作業割当問題における遺伝アルゴリズムの構成法*

当麻 淳治**・飯間 等***・三宮 信夫***

A Method for Constructing Genetic Algorithm in an Operation Assignment Problem*

Junji TOUMA**, Hitoshi IIMA*** and Nobuo SANNOMIYA***

This paper deals with an operation assignment problem. The problem is to process N products at one of parallel H machines in order to minimize the deviation of the processing time of each product from its desirable time interval. Then, we have to determine both the order and the machine to process each product.

The genetic algorithm (GA) is applied to this problem. Several individual descriptions and the related decoding methods are proposed. By examining numerical results, comparison is made among these individual descriptions from the viewpoints of accuracy and computation time. Consequently, the best individual description is found to be the one defined in such a way that both the order and the machine are determined by GA.

1. 緒 言

スケジューリング問題は多くの場合NP困難で、問題の規模が少し大きくなっただけでも可能解の個数は飛躍的に増大し、厳密な最適解を求めることは困難である^{1),2)}。そこで、スケジューリング問題に対する準最適解を求めるために、生物の進化過程を模擬して解を得る遺伝アルゴリズム (Genetic Algorithm : GA)³⁾を適用する試みがある^{4),5)}。GAは複数の探索点を同時にかつ確率的に更新させるので、解空間を幅広く探索することができる。ただし、スケジューリング問題の解法としてGAを用いるには、遺伝演算子、GAパラメータ、個体表現およびデコーディング法を、問題固有の性質を有効に利用して設定することが不可欠である。

本論文で扱う問題は、複数の製品を複数台の並列な機械のいずれかに割り当てて作業させる問題（以下作業割

当問題と呼ぶ）である。この問題は、通常の並列機械問題を拡張して、限られた機械で各製品を、できるだけ要求された時間帯内で処理することが目的である。一方、これと似た問題として、日程計画の山くずし⁶⁾がある。これは、作業日程において必要な人員や資材などの資源が限られているときに、資源のピークをくずす方法である。一般に、山くずしの対象となる日程計画には、作業間の先行関係があり、それによって最早処理開始時刻と最遅処理終了時刻がPERT計算により求められる。これに対して本研究の作業割当問題は、各製品には最早処理開始時刻と最遅処理終了時刻が与えられているが、これらの時刻は緩和可能なソフトな制約条件として扱われる。

すなわち、最早処理開始時刻を早めることは、その前工程で製品の部品の調達を早めさせることになり、他方最遅処理終了時刻を遅らせると、納期が遅れ、後工程に悪影響を与える。しかし、限られた設備に集中した作業をいかに割り当てるかを決めるために、本論文では制約条件は緩和可能であるとして、緩和量をペナルティと考えてその最小化を図る。

本論文では、作業割当問題にGAを適用する際のGAの個体表現とデコーディング法を提案し、数値計算例によりその有効性を考察する。

* 原稿受付 1995年1月26日

** 京都工芸繊維大学 大学院（現在、矢崎総業（株））Graduate School, Faculty of Engineering and Design, Kyoto Institute of Technology ; Goshokaidocho, Matsugasaki, Sakyo ward, Kyoto city, Kyoto 606, JAPAN

*** 京都工芸繊維大学 工学部 Faculty of Engineering and Design, Kyoto Institute of Technology ; ditto

Key Words: genetic algorithm, operation assignment problem, scheduling, parallel machines, individual description

2. 問題の記述

本論文では、 N 種類の製品を、 H 台の並列に配置された同種の機械で処理する場合の製品の処理順序問題を考える。

各製品は、並列な機械のうちいずれか 1 台の機械で処理するものとする。製品 i ($i=1,2,\dots,N$) の機械での処理時間 c_i 、最早処理開始時刻 a_i および最遅処理終了時刻 b_i が与えられており、できれば時間区間 (a_i, b_i) で処理したいとする。ただし、 $H < N$ および $b_i - a_i > c_i$ とし、時刻は離散量で与えられるものとする。このとき、区間 (a_i, b_i) 外での処理時間が最も少なくなるような製品の処理順序を求める問題を考える。

この問題を数理計画問題として定式化をする。 s_i および e_i を、それぞれ製品 i の実際の処理開始時刻および終了時刻とすれば、問題は次式で与えられる。

目的関数：

$$\min Z, \quad (1)$$

$$Z = \sum_{i=1}^N [\max(e_i - b_i, 0) + \max(a_i - s_i, 0)]$$

制約条件：

$$\sum_{i=1}^N x_{ij}(k) \leq 1, \quad (2)$$

$$k = K_0, K_0 + 1, \dots, K_f - 1; j = 1, 2, \dots, H$$

$$x_{ij}(k) = \begin{cases} 1, & s_i \leq k \leq e_i - 1 \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

$$e_i = s_i + c_i \quad (4)$$

$$K_0 = \min_i s_i \quad (5)$$

$$K_f = \max_i e_i \quad (6)$$

ここに、 $x_{ij}(k)$ は、区間 $(k, k+1)$ で機械 j が製品 i を処理するとき 1、否のとき 0 の値をとる変数である。

このような問題は、スケジューリング問題の分類記号で表すと $P \parallel \sum T_j$ となる並列機械問題を拡張したもので、NP 困難な問題である²⁾。本論文では、この問題に GA を適用して準最適解を求める方法を提案する。

3. GA の適用

本論文では、GA で用いられる用語にしたがって、解を個体、解集合を個体群、解計算の繰り返し回数を世代数と呼ぶことにする。

3. では、まず全体のアルゴリズムの概要を述べ、その後各ステップで行われる GA の計算手続きを説明する。全体のアルゴリズムとしては基本的なものであり、特別技巧的な工夫は施していない。むしろ、以下に述べるように、GA を適用する場合の個体表現およびデコーディング法を工夫することによって、比較的基本的な GA の枠組でも十分有用な解が得られることを示す。

3.1 アルゴリズムの概要

GA の手順は次のステップからなり、世代数が t^* になると終了する。なお、個体数は世代を通じて一定で、 M 個とする。

ステップ 1 M 個の個体をランダムに生成して、初期個体群を設定する。世代を $t \leftarrow 1$ とする。

ステップ 2 各個体の適応度を計算して個体の淘汰を行う。

ステップ 3 M 個の個体をランダムに 2 個ずつ組み合わせ、確率 P_c で交叉を行って新しい個体を生成し、その生成に関与した古い個体と置き換える。

ステップ 4 M 個の個体の各々について、確率 P_m で突然変異を行って、得られた個体を古い個体と置き換える。

ステップ 5 $t = t^*$ ならば、最終世代における適応度が最大の個体を問題の準最適解として、計算を終了する。そうでないならば、 $t \leftarrow t+1$ としてステップ 2 へ戻る。

3.2 個体表現

GA において個体表現をいかにするかは問題に依存して定めるべきである。本問題では製品の並列機械による処理順序を決定するので、以下の 3 種類の方法を採用する。

[ID-1] 各製品を処理する機械を個体で定めることとし、対立遺伝子は $\{1, 2, \dots, H\}$ である。よって、個体は機械番号を N 個並べた列 $X_1 X_2 \dots X_N$ で表す。

[ID-2] 各製品の処理順序を個体で定めることとし、対立遺伝子は $\{1, 2, \dots, N\}$ である。よって、個体は製品番号を N 個並べた列 $Y_1 Y_2 \dots Y_N$ で表す。

[ID-3] 各製品を処理する機械およびその処理順序を個体で定めることとし、2 種類の遺伝子列で個体表現を行う。すなわち、機械番号 $\{1, 2, \dots, H\}$ を N 個並べた列 $X_1 X_2 \dots X_N$ および製品番号 $\{1, 2, \dots, N\}$ を N 個並べた列 $Y_1 Y_2 \dots Y_N$ で表す。

これらの個体表現の具体例は、後述する Fig. 4 のガントチャートに対応して (15) および (16) 式のように示されているので、参照されたい。

3 種類の個体表現のうち、ID-2 はすでに著者ら^{7), 8)} によって提案されている。本論文では、新たに ID-1 および

ID-3を提案し、これらの個体表現の優劣を考察する。

3.3 デコーディング法

個体（遺伝子型）から各製品を処理する機械番号および処理開始時刻（表現型）を計算する方法を、デコーディング法と呼ぶ。ここでは、デコーディング法を各個体表現に対応させて考える。以下に示すデコーディングのアルゴリズムにより、それらの値が決まれば、(2)式より目的関数値 Z が定まり、(2)～(6)式を満たす決定変数 $x_{ij}(k)$ が定まる。

3.3.1 ID-1の場合

ID-1では、まず、個体表現から機械 m に割り当てられる製品の集合 S_m を、次の手順で求める。

ステップ 1 $m=1,2,\dots,H$ について、 $S_m \leftarrow \phi$.

ステップ 2 $i=1,2,\dots,N$ について、 $m \leftarrow X_i$,

$$S_m \leftarrow S_m \cup \{i\}.$$

つぎに、集合 S_m に属する各製品の s_i および e_i を求める。その方法をここでは3種類考えるが、それらはすべて機械ごとに独立に s_i および e_i を決めることになるので、1機械、すなわちある機械 m に対して製品を割り当てるアルゴリズムである。

[方法 I] ある時点で処理可能な製品のうち、最遅処理終了時刻 b_i が最も小さい製品を割り当てる。そのアルゴリズムは、次のようになる。

ステップ 1 $t \leftarrow 0$.

ステップ 2 $i \in S_m$ なる i について、 $a_i \leq t$ を満たす i を探す。そのような i があれば、それらのうち b_i が最小のものを i^0 として、ステップ 4 に進む。そのような i がなければ、ステップ 3 に進む。

ステップ 3 $i \in S_m$ なる i について、 a_i が最小のものを i^0 とし、 $t \leftarrow a_{i^0}$.

ステップ 4 $s_{i^0} \leftarrow t$, $e_{i^0} \leftarrow s_{i^0} + c_{i^0}$, $t \leftarrow e_{i^0}$, $S_m \leftarrow S_m - \{i^0\}$.

ステップ 5 $S_m \neq \phi$ ならば、ステップ 2 に戻る。 $S_m = \phi$ ならば終了。

[方法 II] 各製品を、仮に区間 (a_i, b_i) 内に割り当て、処理時間の重なっている場合は、各製品の前後関係を崩さないように、処理時間を前後にずらして重ならないようにする。そのアルゴリズムは、次のようになる。

ステップ 1 $i \in S_m$ なる i について $s_i \leftarrow (b_i + a_i - c_i)/2$, $e_i \leftarrow s_i + c_i$.

ステップ 2 $(s_i + e_i)/2$ の昇順になるように S_m の要素をソートし、その結果を $\{q\}$ に入れる。

ステップ 3 $k_1 \leftarrow S_m$ の集合の要素の数, $k_2 \leftarrow \lceil k_1/2 \rceil$.

ステップ 4 $l = k_2 - 1, \dots, 2, 1$ について、 $p \leftarrow q_l$, $p' \leftarrow q_{l+1}$ とし、 $e_p > s_{p'}$ ならば、 $e_p \leftarrow s_{p'}$, $s_p \leftarrow e_p - c_p$.

ステップ 5 $l = k_2 + 1, \dots, k_1 - 1, k_1$ について、 $p \leftarrow q_l$, $p' \leftarrow q_{l-1}$ とし、 $s_p < e_{p'}$ ならば $s_p \leftarrow e_{p'}$, $e_p \leftarrow s_p + c_p$.

[方法 III] 方法 II を以下のように変更する。すなわち、仮の割り当てで生じた機械の空きを、その時点での目的関数値が変わらないように埋め、さらに全体の処理時間を動かすことによって目的関数値の改善ができる場合にはそれを実行する。方法 II のステップ 4 以降を以下のように変更する。

ステップ 4 $l = k_2 - 1, \dots, 2, 1$ について、 $p \leftarrow q_l$, $p' \leftarrow q_{l+1}$ とし、 $e_p > s_{p'}$ ならば $e_p \leftarrow s_{p'}$, $s_p \leftarrow e_p - c_p$. $e_p < s_{p'}$ ならば $e_p \leftarrow \min(s_{p'}, b_p)$, $s_p \leftarrow e_p - c_p$, $s_{p'} \leftarrow \max(e_p, \min(s_{p'}, a_{p'}))$, $e_{p'} \leftarrow s_{p'} + c_{p'}$.

ステップ 5 $l = k_2 + 1, \dots, k_1 - 1, k_1$ について $p \leftarrow q_l$, $p' \leftarrow q_{l-1}$ とし、 $s_p < e_{p'}$ ならば $s_p \leftarrow e_{p'}$, $e_p \leftarrow s_p + c_p$. $s_p > e_{p'}$ ならば $s_p \leftarrow \max(e_{p'}, a_p)$, $e_p \leftarrow s_p + c_p$, $e_{p'} \leftarrow \min(s_{p'}, \max(b_{p'}, e_{p'}))$, $s_{p'} \leftarrow e_{p'} - c_{p'}$.

ステップ 6 $k=1,2,3,4$ について、 $y_k \leftarrow 0$.

ステップ 7 $i \in S_m$ なる i について、 $s_i < a_i$ ならば $y_1 \leftarrow y_1 + 1$. $e_i > b_i$ ならば $y_2 \leftarrow y_2 + 1$. $s_i = a_i$ ならば $y_3 \leftarrow y_3 + 1$. $e_i = b_i$ ならば $y_4 \leftarrow y_4 + 1$.

ステップ 8 $y_1 > y_2 + y_4$ ならば、 $i \in S_m$ なる i について、 $s_i \leftarrow s_i + 1$, $e_i \leftarrow e_i + 1$ として、ステップ 6, 7 を計算させ、ステップ 9 へ進む。 $y_1 \leq y_2 + y_4$ かつ $y_2 > y_1 + y_3$ ならば、 $i \in S_m$ なる i について、 $s_i \leftarrow s_i - 1$, $e_i \leftarrow e_i - 1$ として、ステップ 6, 7 を計算させ、ステップ 10 へ進む。 $y_1 \leq y_2 + y_4$ かつ $y_2 \leq y_1 + y_3$ ならば終了。

ステップ 9 $y_1 > y_2 + y_4$ ならば、ステップ 8 に戻る。否ならば、終了。

ステップ 10 $y_2 > y_1 + y_3$ ならば、ステップ 8 に戻る。否ならば、終了。

ID-1 の個体表現から、三つのデコーディング法による表現型の作成例を付録に示す。

3.3.2 ID-2 の場合

ID-2 では、個体は製品の処理の順序を表している。したがって、ID-1 や以下に述べる ID-3 のように 1 機械ごとに独立に各製品の s_i および e_i を決めることはできない。参考文献 7) および 8) では、その計算法を 5 種類与えた。それらの概要は、以下に示すが、計算法の詳細は方法 I ~ III については参考文献 7) を、方法 IV および V については参考文献 8) を参照されたい。

[方法 I] 個体表現の製品番号順に、製品をその最早開始時刻よりも後で最も早く空く機械に割り当てる。

[方法 II] 個体表現の製品番号順に製品を、一定の順序に定められた機械に割り当てる。このとき、それまでに割り当てられた製品の処理時間と重なる場合には、 $(s_i + e_i)/2$ の小さい順になるように、それまでに割り当てられた製品の処理時間を前後にずらして重ならないようにする。

[方法 III] 方法 II を以下のように変更する。すなわち、機械の割り当ての際、それぞれの機械ごとに、製品 Y_j を割り当てることによって生ずる処理時間のずれを計算して、それが最も少ない機械に割り当てるようにする。

[方法 IV] 方法 II を以下のように変更する。すなわち、各機械に最後に割り当てられる製品 k の s_k および e_k は、この製品を割り当てることによって生ずるずれの合計が最も少なくなるように設定する。

[方法 V] 方法 III を以下のように変更する。すなわち、一番最後に機械に割り当てられる製品 k については、 s_k, e_k を、その製品が割り当てられることによって生ずるずれの合計が最も少なくなるように設定する。

3.3.3 ID-3 の場合

ID-3 では、まず、個体表現から機械 m で i 番目に処理する製品を表す配列 $I(m, i)$ を、次の手順で定める。

ステップ 1 $m = 1, 2, \dots, H$ について、 $S_m \leftarrow \phi$ 。

ステップ 2 $i = 1, 2, \dots, N$ について、 $m \leftarrow X_i$ ，
 $S_m \leftarrow S_m \cup \{i\}$ 。

ステップ 3 $m = 1, 2, \dots, H$ について、 $k_m \leftarrow 1$ 。

ステップ 4 $i = 1, 2, \dots, N$ について、 $Y_i \in S_m$ である m に
対して $I(m, k_m) \leftarrow Y_i$ ， $k_m \leftarrow k_m + 1$ 。

この結果、製品 i を機械 X_i に割り当て、その処理順序は、染色体 $\{Y_j\}$ に左から現れる順とすることになる。

つぎに、各製品の s_i および e_i を決めることになるが、前の段階で $I(m, i)$ を決めておくことにより、すべて機械ごとに独立に s_i および e_i を決めることができる。したがって、 s_i および e_i を決める下記のアルゴリズムは、1 機械に対して製品を割り当てるアルゴリズムである。ただし、 $q(m)$ は各機械に割り当てられた製品の数とする。

[方法 I] $I(m, i)$ の順に、最早処理開始時刻より後に割り当てる。

ステップ 1 $t \leftarrow 0, i \leftarrow 1$ 。

ステップ 2 $p \leftarrow I(m, i)$ 。

ステップ 3 $s_p \leftarrow \max(a_p, t)$ ， $e_p \leftarrow s_p + c_p$ ，
 $t \leftarrow e_p$ 。

ステップ 4 $i \leftarrow i + 1$ 。

ステップ 5 $i > q(m)$ ならば終了。否ならばステップ 2 に戻る。

[方法 II] $I(m, i)$ の順に、一定に定められた位置に割り当てる。このとき、それまでに割り当てられた製品の処理時間と重なる場合には、 $(s_i + e_i)/2$ の小さい順になるように、それまでに割り当てられた製品の処理時間を前後にずらして重ならないようにする。そのアルゴリズムは、 $x(m, l)$ および $r(m, l)$ をそれぞれ機械 m の l 番目に処理する製品の番号および処理の中間点とすれば、つぎのようになる。

ステップ 1 $i \leftarrow 1, k_m \leftarrow 0$ 。

ステップ 2 $j \leftarrow I(m, i)$ ， $s_j \leftarrow (b_j + a_j - c_j)/2$ ， $e_j \leftarrow s_j + c_j$ 。

ステップ 3 $l = 1, 2, \dots, k_m$ について $i_0 \leftarrow x(m, l)$ ， $r(m, l) \leftarrow (s_{i_0} + e_{i_0})/2$ 。
また、 $r(m, 0) \leftarrow -\infty$ ， $r(m, k_m + 1) \leftarrow +\infty$ 。

ステップ 4 $r(m, w-1) \leq (s_j + e_j)/2 \leq r(m, w)$ が成り立つ w をさがす。

ステップ 5 $l = k_m, k_m - 1, \dots, w$ について
 $x(m, l+1) \leftarrow x(m, l)$ ，
 $x(m, w) \leftarrow j$ 。

ステップ 6 $l \leftarrow w + 1$ ， $i_0 \leftarrow x(m, l)$ ， $i_1 \leftarrow x(m, l-1)$ 。

ステップ 7 $e_{i_1} > s_{i_0}$ が成り立つ間 $s_{i_0} \leftarrow e_{i_1}$ ，
 $e_{i_0} \leftarrow s_{i_0} + c_{i_0}$ ， $l \leftarrow l + 1$ ， $i_0 \leftarrow x(m, l)$ ， $i_1 \leftarrow x(m, l-1)$ を実行する。
 $l \geq k_m + 1$ ならばステップ 8 に進む。

ステップ 8 $l \leftarrow w - 1$ ， $i_0 \leftarrow x(m, l)$ ， $i_1 \leftarrow x(m, l+1)$ 。

ステップ 9 $s_{i_1} < e_{i_0}$ が成り立つ間 $e_{i_0} \leftarrow s_{i_1}$,
 $s_{i_0} \leftarrow e_{i_0} - c_{i_0}$, $l \leftarrow l - 1$, $i_0 \leftarrow$
 $x(m, l)$, $i_1 \leftarrow x(m, l + 1)$ を実行
 する. $l \leq 0$ ならばステップ 10 に
 進む.

ステップ 10 $k_m \leftarrow k_m + 1$, $i \leftarrow i + 1$. $i \leq q(m)$
 ならばステップ 2 に戻る. 否なら
 ば終了.

[方法 III] 方法 II を以下のように変更する. すなわち, 各
 機械に最後に割り当てられる製品 k の s_k およ
 び e_k は, この製品を割り当てることによって
 生ずるずれの合計がもっとも少なくなるように
 設定する. このとき, 方法 II のステップ 2 を
 以下のように変更する.

ステップ 2 $j \leftarrow I(m, i)$ とし, $i < q(m)$ なら
 ば $s_j \leftarrow (b_j + a_j - c_j)/2$, $e_j \leftarrow$
 $s_j + c_j$. 否ならば, $s_j = a_j, a_j +$
 $1, \dots, b_j - c_j$, $e_j = a_j + c_j, \dots, b_j -$
 $1, b_j$ について, $y(s_j) \leftarrow 0$ とした
 上で方法 II のステップ 3~10 を実行
 する. このとき, ステップ 7 で
 の $s_{i_0} \leftarrow e_{i_1}$ およびステップ 9 で
 の $e_{i_0} \leftarrow s_{i_1}$ の実行の前に, それ
 ぞれ $y(s_j) \leftarrow y(s_j) + e_{i_1} - s_{i_0}$ お
 よび $y(s_j) \leftarrow y(s_j) + e_{i_0} - s_{i_1}$ を
 実行する. その後, 変更を受けた
 s_j および e_j を元に戻す. そして,
 $y(s_j)$ が最も少ないときの s_j^0, e_j^0
 を s_j, e_j とする.

なお, (2) 式から目的関数値を計算するとき, 各個体表
 現の方法 I は, a_i よりも前に割り当てることはないので,
 (2) 式の第 2 項は必ず 0 である.

3.4 遺伝演算子

本論文では, 3.1 のステップ 2, 3 および 4 で実行する
 遺伝演算子をつぎのように定める.

まず淘汰規則は, ルーレット方式を確定的に適用する.
 すなわち, 各個体の適応度に比例して次世代で生き残る個
 体数を定める. 各世代における各個体 $p_j (j=1, 2, \dots, M)$
 の適応度関数 f_j を次式により定める⁵⁾.

$$f_j = U \min_{1 \leq q \leq M} Z_q + \max_{1 \leq q \leq M} Z_q - Z_j \quad (7)$$

ここに Z_j は個体 p_j の目的関数値である. また, U は淘汰
 速度を調節するための淘汰圧パラメータである.

つぎに交叉規則は, 以下のように定める. すなわち, 製
 品番号列を染色体とする ID-2 および ID-3 の記号列 $\{Y_j\}$
 は, 製品番号の重複を許さない順列である. したがって,

Fig. 1 Variation of suboptimal objective value with t

交叉規則としては order crossover または cycle crossover
 が有効であると考えられる. 試算の結果, cycle crossover
 が有効であることが確かめられたので^{7),8)}, ここでも cy-
 cle crossover を用いる. 一方, 機械番号列を染色体とす
 る ID-1 および ID-3 の記号列 $\{X_i\}$ は, 機械番号の重複
 使用を許す順列で, その意味で遺伝子配列に関して制約
 がない. したがって, 交叉規則としては最も単純な方法
 である 2-point crossover³⁾ を用いる.

突然変異規則は, 個体上でランダムに選んだ二つの遺
 伝子座の遺伝子を交換する方法を用いる.

GA パラメータとして, チューニングしなければならない
 のはつぎの五つの値である.

- M : 個体数
- t^* : 世代数
- U : 淘汰圧
- P_c : 交叉確率
- P_m : 突然変異確率

これらの値を決定するために, 製品数 $N=24$, 機械数
 $H=4$ の問題を 20 例作成して, この例題に GA を適用
 して GA パラメータのチューニングを行った. GA の性
 能として, 個体の収束特性と個体群の多様性維持の両面
 を考慮してパラメータ値を決定した. その結果, つぎの
 値が得られた.

$$\begin{aligned} M &= 100 \\ U &= 0.01 \\ P_c &= 1.0 \\ P_m &= \begin{cases} 0.001, & \text{for ID-1 and ID-2} \\ 0.0005, & \text{for ID-3} \end{cases} \end{aligned}$$

なお, 世代数 t^* は, 問題の規模に応じて変えることと
 した. Fig. 1 は, $(N, H) = (24, 4)$ の事例において, 上記の
 GA パラメータをとったときの, 世代に対する準最適目的
 関数値の変化を示す. また Fig. 2 は, 個体群中の異なる
 個体の個数 $d(t)$ の変化を示す. なお, これらの値は 20 事
 例の平均値である. Fig. 1 および Fig. 2 より, 準最適値へ
 の収束は速やかであり, そのとき個体群の多様性は, 個

Fig. 2 Variation of average number of different individuals with t

体群中異なる個体が60%以上あるので、十分であることがわかる。

4. 数値例

計算例として、製品数および機械数を、それぞれ $(N, H) = (16, 3), (24, 4), (50, 3), (70, 3), (100, 3)$ の5種類について、問題のパラメータ a_i, b_i, c_i に種々の値をランダムに与えて事例を作成した。事例の個数は、 $N \leq 24$ に対して20事例、 $N \geq 50$ に対して10事例である。

4.1 問題の特徴を表すパラメータ

本問題の特徴を表すパラメータとして、つぎの三つの量を考える。まず、各製品ごとの処理時間に対する割り当て可能領域の割合の平均値を自由度と呼び、次式で与える。

$$D = \frac{1}{N} \sum_{i=1}^N \left(\frac{b_i - a_i}{c_i} \right) \quad (8)$$

この値が大きいほど、各製品を処理するときの制約がゆるく、問題が簡単である。

つぎに、全体の製品処理可能領域に占める処理時間の合計の割合を余裕度と呼び、次式で与える。

$$F = \frac{\sum_{i=1}^N c_i}{\sum_{i=1}^H (b_i^+ - a_i^-)} \quad (9)$$

ただし、 a_i^- は a_i を昇順にソートしたものであり、 b_i^+ は b_i を降順にソートしたものである。(9)式の右辺の分母は、 H 台の機械が製品処理可能な最大時間間隔の合計とみなされる。よって、この値に対する製品処理時間の総和の割合である F の値が大きいほど問題が難しくなる。

最後に、製品処理の重なりをピークを表す値を集中度と呼び、次式で与える。

Table 1 Average values of D, F and P

(N, H)	$\bar{D}$	$\bar{F}$	$\bar{P}$
(16, 3)	2.749	1.060	2.700
(24, 4)	2.708	1.047	2.438
(50, 3)	5.287	1.104	4.467
(70, 3)	6.053	1.131	4.467
(100, 3)	7.946	1.087	4.533

$$P = \frac{\max_{k_1 \leq k \leq k_2 - 1} z(k)}{H} \quad (10)$$

ただし、

$$z(k) = \sum_{i=1}^N t_i(k) \quad (11)$$

$$t_i(k) = \begin{cases} 1, & a_i \leq k \leq a_i + c_i - 1 \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

$$k_1 = \min_i a_i \quad (13)$$

$$k_2 = \max_i (a_i + c_i) \quad (14)$$

であり、この P の値が大きいほど問題が難しくなる。

N および H が指定された各問題で、 D, F, P の平均値を Table 1 に示す。これより、各問題とも $\bar{F} \approx 1$ でかつ $\bar{P} > 1$ となり、処理時間を可能領域内にはほぼ割り当てられるものの処理の希望時間が重なり、 $N \geq 50$ の場合には処理能力の4倍の要求がピーク時に課せられていることがわかる。これに対して、 $N \leq 24$ の場合には集中度が2倍強で比較的緩やかに希望の処理時間が分布している。

4.2 実験結果とその考察

5種類の (N, H) の組み合わせについて、各個体表現およびデコーディング法を適用して計算した結果を Tables 2 ~ 4 に示す。なお、同表において (a) は平均目的関数値、(b) は1事例あたりの平均計算時間である。表中の平均目的関数値は、 $N \leq 24$ の場合は20事例、 $N \geq 50$ の場合は10事例をそれぞれ2回ずつ計算して得られた値の平均値である。なお、計算機として SUN SPARC station10 を用いた。世代数 t^* は、 $(N, H) = (16, 3), (24, 4), (50, 3), (70, 3), (100, 3)$ の場合でそれぞれ、120, 250, 800, 1200, 2000 に設定した。

同表の結果より、 $N \geq 50$ の集中度の大きい事例では、いずれの個体表現でも、方法 I の目的関数値は、他のデコーディング法による目的関数値の約2倍になる。これは方法 I が製品の処理の時間帯を後ろにしかずらせないことによると考えられる。しかし、 $N \leq 24$ の場合には集

Table 3 Computational results for ID-2

(a) Average objective value

(N, H)	Method I	Method II	Method III	Method IV	Method V
(16,3)	13.93	13.66	13.49	12.96	13.08
(24,4)	24.28	23.55	22.42	21.91	21.87
(50,3)	185.55	106.40	112.00	105.00	112.20
(70,3)	498.00	257.95	269.05	253.95	263.80
(100,3)	981.15	444.60	493.55	440.70	484.30

(b) Average CPU time [s]

(N, H)	Method I	Method II	Method III	Method IV	Method V
(16,3)	1.9	4.6	11.1	13.1	21.9
(24,4)	6.2	15.1	43.5	44.7	85.1
(50,3)	37.4	157.8	448.1	469.9	957.7
(70,3)	76.9	401.1	1198.2	1185.1	2068.4
(100,3)	182.0	1004.0	2825.3	3652.3	5578.7

Table 2 Computational results for ID-1

(a) Average objective value

(N, H)	Method I	Method II	Method III
(16,3)	14.13	14.25	12.90
(24,4)	24.65	22.82	21.98
(50,3)	200.90	127.55	119.60
(70,3)	554.20	315.75	303.45
(100,3)	1124.80	563.10	536.35

(b) Average CPU time [s]

(N, H)	Method I	Method II	Method III
(16,3)	3.2	2.4	3.2
(24,4)	10.3	7.5	10.3
(50,3)	131.1	54.2	90.8
(70,3)	361.8	119.7	186.8
(100,3)	1190.8	287.7	574.3

Table 4 Computational results for ID-3

(a) Average objective value

(N, H)	Method I	Method II	Method III
(16,3)	15.45	14.70	14.03
(24,4)	26.20	23.13	21.68
(50,3)	193.65	104.70	103.85
(70,3)	518.20	248.00	246.60
(100,3)	1084.40	426.30	422.95

(b) Average CPU time [s]

(N, H)	Method I	Method II	Method III
(16,3)	1.7	6.4	13.9
(24,4)	5.4	18.9	47.9
(50,3)	34.6	169.2	535.5
(70,3)	68.8	409.8	1159.5
(100,3)	161.5	1224.9	3740.1

中度が比較的小さいので、方法 I でもそれほど悪化していない。

各個体表現別では Table 2~4 の結果より、ID-1 では方法 III、ID-2 では方法 IV、ID-3 では方法 III が、それぞれデコーディング法として良い結果を得ていることがわかる。これらの方法はいずれも、方法 II に簡単な local search を付加したものである。それぞれの local search が有効に作用していることがわかる。もっとも、local search を用いることにより、計算時間も増えるので（同表(b) 参照）その優劣は精度と計算時間のトレードオフで評価さ

れるべきであろう。

ID-1, 2 および 3 を最良のデコーディング法で比較すると、ID-1 が全般的に悪い値となった。ID-1 は、各製品に割り当てる機械を GA から直接決め、それらをどの時間帯に割り当てるかはヒューリスティックで決めているが、そのヒューリスティック法の性能が必ずしも良くないのがこのような結果の原因と思われる。ID-2 と ID-3 の比較では、全般に ID-3 の方が良い結果を得た。これは、つぎのように説明される。すなわち、ID-2 では各製品に割り当てる機械をヒューリスティックで決め、製品の処理順序

Fig. 3 Gantt chart of initial solution

はGAによって定める．これに対して，ID-3では2種類の遺伝子列からなる個体を用いることにより，各製品に割り当てる機械と製品の処理順序の両方をGAで探索することになり，GAが有効に機能していることを表している．

以上を総括すると，個体表現の違いによって解の精度が影響を受けることがわかる．すなわち，個体の表現としてはもっとも簡単なID-1の精度がもっとも悪く，機械の割り当てと処理順序の両方を表すという意味で個体表現としては完全なID-3の精度が最良となった．

最後に， $(N, H) = (24, 4)$ の場合の一例について，GAの初期解（目的関数値30）のガントチャートをFig. 3に，最終世代に得られた準最適解（目的関数値4， $t^* = 250$ ）のガントチャートをFig. 4に示す．なお，GAの個体表現はID-3，デコーディング法は方法IIIを用いた．また，同図において，縦軸に製品番号，横軸に時刻をとっている．また，各製品ごとに縦線を入れた領域が区間 (a_i, b_i) を表し，黒または斜線の部分がそれぞれの機械に対する処理時間を表す．同図より，GAで求めた準最適解は初期解に比べて十分改善されていることがわかる．

Fig. 4に示された準最適解に対する個体表現は，次のようになる．

$$X: 1241312423 \dots 2 \quad (15)$$

$$Y: 123456781011 \dots 20 \quad (16)$$

Fig. 4 Gantt chart of suboptimal solution

ここに，個体表現ID-1では(15)式を，ID-2では(16)式を，またID-3では(15)および(16)式を用いる．

5. 結 言

作業割当問題において，GAを適用してその具体的な構成法を提案し，計算結果からその有効性を考察した．本論文では3種類の個体表現と，それらに対する複数のデコーディング法を提案した．規模の小さい比較的容易な問題に対しては，ID-3のように複数の染色体を個体表現に導入しなくても，ID-1やID-2で良い解が得られた．それに対して，規模の大きい難解な問題では，ID-3の有効性が示された．

ID-3では製品の処理順序と処理する機械の両方をGAで探索するので，ID-1やID-2のようにその一方のみをGAで探索し他方をヒューリスティックに決める方法に比べてID-3が優れているといえる．計算時間もID-2に比べてID-3は少し増加する程度なので，精度を要求する大規模な問題に対してはID-3を適用することが推奨される．

なお，通常の山くずしのように作業間に先行関係がある問題では，個体表現に制約条件が付加される．その場合には致死遺伝子が生ずることになり，その扱いに注意を払う必要がある．そのような問題への本解法の拡張については，今後の検討課題としたい．

参考文献

- 1) 茨木：組合せ最適化，産業図書，pp. 12-33 (1983)
- 2) 茨木：スケジューリング問題と計算の複雑さ；オペレーションズリサーチ，Vol. 39, No. 10, pp. 541-546 (1994)
- 3) D. E. Goldberg: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley (1989)
- 4) 西川，玉置：ジョブショップスケジューリング問題に対する遺伝アルゴリズムの一構成法；計測自動制御学会論文誌，Vol. 27, No. 5, pp. 593-599 (1991)
- 5) 飯間，三宮：遺伝アルゴリズムを用いた変形フローショップ問題の解法；システム制御情報学会論文誌，Vol. 6, No. 10, pp. 437-445 (1993)
- 6) 関根：PERT・CPM，日科技連，pp. 59-76 (1965)
- 7) 当麻，三宮，川合：遺伝アルゴリズムによる作業制当問題の解法；システム制御情報学会論文誌，Vol. 7, No. 10, pp. 433-435 (1994)
- 8) 当麻，飯間，三宮：作業制当問題における遺伝アルゴリズムの個体表現；第37回自動制御連合講演会前刷，pp. 339-340 (1994)

付 録

ID-1 の場合のデコーディングの例

3.3.1 で提案された三つのデコーディング法の手順を例示する．一例として， $N=6$ および $H=2$ のとき各製品の最早処理開始時刻 a_i ，最遅処理終了時刻 b_i および処理時間 c_i が，Table A1 のように与えられているとする．

いま，ID-1 の場合の個体が

$$X : 122112$$

のように表されると，製品の機械への割り当てはつぎのようになる．

$$S_1 = \{1, 4, 5\}$$

$$S_2 = \{2, 3, 6\}$$

Table A1 An example of product data

i	a_i	b_i	c_i
1	0	8	4
2	1	14	7
3	3	17	5
4	5	12	3
5	6	21	9
6	7	19	6

Fig. A1 Gantt chart for Method I

Fig. A2 Gantt chart for Method II

Fig. A3 Gantt chart for Method III

このとき、デコーディングの手順をガントチャートで示せば、方法 I, II および III はそれぞれ Figs. A1, A2 および A3 のようになる。ここに、各図とも横軸は時刻を、縦軸は機械番号（機械 1 および 2 を、それぞれ M_1 および M_2 と書く）を表す。また、棒線の左端および右端の数字は、それぞれ s_i および e_i を示し、棒線中央のカッコ付きの数字は製品番号を示す。各図とも上から下へ手順が進み、最下段に示す表現型が完成する。