boldsymbol{x}_{N-1}, \boldsymbol{x}_N) + \Psi_{N-1}(\boldsymbol{x}_{N-1}), \ \forall\boldsymbol{\mathcal{E}_N}. \end{cases} \quad (24)$$

Conversely, when $\boldsymbol{x}_N$ is fixed and $\Psi_N(\boldsymbol{x}_N) = \bar{\Psi}$ is assumed, the following (backward) recurrence inequalities can be obtained by subtracting $\bar{\Psi_N}$ in the both side of (24):

$$\begin{cases} \Delta_0(\boldsymbol{x}_0) \le \psi(\boldsymbol{x}_0, \boldsymbol{x}_1) + \Delta_1(\boldsymbol{x}_1), \quad \forall\boldsymbol{\mathcal{E}}_N \\ \Delta_1(\boldsymbol{x}_1) \le \psi(\boldsymbol{x}_1, \boldsymbol{x}_2) + \Delta_2(\boldsymbol{x}_2), \\ \quad \vdots \\ \Delta_{N-2}(\boldsymbol{x}_{N-2}) \le \psi(\boldsymbol{x}_{N-2}, \boldsymbol{x}_{N-1}) + \Delta_{N-1}(\boldsymbol{x}_{N-1}), \\ \Delta_{N-1}(\boldsymbol{x}_{N-1}) \le \psi(\boldsymbol{x}_{N-1}, \boldsymbol{x}_N), \qquad \forall\{\boldsymbol{e}_N\}. \end{cases} \quad (25)$$

where $\Delta_k(\boldsymbol{x}_k) = \bar{\Psi}_N - \Psi_k(\boldsymbol{x}_k)$. Consequently, $\Delta_0(\boldsymbol{x}_0)$ will become equal to $\bar{\Psi}_N$.

5. NUMERICAL EXAMPLES

5.1. Example 1 (Vending Machine in Japan)

Consider an example of buying canned drinks (60-yen) at a vending machine when a customer has (50-yen) coins

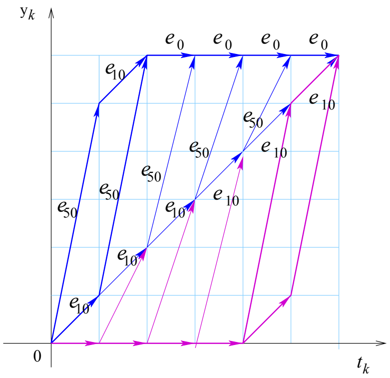

Fig. 3 An another graph of the above vending machine.

and (10-yen) coins. With regard to six-states transition as shown in Fig. 2, there exist the following cases:

$$\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_3\\\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\\\hat{x}_0\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{\boldsymbol{I}}$$

$$\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_3\\\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\\\hat{x}_0\end{bmatrix}\xrightarrow{e_{50}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{\boldsymbol{I}+ch40}$$

$$\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_3\\\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\\\hat{x}_2\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\\\hat{x}_0\\\hat{x}_1\end{bmatrix}\xrightarrow{e_{50}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{\boldsymbol{I}+ch30}$$

and so on. Event sequences are given as:

$$e_k \in \{e_{10}, e_{50}, e_{\phi}\}, \quad k = 1, 2, \cdots, 6.$$

Here, e_{10} and e_{50} are the cases when (10-yen) and (50-yen) coins inserted into the machine, respectively. Moreover, $\boldsymbol{I}$ and $ch40$ are the item and, for example, (40-yen) returned change, respectively. [8].

Efficient (optimal?) processes will be given below:

$$\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_1\\\hat{x}_0\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{50}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}$$

$$\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{50}}\begin{bmatrix}\hat{x}_5\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_0\end{bmatrix}\xrightarrow{e_{10}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_5\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_0\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_0\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_0\end{bmatrix}\xrightarrow{e_{\phi}}\begin{bmatrix}\hat{x}_0\\\hat{x}_1\\\hat{x}_2\\\hat{x}_3\\\hat{x}_4\\\hat{x}_0\end{bmatrix}.$$

In this example, the costs (efforts/labors) of state-transitions are assumed to be

$$\psi(\boldsymbol{x}_i, \boldsymbol{x}_j)_{e_{10}} = \psi(\boldsymbol{x}_i, \boldsymbol{x}_j)_{e_{50}} = c_\ell = \text{const.} > 0, \;\; \forall i, j.$$

Obviously, $\psi(\boldsymbol{x}_i, \boldsymbol{x}_i)_{e_\phi} = 0$. The relation between the effort of user and the amount of stocked-money y_k is easily drawn in Fig. 3. The above transitions can be written by matrix expressions (i.e., permutation (0,1)- matrix):

The above transitions can be written by the following matrix expressions (i.e., permutation (0,1)- matrix):

$$\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}^4\begin{bmatrix}0&1&0&0&0&0\\1&0&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}^2$$

for the event sequence $\mathcal{E}_{N1} = \{e_{10}, e_{50}, e_\phi, e_\phi, e_\phi, e_\phi\}$, and

$$\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}^4\begin{bmatrix}0&0&0&0&0&1\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\1&0&0&0&0&0\end{bmatrix}^2$$

for the event sequence $\mathcal{E}_{N2} = \{e_{10}, e_{50}, e_\phi, e_\phi, e_\phi, e_\phi\}$, respectively.

In these cases, $\Psi_{N1} = \Psi_{N2} = 2 \cdot c_\ell$. When (10-yen) coins are inserted to the machine successively. The matrix expression is given below:

$$\begin{bmatrix}0&0&0&0&0&1\\1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\end{bmatrix}^6.$$

As is obvious, in this case, $\Psi_{Nn} = 6 \cdot c_\ell$. Consequently,

$$\Psi_{N1} = \Psi_{N2} \le \Psi_{Nn}, \quad \forall \mathcal{E}_{Nn}, \;\; n = 1, 2, \cdots, 11.$$

5.2. Example 2 (A Simplified Trajectory Problem)

This example is a re-presentation from [5]. The problem is to minimize the sum of numbers(costs) from A to B in Fig. 4. Although this problem can be treated by a discrete-event system with 4×4 states, we will consider the states in the coordinates as shown in [5]. In this paper, as is described in Example 1 (i.e., a discrete event system) the following state vector with 7 elements will be considered:

$$\boldsymbol{x}(t_k) = [x_0, x_1, \cdots, x_6]^T.$$

Figure 5 shows the case where the costs are evaluated backward from B to A. On the other hand, Fig. 6 shows the case where the reverse (forward) algorithm is executed. Here, event e_k is considered as $e_k \in \{e_R, e_L\}$ (R: Right, L: Left, $k = 1, 2, \cdots, 6$). In these figures, the 'magenta' lines show minimum paths obtained [9]. Note that we can obtain the same result regradless of forward/backward algorithm or not.

[8] 'The blue notations mean that the state does not change because of no action.

[9] In Fig. 6, 'magenta' 14 is probably a mistake, 11, in [5].

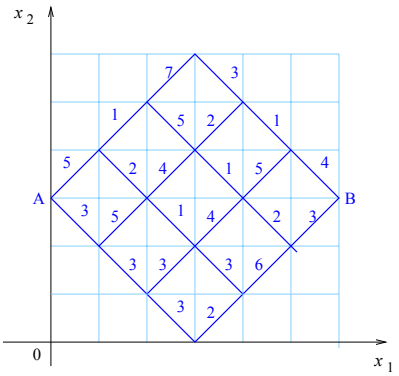

Fig. 4 A simplified trajectory problem.

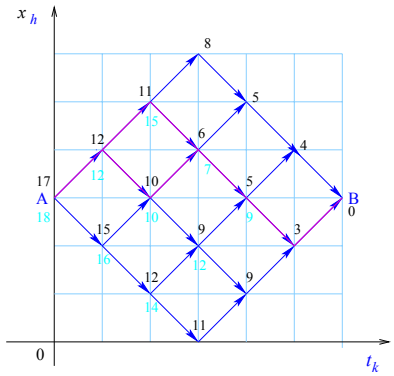

Fig. 5 A minimum trajectory problem.

The trasferred vectors are given below:

$$\begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\end{bmatrix} \overset{e_L}{\to} \begin{bmatrix}\hat{x}_6\\ \hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_L}{\to} \begin{bmatrix}\hat{x}_5\\ \hat{x}_6\\ \hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\end{bmatrix} \overset{e_R}{\to} \begin{bmatrix}\hat{x}_6\\ \hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_R}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\end{bmatrix} \overset{e_R}{\to} \begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\\ \hat{x}_0\end{bmatrix} \overset{e_L}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\end{bmatrix}$$

$$\begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\end{bmatrix} \overset{e_L}{\to} \begin{bmatrix}\hat{x}_6\\ \hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_R}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\end{bmatrix} \overset{e_L}{\to} \begin{bmatrix}\hat{x}_6\\ \hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_R}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\end{bmatrix} \overset{e_R}{\to} \begin{bmatrix}\hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\\ \hat{x}_0\end{bmatrix} \overset{e_L}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_6\end{bmatrix}.$$

Also in these cases, permutation-matrix expressions can be used.

Event sequences are given as:

$$\begin{cases}\mathcal{E}_1 = \{e_L, e_L, e_R, e_R, e_R, e_L\}\\ \mathcal{E}_2 = \{e_L, e_R, e_L, e_R, e_R, e_L\}.\end{cases}$$

The sum of costs from B to A (or from A to B) are obtaned as $\Psi_{61} = \Psi_{62} = 17$. Obviously,

$$\Psi_{61} = \Psi_{62} \le \Psi_{6n}, \quad \forall \mathcal{E}_{6n}.$$

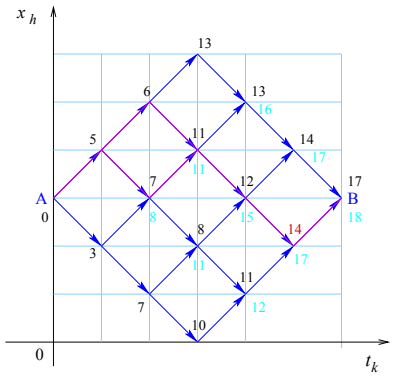

Fig. 6 A reverse solution.

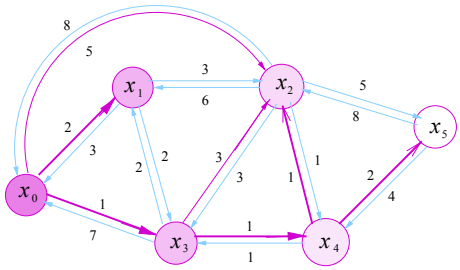

Fig. 7 Dijkstra's algorithm.

5.3. **Example 3** (A Packet Switching Network)

As described in 5.2, 'dynamic' in *Dyamic Programming* can be interpreted as a procedure based on the recurrence inequalities. This shortest path (fewer hops) problem was also introduced in [9, 18] as Bellman-Ford algorithm.

Dijkstra's Algorithm

Dijkstra's Algorithm [9, 17] is given by the following steps:

(1) $L(x_p) = \min_{x_j \notin T} L(x_j)$,

(2) $L(x_n) = \min_{x_q \notin T} \{L(x_q),\ L(x_p) + w(x_p, x_q)\}$,

where

- $x_0 = s$: starting (source) node
- $L(x_j)$: cost of the least-cost path from node x_0 to nodes x_j that is currently known to the algorithm
- $w(x_i, x_j)$: link cost from node x_i to nodes x_j

Thus, as for trees,

$$T = \{x_0\}, \{x_0, x_3\}, \{x_0, x_1, x_3\}, \{x_0, x_1, x_3, x_4\},$$
$$\{x_0, x_1, x_2, x_3, x_4\}, \{x_0, x_1, x_2, x_3, x_4, x_5\}\},$$

costs of the least-cost path $L(x_1) = 2$, $L(x_2) = 3$, $L(x_3) = 1$, $L(x_4) = 2$, $L(x_5) = 4$ are obtained. The result of recurrence trees considered here becomes as shown in Fig. 7.

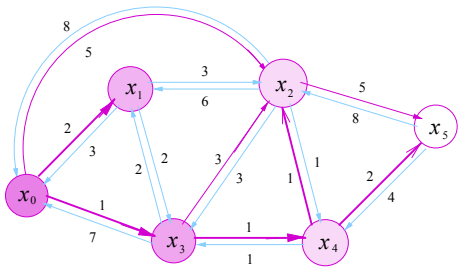

Fig. 8 Bellman-Ford algorithm.

Bellman- Ford Algorithm

On the other hand, Bellman- Ford Algorithm [9, 18] is written as:

$$L_{h+1}(x_n) = \min_{j<n}\{w(x_j, x_n) + L_h(x_j)\},$$

where

- $L_h(x_j)$: cost of the least-cost path from node x_0 to nodes x_j under the constraint of no more than h links
- $h = 0, 1, 2, 3, 4$: maximum number of links in a path

As is obvious, the above algorithm is simply written by the followig recurrence inequalities:

$$L_{h+1}(x_n) \leq w(x_j, x_n) + L_h(x_j), \quad \forall j < n$$

Thus, costs of the least-costs path are given as $L_1(x_1) = 2$, $L_2(x_2) = 4 \to 3$, $L_3(x_3) = 1$, $L_4(x_4) = 2$, $L_4(x_5) = 4$. In this algorithm, the recurrence process can be drawn as shown in Fig. 8.

The preferable transferred vectors are written as:

$$\begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_{03}}{\to} \begin{bmatrix}\hat{x}_3\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_0\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_{34}}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_4\\ \hat{x}_0\\ \hat{x}_5\end{bmatrix} \overset{e_{45}}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_4\\ \hat{x}_5\\ \hat{x}_0\end{bmatrix}.$$

An another transferred vectors are given as:

$$\begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_{02}}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_{25}}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix} \overset{e_{\phi}}{\to} \begin{bmatrix}\hat{x}_0\\ \hat{x}_1\\ \hat{x}_2\\ \hat{x}_3\\ \hat{x}_4\\ \hat{x}_5\end{bmatrix}.$$

The (0,1)-matrix of above case will be written below:

$$\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}.$$

6. CONCLUSION

The concept of dynamic programming for discrete-event and hybrid systems has been reconsidered connected with the 100 anniversary of Bellman's birth. The optimal search techniques appropriate for the present computer-processing were examined. Especially, in this paper, the comparability of systems performance (i.e., recurrent inequalities expression) was emphasized rather than the optimality based on the usual min/max notation.

REFERENCES

[1] Richard Bellman, *Stability Theory of Differential Equations*, MacGraw-Hill, 1953.

[2] Richard Bellman, *Dynamic Programming*, Princeton University Presss, 1957, Dover Edition, 2003.

[3] Richard Bellman, *Adaptive Control Processes: A Guided Tour*, Princeton University Press, 1961.

[4] E. Beckenbach and Richard Bellman, *An Introduction to Inequalities*, The Mathematical Association of America, 1961

[5] Richard Bellman and S. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, 1962.

[6] Richard Bellman and R. Kalaba, *Dyanamic Programming and Modern Cintrol Theory*, Academic Press, New York, 1965.

[7] Richard Bellman, *Introduction to Matrix Analysis*, MacGraw-Hill, 1970, Republished by SIAM (2nd ed.), 1997.

[8] Richard Bellman, K. I. Cooke, and J. A. Lockett, *Algorithms Graphs and Computers*, Academic Press, 1970.

[9] W. Stallings, *Data and Computer Communications* (10th ed.) Pearson, UK, 2014.

[10] W. Stallings, *Wireless Communications and Networks* (2nd ed.) Pearson, USA, 2005.

[11] Y. Okuyama, "Stability and Security Analyses of Event-Driven Discrete Control Systems Based on Lattice Theory", *Proc. of SICE*, Nara, Japan, 2018.

[12] Y. Okuyama, "Stability Analysis of Discrete Event Control Systems Based on Connection Matrices and Graphs", *Proc. of the 12th Asian Control Conference*, Kitakyushu, Japan, 2019.

[13] Y. Okuyama, *Discrete Control Systems*, Springer-Verlag, London, 2014.

[14] R. Alur, T. A. Henzinger, and E. D. Sontag (des), *Hybrid Systems III - Verification and Control* - , Springer-Verlag, Berlin, 1996.

[15] X. Koutsoukos, P. J. Antsaklis, J. A. Stiver, and M. D. Lemmon, "Supervisory Control of Hybrid Systems", *Proc. of the IEEE*, pp. 1026-1049, 2000.

[16] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems, - Modeling, Stability, and Robustness* -, Princeton University Press, 2012.

[17] E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs", *Numerische Mathematik*, **1**, pp. 269-271, 1959.

[18] L. R. Ford Jr. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.