2015 年度　修士論文

Max-min-plus scaling システムでのモデル予測制御に基づくスケジューリング方法
～滑走路のスケジューリングを例にして～

MODEL PREDICTIVE CONTROL-BASED SCHEDULER FOR MAX-MIN-PLUS SCALING SYSTEM

~A CASE OF A RUNWAY SCHEDULE~

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

Hosei University, Department of Science and Engineering,
Major of system engineering, Master degree program

14R6206

豊田（とよだ）　一裕（かずひろ）

Kazuhiro TOYODA

指導教員：　五島洋行

Abstract

In this thesis, we perform a scheduling based on model predictive control for max-min-plus-scaling systems. As an example of the scheduling the max-min-plus scaling system, there is a runway schedule. For the runway scheduling, we need max, plus and min operations, if we select the order of take-off and landing airplanes. However, if the order of take-off and landing airplanes is determined, the min operation is not necessary. If the min operation is not necessary, we can express the runway scheduling by a max-plus linear system. In this research, we determine the order of take-off and landing by a complete search and a genetic algorithm. In addition, we adjust the use time of the runway and the input time for the runway by the scheduling method based on model predictive control for the max-plus linear system. It is because the landing and take-off airplanes pass the different course for each one, we get a precedent matrix which has information of the airplane route. Moreover, we obtain a state and output equations by using the precedent matrix. In numerical simulation, we found that it is possible to make a schedule which satisfies the condition of the max-min-plus scaling system by the proposed method. However, for the complete search, if the number of the predictive horizon is increased, the computation time is very long. It is not practical. In the case study, the suitable solution was found by the genetic algorithm. The computation time of the genetic algorithm is faster than that of the complete search, if the predictive horizon is 7.

目次

1. はじめに

1.1. 研究背景

スケジューリングとは，時間軸情にタスクやジョブ，アクティビティを割り当てることである．ここでのタスクやジョブ，アクティビティとは，例えば工場での機械や作業，学校での授業，空港での滑走路や誘導路を示す．スケジューリングでよく用いられるのがガントチャート(Gantt chart)である．ガントチャートは時間軸上にタスクや資源などを割り振ってグラフ表現したものである．1950 年代になってスケジューリングの問題を解決するモデルが出始め，徐々に普及していった．スケジューリングには，割り当ての決定と順序づけの決定と 2 種類の意思決定が含まれている

[1]．本研究で扱うスケジューリング問題で考えると，限られた時間の割り当てと，システムを利用する順序の決定が含まれている．

スケジューリング問題では様々な制約条件を満たしている中で，最短時間やコスト最小化などの目標やポリシーを持ってスケジュールを求める．その中でも近年注目されている考え方が Just In Time をいう考え方である．Just In Time とは，必要なモノを必要なとき必要なだけ、という考え方であり，その目的は徹底的に無駄を取り除くことである．例えば，生産に必要な材料や部品を，必要なときに必要な分だけ購入して生産工程に届けることや，顧客が求める製品を顧客が必要とするタイミングに合わせ，注文の分だけ生産して納入することなどが挙げられる．生産の基本は需要に合わせて供給するということだが，実現することは難しい．実際には必要以上に生産して在庫を抱えてしまったり，需要の増減に合わせてラインや設備，人員配置の調整がうまくできなかったりすることが多い．Just In Time の考え方を用いることで，売り上げが伸びなくても利益を生み出すような経営効果に結び付けることも可能である[2]．

この Just In Time を実現するのに有功な手段の一つがモデル予測制御(Model Predictive Control: MPC)である．モデル予測制御は，制御工学の分野で開発され発展してきた手法である．未来の予測とその予測に基づく制御という考え方が合わさっており， 経営分野であまり議論されていなかった．制御工学の分野では過去の入出力のデータや，状態空間モデル，伝達関数などが使用されているが，モデルの記述法に指定がないため，柔軟性があり適用範囲が広い．主に石油化学産業の経営分野で使用され，現在では他の業界での生産管理などの適用例も増加している[3]．モデル予測制御の考え方は，望ましい目標値に出力を近づけるために，どのようなシステムの入力やシステム内部のパラメータを与えればよいかを求める手法である．現時点から数ステップ先の未来のシステムの状態を利用し，先読みした

情報から早めに入力やシステムパラメータの調整ができ，優れたシステムの入力や状態が得られる[4]．経営的な面から言及すると，優れた部品の入力や生産時間に調整することで，納期に出荷を近づけ Just In Time の実現の手助けとなると考えられる．

一方で，max-min-plus scaling システムと呼ばれるシステムモデルがある．Max-min-plus scaling システムとは，max 演算，min 演算，加法演算を用いて表現できるシステムモデルのことである．このシステムモデルで表現可能な例としては，プロジェクト管理や生産システム，空港の滑走路などがあげられる．しかし 2 章で言及する Max-plus 線形システムでのモデル予測制御は線形計画問題で解けるが，max-min-plus scaling システムでのモデル予測制御は非線形問題で非凸問題である．そこで本研究では max-min-plus scaling システムで表現可能なシステムに対して，max-plus 線形システムで表現する．その際に，システムの利用順序の決定が必要であり，この利用順序の決定を組み合わせ最適化の問題として考える．

組み合わせ最適化の問題は様々な場面で用いられている．多くの仕事を行う上で，様々な順序の中から一つの順序を選ぶ場合には，必ず組み合わせ最適化の問題が生じる．例えば，製造工場におけるタスクや滑走路が空くのを待っている航空機，銀行の窓口で順番待ちをしている顧客，計算センターで処理されるプログラム，家庭内の雑用など様々である．組み合わされる仕事の内容は異なるが，それらには共通して組み合わせ最適化問題があると考えられる[5]．

本研究では，max-min-plus scaling システムのスケジューリングを，モデル予測制御の理論に基づき行う．Max-min-plus scaling システムでのスケジューリング例として，空港の滑走路のスケジューリングが挙げられる．滑走路のスケジューリングでは，離陸機と着陸機の滑走路の使用順序を決めるときに max 演算と plus 演算に加えて min 演算が必要である．しかし離発着機の順序が決まっていれば，min 演算が必要でなくなる．Min 演算がなければ，max-plus 線形システムに基づいて滑走路のスケジューリングを表現できる．そこで本研究では，離着陸順を全数探索と遺伝的アルゴリズムで決め，滑走路の使用時間や滑走への侵入時刻は，max-plus 線形システムでのモデル予測制御に基づいたスケジューリング手法で調整する．

1.2. 先行研究

スケジューリングの研究ではさまざまなアプローチがなされてきた．その中でも本節では，モデル予測制御を用いたスケジューリングの研究を紹介する．また本研究では，空港の滑走路のスケジューリングを行うが，空港の滑走路でモデル予測制御を用いたスケジューリングの先行研究を紹介する．

Max-plus 線形システムでの MPC に基づいたスケジューリング

Max-plus 線形システムを用いた離散事象システムのモデリングとモデル予測制御に関する研究は，Masuda ら[6]や Schutter ら[7][8]によって盛んに研究報告がされてきた．その中でも，Goto[9]は，容量制約を持つ，反復的な離散事象システムでのモデル予測制御を行っている．文献によって提案された，制約条件を線形制約条件に緩和し，最適解の導出を簡単化することを用いて，最適な作業時間と外部からの入力時刻を求めるアプローチを採る検討を行う．この最適化問題を解く際，最初に生産にかかる時間に関しての最適化を行った後，外部からの部品の入力時刻に関する最適化を行っている．

Max-min-plus scaling システムでのモデル予測制御

Max-Min-Plus Scaling システム(MMPS)とは，max-plus 代数の延長で，max 演算，min 演算，加法演算を基本演算とする代数系を用いてモデル化されるシステムモデルのことである．主に離散事象システムで，事象の発生回数によるシステムの振る舞いをモデル化することに適しており，Schutter ら[10][11]によってモデル予測制御（MPC）への応用が行われている．その中で，Max-min-plus scaling システムでのモデル予測制御は非線形問題で非凸問題であることが紹介されている．

モデル予測制御を用いた着陸機のスケジューリング

モデル予測制御の理論を用いた滑走路のスケジューリングの研究に，Hu ら[12]がある．ここでは，モデル予測制御の手順の一つである後退ホライズン法を用いて，着陸機のスケジューリングを行っている．しかし滑走路のスケジューリングには，離陸機の考慮が必要である．そこで本研究では，離陸機と着陸機の両方を考慮した滑走路のスケジューリングを可能にするため，滑走路や誘導路を max-min-plus scaling システムを用いてモデル化し，モデル予測制御の理論に基づき行う．

1.3. 研究目的

本研究では，max-min-plus scaling システムのスケジューリングを，モデル予測制御の理論に基づき行う．Max-min-plus scaling システムでのスケジューリング例として，空港の滑走路のスケジューリングが挙げられる．滑走路のスケジューリングでは，離陸機と着陸機の滑走路の使用順序を決めるときに max 演算と plus 演算に加えて min 演算が必要である．先行研究のとおり，一般的に Max-min-plus scaling システムでのモデル予測制御は，非線形で非凸問題であることが知られている．しかし離発着機の順序が決まっていれば，min 演算が必要でなくなる．Min

演算がなければ，max-plus 線形システムに基づいて滑走路のスケジューリングを表現できる. そこで本研究では, 離着陸順を全数探索と遺伝的アルゴリズムで決め, 滑走路の使用時間や滑走への侵入時刻は，max-plus 線形システムでのモデル予測制御に基づいたスケジューリング手法で調整する. 着陸機と離陸機はそれぞれ違った経路をたどるため，経路の情報が入った先行関行列を導出する．さらに，この先行関係行列を用いて，状態予測式と出力予測式を導出する.

2. Max-plus 代数

Max-plus 代数とは max 演算と加法演算を基本演算とする代数系のことである．主に離散事象システムにおいて，事象の発生回数によって，システム内の状態がどのように変化していくかを記述することができる．特にスケジューリングを目的とする生産システムのモデリングや，モデル予測制御（MPC）の分野で様々な応用が試みられている．主なメリットは，システムの内部の状態を簡単な線形の式で表すことができる点である．

Max-plus 代数は Carre[13]によりアイディアが出され，Cohen ら[14]により一つの理論体系となった．

本章では，この Max-plus 代数の定義，使用例，スケジューリングとの関係について言及する．

2.1. 演算子の定義

実数全体を$\mathbb{R}$とすると max-plus 代数の定義域は$\mathbb{R}_{\max} = \mathbb{R} \cup (-\infty)$と定義される．Max-plus 代数の加算と乗算は以下のように定義する．ただしx，$y \in \mathbb{R}_{\max}$である．

$$x \oplus y = \max(x, y) \tag{1}$$
$$x \otimes y = x + y \tag{2}$$

通常の演算子において，加法演算では零元，×演算では単位元が定義されているが，max-plus 代数では，$\oplus$の単位元を$\varepsilon = -\infty$，$\otimes$の単位元を$e = 0$と定める． これにより以下の等式が成り立つ．

$$x \oplus \varepsilon = \varepsilon \oplus x = x \tag{3}$$
$$x \otimes e = e \otimes x = x \tag{4}$$
$$x \otimes \varepsilon = \varepsilon \otimes x = \varepsilon \tag{5}$$

また Min 演算は以下のように定義される．

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

Min 演算子∧は，max-min-plus scaling システムや max-plus 代数を用いた CCPM

法などで用いられる．Min 演算の単位元は∞であるがここでは特に定義はしない.

通常の計算で扱われる演算子での乗法と除法が, 加法と減法より計算の優先度が高いのと同様に，max-plus 代数で扱われる演算子の優先度は，plus 演算子⊗の方が max 演算子である⊕より計算の優先順位が高い.

例：

$$\begin{aligned} 5 \oplus 4 \otimes 2 &= 5 \oplus (4 \otimes 2) \\ &= \max(5, 4+2) \\ &= 6 \end{aligned} \tag{6}$$

またスカラー$r(\geq 0)$に対して, max-plus 代数での指数は以下のように定義される.

$$x^{\otimes r} = x \otimes x \otimes \cdots \otimes x = r \times x \tag{7}$$

以上のように定義した max-plus 代数は，以下の法則が成り立つ.

・ 交換法則

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

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

・ 結合法則

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

$$x \otimes (y \otimes z) = (x \otimes y) \otimes z \tag{11}$$

・ 分配法則

$$x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z) \tag{12}$$

これらのように通常の演算子と同じような性質を持つ一方で, 通常の演算子とは異なる以下の性質も持つ.

・ 加法のべき等

$$x \oplus x = x \tag{13}$$

・ 加法の逆元が存在しない.

$$x \oplus \hat{x} = \hat{x} \oplus x = \varepsilon \tag{14}$$

となる$\hat{\boldsymbol{x}}$が存在しない．そのため⊕の逆元は存在しない．

2.2. 行列・ベクトル演算

Max-plus での，通常の演算子でのシグマ演算(∑)に対応する演算子を，以下のように定義する．

$$\bigoplus_{k=1}^{n} x_k = x_1 \oplus x_2 \oplus \cdots \oplus x_n = \max(x_1, x_2, \cdots, x_n) \tag{15}$$

行列$\boldsymbol{X}$，$\boldsymbol{Y} \in \mathbb{R}_{\max}^{n\times m}$，　$\boldsymbol{Z} \in \mathbb{R}_{\max}^{m\times p}$，スカラー$\alpha \in \mathbb{R}_{\max}$とするとき，max-plus 代数の行列演算は以下のように定義される．

$$[\alpha \otimes \boldsymbol{X}]_{\mathrm{ij}} = \alpha \otimes [\boldsymbol{X}]_{ij} \tag{16}$$

$$[\boldsymbol{X} \oplus \boldsymbol{Z}]_{ij} = [\boldsymbol{X}]_{\boldsymbol{ij}} \oplus [\boldsymbol{Y}]_{\boldsymbol{ij}} \tag{17}$$

$$\begin{aligned} [\boldsymbol{X} \otimes \boldsymbol{Z}]_{ij} &= \bigoplus_{l=1}^{n} ([\boldsymbol{X}]_{il} \otimes [\boldsymbol{Z}]_{lj}) \\ &= \max_{l=1,2,\cdots,n} ([\boldsymbol{X}]_{il} + [\boldsymbol{Z}]_{lj}) \end{aligned} \tag{18}$$

Max-plus 代数でも通常の演算子の行列演算と同様に，$m \times n$行列と$n \times p$行列の積は$m \times p$行列である．さらに行列の積についても分配法則が成立する．

$$\begin{aligned} [(\boldsymbol{X} \oplus \boldsymbol{Y}) \otimes \boldsymbol{Z}]_{ij} &= \bigoplus_{l=1}^{n} \{([\boldsymbol{X}]_{il} \oplus [\boldsymbol{Y}]_{il}) \otimes [\boldsymbol{Z}]_{lj}\} \\ &= \bigoplus_{l=1}^{n} \{[\boldsymbol{X}]_{il} \otimes [\boldsymbol{Z}]_{lj} \oplus [\boldsymbol{Y}]_{il} \otimes [\boldsymbol{Z}]_{lj}\} \\ &= \bigoplus_{l=1}^{n} \{[\boldsymbol{X}]_{il} \otimes [\boldsymbol{Z}]_{lj}\} \oplus \bigoplus_{l=1}^{n} \{[\boldsymbol{Y}]_{il} \otimes [\boldsymbol{Z}]_{lj}\} \\ &= [\boldsymbol{X} \otimes \boldsymbol{Z}]_{ij} \oplus [\boldsymbol{Y} \otimes \boldsymbol{Z}]_{ij} \end{aligned} \tag{19}$$

Max-plus 代数の行列演算の加算と乗算の単位元はそれぞれ以下のように定義する．

$$\boldsymbol{\varepsilon} \in \mathbb{R}_{\max}^{m\times n} = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} & \cdots & \varepsilon_{1n} \\ \varepsilon_{21} & \varepsilon_{22} & \cdots & \varepsilon_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \varepsilon_{m1} & \varepsilon_{m2} & \cdots & \varepsilon_{mn} \end{bmatrix} \tag{20}$$

$$\boldsymbol{e} \in \mathbb{R}_{\max}^{m\times m} = \begin{cases} e\text{: diagonal elements} \\ \varepsilon\text{: otherwise} \end{cases} \tag{21}$$

以上のように Max-plus 代数の行列演算も，通常の演算子と同様に計算できるが，⊕の逆元が存在しないため，逆行列は定義できない.

2.3. Max-plus 代数の使用例

ここでは， 2.1 章，2.2 章で定義した max-plus 代数の具体的な計算例を紹介する.

スカラー演算

$5 \oplus 3 = \max(5,3) = 5$

$e \oplus -3 = \max(0,-3) = e$

$5 \otimes 3 = 5 + 3 = 8$

$\varepsilon \otimes 3 = -\infty + 3 = -\infty$

$5 \otimes (3 \oplus 7)) = (5 \otimes 3) \oplus (5 \otimes 7) = \max(5+3, 5+7) = 12$

$5 \wedge 3 = \min(5,3) = 3$

行列演算

$$\begin{bmatrix} 3 & 1 \\ e & \varepsilon \end{bmatrix} \oplus \begin{bmatrix} 4 & \varepsilon \\ -2 & e \end{bmatrix} = \begin{bmatrix} 3 \oplus 4 & 1 \oplus \varepsilon \\ e \oplus -2 & \varepsilon \oplus e \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ e & e \end{bmatrix}$$

$$\begin{bmatrix} 4 & \varepsilon & -1 \\ e & -2 & 1 \end{bmatrix} \otimes \begin{bmatrix} 2 & \varepsilon \\ -5 & 3 \\ e & 1 \end{bmatrix} = \begin{bmatrix} 4 \otimes 2 \oplus \varepsilon \otimes -5 \oplus -1 \otimes e & 4 \otimes \varepsilon \oplus \varepsilon \otimes 3 \oplus -1 \otimes 1 \\ e \otimes 2 \oplus -2 \otimes -5 \oplus 1 \otimes e & e \otimes \varepsilon \oplus -2 \otimes 3 \oplus 1 \otimes 1 \end{bmatrix}$$

$$= \begin{bmatrix} \max\{4+2, (-\infty)+(-5), -1+0\} & \max\{4+(-\infty), (-\infty)+3, -1+1\} \\ \max\{0+2, (-2)+(-5), 1+0\} & \max\{0+(-\infty), (-2)+3, 1+1\} \end{bmatrix}$$

$$= \begin{bmatrix} 6 & e \\ 2 & 2 \end{bmatrix}$$

2.4. Max-plus 代数とスケジューリングとの関係

本節では，簡単な PERT 図で示した日程計画問題を例にして，max-plus 代数での表現と関係について言及する．PERT とは，Performance Evaluation and Review Technique の頭文字をとったものであり，英国国防省がミサイル開発のプロジェクトの工期最適化を目的に開発された手法である. ノードの接続関係とアークの重みの情報を用いて定式化するものである. PERT 図は主に有向グラフを用いて書かれることが多く, アクティビティをノードにとり作業などをアークで表現する AOA(Activity on Arrow)と，作業をノードにとり，作業順序などをアークで表現する AON(Activity on Node)と 2 種類の表現技法がある．本研究では，ガントチャートとの類似性が強い AON を採用する．ガントチャート(Gantt Chart)とは，負荷やプロジェクトを時間軸上に割り当ててグラフ表現したものである.

Max-plus 代数とスケジューリングとの関係を示すため，簡単な生産工程を例に説明する. 表 1 は四つの作業を持つ生産工程の例である. 表 1 ように各作業には，生産時間，先行作業，外部からの部品の有無の情報が与えられている．作業 2 で生産した部品を作業 3 に送り，さらに作業 3 で生産した部品と作業 1 で生産した部品を同期して，作業 4 で組み合わせ出荷する生産システム例である.

表 1　3 つの作業を持つ生産工程の作業時間と先行作業，外部からの部品の有無

作業	作業時間	先行作業	外部からの部品
1	d_1	なし	あり
2	d_2	1	なし
3	d_3	なし	あり
4	d_4	2,3	なし

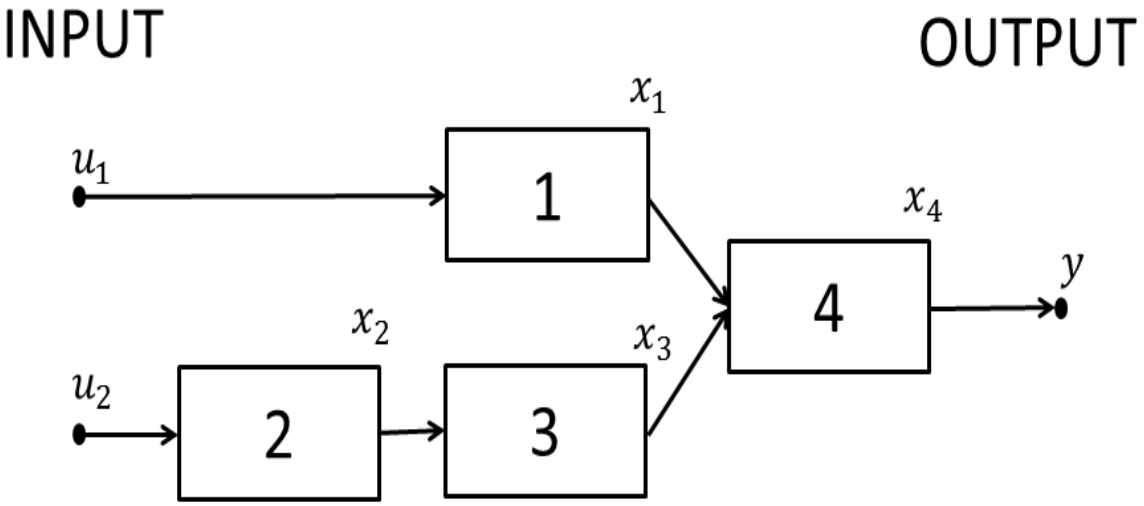

図 1 簡単なシステム例の PERT 図

表 1 の情報を基に図 1 のような PERT 図をつくることができる．作業iの完了時刻をx_i，それぞれの作業時間をd_i，外部からの入力時刻をu_1，u_2とする．このときの各工程の作業完了時刻は以下の式で表せる．

$$\begin{aligned} x_1 &= d_1 + u_1 = d_1 \otimes u_1 \\ x_2 &= d_2 + u_2 = d_2 \otimes u_2 \\ x_3 &= d_3 + x_2 = d_3 \otimes x_2 \\ x_4 &= d_4 + \max(x_1, x_3) = d_4 \otimes (x_1 \oplus x_3) \end{aligned} \tag{22}$$

右辺の x_iを消去し，u_iとd_iのみで行列表記すると以下のようになる．

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} d_1 \otimes u_1 \\ d_2 \otimes u_2 \\ d_3 \otimes d_2 \otimes u_2 \\ d_4 \otimes (d_1 \otimes u_1 \oplus d_3 \otimes d_2 \otimes u_2) \end{bmatrix} \tag{23}$$

よってこの生産工程の最早完了時刻は,

$$\begin{aligned} y &= x_3 \\ &= d_4 \otimes (d_1 \otimes u_1 \oplus d_3 \otimes d_2 \otimes u_2) \end{aligned} \tag{24}$$

と求めることができる．また，$\boldsymbol{x} = (x_1\ x_2\ x_3\ x_4)^T$，$\boldsymbol{u} = (u_1\ u_2)^T$とすると式(23)は以下のように表現できる．

$$\boldsymbol{x} = \boldsymbol{A} \otimes \boldsymbol{x} \oplus \boldsymbol{B} \otimes \boldsymbol{u} \tag{25}$$

$$\boldsymbol{y} = \boldsymbol{C} \otimes \boldsymbol{x} \tag{26}$$

ここで，

$$\boldsymbol{A} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & d_3 & \varepsilon & \varepsilon \\ d_4 & \varepsilon & d_4 & \varepsilon \end{bmatrix} \tag{27}$$

$$\boldsymbol{B} = \begin{bmatrix} d_1 & \varepsilon \\ \varepsilon & d_2 \\ \varepsilon & \varepsilon \\ \varepsilon & \varepsilon \end{bmatrix} \tag{28}$$

$$\boldsymbol{C} = [\varepsilon \quad \varepsilon \quad \varepsilon \quad e] \tag{29}$$

である．行列Aの対角成分はeで，作業iが先行作業jを持つ場合には，$[\boldsymbol{A}]_{ij}$に作業iの作業時間が入る．すなわち行列Aは重み付きの接続行列の一種であると考えることができる[15]．また行列Bは作業jが生産工程の始まりの場合はe，それ以外はεであるため入力ベクトルと考えることができる．

式(25)は線形連立方程式であるが，max-plus 代数では逆行列が定義されていないため，通常の演算子と同様に解くことができない．そこで右辺の $\boldsymbol{x}$に $\boldsymbol{x} = \boldsymbol{A} \otimes \boldsymbol{x} \oplus \boldsymbol{B} \otimes \boldsymbol{u}$を代入すると，

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{A} \otimes (\boldsymbol{A} \otimes \boldsymbol{x} \oplus \boldsymbol{B} \otimes \boldsymbol{u}) \oplus \boldsymbol{B} \otimes \boldsymbol{u} \\ &= \boldsymbol{A}^{\otimes 2} \otimes \boldsymbol{x} \oplus (e \oplus \boldsymbol{A}) \otimes \boldsymbol{B} \otimes \boldsymbol{u} \end{aligned} \tag{30}$$

となる．同様に右辺の $\boldsymbol{x}$に $\boldsymbol{x} = \boldsymbol{A} \otimes \boldsymbol{x} \oplus \boldsymbol{B} \otimes \boldsymbol{u}$を代入することを繰り返すと，式(30)は

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{A}^{\otimes 3} \otimes \boldsymbol{x} \oplus (e \oplus \boldsymbol{A}) \otimes \boldsymbol{B} \otimes \boldsymbol{u} \\ \cdots &= \boldsymbol{A}^{\otimes s} \otimes \boldsymbol{x} \oplus \left(e \oplus \boldsymbol{A} \oplus \boldsymbol{A}^{\otimes 2} \oplus \cdots \oplus \boldsymbol{A}^{\otimes s-1}\right) \otimes \boldsymbol{B} \otimes \boldsymbol{u} \end{aligned} \tag{31}$$

と変形できる．このとき，$\boldsymbol{A}^{\otimes s} = \boldsymbol{\varepsilon}$，$\boldsymbol{A}^{\otimes s-1} \neq \boldsymbol{\varepsilon}$となる$s$が存在する．$\boldsymbol{A}^*$を以下のように定義すると，

$$\boldsymbol{A}^* = \boldsymbol{e} \oplus \boldsymbol{A} \oplus \boldsymbol{A}^{\otimes 2} \oplus \cdots = \bigoplus_{l=0}^{\infty} A^{\otimes l} \tag{32}$$

式(30)は以下のようにシンプルにかくことができる．

$$\boldsymbol{x} = \boldsymbol{A}^* \otimes \boldsymbol{B} \otimes \boldsymbol{u} \tag{33}$$

ここで，$\boldsymbol{A}^*$はクリーネ閉包(Kleene star)と呼ばれ，$[\boldsymbol{A}]_{ij}$はjからiまでの最長の時間を示し，jからiまでパスが存在しない場合にはεが入る．また$[\boldsymbol{A}^{\otimes s}]_{ij}$は$j$から$i$までの最長の時間を示し，$j$から$i$までパスが存在しない場合には$\varepsilon$が入り，$j$から$i$までの$s$回状態を移動するときの可到達性を示している．図 1 の例の場合の$\boldsymbol{A}^*$は以下である．

$$\boldsymbol{A}^* = \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon \\ \varepsilon & d_3 \otimes d_4 & \varepsilon & e \end{bmatrix} \tag{34}$$

また，$\boldsymbol{A}^*$は以下のように表現することもできる．

$$\boldsymbol{A}^* = (\boldsymbol{P} \otimes \boldsymbol{F})^* \otimes \boldsymbol{P} \tag{35}$$

ここで，

$$\boldsymbol{F} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ e & \varepsilon & e & \varepsilon \end{bmatrix} \tag{36}$$

$$\boldsymbol{P} = \begin{bmatrix} d_1 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & d_2 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & d_3 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & d_4 \end{bmatrix} \tag{37}$$

$$(\boldsymbol{PF})^* = \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon \\ \varepsilon & d_3 & \varepsilon & e \end{bmatrix} \tag{38}$$

である．また，

$$(\boldsymbol{P} \otimes \boldsymbol{F})^* = \boldsymbol{e} \oplus (\boldsymbol{P} \otimes \boldsymbol{F}) \oplus (\boldsymbol{P} \otimes \boldsymbol{F})^2 \oplus \cdots \oplus (\boldsymbol{P} \otimes \boldsymbol{F})^l \tag{39}$$

である．このとき$\boldsymbol{F}$は先行関係を表している行列であることがわかる．以上のことから，max-plus 代数を用いると先行関係や作業時間などの情報だけで，システムの内部や出力時刻などが表現できる．

次に，図 1 に示した生産工程が繰り返し型の場合を考える．式(22)や式(25), (26)では，事象の発生は 1 回のみしか考慮されていない．システムの動的な挙動を調べるためには，同じ条件で事象を何度も発生させる必要がある．そこで初期状態からの事象の発生回数を表す独立変数を用いる．この独立変数のことをイベントカウンタと呼び，kとする．時間軸とか別に，イベントカウンタを用いて表すシステムのことを事象駆動システムといい，このシステムでは事象の発生にともなってシステムの内部に変化が生じる．

図 1 が事象駆動システムであるとき，kステップ目のから$k+1$ステップ目を見たシステムの状態は，式(22)(24)にイベントカウンタを導入して以下のように表せ

る．

$$
\begin{aligned}
x_1(k+1) &= \max\{x_1(k), u_1(k+1)\} + d_1(k+1) \\
&= \{x_1(k) \oplus u_1(k+1)\} \otimes d_1(k+1) \\
x_2(k+1) &= \max\{x_2(k), u_2(k+1)\} + d_2(k+1) \\
&= \{x_1(k) \oplus u_1(k+1)\} \otimes d_2(k+1) \qquad (40) \\
x_3(k+1) &= \max\{x_3(k), x_2(k+1)\} + d_3(k+1) \\
&= \{x_3(k) \oplus x_2(k+1)\} \otimes d_3(k+1) \\
x_3(k+1) &= \max\{x_4(k), x_1(k+1), x_3(k+1)\} + d_4(k+1) \\
&= \{x_4(k+1) \oplus x_1(k+1) \oplus x_3(k+1)\} \otimes d_4(k+1) \\
y(k+1) &= x_4(k+1) \qquad (41)
\end{aligned}
$$

さらに式(25)，(26)の行列表記にもイベントカウンタを導入すると，

$$\boldsymbol{x}(k+1) = \boldsymbol{A} \otimes \boldsymbol{x}(k) \oplus \boldsymbol{B} \otimes \boldsymbol{u}(k+1) \qquad (42)$$

$$\boldsymbol{y}(k+1) = \boldsymbol{C} \otimes \boldsymbol{x}(k+1) \qquad (43)$$

と表現できる．ここで，

$$\boldsymbol{A} = \qquad (44)$$

$$\begin{bmatrix} d_1(k+1) & \varepsilon & \varepsilon \\ \varepsilon & d_2(k+1) & \varepsilon \\ \varepsilon & d_2(k+1) \otimes d_3(k+1) & d_3(k+1) \\ d_1(k+1) \otimes d_4(k+1) & d_2(k+1) \otimes d_3(k+1) \otimes d_4(k+1) & d_3(k+1) \otimes \end{bmatrix} \qquad (45)$$

$$\boldsymbol{B} = \begin{bmatrix} d_1(k+1) & \varepsilon \\ \varepsilon & d_2(k+1) \\ \varepsilon & d_2(k+1) \otimes d_3(k+1) \\ d_1(k+1) \otimes d_4(k+1) & d_2(k+1) \otimes d_3(k+1) \otimes d_4(k+1) \end{bmatrix} \qquad (46)$$

$$\boldsymbol{C} = [\varepsilon \quad \varepsilon \quad \varepsilon \quad e]$$

である．$\boldsymbol{x}(k), \boldsymbol{u}(k), \boldsymbol{y}(k)$はそれぞれ状態ベクトル，入力ベクトル，出力ベクトル，$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$はそれぞれシステム行列，入力行列，出力行列と呼ばれる．以上のようにmax-plus 代数を用いて線形で表現できるシステムモデルのことを max-plus 線形システムという．この max-plus 線形システムでのモデル予測制御は線形計画問題を用いて解くことが可能である．

2.5. Max-min-plus-scaling システム

Max-min-plus scaling システムとは，max-plus 線形システムで用いた max 演算，加法演算に加えて，min 演算を用いて表現できるシステムモデルのことである．このシステムモデルで表現可能な例としては，プロジェクト管理や生産システ

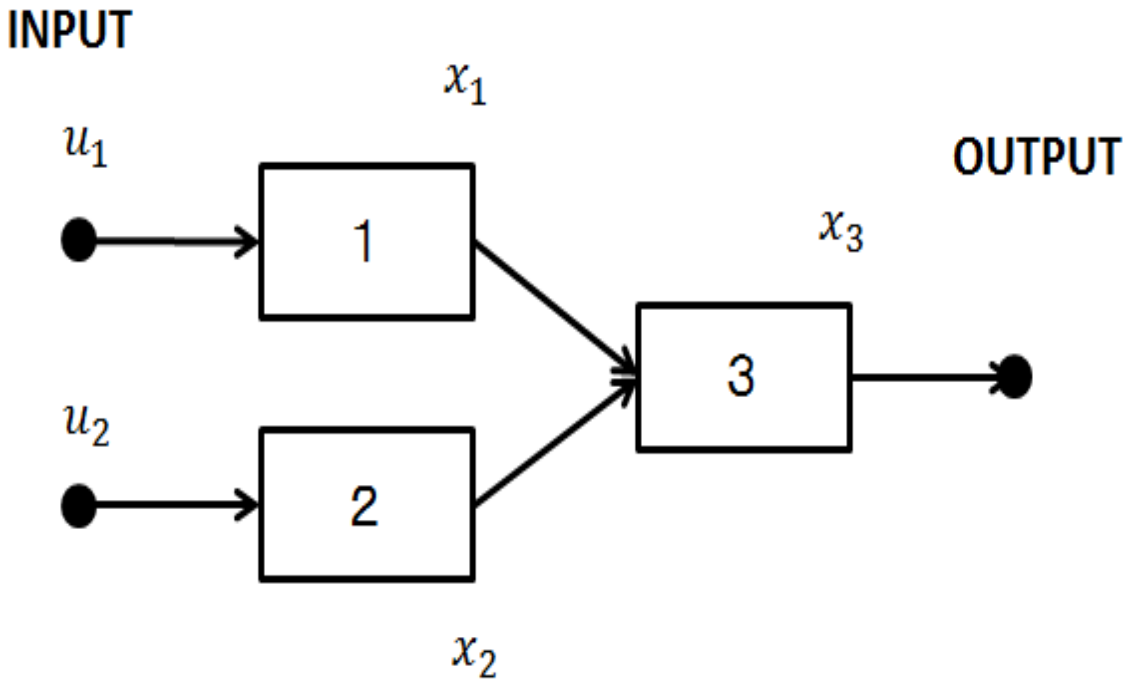

図 2　Max-min-plus-scaling システムの例

ム，空港の滑走路などがあげられる．

図 2 は max-min-plus scaling システムの例である．このシステムは task1 と task2 の完了時刻の早い方が，先に task3 を使用できるというシステムモデルである．これらの関係を max-plus 代数を用いて表現すると以下になる．

$$\begin{aligned} x_1(k+1) &= \max\{x_1(k), u_1(k+1)\} + d_1(k+1) \\ &= \{x_1(k) \oplus u_1(k+1)\} \otimes d_1(k+1) \end{aligned} \tag{47}$$

$$\begin{aligned} x_2(k+1) &= \max\{x_2(k), u_2(k+1)\} + d_2(k+1) \\ &= \{(x_1(k) \oplus u_1(k+1)\} \otimes d_2(k+1) \end{aligned} \tag{48}$$

$$\begin{aligned} x_3(k+1) &= \max[x_3(k+1), \min\{x_1(k+1), x_2(k+1)\}] + d_3(k+1) \\ &= [x_3(k+1) \oplus \{(k+1) \wedge x_2(k+1)\}] \otimes d_3(k+1) \end{aligned} \tag{49}$$

$$y(k+1) = x_3(k+1)$$

このように max-plus 代数に min 演算の記号を用いて表現可能である．Max-min-scaling システムも max-plus 線形システムと同様にイベントカウンタkを$k+1$にすることを繰り返すことで，何ステップ先のシステムの状態を数式で表

すことができる．しかし 2.4 で言及した max-plus 線形システムでのモデル予測制御の問題は線形計画問題で解けるが，max-min-plus scaling システムでのモデル予測制御は非線形問題で非凸問題である．そこで max-plus 線形システムを用いて，max-min-plus scaling システムの条件を満たしたスケジュールを得る方法を次章で述べる．

3. Max-min-plus scaling システムでの MPC

本章では，モデル予測制御に関して言及した後に，max-min-plus scaling システムでのモデル予測制御を行うための文字の定義，数式の導出，制御目標の設定を行う．

3.1. モデル予測制御

モデル予測制御の考え方は，望ましい目標値に出力を近づけるために，どのようなシステムの入力やシステム内部のパラメータを与えればよいかを求める手法である．現時点から数ステップ先の未来のシステムの状態を利用し，先読みした情報から早めに入力やシステムパラメータの調整ができ，優れたシステムの入力や状態が得られる[4]．経営的な面から言及すると，優れた部品の入力や生産時間に調整することで，納期に出荷を近づけ Just In Time の実現の手助けとなる．

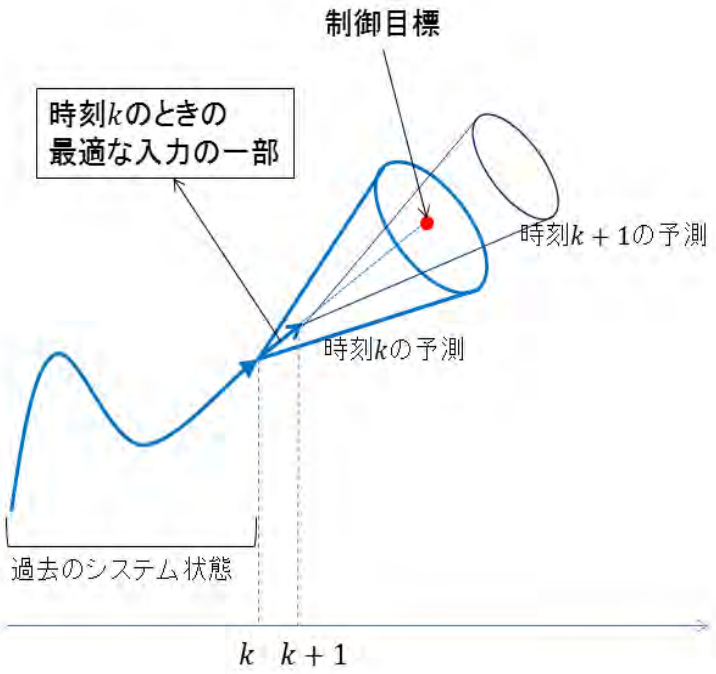

図 3　Receding Horizon 法のイメージ図

モデル予測制御は，制御工学の分野で開発され発展してきた手法である．未来の予測とその予測に基づく制御という考え方が合わさっており， 経営分野であまり議論されていなかった．制御工学の分野では，過去の入出力のデータや，状態空間モデル，伝達関数などが使用されているが，モデルの記述法に指定がないため，柔軟性があり適用範囲が広い．主に石油化学産業の経営分野で使用され，現在では他の業界での生産管理などの適用例も増加している[3].

モデル予測制御では入力をそのまま用いることはせず, 最適なシステムの入力の一部のみを用いる．このような手法を後退ホライズン(Receding Horizon)法という．図 3 はその後退ホライズン法のイメージを図に表したものである．

後退ホライズン法はよく，夜間の車の運転に例えられて説明される．夜間の運転では, ヘッドライトに照らされた範囲を見てこの先の運転のイメージをしながらハンドルの操作を行う．車が前進するとヘッドライトで照らされる範囲も前進し，新たに照らされる範囲の情報をもとにハンドルの操作を行う, ということを繰り返す．モデル予測制御では，ヘッドライトの範囲が予測ステップ数に相当し，運転のイメージをするところがシステムパラメータや入力の最適化, ハンドルの操作が実際のシステムへの入力にそれぞれ相当する. さらに前進したことで得られる新たな情報をもとに運転のイメージし直すところがレシーディング・ホライズン法に相当する．このように後退ホライズン法を用いることよって, 現時点での最新のシステムの状態の情報を用いてシステムの入力を計算でき, 常に最新の最適なシステムパラメータとシステム入力を使い続けることが可能となる．

モデル予測制御の基本的な手順は以下のように構成されている．

1. システムモデルを用いた状態予測式の導出
2. 予測式に基づいたシステムパラメータとシステムの入力の決定
3. 後退ホライズン法の適応

3.2. 文字の定義

本節では，本研究で扱う文字，ベクトル，行列の定義をする．

n：状態数
m：外部からの入力数
N_p：滑走路を使用する飛行機数
N：予測ステップ数
$\boldsymbol{d}(k) \in \mathbb{R}^{m \times n}_{\max}$:ステップ$k$での状態の時間のベクトル.
$\boldsymbol{u}(k) \in \mathbb{R}^{m}_{\max}$:ステップ$k$での外部からの入力時刻のベクトル.

$\boldsymbol{x}(k) \in \mathbb{R}_{\max}^{n}$:ステップ$\boldsymbol{k}$での各状態の終了時刻のベクトル.
$\boldsymbol{y}(k) \in \mathbb{R}_{\max}^{N}$:ステップ$\boldsymbol{k}$での外部への出力時刻のベクトル.
$\boldsymbol{r}(k) \in \mathbb{R}^{N}$:ステップ$\boldsymbol{k}$での納期のベクトル.
$\boldsymbol{P}_k \in \mathbb{R}_{\max}^{n\times n} := \mathrm{diag}\{d_i(k)\}$

$$[\boldsymbol{w}_k]_i \in \mathbb{R}_{\max}^{n} := \begin{cases} e: \text{状態}\boldsymbol{i}\text{を通る} \\ \varepsilon: \text{状態}\boldsymbol{i}\text{を通らない} \end{cases}$$

$\boldsymbol{W}_k = \mathrm{diag}(\boldsymbol{w_k})$

$$[\boldsymbol{F_0}]_{ij} \in \mathbb{R}_{\max}^{n\times n} := \begin{cases} e: \text{状態}\boldsymbol{i}\text{が先行状態}\boldsymbol{j}\text{を持つ} \\ \varepsilon: \text{状態}\boldsymbol{i}\text{が先行状態}\boldsymbol{j}\text{を持たない} \end{cases}$$

$$[\boldsymbol{B_0}]_{ij} \in \mathbb{R}_{\max}^{n\times m} := \begin{cases} e: \text{状態}\boldsymbol{i}\text{が外部からの入力}\boldsymbol{j}\text{を持つ} \\ \varepsilon: \text{状態}\boldsymbol{i}\text{が外部からの入力}\boldsymbol{j}\text{を持たない} \end{cases}$$

$$[\boldsymbol{C}_0]_i \in \mathbb{R}_{\max}^{n} := \begin{cases} e: \text{状態}\boldsymbol{i}\text{が外部への出力を持つ} \\ \varepsilon: \text{状態}\boldsymbol{i}\text{が外部への出力を持たない} \end{cases}$$

特に，$\boldsymbol{x}(k)$，$\boldsymbol{u}(k)$，$\boldsymbol{y}(k)$はそれぞれ状態ベクトル，入力ベクトル，出力ベクトルを示し，$\boldsymbol{F}_0$，$\boldsymbol{B}_0$，$\boldsymbol{C}_0$はそれぞれ接続行列，入力行列，出力行列を示している.

3.3. 予測式の導出

Max-min-plus scaling システムの状態予測式と出力予測式の導出の説明の前に，max-plus 線形システムの状態予測式と出力予測式の導出の説明を行う.

Max-plus 線形システムでの状態予測式と出力予測式

式(42)，(43)より，ステップ$\boldsymbol{k}$から見た，ステップ$\boldsymbol{k+1}$での各状態の終了時刻と発着時刻は以下の式で表せる.

$$\boldsymbol{x}(k+1) = \boldsymbol{A}_k \otimes \boldsymbol{x}(k) \oplus \boldsymbol{B}_{k+1} \otimes \boldsymbol{u}(k+1) \tag{50}$$
$$\boldsymbol{y}(k+1) = \boldsymbol{C}_0 \otimes \boldsymbol{x}(k+1) \tag{51}$$

ここで，$\boldsymbol{A}_k$，$\boldsymbol{B}_{k+1}$は以下のように決まる.

$$\boldsymbol{A}_k = \boldsymbol{P}_k \otimes (\boldsymbol{F}_0 \otimes \boldsymbol{P}_k)^* \tag{52}$$
$$\boldsymbol{B}_{k+1} = \boldsymbol{P}_k \otimes (\boldsymbol{F}_0 \otimes \boldsymbol{P}_k)^* \otimes \boldsymbol{B}_0 \tag{53}$$

ただし,

$$(\boldsymbol{F}_0 \otimes \boldsymbol{P}_k)^* = e \oplus (\boldsymbol{F}_0 \otimes \boldsymbol{P}_k) \oplus (\boldsymbol{F}_0 \otimes \boldsymbol{P}_k)^2 \oplus \cdots \oplus (\boldsymbol{F}_0 \otimes \boldsymbol{P}_k)^l \tag{54}$$

である．式(50)，(51)はそれぞれ状態方程式，出力予測式と呼ばれている．また式(50)，(51)は右辺のk を$k+1$ にすることで，ステップ$k+2$ の状態と出力を予測する式とすることができる．これを繰り返し行いNステップ先の状態を予測すると以下の式になる．

$$\begin{aligned}\boldsymbol{x}(k+N) = {} & \boldsymbol{A}_{k+N-1} \otimes \cdots \otimes \boldsymbol{A}_k \otimes \boldsymbol{x}(k) \\ & \oplus \boldsymbol{A}_{k+N-1} \otimes \cdots \otimes \boldsymbol{A}_k \otimes \boldsymbol{B}_k \otimes \boldsymbol{u}(k) \\ & \oplus \cdots \oplus \boldsymbol{B}_{k+N-1} \otimes \boldsymbol{u}(k+N)\end{aligned} \tag{55}$$

さらに，

$$\boldsymbol{X}(k+1) = [\boldsymbol{x}^T(k+1) \quad \boldsymbol{x}^T(k+2) \quad \cdots \quad \boldsymbol{x}^T(k+N)]^T \tag{56}$$

$$\boldsymbol{Y}(k+1) = [\boldsymbol{y}^T(k+1) \quad \boldsymbol{y}^T(k+2) \quad \cdots \quad \boldsymbol{y}^T(k+N)]^T \tag{57}$$

$$\boldsymbol{U}(k+1) = [\boldsymbol{u}^T(k+1) \quad \boldsymbol{u}^T(k+2) \quad \cdots \quad \boldsymbol{u}^T(k+N)]^T \tag{58}$$

$$\boldsymbol{\Gamma}_k = \begin{bmatrix} \boldsymbol{C}_0 \otimes \boldsymbol{A}_k \\ \boldsymbol{C}_0 \otimes \boldsymbol{A}_{k+1} \otimes \boldsymbol{A}_k \\ \vdots \\ \boldsymbol{C}_0 \otimes \boldsymbol{A}_{k+N-1} \otimes \cdots \otimes \boldsymbol{A}_k \end{bmatrix} \tag{59}$$

$$\boldsymbol{\Delta}_{k+1} = \begin{bmatrix} \boldsymbol{C}_0 \otimes \boldsymbol{B}_{k+1} & \cdots & \varepsilon \\ \boldsymbol{C}_0 \otimes \boldsymbol{A}_{k+1} \otimes \boldsymbol{B}_{k+1} & & \vdots \\ \vdots & \ddots & \varepsilon \\ \boldsymbol{C}_0 \otimes \boldsymbol{A}_{k+N-1} \otimes \cdots \otimes \boldsymbol{A}_{k+1} \otimes \boldsymbol{B}_{k+1} & \cdots & \boldsymbol{C}_0 \otimes \boldsymbol{B}_{k+N-1} \end{bmatrix} \tag{60}$$

を用いると各ステップの状態と出力予測式は，

$$\boldsymbol{Y}(k+1) = \boldsymbol{\Gamma}_k \otimes \boldsymbol{x}(k) \oplus \boldsymbol{\Delta}_{k+1} \otimes \boldsymbol{U}(k+1) \tag{61}$$

と表現できる．

Max-min-plus scaling システムを対象にした状態予測式と出力予測式

max-min-plus scaling システムを考える．ここでは，図 2 の max-min-plus scaling システムを，空港の滑走路のスケジューリングを例に，task1 を誘導路，task2 を着陸機が滑走路に進入する経路，task3 を滑走路として説明する．滑走路は，空から滑走路へ侵入し着陸する機体と，空港のゲートを離れて滑走路へ続く誘

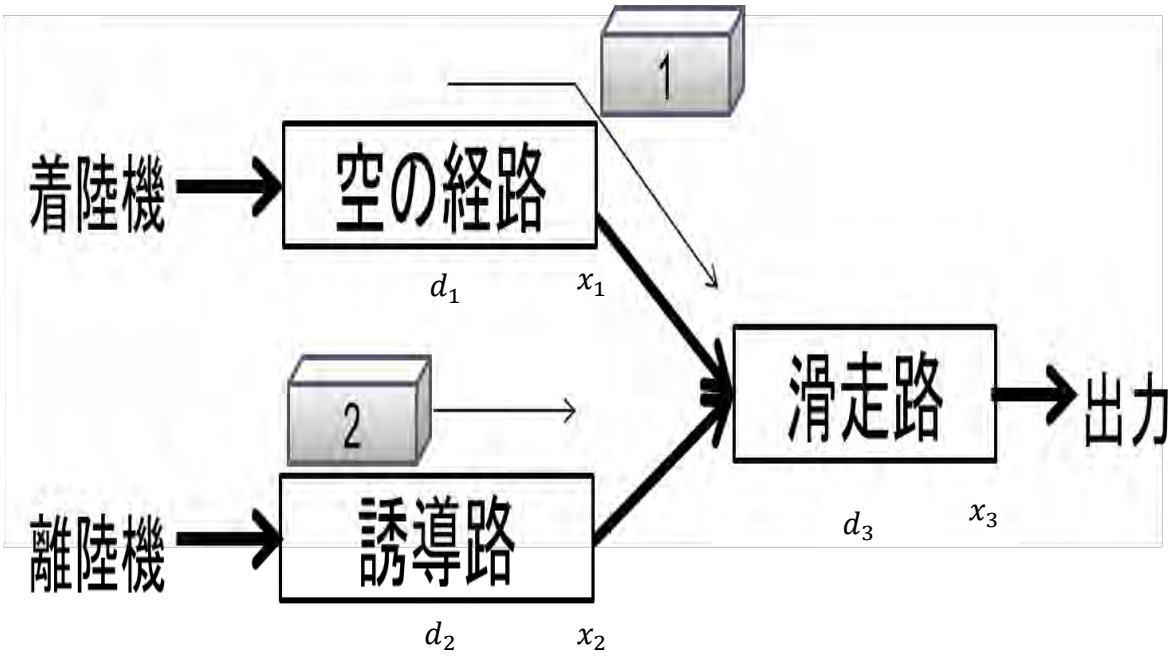

図 4　簡単な滑走路のモデル図

導路を通って滑走路行く機体が使用する. 先に滑走路に到着する機体から滑走路を使用できる．仮に，図 4 のように着陸機である機体 1 が，離陸機である機体 2 より先に滑走路へ到着する場合，

$$\boldsymbol{A'}_1 = \begin{bmatrix} d_1(1) & \varepsilon & \varepsilon \\ \varepsilon & 0 & \varepsilon \\ d_1(1) \otimes d_3(1) & \varepsilon & d_3(1) \end{bmatrix}, \quad \boldsymbol{B'}_1 = \begin{bmatrix} d_1 & \varepsilon \\ \varepsilon & \varepsilon \\ d_1(1) \otimes d_3(1) & \varepsilon \end{bmatrix} \tag{62}$$

$$\boldsymbol{A'}_2 = \begin{bmatrix} 0 & \varepsilon & \varepsilon \\ \varepsilon & d_2(2) & \varepsilon \\ \varepsilon & d_2(2) \otimes d_3(2) & d_3(2) \end{bmatrix}, \quad \boldsymbol{B'}_2 = \begin{bmatrix} \varepsilon & \varepsilon \\ \varepsilon & d_2(2) \\ \varepsilon & d_2(2) \otimes d_3(2) \end{bmatrix} \tag{63}$$

とすると状態予測式は,

$$\boldsymbol{x}(1) = \boldsymbol{A'}_1 \boldsymbol{x}(0) \oplus \boldsymbol{B'}_1 \boldsymbol{u}(1) \tag{64}$$

$$\boldsymbol{x}(2) = \boldsymbol{A'}_2 \boldsymbol{x}(1) \oplus \boldsymbol{B'}_2 \boldsymbol{u}(2) \tag{65}$$

と表すことができる．ここで$\boldsymbol{x}(0)$はシステムの初期状態を表し，空の状態である$\boldsymbol{x}(0) = [\varepsilon \quad \varepsilon \quad \varepsilon]^T$とすることが多い．機体 1，2 それぞれに対して，その状態を通る場合をe，通らない場合εとする，通り道ベクトル$\boldsymbol{w_1}$，$\boldsymbol{w_2}$は以下である．

$$\boldsymbol{w_1} = [e \quad \varepsilon \quad \varepsilon] \tag{66}$$

$$\boldsymbol{w_2} = [\varepsilon \quad e \quad e] \tag{67}$$

さらに，

$$\boldsymbol{W}_1 = \mathrm{diag}(\boldsymbol{w}_1) \tag{68}$$

$$\boldsymbol{W}_2 = \mathrm{diag}(\boldsymbol{w}_2) \tag{69}$$

とすると，

$$\boldsymbol{F'}_1 = \boldsymbol{W}_1 \otimes \boldsymbol{F}_0 \otimes \boldsymbol{W}_1 \tag{70}$$

$$\boldsymbol{F'}_2 = \boldsymbol{W}_2 \otimes \boldsymbol{F}_0 \otimes \boldsymbol{W}_2 \tag{71}$$

である．$\boldsymbol{F'}_1$，$\boldsymbol{F'}_2$は通り道のみ情報が入った先行関係を表す行列である．(64)，(65)の$\boldsymbol{A'}_1$，$\boldsymbol{A'}_2$は以下のように表せる．

$$\boldsymbol{A}_1 = \boldsymbol{P}_1 \otimes (\boldsymbol{F'}_1 \otimes \boldsymbol{P}_1)^* \tag{72}$$

$$\boldsymbol{A}_2 = \boldsymbol{P}_2 \otimes (\boldsymbol{F'}_2 \otimes \boldsymbol{P}_2)^* \tag{73}$$

ここで，

$$\boldsymbol{P}_1 = \mathrm{diag}[d_1(1) \quad 0 \quad d_3(1)] \tag{74}$$

$$\boldsymbol{P}_2 = \mathrm{diag}[0 \quad d_2(2) \quad d_3(2)] \tag{75}$$

また$\boldsymbol{B'}_1$，$\boldsymbol{B'}_2$は以下のように表せる．

$$\boldsymbol{B'}_1 = \boldsymbol{P}_1 \otimes (\boldsymbol{F'}_1 \otimes \boldsymbol{P}_1)^* \otimes \boldsymbol{W}_1 \otimes \boldsymbol{B}_0 \tag{76}$$

$$\boldsymbol{B'}_2 = \boldsymbol{P}_2 \otimes (\boldsymbol{F'}_2 \otimes \boldsymbol{P}_2)^* \otimes \boldsymbol{W}_2 \otimes \boldsymbol{B}_0 \tag{77}$$

さらに機体 1，2 に対する出力予測式は，

$$\boldsymbol{y}(1) = \boldsymbol{C'}_1 \otimes \boldsymbol{x}(1) \tag{78}$$

$$\boldsymbol{y}(2) = \boldsymbol{C'}_2 \otimes \boldsymbol{x}(2) \tag{79}$$

である．ここで，

$$\boldsymbol{C'}_1 = \boldsymbol{C}_0 \otimes \boldsymbol{W}_1 \tag{80}$$

$$\boldsymbol{C'}_2 = \boldsymbol{C}_0 \otimes \boldsymbol{W}_2 \tag{81}$$

である．

これらを一般的に表す．イベントカウンタkのとき，通り道のみの情報が入った先行関係の行列は以下である．

$$\boldsymbol{F'}_k = \boldsymbol{W}_k \otimes \boldsymbol{F}_0 \otimes \boldsymbol{W}_k \tag{82}$$

このとき,

$$(\boldsymbol{F'}_k \otimes \boldsymbol{P}_k)^* = \boldsymbol{e} \oplus (\boldsymbol{F'}_k \otimes \boldsymbol{P}_k) \oplus (\boldsymbol{F'}_k \otimes \boldsymbol{P}_k)^2 \oplus \cdots \oplus (\boldsymbol{F'}_k \otimes \boldsymbol{P}_k)^l \tag{83}$$

$$\boldsymbol{A'}_k = \boldsymbol{P}_k \otimes (\boldsymbol{F'}_k \otimes \boldsymbol{P}_k)^* \tag{84}$$

$$\boldsymbol{B'}_{k+1} = \boldsymbol{P}_k \otimes (\boldsymbol{F'}_k \otimes \boldsymbol{P}_k)^* \otimes \boldsymbol{W}_k \otimes \boldsymbol{B}_0 \tag{85}$$

$$\boldsymbol{C'}_k = \boldsymbol{C}_0 \otimes \boldsymbol{W}_k \tag{86}$$

とすると，状態予測式，出力予測式は以下のように表せる．

$$\boldsymbol{x}(k+1) = \boldsymbol{A'}_k \otimes \boldsymbol{x}(k) \oplus \boldsymbol{B'}_{k+1} \otimes \boldsymbol{u}(k+1) \tag{87}$$

$$\boldsymbol{y}(k+1) = \boldsymbol{C'}_k \otimes \boldsymbol{x}(k+1) \tag{88}$$

そして，max-plus 線形システムのときと同様に式は右辺のk を$k+1$ にすることを繰り返し行いNステップ先の状態を予測すると以下の式になる．

$$\begin{aligned}\boldsymbol{x}(k+N) = {} & \boldsymbol{A'}_{k+N-1} \otimes \cdots \otimes \boldsymbol{A'}_k \otimes \boldsymbol{x}(k) \\ & \oplus \boldsymbol{A'}_{k+N-1} \otimes \cdots \otimes \boldsymbol{A'}_k \otimes \boldsymbol{B'}_k \otimes \boldsymbol{u}(k) \\ & \oplus \cdots \oplus \boldsymbol{B'}_{k+N-1} \otimes \boldsymbol{u}(k+N)\end{aligned} \tag{89}$$

さらに,

$$\boldsymbol{\Gamma'}_k = \begin{bmatrix} \boldsymbol{C'}_{k+1} \otimes \boldsymbol{A'}_k \\ \boldsymbol{C'}_{k+2} \otimes \boldsymbol{A'}_{k+1} \otimes \boldsymbol{A'}_k \\ \vdots \\ \boldsymbol{C'}_{k+N} \otimes \boldsymbol{A'}_{k+N-1} \otimes \cdots \otimes \boldsymbol{A'}_k \end{bmatrix} \tag{90}$$

$$\boldsymbol{\Delta'}_{k+1} = \begin{bmatrix} \boldsymbol{C'}_{k+1} \otimes \boldsymbol{B'}_{k+1} & \cdots & \varepsilon \\ \boldsymbol{C'}_{k+2} \otimes \boldsymbol{A'}_{k+1} \otimes \boldsymbol{B'}_{k+1} & & \vdots \\ \vdots & \ddots & \varepsilon \\ \boldsymbol{C'}_{k+N} \otimes \boldsymbol{A'}_{k+N-1} \otimes \cdots \otimes \boldsymbol{A'}_{k+1} \otimes \boldsymbol{B'}_{k+1} & \cdots & \boldsymbol{C'}_{k+N} \otimes \boldsymbol{B'}_{k+N-1} \end{bmatrix} \tag{91}$$

を用いると各ステップの状態と出力予測式は,

$$\boldsymbol{Y}(k+1) = \boldsymbol{\Gamma'}_k \otimes \boldsymbol{x}(k) \oplus \boldsymbol{\Delta'}_{k+1} \otimes \boldsymbol{U}(k+1) \tag{92}$$

である．

このように max-min-plus scaling システムを対象にして，max-plus 線形システムでの表現をしたが，これは離発着の順序が決まっていることが前提条件にしている．一般的に離発着順は決まっておらず，別で決定する必要がある．この離発着の順序の決定に関しては 3.5，3.6 で述べる．

3.4. 制御目標

本節では，3.3 で導出した予測式を用いて，どのようなポリシーに従ったスケジュールにするか, そのためにどのような最適化問題を解くかを空港の滑走路と誘導路のスケジューリングを例にして説明する．

各ステップでの離陸時刻とゲートへの到着時刻に対しての遅れを，

$$\mu(k+l) = \max[\{y(k+l) - r(k+l)\}, 0] \tag{93}$$

とする. 各ステップで出力予測時刻が発着時刻と遅れの和より小さくならなければいけない．そのため，ステップ$k+l$ に関しての制約条件は式(92)を用いて以下のように与えられる．

$$\boldsymbol{\Gamma'}_k \otimes \boldsymbol{x}(k) \oplus \boldsymbol{\Delta'}_{k+1} \otimes \boldsymbol{U}(k+1) \leq \boldsymbol{R}(k+1) + \overline{\boldsymbol{\mu}} \tag{94}$$

ここで,

$$\boldsymbol{R}(k+1) = [\boldsymbol{r}^T(k+1) \quad \boldsymbol{r}^T(k+2) \quad \cdots \quad \boldsymbol{r}^T(k+N)]^T \tag{95}$$

$$\overline{\boldsymbol{\mu}}(k+1) = [\boldsymbol{\mu}^T(k+1) \quad \boldsymbol{\mu}^T(k+2) \quad \cdots \quad \boldsymbol{\mu}^T(k+N)]^T \tag{96}$$

である．また各状態の調整可能範囲として以下の制約条件を与える．

$$0 < d_{\min} \leq d_i(k+l) \leq d_{\max} \quad (0 \leq i \leq n) \tag{97}$$

本研究では，より無駄の少ないスケジュールを組むため，誘導路の使用時間$\boldsymbol{d}_i$と誘導路への入力時刻$\boldsymbol{u}_k$の最適化を同時に行う．遅れを出さず，誘導路の使用時間$\boldsymbol{d}$と外部からの入力時刻$\boldsymbol{u}$を大きくすることを考える．これらを定式化すると以下の式になる．

$$J = \alpha \sum_{j=1}^{N} \mu(k+j) + \beta \sum_{j=1}^{N} \sum_{i=1,2}^{n} d_i \quad (k+j) - \gamma \sum_{j=1}^{N} \sum_{i=1,2}^{m} u_i \quad (k+j) \tag{98}$$

ここで，α は遅れに対するペナルティを，β，$\gamma\ (\geq 0)$ はそれぞれ滑走路や誘導路の使用時間と外部入力の調整パラメータである．

線形計画問題で$\boldsymbol{d}(k+l)$，$\boldsymbol{u}(k+l)$，$\boldsymbol{\mu}(k+l)$について同時に最適化する最適化問題は，以下のように表せる．

$$J \rightarrow \min \quad \text{s.t.} \quad \text{Eqs. (94), (97)} \tag{99}$$

3.5. 全数探索を用いた離発着順の決定

滑走路のスケジューリングでは, 離陸機と着陸機の滑走路の使用順序を決めるときに max 演算と plus 演算に加えて min 演算が必要である．離発着機の順序が決まっていれば，min 演算が必要でなくなる．Min 演算がなければ，max-plus 線形システムとして滑走路のスケジューリングを表現できる．そこで本研究では，離着陸順を全数探索で，滑走路の使用時間や滑走への侵入時刻は，前節で述べた max-plus 線形システムでのモデル予測制御に基づいたスケジューリング手法で調整する．全数探索は，離発着順のすべてのパターンを調べて，その中で最も式(98)の値が小さいときを最適解とする手法である．予測ステップないに含まれる離発着機の，ありえる全ての順序を調べるため，組合せの数は予測ステップ数が$N = 5$のときは$5! = 120$通り，$N = 8$のときは　$8! = 40{,}320$通り，$N = 10$のときは$10! = 3{,}628{,}800$通りである．確実に最適な離発着順が求まるが，計算時間が膨大にかかる恐れがある．

3.6. 遺伝的アルゴリズムを用いた離発着順の決定

全数探索を用いて厳密解を得るには，非常に長い計算時間が必要である．モデル予測制御は，未来の予測をしながら，1 ステップ先の入力やシステムパラメータを決めるものであり，計算時間が大きいとその特徴に適していない．そのため良い解を比較的短時間で求める必要がある．そこで本研究では，遺伝的アルゴリズム(GA)を用いて近似解を求める．

GA とは生物の進化の原理から考えられたアルゴリズムである．最適化が求まる保証はないが，良い近似解が得られることが多いことから，さまざまな最適化問題に適用されている．GA は生成・テスト(Generate-and-Test)型のアルゴリズムであり，遺伝的操作を用いて集団を進化させる[16].

1. 選択：個体iに対して，その適応度に応じて次世代に残す子の数を増減させる．
2. 突然変異：図 5 のように各個体に対して，突然変異率で対立遺伝子と入れ替える．
3. 交叉:図 6のように個体の集合内の 2 個ずつをランダムに組み合わせ，交叉率で二つの個体の遺伝子列を部分的に交換する．図 6 は一点交叉と呼ばれる交叉方法である．

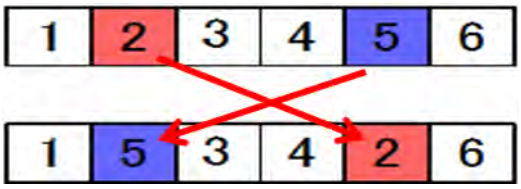

図 5 突然変異の例

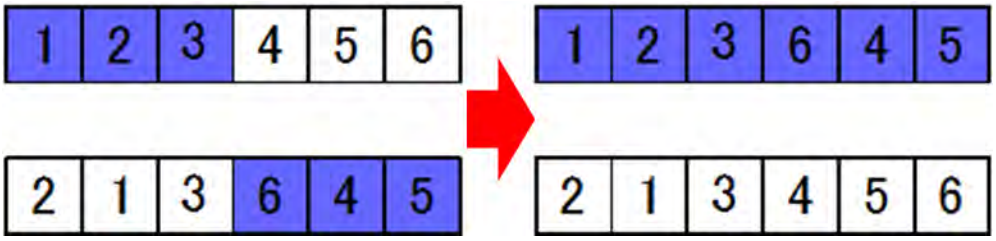

図 6　交叉の例

これらの 3 種類の遺伝的操作を用いて，一般的な GA では以下のようなアルゴリズムで構成されている．なお，一般的には個体集合は$P(t)$と表現されるが，本研究では，3.2 で定義した状態の時間が対角成分に入っている変数$\boldsymbol{P}_k$と区別するため，$P_{ga}(t)$と表記する．

ステップ 0：初期化

ランダムにM個の個体を生成して，初期個体集合$P_{ga}(0)$とする．

ステップ 1：評価

個体集合$P_{ga}(t)$の内の個体の適応度Jを計算する．

ステップ 2：選択

個体集合$P_{ga}(t)$内から次の世代に残す個体を選択する．選択された個体集合を$P'_{ga}(t)$とする．選択方法については以下で言及する．

ステップ 3：交叉

選択された個体集合$P'_{ga}(t)$ないからランダムに 2 つの個体を選び，交叉方法に従って新たに個体集合$P''_{ga}(t)$を作成する．交叉方法については以下で言及する．

ステップ 4：突然変異

$P''_{ga}(t)$内から突然変異率Mに応じて個体の一部を変化させる．

ステップ 5：終了判定

終了条件を見満たしていれば終了，満たしていなければ$P_{ga}(t+1) = P''_{ga}(t)$としてステップ 1 から繰り返す．

選択方法

どの個体をどれくらい残すかを選択する必要がある. この選択方法は様々な方法が提案されている. その中でも代表的な適応度比例戦略, ランク戦略, エリート戦略の三種類の選択方法の説明する.

適応度比例戦略

評価値のよい個体が残りやすくなるように各個体の生存確率を設定する方法である. 個体iの選択確率をp_i, 評価値をg_iとすると, 以下の式を用いて選択確率を求めることが多い.

$$p_i = \frac{g_i}{\sum g_i}$$

評価値の分散が大きい場合, 評価値の悪い個体がほとんど生存せず, その結果解が改善されないまま収束する場合がある.

ランク戦略

評価値に応じたランクを個体に与え, あらかじめ各ランクに対して定めた確率で子孫を残すようにする方法である. 評価値のばらつきが選択確率の影響を受けない特徴を持つ.

エリート戦略

評価値のよい個体を一定数次世代に残す手法である. この方法を採用すると, 良い解が突然変異や交叉で破壊されないという利点がある. しかし, エリートの個体の遺伝子が急速に広がるため, 局所解に陥る危険もある. この手法は, ほかの選択方法と組み合わせて用いられることが多い.

これら三種類の選択方法のうち, 本研究ではランク戦略を採用する. 一番良い個体を必ず次の世代に残すように設定することで, エリート戦略の考え方も採用することになる.

交叉方法

交叉は実行不能解を生成しないような適切な方法を用いる必要がある. 本研究では順序交叉, 周期交叉を用いて, システムを利用する順序を決定する. これらの手法は, 主に巡回セールスマン問題で用いられている手法である
[17]. これらの交叉方法の説明に必要な二点交叉についても言及する.

・ 二点交叉

二つの順序でランダムに 2 点交叉点を選び, 交差点の内側を一つの順序を挿入し, 残りの部分をもう一つの順序から挿入する手法である．この手法では，実行不可能な順序が生まれてしまう．そこで，主に巡回セールスマン問題で用いられている部分一致交叉，順序交叉，周期交叉の 3 種類の交叉方法を用いて，システムを利用する順序を決定する．

・ 順序交叉

順序交叉（Order Crossover, 順序交叉）は，一つの親から順列の一部を受け継ぎ, 残りの部分をもう一つの親の順列の位置を受け継ぐ方法である．二点交叉を基本的な考え方とするが, 正当な順序を維持しようとする手続きでおやの出現順序をなるべく保存しようとする交叉方法である. 順序交叉のアルゴリズムは以下である.

1. 親を 2 点で切断する.

$$p_1 = (1\,2\,3 |\ 4\,5\,6 |\ 7\,8)$$
$$p_2 = (8\,4\,7 |\ 2\,1\,6 |\ 5\,3)$$

2. 切断した中央部を子に受け継ぐ.

$$c_1 = (\mathrm{X\,X\,X} |\ 4\,5\,6 |\ \mathrm{X\,X})$$
$$c_2 = (\mathrm{X\,X\,X} |\ 2\,1\,6 |\ \mathrm{X\,X})$$

3. 中央部を受け継いだ親とは別の親の順列の後ろの切断点から開始し，要素の系列を作成する. 最後の要素になったら先頭から繰り返す. ただし, すでに子に含まれている値は含まない.

   c_1 に対して $3-8-7-2-1$

   c_2 に対して $7-8-3-4-5$

4. 3 で求めた系列順に，後ろの切断点から順に残りの要素を埋める.

$$c_1 = (7\,2\,1 |\ 4\,5\,6 |\ 3\,8)$$
$$c_2 = (3\,4\,5 |\ 2\,1\,6 |\ 7\,8)$$

・ 周期交叉

周期交叉（Cycle Crossover, 周期交叉）は親の順列の相対位置を受け継ぐ方法である．周期交叉のアルゴリズムは以下である.

$$p_1 = (1\,2\,3\,4\,5\,6\,7\,8),\ p_2 = (8\,4\,7\,2\,1\,6\,5\,3)$$

について考える.

1. p_1の最初の要素 1 から開始して，p_2の 1 番目の要素は 8 なので，c_1に 8 を受け継ぐ．p_2の 8 番目の要素は 3 なので，c_1に 3 を受け継ぐ．これらの手順を繰り返すとc_1は以下のように得られる．

$$c_1 = (1\ \mathrm{X}\ 3\ \mathrm{X}\ 5\ \mathrm{X}\ 7\ 8)$$

2. 最後に c_1に入った値は 5 であるが，この位置に対応する p_2の値は 1 である．これ以上は進めないため，残りの要素はp_2からそのまま受け継ぐ．

$$c_1 = (1\ 4\ 3\ 2\ 5\ 6\ 7\ 8)$$

3. p_2に対しても同様の手順を踏むとc_2は以下のように得られる．

$$c_2 = (8\ 2\ 7\ 4\ 1\ 6\ 5\ 3)$$

また，GA で用いる文字は以下のように定義する．

M：個体数

p_m：突然変異率

p_c：交叉率

I：終了までの世代数

本研究での GA を用いた，離発着順の決定のアルゴリズムは以下のとおりである．

0　Nステップ先までの離発着順をランダムにM個，初期の個体として生成し$P_{ga}(0)$とする．

1　$t = t + 1$とする．

2　M個の離発着順に対して，式(99)に示したモデル予測制御に基づいた線形計画問題を解く．本研究では式(99)の目的関数Jの値を GA の評価値として扱い，この値が最小のときを最適解とする．

3　選択方法のランク戦略を採用し，式(99)の目的関数Jの値が小さい順にランク付けをする．あらかじめ，それぞれのランクに設定した確率で次の世代に残す個体を決める．

4　選択された個体の集合を$P'_{ga}(t)$とする．

5　確率p_cで交叉，順序交叉または，周期交叉を用いて個体を交叉させる．

6　4 で交叉されなかった個体を，確率p_mで突然変異させる．具体的には，選ばれた個体の離発着順をランダムに 2 か所入れ替える．

7　5 と 6 で交叉や突然変異をさせたものを$P''_{ga}(t)$とする．

8　終了判定を行う．I回繰り返していれば終了し，I回繰り返していなければ 9 へ進む．

9　$P_{ga}(t+1) = P''_{ga}(t)$として 1 へ戻る．

これらの本研究での GA を用いた, 離発着順の決定のアルゴリズムを図で示したものが図 7 である.

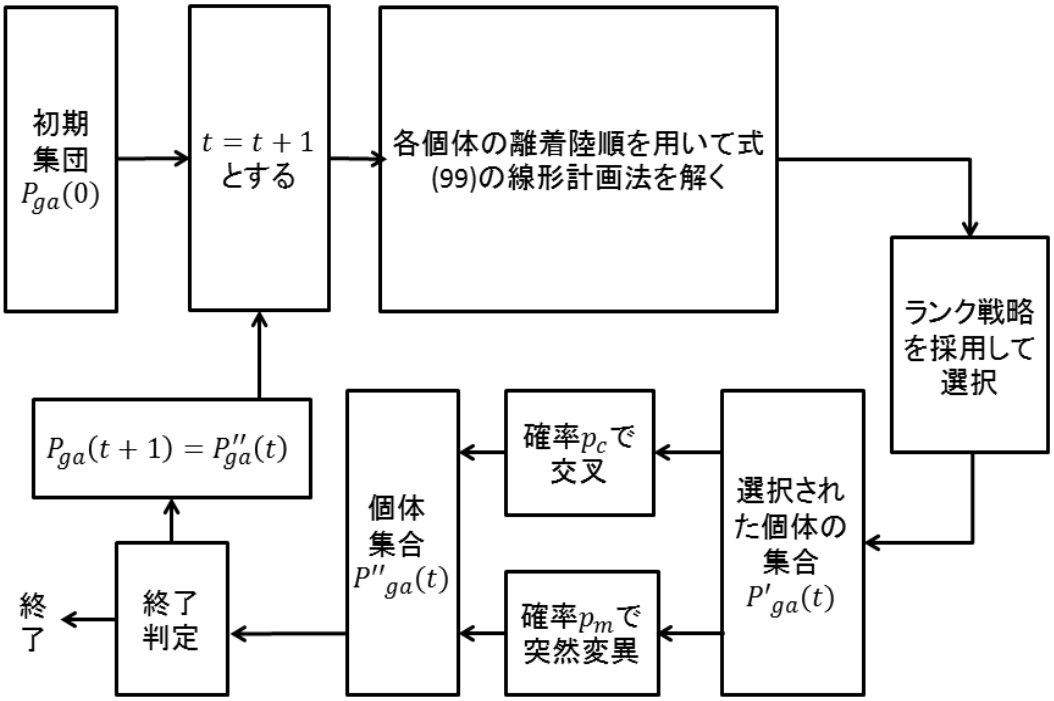

図 7　本研究での GA の手順

4. 数値実験

本章では，ケーススタディとして空港の滑走路のスケジューリング扱い，数値実験を行う．また，ランダムグラフを用いた数値実験も行う．数値実験を行う環境は以下のとおりである．

実験環境

CPU: Intel Core i5-24002.50GHz
OS: Microsoft Windows 7 Enterprise64bit
Memory: 4.0GB
Programming language: MathWorks MATALAB R2013b

4.1. ケーススタディ

本研究では，中部国際空港を例に max-min-plus scaling システムに当てはめ，滑走路や誘導路のスケジューリングを行う．図 8 は中部国際空港の概要図である．図 9 は図 8 をもとに誘導路と滑走路を PERT 図に示したものである．図 9 の状態の値は図 8 の値に対応している．

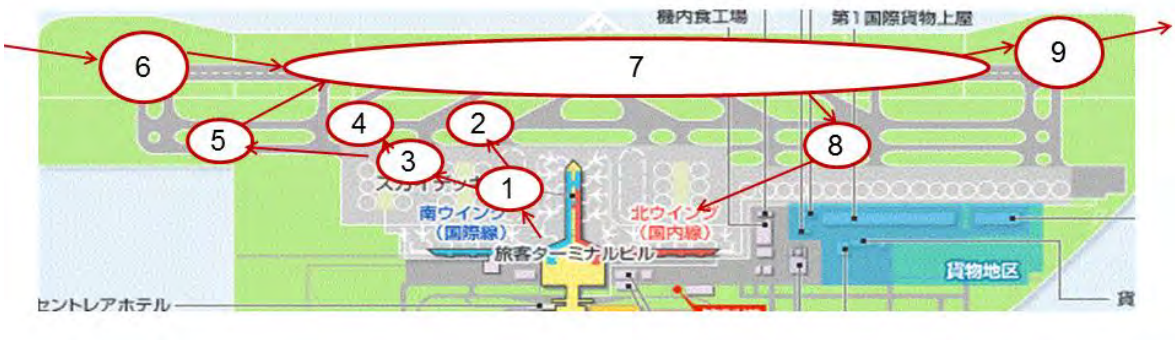

図 8　中部国際空港のイメージ図

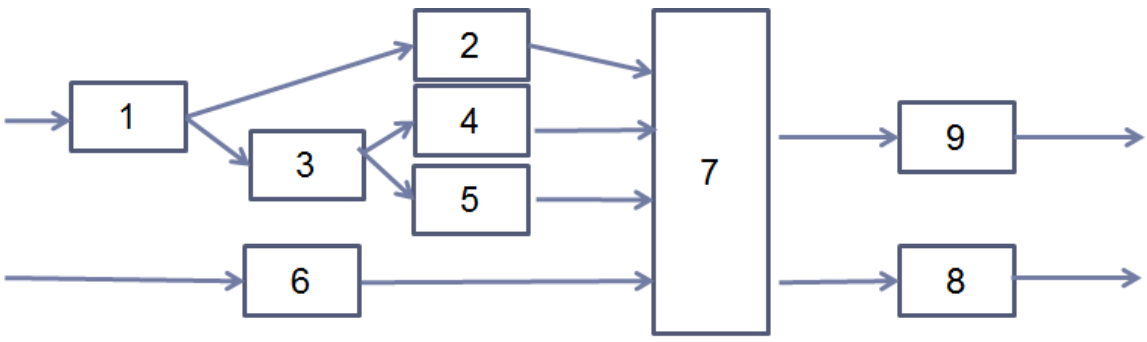

図 9 空港のモデル図

これらの誘導路のスケジュールを求める入力と出力は以下である.

入力：

- 先行関係：$\boldsymbol{F}_0 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ e & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & e & e & e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & e & \varepsilon & \varepsilon \end{bmatrix}$
- 入力行列：$\boldsymbol{B}_0 = \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & \varepsilon & e & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}^T$
- 出力行列：$\boldsymbol{C}_0 = [\varepsilon \quad \varepsilon \quad \varepsilon \quad \varepsilon \quad \varepsilon \quad \varepsilon \quad \varepsilon \quad e \quad e]$
- 中部国際空港の発着時刻を基に作成した表 2 の入力データを用いる.

出力：

- 誘導路の使用時間：$\boldsymbol{d}(k+l)$
- 各入力時刻：$\boldsymbol{u}(k+l)$
- 順序:c_1

また数値実験を行う上で以下のような前提条件を設ける.

- 着陸機，離陸機ともに滑走路への侵入時刻は自由に与えられる.
- 飛行機が安全に離着陸するための滑走路使用時間は 3 分以上である.

表 2　ケーススタディの入力データ

飛行機 No.	時刻表の時刻		滑走路への侵入の task 番号	最早入力時刻		最遅入力時刻		予定発着時刻	
	時	分		時	分	時	分	時	分
1	7	25	4	7	20	7	35	7	35
2	7	30	5	7	25	7	40	7	45
3	7	30	6	7	25	7	40	7	30
4	7	35	2	7	30	7	45	7	50
5	7	40	4	7	35	7	50	7	55
6	7	45	5	7	40	7	55	8	0
7	7	45	2	7	40	7	55	8	0
8	7	50	4	7	45	8	0	8	5
9	7	50	5	7	45	8	0	8	5
10	7	55	5	7	50	8	5	8	10
11	8	0	6	7	55	8	10	8	0

4.2. ケーススタディでの実験結果

本節では，4.1 で述べたケーススタディの数値実験結果を示す.

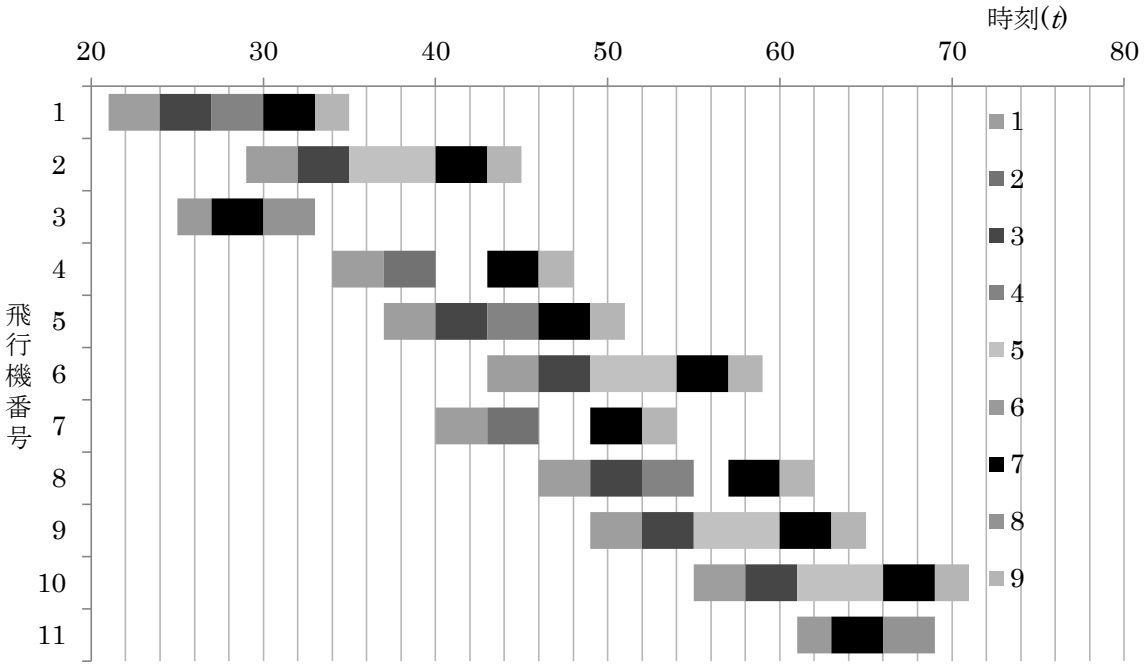

図 10　ケーススタディのスケジュール

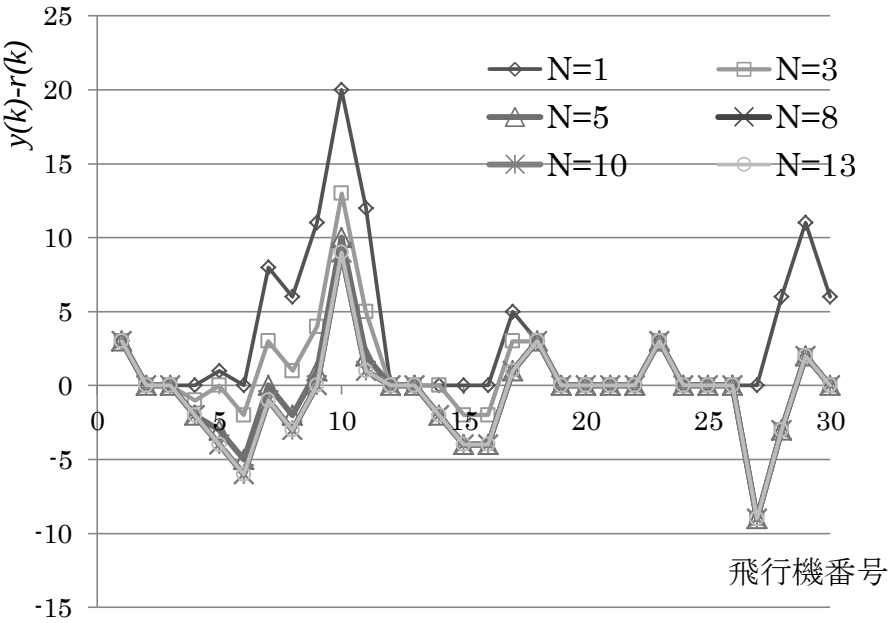

図 11　$y(k)-r(k)$の変化

表 3　離発着順

飛行機番号	1	2	3	4	5	6	7	8	9	10	11
離発着順	3	1	2	4	5	7	6	8	9	11	10

表 4 評価関数と制約条件の変化

予測ステップ数	T	制約条件の数
$N=1$	22,706	24
$N=3$	-7,096	185
$N=5$	-17,031	734
$N=8$	-22,017	3,577
$N=10$	-24,091	6,809
$N=13$	-26,832	20,141

図 10 は予測ステップ数が$N=8$ のとき，全数列挙を行い求まった，誘導路と滑走路のスケジュールである．飛行機 No.1~11 のスケジュールのみを図にした．色分けした数字は図 9 で示した状態の番号にあたる．どの飛行機もほかの飛行機と滑走路の割り当てが被ることなくスケジュールが組まれている．飛行機 1 の誘導路への入力時刻が，飛行機 3 よりも早いが，飛行機 3 の方が先に滑走路に到着し，表 3 の離発着順からもわかるように，先に滑走路を使用している．これらのことから，先に滑走路に到着した飛行機から，滑走路を使用するという max-min-plus scaling システムの条件を満たしているスケジュールが組まれている．

図 11 は各飛行機の予測離陸時刻や予測到着時刻から予定時刻を引いた値である．正の場合は遅れを表し，負の場合は離発着予定時刻よりも早く離陸や到着したことを表す．予測ステップ数が少ない$N=1$や$N=3$のとき，非常に多い遅れが生じている．一方で，予測ステップ数が多い$N=8$や$N=10$，$N=13$のとき，比較的遅れが少ない．特に飛行機番号 29 のとき，$N=1$や$N=3$では，遅れが生じているが，$N=8$や$N=10$，$N=13$のときは少ない遅れで済んでいる．これは，飛行機番号 26,27,28 の離発着時刻を早めることで，ペナルティの大きい遅れを回避しているからである．

表 4 は予測ステップ数の増加による，評価関数Tの値と制約条件の数の変化を表にしたものである．評価関数Tは以下のように定義する．

$$T=\alpha\sum_{k=1}^{N_p}\max\{0,\mu(k)\}+\beta\sum_{k=1}^{N_p}\sum_{i=1}^{n}d_i(k)-\gamma\sum_{k=1}^{N_p}\sum_{i=1}^{m}u_i(k)$$

この評価関数の値は，小さければ小さいほどよい解が得られたことを示す．予測ステップ数が大きいほど評価関数の値がよい．表 4 の制約条件の数とは，1 つの入力を決めるための制約条件の数である．制約条件の数は予測ステップ数が増えるにしたがって増えている．制約条件が多くなるほど，線形計画法で問題を解くのに時間がかかる．

表 5 予測ステップ数と近似比

手法			予測ステップ数		
	I	M	$N=3$	$N=5$	$N=8$
順序交叉	100	10	1.0000	1.0000	1.0185
		20	1.0000	1.0000	1.0000
	200	10	1.0000	1.0000	1.0000
		20	1.0000	1.0000	1.0000
周期交叉	100	10	1.0000	1.0000	1.0315
		20	1.0000	1.0000	1.0000
	200	10	1.0000	1.0000	1.0000
		20	1.0000	1.0000	1.0000

表 6 計算時間の平均(s)

手法			予測ステップ数		
	I	M	$N=3$	$N=5$	$N=8$
全数探索			0.4206	36.25	18,040
順序交叉	100	10	49.1	339.2	820.1
		20	104.8	859.9	1,190.0
	200	10	84.4	611.5	1083.5
		20	170.7	1174.7	2071.6
周期交叉	100	10	50.8	247.6	857.0
		20	96.9	709.1	1295.0
	200	10	84.1	526.3	1045.5
		20	167.3	1346.9	2212.0

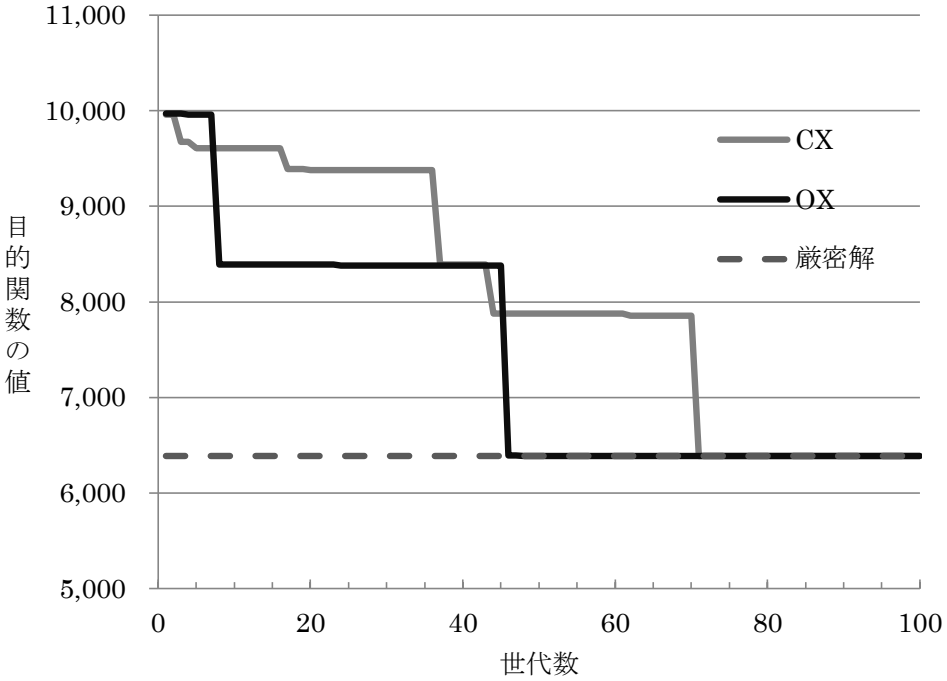

図 12　GA の解の改善課程

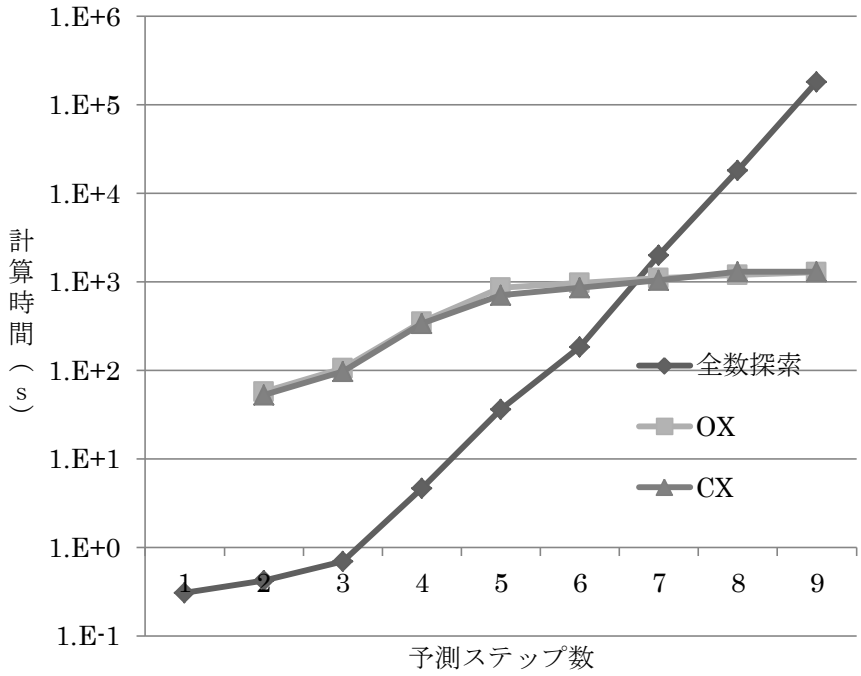

図 13　各手法と計算時間の変移

表 5 は予測ステップ数が増えたときの GA での近似比を表にしたものである．GA の交叉率は 0.3，突然変異率は 0.3 と設定した．近似比は近似解を厳密解で割った値で，近似比が 1 のときは厳密解が得られたときであり，1 に近い方がよい解であることを示す．$N = 8$のとき世代数が 100 で個体数が 10 のとき以外は，厳密解が求まった．表 6 は各手法の計算時間を表にしたものである．予測ステップ数が$N = 8$ のとき，全数探索を用いて離発着順を決めると約 6 時間かかる．しかし最適解が得られた，個体数 20，世代数 100 の順序交叉を用いた GA での計算時間は約 18 分であった．個体数が 10 から 20 にすると計算時間は約 2 倍となり，また世代数が 100 から 200 にかわると計算時間も約 2 倍となる．

図 12 は個体数 20 で，交叉率 0.3，突然変異率 0.3 のときの順序交叉と周期交叉の解の改善課程を表にしたものである．横軸は世代数，縦軸は目的関数の値である．破線は厳密解の値であり，この厳密解にどのように近づいていくかを表している．順序交叉は 42 世代目，周期交叉は 73 世代目で厳密解に到達した．順序交叉を用いた場合の方が速く最適解に到達した理由としては，入力した時刻表に基づいたデータが，半順列でできているためであると考えられる．また最適解に到達した世代数をみると，およそ 40 世代から 100 世代の間である．

図 13 は各手法の計算時間をグラフにしたものである．横軸は予測ステップ数，縦軸は計算時間の対数を示す．順序交叉と周期交叉は交叉率 0.3，突然変異率 0.3，個体数 20，世代数 100 のときの計算時間である．全数探索は指数関数に似た増加をしている．一方で GA の計算時間は予測ステップ数が 6 以下のとき，全数探索より時間がかかる．予測ステップ数が増えても計算時間は大幅に変わらないため，予測ステップ数が 7 以上のときは GA を用いて計算した方がよいと考えられる．

4.3. ランダムグラフでの実験結果

次にランダムに先行関係や入力を決定するランダムグラフを用いた数値実験を行う．

ランダムグラフの生成方法

具体的なランダムにシステム構造を生成する仕方は以下である．

- 先行関係の決定：サイクルグラフにならないように，状態 1 は先行作業を持たず，状態iの先行状態は状態1から状態$i - 1$間に，一定の確率で持つようにする．
- 入力の決定：状態 1 から状態nでランダムに入力を持つようにする．

各状態数に対して 10 サンプルのランダムグラフを作成し，数値実験を行う．

実験結果

表 7 は一つの順序に対して，滑走路の使用時間と侵入時刻を調整するのに必要な制約条件の数の平均である．ただし状態数が 10 で$N = 10$のときは，滑走路の使用時間と侵入時刻を調整するために要した計算時間は，実験の環境上，測定ができなかった．表 8 は一つの順序に対して，滑走路の使用時間と侵入時刻を調整するのに必要な計算時間の平均である．状態の数が増えることよりも，予測ステップ数が増えることの方が，計算時間や制約条件の数に与える影響が大きい．

表 7　制約条件数の平均

状態数	$N = 3$	$N = 5$	$N = 8$	$N = 10$
3	93.6	320.5	1,188	2,340
5	249.7	936.6	3,850	5,682
8	591.4	2,130	8,880	18,994
10	746.2	3,473	19,356	48,276

表 8　計算時間の平均(s)

状態数	$N = 3$	$N = 5$	$N = 8$	$N = 10$
3	0.0305	0.0705	0.6886	1.2976
5	0.0753	0.2117	0.8632	2.7851
8	0.2300	3.4287	530.9	6802.5
10	0.6573	4.2765	6511.4	–

表 9　順序交叉の近似比

		交叉率			
		0.1	0.2	0.3	0.4
突然変異率	0.1	1.0026	1.0252	1.0185	1.0601
	0.2	1.0000	1.0209	1.0000	1.0363
	0.3	1.0000	1.0000	1.1832	1.0231
	0.5	1.0000	1.0000	1.0000	1.0000
	0.7	1.0000	1.0000	–	–

表 10　周期交叉の近似比

		交叉率			
		0.1	0.2	0.3	0.4
突然変異率	0.1	1.1238	1.0212	1.0920	1.0409
	0.2	1.0000	1.0000	1.1959	1.1471
	0.3	1.0000	1.0000	1.0583	1.0223
	0.5	1.0000	1.0000	1.0000	1.0000
	0.7	1.0000	1.0000	–	–

表 9，表 10 は，世代数は 100，個体数は 20 の GA で，交叉方法を順序交叉と周期交叉を用いた時の近似比の平均である. 状態数が 8 のランダムグラフを用いて，予測ステップ数が$N = 8$ であるときの平均である．表 9，表 10 より突然変異率が大きいほうががよい解が得られる傾向があるようである. また周期交叉と順序交叉の同じ交叉率と突然変異率を比べると, 順序交叉を用いた時の方がよい解が得られることの方が多い．そのため，順序交叉を用いて，突然変異率を高く設定するとよい解が得られると考えられる.

4.4. 数値実験のまとめ

数値実験により, 提案手法によって早く滑走路に到着した機体から先に滑走路を使用できるような，max-min-plus scaling システムの条件を満たすスケジュールを組むことが可能であることがわかった．全数探索で離発着順の決定を行う場合，計算時間は，指数関数に従うような形で増えていく．例えば予測ステップ数が 8 のときに，計算時間は約 6 時間かかり，実用的ではない．一方で GA での計算時間は$N = 8$のときでも約 20 分で，予測ステップ数が増加しても計算時間は緩やかに増加している.

ケーススタディでは GA を用いて最適解が得られた． GA は近似的に解を見つける方法なので，最適解が求まる保証はされていない．しかし，ランダムグラフを用いた場合でも，突然変異の確率を高く設定することで，よい解が得られることがわかり， GA を用いて離発着順を求める方法は有効であることがわかった．また GA を用いる場合には，順序交叉を用いて，突然変異率を高く設定するとよい解が得られると考えられる.

計算時間で比較すると，予測ステップ数が 6 までのときは全数探索を用いて，予測ステップ数が 7 以上のときは GA を用いた方がよいと考えられる.

5. おわりに

本研究では，Max-min-plus scaling システムでのモデル予測制御を行った．一般的に Max-min-plus scaling システムでのモデル予測制御は，非線形問題で非凸問題である．滑走路のスケジューリングでは，離陸機と着陸機の滑走路の使用順序を決めるときに max 演算と plus 演算に加えて min 演算が必要である．離発着機の順序が決まっていれば，min 演算が必要でなくなる．Min 演算がなければ，max-plus 線形システムとして滑走路のスケジューリングを表現できる．そこで本研究では，離着陸順を全数探索と GA で決め，滑走路の使用時間や滑走への侵入時刻は，max-plus 線形システムでのモデル予測制御に基づいたスケジューリング手法で調整した．着陸機と離陸機はそれぞれ違った経路をたどるため，経路の情報が入った先行関係を導出した．さらに，この先行関係行列を用いて，状態予測式と出力予測式を導出した．

数値実験の結果，提案手法によって max-min-plus scaling システムの条件を満たす滑走路のスケジュールを組むことが可能であることがわかった．しかし，全数探索で離発着機の順序を求める場合，予測ステップ数が増えるにしたがって計算時間が膨大に長くなり，実用的でないことがわかった．ケーススタディでは，GA を用いても最適解が求まった．全数探索と GA の計算時間を比較すると，予測ステップ数が 6 までのときは全数探索を用いた方が，予測ステップ数が 7 以上のときは GA を用いた方がよいことがわかった．

今後の課題としては，他の手法をもちいた離発着順の決定方法などがあげられる．

参考文献

[1] 森村英典，刀根薫，伊理正夫：「経営科学 OR 用語大事典」，朝倉書店，2002.

[2] 古谷誠，杉本惇：「会社を強くするジャスト・イン・タイム生産の実行手順」，中経出版，2009.

[3] Jan M. Maciejowski:「Predictive Control with Constrains」，東京電機大学出版局，2005.

[4] 雨宮孝，竹安数博，増田士朗: 「新しい経営・経済数学」， 中央経済社，2004.

[5] 関根智明:「スケジューリング理論」，日刊工業新聞社，1971.

[6] 増田士朗，五島洋行，“max-plus 線形システムによる離散事象システムモデリングとモデル予測制御”，システム制御情報学会誌，vol.47，no.12, pp.577–582，2003.

[7] B. De. Shutter, T. J. J. van den Boom, “MPC for perturbed max-plus-liniear systems”, Automatica, vol.37, no.7, pp.1049–1056, 2001.

[8] B. De. Shutter, T. J. J. van den Boom, “MPC for perturbed max-plus-linear systems”, proceedings of the European Control Conference 2001, pp.3783–3788, 2001.

[9] H. Goto, “Model predictive control-based scheduler for repetitive discrete event systems with capacity constraints, An International Journal of Optimization and Control: Theories & Applications, vol.3, no.3, pp.73–83, 2013.

[10] B. De Schutter, and T. van den Boom, ”Model predictive control for max-min-plus systems”, The Kluwer International series in Engineering and Computer Science, vol.569, pp201–208, 2000.

[11] B. De Schutter, and T. van den Boom, “Model predictive control for max-min- plus scaling systems – Efficient implementation”, Proceeding of the 6th International Workshop on Discrete Event Systems, pp.343–348, 2002.

[12] Xiao-Bing Hu, and Wen-Hua Chen: “Receding horizon control for aircraft arrival sequencing and scheduling”, Institute of Electrical and Electronics Engineers, vol.6, pp189–197, 2005

[13] B. A. Carre, “An algebra for network routing problems”, IMA Journal of Applied Mathematics, vol.7, pp.273–294, 1971.

[14] G. Cohen, D. Dubois, J. P. Quadrat, “A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing”, IEEE Transaction on Automatic Control, vol. AC–30, no.3, pp.210–220, 1985.

[15] 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.

[16] 北野宏明：「遺伝的アルゴリズム」，産業図書，(1993).
[17] 三宮信夫，喜多一，玉置久，岩本貴司：「遺伝的アルゴリズムと最適化」，朝倉書店，(1998).