American Journal of Operations Research, 2016, 6, 480-488
http://www.scirp.org/journal/ajor
ISSN Online: 2160-8849
ISSN Print: 2160-8830

Resolution of Resource Contentions in the CCPM-MPL Using Simulated Annealing and Genetic Algorithm

Hajime Yokoyama, Hiroyuki Goto

Department of Industrial & System Engineering, Hosei University, Tokyo, Japan
Email: hajime.yokoyama.3m@stu.hosei.ac.jp, goto-h@hosei.ac.jp

How to cite this paper: Yokoyama, H. and Goto, H. (2016) Resolution of Resource Contentions in the CCPM-MPL Using Simulated Annealing and Genetic Algorithm. *American Journal of Operations Research*, **6**, 480-488.
http://dx.doi.org/10.4236/ajor.2016.66044

Received: September 20, 2016
Accepted: November 18, 2016
Published: November 21, 2016

Abstract

This research aims to plan a "good-enough" schedule with leveling of resource contentions. We use the existing critical chain project management-max-plus linear framework. Critical chain project management is known as a technique used to both shorten the makespan and observe the due date under limited resources; the max-plus linear representation is an approach for modeling discrete event systems as production systems and project scheduling. If a contention arises within a single resource, we must resolve it by appending precedence relations. Thus, the resolution framework is reduced to a combinatorial optimization. If we aim to obtain the exact optimal solution, the maximum computation time is longer than 10 hours for 20 jobs. We thus experiment with Simulated Annealing (SA) and Genetic Algorithm (GA) to obtain an approximate solution within a practical time. Comparing the two methods, the former was beneficial in computation time, whereas the latter was better in terms of the performance of the solution. If the number of tasks is 50, the solution using SA is better than that using GA.

Keywords

Critical Chain Project Management, Max-Plus Algebra, CCPM-MPL, Simulated Annealing, Genetic Algorithm

1. Introduction

This research aims to plan a "good-enough" schedule with leveling of resource contentions. References [1] and [2] modified and extended a scheduling methodology referred to as critical chain project management-max-plus linear (CCPM-MPL), in which CCPM [3] [4] was applied to the max-plus algebra [5]. CCPM is a technique for project

management invented by E. M. Goldratt and developed by L. P. Leach. The objective of this method is to both shorten the makespan and observe the due date under a limited number of resources. The max-plus algebra is an algebraic system wherein the max and plus operations are defined as addition and multiplication, respectively. MPL [1] representation is an approach for modeling and analyzing a class of discrete-event systems such as production and project scheduling, in which the behavior of the target system is represented by linear equations in the max-plus algebra. MPL representation can formulate systems with structures of non-concurrency, synchronization, parallel processing of multiple tasks, and so on [1]. We assume that a single resource cannot process multiple tasks simultaneously. If a contention arises within a single resource, we must resolve it by appending precedence relations. Thus, the resolution framework is reduced to a combinatorial problem. To level resource contentions, we developed a complete enumeration method by which an exact solution is obtained. In a numerical simulation, the maximum computation time was longer than 10 hours for 20 jobs. Reference [6] obtained a "good-enough" schedule by using Genetic Algorithm (GA) within a short computation time. On the other hand, we uncovered a preliminary design to obtain a "good-enough" schedule by using Simulated Annealing (SA) [7]. However, it is not currently clear which of the two methods is better. Hence, this research compares the two methods in terms of the performances of their solutions, computation times, and values.

2. CCPM-MPL Framework

After defining the max-plus algebra, we define the CCPM-MPL framework with the help of [1] and [2].

2.1. Max-Plus Algebra

We define a set $\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$, where $\mathbb{R}$ is the whole real line. Then, for $x, y \in \mathbb{R}_{\max}$, we define the operators:

$$x \oplus y = \max(x, y), \tag{1}$$

$$x \otimes y = x + y. \tag{2}$$

The priority of operator $\otimes$ is higher than that of $\oplus$. If $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}_{\max}^{m \times n}$ and $\boldsymbol{Z} \in \mathbb{R}_{\max}^{n \times q}$,

$$[\boldsymbol{X} \oplus \boldsymbol{Y}]_{ij} = [\boldsymbol{X}]_{ij} \oplus [\boldsymbol{Y}]_{ij}, \tag{3}$$

$$[\boldsymbol{X} \otimes \boldsymbol{Z}]_{ij} = \max_{k=1}^{n} [\boldsymbol{X}]_{ik} \otimes [\boldsymbol{Y}]_{kj}. \tag{4}$$

The zero and unit elements for operators $\oplus$ and $\otimes$ are denoted by $\varepsilon(=-\infty)$ and $e(=0)$, respectively. $\boldsymbol{\varepsilon}$ is a matrix whose elements are all ε, and $\boldsymbol{e}$ is a matrix whose diagonal elements are e and whose off-diagonal elements are ε. If $\boldsymbol{X} \in \mathbb{R}_{\max}^{n \times n}$, the following operator $\boldsymbol{X}^*$ is the Kleene star:

$$\boldsymbol{X}^* = \boldsymbol{e} \oplus \boldsymbol{X} \oplus \boldsymbol{X}^{\otimes 2} \oplus \cdots \oplus \boldsymbol{X}^{\otimes (s-1)}, \tag{5}$$

where $s\left(1 \le s \le n\right)$ is an instance that satisfies $\boldsymbol{X}^{\otimes(s-1)} \neq \boldsymbol{\varepsilon}$ and $\boldsymbol{X}^{\otimes s} = \boldsymbol{\varepsilon}$. In addition, we define

$$x \setminus y = -x + y. \tag{6}$$

2.2. Formulation of the CCPM-MPL Framework

We define the following relevant matrices and vectors:

- n: number of tasks;
- p : number of external outputs;
- q : number of external inputs;
- $\boldsymbol{B}_0 \in \mathbb{R}_{\max}^{n\times q}$: input matrix, $\left[\boldsymbol{B}_0\right]_{ij}$ = {e: if task i has an input transition j, ε: otherwise};
- $\boldsymbol{C}_0 \in \mathbb{R}_{\max}^{p\times n}$: output matrix, $\left[\boldsymbol{C}_0\right]_{ij}$ = {e: if task j has an output transition i, ε: otherwise};
- $\boldsymbol{F}_0 \in \mathbb{R}_{\max}^{n\times n}$: adjacency matrix, $\left[\boldsymbol{F}_0\right]_{ij}$ = {e: if task i has a preceding task j, ε: otherwise};
- $\boldsymbol{d} \in \mathbb{R}_{\max}^{n}$: system parameter, $\left[\boldsymbol{d}\right]_i$: duration time in task i;
- $\boldsymbol{u} \in \mathbb{R}_{\max}^{q}$: input vector, $\left[\boldsymbol{u}\right]_i$: input time to external input i;
- $\boldsymbol{y} \in \mathbb{R}_{\max}^{p}$: output vector, $\left[\boldsymbol{y}\right]_i$: output time from external output i;
- $\boldsymbol{x} \in \mathbb{R}_{\max}^{n}$: state vector, $\left[\boldsymbol{x}\right]_i$: start or completion time of task i.

The earliest task-completion times of all tasks, $\boldsymbol{x}_{\mathrm{E}}$, are calculated using

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A}_0 \otimes \boldsymbol{B}_0 \otimes \boldsymbol{u}, \tag{7}$$

where

$$\boldsymbol{A}_0 = \left(\boldsymbol{P}_0 \otimes \boldsymbol{F}_0\right)^{*} \otimes \boldsymbol{P}_0, \tag{8}$$

$$\boldsymbol{P}_0 = \mathrm{diag}\left(\boldsymbol{d}\right). \tag{9}$$

Matrixes $\boldsymbol{A}_0$ and $\boldsymbol{P}_0$ are the transition and weight matrices, respectively. The earliest output times to all output transitions, $\boldsymbol{y}_{\mathrm{E}}$, are then calculated by

$$\boldsymbol{y}_{\mathrm{E}} = \boldsymbol{C}_0 \otimes \boldsymbol{x}_{\mathrm{E}}. \tag{10}$$

Then, the latest task-starting times, $\boldsymbol{x}_{\mathrm{L}}$, are calculated using Equation (10):

$$\boldsymbol{x}_{\mathrm{L}} = \left(\boldsymbol{C}_0 \otimes \boldsymbol{A}_0\right) \setminus \boldsymbol{y}_{\mathrm{E}}. \tag{11}$$

The latest input times, $\boldsymbol{u}_{\mathrm{L}}$, are calculated in terms of $\boldsymbol{x}_{\mathrm{L}}$:

$$\boldsymbol{u}_{\mathrm{L}} = \boldsymbol{B}_0 \setminus \boldsymbol{x}_{\mathrm{L}}. \tag{12}$$

As a consequence, the total floats of all tasks can be calculated using Equations ((5) and (8)):

$$\boldsymbol{m}_0 = \left(\boldsymbol{x}_{\mathrm{L}} + \boldsymbol{d}\right) - \boldsymbol{x}_{\mathrm{E}}. \tag{13}$$

All tasks can be classified into two types: $\left[\boldsymbol{m}_0\right]_i = 0$ and $\left[\boldsymbol{m}_0\right]_i > 0$ are a critical and a non-critical task, respectively. We define two vectors, $\boldsymbol{a}_0, \boldsymbol{b}_0 \in \mathbb{R}_{\max}^{n}$, to classify each task as either critical or non-critical:

$$[\boldsymbol{a}_0]_i = \{e : \text{if } [\boldsymbol{m}_0]_i = 0,\ \varepsilon : \text{if } [\boldsymbol{m}_0]_i > 0\}, \tag{14}$$

$$[\boldsymbol{b}_0]_i = \{e : \text{if } [m_0]_i > 0,\ \varepsilon : \text{if } [\boldsymbol{m}_0]_i = 0\}. \tag{15}$$

We then calculate two matrices, $\boldsymbol{P}_\mathrm{a}, \boldsymbol{P}_\mathrm{b} \in \mathbb{R}_{\max}^{n\times n}$, as follows:

$$\boldsymbol{P}_\mathrm{a} = \mathrm{diag}(\boldsymbol{a}_0) \otimes P_0, \tag{16}$$

$$\boldsymbol{P}_\mathrm{b} = \mathrm{diag}(\boldsymbol{b}_0) \otimes P_0. \tag{17}$$

In the CCPM framework, there are two types of buffers, referred to as feeding and project buffers. The size of each buffer is calculated by

$$\boldsymbol{r}_\mathrm{feed} = \left[\boldsymbol{P}_\mathrm{b} \otimes (\boldsymbol{F}_0 \otimes \boldsymbol{P}_\mathrm{b})^* \otimes \boldsymbol{g}_0\right] \big/ 2, \tag{18}$$

$$\boldsymbol{r}_\mathrm{proj} = \left[\boldsymbol{P}_\mathrm{a} \otimes (\boldsymbol{F}_0 \otimes \boldsymbol{P}_\mathrm{a})^* \otimes \boldsymbol{g}_0\right] \big/ 2, \tag{19}$$

where

$$\boldsymbol{g}_0 = [e \ \ e \ \cdots \ e]^\mathrm{T} \in \mathbb{R}_{\max}^{n}. \tag{20}$$

Feeding buffers are inserted wherever a non-critical task joins into a critical one. A project buffer is inserted on the eve of an output to avoid tardiness of the project. To reflect the insertion of the two types of buffers, we incur weights to the adjacency matrix $\boldsymbol{F}_0$ and the output matrix $\boldsymbol{C}_0$:

$$\boldsymbol{F}_\mathrm{bufs} = \boldsymbol{F}_0 \oplus \mathrm{diag}(\boldsymbol{a}_0) \otimes \boldsymbol{F}_0 \otimes \mathrm{diag}(\boldsymbol{r}_\mathrm{feed}), \tag{21}$$

$$\boldsymbol{C}_\mathrm{bufs} = \boldsymbol{C}_0 \otimes \left[\mathrm{diag}(\boldsymbol{r}_\mathrm{proj}) \oplus \mathrm{diag}(\boldsymbol{r}_\mathrm{feed})\right]. \tag{22}$$

Now we can calculate the earliest task-completion, $\boldsymbol{x}'_\mathrm{E}$, and the output time, $\boldsymbol{y}'_\mathrm{E}$, after inserting the time buffers. The earliest task-completion times of all nodes, $\boldsymbol{x}'_\mathrm{E}$, are calculated using

$$\boldsymbol{x}'_\mathrm{E} = \boldsymbol{A}'_0 \otimes \boldsymbol{B}_0 \otimes \boldsymbol{u}, \tag{23}$$

$$\boldsymbol{A}'_0 = \boldsymbol{P}_0 \otimes (\boldsymbol{F}_\mathrm{bufs} \otimes \boldsymbol{P}_0)^*. \tag{24}$$

Matrix $\boldsymbol{A}'_0$ is the transition matrix, in which the insertion of the two types of buffers is reflected. The earliest output times to all output transitions, $\boldsymbol{y}'_\mathrm{E}$, are then calculated by

$$\boldsymbol{y}'_\mathrm{E} = \boldsymbol{C}_\mathrm{bufs} \otimes \boldsymbol{x}'_\mathrm{E}. \tag{25}$$

We treat $\boldsymbol{y}'_\mathrm{E}$ as the objective function and consider minimizing $[\boldsymbol{y}'_\mathrm{E}]_i$.

3. Resolution of Resource Contentions

We explain the resolution of resource contentions with the help of [8]. We add relevant matrices and vectors as follows:

- l: number of resources,
- s: maximum number of tasks that a single resource processes,
- $\boldsymbol{S} \in \mathbb{N}^{l\times s}$: processing order of tasks,

$$[\boldsymbol{S}]_{ij} = \{k : \text{resource } i \text{ processes task } k \text{ in the } j\text{th processing}, 0 : \text{otherwise}\}.$$

We consider Figure 1 as an example project with five tasks before the leveling of resource contentions. The duration time of each task is $\boldsymbol{d} = [3 \quad 4 \quad 5 \quad 2 \quad 1]^{\mathrm{T}}$. The adjacency matrix, $\boldsymbol{F}_0$, is

$$\boldsymbol{F}_0 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & e & \varepsilon \end{bmatrix}. \tag{26}$$

Tasks 1, 4, and 5 are processed by resource 1, whereas tasks 2 and 3 are processed by resource 2. If resource 2 processes tasks 2 and 3 simultaneously, a resource contention occurs. We avoid contentions between tasks 2 and 3 by appending two precedence relations, which are expressed by broken arrows in Figure 2, which shows the project after the resource contention is leveled. We then reflect the leveling of the resource contentions using the following adjacency matrix, $\boldsymbol{F}_0'$:

$$\boldsymbol{F}_0' = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & e & \varepsilon \end{bmatrix}. \tag{27}$$

In addition, the processing order of tasks, $\boldsymbol{S}$, is

$$\boldsymbol{S} = \begin{bmatrix} 1 & 4 & 5 \\ 3 & 2 & 0 \end{bmatrix}. \tag{28}$$

Our numerical simulation to level resource contentions shows that the maximum

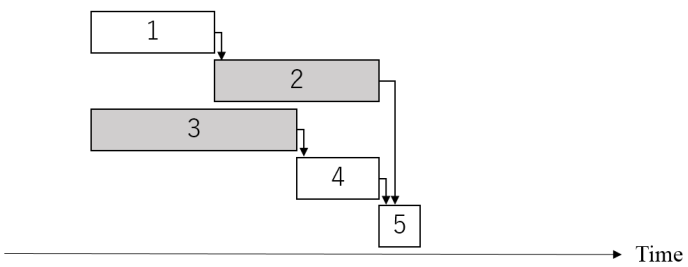

Figure 1. Example of a fifth-task project before a resource contention is leveled.

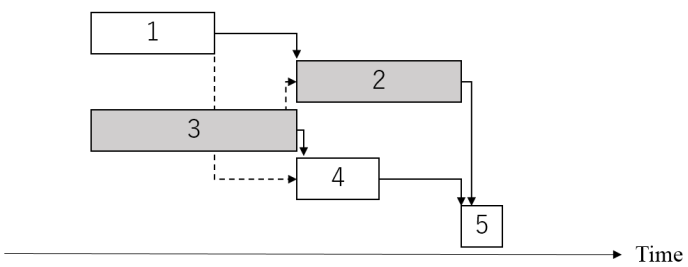

Figure 2. Example of the fifth-task project after the resource contention is leveled.

computation time was longer than 10 hours for 20 jobs. We thus consider the development of approximate methods within a short computation time.

4. Two Metaheuristics

We experiment with an optimization based on Genetic Algorithm (GA) and Simulated Annealing (SA), both of which are common metaheuristics.

4.1. Genetic Algorithm

We experiment with an optimization based on the GA developed in [6], which is used to obtain an approximate solution.

Algorithm 1: Genetic Algorithm

STEP 1. Set mutation parameter and termination condition, m_{ut} and i, respectively.

STEP 2. Generate a random solution, $\boldsymbol{S}$, and create the earliest output times, $\left[\boldsymbol{y}'_{\mathrm{E1}}\right]_i$.

STEP 3. Focusing on a row vector of $\boldsymbol{S}$ that gives the list of tasks for a single resource, split the vector into two. We then swap the two vectors, followed by generating a new solution, $\boldsymbol{S}'$.

STEP 4. If a mutation occurs having probability m_{ut}, then proceed to STEP 5. Otherwise, *i.e.*, if a mutation does not occur, then follow STEP 6.

STEP 5. Swap two tasks of $\boldsymbol{S}'$ at the same resource randomly.

STEP 6. Create the new earliest output times, $\left[\boldsymbol{y}'_{\mathrm{E2}}\right]_i$ using $\boldsymbol{S}'$.

STEP 7. If $\left[\boldsymbol{y}'_{\mathrm{E2}}\right]_i < \left[\boldsymbol{y}'_{\mathrm{E1}}\right]_i$ follows, then set $\boldsymbol{S} \coloneqq \boldsymbol{S}'$ and $\left[\boldsymbol{y}'_{\mathrm{E1}}\right]_i \coloneqq \left[\boldsymbol{y}'_{\mathrm{E2}}\right]_i$.

STEP 8. Terminate if the termination condition is satisfied. If otherwise, return to STEP4.

4.2. Simulated Annealing

SA [9] is known as an approach to efficiently obtain an approximate solution. We use the 2-opt neighborhood method [10] based on a local search to generate a neighboring solution.

Algorithm 2: Simulated Annealing

STEP 1. Initialize the temperature and cooling parameters, T_0 and γ, respectively.

STEP 2. Generate a random solution, $\boldsymbol{S}$, and create the earliest output times, $\left[\boldsymbol{y}'_{\mathrm{E1}}\right]_i$.

STEP 3. Generate a neighboring solution, $\boldsymbol{S}'$, and create the new earliest output times, $\left[\boldsymbol{y}'_{\mathrm{E2}}\right]_i$.

STEP 4. Calculate $\Delta E = \left[\boldsymbol{y}'_{\mathrm{E2}}\right]_i - \left[\boldsymbol{y}'_{\mathrm{E1}}\right]_i$.

STEP 5. If $\Delta E < 0$ holds, then set $\boldsymbol{S} \coloneqq \boldsymbol{S}'$ and $\left[\boldsymbol{y}'_{\mathrm{E1}}\right]_i \coloneqq \left[\boldsymbol{y}'_{\mathrm{E2}}\right]_i$ with probably 1. Otherwise, *i.e.*, if $\Delta E \geq 0$ holds, then set $\boldsymbol{S} \coloneqq \boldsymbol{S}'$ and $\left[\boldsymbol{y}'_{\mathrm{E1}}\right]_i \coloneqq \left[\boldsymbol{y}'_{\mathrm{E2}}\right]_i$ with probably $\exp\left(-\Delta E/T\right)$.

STEP 6. Update the temperature parameter, $T_k = \gamma^k T_0 \left(0.8 \leq \gamma \leq 0.99\right)$. The value of γ is a cooling parameter which decreases the temperature parameter, T_0.

STEP 7. Repeat STEPS3-6.

STEP 8. Terminate if the temperatures are sufficiently low. If otherwise, return to STEP3.

5. Approximate Ratio and Computation Time

We obtain solutions using the SA- and GA-based algorithms. We use a personal computer with the following execution environment:

- machine: Dell Optiplex 9020;
- CPU: Intel® Core™ i7-4790 3.60 GHz;
- OS: Microsoft Windows 7 Professional;
- memory: 4.0 GB;
- programming language: MATLAB R2015b.

The test cases for the numerical experiment are generated under the following conditions:

- number of cases: 100;
- number of resources, l: for 10, 15, and 20 tasks, we set the number of resources to 3, 5, and 7, respectively;
- number of resources for task, i: uniformly random integer numbers;
- duration time, $[\boldsymbol{d}]_i$: uniformly random integer numbers.

We obtain approximate solutions using two metaheuristics. The temperature and cooling parameter are set to $T=1$ and $\gamma=0.85$, respectively, for the SA-based algorithm. The mutation parameter and the end condition are set to $m_{ut}=0.2$ and $i=200$, respectively, for the GA-based algorithm. We compare the exact and approximate solutions. The performance of the solution is shown in Table 1 for the two metaheuristics. We can confirm that the performance of the solution using the GA-based algorithm is smaller than that using the SA-based algorithm. The average computation times are shown in Figure 3. We can confirm that the computation using the SA-based algorithm is faster than that using the GA-based algorithm. The computation time of the GA-based algorithm is more than16-fold longer than that of the SA-based algorithm. The average values of the solutions are shown in Table 2. If the number of tasks is 10 or 20, the values of the solutions using the two methods are not remarkably different; if the number of tasks is 30 or 40, the value of the solution using the GA-based algorithm is smaller than that of the SA-based algorithm. However, we should note

Table 1. Performance of the solutions.

Method	Number of tasks		
	10	15	20
Simulated Annealing	1.001	1.002	1.006
Genetic Algorithm	1.000	1.001	1.000

Table 2. Average values of the solutions.

Method	Number of tasks				
	10	20	30	40	50
Simulated Annealing	155.335	260.48	365.485	467.925	563.795
Genetic Algorithm	155.215	259.055	361.755	465.45	568.955

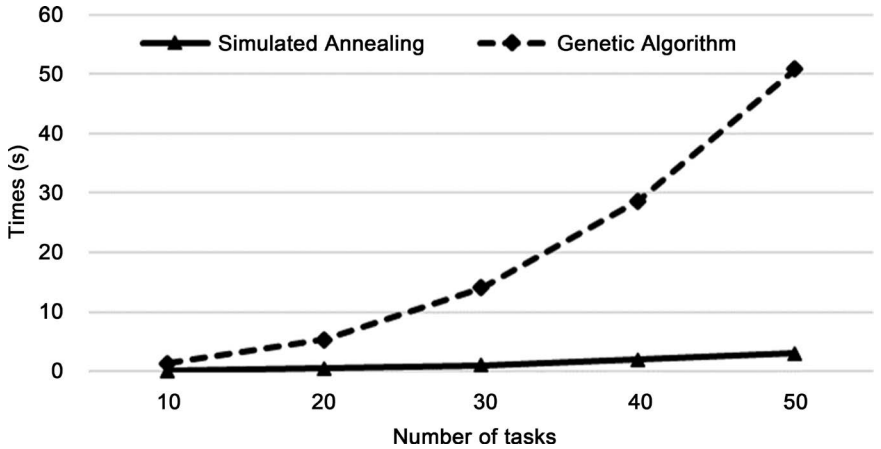

Figure 3. Average computation times in seconds.

here that the solution using the SA-based algorithm is better than that using the GA if the number of tasks is 50.

6. Conclusions

This research has studied the planning of a "good-enough" schedule with leveling of resource contentions. We utilized an existing framework called the CCPM-MPL. The resolution framework was reduced to a combinatorial problem. We used SA- and GA-based algorithms to obtain approximate solutions within short time. Moreover, we used the 2-opt neighborhood to generate a neighboring solution in the SA-based algorithm. Comparing the two methods, the SA-based algorithm was beneficial in terms of computation time, whereas the latter was better in terms of the performance of the solution. In addition, when the number of tasks was 50, the value of the solution using the SA-based algorithm was better than that using the GA-based algorithm.

Developing an original heuristic to level resource contentions within a short computation time is our future work.

References

[1] Goto, H. (2017) Forward-Compatible Framework with Critical-Chain Project Management using a Max-Plus Linear Representation. *OPSEARCH*, **54**, 16 p..
http://dx.doi.org/10.1007/s12597-016-0276-3

[2] Goto, H., Truc, N.T.N. and Takahashi, H. (2013) Simple Representation of the Critical Chain Project Management Framework in a Max-Plus Linear Form. *SICE Journal of Control, Measurement, and System Integration*, **6**, 341-344.
http://dx.doi.org/10.9746/jcmsi.6.341

[3] Goldratt, E.M. (1997) Critical Chain. North River Press, Great Barrington.

[4] Leach, L.P. (2005) Critical Chain Project Management. 2nd Edition, Artech House, Boston.

[5] Heidergott, B., Olsder, G.J. and Woude, L. (2006) Max Plus at Work: Modeling and Analysis of Synchronized Systems. Princeton University Press, New Jersey.

[6] Koga, H., Goto, H. and Chiba, E. (2014) Resolution of Resource Conflicts in the CCPM Framework: Utilization of a Local Search Method or Genetic Algorithm, **50**, 7-12. (In Japa-

nese)

[7] Yokoyama, H. and Goto, H. (2016) Resolution of Resource Contentions in the Critical Chain Project Management Based on Simulated Annealing. *Proceedings of the 6th International Conference on Industrial Engineering and Operations Management*, Kuala Lumpur, 8 March 2016, 29.

[8] Koga, H., Goto, H. and Chiba, E. (2014) Resolution of Resource Conflicts in the CCPM Framework Using a Local Search Method. *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*, Bandar Sunway, 10 December 2014, 94-98. http://dx.doi.org/10.1109/ieem.2014.7058607

[9] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) Optimization by Simulated Annealing. *Science*, **220**, 671-680. http://dx.doi.org/10.1126/science.220.4598.671

[10] Croes, G.A. (1958) A Method for Solving Traveling-Salesman Problems. *Operation Research*, **6**, 791-812. http://dx.doi.org/10.1287/opre.6.6.791