2016 年度修士論文

Max-plus 線形表現に基づいた離散事象システムの解析と制御

ANALYSIS AND CONTROL OF DISCRETE EVENT SYSTEMS
BASED ON MAX-PLUS LINEAR REPRESENTATIONS

法政大学大学院理工学研究科

システム理工学専攻経営システム系修士課程

15R6210

氏名　横山朔

指導教員　五島洋行

Abstract

In this research, we study analysis and control of discrete event systems based on Max-plus linear representations. Max-plus linear representation is known as an approach to model and analyze a class of discrete event systems as production systems and project scheduling, in which the behavior of a target system is represented by linear equations in max-plus algebra. Max-plus algebra is an algebraic system wherein the max and plus operations are defined as addition and multiplication, respectively.

This research aims to plan a “good-enough” schedule with resource contentions leveled. We base on an existing framework called Critical Chain Project Management-Max-Plus-Linear (CCPM-MPL). The Critical Chain Project Management is known as a technique to both shorten the makespan and observe the due date under limited number of resources. 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 twenty jobs. We thus experiment a Simulated Annealing (SA) and Genetic Algorithm (GA) to obtain an approximate solution within a practical time. Compared with the two methods, the former was beneficial in computation time, while the latter was better in terms of performance of the solution. If the number of tasks is ninety, the solution using the SA is better than that of the GA.

This research constructs a solution method for project scheduling using a max-plus-linear (MPL) representation. Several types of MPL equations can be reduced to a constraint satisfaction problem (CSP) for mixed integer programming (MIP). The resulting formulation is flexible and easy-to-use for project scheduling; for example, the earliest output times, latest task-starting times, and latest input times can be computed using a general-purpose MIP solver. We also develop a key method to identify critical tasks within a CSP-based framework. The developed methods are validated through numerical examples.

目次

1　はじめに

本論文では, Max-plus 線形表現に基づいた離散事象システムの解析と制御について研究する．Max-plus 線形表現とは，生産システムやプロジェクトスケジューリングなどの離散事象システムを定式化し分析するための手法として知られており，システムの挙動は Max-plus 代数における線形方程式によって表現される．Max-plus 代数[1][2]とは，基本演算子である max 演算と+演算子をそれぞれ加算と乗算として定義された代数系である．Max-plus 線形表現は非競合，同期や複数事象の並行動作などの構造をしたシステムを定式化することが可能である．本論文は，製造業などの生産や開発におけるプロジェクトの日程計画において，Critical Chain Project Management-Max-Plus Linear[3]と呼ばれる日程計画の方法論を用いて効率的に日程を計画する．また，Max-plus 線形方程式によって表現されるプロジェクトの日程計画において新たな解法を構築する．

近年, 製造業などの生産や開発の現場において, 人や設備などの資源が限られている状況下で, 複数のプロジェクトの遂行や製品の生産を行う際に，完成までの所要時間（工期）を短縮し，かつ納期遅れを防止するような最適な日程を計画する方法が研究されている．なぜなら，作業や工程などのタスクの実行時間の不確実性が大きく，かつ少数の作業者あるいは設備でのやりくりが必要など，中小規模の組織において数カ月程度の期間で生産業務を行うからである．本研究では，クリティカルチェーンプロジェクトマネジメント(CCPM)に Max-plus 線形表現を適用した，Critical Chain Project Management-Max-Plus Linear(CCPM-MPL)[3]と呼ばれる日程計画の方法論を用いる．CCPM は，Goldratt[4]によって発明され，Leach[5]によって開発された，プロジェクトマネジメントのための手法の一つであり，目的は資源数が限られた状況下において最大完了時間を短縮し，工期の遅れを防止することである．プロジェクトの日程を計画する際，一つの資源が同時に複数工程を処理することはできないため，一つの資源において競合が発生するとき，先行関係を追加することによって競合を解消しなければならない．それゆえ，これは組合せ最適化問題と言われる問題の一種で，プロジェクトにおいて工程数が多いとき，スケジュールを作成するのに何ヶ月あるいは何年も計算時間を要する．資源競合を解消するために，全数列強法を用いて数値実験を行った結果，工程数が 20 のとき最大計算時間が 10 時間を越えた．この問題に対して先行研究である古賀[6]は，遺伝的アルゴリズム(GA)に基づいたアルゴリズムを提案し，短い計算時間で効率的に日程を計画している．しかし，他の近似解法との比較を行っておらず，GA に基づいたアルゴリズムが最も有効な手法なのかは定かではない．そこで本研究では，同じメタヒューリスティクスの一つである焼きなまし法に基づいたアルゴリズムを提案し，近似比による解の性能，評価値，計算時間の点において，古賀[6]が提案するアルゴリズムとの比較を行う．

Max-plus 線形方程式によって表現されるプロジェクトの日程計画において新たな解法を構築する．先行研究である Goto[7]は，Max-plus 代数における二種類の線形方程式: $\boldsymbol{A} \otimes \boldsymbol{x} = \boldsymbol{b}$および $\boldsymbol{x} = \boldsymbol{A} \otimes \boldsymbol{x} \oplus \boldsymbol{b}$の解法を構築した．これらの方程式は Max-plus 線形表現と呼ばれ，混合整数計画問題(MIP)における制約充足問題(CSPs)として解かれている．日程計画において，線形方程式$\boldsymbol{x} =$

$\boldsymbol{A} \otimes \boldsymbol{x} \oplus \boldsymbol{b}$は最早開始時刻を，$\boldsymbol{A} \otimes \boldsymbol{x} = \boldsymbol{b}$は最遅開始時刻を得るために用いられる．行列$\boldsymbol{A}$およびベクトル$\boldsymbol{b}$の成分が定数のとき，最適解は定数ベクトルとなり，解法は既に存在する[8]．一方，行列$\boldsymbol{A}$もしくはベクトル$\boldsymbol{b}$の成分に変数を含むとき，最適解は関数になる．例えば，行列$\boldsymbol{A}$が実行時間に相当するシステムパラメータを含むときである[9]．行列$\boldsymbol{A}$もしくはベクトル$\boldsymbol{b}$に他の制約が組み込まれると，既存の解法を用いることはできない．それゆえに，Goto[7]は二つの Max-plus 線形方程式を緩和し混合整数計画問題における制約充足問題として定式化することによって，これらの方程式の解法を構築した．しかし，文献[7]は日程計画のための重要な計算のための解法は定式化されていない．そこで本研究では，Max-plus 線形表現によって表された日程計画における最早完了時刻，最遅開始時刻および最遅入力時刻の計算のための解法を構築する．さらに，制約充足問題の下でクリティカルな工程を判定するための方法を提案するが，これは本研究の要となる技法である．

2　Max-plus 線形表現

本章では，文献[10]を基に Max-plus 代数[1][2]の基本演算子を述べた後，簡単な生産システムを例に Max-plus 線形システムの状態空間表現について記述する．また，離散事象システムの挙動を Max-plus 代数における線形方程式によって表現する．

2.1 定義

Max-plus 代数とは, 実数全体を$\mathbb{R}$としたとき$\mathbb{R}_{\max} = \mathbb{R} \cup \{-\infty\}$と定義する. $x, y \in \mathbb{R}_{\max}$のとき，加法と乗法に関する二つの基本演算子$\oplus$，$\otimes$を以下のとおり定義する．

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

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

演算子$\otimes$の優先順位は，演算子$\oplus$よりも高い．加算および乗算の単位元は，それぞれ$\varepsilon(= -\infty)$と$e(= 0)$であり，以下の関係が成り立つ．

$$x \oplus \varepsilon = \varepsilon \oplus x = x, \tag{3}$$

$$x \otimes e = e \otimes x = x, \tag{4}$$

$$x \otimes \varepsilon = \varepsilon \otimes x = \varepsilon. \tag{5}$$

通常の代数系と同様に結合法則，交換法則および分配法則の性質がある．$z \in \mathbb{R}_{\max}$を加え，以下に示す．

- 結合法則

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z, \tag{6}$$

$$x \otimes (y \otimes z) = (x \otimes y) \otimes z. \tag{7}$$

- 交換法則

$$x \oplus y = y \oplus x, \tag{8}$$

$$x \otimes y = y \otimes x. \tag{9}$$

- 分配法則

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z). \tag{10}$$

Max-plus 代数には，加法のべき等性という性質がある．

$$x \oplus x = x. \tag{11}$$

$n \in \mathbb{N}$としたとき，$x \in \mathbb{R}_{\max}$のべき乗は，

$$x^{\otimes n} = x \otimes x \otimes \cdots \otimes x = x + x + \cdots + x = n \times x. \tag{12}$$

ただし，$x^{\otimes 0} = e$である．$x_k \in \mathbb{R}_{\max}$のとき，二つの基本演算子を用いて，

$$\bigoplus_{k=1}^{n} x_k = x_1 \oplus x_2 \oplus \cdots \oplus x_n = \max(x_1, x_2, \cdots, x_n), \tag{13}$$

$$\bigotimes_{k=1}^{n} x_k = x_1 \otimes x_2 \otimes \cdots \otimes x_n = x_1 + x_2 + \cdots + x_n, \tag{14}$$

と記述することができる．行列に拡張し，二つの基本演算子を定義する．$\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}_{\max}^{m\times n}$であり$\boldsymbol{Z} \in \mathbb{R}_{\max}^{n\times q}$のとき，

$$[\boldsymbol{X} \oplus \boldsymbol{Y}]_{ij} = [\boldsymbol{X}]_{ij} \oplus [\boldsymbol{Y}]_{ij} = \max([\boldsymbol{X}]_{ij}, [\boldsymbol{Y}]_{ij}), \tag{15}$$

$$[\boldsymbol{X} \otimes \boldsymbol{Z}]_{ij} = \bigoplus_{k=1}^{l} ([\boldsymbol{X}]_{ik} \otimes [\boldsymbol{Z}]_{kj}) = \max_{k=1,\cdots,l}([\boldsymbol{X}]_{ik} + [\boldsymbol{Z}]_{kj}). \tag{16}$$

$\boldsymbol{\varepsilon}$はすべての成分がεである．一方$\boldsymbol{e}$は対角成分がeであり，非対角成分がεである対角行列である．行列においても通常の代数系と同様に結合法則，交換法則および分配法則の性質がある．$\boldsymbol{X} \in \mathbb{R}_{\max}^{n\times n}$であるとき，

$$\boldsymbol{X}^* = \boldsymbol{e} \oplus \boldsymbol{X} \oplus \boldsymbol{X}^{\otimes 2} \oplus \cdots \oplus \boldsymbol{X}^{\otimes(s-1)}, \tag{17}$$

と記述することができ，$\boldsymbol{X}^*$はクリーネ閉包と呼ばれている．ただし，$\boldsymbol{X}^{\otimes(s-1)} \neq \boldsymbol{\varepsilon}, \boldsymbol{X}^{\otimes s} = \boldsymbol{\varepsilon}$であり，$s\ (1 \leq s \leq n)$を満たす．

表記の簡素化のため，$\overline{\mathbb{R}}_{\max} = \mathbb{R} \cup \{-\infty\} \cup \{+\infty\}$と定義し，二つの演算子∧，\を追加する．

$$x \wedge y = \min(x, y), \tag{18}$$

$$x \backslash y = -x + y. \tag{19}$$

演算子\の優先順位は，演算子∧よりも高い．演算子∧の単位元は$\bar{\varepsilon}(= +\infty)$であり，演算子\の左単位元は$e(= 0)$である．

$$x \wedge \bar{\varepsilon} = \bar{\varepsilon} \wedge x = x, \tag{20}$$

$$x \backslash e = -x, e \backslash x = x. \tag{21}$$

また演算子⊗のために，次のように定義する．

$$\varepsilon \otimes \bar{\varepsilon} = \bar{\varepsilon} \otimes \varepsilon = \varepsilon. \tag{22}$$

演算子\のために，次の関係性を定義する．

$$\varepsilon \backslash \varepsilon = \bar{\varepsilon} \backslash \bar{\varepsilon} = \bar{\varepsilon}. \tag{23}$$

$x_k \in \overline{\mathbb{R}}_{\max}$のとき，

$$\bigwedge_{k=1}^{n} x_k = x_1 \wedge x_2 \wedge \cdots \wedge x_n = \min(x_1, x_2, \cdots, x_n), \tag{24}$$

と記述することができる．行列に拡張し，二つの演算子を定義する．$\boldsymbol{X}, \boldsymbol{Y} \in \overline{\mathbb{R}}_{\max}^{m\times n}$であり$\boldsymbol{Z} \in \overline{\mathbb{R}}_{\max}^{n\times q}$のとき，

$$[\boldsymbol{X} \wedge \boldsymbol{Y}]_{ij} = [\boldsymbol{X}]_{ij} \wedge [\boldsymbol{Y}]_{ij} = \min([\boldsymbol{X}]_{ij}, [\boldsymbol{Y}]_{ij}), \tag{25}$$

$$[\boldsymbol{X} \odot \boldsymbol{Z}]_{ij} = \bigwedge_{k=1}^{l} \left([\boldsymbol{X}]_{ik} \backslash [\boldsymbol{Z}]_{kj}\right) = \min_{k=1,\cdots l}\left(-[\boldsymbol{X}]_{ik} + [\boldsymbol{Z}]_{kj}\right), \tag{26}$$

ただし，

$$\boldsymbol{X}^{\mathrm{T}} \odot \boldsymbol{Z} = \boldsymbol{X} \backslash \boldsymbol{Z}. \tag{27}$$

2.2 Max-plus 線形システムによる状態空間表現

図 1 は，文献[10]に用いられている生産システムの例である．このシステムは，三つの工程，二つの外部入力と一つの外部出力で構成されている．工程 1 は外部入力 1 から原材料を受け取り，処理し，加工物を工程 3 へと運ぶ．同様に，工程 2 は外部入力 2 から原材料を受け取り，処理し，加工物を工程 3 へと運ぶ．工程 3 は，工程 1 と工程 2 それぞれで処理された材料を受け取り，処理し，製品を外部出力へと送る．この処理を繰り返し行うと仮定し，工程の最早開始時刻について検討する．工程 1 および工程 2 は，一つ前の工程での処理が完了し，必要な加工物が運ばれてきたと同時に処理を開始することができる．工程 3 は，一つ前の工程での処理が完了し，工程 1 および工程 2 から加工物を受け取ったと同時に処理を開始することができる．k番目の工程iにおいて最早処理開始時刻および実行時間を，それぞれ$x_i(k)$およびd_iとする．また，外部からの入力iにおける材料の投入する時刻および外部へ出力する時刻を，それぞれ$u_i(k)$および$y(k)$とする．このとき図 1 の生産システムにおいて，最早開始時刻および最早完了時刻を式(28)から式(31)で表すことができる．

$$x_1(k) = \max\{x_1(k-1) + d_1, u_1(k)\}, \tag{28}$$

$$x_2(k) = \max\{x_2(k-1) + d_2, u_2(k)\}, \tag{29}$$

$$x_3(k) = \max\{x_3(k-1) + d_3, x_1(k) + d_1, x_2(k) + d_2\}, \tag{30}$$

$$y(k) = x_3(k) + d_3. \tag{31}$$

上式からわかるように，方程式は max 演算子と+演算子のみで表現され，max 演算子を用いていることから，非線形方程式であることがわかる．そこで Max-plus 代数を適用することで，線形方程式で表現する．式(28)から式(31)に Max-plus 代数を適用し，式(32)から式(35)で表す．

$$x_1(k) = x_1(k-1) \otimes d_1 \oplus u_1(k), \tag{32}$$

$$x_2(k) = x_2(k-1) \otimes d_2 \oplus u_2(k), \tag{33}$$

$$x_3(k) = x_3(k-1) \otimes d_3 \oplus x_1(k) \otimes d_1 \oplus x_2(k) \otimes d_2, \tag{34}$$

$$y(k) = x_3(k) \otimes d_3. \tag{35}$$

式(32)および式(33)を式(34)に代入する．

$$\begin{aligned} x_3(k) = {} & x_1(k-1) \otimes d_1^2 \oplus x_2(k-1) \otimes d_2^2 \oplus x_3(k-1) \otimes d_3 \\ & \oplus u_1(k) \otimes d_1 \oplus u_2(k) \otimes d_2. \end{aligned} \tag{36}$$

上式からわかるように，max-plus 代数を適用することによって，線形方程式として表現することが可能である．式(32)から式(36)は，

$$\boldsymbol{x}(k) = \boldsymbol{A} \otimes \boldsymbol{x}(k-1) \oplus \boldsymbol{B} \otimes \boldsymbol{u}(k), \tag{37}$$

$$\boldsymbol{y}(k) = \boldsymbol{C} \otimes \boldsymbol{x}(k). \tag{38}$$

と記述することが出来る．ただし，

$$\boldsymbol{x}(k) = \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix}, \boldsymbol{u}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}, \boldsymbol{y}(k) = [y(k)], \boldsymbol{A} = \begin{bmatrix} d_1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ d_1^2 & d_2^2 & d_3 \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} e & \varepsilon \\ \varepsilon & e \\ d_1 & d_2 \end{bmatrix}, \boldsymbol{C} = \begin{bmatrix} \varepsilon \\ \varepsilon \\ d_3 \end{bmatrix}^{\mathrm{T}}.$$

式(37)および式(38)は，Max-plus 線形システムと呼ばれており，現代制御理論の状態方程式に類似している．

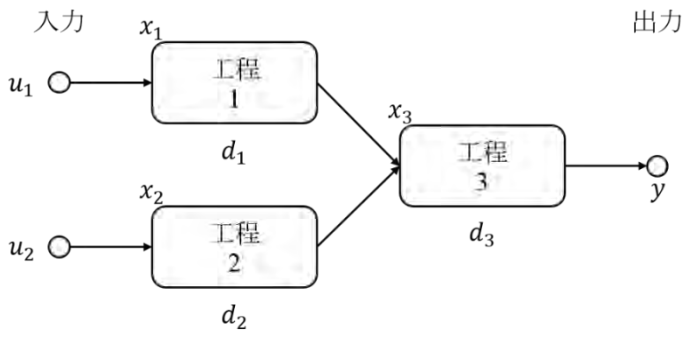

図 1. 2 入力，1 出力と 3 工程で構成される生産システムの例．(文献[10]より参照)

2.3 Max-Plus 線形システム表現

文献[3][10][11]を基に生産システムやプロジェクトスケジューリングなどの離散事象システムの挙動を，Max-plus 代数における線形方程式で表現する．以下に関連する行列とベクトルを定義する．

- n：工程数,
- p：出力数,
- q：入力数,
- $\boldsymbol{B} \in \mathbb{R}_{\max}^{n\times q}$：入力行列，$[\boldsymbol{B}]_{ij}$ ={e：工程iが外部入力jを持つとき，ε：それ以外のとき},
- $\boldsymbol{C} \in \mathbb{R}_{\max}^{p\times n}$：出力行列，$[\boldsymbol{C}]_{ij}$ ={e：工程jが外部出力iを持つとき，ε：それ以外のとき},
- $\boldsymbol{F} \in \mathbb{R}_{\max}^{n\times n}$：隣接行列，$[\boldsymbol{F}]_{ij}$ ={e：工程iが先行工程jを持つとき，ε：それ以外のとき},
- $\boldsymbol{d} \in \mathbb{R}_{\max}^{n}$：システムパラメータ，$[\boldsymbol{d}]_i$：工程$i$の実行時間,
- $\boldsymbol{u} \in \mathbb{R}_{\max}^{q}$：入力ベクトル，$[\boldsymbol{u}]_i$：工程$i$における外部からの入力時刻,
- $\boldsymbol{y} \in \mathbb{R}_{\max}^{p}$：出力ベクトル，$[\boldsymbol{y}]_i$：工程$i$における外部への出力時刻,
- $\boldsymbol{x} \in \mathbb{R}_{\max}^{n}$：状態ベクトル，$[\boldsymbol{x}]_i$：工程$i$における開始時刻もしくは終了時刻.

全ての工程における最早完了時刻$\boldsymbol{x}_{\mathrm{E}}$は,

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{P} \otimes (\boldsymbol{F} \otimes \boldsymbol{x}_{\mathrm{E}} \oplus \boldsymbol{B} \otimes \boldsymbol{u}). \tag{39}$$

ここで$\boldsymbol{A} = \boldsymbol{P} \otimes \boldsymbol{F}$，$\boldsymbol{b} = \boldsymbol{P} \otimes \boldsymbol{B} \otimes \boldsymbol{u}$とすると,

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A} \otimes \boldsymbol{x}_{\mathrm{E}} \oplus \boldsymbol{b}, \tag{40}$$

ただし,

$$\boldsymbol{P} = \mathrm{diag}(\boldsymbol{d}). \tag{41}$$

行列$\boldsymbol{A} \in \mathbb{R}_{\max}^{n\times n}$は重み付き遷移行列であり，ベクトル$\boldsymbol{b} \in \mathbb{R}_{\max}^{n}$は$\boldsymbol{b} = \boldsymbol{P} \otimes \boldsymbol{B} \otimes \boldsymbol{u}$を満たす重み付き入力ベクトルである．式(40)の左辺および右辺に$\boldsymbol{x}_{\mathrm{E}}$があるため，方程式を変形する．通常の代数系の場合，左辺第一項を左辺に移項して$(\boldsymbol{I} - \boldsymbol{A})^{-1}$を左から乗じることで$\boldsymbol{x}_{\mathrm{E}}$が求めることが可能だが，Max-plus 代数の場合は減算や逆行列が定義されていないため，右辺全体を代入することで最早完了時刻$\boldsymbol{x}_{\mathrm{E}}$を求める．

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A}^{\otimes 2} \otimes \boldsymbol{x}_{\mathrm{E}} \oplus (\boldsymbol{e} \oplus \boldsymbol{A}) \otimes \boldsymbol{b}. \tag{42}$$

同様に代入を繰り返すことによって,

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A}^{\otimes s} \otimes \boldsymbol{x}_{\mathrm{E}} \oplus \left(\boldsymbol{e} \oplus \boldsymbol{A} \oplus \boldsymbol{A}^{\otimes 2} \oplus \cdots \oplus \boldsymbol{A}^{\otimes s-1}\right) \otimes \boldsymbol{b}. \tag{43}$$

式(43)は非負重みをもつ有向非サイクルグラフとなる．$\boldsymbol{A} \in \mathbb{R}_{\max}^{n\times n}$のクリーネ閉包[12]が現れ，$s\ (1 \le s \le n)$において$\boldsymbol{A}^{\otimes(s-1)} \neq \boldsymbol{\varepsilon}$と$\boldsymbol{A}^{\otimes s} = \boldsymbol{\varepsilon}$を満たす．この結果から最早完了時刻$\boldsymbol{x}_{\mathrm{E}}$は，

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A} \otimes \boldsymbol{B} \otimes \boldsymbol{u}, \tag{44}$$

となる．ただし，

$$\boldsymbol{A} = (\boldsymbol{P} \otimes \boldsymbol{F})^{*} \otimes \boldsymbol{P}. \tag{45}$$

$[(\boldsymbol{P} \otimes \boldsymbol{F})^{*}]_{ij}$は，ノード間におけるノード$j$からノード$i$への最長経路を表す．また，2 工程間の可到達性を表している．全ての出力遷移における最早終了時刻$\boldsymbol{y}_{\mathrm{E}}$は，

$$\boldsymbol{y}_{\mathrm{E}} = \boldsymbol{C} \otimes \boldsymbol{x}_{\mathrm{E}}. \tag{46}$$

次に全ての工程における最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$は，

$$\boldsymbol{x}_{\mathrm{L}} = \boldsymbol{P}\backslash(\boldsymbol{F}\backslash\boldsymbol{x}_{\mathrm{L}} \wedge \boldsymbol{C}\backslash\boldsymbol{y}_{\mathrm{E}}). \tag{47}$$

ここで$\boldsymbol{A} = \boldsymbol{P}\backslash\boldsymbol{F}$，$\boldsymbol{c} = \boldsymbol{P}\backslash(\boldsymbol{C}\backslash\boldsymbol{y}_{\mathrm{E}})$とすると，

$$\boldsymbol{x}_{\mathrm{L}} = \boldsymbol{A}\backslash\boldsymbol{x}_{\mathrm{L}} \wedge \boldsymbol{c}. \tag{48}$$

ベクトル$\boldsymbol{c} \in \mathbb{R}_{\max}^{n}$は$\boldsymbol{c} = \boldsymbol{P}\backslash(\boldsymbol{C}\backslash\boldsymbol{y}_{\mathrm{E}})$を満たす重み付き出力ベクトルである．式(48)の左辺および右辺に$\boldsymbol{x}_{\mathrm{L}}$があるため，式(40)と同様に最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$を求めるために，右辺全体を左辺に代入する．

$$\boldsymbol{x}_{\mathbf{L}} = \boldsymbol{A}^{\otimes 2}\backslash\boldsymbol{x}_{\mathrm{L}} \wedge (\boldsymbol{e} \oplus \boldsymbol{A})\backslash\boldsymbol{c}. \tag{49}$$

同様に代入を繰り返すことによって，

$$\boldsymbol{x}_{\mathrm{L}} = \boldsymbol{A}^{\otimes s}\backslash\boldsymbol{x}_{\mathrm{E}} \wedge \left(\boldsymbol{e} \oplus \boldsymbol{A} \oplus \boldsymbol{A}^{\otimes 2} \oplus \cdots \oplus \boldsymbol{A}^{\otimes s-1}\right)\backslash\boldsymbol{c}. \tag{50}$$

よって最早完了時刻$\boldsymbol{x}_{\mathrm{L}}$は，

$$\boldsymbol{x}_{\mathrm{L}} = \boldsymbol{A}\backslash\boldsymbol{c}, \tag{51}$$

となり，$\boldsymbol{c} = \boldsymbol{P}\backslash(\boldsymbol{C}\backslash\boldsymbol{y}_{\mathrm{E}})$であるため，次のように表現することができる．

$$\boldsymbol{x}_{\mathrm{L}} = (\boldsymbol{C} \otimes \boldsymbol{A})\backslash\boldsymbol{y}_{\mathrm{E}}. \tag{52}$$

ただし，

$$\boldsymbol{A} = \boldsymbol{P} \otimes (\boldsymbol{F} \otimes \boldsymbol{P})^{*}. \tag{53}$$

最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$によって導かれる最遅入力時刻$\boldsymbol{u}_{\mathrm{L}}$は，

$$\boldsymbol{u}_{\mathrm{L}} = \boldsymbol{B} \backslash \boldsymbol{x}_{\mathrm{L}}. \tag{54}$$

これらより，全ての工程における全体の余裕時間$\boldsymbol{m}_0$は，

$$\boldsymbol{m} = (\boldsymbol{x}_{\mathrm{L}} + \boldsymbol{d}) - \boldsymbol{x}_{\mathrm{E}}. \tag{55}$$

全ての工程は，$[\boldsymbol{m}]_i = 0$および$[\boldsymbol{m}]_i > 0$に従う 2 種類に分類することができ，前者はクリティカルな工程を判定し，一方後者はノンクリティカルな工程を判定する．

3　メタヒューリスティクスを用いた CCPM-MPL における資源競合の解消

本章では，文献[13][14]を基にクリティカルチェーンプロジェクトマネジメント[4][5]について述べ，本研究で用いる Critical Chain Project Management-Max-Plus Linear[3]と呼ばれる日程計画の方法論を定義し，資源の競合を解消した際の日程を計画する．効率的に日程を計画するために，先行研究が提案するアルゴリズム[6]と本研究が提案するアルゴリズム[15]を述べ，二つの方法を比較するために計算実験を行う．

3.1 クリティカルチェーンプロジェクトマネジメント

クリティカルチェーンプロジェクトマネジメントの概念について述べる．製造業などのプロジェクトは，しばしば初期に計画される日程を超過する場合がある．なぜなら外的要因などによる不測の実行時間が不確実な事象によって引き起こされるからである．このような事象を解決するために，クリティカルチェーンプロジェクトマネジメント(CCPM)に着目する．CCPM とは，制約理論[16]に基づき Goldratt[4]によって発明され，Leach[5]によって開発された，工期の遅れを防止するための有効なプロジェクトマネジメントのための手法の一つである．

CCPM は，Program Evaluation and Review Technique(PERT)におけるいくつかの欠点を改善しており，日程計画のための手法として広く使われている．PERT とはプロジェクト全体における，最長経路を計算するために，クリティカルパスを特定する手法の一つである．PERT では，プロジェクトにおける各工程の作業が，初期に計画された日程に従い処理されることを前提としている．これにより，各工程の作業が実行時間内に完了できるように，日程を計画する際に各工程に十分な実行時間を与える．よって各工程の作業に遅れが生じた場合，プロジェクト全体の遅れへとつながる．これに対して CCPM では十分な実行時間を与えず，各工程の作業が 50%の確率で実行時間内に完了できるように日程を計画し，プロジェクト最大完了時間を短縮する．

CCPM は，人員や機械などの資源の競合を解決することを目的の一つとする．PERT などの従来の方法においては資源を考慮しておらず，資源の不足や競合によってプロジェクト全体に遅れを引き起こす可能性がある．CCPM は資源の競合を解決するために，初期の日程計画において資源競合が起きている工程の先行関係を変更する．したがって，プロジェクトの初期に計画される日程と資源の競合を解消した際の日程では，クリティカルパスが変化する可能性がある．CCPM では，資源の競合を解消した際のクリティカルパスを，クリティカルチェーンと呼ぶ．

CCPM では，各工程の作業を 50%の確率で時間内に完了できるように日程を計画するため，プロジェクト全体の完了時刻に遅れが生じる可能性がある．そこで各工程に遅れが生じた場合，バッファと呼ばれる二種類の仮想的な工程を用いて各工程での遅れを防止し，プロジェクト全体の遅れを防止する．

これらのことから CCPM の目的とは，資源数が限られた状況下において，最大完了時間を短縮し，工期の遅れを防止することであると言える．

3.2 Critical Chain Project Management-Max-Plus Linear

以下に関連する行列およびベクトルを定義し，第 2 章に基づいて Critical Chain Project Management-Max-Plus Linear(CCPM-MPL)[3]を定義する．

- n：工程数,
- p：出力数,
- q：入力数,
- $\boldsymbol{B}_0 \in \mathbb{R}_{\max}^{n\times q}$：入力行列，$[\boldsymbol{B}_0]_{ij}$ ={e：工程iが外部入力jを持つとき，ε：それ以外のとき},
- $\boldsymbol{C}_0 \in \mathbb{R}_{\max}^{p\times n}$：出力行列，$[\boldsymbol{C}_0]_{ij}$ ={e：工程jが外部出力iを持つとき，ε：それ以外のとき},
- $\boldsymbol{F}_0 \in \mathbb{R}_{\max}^{n\times n}$：隣接行列，$[\boldsymbol{F}_0]_{ij}$ ={e：工程iが先行工程jを持つとき，ε：それ以外のとき},
- $\boldsymbol{d} \in \mathbb{R}_{\max}^{n}$：システムパラメータ，$[\boldsymbol{d}]_i$：工程$i$の実行時間,
- $\boldsymbol{u} \in \mathbb{R}_{\max}^{q}$：入力ベクトル，$[\boldsymbol{u}]_i$：工程$i$における外部からの入力時刻,
- $\boldsymbol{y} \in \mathbb{R}_{\max}^{p}$：出力ベクトル，$[\boldsymbol{y}]_i$：工程$i$における外部への出力時刻,
- $\boldsymbol{x} \in \mathbb{R}_{\max}^{n}$：状態ベクトル，$[\boldsymbol{x}]_i$：工程$i$における開始時刻もしくは終了時刻.

全ての工程における最早完了時刻$\boldsymbol{x}_{\mathrm{E}}$は,

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A}_0 \otimes \boldsymbol{B}_0 \otimes \boldsymbol{u}, \tag{56}$$

ただし,

$$\boldsymbol{A}_0 = (\boldsymbol{P}_0 \otimes \boldsymbol{F}_0)^* \otimes \boldsymbol{P}_0, \tag{57}$$

$$\boldsymbol{P}_0 = \mathrm{diag}(\boldsymbol{d}). \tag{58}$$

行列$\boldsymbol{A}_0 \in \mathbb{R}_{\max}^{n\times n}$と$\boldsymbol{P}_0 \in \mathbb{R}_{\max}^{n\times n}$は，それぞれ重み付き隣接行列と重み行列である．全ての出力遷移における最早終了時刻$\boldsymbol{y}_{\mathrm{E}}$は,

$$\boldsymbol{y}_{\mathrm{E}} = \boldsymbol{C}_0 \otimes \boldsymbol{x}_{\mathrm{E}}. \tag{59}$$

次に式(59)を用いて，全ての工程における最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$は,

$$\boldsymbol{x}_{\mathrm{L}} = (\boldsymbol{C}_0 \otimes \boldsymbol{A}_0) \backslash \boldsymbol{y}_{\mathrm{E}}. \tag{60}$$

最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$によって導かれる最遅入力時刻$\boldsymbol{u}_{\mathrm{L}}$は,

$$\boldsymbol{u}_{\mathrm{L}} = \boldsymbol{B}_0 \backslash \boldsymbol{x}_{\mathrm{L}}. \tag{61}$$

これらより，全ての工程における全体の余裕時間$\boldsymbol{m}_0$は,

$$\boldsymbol{m}_0 = (\boldsymbol{x}_\mathrm{L} + \boldsymbol{d}) - \boldsymbol{x}_\mathrm{E}. \tag{62}$$

全ての工程は，2 種類に分類することが可能である．$[\boldsymbol{m}_0]_i = 0$と$[\boldsymbol{m}_0]_i > 0$は，それぞれクリティカルな工程とノンクリティカルな工程である．それぞれの工程をクリティカルもしくはノンクリティカルかのどちらかに分類するために，二つのベクトル$\boldsymbol{a}_0, \boldsymbol{b}_0 \in \mathbb{R}_{\max}^n$を定義する．

$$[\boldsymbol{a}_0]_i = \{e: \text{if } [\boldsymbol{m}_0]_i = 0, \varepsilon: \text{if } [\boldsymbol{m}_0]_i > 0\}, \tag{63}$$
$$[\boldsymbol{b}_0]_i = \{e: \text{if } [\boldsymbol{m}_0]_i > 0, \varepsilon: \text{if } [\boldsymbol{m}_0]_i = 0\}. \tag{64}$$

このとき，二つの行列$\boldsymbol{P}_\mathrm{a}, \boldsymbol{P}_\mathrm{b} \in \mathbb{R}_{\max}^{n\times n}$を定義する．

$$\boldsymbol{P}_\mathrm{a} = \mathrm{diag}(\boldsymbol{a}_0) \otimes \boldsymbol{P}_0, \tag{65}$$
$$\boldsymbol{P}_\mathrm{b} = \mathrm{diag}(\boldsymbol{b}_0) \otimes \boldsymbol{P}_0. \tag{66}$$

CCPM において，フィーディングバッファ(FB)とプロジェクトバッファ(PB)と呼ばれている 2 種類のバッファがある．それぞれのバッファは，

$$\boldsymbol{r}_\mathrm{feed} = [\boldsymbol{P}_\mathrm{b} \otimes (\boldsymbol{F}_0 \otimes \boldsymbol{P}_\mathrm{b})^* \otimes \boldsymbol{g}_0]/2, \tag{67}$$
$$\boldsymbol{r}_\mathrm{proj} = [\boldsymbol{P}_\mathrm{a} \otimes (\boldsymbol{F}_0 \otimes \boldsymbol{P}_\mathrm{a})^* \otimes \boldsymbol{g}_0]/2, \tag{68}$$

ただし，

$$\boldsymbol{g}_0 = [e\ e\ \cdots e]^\mathrm{T} \in \mathbb{R}_{\max}^n. \tag{69}$$

FB はノンクリティカルな工程がクリティカルな工程に同期する位置に挿入する．PB はプロジェクトの遅れを防止するために，外部出力の直前に挿入される．図 2 は，それぞれのバッファを挿入する位置の例を示している．2 種類のバッファの挿入した際の先行関係を反映するために，隣接行列$\boldsymbol{F}_0$と出力行列$\boldsymbol{C}_0$に重みを被せる．

$$\boldsymbol{F}_\mathrm{bufs} = \boldsymbol{F}_0 \oplus \mathrm{diag}(\boldsymbol{a}_0) \otimes \boldsymbol{F}_0 \otimes \mathrm{diag}(\boldsymbol{r}_\mathrm{feed}), \tag{70}$$
$$\boldsymbol{C}_\mathrm{bufs} = \boldsymbol{C}_0 \otimes \left[\mathrm{diag}\left(\boldsymbol{r}_\mathrm{proj}\right) \oplus \mathrm{diag}(\boldsymbol{r}_\mathrm{feed})\right]. \tag{71}$$

ここで，二つのバッファ挿入後の最早完了時刻$\boldsymbol{x'}_\mathrm{E}$と最早終了時刻$\boldsymbol{y'}_\mathrm{E}$を計算する．全ての工程における最早完了時刻$\boldsymbol{x'}_\mathrm{E}$は，

$$\boldsymbol{x'}_\mathrm{E} = \boldsymbol{A'}_0 \otimes \boldsymbol{B}_0 \otimes \boldsymbol{u}, \tag{72}$$
$$\boldsymbol{A'}_0 = \boldsymbol{P}_0 \otimes (\boldsymbol{F}_\mathrm{bufs} \otimes \boldsymbol{P}_0)^*. \tag{73}$$

行列$\boldsymbol{A'}_0 \in \mathbb{R}_{\max}^{n\times n}$は，挿入された二種類のバッファを反映した遷移行列である．全ての出力遷移における最早終了時刻$\boldsymbol{y'}_\mathrm{E}$は，

$$\boldsymbol{y'}_\mathrm{E} = \boldsymbol{C}_\mathrm{bufs} \otimes \boldsymbol{x'}_\mathrm{E}. \tag{74}$$

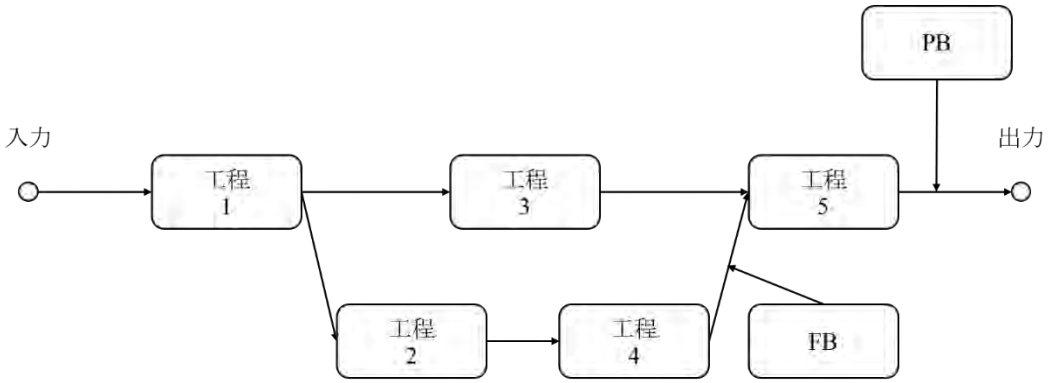

図 2. PB と FB の挿入する位置の例.

本研究ではプロジェクトの出力を一つとするため，目的関数を最早終了時刻y'_{E}とし，最小化する．

3.3 資源競合の解消

関連する行列およびベクトルを追加し，資源競合の解消の方法について述べる．

- l：資源数,
- s：一つの資源が処理する最大工程数,
- $\boldsymbol{S} \in \mathbb{N}^{l \times s}$：工程の処理順番，$[\boldsymbol{S}]_{ij}$ ={k：資源iがj番目に工程kを処理するとき，0：それ以外のとき}.

図 3 は，資源競合を解消する前の 5 工程のプロジェクトの例である．それぞれの工程の実行時間は，$\boldsymbol{d} = [3 \quad 4 \quad 5 \quad 2 \quad 1]^{\mathrm{T}}$．隣接行列$\boldsymbol{F}_0$は，

$$\boldsymbol{F}_0 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & e & \varepsilon \end{bmatrix}. \tag{75}$$

資源 1 は工程 1，4，と 5 を処理し，資源 2 は工程 2 と 3 を処理する．資源 2 が工程 2 と工程 3 を同時に処理するとき，資源競合が生じる．図 4 は，資源競合を解消した後の 5 工程のプロジェクトの例である．二つの先行関係を追加することによって，工程 2 と工程 3 の競合を避け，図 4 において破線によって表される．次に，資源競合の解消を反映した隣接行列$\boldsymbol{F'}_0$は，

$$\boldsymbol{F'}_0 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & e & \varepsilon \end{bmatrix}. \tag{76}$$

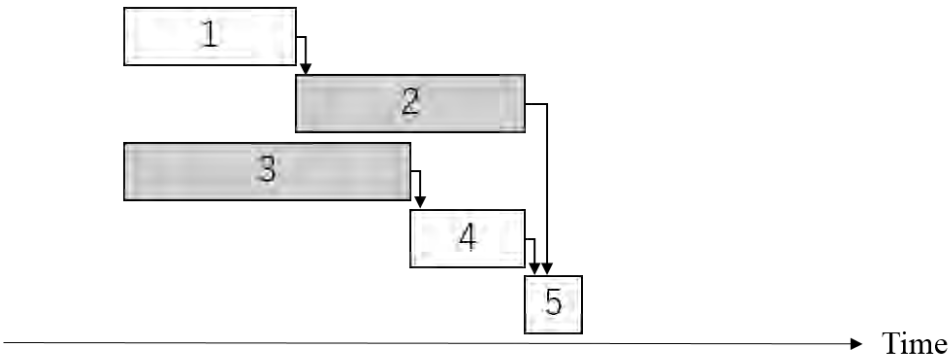

図 3. 資源競合解消前の 5 工程のプロジェクト例.

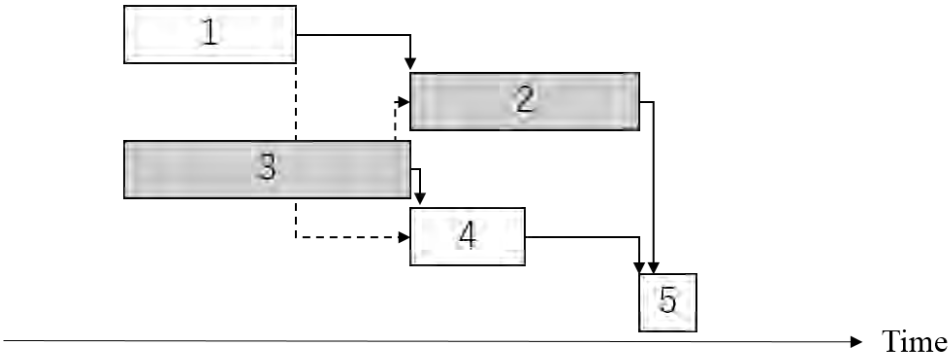

図 4. 資源競合解消後の 5 工程のプロジェクト例.

また，工程処理順番$\boldsymbol{S}$は

$$\boldsymbol{S} = \begin{bmatrix} 1 & 4 & 5 \\ 3 & 2 & 0 \end{bmatrix}. \tag{77}$$

資源競合を解消するために全数列挙法を用いた数値実験の結果，工程数が 20 のとき最大計算時間が 10 時間を越えた．それゆえ本研究では短い計算時間では，効率的に日程を計画するための近似解法の開発を考える．

3.4 二つのメタヒューリスティクスの提案

近似解法としてメタヒューリスティクスを基づいたアルゴリズムを提案する．先行研究である古賀[6]が提案する遺伝的アルゴリズム(GA)に基づいたアルゴリズムと，本研究が提案する焼きなまし法(SA)に基づいたアルゴリズム[15]について述べる．初期解を得るために，三つの近似解法のアルゴリズム[6]を用いる．

3.4.1三つの近似解法に基づいたアルゴリズム

初期解を得るために最早開始時刻法，最遅開始時刻法，中間開始時刻法による三つの近似解法を用いる．最早開始時刻法とは，式(56)を用いて最早開始時刻$\boldsymbol{s}_{\mathrm{E}}$を求め，各資源に工程を分け，開始時刻の早い順に工程を処理する近似解法である．最遅開始時刻法とは，最早開始時刻法と同様に，式(60)を用いて最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$を求め，各資源に工程を分け，開始時刻の早い順に工程を処理する近似解法である．中間時刻法とは，最早開始時刻$\boldsymbol{s}_{\mathrm{E}}$および最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$を用いて中間開始時刻$\boldsymbol{x}_{\mathrm{M}}$を求め，各資源に工程を分け，開始時刻の早い順に工程を処理する近似解法である．最早開始時刻$\boldsymbol{s}_{\mathrm{E}}$，最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$，中間開始時刻$\boldsymbol{x}_{\mathrm{M}}$の求め方は，以下のとおりである．

$$\boldsymbol{s}_{\mathrm{E}} = \boldsymbol{x}_{\mathrm{E}} - \boldsymbol{d}, \tag{78}$$

$$\boldsymbol{x}_{\mathrm{L}} = (\boldsymbol{C}_0 \otimes \boldsymbol{A}_0)\backslash \boldsymbol{y}_{\mathrm{E}}, \tag{79}$$

$$\boldsymbol{x}_{\mathrm{M}} = (\boldsymbol{s}_{\mathrm{E}} + \boldsymbol{x}_{\mathrm{L}})/\boldsymbol{2}. \tag{80}$$

本研究では三つの近似解法を用いて，各資源の工程処理順序$\boldsymbol{S}$を求め，初期解とする．工程数を$\boldsymbol{i}$としたとき，以下に三つの近似解法に基づいたアルゴリズムを示す．

三つの近似解法に基づいたアルゴリズム

STEP 1. 最早開始時刻$\boldsymbol{s}_{\mathrm{E}}$，最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$，中間開始時刻$\boldsymbol{x}_{\mathrm{M}}$を求める．

STEP 2. $[\boldsymbol{s}_{\mathrm{E}}]_i$，$[\boldsymbol{x}_{\mathrm{L}}]_i$，$[\boldsymbol{x}_{\mathrm{M}}]_i$を比較し，小さい順に工程処理順序$\boldsymbol{S}$にソートする．

STEP 3. 全ての工程を比較するまで，STEP.2 を繰り返す．

先行研究で提案されているアルゴリズムと，本研究で提案するアルゴリズムの初期解に，上記のアルゴリズムで得る初期解を用いる．

3.4.2遺伝的アルゴリズムに基づいたアルゴリズム

先行研究[6]が提案する，遺伝的アルゴリズム(GA)に基づいたアルゴリズムについて述べる．GA とは生物の進化の過程を模したアルゴリズムであり，組合せ最適化問題に用いられるメタヒューリスティクスの一つである．先行研究では交叉を二点交叉とし，生存をランキング選択としアルゴリズムを構築している．二点交叉とは，二つの異なる工程処理順序において，工程をそれぞれランダムに一つ以上選び，選ばれた工程を入れ替える手法である．ランキング選択とは，評

価値の良い解にランクを付け，ランクのうち新たな解として残す個数を決め，ランダムで解を選ぶ手法である．具体的に選択の方法は，最も評価値の良い解をランク A，その他の解をランク B とし，ランク A から一つ，ランク B から二つ新たな解として残す．最も良い解を必ず新たな解に残こすため，エリート選択も採用している．以下に GA に基づいたアルゴリズムを示す.

GA に基づいたアルゴリズム

STEP 1. 突然変異パラメータm_{ut}と終了条件を初期化する．

STEP 2. 初期解$\boldsymbol{S}$を生成し，最早終了時刻y'_{E1}を求める．

STEP 3. $\boldsymbol{S}$における行ベクトルを二つに分割し入れ替え，新たな解$\boldsymbol{S}'$を生成する．

STEP 4. m_{ut}の確率で突然変異が起こるとき，STEP 5.へ進む．それ以外の突然変異が起こらないとき，STEP 6.へ進む.

STEP 5. 同一資源において，$\boldsymbol{S}'$における二つの工程をランダムに入れ替える．

STEP 6. $\boldsymbol{S}'$を用いて，新たな最早終了時刻y'_{E2}を求める．

STEP 7. $y'_{\mathrm{E2}} < y'_{\mathrm{E1}}$のとき，$\boldsymbol{S} := \boldsymbol{S}'$と$y'_{\mathrm{E1}} := y'_{\mathrm{E2}}$を更新する．

STEP 8. 終了条件を満たしたとき，停止する．そうでなければ，STEP 4.へ戻る．

3.4.3焼きなまし法に基づいたアルゴリズム

本研究では，焼きなまし法(SA)に基づいたアルゴリズム[15]を提案する．SA とは，金属の焼きなましを模倣したアルゴリズムである. SA はランダム性を用いることで, 局所最適解に陥らず, 大域最適解に到達することが可能である．また，メタヒューリスティクスの一つであり，近似解を効率的に得るための手法として知られている．文献[17]は最も基本とされる考え方であるが，本研究では文献[18]の手法を用いる．本研究は近傍解を生成するために，局所探索法である 2-opt neighborhood[19]を用いる．局所探索法とは，メタヒューリスティックの近傍の解の更新に用いられ，多くのメタヒューリスティックは局所探索法に基づいている．本研究では工程処理順序$\boldsymbol{S}$において，同一資源内の二つの工程を入れ替え，解を更新する際に用いる．式(82)は，式(81)に 2-opt neighborhood を用いて解を更新した際の例である．

$$\boldsymbol{S} = \begin{bmatrix} 1 & 4 & 5 \\ 3 & 2 & 0 \end{bmatrix}, \tag{81}$$

$$\boldsymbol{S}' = \begin{bmatrix} 1 & 5 & 4 \\ 3 & 2 & 0 \end{bmatrix}. \tag{82}$$

以下に SA に基づいたアルゴリズムを示す.

SA に基づいたアルゴリズム

STEP 1. 温度パラメータT_0と温度減少パラメータγを初期化する.

STEP 2. 初期解$\boldsymbol{S}$を生成し，最早終了時刻$\boldsymbol{y'}_{\mathrm{E1}}$を求める.

STEP 3. 近傍解$\boldsymbol{S'}$を生成し，新たな最早終了時刻$\boldsymbol{y'}_{\mathrm{E2}}$を求める.

STEP 4. $\Delta E = [\boldsymbol{y'}_{\mathrm{E2}}]_i - y'_{\mathrm{E1}}$を計算する.

STEP 5. $\Delta E < 0$のとき，$\boldsymbol{S} := \boldsymbol{S'}$と$\boldsymbol{y'}_{\mathrm{E1}} := \boldsymbol{y'}_{\mathrm{E2}}$を更新する．$\Delta E \geq 0$のとき，$\exp(-\Delta E/T)$の確率で$\boldsymbol{S} := \boldsymbol{S'}$と$\boldsymbol{y'}_{\mathrm{E1}} := \boldsymbol{y'}_{\mathrm{E2}}$を更新する.

STEP 6. 温度パラメータT_kを$T_k = \gamma^k T_0 \ \ (0.8 \leq \gamma \leq 0.99)$と更新する.

STEP 7. STEPS 3–6 を繰り返す.

STEP 8. 温度パラメータが十分に小さくなったとき，停止する．そうでなければ，STEP 3 に戻る.

ただし，γは温度パラメータT_0を減少するための温度減少パラメータである.

3.5 計算実験

実験環境と計算実験に用いたサンプルデータについて述べ，全数列挙法を用いて厳密値と平均の計算時間を求めた後，本研究が提案する SA に基づいたアルゴリズムと，先行研究が提案する GA に基づいたアルゴリズムを用いて得る近似解を，平均の近似比，評価値，計算時間の観点において比較する.

3.5.1実験環境とサンプルデータ

数値実験に用いる PC の実行環境は，以下のとおりである.

- 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.

数値実験のためのテストケースは，以下の条件によって生成される.

- ケース数：100,
- 資源数l：10，15，20，30，40，50，60，70，80，90 工程のとき，資源数はそれぞれ 3，5，7，7，7，7，8，8，9，9 である,
- 工程iにおける資源数：[1，l]∈ ℕの一様乱数,

- 工程iにおける実行時間$[d]_i$：[5, 20] ∈ ℕの一様乱数.

3.5.2実験結果

表 1 は，全数列挙法による厳密値および計算時間(s)の平均である．表 1 から，工程数 10，15 のとき平均の計算時間は 1s より短いが，工程数 20 のとき平均の計算時間が 377s となり長いことがわかる．

SA と GA に基づいたアルゴリズムを用いて近似解を得る．文献[15]より SA に基づいたアルゴリズムにおいて，温度パラメータと温度減少パラメータをそれぞれ$T = 1$と$\gamma = 0.85$とする．文献[6]より GA に基づいたアルゴリズムにおいて，突然変異パラメータと終了条件をそれぞれ$m_{ut} = 0.2$と$i = 200$とする．表 2 は，SA と GA に基づいたアルゴリズムによる近似値と，厳密値を比較した際の近似比による解の性能を表している．表 2 において，全ての工程数で GA による解の性能は，SA による解の性能より良い．表 3 は，SA と GA に基づいたアルゴリズムによる近似値の平均である．表 3 において，工程数 10 から 40 においては GA による近似値が小さく，工程数 50 から 90 においては SA による近似値が小さい．図 5 は，SA と GA に基づいたアルゴリズムによる計算時間(s)の平均である．図 5 において，全ての工程において SA による近似解の計算時間は，GA による近似解の計算時間より短い．また 90 工程において，GA による計算時間は，SA による計算時間より約 14 倍長いことがわかる．

これらの結果から，SA に基づいたアルゴリズムは計算時間の観点において有効であった．一方，GA に基づいたアルゴリズムは近似解の性能の観点において良い値であった．また，工程数 10 から 40 においては，GA による近似解が良く，工程数 50 から 90 においては，SA による近似解が良いことがわかった．

表 1. 全数列挙法による厳密値および計算時間(s)の平均．工程数は 10 から 20 に固定，各ケース数は 100.

工程数	10	15	20
厳密値	155.22	215.50	259.00
計算時間(s)	0.022	0.928	377.704

表 2. SA と GA に基づいたアルゴリズムによる近似比による解の性能．工程数は 10 から 20 に固定，各ケース数は 100.

近似解法	工程数		
	10	15	20
SA	1.001	1.002	1.006
GA	1.000	1.001	1.000

表 3. SA と GA に基づいたアルゴリズムによる近似値の平均．工程数は 10 から 90 に固定，各ケース数は 100.

近似解法	工程数								
	10	20	30	40	50	60	70	80	90
SA	155.34	260.48	365.49	467.93	563.80	624.35	737.87	790.88	889.40
GA	155.22	259.06	361.76	465.45	568.96	631.48	754.70	813.60	910.81

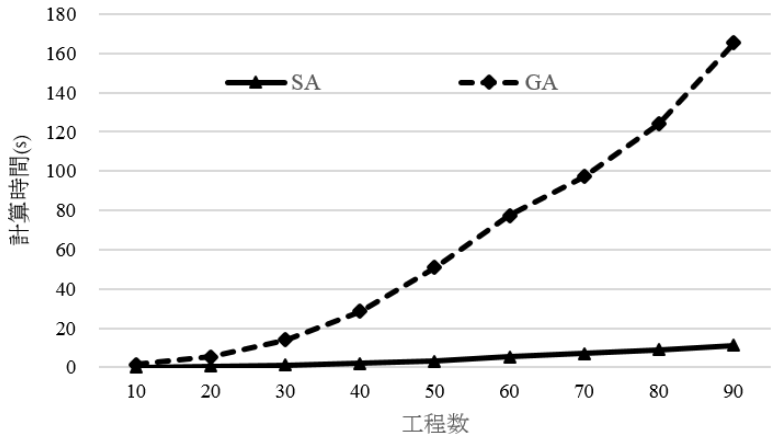

図 5. SA と GA に基づいたアルゴリズムによる計算時間(S)の平均．工程数は 10 から 90 に固定，各ケース数は 100，横軸に工程数，縦軸に計算時間.

4　混合整数計画問題のための制約充足問題における Max-Plus 線形システムの緩和と解析

本章では，文献[7][20]を基に Max-plus 代数[1][2]における基本演算子を制約充足問題として緩和し，第 2 章に基づいて Max-Plus 線形表現を定義した後，Max-Plus 線形表現を緩和し混合整数計画問題における制約充足問題として定式化する．二つのパラメータの設定の方法を述べた後，計算実験を通じて定式化によって解が得られることを示す.

4.1 制約充足問題における演算子の緩和

Max-plus 代数において基本演算子である加算と乗算を緩和し，混合整数計画問題における制約充足問題として定式化する． $x, y, z \in \mathbb{R}_{\max}$のとき，

$$z = x \oplus y, \tag{83}$$

$$z = x \otimes y, \tag{84}$$

に着目する. 式(83)に関して, Max-plus 代数における 2 数の加算を制約充足問題として緩和する.

$$\begin{gathered} z \geq x, \\ z \geq y, \\ z \leq x + M(1 - s_1), \\ z \leq y + M(1 - s_2), \\ (s_1, s_2) \in \{0,1\}, \\ s_1 + s_2 \geq 1, \end{gathered} \tag{85}$$

ただし，s_1, s_2はバイナリ変数であり，Mは big-M と呼ばれる大きな正の定数である．big-M およびs_1，s_2は，等号を制御する役割を果たす．加算$z = \bigoplus_{i=1}^{n} x_i$を以下のとおり制約充足問題として緩和する.

$$\begin{gathered} z \geq x_i \ \forall i \in \mathcal{V}_n, \\ z \leq x_i + M(1 - s_i) \ \forall i \in \mathcal{V}_n, \\ s_i \in \{0,1\} \ \forall i \in \mathcal{V}_n, \\ \sum_{k=1}^{n} s_k \geq 1, \end{gathered} \tag{86}$$

ただし，$\mathcal{V}_n = \{1,2,\cdots,n\}$である．$\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}_{\max}^{n \times m}$であり$\boldsymbol{Z} \in \mathbb{R}_{\max}^{m \times r}$であるとき，二つの行列における加算$\boldsymbol{P} = \boldsymbol{X} \oplus \boldsymbol{Y}$を以下のとおり定式化する.

$$
\begin{aligned}
p_{ij} \geq x_{ij} \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\
p_{ij} \geq y_{ij} \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\
p_{ij} \leq x_{ij} + M\left(1 - s_{ij1}\right) \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\
p_{ij} \leq y_{ij} + M\left(1 - s_{ij2}\right) \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\
\left(s_{ij1}, s_{ij2}\right) \in \{0,1\} \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\
s_{ij1} + s_{ij2} \geq 1 \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m.
\end{aligned} \tag{87}
$$

式(84)に関して，Max-plus 代数における 2 数の乗算を以下のとおり定式化する．

$$
z = x + y. \tag{88}
$$

二つの行列における乗算$\boldsymbol{Q} = \boldsymbol{Y} \otimes \boldsymbol{Z}$を，以下のとおり定式化する．

$$
\begin{aligned}
q_{ij} \geq y_{ik} + z_{kj} \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_r, \forall k \in \mathcal{V}_m, \\
q_{ij} \leq y_{ik} + z_{kj} + M\left(1 - s_{ijk}\right) \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_r, \forall k \in \mathcal{V}_m, \\
s_{ijk} \in \{0,1\} \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_r, \forall k \in \mathcal{V}_m, \\
\sum\nolimits_{l=1}^{m} s_{ijl} \geq 1 \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_r.
\end{aligned} \tag{89}
$$

次に，二つの補完的な演算子∧, ⊙に着目する．

$$
z = x \wedge y, \tag{90}
$$

$$
z = x \odot y. \tag{91}
$$

$x \oplus y$の緩和と同様に，式(90)を緩和する．

$$
\begin{aligned}
z \leq x, \\
z \leq y, \\
z \geq x - M(1 - s_1), \\
z \geq y - M(1 - s_2), \\
(s_1, s_2) \in \{0,1\}, \\
s_1 + s_2 \geq 1.
\end{aligned} \tag{92}
$$

最小化$z = \wedge_{i=1}^{n} x_i$を以下のとおり緩和する．

$$
\begin{aligned}
z \leq x_i \ \forall i \in \mathcal{V}_n, \\
z \geq x_i - M(1 - s_i) \ \forall i \in \mathcal{V}_n, \\
s_i \in \{0,1\} \ \forall i \in \mathcal{V}_n, \\
\sum\nolimits_{k=1}^{n} s_k \geq 1.
\end{aligned} \tag{93}
$$

$\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}_{\max}^{\mathrm{n}\times\mathrm{m}}$であり$\boldsymbol{Z} \in \mathbb{R}_{\max}^{n\times\mathrm{r}}$であるとき，二つの行列における最小化$\boldsymbol{P} = \boldsymbol{X} \wedge \boldsymbol{Y}$を以下のとおり定式化する．

$$\begin{gathered} p_{ij} \le x_{ij} \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\ p_{ij} \le y_{ij} \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\ p_{ij} \ge x_{ij} - M\left(1 - s_{ij1}\right) \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\ p_{ij} \ge y_{ij} - M\left(1 - s_{ij2}\right) \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\ \left(s_{ij1}, s_{ij2}\right) \in \{0,1\} \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m, \\ s_{ij1} + s_{ij2} \ge 1 \ \forall i \in \mathcal{V}_n, \forall j \in \mathcal{V}_m. \end{gathered} \tag{94}$$

式(91)は以下のとおり定式化する．

$$z = -x + y. \tag{95}$$

擬除算子$\boldsymbol{Q} = \boldsymbol{X}^{\mathrm{T}} \odot \boldsymbol{Z}$を以下のとおり定式化する．

$$\begin{gathered} q_{ij} \le -x_{ki} + w_{kj} \ \forall i \in \mathcal{V}_m, \forall j \in \mathcal{V}_r, \forall k \in \mathcal{V}_n, \\ q_{ij} \ge -x_{ki} + w_{kj} + M\left(1 - s_{ijk}\right) \ \forall i \in \mathcal{V}_m, \forall j \in \mathcal{V}_r, \forall k \in \mathcal{V}_n, \\ s_{ijk} \in \{0,1\} \ \forall i \in \mathcal{V}_m, \forall j \in \mathcal{V}_r, \forall k \in \mathcal{V}_n, \\ \sum_{l=1}^{n} s_{ijl} \ge 1 \ \forall i \in \mathcal{V}_m, \forall j \in \mathcal{V}_r. \end{gathered} \tag{96}$$

4.2 Max-plus 線形方程式のための解法

4.2.1Max-plus 線形方程式

関連する行列とベクトルを定義した後に, 第 2 章に基づいて Max-plus 線形方程式を定義する．

- n：工程数,
- p：出力数,
- q：入力数,
- $\boldsymbol{B} \in \mathbb{R}_{\max}^{n\times q}$：入力行列，$[\boldsymbol{B}]_{ij}$ ={e：工程iが外部入力jを持つとき，ε：それ以外のとき},
- $\boldsymbol{C} \in \mathbb{R}_{\max}^{p\times n}$：出力行列，$[\boldsymbol{C}]_{ij}$ ={e：工程jが外部出力i持つとき，ε：それ以外のとき},
- $\boldsymbol{d} \in \mathbb{R}_{\max}^{n}$：システムパラメータ，$[\boldsymbol{d}]_i$：工程$i$の実行時間,
- $\boldsymbol{u} \in \mathbb{R}_{\max}^{q}$：入力ベクトル，$[\boldsymbol{u}]_i$：工程$i$における外部からの入力時刻,
- $\boldsymbol{y} \in \mathbb{R}_{\max}^{p}$：出力ベクトル，$[\boldsymbol{y}]_i$：工程$i$における外部への出力時刻,
- $\boldsymbol{x} \in \mathbb{R}_{\max}^{n}$：状態ベクトル，$[\boldsymbol{x}]_i$：工程$i$における開始時刻もしくは終了時刻.

全ての工程における最早完了時刻$\boldsymbol{x}_{\mathrm{E}}$，および全ての出力遷移における最早終了時刻$\boldsymbol{y}_{\mathrm{E}}$は,

$$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A} \otimes \boldsymbol{x}_{\mathrm{E}} \oplus \boldsymbol{b}, \tag{97}$$

$$\boldsymbol{y}_{\mathrm{E}} = \boldsymbol{C} \otimes \boldsymbol{x}_{\mathrm{E}}. \tag{98}$$

行列$\boldsymbol{A} \in \mathbb{R}_{\max}^{n \times n}$は重み付き遷移行列であり，ベクトル$\boldsymbol{b} \in \mathbb{R}_{\max}^{n}$は$\boldsymbol{b} = \boldsymbol{P} \otimes \boldsymbol{B} \otimes \boldsymbol{u}$を満たす重み付き入力ベクトルである．全ての工程における最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$，および最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$によって導かれる最遅入力時刻$\boldsymbol{u}_{\mathrm{L}}$は，

$$\boldsymbol{x}_{\mathrm{L}} = \boldsymbol{A}^{\mathrm{T}} \odot \boldsymbol{x}_{\mathrm{L}} \wedge \boldsymbol{c}, \tag{99}$$

$$\boldsymbol{u}_{\mathrm{L}} = \boldsymbol{B}^{\mathrm{T}} \odot \boldsymbol{x}_{\mathrm{L}}. \tag{100}$$

ベクトル$\boldsymbol{c} \in \mathbb{R}_{\max}^{n}$は$\boldsymbol{c} = \boldsymbol{P} \backslash (\boldsymbol{C} \backslash \boldsymbol{y}_{\mathrm{E}})$を満たす重み付き出力ベクトルである．これらより，全ての工程における全体の余裕時間$\boldsymbol{m}_0$は，

$$\boldsymbol{m} = (\boldsymbol{x}_{\mathrm{L}} + \boldsymbol{d}) - \boldsymbol{x}_{\mathrm{E}}. \tag{101}$$

全ての工程は，$[\boldsymbol{m}]_i = 0$および$[\boldsymbol{m}]_i > 0$に従う 2 種類に分類することができ，前者はクリティカルな工程を判定し，一方後者はノンクリティカルな工程を判定する．

4.2.2制約充足問題における Max-plus 線形方程式の緩和

式(97)における解法について検討する．簡単な方法は，$\boldsymbol{x}_{\mathrm{E}} \geq \boldsymbol{A} \otimes \boldsymbol{x}_{\mathrm{E}} \oplus \boldsymbol{b}$のように緩和することである．行列$\boldsymbol{x}_{\mathrm{E}}$は，$\boldsymbol{x}_{\mathrm{E}} = \boldsymbol{A}^{*} \otimes \boldsymbol{b}$によって与えられる．以下とおり，式(97)を混合整数計画問題における制約充足問題として緩和する．

$$\begin{gathered}
x_{\mathrm{E}i} \geq a_{ik} + x_{\mathrm{E}k} \ \ \forall (i,k) \in \mathcal{E}_{\boldsymbol{A}}, \\
x_{\mathrm{E}i} \leq a_{ik} + x_{\mathrm{E}k} + M(1 - s_{ik}) \ \ \forall (i,k) \in \mathcal{E}_{\boldsymbol{A}}, \\
x_{\mathrm{E}i} \geq b_i \ \ \forall (i,k) \notin \mathcal{E}_{\boldsymbol{A}}, \\
x_{\mathrm{E}i} \leq b_i + M(1 - s_{ik}) \ \ \forall (i,k) \notin \mathcal{E}_{\boldsymbol{A}}, \\
s_{ik} \in \{0,1\} \ \ \forall i,k \in \mathcal{V}_n, \\
\textstyle\sum_{l=1}^{n} s_{il} \geq 1 \ \ \forall i \in \mathcal{V}_n.
\end{gathered} \tag{102}$$

$a_{ik} = \varepsilon$のとき，$a_{ik} + x_{\mathrm{E}kj} = \varepsilon$である．式(102)は，$\boldsymbol{A}^{*}$の計算なしに$\boldsymbol{A}^{*} \otimes \boldsymbol{b}$を直接計算することができる．式(98)を緩和したあと，式(89)を基に混合整数計画問題における制約充足問題として緩和する．

$$\begin{gathered}
y_{\mathrm{E}i} \geq C_{ik} + x_{\mathrm{E}k} \ \ \forall i \in \mathcal{V}_p, \forall k \in \mathcal{V}_n, \\
y_{\mathrm{E}i} \leq C_{ik} + x_{\mathrm{E}k} + M(1 - s_{ik}) \ \ \forall i \in \mathcal{V}_p, \forall k \in \mathcal{V}_n, \\
s_{ik} \in \{0,1\} \ \ \forall i \in \mathcal{V}_p, \forall k \in \mathcal{V}_n, \\
\textstyle\sum_{l=1}^{n} s_{il} \geq 1 \ \ \forall i \in \mathcal{V}_p.
\end{gathered} \tag{103}$$

次に，式(99)のための解法を検討する．簡単な方法は$\boldsymbol{x}_{\mathrm{L}} \leq \boldsymbol{A}^{\mathrm{T}} \odot \boldsymbol{x}_{\mathrm{L}} \wedge \boldsymbol{c}$のように緩和することである．行列$\boldsymbol{x}_{\mathrm{L}}$は，$\boldsymbol{x}_{\mathrm{L}} = \boldsymbol{A}^{*} \backslash \boldsymbol{c}$によって与えられる．以下とおり，式(99)を混合整数計画問題における制約充足問題として緩和する．

$$
\begin{gathered}
x_{\mathrm{L}i} \leq -a_{ik} + x_{\mathrm{L}k} \ \forall (i,k) \in \mathcal{E}_{\mathbf{A}}, \\
x_{\mathrm{L}i} \geq -a_{ik} + x_{\mathrm{L}k} - M(1 - s_{ik}) \ \forall (i,k) \in \mathcal{E}_{\mathbf{A}}, \\
x_{\mathrm{L}i} \geq c_i \ \forall (i,k) \notin \mathcal{E}_{\mathbf{A}}, \\
x_{\mathrm{L}i} \leq c_i - M(1 - s_{ik}) \ \forall (i,k) \notin \mathcal{E}_{\mathbf{A}}, \\
s_{ik} \in \{0,1\} \ \forall i,k \in \mathcal{V}_n, \\
\sum_{l=1}^{n} s_{il} \geq 1 \ \forall i \in \mathcal{V}_n.
\end{gathered}
\tag{104}
$$

$a_{ik} = \varepsilon$のとき，$-a_{ki} + x_{\mathrm{L}kj} = \varepsilon$である．式(104)は，$\boldsymbol{A}^{*}$の計算なしに$\boldsymbol{A}^{*} \backslash \boldsymbol{c}$を直接計算することができる．式(96)を基に，式(100)を混合整数計画問題における制約充足問題として緩和する．

$$
\begin{gathered}
u_{\mathrm{L}i} \leq -B_{ik} + x_{\mathrm{L}k} \ \forall i \in \mathcal{V}_q, \forall k \in \mathcal{V}_n, \\
u_{\mathrm{L}i} \geq -B_{ik} + x_{\mathrm{L}k} + M(1 - s_{ik}) \ \forall i \in \mathcal{V}_q, \forall k \in \mathcal{V}_n, \\
s_{ik} \in \{0,1\} \ \forall i \in \mathcal{V}_q, \forall k \in \mathcal{V}_n, \\
\textstyle\sum_{l=1}^{n} s_{il} \geq 1 \ \forall i \in \mathcal{V}_q.
\end{gathered}
\tag{105}
$$

最後に式(101)を以下のとおり定式化する．

$$
m_i = (x_{\mathrm{L}i} + d_i) - x_{\mathrm{E}i} \ \forall i \in \mathcal{V}_n. \tag{106}
$$

クリティカルな工程を判定するために，式(106)を以下のとおり混合整数計画問題における制約充足問題として緩和する．

$$
\begin{gathered}
m_i - M\alpha_i \leq 0 \ \forall i \in \mathcal{V}_n, \\
m_i - (1/M)\alpha_i \geq 0 \ \forall i \in \mathcal{V}_n, \\
\alpha_i \in \{0,1\} \ \forall i \in \mathcal{V}_n.
\end{gathered}
\tag{107}
$$

余裕時間m_iが実数であるとき，式(107)は小さな正の定数$(1/M)$を用いて計算される．$\alpha_i = 0$のとき，$m_i = 0$に従い工程iはクリティカルである．一方$\alpha_i = 1$のとき，$(1/M) \leq m_i \leq M$となるため，工程iはノンクリティカルである．これは全ての工程においてクリティカルもしくはノンクリティカル判別するための技法であり，重要である．

4.3 正の定数Mの設定

max 演算子を制約充足問題として定式化する際に big-M を用いるため，文献[7]を基に正の定数Mを設定する．また混合整数計画問題をとくためのソルバーは，ゼロ元$\varepsilon(= -\infty)$を扱えないた

め，正の定数を設定する．重み付き遷移行列$\boldsymbol{A} \in \mathbb{R}_{\max}^{n\times m}$と重み付き入力ベクトル$\boldsymbol{b} \in \mathbb{R}_{\max}^{n}$の成分を用いて，

$$\varepsilon > a_+ + a_- - b_{\min}, \tag{108}$$

$$M > 2\cdot\varepsilon + a_+ + 2a_- + b_{\max}, \tag{109}$$

ただし，

$$a_+ = \sum\nolimits_{\{i,j|a_{ij}>0,\, a_{ij}\neq\varepsilon\}} a_{ij}\,, a_- = -\sum\nolimits_{\{i,j|a_{ij}<0,\, a_{ij}\neq\varepsilon\}} a_{ij}, \tag{110}$$

$$b_{\max} = \max\nolimits_{\{i|b_i\neq\varepsilon\}} b_i, b_{\min} = \min\nolimits_{\{i|b_i\neq\varepsilon\}} b_i. \tag{111}$$

4.4 計算実験

計算実験に用いる PC の実行環境は，以下のとおりである．

- Machine：Lenovo ThinkPad X240,
- CPU：Intel® Core™ i5-4210 1.70 GHz,
- OS：Microsoft Windows 8 Enterprise,
- Memory：4.0 GB.

制約充足問題を解くために，ソルバーである SCIP version 3.2.1.[21]を用いる．図 6 は，2 入力 1 出力の構造をもつ，5 工程の簡素な生産システムである[7]．入力 1 および 2 がそれぞれ$t = 3$および$t = 0$のとき，原材料を送る．

$$\boldsymbol{d} = \begin{bmatrix}1\\2\\3\\4\\5\end{bmatrix}, \boldsymbol{A} = \begin{bmatrix}\varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon\\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon\\ e & \varepsilon & \varepsilon & \varepsilon & \varepsilon\\ \varepsilon & e & \varepsilon & \varepsilon & \varepsilon\\ \varepsilon & \varepsilon & e & e & \varepsilon\end{bmatrix}, \boldsymbol{b} = \begin{bmatrix}4\\2\\\varepsilon\\\varepsilon\\\varepsilon\end{bmatrix}, \boldsymbol{c} = \begin{bmatrix}\bar{\varepsilon}\\\bar{\varepsilon}\\\bar{\varepsilon}\\\bar{\varepsilon}\\7\end{bmatrix}.$$

ベクトルおよび行列である$\boldsymbol{d}$，$\boldsymbol{A}$，$\boldsymbol{b}$および$\boldsymbol{c}$は，処理時間，先行制約，外部入力および外部出力を表している．ソルバーはゼロ元εを扱うことができないため，十分に注意して big-M およびゼロ元εを設定する必要がある．式(108)および式(109)を用いて，$M = 70$と$\varepsilon = 20$と設定する．

ソルバーを用いて実験を行った結果は，最早完了時刻$\boldsymbol{x}_{\mathrm{E}} = [4\ 2\ 7\ 6\ 12]^{\mathrm{T}}$，最早終了時刻$\boldsymbol{y}_{\mathrm{E}} = 12$，最遅開始時刻$\boldsymbol{x}_{\mathrm{L}} = [3\ 1\ 4\ 3\ 7]^{\mathrm{T}}$，最遅入力時刻$\boldsymbol{u}_{\mathrm{L}} = [3\ 1]^{\mathrm{T}}$，余裕時間$\boldsymbol{m} = [0\ 1\ 0\ 1\ 0]^{\mathrm{T}}$，およびクリティカルな工程$\boldsymbol{\alpha} = [0\ 1\ 0\ 1\ 0]^{\mathrm{T}}$である．また変数の数は 94，制約式の数は 163，計算時間(S)は 0.02 であった．これらの結果から，MPL 方程式を混合整数計画問題における制約充足問題に緩和することによって，解を得ることが可能となった．クリティカルな工程$\boldsymbol{\alpha}$を求めるために，小さな正の定数$(1/M)$を設定したことによって，0 もしくは 1 で判定することができている．

これらの結果から，MPL 方程式を混合整数計画問題における制約充足問題に緩和することによって，最早完了時刻$\boldsymbol{x}_{\mathrm{E}}$，最早終了時刻$\boldsymbol{y}_{\mathrm{E}}$，最遅開始時刻$\boldsymbol{x}_{\mathrm{L}}$，最遅入力時刻$\boldsymbol{u}_{\mathrm{L}}$，余裕時間$\boldsymbol{m}$，およびクリティカルな工程$\boldsymbol{\alpha}$を求めることが可能となった．

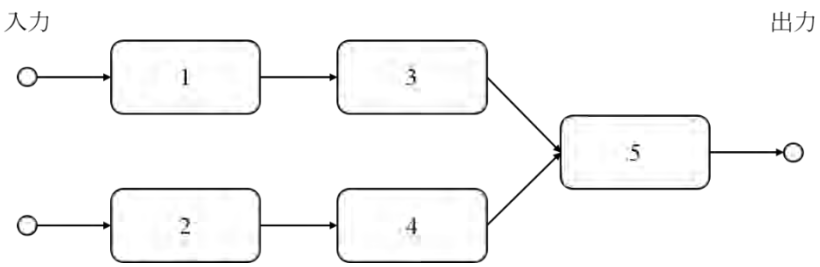

図 6. 2 入力 1 出力の 5 工程プロジェクト．(文献[7]より参照)

5　おわりに

本論文では, Max-plus 線形表現に基づいた離散事象システムの解析と制御について研究した．製造業などのプロジェクトにおいて，資源が限られている状況下で工期を短縮し，かつ納期遅れを防止するために CCPM-MPL と呼ばれる日程計画の方法論を用い，資源競合を解消後の効率的な日程を計画した．資源競合の解消は，組み合わせ最適化問題であった．短い計算時間で近似解を得るために，SA と GA に基づいたアルゴリズムを用いた．二つの手法を比較した結果，SA に基づいたアルゴリズムは，計算時間の観点において有効であった．一方，GA に基づいたアルゴリズムは，近似解の性能の観点において良い値であった．

Max-plus 線形方程式において，行列$\boldsymbol{A}$もしくはベクトル$\boldsymbol{b}$の成分に変数を含むとき最適解は関数になるため，混合整数計画問題における制約充足問題として，Max-plus 線形表現にるプロジェクトの日程計画のための解法を構築した．最早完了時刻，最遅開始時刻および最遅入力時を計算するために，定式を緩和することで解いた．さらに，小さな正の定数$(1/M)$を用いることによって，全ての工程をクリティカルもしくはノンクリティカルな工程に分類するための方法を構築した．

参考文献

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

[2] Baccelli, F., Cohen, G., Older, G.J., and Quadrat, J.P.: Synchronization and Linearity. An Algebra for Discrete Event Systems, John Wiley & Sons, New York, 1992.

[3] Goto, H.: Forward-Compatible Framework with Critical-Chain Project Management using a Max-Plus Linear Representation, OPSEARCH, vol.53, pp.1–16, 2016.

[4] Goldratt, E.M.: Critical Chain. North River Press, Great Barrington, 1997.

[5] Leach, L.P.: Critical Chain Project Management. 2nd Edition, Artech House, Boston, 2005.

[6] 古賀裕紀: Critical Chain 法の枠組みにおいて効率的に資源を活用するためのスケジューリング方法, 法政大学大学院 理工学研究科 システム工学専攻 修士課程論文, 2015.

[7] Goto, H.: Reduction of Max-Plus Algebraic Equations to Constraint Satisfaction Problems for Mixed Integer Programming, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, vol.E100-A, no.2, pp.427–430, 2017.

[8] Goto, H., Masuda, S.: On the Properties of the Greatest Subsolution for Linear Equations in the Max-Plus Algebra, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, vol.E87-A, no.2, pp.424–432, 2004.

[9] Goto, H.: A lightweight model predictive controller for repetitive discrete event systems, Asian J. Control, vol.15, no.4, pp.1081–1090, 2013.

[10] Goto, H.: Modelling Methods Based on Discrete Algebraic Systems, Aitor Goti Eds., Discrete Event Simulations, chapter 3, pp.35–62, 2010.

[11] 五島洋行: Max-plus 代数を用いて日程計画問題を考える, 計測と制御, vol.52, no.12, pp.1083–1089, 2013.

[12] Goto, H.: A fast computation for the state vector in a max-plus algebraic system with an adjacency matrix of a directed acyclic graph, Journal of Control, Measurement, and System Integration, vol.4, no.5, pp.361–364, 2011.

[13] Takahashi, H., Goto, H., and Kasahara, M.: Application of a critical chain projectmanagement based framework on max-plus linear systems, International Journal of Computational Science, vol.3, no.2, pp.117–132, 2009.

[14] 吉田翔太郎: 作業時間の不確実性を考慮した max-plus 線形システム, 長岡技術大学大学院 工学研究科 経営情報システム工学専攻 修士課程論文, 2012.

[15] Yokoyama, H., Goto, H.: Resolution of Resource Contentions in the CCPM-MPL Using Simulated Annealing and Genetic Algorithm. American Journal of Operations Research, vol.6, no.6, pp.480–488.

[16] Goldratt, E.M.: Theory of Constraints: And how it should be implemented. North River Pr, 1990.

[17] Metropolis, N., Rosenbluth A.W., Rosenbluth, M.N., and Teller, A.H.: Equation of state calculations by fast computing machines, The Journal of Chemical Physics, vol.21, No.6, pp.1087–1092, 1953.

[18] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P.: Optimization by Simulated Annealing, Science, vol.220, no.4598, pp.671–680, 1983.

[19] Croes, G.A.: A method for Solving Traveling-Salesman Problems, Operation Research, vol.6, no.6, pp.791–812, 1958.

[20] Yokoyama, H., Goto, H.: Reduction and analysis of a max-plus linear system to a constraint satisfaction problem for mixed integer programming, American Journal of Operations Research, vol.7, 2017.

[21] 宮代 隆平: 整数計画ソルバー入門, Operation Research, vol.57, pp.183–189, 2012.

研究業績

研究業績 1：学術論文での論文掲載

タイトル：「Resolution of resource contentions in the CCPM-MPL using simulated annealing and genetic algorithm」

名称等：「American Journal of Operations Research」，査読付き　vol.6, no.6, pp.480-488

発行年月日：2016 年 11 月 21 日

著者：Hajime Yokoyama and Hiroyuki Goto

研究業績 2：学術論文での論文掲載

タイトル：「Reduction of a Max-Plus Linear System to Constraint Satisfaction Problems for Mixed Integer Programming」

名称等：「American Journal of Operations Research」，査読付き

Acceptance Notification：2017 年 2 月 10 日

著者：Hajime Yokoyama and Hiroyuki Goto

研究業績 3：国際学会での研究発表

タイトル：「Minimization of the Makespan a Project in the Critical Chain Project Management Framework using a Max-Plus Linear Representation」

開催地：Suntec City Convention Centre in Singapore

開催日：2015 年 12 月 6 日～9 日

学会名：「2015 IEEE International Conference on Industrial Engineering and Engineering Management」

発表者：Hiroyuki Goto and Hajime Yokoyama

研究業績 4：国際学会での研究発表

タイトル：「Resolution of Resource Contentions in the Critical Chain Project Management based on Simulated Annealing」

開催地：JW Marriot Hotel KL, Kuala Lumpur in Malaysia

開催日：2016 年 3 月 8 日～10 日

学会名：「Sixth International Conference on Industrial Engineering and Operations Management」

発表者：Hajime Yokoyama and Hiroyuki Goto

研究業績 5：国際学会での研究発表

タイトル：「Resolution of resource contentions in the CCPM using two metaheuristics」

開催地：Tsukuba International Congress Center, Tsukuba in Japan

開催日：2016 年 9 月 20 日～23 日

学会名：「SICE Annual Conference 2016」

発表者：Hajime Yokoyama and Hiroyuki Goto

研究業績 6：国内学会での研究発表

タイトル：「CCPM-MPL における Simulated Annealing に基づいた資源競合の解消方法」

開催地：滋賀県大津市 ウカルちゃんアリーナ（滋賀県立体育館）

開催日：2016 年 12 月 6 日～8 日

学会名：計測自動制御学会システム・情報部門学術講演会 2016

発表者：横山朔，五島洋行