中立突然変異型 GA による ジョブショップ・スケジューリング問題の解法*

大倉 和博^{*1}, 郷 東 末 和^{*2}, 上 田 完 次^{*1}

Solving Job-Shop Scheduling Problems Using a Genetic Algorithm with Neutral Mutations

Kazuhiro OHKURA, Tomokazu GOHTOH and Kanji UEDA

This paper presents a new genetic algorithm (GA)-based approach to job-shop scheduling problems (JSSPs). The proposed method adopts an extended GA, called GA with neutral mutations, which has originally been proposed in order to improve GA robustness for function optimization performance in solving various GA-hard problems. In order to apply this GA to JSSPs, a redundant genotype is constructed to have three layers: substring layer, operon layer and locus layer. The developmental process from the genotype to the phenotype is defined as the process of obtaining an active schedule using a mature function which includes the functionality of GT-algorithm. Computer simulations are conducted on two benchmark problems in order to compare the performance of the proposed approach with those of others.

Key Words: Genetic Algorithms, Job-Shop Scheduling Problems, Operons, Activation/Inactivation, Mature Function, GT-algorithm

1. は じ め に

ジョブショップスケジューリング問題(JSSP)はNP完全なスケジューリング問題の中でも最も難しい部類に属していることが知られている。生産システム・スケジューリング問題の典型であるこの問題は、実際の生産現場の問題に対して即時応用が期待されることから、その解法は、(1)迅速に適切な解を求められること、(2)解の信頼性が高い、すなわち優れた最適化能力をもつこと、が重要である。しかし、一般には、これを同時に満足するのは難しく、高速な解法であれば解の信頼度は低く、信頼度が高ければそれに見合う計算量が必要になる。

JSSPは、オペレーションズリサーチ分野で分岐限定法などの数理解析法を使ってたくさんの研究事例が報告されているが、遺伝的アルゴリズム(GA⁽⁶⁾)をこれに適用した研究もさまざまなグループによりなされている。GAからのアプローチでは、計算量を大きくするほど良質の解が求まる手法が提案されているほどには熟しておらず、まずは最適化能力を向上させるこ

とを目標に、共通のベンチマーク問題を使いながら各提案手法の検証が行われているのが現状である。

一方、一般の関数最適化ツールとしてGAを見た場合、その探索能力には限界があることが指摘されている。これは典型的には騙し問題や超多峰性関数問題などのいわゆる“GA困難”と呼ばれる問題で顕著に現れることが知られている。これを克服するために、いくつかの拡張法[例えば、文献(3), (6), (16)]が提案されており、典型的なGA困難な問題でそれらの最適化能力が比較検討されている。

その一つの拡張法に中立突然変異型GA⁽¹⁴⁾⁽¹⁵⁾がある。この拡張GAでは、表現型と遺伝子型とを陽に分離して、遺伝子型に目的関数に対する冗長性を導入するとともに二層からなる階層構造を定め、標準GAで用いられる交叉・突然変異のほかに、この遺伝子型に適した新しい遺伝的操作を加えて適応的な遺伝的変異が生成されるように拡張されている。本論文では、この拡張GAを使ったJSSPを解く新しいアプローチを示し、他の文献と同じベンチマーク問題を使って計算機実験を行いその最適化能力を調べる。

これまでのJSSPにGAを用いるアプローチでは、探索空間内の個体をバイナリ表現するか否かで大まかに二つに分けることができる。

* 原稿受付 1996年6月21日。

*¹ 正員, 神戸大学工学部 (〒657 神戸市灘区六甲台町1)。

*² 神戸大学大学院。

●バイナリ表現を使う方法

- 同一機械上での仕事間の作業先行関係を 0/1 で表現し、そのビット列を個体とする⁽¹³⁾。
- 選択グラフで問題を表現した後、それを 0/1 で表して個体とする⁽¹⁷⁾。

●非バイナリ表現を使う方法

- 各作業の完了時間を遺伝子とし、その集合を個体とする⁽¹⁸⁾。
- 仕事のガントチャートへの割り当て順に仕事を並べたものを個体とする⁽⁴⁾⁽⁸⁾。
- 各機械上の仕事の処理順序を並べて個体とする⁽⁹⁾。

バイナリ表現を用いた場合、GA の遺伝的操作として通常の交叉や突然変異が使われるが、非バイナリ表現では、通常の遺伝的操作を行うと致死個体が生成されてしまうので、それぞれの表現法にあわせた特別な遺伝的操作が使われることが多い。

JSSP の最適解は、必ず、どの作業も他の作業の開始時刻を遅らせることなしにその処理順序を早めることができないスケジュール、すなわちアクティブスケジュール⁽¹⁰⁾であることが知られているため、探索点空間をアクティブスケジュールに限定することが望ましい。Yamada ら⁽¹⁸⁾は常にアクティブスケジュールを生成する GA/GT 交叉を用いて、平野⁽⁸⁾は常にアクティブスケジュールを生成する個体表現法を使って、また、Kobayashi ら⁽⁹⁾は交叉の後に任意のスケジュールからアクティブスケジュールを生成する GT アルゴリズム⁽¹⁰⁾を適用してこれを達成している。本提案手法でも、Kobayashi らと同様に、GT アルゴリズムを表現型を生成する段階で用いることにする。

本論文は以下のように構成される。第2章では、JSSP を紹介する。第3章では、中立突然変異型 GA の JSSP への適用手法を示す。第4章では、ベンチマーク問題としてよく用いられる Muth and Thompson⁽¹¹⁾の 10×10 問題および 20×5 問題を使って計算機実験を行い、他の文献の結果と比較する。第5章では本論文をまとめる。

2. ジョブショップスケジューリング問題

典型的なジョブショップスケジューリング問題 (JSSP)⁽¹²⁾は、 N 個の仕事と M 台の機械に対して、

- 各仕事は同時に二つ以上の機械で処理されない。
- 各機械は同時に二つ以上の仕事を処理できない。
- 各仕事の各機械における処理は中断されない。
- 各仕事は各機械で 1 回だけ処理される。

という条件の下で、各仕事が処理される機械の順序と

各機械上での作業時間が与えられたときに総所要時間 (makespan) を最小にするスケジュールを求める問題である。一般的な JSSP では、上記の 4 番めの条件が不要であるが、簡便のためにこれを使うことにする。

3. 中立突然変異型 GA

3.1 中立突然変異型 GA の計算手順 中立突然変異型 GA では、個体表現法と遺伝的操作こそ異なるが、そのアルゴリズムは標準 GA がとる手続き⁽¹⁾と何ら変わらず、以下のように書くことができる：

- STEP 0：探索点集団を初期化する。
- STEP 1：各個体の遺伝子型から求められる表現型を目的関数で評価し、目的関数値を得る。その後、その個体の適応度を算出する。
- STEP 2：各個体の適応度をもとに次世代の集団を生成する。
- STEP 3：各個体の遺伝子型に遺伝的操作を作用させる。
- STEP 4：前もって定められた終了条件を満足するかどうか調べ、しなければ STEP 1 へもどる。

3.2 個体表現法

3.2.1 遺伝子型 中立突然変異型 GA を JSSP に適用するにあたって、その遺伝子型に図 1 に示すものを使う。これにあるように、遺伝子型は三層の階層構造をもつと考える。第 1 層は各機械での作業順序を表現する部分ストリング層、第 2, 3 層は、中立突然変異型 GA で定義されていた二つの層、すなわち、遺伝子座クラスタ (オペロンと呼ぶ) が存在するオペロン層、遺伝子座の構造が見える遺伝子座層と考える。各遺伝子座にある識別子と活性値は、対応する機械での仕事の処理順序と仕事の種類をそれぞれ表すものとする。これから、個体は以下のように記述することができる：

$$\text{string} = \{\text{substring}_1, \dots, \text{substring}_M\} \dots\dots\dots (1)$$

$$\text{substring}_m = \{\text{operon}_{m1}, \dots, \text{operon}_{mI_m}\} \dots\dots\dots (2)$$

$$\text{operon}_{mi} = \{\text{locus}_{mn} | n \in O_{mi}\} \dots\dots\dots (3)$$

$$\text{locus}_{mn} = \{\text{label}_{mn}, v_{mn1}, v_{mn2}, \dots, v_{mnd}\} \dots\dots\dots (4)$$

ここで、 substring_m は機械 M_m に対応し、 operon_{mi} は substring_m に含まれる i 番のオペロン、 n 番の遺伝子座とする。それぞれの添字は以下に示す範囲の値をとる：

$$m \in \{1, \dots, M\} \dots\dots\dots (5)$$

$$i \in \{1, \dots, I_m\} \dots\dots\dots (6)$$

$$n \in \{1, \dots, N\} \dots\dots\dots (7)$$

ここで、 I_m は substring_m に含まれるオペロンの数である。さらに、 O_m を substring_m に含まれている全遺

(a) Conceptual representation

(b) Representation divided by the structure and the value lists

Fig. 1 A proposed genotype

伝子座集合($=\{1, \dots, N\}$), O_{mi} を operon mi が含む遺伝子座の番号を要素とする集合として, $\cup_i O_{mi} = O_m$, $O_{mi} \neq O_{mj}$, $i \neq j$ であるとする。

式(4)にあるように, バイナリ探索空間での関数最適化を目的とした中立突然変異型 GA と同様, 各遺伝子座はその識別子 $label_{mn}$ と目的関数に評価される値(活性値) v_{mn1} と評価されない値(不活性値) $v_{mn2}, \dots, v_{mnd}$ を持ち, $v_{mnk} \in \{1, \dots, N\}$, $k=1, \dots, d$ の値をとるとする。また, 部分ストリング以下をその構造と値に分けて表記すれば, 図1(b)のようにかける。

3.2.2 個体の評価 この拡張 GA では遺伝子型と表現型を陽に区別する。そのため, STEP1では, 各個体の遺伝子型から表現型を計算し, それをもとに評価値を求めて各個体の適応度を計算する, というプロセスを経るとする。すなわち, 各個体の適応度は以下に示す過程を経て算出される(図2)：

$$P: x \rightarrow p \quad \dots\dots\dots (8)$$

$$V: p \rightarrow v \quad \dots\dots\dots (9)$$

$$F: v \rightarrow f \quad \dots\dots\dots (10)$$

ここで, x, p, v, f は, それぞれ個体の遺伝子型, 表現型, 評価値, 適応度である。また, P は成熟関数⁽⁷⁾であり, 遺伝子型から表現型を計算する手続きを示し, V は表現型からスケジュールの makespan を算出する目的関数, F は適応度を算出するための関数である。これを単純 GA での適応度計算に当てはめると, P は単なる恒等関数であり, F はスケールリングやランキング計算などに相当する。

3.2.3 未成熟表現型・成熟表現型 遺伝子型からスケジュールが算出できる各機械での仕事列を表す表現型を求める。この過程は個体の発生過程に相当す

Fig. 2 Development and evaluation process of an individual

る。このとき, 探索をアクティブスケジュールを表すものに限定するために, 発生を表す関数 P を P' と P'' の二つの関数の合成である考え, P'' を GT アルゴリズムとする。すなわち, 遺伝子型から P' によって仕事列を表す中間的な表現型 p' (未熟表現型と呼ぶ) を求め, それを P'' によってアクティブスケジュールを表す makespan を計算すべき表現型 p (成熟表現型と呼ぶ) に変換する。この発生過程は以下のように書ける：

$$P = P' \circ P'' \quad \dots\dots\dots (11)$$

$$P': x \rightarrow p' \quad \dots\dots\dots (12)$$

$$P'': p' \rightarrow p \quad \dots\dots\dots (13)$$

ここで, P' は以下の p' を生成する関数である：

$$P': x \rightarrow p' \quad \dots\dots\dots (14)$$

where

$$x = \text{string}$$

$$p' = \{\{v_{111}, \dots, v_{1N1}\}, \dots, \{v_{MN1}, \dots, v_{MNN1}\}\}$$

すなわち, 各遺伝子座の活性値のみを取出して部分ストリングの添字と遺伝子座の識別子に従って並べて

p' を生成する関数である。一般に、 p' によってスケジュールは定まるが、これはアクティブスケジュールではないし、実行可能性も保証されない。これに P'' を施せばアクティブスケジュールとなる仕事列を表す p が得られる：

$$P'' : p' \rightarrow p \dots\dots\dots (15)$$

where

$$p = \{\{\tilde{v}_{111}, \dots, \tilde{v}_{1N1}\}, \dots, \{\tilde{v}_{MN1}, \dots, \tilde{v}_{MNN}\}\}$$

以下では、特に断らない限り成熟表現型を単に表現型ということにする。

さらに STEP 2 では、生成された表現型 p の各要素 $\tilde{v}_{ij1}$ を遺伝子型内の対応する遺伝子座の活性値 v_{ij1} に書きうつす。

$$Q : p \rightarrow x \dots\dots\dots (16)$$

$$\text{where } \forall i, \forall j, v_{ij1} = \tilde{v}_{ij1}$$

これにより個体の獲得形質が遺伝することになるから、Lamarckism⁽⁷⁾ の導入であるといえる。

3.2.4 正当な個体・不当な個体・個体間の距離

以下で用いるために、表現型の諸性質を定義しておく。ある表現型に各機械のスケジュールに対応する部分ストリング内の遺伝子座の活性値にその機械で処理されなければならない仕事の番号がない場合、これを不当な個体、逆にすべてそろっているものを正当な個体と呼ぶことにする。

また、二つの個体の表現型 p^1, p^2 の各要素がそれぞれ $\tilde{v}_{mn1}^1, \tilde{v}_{mn1}^2$ と表されるとき、この個体間の距離 L を以下のように定める：

$$L = \sum_{m=1}^M \sum_{n=1}^N d_{mn} \dots\dots\dots (17)$$

where

$$d_{mn} = \begin{cases} 0 & \text{if } \tilde{v}_{mn1}^1 = \tilde{v}_{mn1}^2 \\ 1 & \text{otherwise} \end{cases} \dots\dots\dots (18)$$

すなわち、距離 L に対応する表現型内の各値の異なる数とする。

3.2.5 表現型の評価 JSSP での目的関数 V は各個体の表すスケジュールの総所要時間 makespan を計算する関数である。makespan は GT アルゴリズム P'' でアクティブスケジュールを表す仕事列を生成するときに同時に求められるため、この値を用いればよい。

3.3 遺伝的操作 3.2.1 項で定めた遺伝子型に対し、STEP 3 で遺伝的操作を作用させるが、各層に対して以下のような遺伝的操作を用意する：

- 部分ストリング層…交叉
- オペロン層…結合・分割

●遺伝子座層…重複・逆位・欠失・点突然変異

各遺伝的操作はその階層での単位を操作対象として各遺伝的操作が行われるとする。すなわち、交叉は部分ストリングを、オペロン層ではオペロンを、遺伝子座層では遺伝子座をそれぞれ単位として遺伝的操作を行うよう定める。

3.3.1 部分ストリング層での遺伝的操作 遺伝子型が遺伝子座の単純な列からなる単純 GA では、交叉で壊れない最小単位は遺伝子座であった。本拡張 GA では、それに相当する最小単位を部分ストリングであると考えて単純 GA と同様の交叉を行う。

- (1) 集団中から 2 個体を選び出す。
- (2) 各個体の部分ストリングを添字順に並べる。
- (3) 単純 GA での交叉と同様の操作を部分ストリングを最小単位として行う。

3.3.2 オペロン層での遺伝的操作 オペロン層には、図 3 に示すように、オペロンそのものを操作対象とする結合 g_{con} ・分割 g_{div} と呼ばれる二つの遺伝的操作を適用する。結合 g_{con} は二つのオペロンを一つのオペロンに、分割 g_{div} はその逆をする遺伝的操作であるとし、以下のように定義する：

$$\exists i, \exists j, i < j, i, j \in \{1, \dots, I_m\}$$

$$g_{con} : \text{operon}_{mi}, \text{operon}_{mj} \rightarrow \text{operon}'_{mi} \dots\dots\dots (19)$$

$$\left(\begin{array}{l} O'_{mi} = O_{mi} \cup O_{mj}, \\ \forall k > j, \text{operon}'_{m(k-1)} = \text{operon}_{mk}, \\ I'_m = I_m - 1 \end{array} \right)$$

$$g_{div} : \text{operon}_{mi} \rightarrow \text{operon}'_{mi}, \text{operon}'_{mI_m} \dots\dots\dots (20)$$

$$\left(\begin{array}{l} O'_{mi} \cup O'_{mI_m} = O_{mi}, O'_{mi} \cap O'_{mI_m} = \phi, \\ I'_m = I_m + 1 \end{array} \right)$$

3.3.3 遺伝子座層での遺伝的操作 遺伝子座層では、図 4 に示すように、確率的に選択される一つのオペロンに対して含まれるすべての遺伝子座を操作対象とする重複 g_{dup} 、欠失 g_{del} 、逆位 g_{inv} および点突然変異 g_{mut} の四つの遺伝的操作を用いるとする。このうち、 $g_{dup}, g_{del}, g_{inv}$ の一般形 g は以下のように書け

Fig. 3 Examples of the genetic operations in operon layer. (a) Connection between operon_{m2} and operon_{m3} , (b) Division on operon_{m2}

Fig. 4 Examples of the genetic operations in locus layer

る：

$$\exists i, \forall locus_{mn} \in operon_{mi},$$

$$\{label_{mn}, v_{mn1}, \dots, v_{mnd}\} \xrightarrow{g} \{label_{mn}, v'_{mn1}, \dots, v'_{mnd}\} \dots \dots \dots (21)$$

$$\text{where, } g \in \{g_{dup}, g_{del}, g_{inv}\}$$

さらに詳しく書けば以下ようになる：

$$g_{dup} : \{\forall k \geq 2, v'_{mn1} = v_{mn1}, v'_{mnk} = v_{mn(k-1)}\} \dots \dots \dots (22)$$

$$g_{del} : \{\forall k \geq 2, v'_{mn(k-1)} = v_{mnk}, v'_{mnd} = v_{new}\} \dots \dots \dots (23)$$

$$g_{inv} : \{\exists k \neq 1, v'_{mn1} = v_{mnk}, v'_{mnk} = v_{mn1}\} \dots \dots \dots (24)$$

g_{mut} は導入されている遺伝子型内の構造にかかわらず、任意の値 v_{mnk} を変更する関数として以下のように書ける：

$$g_{mut} : \{\exists k, v'_{mnk} = v_{new}\} \dots \dots \dots (25)$$

ここで、 v_{new} は $1, \dots, N$ のうちどれかランダムに選ばれる値とする。これらの定義から明らかなように、 g_{dup} では遺伝的操作を施した結果が表現型に現れないが、 g_{inv} , g_{del} および g_{mut} では、活性値が変更されて不当な個体が生成される可能性がある遺伝的操作である。そのため、これらの遺伝的操作が施された後、以下の手続きで正当なものに書き換える(図5参照)：

(1) 同一部分ストリング内に同じ活性値をもつ遺伝子座があるかどうか調べる。なければ、その個体は正当である。

(2) ある場合、その一つの活性値に注目して遺伝子座集合を A とする。また、その部分ストリングに含まれてない活性値の集合を B とする。

(3) A の各要素に以下の基準をもとにその活性値を書き換えないかどうか優先順位をつける。

- 遺伝的操作を受けた遺伝子座を優先
- 識別子の小さい遺伝子座を優先

Fig. 5 Cancellation algorithm for improper phenotype

これによって A の要素は完全に順序づけられる。

(4) A のうち最も優先順位の高いと計算されたものを A から除き、その活性値を確定させる。

(5) A の中で最も優先順位の高いと計算された遺伝子座を A から取出し、その活性値を B からランダムに選んだ値に書き換える。また、その値を B から取除く。

(6) 前手続き 5 を $A = \phi$ となるまで繰返す。

(7) 手続き 1 へ戻る。

4. 計算機実験

4.1 問題と諸設定 ここでは、JSSP に対して GA を用いる研究でベンチマーク問題としてよく取上げられる Muth and Thompson⁽¹¹⁾ の 10 仕事 10 機械からなる問題(10×10 問題)および 20 仕事 5 機械からなる問題(20×5 問題)を計算機上で解き、その最適化能力を検証する。

計算機実験では、二つの問題ともに GA の集団サイズ $|P_0|$ を 500 とし、各個体の適応度 f は、集団中のランクを用いて以下のように算出した：

$$f = (|P_0| - I_{rank}) / |P_0| \dots \dots \dots (26)$$

ここで、 I_{rank} はその個体の集団におけるランクである。選択過程では、エリート保存戦略とルーレット選択法を用いるが、集団は探索空間内での距離を指標として部分集団に分割し各部分集団で選択を行う分散エリート戦略と集団分割選択法⁽¹⁵⁾(エリート個体間距離 20, 最大部分集団数 100)を採用した。また、交叉は、近親交配を禁止(incest prevention⁽⁹⁾) (個体間距離 20 以下)した一様交叉とした。実験中の各パラメータは、予備実験を行って適切と思える値を設定した。それら

を表 1 に示す。また、常に表現型に影響しないオペロン層の遺伝的操作を用いない場合には二つの問題とも最適解が見つからなかった。これは、中立突然変異型 GA のオペロン層における自己組織化能力に基づく探索機能を使わないことになるため、十分な探索能力が発揮されないためと考えられる。

4・2 結果 本アプローチでの結果を示す前に、この二つのベンチマーク問題を解いた代表的な論文の結果を表 2 にまとめる。10×10 問題では、文献(9)の 100 試行中の 51 試行で 930 の最適値を求めたのが飛び抜けてよく、その他では 0~5%の成功率を記録しているに過ぎない。20×5 問題は 10×10 問題に比べてかなり難しい問題であるため、最適解を見つけることができたのは、平野⁽⁸⁾の集団サイズを 300 としたときの 300 試行中 1 試行のみで、その他の方法では見つかることができていない。本アプローチによる計算機実験では、これら二つの問題でそれぞれ 100 回ずつの試行を行った。各試行において 3 000 世代、すなわち $500 \times 3\,000 = 1.5 \times 10^6$ 回の個体評価を行っている。結果を図 6~9 に示す。

4・2・1 10×10 問題 10×10 問題の最良値頻度グ

Table 1 Parameters for simulations

parameters	value
population size	500
mutation rate	0.01
crossover rate	0.6
duplication rate	0.3
deletion rate	0.3
inversion rate	0.3
connection rate	0.05
division rate	0.05

ラフを図 6 に示す。これにあるように、最適値である makespan=930 の値は 51 回求められており、全試行の最良値の平均は 935.36 であった。最適値が見つからない試行でもその値に非常に近い値をとっており、最悪でも 956 であった。各試行の最良値の推移を示したのが図 7 である。最適値が最も早く求まった試行では 346 世代め (0.17×10^6 評価回数)で求まっている。これまでの研究で最もよい結果を示している文献(9)では、集団サイズを $|P_0|$ として $O(|P_0|^2)$ の計算量が必要な SXX と呼ばれる特殊な交叉方法を使っている。本アプローチは、遺伝的操作を多用するものの、その計算量は高々 $O(|P_0|)$ である。また、文献(9)の実験では、各試行で 3.0×10^6 回(本実験の 2 倍)個体評価した平均で 935 程度 [文献(9)図 6 から] であったことも考えれば、この実験結果はそれよりもよいものであるといえる。

4・2・2 20×5 問題 20×5 問題の最良値頻度グラフを図 8 に示す。これにあるように、makespan=1 165 の最適値が 100 試行中 5 回求められた。この問題で最適値を求められた他の方法は文献(8)(300 試行中 1 試行)だけであるから、本実験での最適解到達頻度はかなり向上したといえる。図 8 をこれ相当する文献(8)の図 9 のグラフと比較すれば、解の分布は全体に左にシフトしており、最悪解も 1 192 [文献(8)では 1 221] とかなり改善されている。しかし、文献(8)で 300 試行中 60 試行で最良値が 1 180 にとどまってしまうのと同様に、本実験でも 100 試行中 26 試行がその値にとどまってしまうことが観測された。図 9 は全試行のうちの最良と平均の推移を示したグラフである。これからわかるように、最も早いものでは 659 世代で最適解を発見したが、全体的な平均をとれば、1 178 と 1 180 にとどまってしまう試行の回数が多

Table 2 Comparison with the other GA based approaches

Problem	10 × 10		20 × 5	
	success ratio	evaluation counts	success ratio	evaluation counts
Nakano(1991) [13]	0/-	1.5×10^5	0/-	1.5×10^5
Yamada, et al.(1992) [18]	4/600	-	0/-	-
Davidor, et al.(1993) [2]	5/200	-	0/200	-
Fang, et al.(1993) [4]	0/-	1.5×10^5	0/-	1.5×10^5
Hirano(1995) [8]	4/300	1.2×10^6	1/300	0.9×10^6
Kobayashi, et al.(1995) [9]	51/100	3.0×10^6	0/-	-
The proposed method	51/100	1.5×10^6	5/100	1.5×10^6

†The symbol '-' means 'unknown'.

‡'A/B' means the optimal value is obtained A times among B trials.

Fig. 6 Frequency distribution of 100 trials for 10×10 problem Fig. 8 Frequency distribution of 100 trials for 20×5 problem Fig. 7 Best and average results of 100 trials for 10×10 problem Fig. 9 Best and average results of 100 trials for 20×5 problem

かったため、全試行の平均は 1180.34 であった。これからも、この問題は本拡張にとっても難しい問題であることがわかる。

5. おわりに

中立突然変異型 GA を使って JSSP を解く新しいアプローチを示し、他の GA による解法でよく用いられる Muth and Thompson の 10×10 問題と 20×5 問題を使って計算機実験を行い、その最適化能力を調べた。本アプローチによる計算機実験では、両方の問題で最適解が得られた。 10×10 問題では 100 試行中半数以上で、 20×5 問題に対しては 5 試行で最適解が求められた。この問題を扱った他の GA による JSSP 問題解法の文献結果と比較することにより、提案手法が優れた最適化能力をもつことが確認することができた。しかし、本アプローチは他の方法に比較してよい結果を残したものの、難しい 20×5 問題では 5% 程度の探索成功確率しか記録できなかったことから、今後さらなる改良が必要であると考えられる。

JSSP は本質的に難しい問題であって、個体表現法をいかに設計しようと、適応度空間はあまりに広大であまりにでこぼこしている。そのためか、バイナリ探索空間で探索能力の向上を目指した他の拡張 GA が JSSP の有効な解法になるという報告を著者らは知ら

ない。記号列で表現した個体であれば、個体表現方法に基づく交叉をいかに定義したところで、固定された遺伝子座間の位置関係が問題にならざるを得ないし、それを全く考慮しない一様交叉ではうまく積木が見つからない。また、多くの拡張 GA においては、いかに選択圧を緩めても世代が進むごとに結果的に集団中の多様性を失っていくだけであって、GA がいったん探索しないと決めた超平面には次世代以降入り込むことが非常に困難である。一方、中立突然変異型 GA では、オペロン層の存在によって遺伝子座間の位置関係は動的に変化し、また、遺伝子型内の選択圧のかからない内部構造と不活性域の多様性に基づいて、さまざまな大きさの遺伝的変異を生成すること、すなわち多様性の自律的制御が可能である。そのため、ある世代で失ったあるいは最初から集団中になかった超平面を以降の世代で再獲得可能になっている。提案手法の探索能力はこれらに基づくものと考えられる。

付録：GT アルゴリズム⁽¹²⁾

仕事 J_n の u 番めの処理を作業 $O_{n,u}$ といい、その作業が処理される機械を $M_{\mu,n,u}$ で表す。まず、一時的にそれぞれの作業の割り当てを考える。まだ割り当てられていない作業のうちで各仕事において最も早く処理されるべき作業のすべてをセット C 、それぞれの作業

$O_{n,u} \in C$ に対して(可能な)最早完了時間を $EC_{n,u}$ と定義する。このとき、以下のステップ(1)~(3)をすべての作業が割り当てられるまで繰返すとアクティブスケジュールが得られる:

(1) C のうちで最小の EC をもつ $O_{n*,u*}$ を探す。 $O_{n*,u*}$ と同じ機械 $M_{\mu_{n*,u*}}$ を共有し、また、 $O_{n*,u*}$ とプロセスが重なる $O_{n,u} \in C$ からなる作業のセット G を特定する。

(2) G のうちで機械 $M_{\mu_{n*,u*}}$ において作業 O_{n_s,u_s} を選び、この作業 O_{n_s,u_s} を EC_{n_s,u_s} に従って割り当てる。

(3) C と EC を更新する。

本アプローチでは、STEP 2 において選択される作業は、 p' が表す仕事列において、 G にうちで最も作業順序が早いものとしている。

文 献

- (1) DeJong, K. A. (Männer, R. and Manderick, B. 編), Are genetic algorithms function optimizer?; *Proc. of Parallel Problem Solving From Nature*, 2 (1992), 3-13, Elsevier Science Publishers.
- (2) Davidor, Y., Yamada, T. and Nakano, R. (Forrest, S. 編), The ECOlogical Framework II: Improving GA Performance At Virtually Zero Cost, *Proc. of The Fifth Int. Conf. Genetic Algorithms*, (1995), 171-176, Morgan Kaufmann.
- (3) Eschelmann, L. J. (Rawlins, G. J. E. 編), The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination *Foundations of Genetic Algorithms*, (1991), 265-283, Morgan Kaufmann.
- (4) Fang, H.-L., Ross, P. and Come, D. (Forrest, S. 編), A Promising Genetic Algorithm Approach to Jop-Shop Scheduling Rescheduling and Open-Shop Scheduling Problems, *Proc. of The Fifth Int. Conf. Genetic Algorithms*, (1993), 375-382.
- (5) Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, (1989), Addison-Wesley.
- (6) Goldberg, D. E., Korb, B. and Deb, K., Messy Genetic Algorithms: Motivation, Analysis, and First Results, *Complex Systems*, 3 (1989), 493-530.
- (7) Hart, W. E., Kammeyer, T. E. and Below, R. K. (Whitley, L. D. and Vose, M. D. 編), The Role of Development in Genetic Algorithm, *Foundations of Genetic Algorithms*, 3 (1995), 315-332, Morgan Kaufmann.
- (8) 平野広美, クラスタ平均化法を組み込んだ遺伝的アルゴリズムによるジョブショップスケジューリング問題の解法, *人工知能学会誌*, 10-5 (1995), 769-777.
- (9) Kobayashi, S., Ono, I. and Yamamura, M., An Efficient Genetic Algorithm for Job Shop Scheduling Problems, *Proc. of Sixth Int. Conf. Genetic Algorithms*, (1995), 506-511.
- (10) 今野浩・鈴木久敏, 整数計画法と組合せ最適化, (1982), 日科技連.
- (11) Muth, J. F. and Thompson, G. L., *Industrial Scheduling*, (1963), Prentice-Hall, Englewood Cliffs, N. J.
- (12) 鍋島一郎, スケジューリング理論, (1974), 森北出版.
- (13) Nakano, R. and Yamada, T. (Below, R. K. and Booker, L. B. 編), Conventional Genetic Algorithm for Job Shop Problems, *Proc. of The Fourth International Conference on Genetic Algorithms*, (1991), 474-479, Morgan Kaufmann.
- (14) 大倉和博・上田完次, 非定常関数最適化問題のための遺伝的アルゴリズム, システム制御情報学会論文誌, 8-6 (1995), 269-276.
- (15) 大倉和博・上田完次, 中立突然変異型 GA による騙し問題の最適化一探索戦略の創発一, 計測自動制御学会論文誌, 32-10 (1996), 1461-1469.
- (16) Sakanashi, H., Suzuki, K. and Kakazu, Y. (Davidor, Y. 編), Controlling Dynamics of GA through Filtered Evaluation Function, *Proc. of Parallel Problem Solving From Nature*, 3 (1994), 239-248, Springer-Verlag.
- (17) Tamaki, H. and Nishikawa, Y., A Paralleled Genetic Algorithm based on a Neighborhood Model and Its Application to the Jopshop Scheduling, *Parallel Problem Solving from Nature*, 2 (1992), 573-582.
- (18) Yamada, T. and Nakano, R., A Genetic Algorithm Applicable to Large-Scale Jop-Shop Problems, *Parallel Problem Solving from Nature*, 2 (1992) 281-290.