Application of Critical Path Method to Stochastic Processes with Historical Operation Data

Yuya Takakura[a], Tomoyuki Yajima[b], Yoshiaki Kawajiri[*,c], Susumu Hashizume[d]

[*] Corresponding author

Affiliations

[a, b, c] Dept. of Materials Process Engineering, Nagoya University, Nagoya, Japan.
[c] Dept. of Chemical & Biomolecular Engineering, Georgia Institute of Technology, USA.
[d] Dept. of Control Engineering, National Institute of Technology, Nara College, Nara, Japan.

E-mail address

[a] takakura.yuuya@g.mbox.nagoya-u.ac.jp
[b] yajima@nuce.nagoya-u.ac.jp
[c] kawajiri@nagoya-u.jp
[d] hashizume@ctrl.nara-k.ac.jp

Notation

INDICES

s	Source task
e	Sink task
i	Task
j	Path from source to sink
k_i	Index of random variable $t_i^{(k_i)}$
v_j	Index of random variable $T^{(v_j)}$
k_i'	Realization of integer k_i
v_j'	Realization of integer v_j
$v_{j\mathrm{crit}}$	Realization of integer v_j that satisfies Eq. (32)
v_j^*	Realization of integer v_j that corresponds to the optimal solution of *Path-Oriented Formulation* proposing in Section 5
v_j^{ref}	Realization of integer v_j for local search method in Section 6
adj	Index of a task next to task i

i' Task that improvement of dispersion is allowed in Supplementary Material E

SETS

W Set of tasks except for source and sink

V Set of paths

p_j Set of tasks on path j except for dummy tasks

A Set of realizations for $k_i, i \in W$

M Set of realizations for $v_j, j \in V$

$\bar{M}_j$ Set of realizations for the local search method

E Set of arcs in process network

$H^{(v_1, v_2, \cdots, v_r)}$ Set that includes all $(k_1, k_2, \cdots, k_n)$ satisfying condition $\sum_{i \in p_j} t_i^{(k_i)} \le T_j^{(v_j)}, j \in V$

$\bar{H}^{(v_1, v_2, \cdots, v_r)}$ Set that includes all $(k_1, k_2, \cdots, k_n)$ satisfying condition $\sum_{i \in p_j} t_i^{(k_i)} \le T_j^{(v_j)}, j \in V$ for the local search method

PARAMETERS

n Number of tasks

r Number of paths

c_i^U Crash cost of task i

$\mathbf{c_i}^{\mathbf{U}}$ Vector of the crash cost of task i

C Maximum total cost

Γ Target process completion time

$\alpha_i^{(k_i)}$ Probability in bin k_i

$N_i[t_i^{(k_i)}]$ Number of samples within $t_i^{(k_i-1)} < t_i \le t_i^{(k_i)}$

λ_i Proportionality constant for reducing duration of task i per additional cost in classical CPM formulation

$\lambda_{M,i}$ Proportionality constant for reducing expected value of duration of task i per additional cost in

proposing formulations

$\lambda_{D,i}$ Proportionality constant for dispersion of duration of task *i* per additional cost in proposing formulations

t_i^{ave} Average duration of task *i* without improvement

$t_i^{(k_i)}$ Duration of task *i* without improvement in discretized form

$T^{(v_j)}$ Discretized time

$h^{(v_1, v_2, \cdots, v_r)}$ Probability that sum of task durations $\sum_{i \in p_j} t_i^{(k_i)}$ is smaller than $T_j^{(v_j)}$ for all paths, $j \in V$

β Search range in the local search method

$c_{M,i}^{\text{ref}}$ Realization of allocating cost on task *i* for the local search method as a reference value

$c_{M,i}^{*}$ Optimal value of $c_{M,i}$ of the *Path-Oriented Formulation* in Section 5

a Bin width for histograms in case studies

CONTINUOUS VARIABLES

c_i Allocating cost to task *i* in the classical CPM formulation

$\mathbf{c_i}$ Vector of allocating cost to task *i* in proposing formulations

$c_{M,i}$ Allocating cost to reduce expected value of task duration

$c_{D,i}$ Allocating cost to reduce dispersion of task duration

t_i Duration of task *i* without improvement

$\tilde{t}_i$ Duration of task *i* after improvement

$\tilde{t}_i^{(k_i)}$ Duration of task *i* after improvement in discretized model

BINARY VARIABLES

$x^{(k_1, k_2, \cdots, k_n)}$ Binary variable in the *Task-Oriented Formulation* in Section 4

$z^{(v_1, v_2, \cdots, v_r)}$ Binary variable in the *Path-Oriented Formulation* in Section 5

1. Introduction

Network-based methods have been applied to many project management problems. A project is a set of activities with objectives and completion deadlines. Various projects such as production, building construction, system development, and research projects can be modeled as networks, which can be handled by advanced management techniques. In many cases, completing a project by the deadline is the most critical requirement. For example, meeting the delivery date of a product, which is set as a deadline, is a critical constraint for many manufacturers.

The *Critical path method* (CPM) and the *Project evaluation and review technique* (PERT) are widely used as network-based methods for project management, which were proposed for the first time in 1950s (Malcolm et al., 1959) . The PERT method identifies the critical path (bottleneck), which is a sequence of tasks that has the longest duration in a project network. Identifying the critical path is a key step for project management because it determines the completion time of the project. In CPM problems, the trade-offs can be analyzed between cost allocation and shortening project the completion time, in addition to identifying the critical path.

PERT/CPM methods are powerful techniques also for many chemical engineering problems (Kopanos et al., 2014; Kyriakidis et al., 2012). For example, in batch chemical processes where multiple tasks can be expressed as project network, finding the capital investment strategy to optimize the scheduling can be formulated as a CPM problem. Furthermore, the PERT method can be applied to scheduling problems such as estimating uncertain completion times in chemical processes. These problems that can be handled by the PERT and CPM methods exist in many process industries such as steel, pharmaceuticals, semiconductor, and food. In addition, the PERT and CPM methods can be applied to construction of chemical plants as previously reported (Walton, 1964).

In classical PERT/CPM methods, task durations are handled as fixed values. However, actual task durations in many real applications have uncertainty for unpredictable reasons such as weather, human resources, equipment failure, etc. Examples of production systems with uncertain task durations are chemical and steel processes. Furthermore, there can be manual operations where the durations are highly uncertain. To represent realistic project management scenarios, handling uncertainty is often crucial, and various modifications have been proposed for the classical PERT/CPM methods. In following discussion, we discuss a classification based on the following three viewpoints: modeling of path durations, handling time-cost trade-off, and solving the problems by heuristic or deterministic approaches.

Here we classify approaches for modeling uncertainty in task or path durations into the following four classes: fixed time (point estimate), fuzzy numbers, probability distributions (continuous distributions), and histogram (discrete distributions). The first approach is to regard task durations using a few representative values. A common approach in this class is the *three-point estimation method*, where the distributions of the task durations are approximated by the following three values—the most likely, most optimistic, and most pessimistic values. The classical PERT method employs this method. However, it has been reported that approximations based on this approach may deviate from the true distributions (Hajdu and Bokor, 2014). The second approach is to regard task durations as fuzzy numbers described a membership functions. Examples of methods that use fuzzy numbers have been proposed by Sadjadi et al. (2012), Kaur and Kumar (2014), Xu et al. (2012), and Chen and Hsueh (2008). The third approach is to assume the task durations follow some known continuous distribution functions. For example, in Golenko-Ginzburg and Gonik (1997), it is assumed that the task durations follow normal or beta distributions. The fourth approach is to handle task or path durations as histograms, as in Herroelen and Leus (2004) and Bruni et al. (2009). Since the actual data of task or path durations is given as discrete values, handling the data histograms without approximation can be an advantage. In this paper, we also employ the fourth approach for production systems where histograms of task durations can be obtained from historical operation data.

The time-cost trade off problem is one form of scheduling methods to find the optimal cost allocation to tasks or paths. For this problem, there have been various formulations proposed for the objective function. Some examples include Kelley and Walker (1959), Xu et al. (2012), and Hasuike (2013); among these studies, in Kelley and Walker (1959), the objective function is to minimize the project completion time or total allocation cost. In Xu et al., (2012), four kinds of optimization problems that have different objective functions were proposed such as minimizing the total allocating cost. On the other hand, Hasuike (2013) aimed to maximize an approximated project completion probability within a target completion time. In this paper, we employ a similar objective function as in Hasuike (2013) where the probability is maximized, while in this paper we evaluate the objective function is without approximation. For time-cost trade off problems, there are two approaches to model improvement of task or path durations by allocating costs to tasks or paths. The first approach is continuous improvement per allocating costs, which assumes task or path durations decreases in proportion to allocated costs continuously. In this approach, the relationship between the allocated cost and reduction of task (or path) durations must be modeled as a continuous function to formulate an optimization problem (Hasuike, 2013; Kelley and Walker, 1959; Xu et al., 2012). The other approach is to employ discrete improvement per allocating costs, which assumes task or path durations have discrete candidates by allocating some cost. In this approach, the relationship between the allocated cost and reduction of task (or path) durations must be modeled as a discrete function to formulate an optimization problem (Tao et al., 2017).

The other form of the scheduling method does not consider time-cost trade off. These methods generally have a single objective to estimate some indicators of the target project. Examples of these methods were proposed by Malcolm et al., (1959), Hajdu and Bokor (2014), Ke and Liu (2005). Among these studies, the classical PERT method (Malcolm et al., 1959) which aim to estimate the project completion time. Bruni et al. (2009) proposed a way to estimate minimum makespan that probability or "reliability level" of project completion is larger than a constant. Ke and Liu (2005) proposed three approaches to estimate such as minimum cost expectation value or probability that the cost exceeds the budget. In addition, Hajdu and Bokor (2014) proposed a way to estimate how the shape of the distribution function of each task durations affects the project completion time.

We can also classify the project scheduling methods with uncertainty based on the solution approaches. These problems can be solved either by a heuristic or deterministic algorithm. Due to the difficulty of the CPM problems with uncertainty, many heuristic approaches have been proposed. Heuristic algorithms are often used for finding approximate solutions when a short computational time is desired, or the problem is difficult to be formulated as a deterministic problem. Such approaches include the Monte Carlo simulation (Li and Womer, 2015), genetic algorithm (Azaron et al., 2005), and combining these two numerical techniques (Huang and Ding, 2011; Ke and Liu, 2005). On the other hand, in deterministic approaches, the exact solution can be found without an approximation. Examples include Linear Programming (LP) by Kelley and Walker (1959), LP using fuzzy parameters by Kaur and Kumar (2014), and Mixed-Integer Linear Programming (MILP) by Bruni et al. (2009).

In this paper, we propose a method to expand applications of CPMs in chemical and process engineering by considering uncertainty of task durations. Our proposed method is based on deterministic optimization where the task durations are handled as histograms considering time-cost trade-off. The proposed approach has the following three advantages. Firstly, by handling task durations as histograms, we can handle operation data without approximation and losing information. Secondly, we can consider time-cost trade off problem and find the optimal cost allocation that maximizes the project completion probability within a given completion time. Note that we consider two kinds of improvement of task durations to enable more flexible modeling, while past studies consider only a single way to improve task durations (Hasuike, 2013; Kelley and Walker, 1959; Xu et al., 2012). Thirdly, by formulating the problem as an MILP problem that can be solved by deterministic algorithm, the exact solution can be found without an approximation in contrast to the heuristic approaches where only an approximate solution can be found.

This paper is organized as follows. In Section 2, we discuss the classical CPM problem, and handling uncertainty as a background. In Section 3, we describe our assumptions in our problem formulation. In Section 4, we propose a new approach called the Task-Oriented Formulation. In Section 5, we reformulate the same

problem to decrease the computational time. In Section 6, we propose an iterative local search method to shorten the computational time. In Section 7, we show examples. In Section 8, we conclude this paper.

2. Background

2.1 CPM without uncertainty

The CPM is a method to analyze and optimize a project and production system. In this method, the entire project or production system is represented as a network figure that contains activities and sequences. There are two equivalent expressions for the CPM. In the *activity on node* type, activities with given durations are expressed as nodes. On the other hand, in the *activity on arc* type, they are expressed as arcs. These two expressions are equivalent, and in this paper, we use the former type. An example of activity on arc type project network is shown in Figure 1. In addition to the four activities, 1, 2, 3, and 4 shown in the middle of this figure, we consider a single source *s*, as well as a sink *e*. These source and sink are treated as dummy tasks that have no durations, where the solution remains the same. In this paper, we consistently refer to activities as *tasks*, since we consider only production systems. For the same reason, we call the project as a *process* and call the project completion time where all tasks are completed as *process completion time*.

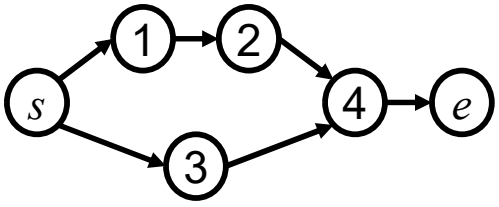

Figure 1. Example of project network [2-column fitting image]

In addition to finding the critical path in a process, this approach can also be used to identify the most efficient improvement to reduce the process completion time. The objective of this problem is to minimize the process completion time of the production system. The critical path, which is the path that has the longest time, determines the process completion time. Thus, the process completion time can be shortened by reducing the task durations on the critical path.

The CPM problems without uncertainty, which aim to minimize the process completion time, can be formulated as min-max problem which assumes that task durations are reduced linearly with additional cost.

$$\begin{aligned} &\text{Minimize}: \max_{j \in V} \left\{ \sum_{i \in p_j} t_i - \sum_{i \in p_j} \lambda_i c_i \right\} \\ &\text{s.t.} \quad \sum_{i \in W} c_i \le C \\ &\qquad 0 \le c_i \le c_i^U, \quad i \in W \end{aligned} \tag{1}$$

where *i* is an index for tasks; *n* is the number of tasks excluding source and sink; $W = \{1, 2, \cdots, n\}$ is the set of tasks except for source and sink; *j* is a path from source to sink; *r* is the number of paths from the source to sink; $V = \{1, 2, \cdots, r\}$ is the set of all paths from source to sink; p_j is the set of tasks excluding dummy tasks on path $j \in V$; t_i is the duration of task *i*; λ_i is the proportionality constant of duration of task *i* per additional cost; c_i^U is the crash cost (maximum cost) of task *i*; *C* is the maximum total cost. These sets and parameters above are assumed to be known. On the other hand, c_i, which is allocated cost of task $i \in W$, is a decision variable. Note that generally this problem can be formulated as Linear Programming (LP) (Kelley and Walker, 1959). The general formulation is shown as (A.1) in Supplementary Material A.

The classical CPM problem without uncertainty assumes that the task durations are known and constant. However, such an assumption is often invalid in chemical and steel processes. To accommodate uncertainty in the CPM, it has been attempted to use the average or mean of task duration, or three-point estimation values to represent the distribution of the task duration based on the most likely, most optimistic, and most pessimistic values. However, these approaches are unable to fully utilize the information of the task duration distributions.

2.2 CPM with uncertainty

We expand the concept of the CPM without uncertainty, and define "*CPM with uncertainty*" as follows: we aim to maximize the probability of finishing all tasks by a given target completion time Γ. Here, we define a cost vector of allocating some cost to task *i* as $\mathbf{c_i}$, assuming the maximum total cost is given as *C*. Additionally, the duration of task *i* has a probability distribution illustrated in Figure 2 (a), which can be improved by allocating some cost.

Under these assumptions, the problem is generally formulated as

$$\text{Maximize}: \Pr\left[\text{Process completion time} \le \Gamma\right] \tag{2}$$

$$\text{s.t.} \quad \sum_{i \in W} \left\| \mathbf{c_i} \right\| \le C, \tag{3}$$

$$\mathbf{0} \le \mathbf{c_i} \le \mathbf{c_i^U}, \quad i \in W \tag{4}$$

where $\left\| \mathbf{c_i} \right\|$ is the L1 norm of the vector of allocating cost $\mathbf{c_i}$, $\mathbf{c_i}^{\mathbf{U}}$ is a vector of the crash cost (maximum cost) of task *i*. In this formulation, the objective function in (2) is the probability of finishing all tasks by a target completion time Γ; generally, calculating the objective function (2) requires multi-dimensional integrations and convolutions, which are complicated operations that require the information of all paths (Kamburowski, 1992). Furthermore, to handle the task durations in the continuous time domain, it is necessary to model the probability distributions as some functional forms, which may require approximation. In this study, we avoid these two problems by discretizing the time domain, and converting this problem into a Mixed-Integer Linear Programming (MILP) problem as described in the next section.

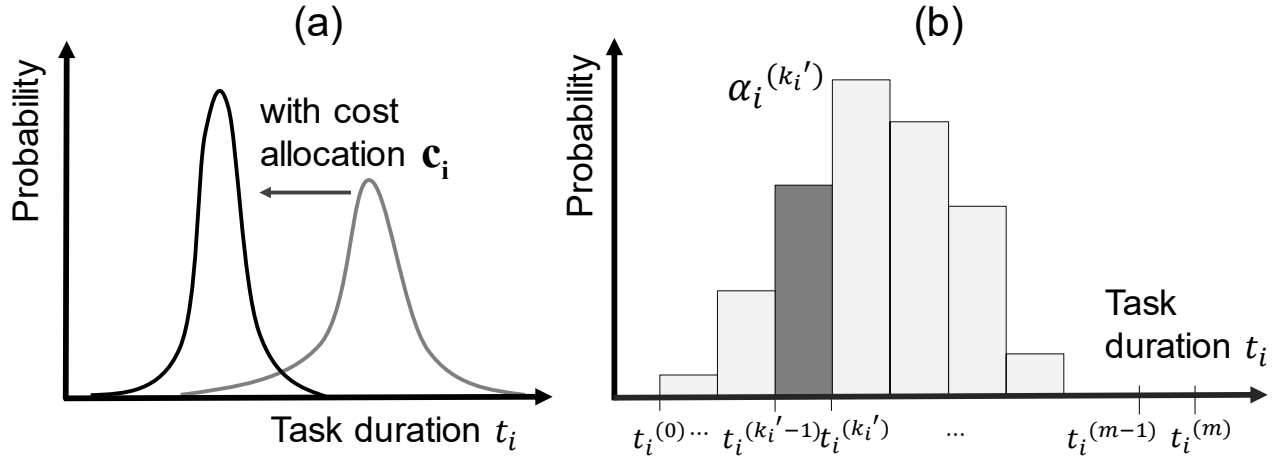

Figure 2. Probability distribution and histogram duration of task *i* [1-column fitting image]

3 . Assumption for proposed methods

3.1 Obtaining discrete probability from operation data

In this paper, we treat operation data as discrete probability distributions, which avoids approximation of the probability distribution to functional forms. Bruni et al. (2009) used discrete data for the same motivation. In our work, historical operation data is assumed to be available. Such data is an accumulated record of task durations, which can be shown graphically as histograms. This assumption does not require any approximation to probability distribution functions, and thus the solution is expected to be more accurate. We also assume that the duration of each task is independent (i.e. the duration of task *i* does not influence that of another task i', $i \neq i'$).

To generate a discretized distribution from the operation data, we discretize the duration of task *i* using an index k_i. Using this index, histograms of task distributions are discretized as illustrated in Figure 2 (b). Here, k_i is an integer which takes a value between 0 and m, and we define a set of all realizations of k_i as A.

$$A = \{0,1,2,\cdots,m-1,m\}. \tag{5}$$

The duration of task *i* without cost allocation, which is given as $t_i^{(k_i)}$, is uncertain and treated as a random variable, where the scenario (realization) is specified by the integer k_i. As a result of the discretization, $t_i^{(k_i)}$ has multiple scenarios; $t_i^{(0)}, t_i^{(1)}, \cdots, t_i^{(m-1)}, t_i^{(m)}$ (see Figure 2 (b)). Note that one realization of the discrete random variables from Task 1 to *n* as $\left(t_1^{(k_1)}, t_2^{(k_2)}, \cdots, t_n^{(k_n)}\right)$ corresponds to a scenario that duration of task *i* is within $t_i^{(k_i-1)} < t_i \le t_i^{(k_i)}$ for all tasks $i \in W$, where t_i is duration of task *i* (given as a continuous variable). Note that the number of all scenarios is m^n, which is a large number.

It should also be noted that the normalized distribution obtained from such data approaches the probability distribution as the historical record becomes larger with repeated operations. Figure 2 (b) is an illustration of the normalized distributions. We divide duration of task *i* into *m* bins, which give the probability $\alpha_i^{(k_i')}$ to satisfy $t_i^{(k_i)} = t_i^{(k_i')}$:

$$\begin{aligned} \alpha_i^{(k_i')} &= \Pr[t_i^{(k_i)} = t_i^{(k_i')}] \\ &= \frac{N_i[t_i^{(k_i')}]}{\sum_{k_i \in A} N_i[t_i^{(k_i)}]}, i \in W, \end{aligned} \tag{6}$$

where k_i' is the index for a certain realization of the random variable $t_i^{(k_i')}$, $N_i[t_i^{(k_i)}]$ is defined as the number of samples within $t_i^{(k_i-1)} < t_i \le t_i^{(k_i)}$.

3.2 Approaches to improve task durations

Improvement (or shortening) task durations can be performed in a number of different ways, and outcomes can be difficult to predict. Some examples of such improvement are as follows: increasing the number of operators, and preparing a training manual for operators. Some more examples in chemical industries include upgrading the catalyst in a reactor, and increasing the pump capacity in a liquid transport unit. As a result of such improvement measures, the task durations can be shortened, and in addition the profile of the probability distributions (or histograms of the task duration data) may change significantly. It would not be easy to model and predict the change of probability distributions.

In this study, we use simple approaches to model the improvement of task durations by using two parameters, the expected value (mean) and dispersion (variance) of the probability distributions. The first approach is to decrease the expected value of task durations linearly per additional cost while keeping the shape of the distribution as shown in Figure 3 (a). This approach shifts the entire distribution horizontally:

$$\tilde{t}_i^{(k_i)} = t_i^{(k_i)} - \lambda_{M,i} c_{M,i}, i \in W, \tag{7}$$

where $\tilde{t}_i^{(k_i)}$ is improved duration of task *i* by cost allocation; $c_{M,i}$ is allocated cost to reduce expected value of task duration; $\lambda_{M,i}$ is the decrease rate of $\tilde{t}_i^{(k_i)}$ per additional cost, which is assumed to be constant. We call this approach as *improvement of expected value*.

The second approach to improve the task duration is to narrow the dispersion while keeping the expected value of the distribution constant (Figure 3 (b)). In this approach, the task duration $\tilde{t}_i^{(k_i)}$ of task *i* after improvement is written as

$$\tilde{t}_i^{(k_i)} = t_i^{(k_i)} - \left(t_i^{(k_i)} - t_i^{\text{ave}} \right) \lambda_{D,i} c_{D,i}, i \in W, \tag{8}$$

where $c_{D,i}$ is allocated cost to reduce dispersion of task duration; and $\lambda_{D,i}$ is the proportionality constant of dispersion per additional cost; and t_i^{ave} is average duration of task *i* without improvement, which we define as:

$$t_i^{\text{ave}} = \frac{\sum_{k_i \in A} t_i^{(k_i)} N_i^{(k_i)}}{\sum_{k_i \in A} N_i^{(k_i)}}, i \in W. \tag{9}$$

We call this approach as *improvement of dispersion*.

While the *improvement of expected value* is used commonly as in Kelley and Walker (1959), and Xu et al. (2012), the *improvement of dispersion* is unique and other papers have not considered this approach. Our proposed framework can handle both approaches, which is demonstrated in a case study in Section 7.

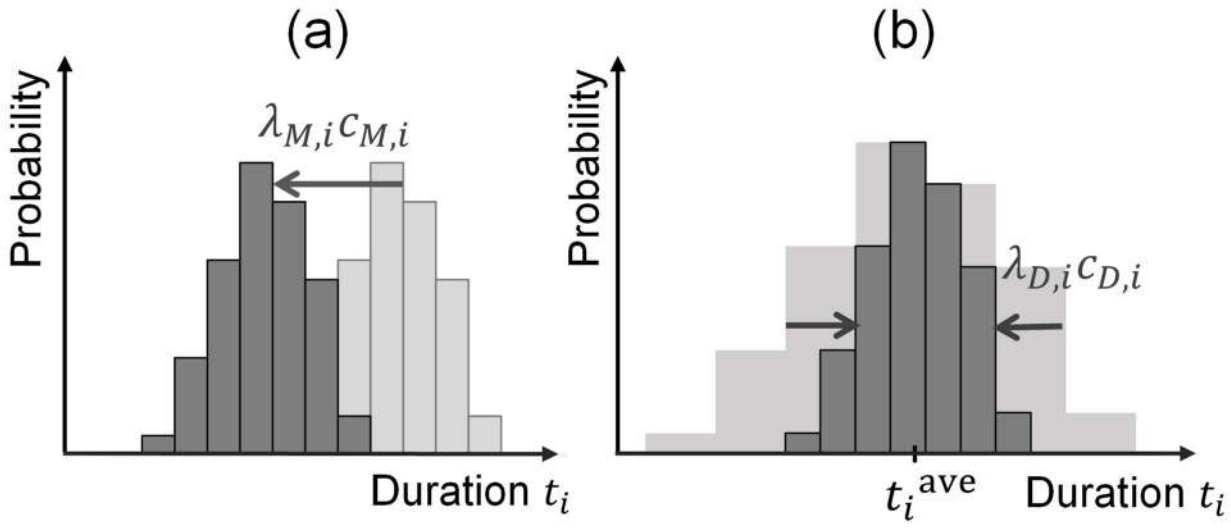

Figure 3. Improvement of duration of task *i* [1-column fitting image]

4. Task-Oriented Formulation

In this section, we propose a reformulation strategy for the problem given by the *CPM with uncertainty* (2) - (4) into a MILP problem. This formulation is referred to as the Task-Oriented Formulation, while we discuss another reformulation in Section 5.

4.1. Preparation for Task-Oriented Formulation

To avoid the complex operation in evaluating the objective function by convolution introduced in Kamburowski (1992), we convert the objective function (2) into a simple linear equation. From (6) we express the joint probability that all tasks finish within the shaded bins in Figure 4 as the multiplication of probabilities $\alpha_i^{(k_i')}$ for all tasks $i \in W$:

$$\Pr\left[t_i^{(k_i)} = t_i^{(k_i')}, i \in W \right] = \alpha_1^{(k_1')} \alpha_2^{(k_2')} \cdots \alpha_n^{(k_n')}. \tag{10}$$

i.e. Eq. (10) is the probability that each task *i* finishes in $t_i^{(k_i)} = t_i^{(k_i')}$ for $i \in W$. Here, it should be noted that there are a large number of combinations for the realizations of indices k_i in the random variables $t_i^{(k_i)}$,

$$\left(k_1, k_2, \cdots, k_n\right) \in A^n. \tag{11}$$

In this work, we assign binary variables $x^{(k_1, k_2, \cdots, k_n)} \in \{0,1\}$ that enumerate all possible realizations of the random variables. We consider a logic condition such that each of these random variables becomes positive only if the process completion time is equal to or shorter than Γ :

$$x^{(k_1, k_2, \cdots, k_n)} = \begin{cases} 1 & \text{if Project completion time} \le \Gamma \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

Using Eq. (10) and (12), we rewrite Eq. (2) in a discretized form as follows:

$$\begin{aligned} &\Pr\left[\text{Project completion time} \le \Gamma\right] \\ &= \sum_{\left(k_1', k_2', \cdots, k_n'\right) \in A^n} \Pr\left[t_i^{(k_i)} = t_i^{(k_i')}, i \in W \right] x^{\left(k_1', k_2', \cdots, k_n'\right)} \\ &= \sum_{\left(k_1', k_2', \cdots, k_n'\right) \in A^n} \alpha_1^{(k_1')} \alpha_2^{(k_2')} \cdots \alpha_n^{(k_n')} x^{\left(k_1', k_2', \cdots, k_n'\right)}. \end{aligned} \tag{13}$$

Here, instead of using the complex operations in evaluating Eq. (2) in the continuous time domain, we discretize the time domain and rely on the logic condition in Eq. (12). We implement this logic constraint within a framework of integer programming as shown below.

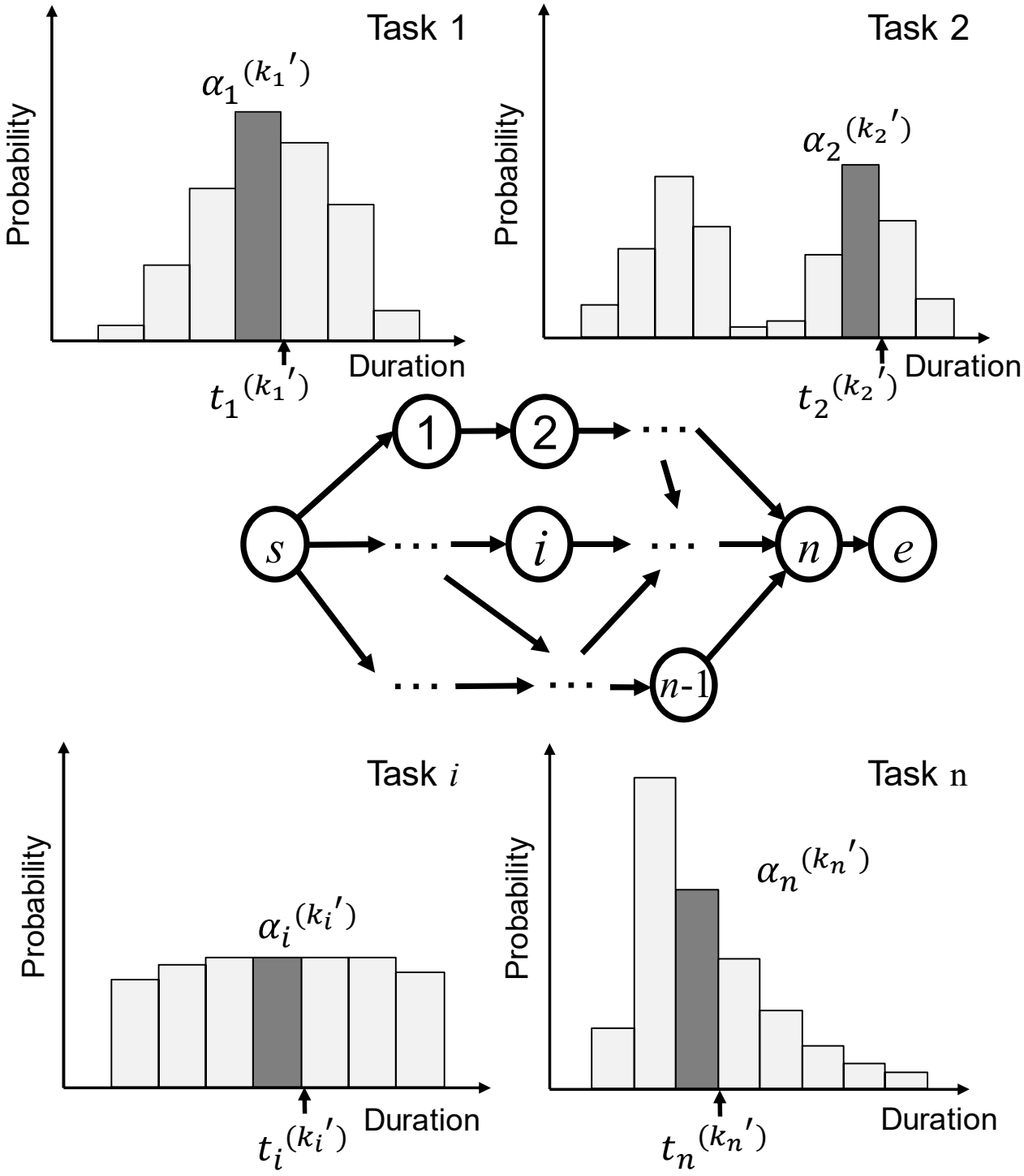

Figure 4. Concept of Equation (10) [1-column fitting image]

4.2. Task-Oriented Formulation

The optimization problem given in Eq. (2) - (4) can be reformulated into the following form using Eq. (12) and (13). Firstly we show overall formulation and then discuss each constraint.

***Task-Oriented Formulation* (*TOF*)**

$$\text{Maximize:} \sum_{(k_1,k_2,\cdots,k_n)\in A^n} \alpha_1^{(k_1)}\alpha_2^{(k_2)}\cdots\alpha_n^{(k_n)}\, x^{(k_1,k_2,\cdots,k_n)} \tag{14}$$

$$\text{s.t.} \quad x^{(k_1,k_2,\cdots,k_n)} \in \{0,1\}, \left(k_1,k_2,\cdots,k_n\right) \in A^n \tag{15}$$

$$\left(\sum_{i\in p_j} t_i^{(k_i)}\right) x^{(k_1,k_2,\cdots,k_n)} - \sum_{i\in p_j}\lambda_{M,i}c_{M,i} - \sum_{i\in p_j}\left(t_i^{(k_i)} - t_i^{\text{ave}}\right)\lambda_{D,i}c_{D,i} \le \Gamma, j\in V, \quad \left(k_1,k_2,\cdots,k_n\right)\in A^n \tag{16}$$

$$\sum_{i\in W} c_{M,i} + \sum_{i\in W} c_{D,i} \le C \tag{17}$$

$$\begin{aligned} 0 \le c_{M,i} \le c_{M,i}^U, \quad i\in W \\ 0 \le c_{D,i} \le c_{D,i}^U, \quad i\in W, \end{aligned} \tag{18}$$

In this problem, the decision variables are $x^{(k_1,k_2,\cdots,k_n)}$ as well as the allocating costs

$$\mathbf{c_i} = \begin{bmatrix} c_{M,i} \\ c_{D,i} \end{bmatrix}, i\in W, \tag{19}$$

where the costs $c_{M,i}$ and $c_{D,i}$, which are two improving approaches are introduced in Section 3.2, are vector elements of allocating cost $\mathbf{c_i}$ defined in (3) in this formulation. It can be seen that Eq. (17) can be given by substituting (19) into Eq. (3).

It is critical to note that Eq. (16) is a constraint for the duration of path *j* that realizes the logic condition (12). Here we recall $\tilde{t}_i^{(k_i)}$ is the duration after improvement from $t_i^{(k_i)}$. Since $\tilde{t}_i^{(k_i)}$ is a result of two kinds of improvement; the *improvement of expected value* (7) and the *improvement of dispersion* (8), $\tilde{t}_i^{(k_i)}$ can be written as

$$\tilde{t}_i^{(k_i)} = t_i^{(k_i)} - \lambda_{M,i}c_{M,i} - \left(t_i^{(k_i)} - t_i^{\text{ave}}\right)\lambda_{D,i}c_{D,i}, i\in W. \tag{20}$$

Here summing (20) over all tasks on path *j* gives

$$\sum_{i\in p_j}\tilde{t}_i^{(k_i)} = \sum_{i\in p_j} t_i^{(k_i)} - \sum_{i\in p_j}\lambda_{M,i}c_{M,i} - \sum_{i\in p_j}\left(t_i^{(k_i)} - t_i^{\text{ave}}\right)\lambda_{D,i}c_{D,i}, i\in W. \tag{21}$$

Here, we express the process completion time as the maximum value of $\sum_{i\in p_j} \tilde{t}_i^{(k_i)}$ among all paths $j \in V$,

$$(\text{Process completion time}) = \max_{j\in V} \sum_{i\in p_j} \tilde{t}_i^{(k_i)} \tag{22}$$

From (22), we rewrite the condition that the process completion time is below Γ:

$$\begin{aligned} &(\text{Process completion time} \le \Gamma) \\ &\Leftrightarrow \max_{j\in V} \sum_{i\in p_j} \tilde{t}_i^{(k_i)} \le \Gamma \\ &\Leftrightarrow \sum_{i\in p_j} \tilde{t}_i^{(k_i)} \le \Gamma, j \in V. \end{aligned} \tag{23}$$

Substituting (21) into the bottom inequality in (23) gives

$$\sum_{i\in p_j} t_i^{(k_i)} - \sum_{i\in p_j} \lambda_{M,i} c_{M,i} - \sum_{i\in p_j} \left(t_i^{(k_i)} - t_i^{\text{ave}} \right) \lambda_{D,i} c_{D,i} \le \Gamma, j \in V. \tag{24}$$

Here we compare Eqs.(16) and (24), and note (16) can be obtained by multiplying binary variables $x^{(k_1,k_2,\cdots,k_n)}$ to the first term of (24). If the left hand side of (24) is over Γ, the binary variables $x^{(k_1,k_2,\cdots,k_n)}$ become zero because the constraint (16) can be satisfied only when $x^{(k_1,k_2,\cdots,k_n)}$ are zero. On the other hand, if the left hand side of (24) is below Γ, the binary variables $x^{(k_1,k_2,\cdots,k_n)}$ can take either value, but the number of positive binary variables $x^{(k_1,k_2,\cdots,k_n)}$ is maximized since the objective function (14) should be maximized. From the discussion above, we see the logic condition (12) is rewritten to the constraints (16).

We note that the problem size of the above formulation *TOF* is very large due to the large number of binary variables $x^{(k_1,k_2,\cdots,k_n)}$. The number of the binary variables $x^{(k_1,k_2,\cdots,k_n)}$ is m^n as they are defined for enumerations $(k_1,k_2,\cdots,k_n) \in A^n$ for all possible random variables $t_i^{(k_i)}, i \in W$. In the next section, we show another formulation that reduces the problem size.

5. Path-Oriented Formulation

We show an alternative formulation to (2) - (4) that has a smaller number of decision variables than *TOF*. The large number of decision variables in *TOF* was due to the large number of discretized bins for task durations. In the reformulation given below, we reduce the decision variables by considering the duration of each path, instead of each task. This reduction in the problem size is possible because the number of all paths in a process is significantly smaller than the total number of tasks. Note that in this reformulation, we only consider the *improvement of the expected value* (7) ignoring the *improvement of dispersion* (8) (i.e. $c_{D,i} = 0, i \in W$).

5.1 Preparation for Path-Oriented Formulation

Firstly we rewrite Eq. (21), and show that the *improvement of dispersion* discussed in Section3.2 cannot be considered in this formulation. Assuming the bin width in the histogram of each task is constant, we define discrete time to describe path durations as

$$T^{(v_j)} = \sum_{i \in p_j} t_i^{(k_i)}, j \in V \tag{25}$$

in which the index v_j for the random variable $T^{(v_j)}$ is within a set M:

$$v_j \in M,\ M = \{0,1,2,\cdots,l-1,l\}, \tag{26}$$

where l is the largest bin number of $v_j, j \in V$. By substituting (26) into (21), we obtain

$$\sum_{i \in p_j} \tilde{t}_i^{(k_i)} = T^{(v_j)} - \sum_{i \in p_j} \lambda_{M,i} c_{M,i} - \sum_{i \in p_j} \left(t_i^{(k_i)} - t_i^{\text{ave}} \right) \lambda_{D,i} c_{D,i}, j \in V. \tag{27}$$

Eq. (27) have two sets of indices, k_i and v_j. Here, v_j, the index in the random variable for the paths, is dependent on that for the tasks, k_i, and the relationship between them cannot be expressed explicitly. In this work, we eliminate k_i from Eq. (27) by ignoring the *improvement of dispersion* as $c_{D,i} = 0, i \in W$.

Under this assumption, we can express $\sum_{i \in p_j} \tilde{t}_i^{(k_i)}$ only with the random variable for the paths $T^{(v_j)}$ as:

$$\sum_{i \in p_j} \tilde{t}_i^{(k_i)} = T^{(v_j)} - \sum_{i \in p_j} \lambda_{M,i} c_{M,i}, j \in V. \tag{28}$$

Using Eq. (28), we rewrite Eq. (2) in a useful form. By substituting (28) into (23), we obtain

$$\begin{aligned}&\Pr\left[\text{Process completion time} \le \Gamma\right]\\&= \Pr\left[T^{(v_j)} - \sum_{i \in p_j} \lambda_{M,i} c_{M,i} \le \Gamma, j \in V \right].\end{aligned} \tag{29}$$

While (29) is important since it is another expression of the objective function (2), here we introduce some definitions that help rewriting (2) using (29) in a useful form. Here we define a parameter $h^{\left(v_1', v_2', \cdots, v_r'\right)}$ which is the probability that the discrete time $T^{(v_j)}$ is smaller than $T^{\left(v_j'\right)}$ for all paths $j \in V$:

$$h^{\left(v_1', v_2', \cdots, v_r'\right)} = \Pr\left[T^{(v_j)} \le T^{\left(v_j'\right)}, j \in V \right], \tag{30}$$

$$\left(v_1', v_2', \cdots, v_r'\right) \in M^r, \tag{31}$$

where v_j' is the index for a given realization of the random variable $T^{\left(v_j'\right)}$. We show the concept of (30) in Figure 5, in which histograms of path durations are shown and the bins satisfying $T^{(v_j)} \le T^{\left(v_j'\right)}$ are colored with gray. Here we define *critical duration of path j* that is equivalent to the target completion time Γ after allocating costs $c_{M,i}, i \in W$ as

$$T^{\left(v_{j\,\mathrm{crit}}\right)} - \sum_{i \in p_j} \lambda_{M,i} c_{M,i} = \Gamma. \tag{32}$$

By substituting (32) into (30),

$$\begin{aligned}h^{\left(v_{1\,\mathrm{crit}}, v_{2\,\mathrm{crit}}, \cdots, v_{r\,\mathrm{crit}}\right)} &= \Pr\left[T^{(v_j)} \le T^{\left(v_{j\,\mathrm{crit}}\right)}, j \in V \right]\\&= \Pr\left[T^{(v_j)} - \sum_{i \in p_j} \lambda_{M,i} c_{M,i} \le \Gamma, j \in V \right].\end{aligned} \tag{33}$$

By substituting (29) into (33), we can rewrite (2) by using $h^{(v_1, v_2, \cdots, v_r)}$ as

$$\begin{aligned}&\Pr\left[\text{Process completion time} \le \Gamma\right]\\&= h^{\left(v_{1\,\mathrm{crit}}, v_{2\,\mathrm{crit}}, \cdots, v_{r\,\mathrm{crit}}\right)}.\end{aligned} \tag{34}$$

Note that the parameter $h^{(v_1, v_2, \cdots, v_r)}$ is calculated by summing the probability (10) as given in Supplementary Material B. This calculation to prepare the parameters $h^{(v_1, v_2, \cdots, v_r)}$ should be executed before solving optimization problems shown in *POF* in the next Section 5.2 and (D.4) in Supplementary Material D. An example for this calculation is shown in case studies in Section 7.

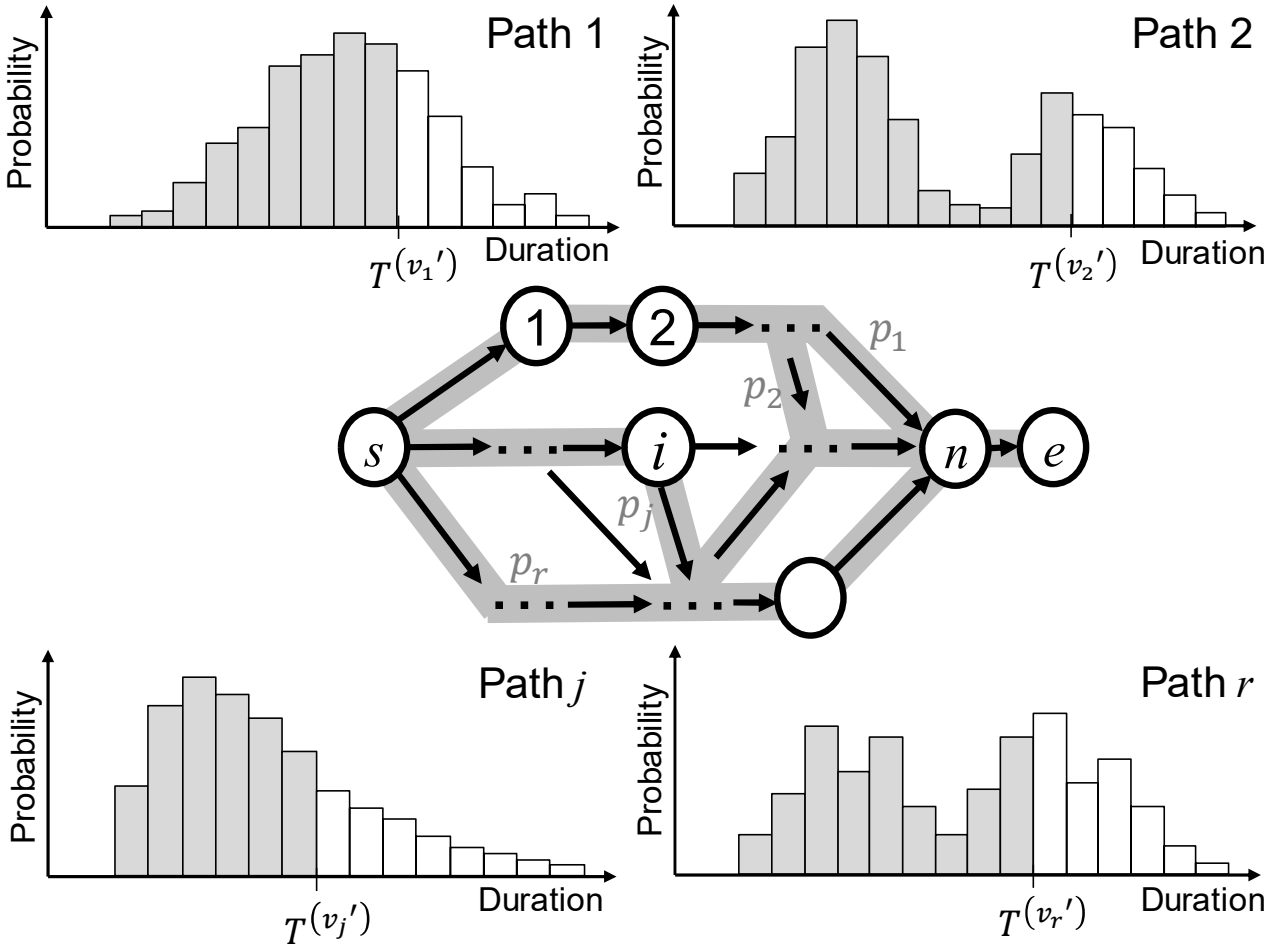

Figure 5. Concept of Equation (30) [1-column fitting image]

5.2 Path-Oriented Formulation

From the discussion above, (2) - (4) can be reformulated into an even simpler form. Firstly we show the overall reformulation and discuss the objective function and constraints later.

***Path-Oriented Formulation* (*POF*)**

$$\text{Maximize: } \sum_{(v_1,v_2,\cdots,v_r)\in M^r} h^{(v_1,v_2,\cdots,v_r)} z^{(v_1,v_2,\cdots,v_r)} \tag{35}$$

$$\text{s.t. } \sum_{(v_1,v_2,\cdots,v_r)\in M^r} z^{(v_1,v_2,\cdots,v_r)} = 1 \tag{36}$$

$$T^{(v_j)} \cdot z^{(v_1,v_2,\cdots,v_r)} - \sum_{i\in p_j} \lambda_{M,i} c_{M,i} \le \Gamma,\ j \in V \tag{37}$$

$$\sum_{i\in W} c_{M,i} \le C \tag{38}$$

$$0 \le c_{M,i} \le c_{M,i}^U,\quad i \in W \tag{39}$$

$$z^{(v_1,v_2,\cdots,v_r)} \in \{0,1\},\ (v_1,v_2,\cdots,v_r) \in M^r \tag{40}$$

where $z^{(v_1,v_2,\cdots,v_r)} \in \{0,1\}$ are binary variables that enumerate all possible realizations of the random variables. Decision variables are the binary variables $z^{(v_1,v_2,\cdots,v_r)}$ and allocating costs of task *i* as $c_{M,i}$. The correspondence between two formulations, the *CPM without uncertainty* and the *POF* is as follows: (2)→{(35), (36), (37), (40)}, (3)→(38), and (4)→(39).

We consider a logic condition such that only one of the binary variables $z^{(v_1,v_2,\cdots,v_r)} \in \{0,1\}$ becomes positive when the condition (32) is satisfied, and other binary variables become zero:

$$z^{(v_1',v_2',\cdots,v_r')} = \begin{cases} 1 \text{ if } T^{(v_j')} = T^{(v_{j\text{crit}})}, j \in V \\ 0 \text{ otherwize} \end{cases}. \tag{41}$$

By multiplying $h^{(v_1,v_2,\cdots,v_r)}$ to $z^{(v_1,v_2,\cdots,v_r)}$ and summing them for all combination of realization of indices $(v_1,v_2,\cdots,v_r)$, $h^{(v_{1\text{crit}},v_{2\text{crit}},\cdots,v_{r\text{crit}})}$ can be expressed as

$$\sum_{(v_1,v_2,\cdots,v_r)\in M^r} h^{(v_1,v_2,\cdots,v_r)} z^{(v_1,v_2,\cdots,v_r)} = h^{(v_{1\text{crit}},v_{2\text{crit}},\cdots,v_{r\text{crit}})}, \tag{42}$$

which is obtained from the logic condition (41). By substituting (42) into (34), it can be shown that Eq. (2) is equivalent with (35) as follows:

$$\Pr[\text{Project completion time} \le \Gamma] = \sum_{(v_1, v_2, \cdots, v_r) \in M^r} h^{(v_1, v_2, \cdots, v_r)} z^{(v_1, v_2, \cdots, v_r)}, \tag{43}$$

while satisfying the logic condition (41).

In *POF*, the logic condition (41) is satisfied in the optimal solution of the (*POF*). The proof is given in Supplementary Material C.

6. Path-Oriented Formulation with Local Search Algorithm

In this section, we propose a local search method to the problem *POF* in order to further reduce decision variables and constraints of *POF*. We denote the *Path-Oriented Formulation with Local Search Algorithm* proposed in this section as (*POF, Local Search*). On the other hand, we call the original *POF* as (*POF, Strict*). A comparison between (*POF, Strict*) and (*POF, Local Search*) is illustrated in Figure 6. Note that the formulation (*POF, Strict*) finds certain indices of discrete random variables $\left(v_1^*, v_2^*, \cdots, v_r^*\right)$ that correspond to the optimal solution of (*POF, Strict*) that satisfies $z^{\left(v_1^*, v_2^*, \cdots, v_r^*\right)} = 1$. This method uses the full search from all candidates of $\left(v_1^*, v_2^*, \cdots, v_r^*\right)$, where the search range is (31). On the other hand, in (*POF, Local Search*), we limit the search within a local region and consider a limited number of candidates for $\left(v_1^*, v_2^*, \cdots, v_r^*\right)$, which is around $\left(v_1^{\text{ref}}, v_2^{\text{ref}}, \cdots, v_r^{\text{ref}}\right)$ that is a realization of some combination of the indices:

$$\bar{M}_j = \left\{ v_j^{\text{ref}} - \beta, \cdots, v_j^{\text{ref}} - 1, v_j^{\text{ref}}, v_j^{\text{ref}} + 1, \cdots, v_j^{\text{ref}} + \beta \right\}, \tag{44}$$

where β is a parameter. Using (44), the narrowed candidates of index for discrete random variables is given by:

$$\left(v_1, v_2, \cdots, v_r\right) \in \bar{M}_1 \times \bar{M}_2 \times \cdots \times \bar{M}_r. \tag{45}$$

After this local search, we search the neighboring regions until the algorithm terminates.

We note that the local search method does not guarantee to find the optimal solution of the original problem, (*POF, Strict*). Since the candidates of the binary variable $z^{\left(v_1^*, v_2^*, \cdots, v_r^*\right)}$ are limited, the optimal solution of (*POF, Strict*) may not be found by (*POF, Local Search*) if (45) does not contain $\left(v_1^*, v_2^*, \cdots, v_r^*\right)$. This disadvantage must be weighed carefully against the advantage of the shorter computational time as demonstrated in our case study. Further details on the proposed local search method are given in Supplementary Material D.

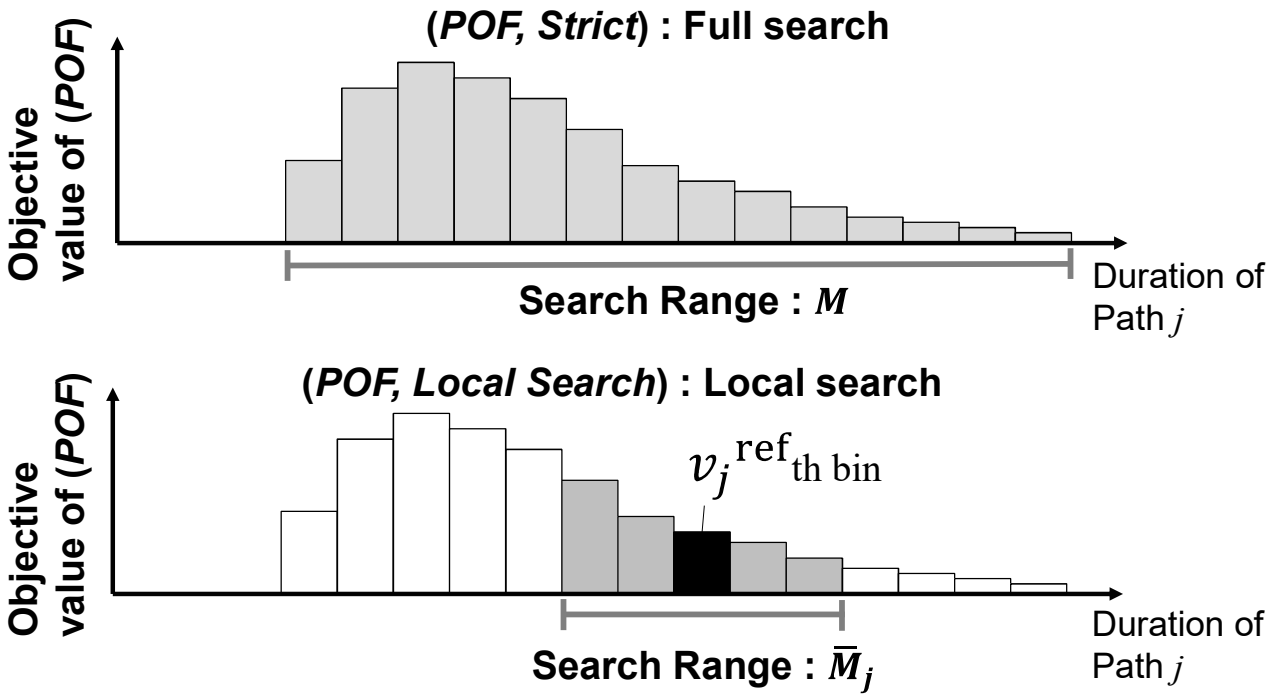

Figure 6. Comparison between the (*POF, Strict*) and (*POF, Local Search*) [1-column fitting image]

7. Case studies

In this section, we present some examples to demonstrate the proposed methods. We compare the following three approaches: the *CPM without uncertainty*, the proposed two formulations, *TOF* and *POF*. In these examples, we assume that the widths of all bins are constant. Thus, we have

$$t_i^{(k_i)} - t_i^{(k_i-1)} = a, \ i \in W \tag{46}$$

In the following case studies, we set $a = 5$. We implemented these approaches on a desktop personal computer with a core i7, 3.4GHz processor. The problem is solved by Numerical Optimizer from NTT DATA Mathematical Systems Inc. (Tokyo, Japan). The algorithm in this solver is based on the branch-and-bound method.

7.1 Example 1

In this example, we consider a production process shown in Figure 7, which shows the structure as well as the normalized distribution of each task. Table 1 shows the historical data for task duration $N_i[t_i^{(k_i)}]$ and given parameters. The sum of samples $\sum_{k_i \in A} N_i[t_i^{(k_i)}]$ are also shown for all tasks. Note that we introduced the historical data for task duration $N_i[t_i^{(k_i)}]$ in Section 3.1. For simplicity, we express $N_i[t_i^{(k_i)}]$ as $N_i^{(k_i)}$. In this example, the maximum total cost and process completion time are given as follows: $C = 250$ and $\Gamma = 100$. Note that the shortest and longest possible makespan are 70 and 165, respectively in this example.

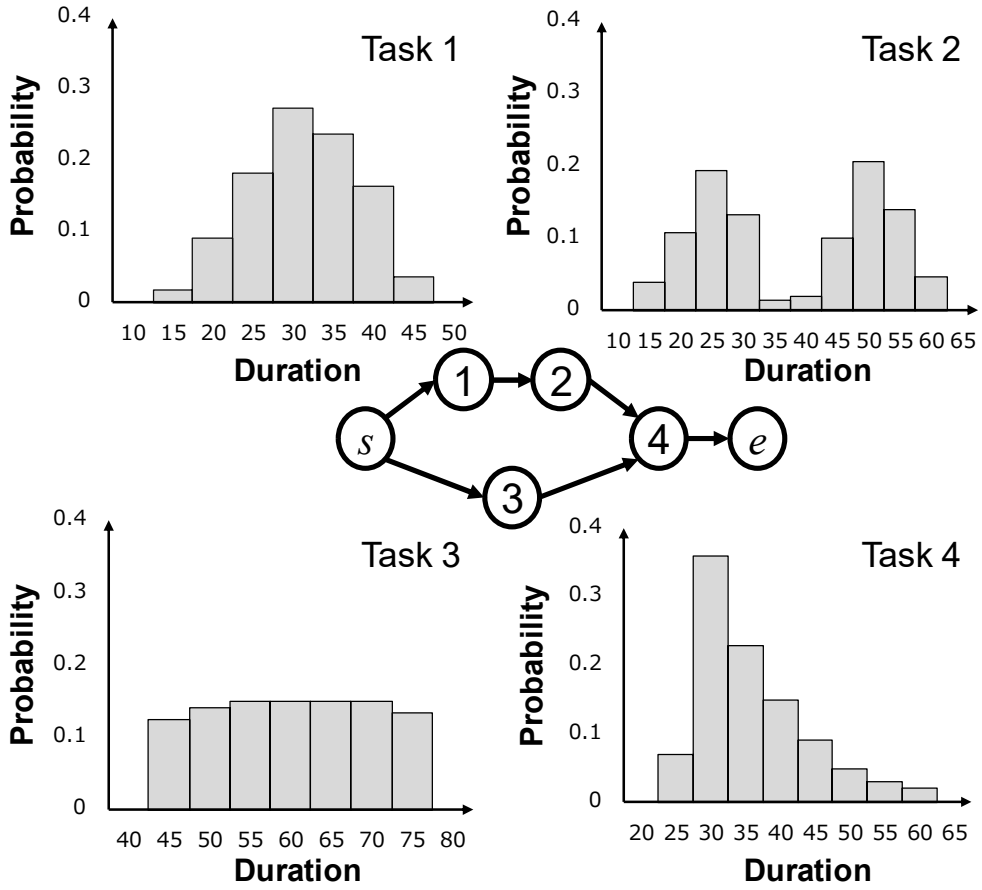

Figure 7. Production process of Example 1 [1-column fitting image]

Table 1 Historical operation data and given parameters for Example 1

Task i	$N_i^{(1)}$	$N_i^{(2)}$	$N_i^{(3)}$	$N_i^{(4)}$	$N_i^{(5)}$	$N_i^{(6)}$	$N_i^{(7)}$	$N_i^{(8)}$	$N_i^{(9)}$	$N_i^{(10)}$	$\sum_{k_i \in A} N_i^{(k_i)}$
1	2	10	20	30	26	18	4	-	-	-	110
2	6	16	29	20	2	3	15	31	21	7	150
3	15	17	18	18	18	18	16	-	-	-	120
4	7	36	23	15	9	5	3	2	-	-	100

Task i	$t_i^{(1)}$	$\lambda_{M,i}$	$c_{M,i}^{U}$	$\lambda_{D,i}$	$c_{D,i}^{U}$
1	15	0.24	40	0.020	30
2	15	0.15	125	0.015	40
3	45	0.20	125	0.010	40
4	25	0.090	100	0.020	30

Table 2 Results of Example 1

	CPM without uncertainty	TOF		
		w/o dispersion improvement	with dispersion improvement	
# of decision variables	9	3924	3928	
# of constraints	20	7849	7857	
Task i	$c_{M,i}$	$c_{M,i}$	$c_{M,i}$	$c_{D,i}$
1	40.00	40.00	40.00	10.00
2	61.80	80.00	1.040	40.00
3	48.20	30.00	25.27	3.685
4	100.0	100.0	100.0	30.00
Probability finished by Γ	0.8652	0.9070	0.9444	
Computational time	<1s	17s	1027s	

Table 2 shows the results for Example 1 by the classical approach, *CPM without uncertainty* and the proposed approach, *TOF*. In this example, we use the original formulation of the *CPM without uncertainty* shown in Eq. (A.1), where task duration $t_i, i \in W$ is fixed while in this example task durations are given as the historical operation data in Table 1. In this study, we use the average duration t_i^{ave} defined in Eq. (9) in place of the duration t_i as $t_i = t_i^{\text{ave}}, i \in W$. On the other hand, for the *TOF*, the proposed formulation in Section 4.2, we implemented two approaches in this example: *without* (*w/o*) *dispersion improvement* and *with dispersion improvement*. In the former approach, we do not consider improvement of dispersions $\left(c_{D,i} = 0, i \in W\right)$ to compare it against the conventional approach on the basis of the same degree of freedom. In the latter approach of *TOF*, we consider both improvements, expected values and dispersions.

The *CPM without uncertainty* leads to a low value of the objective function, 0.8652, or 86.52%, compared to the one calculated by the proposed methods, 0.9070. This value, 0.8652, was calculated by simply applying to the optimal cost allocations obtained by the *CPM without uncertainty* to the original problem that includes the uncertainty of task durations. This result indicates ignoring the problem uncertainty leads to poor cost allocation when task durations are uncertain.

Here note that the objective value of *TOF with dispersion improvement*, 0.9444, is even higher than that of (*TOF*) *without dispersion improvement*, 0.9070; this difference is the result of the higher degrees of freedom by the improvement in the dispersion of task duration histograms.

The advantages in the objective values discussed above are obtained at a cost of significantly longer computational time. In this example, the *CPM without uncertainty* needed only a short computational time (< 1s) because the problem size is very small. In contrast, *TOF without dispersion improvement* needed a significantly longer computational time, 17 s, and that for *TOF with dispersion improvement* is even two orders of magnitudes larger, 1027 s, because of the complexity of algorithm. From this result, we see that considering two improvement approaches for task durations, dispersion in addition to expected value, makes the problem much more difficult to solve. Note that in Supplementary Material E, we discuss why the computational time for the *TOF with dispersion improvement* is much larger than that of *TOF without dispersion improvement*.

We also analyze the optimal solution of *TOF without uncertainty*, and note that the allocation cost $c_{D,2}$, which is the improved dispersion of Task 2, is the highest among all allocation costs $c_{D,i}, i \in W$. This is because the dispersion of the task duration histogram in Task 2 is significantly larger than that of other tasks (see Figure 7), and thus improving this wide profile of task duration is effective.

7.2 Example 2

To further observe the influence of the problem size of the three proposed methods, *TOF*, (*POF, Strict*) and (*POF, Local Search*), we apply these methods to another example that has a larger number of tasks and paths. The production process of Example 2 is shown in Figure 8, and historical operation data and parameters are given in Table 3. In this problem, we set the maximum total cost C and the process completion time Γ as follows: $C = 350$ and $\Gamma = 180$.

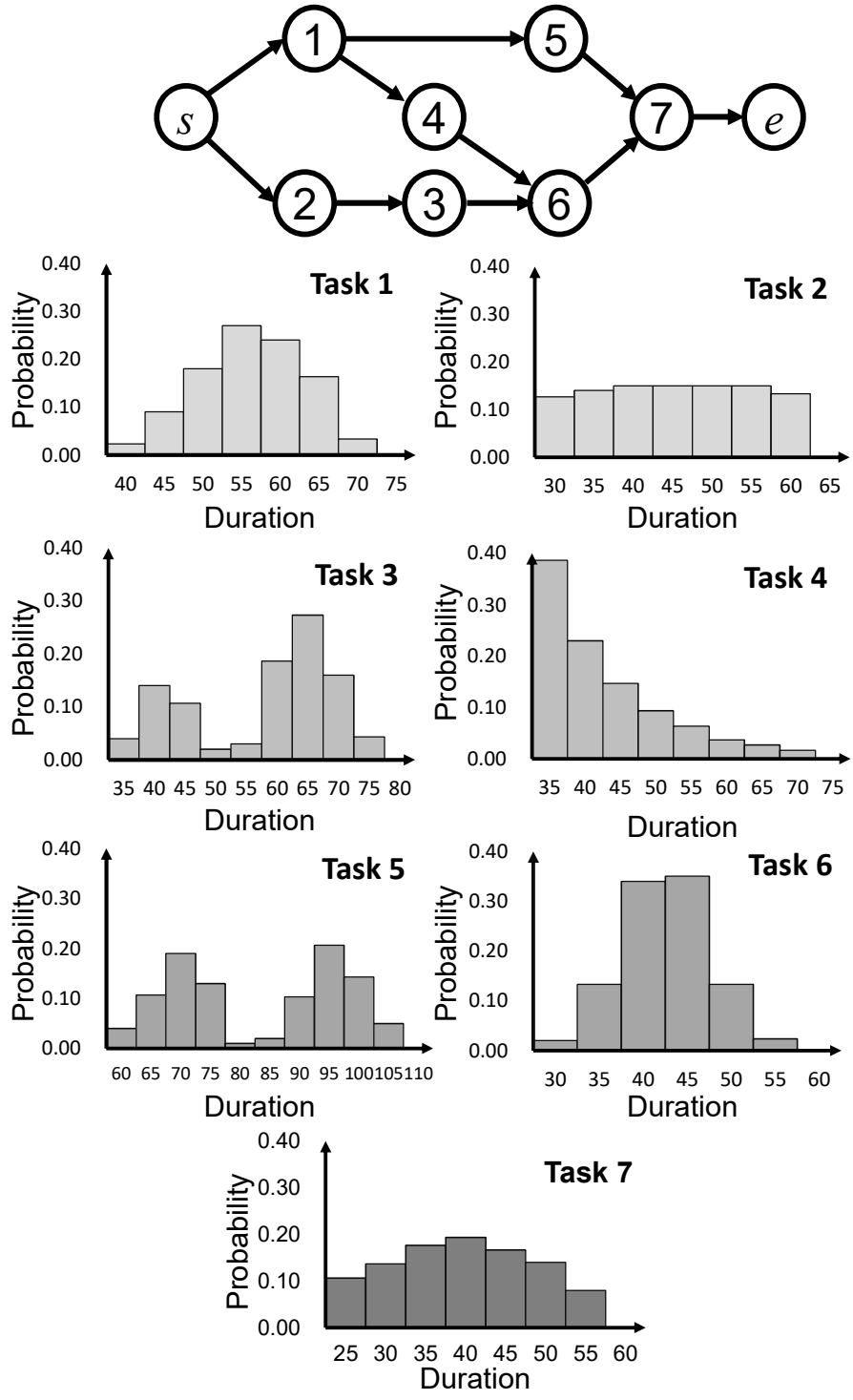

Figure 8. Production process of Example 2 [1-column fitting image]

Table 3 Historical operation data and parameters for Example 2

Task i	$N_i^{(1)}$	$N_i^{(2)}$	$N_i^{(3)}$	$N_i^{(4)}$	$N_i^{(5)}$	$N_i^{(6)}$	$N_i^{(7)}$	$N_i^{(8)}$	$N_i^{(9)}$	$N_i^{(10)}$	$\sum_{k_i \in A} N_i^{(k_i)}$
1	7	27	54	81	72	49	10	-	-	-	300
2	38	42	45	45	45	45	40	-	-	-	300
3	12	42	32	6	9	56	82	48	13	-	300
4	116	69	44	28	19	11	8	5	-	-	300
5	12	32	57	39	3	6	31	62	43	15	300
6	6	40	102	105	40	7	-	-	-	-	300
7	32	41	53	58	50	42	24	-	-	-	300

Task i	$t_i^{(1)}$	$\lambda_{M,i}$	$c_{M,i}^{U}$
1	40	0.15	75
2	30	0.15	125
3	35	0.20	100
4	35	0.25	150
5	60	0.15	175
6	30	0.15	75
7	25	0.10	50

Table 4 Results of Example 2

	TOF w/o dispersion improvement	(POF, Strict)	(POF, Local Search) Search range $\beta = 2$
# of decision variables	2469614	14307	132
# of constraints	7408815	42914	410
Task i	$c_{M,i}$	$c_{M,i}$	$c_{M,i}$
1	-	75.00	75.00
2	-	0.000	0.000
3	-	75.74	100.0
4	-	20.59	15.00
5	-	67.65	76.67
6	-	75.00	33.33
7	-	36.03	50.00
Probability finished by Γ	-	0.8635	0.8635
Calculation time for $h^{(v_1,v_2,\cdots,v_r)}$	-	95s	5s (Sum over two iterations)
Total computational time	>24h	134s	<6s (Sum over two iterations)

Table 4 shows the solutions and computational statistics for Example 2. Note that *TOF without dispersion improvement* in this table is the same method introduced in Example 1; (*POF, Strict*) in this table is the *Path-Oriented Formulation* shown in Section 5; (*POF, Local Search*) is the *Path-Oriented Formulation with Local Search Algorithm* shown in Section 6. It should be noted that (*POF, Strict*) and (*POF, Local Search*) can only consider *improvement of expected value*, and thus we compare *TOF without dispersion improvement*, not with dispersion improvement. In addition, we note that the complex calculation for $h^{(v_1,v_2,\cdots,v_r)}$ that appears in (30) and (D.3), which must be performed before solving the optimization problems for (*POF, Strict*) and (*POF, Local Search*), requires a significant amount of computational time; the computational time for this parameter is shown as "Calculation time for $h^{(v_1,v_2,\cdots,v_r)}$," in this table. Orders of computational time for calculating $h^{(v_1,v_2,\cdots,v_r)}$ in (*POF, Strict*) and (*POF, Local Search*) are discussed in Supplementary Material F.

We compare the problem sizes and computational times in these three methods. It can be seen that *TOF* has the largest number of decision variables, 2469614, and constraints, 7408815. In contrast, (*POF, Strict*) has a significantly smaller number of decision variables,14307, and constraints, 42914. Furthermore, (*POF, Local Search*) has an even smaller number of decision variables and constraints than other two methods. The numbers of decision variables,132, and constraints 410, are for the first iteration, out of the total of the two iterations. Due to the large number of variables and constraints, *TOF* cannot be solved in 24 hours. In contrast, (*POF, Strict*) and (*POF, Local Search*) can be solved much faster than *TOF*: 134 seconds and below 6 second, respectively. Note that the computational time in (*POF, Local Search*) contains all steps of calculations shown in Supplementary Material D.

It can be seen in Table 4 that while the objective values in (*POF, Strict*) and (*POF, Local Search*) are the same, 0.8635, the optimal cost allocations found by these two methods are significantly different. This non-uniqueness of the optimal solution is due to Eq. (32), where many different combinations of the allocating cost $c_i, i \in W$ exist that give a single value of $T^{(v_{j\text{crit}})}$.

We can also find a general rule for cost allocations about tasks that are in series without any branching or merging: in this example, Task 2 and Task 3. In the optimal solution, the allocating cost on Task 2 is zero $\left(c_{M,2} = 0\right)$ in both (*POF, Strict*) and (*POF, Local Search*), while for Task 3, which is the subsequent task to Task 2, a large amount of cost is allocated. This is because the cost coefficient for Task 3, $\lambda_{M,3} = 0.20$, is higher than that for Task 2, $\lambda_{M,2} = 0.15$. Since improvement of either task has the same influence on the process completion time, improving Task 3 should be pursued, which has the greater benefit for a given cost than Task 2.

Finally we note the above problems are for illustrative purposes, and real problems in chemical industries can be significantly larger. For such large problems, one of the proposed approaches, (*POF, Local Search*), would be a promising technique. Further investigations into larger example problems remain as future work.

8. Conclusion

In this paper, we advanced the classical CPM and proposed an optimization approach that maximizes the process completion probability within a target completion time, and utilizes historical data from production systems to handle uncertain task durations. Our method has mainly three advantages; handling the operation data without approximation; considering time-cost trade off by two kinds of improvement of task duration; and finding the optimal solution by formulating the problem as MILP. We proposed two formulations; *Task-Oriented Formulation* (*TOF*), and *Path-Oriented Formulation* (*POF*). Furthermore, we proposed the *Path-oriented Formulation with Local Search*, which applies a local search algorithm to *POF* and shortens the computational time. In addition, we applied these three formulations to two examples and demonstrated effectiveness of our approach.

Finally, we note further room for improvement in the proposed approaches. We modeled the improvement of task durations using two parameters, expected value and dispersion, which should be validated carefully with some realistic data. Furthermore, in the *POF*, we needed to ignore the improvement in dispersion. In addition, the influence of the problem size to the computational time should be investigated. These remaining issues should be resolved in future work, where we will extend the concept of the *improvement of dispersion*, which is considered only in *TOF*, to *POF*, as well as to the local search method.

Acknowledgement

This work was supported by JSPS KAKENHI Grant Number 16K06844.

References

Azaron, A., Perkgoz, C., Sakawa, M., 2005. A genetic algorithm approach for the time-cost trade-off in PERT networks. Appl. Math. Comput. 168, 1317–1339. https://doi.org/10.1016/j.amc.2004.10.021

Bruni, M.E., Guerriero, F., Pinto, E., 2009. Evaluating project completion time in project networks with discrete random activity durations. Comput. Oper. Res. 36, 2716–2722. https://doi.org/10.1016/j.cor.2008.11.021

Chen, S.-P., Hsueh, Y.-J., 2008. A simple approach to fuzzy critical path analysis in project networks. Appl. Math. Model. 32, 1289–1297. https://doi.org/10.1016/j.apm.2007.04.009

Golenko-Ginzburg, D., Gonik, A., 1997. Stochastic network project scheduling with non-consumable limited resources. Int. J. Prod. Econ. 48, 29–37. https://doi.org/10.1016/S0925-5273(96)00019-9

Hajdu, M., Bokor, O., 2014. The Effects of Different Activity Distributions on Project Duration in PERT Networks. Procedia - Soc. Behav. Sci. 119, 766–775. https://doi.org/10.1016/j.sbspro.2014.03.086

Hasuike, T., 2013. . https://doi.org/https://repository.kulib.kyoto-u.ac.jp/dspace/handle/2433/194808, (This paper is in Japanese, from Repository of Kyoto University)

Herroelen, W., Leus, R., 2004. The construction of stable project baseline schedules. Eur. J. Oper. Res. 156,

550–565. https://doi.org/10.1016/S0377-2217(03)00130-9

Huang, W., Ding, L., 2011. Project-scheduling problem with random time-dependent activity duration times. IEEE Trans. Eng. Manag. 58, 377–387. https://doi.org/10.1109/TEM.2010.2063707

Kamburowski, J., 1992. Bounding the distribution of project duration in PERT networks. Oper. Res. Lett. 12, 17–22. https://doi.org/10.1016/0167-6377(92)90017-W

Kaur, P., Kumar, A., 2014. Linear programming approach for solving fuzzy critical path problems with fuzzy parameters. Appl. Soft Comput. J. 21, 309–319. https://doi.org/10.1016/j.asoc.2014.03.017

Ke, H., Liu, B., 2005. Project scheduling problem with stochastic activity duration times. Appl. Math. Comput. 168, 342–353. https://doi.org/10.1016/j.amc.2004.09.002

Kelley, J.E., Walker, M.R., 1959. Critical-path planning and scheduling. Pap. Present. December 1-3, 1959, East. Jt. IRE-AIEE-ACM Comput. Conf. - IRE-AIEE-ACM '59 160–173. https://doi.org/10.1145/1460299.1460318

Kopanos, G.M., Kyriakidis, T.S., Georgiadis, M.C., 2014. New continuous-time and discrete-time mathematical formulation for resource-constrained project scheduling problems. Comput. Chem. Eng. 68, 96–106. https://doi.org/10.1016/j.compchemeng.2014.05.009

Kyriakidis, T.S., Kopanos, G.M., Georgiadis, M.C., 2012. MILP formulations for single- and multi-mode resource-constrained project scheduling problems. Comput. Chem. Eng. 36, 369–385. https://doi.org/10.1016/j.compchemeng.2011.06.007

Li, H., Womer, N.K., 2015. Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. Eur. J. Oper. Res. 246, 20–33. https://doi.org/10.1016/j.ejor.2015.04.015

Malcolm, D.G., Roseboom, J.H., Clark, C.E., Fazar, W., 1959. Application of a technique for research and development program evaluation. Oper. Res. 7, 646–669.

Sadjadi, S.J., Pourmoayed, R., Aryanezhad, M.B., 2012. A robust critical path in an environment with hybrid uncertainty. Appl. Soft Comput. J. 12, 1087–1100. https://doi.org/10.1016/j.asoc.2011.11.015

Tao, S., Wu, C., Sheng, Z., Wang, X., 2017. Stochastic Project Scheduling with Hierarchical Alternatives. Appl. Math. Model. 0, 1–22. https://doi.org/10.1016/j.apm.2017.09.015

Walton, H., 1964. Experience of the Application of the Critical Path Method to Plant Construction. J. Oper. Res. Soc. 15, 9–16. https://doi.org/10.1057/jors.1964.3

Xu, J., Zheng, H., Zeng, Z., Wu, S., Shen, M., 2012. Discrete time-cost-environment trade-off problem for large-scale construction systems with multiple modes under fuzzy uncertainty and its application to Jinping-II Hydroelectric Project. Int. J. Proj. Manag. 30, 950–966. https://doi.org/10.1016/j.ijproman.2012.01.019

Supplementary Material A : The general formulation of the classical CPM

Here we show another formulation of the classical CPM that is more commonly used than (1), which is given as the following Linear Programming (LP) problem.

$$
\begin{aligned}
&\text{Minimize}: y_e \\
&\text{s.t.} \quad \sum_{i \in W} c_i \le C \\
&\qquad 0 \le c_i \le c_i^U, \quad i \in W \\
&\qquad y_i + \left(t_i - \lambda_i c_i\right) \le y_{adj}, \quad \left(i, adj\right) \in E \\
&\qquad y_s = 0 \\
&\qquad c_s = 0 \\
&\qquad y_i \ge 0, \quad i \in W \cup \{e\}
\end{aligned}
\tag{A.1}
$$

where *adj* is the index of a task next to task *i*; *E* is a set of arcs in process network. These sets and parameters above are assumed to be known. On the other hand, decision variables are c_i that is allocating cost of task $i \in W$, and y_i that is the starting time of task $i \in W$. We define y_e as the starting time of sink node *e*, which is equivalent to the process completion time of the production system.

Supplementary Material B : Evaluation of parameters in (30) with (10)

We show that the parameter $h^{\left(v_1',v_2',\cdots,v_r'\right)}$ defined in (30) can be expressed with the probability (10), which is also given as a parameter. From the definition of $T^{(v_j)}$ in (25), $h^{\left(v_1',v_2',\cdots,v_r'\right)}$ defined in (30) is the probability that the following condition is satisfied:

$$T^{(v_j)} = \sum_{i \in p_j} t_i^{(k_i)} \le T^{\left(v_j'\right)},\ j \in V. \tag{B.1}$$

Here we define a set $H^{\left(v_1',v_2',\cdots,v_r'\right)}$, which includes all $\left(k_1,k_2,\cdots,k_n\right)$ that satisfy the condition (B.1) as follows:

$$\begin{aligned} H^{\left(v_1',v_2',\cdots,v_r'\right)} &= \left\{ \left(k_1,k_2,\cdots,k_n\right) \in A^n \mid \sum_{i \in p_j} t_i^{(k_i)} \le T^{\left(v_j'\right)}, j \in V \right\}, \\ &\left(v_1',v_2',\cdots,v_r'\right) \in M^r. \end{aligned} \tag{B.2}$$

Using the set defined in (B.2), $h^{\left(v_1',v_2',\cdots,v_r'\right)}$ can be rewritten from (30) as

$$h^{\left(v_1',v_2',\cdots,v_r'\right)} = \sum_{\left(k_1,k_2,\cdots,k_n\right) \in H^{\left(v_1',v_2',\cdots,v_r'\right)}} \Pr\left[t_i^{(k_i)} = t_i^{\left(k_i'\right)}, i \in W \right]. \tag{B.3}$$

Finally, by substituting (10) into (B.3), we obtain

$$h^{\left(v_1',v_2',\cdots,v_r'\right)} = \sum_{\left(k_1,k_2,\cdots,k_n\right) \in H^{\left(v_1',v_2',\cdots,v_r'\right)}} \alpha_1^{(k_1)} \alpha_2^{(k_2)} \cdots \alpha_n^{(k_n)}. \tag{B.4}$$

Supplementary Material C : Proof that Eq. (41) is satisfied at the optimal solution of (*POF*)

We show that the logic condition (41) is satisfied at the optimal solution of (*POF*). Firstly, we show a condition that the binary variables $z^{(v_1, v_2, \cdots, v_r)}$ satisfies. Here note that from the (30), which is the definition of $h^{(v_1, v_2, \cdots, v_r)}$, it is obvious that the $h^{(v_1, v_2, \cdots, v_r)}$ increases monotonically as the index *j* of random variables $v_j, j \in V$ increases:

$$h^{(v_1, v_2, \cdots, v_j, \cdots, v_r)} \leq h^{(v_1, v_2, \cdots, v_j+1, \cdots, v_r)}, j \in V. \tag{C.1}$$

Therefore, to maximize the objective function (35) under the constraint (36), the following condition must be satisfied:

$$z^{(v_1, v_2, \cdots, v_j, \cdots, v_r)} \leq z^{(v_1, v_2, \cdots, v_j+1, \cdots, v_r)}, j \in V. \tag{C.2}$$

Secondly, from Eq. (32), we have:

$$T^{(v_{j\,\mathrm{eff}})} - \sum_{i \in p_j} \lambda_{M,i} c_{M,i} \begin{cases} > \Gamma \text{ if } v_j > v_{j\,\mathrm{crit}} \\ = \Gamma \text{ if } v_j = v_{j\,\mathrm{crit}} \\ < \Gamma \text{ if } v_j < v_{j\,\mathrm{crit}} \end{cases}, j \in V. \tag{C.3}$$

Finally, from Eq. (C.2), Eq. (C.3) and the constraint (36), the following condition is satisfied under the constraints (37):

$$z^{(v_1', v_2', \cdots, v_r')} = \begin{cases} 0 \text{ if } v_j > v_{j\,\mathrm{crit}} \\ 1 \text{ if } v_j = v_{j\,\mathrm{crit}} \\ 0 \text{ if } v_j < v_{j\,\mathrm{crit}} \end{cases}, j \in V. \tag{C.4}$$

It can be seen that Eq. (C.4) is equivalent with (41). From the discussions above, it is proved that Eq. (41) is satisfied at the optimal solution of (*POF*).

Supplementary Material D. Algorithm of the local search method

The local search method discussed above can be implemented as an algorithm described below. Additionally, a flow chart of this algorithm is given in Figure D.1. Note that if the search range (45) contains the indices for the random variables $\left(v_1^*, v_2^*, \cdots, v_r^*\right)$, we can find the optimal solution to *POF*. In Steps 1 and 2, an initial guess to set the search range (45) is obtained. Step 3 is to prepare parameters to be used in Step 4, which is the step to solve *POF* where the search range is narrowed down. In Step 5, the optimality is checked by comparing the value of the objective function.

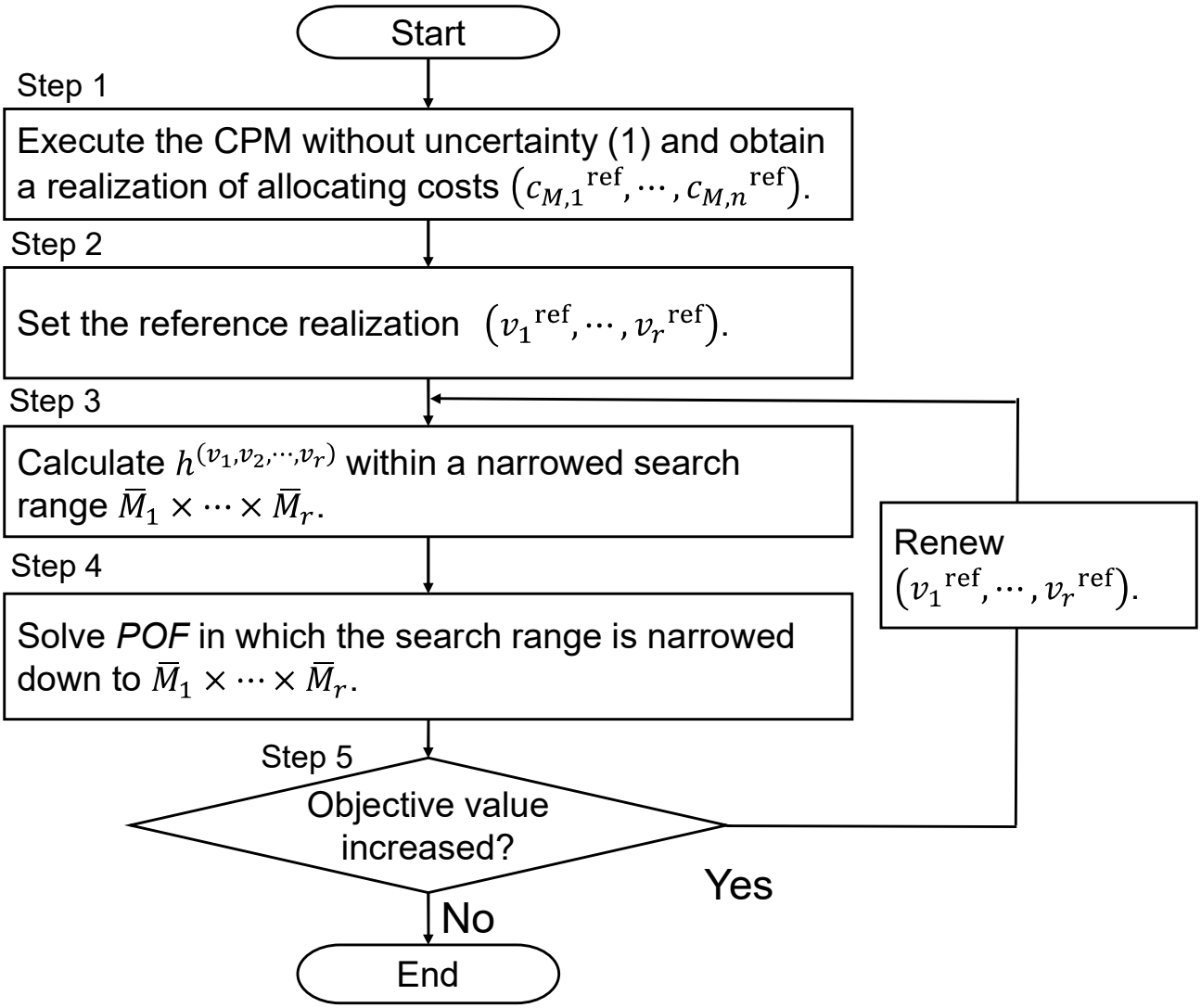

Figure D.1. Flow chart of the algorithm of local search method [1-column fitting image]

[Step 1]

In this step, we find one realization of allocating costs $\left(c_{M,1}{}^{\text{ref}}, c_{M,2}{}^{\text{ref}}, \cdots, c_{M,n}{}^{\text{ref}}\right)$ as a reference solution to find an initial guess to set the search range. Here we attempt to use the solution of the *CPM without uncertainty* shown in (A.1) as the realization of allocating costs $\left(c_{M,1}{}^{\text{ref}}, c_{M,2}{}^{\text{ref}}, \cdots, c_{M,n}{}^{\text{ref}}\right)$. In this study, we assume that the fixed task duration t_i in (A.1) is calculated by $t_i = t_i^{\text{ave}}$.

[Step 2]

In this step, we decide the reference realization of the indices as $\left(v_1{}^{\text{ref}}, v_2{}^{\text{ref}}, \cdots, v_r{}^{\text{ref}}\right)$ for $\left(T_1^{\left(v_1^{\text{ref}}\right)}, T_2^{\left(v_2^{\text{ref}}\right)}, \cdots, T_r^{\left(v_r^{\text{ref}}\right)}\right)$. Using this reference realization, the search range can be determined as shown in (44). To find the optimal solution, the candidates for search (45) must contain $\left(v_1{}^*, v_2{}^*, \cdots, v_r{}^*\right)$. Therefore, the realization $\left(v_1{}^{\text{ref}}, v_2{}^{\text{ref}}, \cdots, v_r{}^{\text{ref}}\right)$, which is at the center of the search range, needs to be sufficiently close to $\left(v_1{}^*, v_2{}^*, \cdots, v_r{}^*\right)$. However, $\left(v_1{}^*, v_2{}^*, \cdots, v_r{}^*\right)$ cannot be found without executing *POF* and find optimal cost allocation $\left(c_{M,1}{}^*, c_{M,2}{}^*, \cdots, c_{M,n}{}^*\right)$. Here, we attempt to use $\left(c_{M,1}{}^{\text{ref}}, c_{M,2}{}^{\text{ref}}, \cdots, c_{M,n}{}^{\text{ref}}\right)$ and set $\left(v_1{}^{\text{ref}}, v_2{}^{\text{ref}}, \cdots, v_r{}^{\text{ref}}\right)$ to satisfy a similar condition to (32) as

$$v_j{}^{\text{ref}} = \arg\min_{v_j}\left(T^{(v_j)} - \sum_{i \in p_j} \lambda_{M,i} c_{M,j}{}^{\text{ref}} = \Gamma \right). \tag{D.1}$$

[Step 3]

In this step, the problem *POF* is prepared by calculating the probability $h^{\left(v_1', v_2', \cdots, v_r'\right)}$ defined in (30) where the search range is narrowed down. Firstly we construct the local search range around the reference indices $\left(v_1{}^{\text{ref}}, v_2{}^{\text{ref}}, \cdots, v_r{}^{\text{ref}}\right)$ as given in (44), where the candidates of discrete random variables is given by (45). Similarly with (B.2), here we define a set of $\left(k_1, k_2, \cdots, k_n\right)$ that satisfies the condition (B.1) for the indices (45) where the search range is narrowed down:

$$\begin{aligned} \bar{H}^{\left(v_1', v_2', \cdots, v_r'\right)} &= \left\{ \left(k_1, k_2, \cdots, k_n\right) \in A^n \mid \sum_{i \in p_j} t_i{}^{(k_i)} \le T^{\left(v_j'\right)}, j \in V \right\}, \\ \left(v_1', v_2', \cdots, v_r'\right) &\in \bar{M}_1 \times \bar{M}_2 \times \cdots \times \bar{M}_r. \end{aligned} \tag{D.2}$$

Finally, we calculate the probability $h^{\left(v_1', v_2', \cdots, v_r'\right)}$ by using (B.4) where the search range is narrowed down:

$$h^{\left(v_1', v_2', \cdots, v_r'\right)} = \sum_{\left(k_1, k_2, \cdots, k_n\right) \in \bar{H}^{\left(v_1', v_2', \cdots, v_r'\right)}} \alpha_1{}^{(k_1)} \alpha_2{}^{(k_2)} \cdots \alpha_n{}^{(k_n)}. \tag{D.3}$$

[Step 4]

We solve the following problem, which is a modification to *POF*:

$$
\begin{aligned}
&\text{Maximize:} \sum_{(v_1,v_2,\cdots,v_r)\in \bar{M}_1\times\bar{M}_2\times\cdots\times\bar{M}_r} h^{(v_1,v_2,\cdots,v_r)}\cdot z^{(v_1,v_2,\cdots,v_r)} \\
&\text{s.t. } z^{(v_1,v_2,\cdots,v_r)} \in \{0,1\}, (v_1,v_2,\cdots,v_r) \in \bar{M}_1\times\bar{M}_2\times\cdots\times\bar{M}_r \\
&T_j^{(v_j)}\cdot z^{(v_1,v_2,\cdots,v_r)} - \sum_{i\in p_j}\lambda_{M,i}c_{M,i} \le \Gamma,\ j\in V \\
&\sum_{(v_1,v_2,\cdots,v_r)\in \bar{M}_1\times\bar{M}_2\times\cdots\times\bar{M}_r} z^{(v_1,v_2,\cdots,v_r)} = 1 \\
&\sum_{i\in W} c_{M,i} \le C \\
&0 \le c_{M,i} \le c_{M,i}^U, \quad i\in W
\end{aligned}
\tag{D.4}
$$

where in contrast to *POF*, the range of possible index of random variables (31) is replaced by the narrowed range (45). Note that the optimal solution of (D.4) may be different from the optimal solution of *POF*; only if the search range (45) contains $(v_1^*, v_2^*, \cdots, v_r^*)$ we obtain $h^{(v_{1\text{crit}}, v_{2\text{crit}}, \cdots, v_{r\text{crit}})}$ as the objective value. Even if (D.4) cannot find the optimal solution of *POF*, by executing (D.4) we can find equivalent or better candidate where the objective value is higher compared to the reference value $h^{(v_1^{\text{ref}}, v_2^{\text{ref}}, \cdots, v_r^{\text{ref}})}$.

[Step 5]

This step is to decide whether the search in Step 3 is sufficient by comparing the objective function. If the binary variables $z^{(v_1,v_2,\cdots,v_r)}$ is the same as those in the previous execution of Step 4, we terminate the algorithm. If the solution is changed, return to Step 3 after replacing the reference indices $(v_1^{\text{ref}}, v_2^{\text{ref}}, \cdots, v_r^{\text{ref}})$ by that in the solution of Step 4.

Supplementary Material E. Consideration about computational time in Example 1

Here we discuss why the computational time for the *Task-Oriented Formulation* (*TOF*) *with dispersion improvement* is much larger than that of *TOF w/o dispersion improvement* in Example 1. To consider this issue, we fix the cost coefficient for the dispersion, $c_{D,i}$ for some tasks. The following constraint is implemented:

$$c_{D,i} = 0,\ i \in W,\ i \neq i' \tag{E.1}$$

where i' is the index for the task where improvement of dispersion is allowed. We change i' one by one, and compare the optimal solution and computational time for each i'.

In Table 5, we show results of the *TOF* where i' is varied from 1 to 4 along with the results that already appeared in Table 2 (*without dispersion improvement* and *with dispersion improvement*). To analyze the complexity of the problem, the numbers of partial problems (linear programming problems with relaxation) in the solution algorithm for the mixed integer programming problem, branch-and-bound method, are also shown there. In this algorithm, linear programming problems where the integer variables are relaxed or fixed are sequentially solved, and unattractive combinations of the integer variables where the objective value exceeds the upper bound are eliminated as the algorithm proceeds.

It can be seen in Table 5 that the number of partial problems increases significantly when the dispersion improvement is allowd, and as a result the computational time increases; the computational time without dispersion improvement is only 17 seconds, but that for allowing dispersion improvement in only one task ($i' = 4$) increases it to 321 seconds. This is because nearly equally attractive options cannot be eliminated early in the branch-and-bound search, and thus a larger number of partial problems must be solved. Comparing the results for $i' = 1, 2, 3,$ and 4 in Table 5, the number of partial problems and computational time for $i' = 2, 4$ are much larger than those for $i' = 1, 3$. From these results, it is estimated that in cases $i' = 2, 4$, improving dispersion of the task durations is nealy as attractive as the improvement of the expected value of task durations.

Table 5 Results of the TOF under different conditions

	w/o dispersion improvement	$i'=1$		$i'=2$		$i'=3$		$i'=4$		with dispersion improvement	
# of decision variables	3924	3925		3925		3925		3925		3928	
# of partial problems	8481	12323		139284		6837		155635		241547	
Computational time (s)	17	31		257		16		321		1027	
Probability finished by Γ	0.9070	0.9070		0.9072		0.9070		0.9138		0.9444	
Task i	$c_{M,i}$	$c_{M,i}$	$c_{D,i}$	$c_{M,i}$	$c_{D,i}$	$c_{M,i}$	$c_{D,i}$	$c_{M,i}$	$c_{D,i}$	$c_{M,i}$	$c_{D,i}$
1	40	40	1.262	40	-	40	-	40	-	40	10
2	80	78.73	-	54.76	25	76	-	62.67	-	62.67	40
3	30	30	-	30	-	30.83	3.173	20	-	20	3.685
4	100	100	-	100	-	100	-	100	25	100	30

Supplementary Material F. Calculation order of the parameter $h^{(v_1,v_2,\cdots,v_r)}$

Since the amount of calculation for the parameters $h^{(v_1,v_2,\cdots,v_r)}$ is significant, here we note the calculation orders of $h^{(v_1,v_2,\cdots,v_r)}$ defined in (30) used in the *Path-Oriented Formulation* (*POF, Strict*) in Section 5 and *Path-Oriented Formulation with Local Search Algorithm* (*POF, Local Search*) in Section 6.

Firstly, we show the calculation order of $h^{(v_1,v_2,\cdots,v_r)}$ in (*POF, Strict*) is $O\left(l^r \times m^n\right)$. From Eq. (26) and (31), the number of $h^{(v_1,v_2,\cdots,v_r)}$ is l^r. Furthermore, from (B.3), the each one of the $h^{(v_1,v_2,\cdots,v_r)}$ is sum of the probabilities $\alpha_1^{\left(k_1'\right)}\alpha_2^{\left(k_2'\right)}\cdots\alpha_n^{\left(k_n'\right)}$ in (10) where the total number of combinations for $\alpha_1^{\left(k_1'\right)}\alpha_2^{\left(k_2'\right)}\cdots\alpha_n^{\left(k_n'\right)}$ is m^n as shown in (5) and (11).

Similarly, the calculation order of $h^{(v_1,v_2,\cdots,v_r)}$ in (*POF, Local Search*) is $O\left(\left(2\beta+1\right)^r \times m^n\right)$, since the range of $\left(v_1,v_2,\cdots,v_r\right)$ is reduced from l to $\left(2\beta+1\right)$ as in (44).