# サクラスケジューラの理論的背景（遺伝的アルゴリズム）

桜ロジック

2011 年 9 月 30 日

**概 要**

サクラスケジューラには、遺伝的アルゴリズムを使った 4 種類のスケジューリング・ロジック GA_G_Scheduler、 GA_S_Scheduler、 GA_H_Scheduler、および GA_T_Scheduler が既定で組み込まれています。この文書ではそれらのスケジューリング・ロジックの理論的な背景を詳述します。サクラスケジューラが扱うスケジューリング問題は、繰り返し加工を許すフレキシブルジョブショップ問題（FJSP：Flexible Job Shop Problem）です。ここで繰り返し加工とは、ジョブがその技術的順序で処理されていく過程で同じ作業（工程）を繰り返すことを意味します。ジョブの原材料が先頭工程に到着する時刻は決まっており、また各作業（工程）を処理する機械が故障や点検のために止まっている場合も考慮します。目的関数は投入進みの二乗と納期遅れの二乗の和の平均、または納期進みの二乗と納期遅れの二乗の和の平均です。遺伝的アルゴリズムの反復改善過程の探索解には、処理する順番を表す順列を使います。探索解に順列を使った場合、暫定解を評価したり最終的に解を求めるために実行可能化（デコード）する必要があります。実行可能化（デコード）の方法として色々な手法が提案されていますが、サクラスケジューラでは探索法、貪欲法、および Giffler & Thompson 法を採用しています。遺伝的アルゴリズムを使ったスケジューリング結果は、ジョブの密度が高い（スケジュールが混んでいる）場合にラグランジュ緩和法よりも最適度が劣り、計算時間もかなり多くなります。また、個体の集合の中での最良個体の目的関数値が収束するのに数多くの世代交代を経る必要があります。

# 1　問題の定式化

サクラスケジューラに組み込まれている遺伝的アルゴリズムを使ったスケジューリング・ロジックでは、投入進みと納期遅れの二乗の和の平均、または納期進みと納期遅れの二乗の和の平均（進み-遅れペナルティ）を最小にするよう、$N$ 個のジョブを $M$ 台の機械で処理するスケジュールを作成する問題を扱う。ジョブはそれぞれに異なった納期を持ち、特定の技術的順序で処理されなければならない工程から成る。各工程は複数の機械で処理が可能であり、スケジューリングの過程で目的関数値が最小となるように工程を処理する機械が決定される。また、ジョブが同じ機械で複数回処理されることを許す。以下の変数がスケジューリング問題の定式化で使われる。

| | |
|---|---|
| $N$ | ジョブ数、 |
| $n_i$ | ジョブ $i$ の工程の数、 |
| $w_i$ | ジョブ $i$ の重み、 |
| $M$ | 機械の数、 |
| $\mathcal{M}_{i,j}$ | ジョブ $i$ の工程 $j$ を処理可能な機械（複数台）、 |
| $D_i$ | ジョブ $i$ の納期、 |
| $R_i$ | ジョブ $i$ の先頭工程の集合、 |
| $G_i$ | ジョブ $i$ の終了工程の集合、 |
| $P_{i,j}$ | ジョブ $i$ の工程 $j$ の先行工程の集合、 |
| $t_{i,j,m}$ | ジョブ $i$ の工程 $j$ の機械 $m$ での加工時間、 |
| $c_{i,j}$ | ジョブ $i$ の工程 $j$ の完了時刻、 |
| $b_{i,j}$ | ジョブ $i$ の工程 $j$ の開始時刻、 |
| $a_{i,j}$ | ジョブ $i$ の先頭工程 $j \in R_i$ に原材料が到着する時刻、 |
| $s_{k,l}$ | 工程 $k$ と後続工程 $l$ の間の余裕時間、 |
| $H$ | 考慮する時刻数。 |

技術的順序は無閉路有向グラフを成し、複数の工程で始まり、複数の工程で終わり得る。工程 $k$ と工程 $l$ の間には、搬送や待機時間を表す余裕時間 $s_{k,l}$ を設定する。ジョブ $i$ の開始工程の集合を $R_i$、そして終了工程の集合を $G_i$ とすると、ジョブの投入進み $I_{i,r}$、納期進み $E_{i,g}$ と納期遅れ $T_{i,g}$ はそれぞれ次のように定義される。

$$I_{i,r} = \max\{0, r_{i,r} - b_{i,r}\}、r \in R_i \tag{1}$$

$$E_{i,g} = \max\{0, d_{i,g} - c_{i,g}\}、g \in G_i \tag{2}$$

$$T_{i,g} = \max\{0, c_{i,g} - d_{i,g}\}、g \in G_i \tag{3}$$

ここで、$r_{i,r}$ と $d_{i,g}$ は、ジョブの納期 $D_i$ を所与とする無限能力を仮定した PERT スケジューリングによって求められた開始工程と終了工程の山積開始時刻と完了時刻を表す。PERT スケジューリングでは、ジョブの納期を守りつつ、各工程はできるだけ遅く開始し、できるだけ早く終了する。また、各工程を処理する初期機械は、各工程で加工時間が一番長い機械、各工程で加工時間が一番短い機械、および納期の早いジョブから順番に各機械の負荷ができるだけ均等となるような機械を選択する 3 種類が採用される。このスケジューリング・ロジックでは、機械での工程の処理の中断は許さないことを仮定する。すべてのジョブの原材料が生産ラインへ到着する時刻は決まっており、それ以前に処理を開始できない。また、機械は常に利用可能であるとは限らず、故障や点検

のために止まっている場合も考慮する。

スケジューリング問題の決定変数は、各工程を処理する機械 $m_{i,j}$ とその機械での開始時刻 $b_{i,j}$ である。スケジューリングによって、次のように定義される目的関数 $J_{IT}$、または $J_{ET}$ を最小とする全ジョブの全工程の処理機械 $m_{i,j}$ と開始時刻 $b_{i,j}$ を決定する。

$$J_{IT} = Q\sum_i w_i \left( \sum_{r\in R_i} I_{i,r}^2 + \sum_{g\in G_i} T_{i,g}^2 \right) 、Q = \frac{1}{\sum_i (|R_i| + |G_i|)} \tag{4}$$

$$J_{ET} = Q\sum_i w_i \left( \sum_{g\in G_i} E_{i,g}^2 + \sum_{g\in G_i} T_{i,g}^2 \right) 、Q = \frac{1}{\sum_i |G_i|} \tag{5}$$

一般に納期遵守率の向上とリードタイムの短縮は生産管理上の重要な目標となる。前者は納期遅れのペナルティ $T_i$ によって、そして後者は投入進みのペナルティ $I_i$ によって作成するスケジュールに反映される。ここで、ジョブ $i$ の工程 $j$ を時刻 $\tau$ に機械 $m_{i,j}$ で処理中の場合は 1、それ以外は 0 となる $\delta_{i,j,m_{i,j},\tau}$ を導入して、機械制約は次のように書ける。

$$\sum_{i,j} \delta_{i,j,m_{i,j},\tau} \le A_{m,\tau}、\tau = 1,2,...,H \tag{6}$$

ここで $A_{m,\tau}$ は、時刻 $\tau$ に機械 $m$ が稼動していれば 1、それ以外は 0 の値を取る。ジョブ $i$ の工程 $j$ の先行工程の集合を $P_{i,j}$ とすると、先行関係制約は以下のとおり。

$$c_{i,k} + s_{i,k} \le c_{i,j} - t_{i,j,m_{i,j}}、k \in P_{i,j} \tag{7}$$

また、到着時刻制約と処理時間要件は次の通り。

$$a_{i,r} \le c_{i,r} - t_{i,r,m_{i,r}}、r \in R_i \tag{8}$$

$$c_{i,j} - b_{i,j} + 1 = t_{i,j} \tag{9}$$

## 2　遺伝的アルゴリズム

近年、生産スケジューリング問題を含む組合せ最適化問題に対して、焼き鈍し法、タブー・サーチ法、そして遺伝的アルゴリズムに代表されるメタヒューリスティック法が提案され、盛んに研究されるとともに現実の問題に適用されつつある。この手法は、暫定解に摂動を加えて得られる解を近傍解とし、近傍解が暫定解より目的関数を良くすればそれを新たに暫定解とする操作を繰り返す反復改善法に基づく。反復改善法を生産スケジューリング問題を含む組合せ最適化問題に適用する際に問題となるのは、反復改善過程で解が局所最適解に陥ってしまうことである。遺伝的アルゴリズムでは、問題の暫定解を一つではなく複数の個体（暫定解）の集合で表現し、生物の遺伝機構と目的関数を良くしない個体もある程度の確率で発生（遺伝的浮動）してある程度の確率で生き残ることを許す自然選択を模倣し、個体集合に対する適応や最適化を行うことでこの欠点を克服する [1]。また、同じく暫定解の生成に確率過程を導入している焼き鈍し法とのもう 1 つの違いは、近傍解の生成に個体間で遺伝情報を交換する交叉演算を導入している点にある。この交叉演算子の導入により、暫定解により大きな変形を加えた近傍解の生成が可能となり、暫定解が局所最適解に陥る可能性を更に低くする [2]。

## 2.1　遺伝子型と表現型

遺伝的アルゴリズムをフレキシブルジョブショップ型の生産スケジューリング問題に適用する場合、その遺伝子型の定義として主に次の３種類がある [3]。ここで遺伝子型とは、選択、交叉、および突然変異といった遺伝的操作の対象となる個体（暫定解）のことを指す。

- 各ジョブの各工程の完了時刻の集合として定義する。
- 各機械で、その機械で処理される工程の順列（順番）として定義する。
- 各機械で、その機械で処理される２つの工程の順番の集合として定義する。

サクラスケジューラの遺伝的アルゴリズムを使ったスケジューリング・ロジックでは、２番目の工程の順列による定義を採用する。１番目の所謂ガントチャートによる定義は、遺伝子型と表現型の変換が不要であるという利点を持つが、生産スケジューリングに付き物の様々な制約を満足する近傍解の生成に多くの計算時間を必要とするという欠点を持つ。また、３番目の所謂離接グラフ（選択グラフ）による定義は、焼き鈍し法でこそ有効な近傍解の生成方法（CB 近傍）が提案されているが [4]、遺伝的アルゴリズムの要諦を成す交叉演算子の構成が困難である。

その定義からも判るように、工程の順列によって表される遺伝子型は生産スケジューリングで必要とされる様々な制約の充足がなされていないという意味で実行不可能であり、何らかの手法を使って表現型に変換する（実行可能化またはデコード）する必要がある。遺伝子型から表現型への変換（実行可能化またはデコード）は、先に述べたように様々な制約を満足しなければいけないために、遺伝的アルゴリズムを使ったスケジューリング・ロジックで一番計算時間を必要とするだけでなく、その手法が反復改善過程の結果として得られる近似最適解の目的関数値の良し悪しを左右する。後述するように、サクラスケジューラの遺伝的アルゴリズムを使ったスケジューリング・ロジックでは、リストスケジューリングの概念に基づく探索法、貪欲法、そして Giffler & Thompson 法 [5] を使う。

## 2.2　遺伝的操作

生物の進化過程においては、個体の集合の中で環境への適応度が高い個体が高確率で次世代に生き残るという自然選択、母親と父親の遺伝子を掛け合わせて次世代の子孫を生むという交叉（遺伝子組み換え）、そして分子レベルの様々な原因で対立遺伝子が変わるという突然変異の３つの遺伝的操作が個体およびその集合に加えられる。遺伝的アルゴリズムにおいても同様で、個体（暫定解）の集合に対して選択演算子、交叉演算子、そして突然変異演算子という３つの演算子を作用させ、次世代の個体（暫定解）の集合を生成する。

サクラスケジューラの遺伝的アルゴリズムを使ったスケジューリング・ロジックでは、選択演算子に各世代の個体の集合の中で上位の適応度を持つ個体については交叉や突然変異の対象とはせず無条件に次世代に残すというエリート選択を採用する [1]。交叉演算においては、既に存在する両親の情報になんらの情報も追加することなく子供を生成するという原則が広く使われており、サクラスケジューラの遺伝的アルゴリズムを使ったスケジューリング・ロジックでも、この原則に則りつつ交叉演算によって両親が満たしていた先行関係制約を子供も満たすという利点を持つ Precedence Preservative Crossover（PPX）交叉演算子を採用する [6]。突然変異演算子は、個体の集合に新たな遺伝情報を追加するという重要な役割を担っており、任意に選択した同じ機械で処理される工程の順番の入れ替え、または工程を処理する機械の入れ替えとして定義する。

### 2.3　基本構成

遺伝的アルゴリズムによる反復改善過程の基本的な構成は次のとおり [1]。

- ステップ 1：ランダムに $M$ 個の個体を生成して初期個体集合 $P(0)$ を作り、世代 $t = 0$ とする。繰り返し回数（最終世代）$T$ を設定する。
- ステップ 2：個体集合 $P(t)$ 内の個体について、その適応度 $g$ を計算する。
- ステップ 3：個体集合 $P(t)$ に選択演算子を適用し、$P'(t)$ を生成する。
- ステップ 4：$P'(t)$ に交叉演算子を適用し、$P''(t)$ を生成する。
- ステップ 5：$P''(t)$ に突然変異演算子を適用し、次世代の個体集合 $P(t+1)$ を生成する。
- ステップ 6：$t \leq T$ ならば $t = t+1$ としてステップ 1 へ。そうでなければ計算を終了し、これまでに得られた最大適応度の個体を近似最適解とする。

ここで、適応度 $g$ は目的関数 (4) の値の逆数として定義する。

## 3　実行可能解を構成する

探索解を表すのに順列表現を使った場合、遺伝的アルゴリズムの反復改善過程で決まるのは各工程を処理する機械と各工程をその機械に割り付ける順序だけであるため、目的関数値、すなわち適応度を計算するのに、その順序に従って機械制約を破らないように各工程を機械に割り付けていく（実行可能化またはデコードする）必要がある。サクラスケジューラに組み込まれている遺伝的アルゴリズムを使ったスケジューリング・ロジックでは、リストスケジューリングの概念に基づく探索法、貪欲法 および Giffler & Thompson 法で実行可能解を作成する。

リストスケジューリングとは、何らかの方法で決定したリストに従って技術的順序制約、処理時間要件 (9)、そして機械制約 (6) を破らないように各工程を機械に割り付けていく手法である。探索法では、探索空間の順列をリストとみなし、リストの順番に各工程を山積開始時刻に割り付けていき、機械制約が破られたときに山積開始時刻を基点として機械に割り当て可能な時刻を前方および後方探索し、見つかった両方向の時刻のうち山積開始時刻に近い方を採用する。貪欲法では、無限能力を仮定した納期を所与とする PERT スケジューリングで求めた各工程の山積開始時刻をその昇順に並べることでリストを作成する。そして、リストの順番での割り付けの過程で機械制約が破られたとき、探索解である順列の順番に貪欲法に基づいてどのジョブのどの工程を機械に割り当てるかを決める。Giffler & Thompson 法では、他のどの作業の処理開始時刻にも影響を与えないで他の作業を飛び越してより早く着手できる作業を順次前に移動することによって、他の作業の着手を遅らせるのでなければもはやどの作業も前に移動できないように工程を割り付ける [5]。

### 3.1　探索法を使った実行可能解の構成

順列から作られる割り付け順序を表すリストは、割り付け順番が早い方が有利になるという意味を持っている。サクラスケジューラの遺伝的アルゴリズムを使ったスケジューリング・ロジックでは、この性質を生かしつつ簡単かつ高速に実行可能解を構成する。詳細な手順は次のとおり。

- ステップ 1：各機械群の利用可能機械台数 $A_{m,\tau}$ を機械台数に設定する。次に、全工程を順列から作ったリストの順番に並べて $r_{i,j} = b_{i,j}$ とし、この工程の集合を $S$ とする。$p \in S$、$p = 1, ..., |S|$ と $p$ を定義し、$p = 1$ とする。
- ステップ 2：工程 $p$ が $b_{i,p}$、...、$b_{i,p} + t_{i,p}$ の間 $A_{m_{i,p},\tau} > 0$ なら工程 $p$ を $b_{i,p}$ に割り付け、ステップ 3 に行く。さもなければ ステップ 4 に行く。
- ステップ 3：$b_{i,p}$、...、$b_{i,p} + t_{i,p}$ の間 $A_{m_{i,p},\tau} = A_{m_{i,p},\tau} - 1$ とし、$p = p + 1$ としてステップ 2 に行く。
- ステップ 4：$k = b_{i,p} + 1$ とする。
- ステップ 5：工程 $p$ が $k$、...、$k + t_{i,p}$ の間 $A_{m_{i,p},\tau} > 0$ なら $K = k$ としてステップ 6 に行く。さもなければ $k = k + 1$ としてステップ 5 を繰り返す。
- ステップ 6：$l = b_{i,p} - 1$ とする。
- ステップ 7：工程 $p$ が $l$、...、$l + t_{i,p}$ の間 $A_{m_{i,p},\tau} > 0$ なら $L = l$ としてステップ 8 に行く。また、$l < 0$ の場合、もしくは $|b_{i,p} - l| \geq |b_{i,p} - K|$ の場合もステップ 8 に行く。さもなければ $l = l - 1$ としてステップ 7 を繰り返す。
- ステップ 8：$|b_{i,p} - L| \geq |b_{i,p} - K|$ なら工程 $p$ を時刻 $K$ に割り付ける。さもなければ時刻 $L$ に割り付ける。$p = |S|$ なら処理終了。さもなければ $p = p + 1$ としてステップ 2 に行く。

## 3.2　貪欲法を使った実行可能解の構成

生産スケジューリングにおける貪欲法の利用は、ラグランジュ緩和法などでは一般的である。ここでは、緩和解の早い順にリストを構成し、投入進みペナルティは緩和解のときの値以上に悪くしないという制約の下で各ジョブの各工程をリストの順番に緩和解の時刻に機械に割り当てていき、時刻 $k$ において機械制約が破られたら緩和解と山積開始時刻の差の大きいものから順番に機械に割り当てる。一方、ForestGreen の遺伝的アルゴリズムを使ったスケジューリング・ロジックでは、機械制約が破られたとき、探索解である順列の順番に機械に割り当てる。実行可能解を構成する詳細な手順は次のとおり。

- ステップ 1：各機械群の利用可能機械台数 $A_{m,\tau}$ を機械台数に設定する。次に、全工程を山積開始時刻 $r_{i,j} = b_{i,j}$ の昇順で並べて $\tau = 0$ とする。この工程の集合を $S$ とする。
- ステップ 2：$b_{i,j} = \tau$ となる工程の集合 $E_\tau$ を抽出する。次いで、$E_\tau$ の工程を割り付け順序の順に並べ替える。
- ステップ 3：$p \in E_\tau$、$p = 1, ..., |E_\tau|$ と $p$ を定義し、$p = 1$ とする。
- ステップ 4：工程 $p$ が $b_{i,p}$、...、$b_{i,p} + t_{i,p}$ の間 $A_{m_{i,p},\tau} > 0$ なら工程 $p$ を $b_{i,p}$ に割り付ける。$b_{i,p}$、...、$b_{i,p} + t_{i,p}$ の間 $A_{m_{i,p},\tau} = A_{m_{i,p},\tau} - 1$ とし、$S = S - \{i, p\}$ とする。さもなければ $b_{i,p} = b_{i,p} + 1$ とし、後続工程を技術的順序を破らないように後ろにずらす。ステップ 5 に行く。
- ステップ 5：$p = |E_\tau|$ ならステップ 6 に行く。そうでないなら $p = p + 1$ としてステップ 4 に行く。

- ステップ 6：$|S| = 0$ なら処理終了。さもなくば、$\tau = \tau + 1$ とし、残っている工程の集合 $S$ を開始時刻の昇順で並べなおす。ステップ 2 に行く。

### 3.3 Giffler & Thompson 法を使った実行可能解の構成

実行可能スケジュールの集合の中でアクティブ・スケジュールの集合の中に最適なスケジュールが存在するは良く知られている [6]。ここで、アクティブ・スケジュールとは、各機械上での作業の処理順序を変更してもよいとして、他の作業の開始時刻を遅らせることなしに、いずれの作業も処理開始を早めることができないスケジュールである。Giffler & Thomspon 法によって、このアクティブ・スケジュールが構成可能である [5]。実行可能解を構成する詳細な手順は次のとおり。

- ステップ 1：各ジョブの開始工程の集合 $S$ を抽出する。
- ステップ 2：集合 $S$ から完了時刻のもっとも早い工程 $p \in S$ を選択する。
- ステップ 3：工程 $p$ と同じ機械で処理される工程の集合 $B$ を抽出する。
- ステップ 4：集合 $B$ から工程 $p$ の完了時刻以降に開始予定の工程を削除する。
- ステップ 5：集合 $B$ から割り付け順序の最も早い工程 $p^*$ を選択する。
- ステップ 6：$p^*$ の開始時刻を探索法と同じロジックを使って決定し、$S = S - \{p^*\}$ とする。
- ステップ 7：$p^*$ の後続工程を集合 $S$ に加える。
- ステップ 8：$|S| = 0$ なら処理終了。さもなければステップ 2 に行く。

## 4 各スケジューリング・ロジック

サクラスケジューラに組み込まれている遺伝的アルゴリズムを使ったスケジューリング・ロジックで採用されている実行可能解の構成（デコード）方法は次のとおり。

- GA_G_Scheduler：各反復改善過程で貪欲法を使って実行可能化（デコード）する。
- GA_S_Scheduler：各反復改善過程で探索法を使って実行可能化（デコード）する。
- GA_H_Scheduler：各反復改善過程では探索法を使い、最良の個体が見つかった段階で貪欲法を使って実行可能化（デコード）し直す。
- GA_T_Scheduler：各反復改善過程で Giffler & Thompson 法を使って実行可能化（デコード）する。

## 参考文献

[1] 三宮 信夫, 喜多 一, 玉置 久, 岩本 貴司, 遺伝的アルゴリズム, 朝倉書店, 1998

[2] 柳浦 睦憲, 茨木 俊秀, 組合せ最適化, 3 章, 朝倉書店, 2001

 

[3] T. Yamada and R. Nakano, A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems, *Parallel Problem Solving from Nature*, Vol.2 pp.281-290, 1992

[4] T. Yamada, B. E. Rosen and R. Nakano, A Simulated Annealing Approach to Job Shop Scheduling using Critical Block Transition Operators, *Neural Networks*, Vol.7 pp.4687-4692. 1994

[5] B. Giffler and G. Thompson, Algorithms for Solving Production Scheduling Problems, *Operations Resarch*, Vol.8 pp.487-503, 1960

[6] C. Bierwirth and D. Mattfeld, Production Scheduling and Rescheduling with Genetic Algorithms, *Evolutionary Computation*, Vol.7 pp.1-17, 1999

 