人 工 知 能 81－8
（1992. 3. 5）

遺伝アルゴリズムによるジョブショップ問題の新解法

山田武士　中野良平

NTT コミュニケーション科学研究所

遺伝アルゴリズム (GA) に基づく，ジョブショップ問題の解法を提案する．本方法では，最適解の候補である個体は，各作業の加工完了時刻のマトリックスをもちいて表現され，交叉オペレータは，Giffler & Thompson のアクティブスケジュール生成法 (GT 法) に基づいて定義される．本交叉法では，任意の 2 個体 (親) は交叉によって，必ずアクティブスケジュールを生成し，生成されたスケジュールは，親の形質を受け継いでいる．

実験の結果，提案する方法は従来難問とされてきた，Muth & Thompson の 10 × 10 問題や，ランダムに生成した，より規模の大きな 20× 20 問題に対し，良い結果を得た．

Genetic Algorithm Applicable to Large-Scale Job Shop Problems

Takeshi Yamada　Ryohei Nakano

NTT Communication Science Laboratories

2-2 Hikaridai Seika-cho Soraku-gun Kyoto 619-02 Japan

This paper proposes a method for solving job shop problems based on genetic algorithms (GA). In this method, individuals, candidates of optimal solutions, are represented by matrix of finishing times of operations, and crossover is defined over them based on Giffler & Thompson's active schedule generating method. Applying the proposed crossover to any two individuals which are called parents, an active schedule is obtained which inherits characteristics of parents.

Experiments showed proposed method can find good solutions for the difficult 10 × 10 problem of Muth & Thompson's benchmark and randomly generated 20 × 20 problems.

1 はじめに

遺伝アルゴリズム (Genetic Algorithms, 以下 GA と略す) は自然界の遺伝と淘汰の仕組みのモデル化を意図して考案された最適化アルゴリズムであり, ミシガン大学の J. Holland 等によって提唱された[Holland, 1975]. GA は, 一つの解ではなく, 解の集団を扱うこと, 問題の構造に依存しないこと, 動作が確率的であること, などの特徴を持ち, 近年, 複雑な解空間を持つ大規模な最適化問題に対しての, 有効な, しかも並列処理に適した解法としての期待が高まっている[Goldberg, 1986].

GA 研究において, 解を固定長のビット列にコード化して表現する方法が良く知られている. しかし, 例えば組合せ最適化問題の一つである, 巡回セールスマン問題 (以下 TSP と略) の解法として用いる場合は, 都市を巡回する順番のリストをそのまま個体として用いる場合が多い. 一般には問題に応じて適切な表現方法を考案しなくてはならない[Goldberg and Lingle, 1985; Whitley *et al.*, 1989].

一方ジョブショップ問題 (以下 JSP と略記) は TSP と同様, 組合せ最適化問題の一つであるが, より複雑な制約を持つ. 例えば, JSP をフローショップ型に制限すると, 各都市間の距離が非対称な場合の TSP に還元できることが知られている[Reddi and Ramamoorthy, 1972].

JSP の理論的研究は 1950 年代から開始され, 多くの理論的解法が既に与えられている. その多くは分枝限定法による解法であり, 解法の評価に, Muth & Thompson の良く知られたベンチマークを用いている[Muth and Thompson, 1963; Balas, 1969; McMahon and Florian, 1975; Barker and McMahon, 1985; Adams *et al.*, 1988; Carlier and Pinson, 1989]. しかし, 問題のサイズの増大に伴う組合せ爆発の壁が大きくは克服されていない.

JSP のように複雑な制約を持つ組合せ最適化問題の場合, 解を固定長のビット列にコード化する方法を見い出すのは一般に困難である. GA を JSP に適用する研究例は比較的最近のものであり, あまり多くない. [Davis, 1985; Liepins and Hilliard, 1987; Cleveland and Smith, 1989; Whitley *et al.*, 1989]. 我々は以前, 2 仕事間の各作業の処理の前後関係に着目した 2 進表現を考案し, それに基づく解法を提案した. そして Muth & Thompson のベンチマークに対して, 分枝限定法による解の品質に迫る GA 解を得ることができた[Nakano and Yamada, 1991].

本稿ではより自然な表現である, 記号表現による解法の可能性を追求する. 即ち, JSP 研究では古典的な, Giffler & Thompson のアクティブスケジュール生成法 (GT 法) [Giffler and Thompson, 1969]に基づいて, 個体の表現と, 交叉オペレータを新たに定義し, GA を用いた JSP の解法の構成法を示した. 本方法を用いることによって, 従来難問とされてきた, Muth & Thompson の 10 × 10 問題[Muth and Thompson, 1963] に対し, 最適解か, それに近い解を求めることができる. また, 最適解の知られていないより規模の大きな問題に適用し, ランダムサンプリングによる解の分布と比較したところ, 良好な結果を得ることができた.

2 ジョブショップ問題

2.1 問題の設定

今, M 台の機械上で N 個の仕事を加工する問題を考える. 各機械上での各仕事の加工のことを作業と呼ぶ. 一般には, 次のような前提条件が与えられる[鍋島, 1974]:

- 各機械は同時に 2 つ以上の仕事を処理できない.
- 各機械の種類は全て異なる.
- 作業の中断はない.
- 各仕事を加工する機械の順番 (技術的順序) は与えられている.
- 各作業の加工時間は与えられている.
- 各機械が各仕事を加工する順序 (仕事列) は未知である.

この時, 各機械上での各仕事の加工順序を適当に定めて, 全仕事を加工し終るまでの総所要時間 (makespan) を最小にする問題をここでは, ジョブショップ問題 (以下 JSP と略) という. 表 1に, Muth & Thompson のベンチマークの一つとして知られる, $M = 6, N = 6$ の場合の JSP の例を示す. 各仕事, および機械には番号が付与されている. 表 1において第 i 行 第 j 列は, 番号 i の仕事の j 番目の作業を表し, 括弧外の数字は機械の番号を, 括弧内の数字は作業の加工時間をそれぞれ表す.

JSP の解の表現法はいくつか知られている. 主なものは次の 3 つである.

A 各作業の処理開始 (完了) 時刻を全て指定する. (ガント・チャート)

B 各機械に対して, 処理する仕事の順番 (仕事列) を全て指定する.

C 同一機械上の各作業のペアについて, 処理の前後関係を指定する. (選択グラフ)

表 1: 6 × 6 ジョブショップ問題

仕事	機械の順番 (加工時間)					
1	3(1)	1(3)	2(6)	4(7)	6(3)	5(6)
2	2(8)	3(5)	5(10)	6(10)	1(10)	4(4)
3	3(5)	4(4)	6(8)	1(9)	2(1)	5(7)
4	2(5)	1(5)	3(5)	4(3)	5(8)	6(9)
5	3(9)	2(3)	5(5)	6(4)	1(3)	4(1)
6	2(3)	4(3)	6(9)	1(10)	5(4)	3(1)

表 2: 6 × 6 ジョブショップ問題の最適解

仕事	作業完了時刻					
1	1	4	22	37	45	55
2	8	13	23	38	48	52
3	6	10	18	27	28	49
4	13	18	27	30	38	54
5	22	25	30	42	51	53
6	16	19	28	38	42	43

表現法 A にしたがって, 表 1の最適解のひとつに対し, 各作業の処理完了時刻を表に表すと, 表 2のようになる. また, 表 2を視覚的に表現したものはガントチャートと呼ばれる (図 1). 但し通常は縦軸に機械をとるが, この図では縦軸に仕事をとっている.

表現法 A とスケジュールとは一対一に対応するが, 表現法 B に対応するスケジュールは, 作業開始時刻はいくらでも遅らせることができるので無限にある. しかし, 最適化の観点にたてば, 与えられた仕事列を満足する範囲でできるだけ早く処理するようなスケジュールを考えるだけで十分である. このようなスケジュールと表現法 B は一対一に対応する. このようなスケジュールはセミアクティブスケジュールと呼ばれる. 即ち, 各機械上の各作業を, 技術的順序と仕事列を満足させながらできるだけ前方に詰めることによって得られるスケジュールがセミアクティブスケジュールである.

2.2 アクティブスケジュール

今, さらに, 技術的順序を保ちながら, 各機械上で, 後の作業が前の作業を飛び越すことを許して (仕事列の変更) できるだけ前方に詰めることにより得られるスケジュールをアクティブスケジュールと呼ぶ.

明らかに, 最適スケジュールはアクティブスケジュールであるから, 最適化の観点に立てば, アクティブスケジュールのみ考えれば十分である.

2.3 Giffer & Thompson の方法 (GT 法)

Giffer & Thompson は 次のようなアクティブスケジュール生成法 (GT 法と呼ぶ) を考案した[Giffler and Thompson, 1969].

今, 仕事と機械にそれぞれ $1,\ldots,N$ 及び $1,\ldots,M$ までの番号がついているとする. 作業 (i,j) とは, 番号 i の仕事の j 番目の作業を指す. また, 作業 (i,j) の加工時間を $PT(i,j)$ とする.

1. 各作業を時間的順序にしたがって順にスケジュールすることを考える.

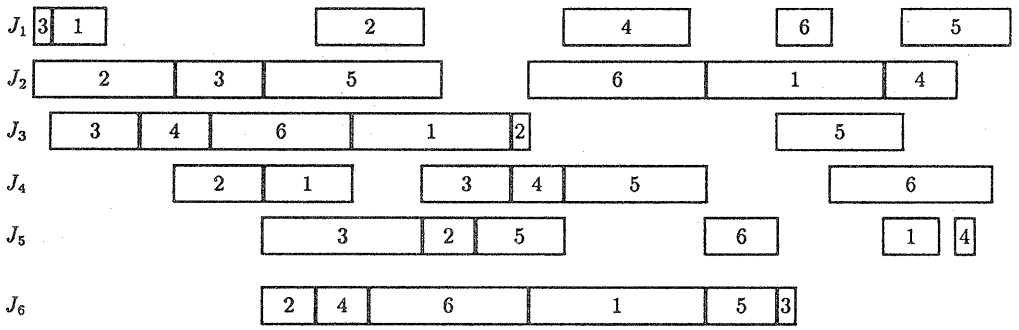

図 1: ガントチャートによる解の表示

各仕事の，未だスケジュールされていない最初の作業全体をカットと呼び，C であらわす．また，C の元 (i,j) に対し，最早完了時刻を，$EC(i,j)$ であらわす．C および，$EC(i,j)$ の初期値は次のようになる．

$C = \{(1,1),(2,1),\ldots,(N,1)\}$

$EC(i,j) = PT(i,j) \qquad ((i,j) \in C)$

2. C の元のうち，EC が最小の作業 (i_0,j_0) を求める．

   $EC(i_0,j_0) = min\{EC(i,j)|(i,j) \in C\}$

3. C の元のうち，(i_0,j_0) と同一マシン上で処理され，加工時間が重複するもの全体 G を求める．この G をコンフリクト集合と呼ぶ．

4. G の中からある作業 (i_c,j_c) を選び，(i_c,j_c) を $EC(i_c,j_c)$ 通りに加工する．

5. C から (i_c,j_c) を取り除き，代わりに技術的順序が (i_c,j_c) の直後である作業 (i_c,j_c+1) を加えたものを，C^+ とする．但し，(i_c,j_c) が技術的順序最後の作業である場合は何も加えない．

6. C^+ の元に対し，EC^+ を次のように定義する．

   5で新たに加わった作業 (i_c,j_c+1)

   (i_c,j_c+1) を処理する機械 m が，スケジュール済みの最後の作業を終えた時刻を T_m とすると，

   $EC^+(i_c,j_c+1)$
   $= max\{T_m, EC(i_c,j_c)\} + PT(i_c,j_c)$

   (i_c,j_c) と同一マシン上の他の作業 (i,j)

   $EC^+(i,j)$
   $= max\{EC(i,j), EC(i_c,j_c)\} + PT(i,j)$

   それ以外の作業 (i,j)

   $EC^+(i,j) = EC(i,j)$

7. $C = C^+, EC = EC^+$ として 2 へ．

上記 2 ～ 7 を 作業の数だけ繰り返すことによって一つのアクティブスケジュールが得られる．

上記 4の G において全ての組合せを考えることによって，全てのアクティブスケジュールを生成することができる[1]．

3 遺伝アルゴリズム

遺伝アルゴリズムでは，解くべき最適化問題の解を適当にコード化したものを考え，個体と呼ぶ．また，その個体の集合を考え，個体集団と呼ぶ．個体集団中の各個体はアルゴリズムを通じて変化するが，個体集団のサイズ (= 個体数) は不変とする．各個体は適応度と呼ばれる指標が与えられる．適応度は解の目的函数に基づいて評価される負でない自然数あるいは，実数である．

遺伝アルゴリズムは，次のような一連の処理の繰り返しからなる[Goldberg, 1986]．

1. 個体の集合である個体集団を初期化する．
2. 個体集団中の各個体を，目的函数に従って評価し，適応度を求める．
3. 各個体の適応度に応じて個体集団を再構成する．即ち，適応度の低い個体を個体集団から取り除き，逆に適応度の高い個体をその高さに応じて増やす．これを淘汰と呼ぶ．但し，個体集団のサイズは変えない．
4. 個体集団の各個体をランダムに二つずつペアにし，このペアに突然変異，交叉を施して新しい個体を作る．
5. 2 ～ 4 が繰り返しの単位であり，世代と呼ぶ．一世代の処理が終ると次の世代の処理に移るため，2 へ戻る．

このような一連の処理の繰り返しによって，個体集

[1] しかしその数は膨大である．

団は全体として適応度の高い個体の集団へと収束する．

3.1 解の表現法

遺伝アルゴリズムでは，解くべき最適化問題の解を，固定長のビット列にコード化して表現する方法が最も良く知られている．この表現方法を，2進表現と呼ぶ．

一方，2進表現の代わりに，各都市に番号を振り，巡回する順に都市の番号を並べたものを，個体とすることがある．このような表現法を，記号表現と呼ぶ．

3.2 突然変異と交叉

GAには，個体を部分的に変更して新たな個体を生成する操作が2種類あり，それぞれ突然変異と交叉と呼ばれている．前者は単に個体を局所的に変更する操作であるのに対し，後者はペアにした二つの個体から，新たに両者の性質を一部あわせ持つような個体を生成する．

2進表現を用いた解法の場合，突然変異は，個体の，ランダムに選んだ一部のビットを反転すること，交叉は，個体のペアのビット列の一部分を互いに交換することでそれぞれ定義される (図 2) [2][Goldberg, 1986].

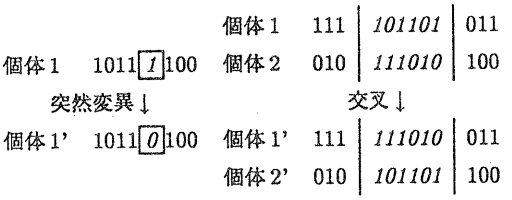

図 2: 突然変異と交叉

しかし，上述の TSP の場合のように記号表現を用いた場合，このような突然変異や，交叉はもはや適用できない．

一般に，突然変異や交叉は，コード化のしかたに応じて適切な定義を考案しなければならない．

[2]図2 の交叉は，切口を2箇所選んで，その間のビットを交換するため，2点交叉 (two-point crossover) と呼ばれている．

3.3 淘汰

淘汰は，解のコード化とは独立した操作である．一般には個体の適応度の良さに比例して，次の世代でのその個体の占める割合を決定する．例えば，今個体集団を P で表す．$p \in P$ の適応度を $f(p)(>0)$ とすると，その個体の次世代に占める割合は，$\frac{f(p)}{\Sigma_{q \in P} f(q)}$ である[Goldberg, 1986].

4 提案する方法

4.1 解の表現

今回提案する方法では，以下の記述を容易にするため，解を 2.1で述べた 3 つの表現法のうち表現法 A を用いて，表 2のようにマトリックスで表現する．マトリックスの (i,j) 成分が，仕事 i の j 番目の作業 (i,j) の作業完了時刻を表す．

個体 psn の (i,j) 成分を $psn_{i,j}$ と書くことにする．この時，各個体 psn が表すスケジュールの，総所要時間 S_{psn} は

$$\begin{aligned} S_{psn} &= max\{psn_{i,j} | 1 \leq i \leq N, 1 \leq j \leq M\} \\ &= max\{psn_{i,j_i} | (i,j_i) \in LastOps\} \end{aligned}$$

として求めることができる．但し $LastOps$ は，技術的順序が最後である作業全体をあらわす．

4.2 突然変異と交叉

今回提案する突然変異と交叉について説明する．

4.2で述べたように，交叉とは，ペアにした二つの個体から，新たな個体を生成する操作である．

ここで元になる2個体を mom, dad として，この2個体から新たに1個体を生成する次のような一連の手続きを考える．

今，新たに生成する個体を kid とする．kid を，各作業の処理完了時刻の早いものから順に次のようにスケジュールする．

1. まず，2.3で述べた GT 法の 1 ～ 3 を実行する．
   カット C，最早完了時刻 EC，コンフリクト集

合 G がそれぞれ得られる.

2. 次に GT 法における 4 に代わる次のような手続きによって, G 中から次に処理すべき作業 (i_c, j_c) を選択する.

   (a) 0 ～ 1 の乱数を発生させ, この乱数があらかじめ定められた値 $M_r \in [0,1)$ (突然変異発生率と呼ぶ) より小さければ, 次に処理する作業はコンフリクト集合 G から任意に一つ選び, (i_c, j_c) とする.

   (b) そうでない時は, まず確率 $\frac{1}{2}$ で mom, dad のうち一方を選ぶ. 今 mom が選ばれたとする.

   任意の作業 $(i,j) \in G$ に対し, mom の (i,j) 成分 $mom_{i,j}$ を調べ, $mom_{i,j}$ が一番小さい作業 (i_c, j_c) を求め, 次に処理する作業とする.

   $mom_{i_c,j_c} = min\{mom_{i,j} | (i,j) \in G\}$

   (c) (i_c, j_c) を $EC(i_c, j_c)$ 通りに加工する. 即ち, $kid_{i_c,j_c} = EC(i_c, j_c)$ となる.

   dad が選ばれた場合も同様にする.

3. GT 法の 5 ～ 7 を実行し, C, EC を更新する.

4. 以上の手続きを, 完全なスケジュールが得られるまで, 即ち, kid の値が全て定まるまで, 作業の数だけ繰り返し実行する.

このようにして新たな個体 kid が得られる.

実際には, mom, dad 1 ペアに対して手続き全体を 2 回適用することによって, 新たに 2 つの個体を得る. それを $kid1, kid2$ とする. これが今回提案する交叉である. なお突然変異は 2a として, 交叉に組み込まれている.

常に今まで一番適応度の高い個体が個体集団に存在するために, 次世代の個体 mom', dad' を, mom, dad, $kid1$, $kid2$ 中から次のように選ぶ.

必要なら名前を付替えて,

$S_{mom} \leq S_{dad}$

$S_{kid1} \leq S_{kid2}$ としておく.

if $(S_{kid2} < S_{mom})$

$mom' = kid1, dad' = kid2$

else

$mom' = mom, dad = kid1$

最後に mom, dad を mom', dad' で置き換える.

5 形質の継承

ここで元になる二つの個体 mom, dad と, この二つの個体を元に新たに生成された個体 kid との関係をもう少し詳しく見ることにする.

簡単のため突然変異のない場合 ($M_r = 0$の場合) を考える.

今, 4.2で述べた交叉が途中まですすんだとして, 4.2-2bの処理を考える. 2.1 で述べた 3 つの表現法のうち, 表現法 B によれば, スケジュールとは, 各機械上の仕事列を決定することである. G は同一機械上の作業から構成されることに注意すると, 4.2-2bは, G が表す機械 (M_G とする) 上の仕事列の構成の仕方を決めている. 今, $\frac{1}{2}$ で選ばれたのが mom であるとする. kid の M_G 上の仕事列を mom の, 同じく M_G 上の仕事列に, できるだけ近づけることを考える. このためには G の元を mom での順序となるべく同じ順序で加工すれば良い. 従って, G の元の中で, mom において一番早く加工される作業は, kid においても一番早く加工すべきである. よって 4.2-2b で述べたように,

$mom_{i_c,j_c} = min\{mom_{i,j} | (i,j) \in G\}$

となるような作業 (i_c, j_c) を選ぶことになる. 実際, 4.2-2bにおいて常に mom が選ばれた場合は, mom と kid は等しくなる.

ここで適当に選んだ mom と dad について, 生成された kid の各作業が, mom, dad のどちらから選ばれたものかを表にしてみる (表 3, 表 4). 但しここでは, スケジュール間の類似性を見やすくするために, スケジュールを, 2.1 で述べた 3 つの表現法のうちの, 表現法 B によって表す. 各行は, 特定の機械上で処理さ

れる各仕事の処理順序＝仕事列を表す．また，*mom* より選ばれた作業は m，*dad* より選ばれた作業は d を添字に付けて表し，G の元がただ一つの場合は，無条件にそれが選ばれるが，その時は □ で表すことにする．

以上より，提案する交叉は，各作業の処理順序を交叉前の2個体にできるだけ近づけるような操作になっていることがわかる．また，この交叉法は3.2で述べた2進表現における交叉法とは一見無関係のようであるが，図3のように，ランダムに選んだ数箇所のビットを交換する交叉方法を考えると良く似ていることがわかる．ちなみにこの交叉は，一様交叉 (uniform crossover) として知られている[Syswerda, 1989].

*mom*1 *1* 1 *1* *0* 1 1 *0* *1* 0 *1* 1

dad 0 *0* 0 *0* *1* 1 0 *1* *0* 1 *0* 0

交叉↓

*kid*1 1_m 0_d 1_m 0_d 1_d 1_m 1_m 1_d 0_d 0_m 0_d 1_m

*kid*2 0_d 1_m 0_d 1_m 0_m 1_d 0_d 0_m 1_m 1_d 1_m 0_d

図 3: 一様交叉

表 3: 交叉前の個体

mom	
機械	仕事列
1	1 3 4 2 5 6
2	6 2 4 1 3 5
3	3 1 2 5 4 6
4	6 3 1 4 2 5
5	2 5 4 1 3 6
6	3 2 5 6 1 4

dad	
機械	仕事列
1	4 1 3 6 5 2
2	4 6 2 5 1 3
3	3 5 1 4 2 6
4	3 6 4 1 5 2
5	5 4 2 6 3 1
6	6 3 4 5 1 2

表 4: 交叉後の個体

*kid*1	
機械	仕事列
1	1_m $4_□$ 3_m 6_d 2_m $5_□$
2	4_d 6_m 2_m 1_m $5_□$ $3_□$
3	3_m 1_m 5_m 2_m $4_□$ $6_□$
4	3_d $6_□$ 4_d $1_□$ 2_m $5_□$
5	2_m 5_d 4_m 3_m 1_m $6_□$
6	6_d $3_□$ 5_d 2_m 1_m $4_□$

*kid*2	
機械	仕事列
1	4_d $1_□$ 3_d 6_d 5_d $2_□$
2	4_d 6_m 2_m 5_d $1_□$ $3_□$
3	3_d 1_m 5_d 4_d $2_□$ $6_□$
4	3_d $6_□$ $4_□$ $1_□$ $5_□$ $2_□$
5	5_d 2_m 4_d 1_m 3_m $6_□$
6	6_d 3_m 5_m 2_m 1_m $4_□$

6 実験結果

6.1 Muth & Thompson ベンチマーク

Muth & Thompson のベンチマークは，以下の3つの問題からなる．

- 6 × 6 (6 仕事, 6 機械) 問題
- 10 × 10 (10 仕事, 10 機械) 問題
- 20 × 5 (20 仕事, 5 機械) 問題

これらの問題が発表以来 20 年以上かかって，段々と解法が改良され，解が改善されている．表5 に，分枝限定法 (BAB) を用いた今までの研究結果[Balas, 1969; McMahon and Florian, 1975; Barker and McMahon, 1985; Adams *et al.*, 1988; Carlier and Pinson, 1989] [3]及び，2進表現を用いた，我々の前回の結果[Nakano and Yamada, 1991]，そして今回の解法で求まったもっとも良い解を示す．なお，下界 (Lower Bound) の計算は,[Carlier and Pinson, 1989] による．

[3]但し，Adams 等の方法は近似解法である．

表 5: Muth & Thompson のベンチマーク

Papers	6 × 6	10 × 10	20 × 5
Balas1969	55	1177	1231
McMahon1975	55	972	1165
Barker1985	55	960	1303
Adams1988	55	930	1178
Carlier1989	55	930	1165
Nakano1991	55	965	1215
Yamada1992	55	930	1184
Optimal	55	930	1165
Lower Bound	52	880	1164

6 × 6 問題は比較的容易で, 個体集団のサイズを小さくしても, 短時間で, ほとんど必ず最適解が求まる.

10 × 10 問題は, Carlier 等が初めて総所要時間 930 を持つ解を求め, その解が最適であることを示した[Adams *et al.*, 1988; Carlier and Pinson, 1989]. 彼らによると, 分枝限定法で解く場合, 下界値 (880) と実際の最適値 (930) が離れているこの問題は, 同様の規模の問題の中でも特に難しい. しかし GA を用いた本方法では 320 回試行したうち,2 回 930 の解が求まった (図 4). 但し, 1 回の試行に要する時間は, Sparc Station 2 上の C プログラムで平均 10 分程度である.

一方, 20 × 5 問題は, 残念ながら最適解を求めることができなかった. 分枝限定法を用いた方法では, 下界値に基づく枝刈りによって探索範囲を絞るので, 下界値と実際の最適解が近い, 20 × 5 問題には適しているが, 逆に, 10 × 10 問題は 枝刈りに頼らない GA による解法が適していると考えられる.

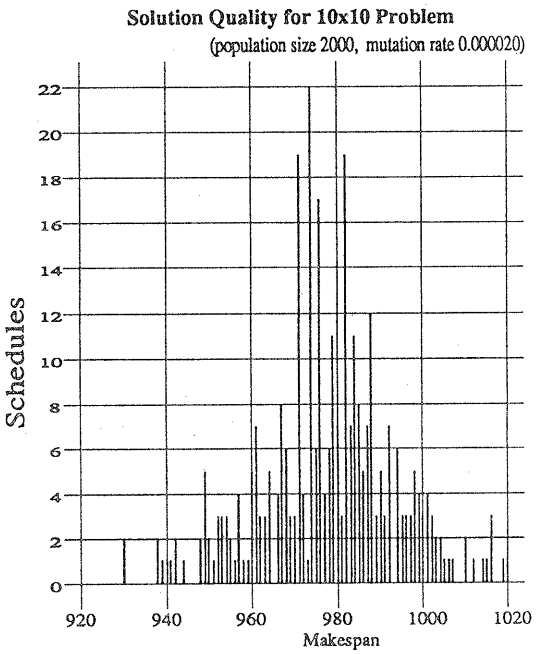

図 4: GA による 10 × 10 問題の解の分布

6.2 より規模の大きな問題

より規模の大きな問題への適用例として, 各作業の技術的順序と, 加工時間をランダムに発生させ, 生成した 仕事数 20, 機械数 20 の 4 つの問題に本方法を適用し, 問題毎に 400000 個のアクティブスケジュールをランダムに生成した結果と比較した. 但しいずれの問題も, 各作業の加工時間は区間 [10,50] よりランダムに選んだ. 得られた結果を表 6に示す. また, 表 6 問題 2 の結果をヒストグラムに表したものを図 5に示す.

表 6: ランダムサンプリングと GA の比較

20 × 20 問題	ランダム		GA
No.	平均	ベスト	求まった解の範囲
1	1318.6	1104	957 - 977
2	1356.8	1126	982 - 996
3	1313.5	1107	958 - 973
4	1414.3	1202	1064 - 1083

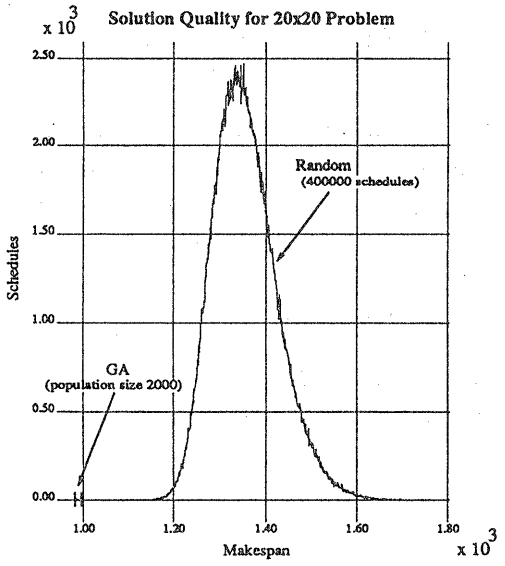

図 5: ランダムサンプリングと GA の比較 (ヒストグラム)

7 結び

本稿では, Giffler & Thompson のアクティブスケジュール生成法に基づいて, GA を用いた JSP の解法を示した. 実験の結果, 分枝限定法が適用できる規模の問題, および, さらに規模の大きな問題に対し, 良い品質の解を得ることができた.

分枝限定法は大きく分枝操作, 限定操作の二つから成り立っているが, 上述の GT 法は JSP を分枝限定法で解く場合の分枝操作として用いられることがある. 限定操作は, 分枝操作によって得られたいくつかの探索枝の候補から, 下界値を用いて枝刈りし, 更に下界値が最小の枝を優先的に探索するというヒューリスティクスを用いている. しかし問題の規模が大きくなると, 下界値と実際の値のギャップが大きくなり, 有効な枝刈りができない. このようなケースに, GA による近似解法が有効であると考えられる. 現段階ではまだ最適解の得られる頻度が低いので, 今後更にアルゴリズムに改良を加え, 効率の良い解法を目指したい.

また今回提案した方法は, 分枝限定法における分枝操作を GA に利用したという見方ができる. したがって他の組合せ最適化問題に対しても対応する分枝限定法の解法をそのまま利用することによって GA を構成することが可能であると考えられ, 今後組合せ最適化問題に GA を適用する際の指針になると期待できる.

参考文献

[Adams *et al.*, 1988] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Manage. Sci.*, 34(3):391–401, 1988.

[Balas, 1969] E. Balas. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Oper. Res.*, 17:941–957, 1969.

[Barker and McMahon, 1985] J.R. Barker and G.B. McMahon. Scheduling the general job-shop. *Manage. Sci.*, 31(5):594–598, 1985.

[Carlier and Pinson, 1989] J Carlier and E. Pinson. An algorithm for solving the job-shop problem. *Manage. Sci.*, 35(2):164–176, 1989.

[Cleveland and Smith, 1989] G.A. Cleveland and S.F. Smith. Using genetic algorithms to sched-

ule flow shop releases. In *Proc. 3rd Int. Conf. on Genetic Algorithms and their Applications (Arlington, Va.)*, pages 160–169, 1989.

[Davis, 1985] L. Davis. Job shop scheduling with genetic algorithms. In *Proc. 1st Int. Conf. on Genetic Algorithms and their Applications (Pittsburgh, PA)*, pages 136–140, 1985.

[Giffler and Thompson, 1969] B. Giffler and G.L. Thompson. Algorithms for solving production scheduling problems. *Oper. Res.*, 8:487–503, 1969.

[Goldberg and Lingle, 1985] D.E. Goldberg and R.Jr. Lingle. Alleles, loci, and the traveling salesman problem. In *Proc. 1st Int. Conf. on Genetic Algorithms and their Applications (Pittsburgh, PA)*, pages 154–159, 1985.

[Goldberg, 1986] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Reading,Mass., 1986.

[Holland, 1975] J.H. Holland. *Adaptation in Natural and Artificial Systems.* Unuv. of Michigan Press, 1975.

[Liepins and Hilliard, 1987] G.E. Liepins and M.R. Hilliard. Greedy genetics. In *Proc. 2nd Int. Conf. on Genetic Algorithms and their Applications (Cambridge, MA)*, pages 90–99, 1987.

[McMahon and Florian, 1975] G. McMahon and M. Florian. On scheduling with ready times and due dates to minimize maximum lateness. *Oper. Res.*, 23(3):475–482, 1975.

[Muth and Thompson, 1963] J.F. Muth and G.L. Thompson. *Industrial Scheduling.* Prentice-Hall, Englewood Cliffs, N.J., 1963.

[Nakano and Yamada, 1991] R. Nakano and T. Yamada. Conventional genetic algorithm for job shop problems. In *Proc. 4th Int. Conf. on Genetic Algorithms and their Applications (San Diego, CA.)*, pages 474–479, 1991.

[Reddi and Ramamoorthy, 1972] S.S. Reddi and C.V. Ramamoorthy. On the flow-shop sequencing problem with no wait in process. *Operational Research Quarterly*, 23(3):323–331, 1972.

[Syswerda, 1989] G. Syswerda. Uniform crossover in genetic algorithms. In *Proc. 3rd Int. Conf. on Genetic Algorithms and their Applications (Arlington, Va.)*, pages 2–9, 1989.

[Whitley *et al.*, 1989] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In *Proc. 3rd Int. Conf. on Genetic Algorithms and their Applications (Arlington, Va.)*, pages 133–140, 1989.

[鍋島, 1974] 鍋島一郎. スケジューリング理論. 森北出版株式会社, 日本, 1974.