論文

SA法による ファジィ・フローショップ・スケジューリング[†]

石渕 久生* 御前 伸太* 田中 英夫*

本研究では、まず、ファジィ納期という概念をフローショップ・スケジューリング問題に導入することにより、ファジィ・フローショップ・スケジューリング問題を定式化する。この問題は、各ジョブのファジィ納期に対する満足度の最小値を最大化するスケジューリング問題として定式化される。次に、この問題に対してシミュレーティッド・アニーリング法 (SA 法) を適用し、ランダムに生成された例題に対する数値実験結果を示す。数値実験により、温度と呼ばれる制御変数の設定を適切に行った場合での SA 法の性能が、局所探索法よりも良いことを示す。さらに、SA 法の探索を効率的に行うために、目的関数の修正方法を提案し、例題に対する数値実験により、提案方法の有効性を示す。最後に、同じ例題に対して改良型 SA 法を適用し、制御変数の設定方法によらず良好な解が得られることを示す。

キーワード：シミュレーティッド・アニーリング、ファジィ・スケジューリング、ファジィ納期、フローショップ・スケジューリング、ファジィ組合せ最適化

1. はじめに

ファジィ目標とファジィ制約を統合したファジィ意思決定という概念が Bellman & Zadeh[1]により提案されて以来、意思決定問題へのファジィ理論の応用が盛んに行われている。特に、数理計画問題をファジィ化したファジィ数理計画問題に関する研究は、田中ら[2]や Zimmermann[3]の研究を契機とし、種々の定式化や解の概念の定義が行われている(坂和[4]や乾口[5,6]参照)。ファジィ数理計画問題では、係数の不明確さや不等号関係のあいまいさを考慮した定式化が行われており、ほとんどの問題が線形計画問題などの取り扱いの容易な数理計画問題に帰着されている。

一方、スケジューリング問題などの組合せ最適化問題は、そのほとんどが効率的な解法の存在しない NP 完全問題であり、この分野へのファジィ

理論の応用は非常に少ない[7]。Ishii ら[8]では、通常のスケジューリング問題における納期をファジィ化したファジィ納期という概念を導入することにより、2 機械オープンショップ問題と並列型一様機械スケジューリング問題をファジィ化し、その解析的な解法を提案している。

本研究では、効率的な厳密解法の存在しない一般的な m 機械 n ジョブ・フローショップ・スケジューリング問題にファジィ納期を導入し、ファジィ・フローショップ・スケジューリング問題を定式化する。次に、組合せ最適化問題の一般的解法であるシミュレーティッド・アニーリング法 (Simulated Annealing 法: SA 法[9,10]) をファジィ・フローショップ・スケジューリング問題に適用し、数値実験により、その有効性を検討する。このとき、SA 法が局所探索法よりも有効な手法であることを示すとともに、SA 法により効率的な探索が行えないような問題が存在することも明らかにする。このような問題に対処するため、ファジィ・フローショップ・スケジューリング問題の目的関数を修正する方法を提案する。さらに、

[†] Simulated Annealing for Fuzzy Flow Shop Scheduling
Hisao ISHIBUCHI, Shinta MISAKI and Hideo TANAKA

* 大阪府立大学 工学部 経営工学科
Department of Industrial Engineering, College of Engineering, University of Osaka Prefecture

筆者らによる改良型SA法[11,12]が、ファジィ・フローショップ・スケジューリング問題に対しても有効であることを示す。

なお、SA法のファジィ組合せ最適化問題への応用は、坂和ら[13]においても行われている。また、坂和ら[14]では、ニューラルネットを用いたファジィ組合せ最適化が試みられている。

2. ファジィ・フローショップ・スケジューリング

2.1 フローショップ・スケジューリング

フローショップ・スケジューリング問題は、 m 個の工程(機械)により順に加工される n 個のジョブの加工順序を決める問題である。本研究では、 n 個のジョブが全ての工程で同じ順序で加工されるとし、この加工順序をベクトル $\mathbf{x}=(x_1, x_2, \dots, x_n)$ で表すことにする。ただし、要素 x_k は、 k 番目に加工されるジョブの番号を表わしている。

フローショップ・スケジューリング問題では、工程 i におけるジョブ j の加工時間を $t_p(i, j)$ とすると、加工順序 $\mathbf{x}$ で加工した場合での各工程における各ジョブの加工終了時間 $t_c(i, j)$ は次のように計算される。

$$t_c(1, x_1) = t_p(1, x_1) \quad (2.1)$$

$$t_c(i, x_1) = t_c(i-1, x_1) + t_p(i, x_1) \\ \text{for } i=2, 3, \dots, m \quad (2.2)$$

$$t_c(1, x_k) = t_c(1, x_{k-1}) + t_p(1, x_k) \\ \text{for } k=2, 3, \dots, n \quad (2.3)$$

$$t_c(i, x_k) = \max\{t_c(i-1, x_k), t_c(i, x_{k-1})\} \\ + t_p(i, x_k) \\ \text{for } i=2, 3, \dots, m; k=2, 3, \dots, n \quad (2.4)$$

ここで、各工程での各ジョブの加工時間 $t_p(i, j)$ は既知であり、加工順序に依存しないものとする。

各ジョブの最終的な加工終了時間は、最終工程 m における加工終了時間 $t_c(m, x_k)$ である。フローショップ・スケジューリング問題は、なんらかの評価基準を最大化あるいは最小化するような加工順序 $\mathbf{x}$ をもとめる問題である(詳細は、例えば、文献[15]-[19]参照)。

2.2 ファジィ納期

ファジィ納期は、各ジョブの加工終了時間に対する満足度を表すファジィ集合である[8]。本研究では、ジョブ j に対するファジィ納期のメンバーシップ関数を $\mu_j(t)$ で表すことにする。ここで、 $\mu_j(t)$ は、ジョブ j の加工終了時間が t の場合での満足度である。したがって、加工順序 $\mathbf{x}$ で加工した場合での各ジョブの満足度は次のようになる。

$$\text{ジョブ } x_k \text{ の満足度: } \mu_{x_k}(t_c(m, x_k)) \quad (2.5)$$

ファジィ・フローショップ・スケジューリング問題の目的として、最小満足度を最大化することを考えれば、目的関数 $f(\mathbf{x})$ を次のように定式化することができる。

$$f(\mathbf{x}) = \min\{\mu_{x_k}(t_c(m, x_k)) \mid k=1, 2, \dots, n\} \\ \longrightarrow \text{最大化} \quad (2.6)$$

したがって、ファジィ・フローショップ・スケジューリング問題は、上式で定義される $f(\mathbf{x})$ が最大になるような加工順序 $\mathbf{x}$ を求める問題である。ジョブ数 n が少ない場合では、 $n!$ 通りの加工順序の組合せに対して $f(\mathbf{x})$ を求めることにより、最適解を得ることができる。しかし、 n が大きい場合では、最適解を求めることは計算時間の点から現実的ではないため、効率的に良好な近似解を求めることが必要となる。

2.3 メンバーシップ関数の設定方法

通常のスケジューリング問題では、クリスプな納期が假定されている(制約条件として納期遅れが許されていない場合)。クリスプな納期は、図1のようなメンバーシップ関数により定義されるクリスプ集合により表現される。このメンバーシップ関数は、次のように書くことができる。

$$\mu_j(t) = \begin{cases} 1; & 0 < t \leq t_d \\ 0; & t_d < t \end{cases} \quad (2.7)$$

すなわち、納期 t_d までに加工が終了すれば完全に満足(満足度1)であり、納期 t_d を過ぎれば完全に不満足(満足度0)となる。なお、式(2.7)では、納期は R^+ (正の実数の集合)上で定義されていると假定している。

一方, Ishii ら[8]により導入されたファジィ納期は, 図2のような線形のメンバーシップ関数により定義されるファジィ集合により表現される. このメンバーシップ関数は, 次のように書くことができる.

$$\mu_j(t) = \begin{cases} 1; & 0 < t \leq t_d \\ 1 - (t - t_d) / (t_f - t_d); & t_d < t \leq t_f \\ 0; & t_f < t \end{cases} \quad (2.8)$$

図2では, t_d までは満足度が1であり, t_d を過ぎれば徐々に満足度が低下し, t_f 以降は満足度が0となっている. 直観的に言えば, t_d は“これより早ければ完全に満足”という納期であり, t_f は“これより遅ければ完全に不満足”という納期である. 図2のメンバーシップ関数は, この2点を直線で補間したものである.

ファジィ納期のメンバーシップ関数は, 各ジョブの加工終了時間に対する意思決定者の主観的な満足度を表すものである. そのため, ファジィ納期のメンバーシップ関数は, 種々の形状を取り得ると考えられる. そこで本研究では, 3章以降において, ファジィ・フローショップ・スケジューリング問題に対して, メンバーシップ関数の形状を仮定しない一般的な解法を示す.

ファジィ納期のメンバーシップ関数は, 意思決定者により主観的に決定されるべきものであるが,

図1 クリスポな納期を表すメンバーシップ関数

図2 ファジィ納期を表す線形のメンバーシップ関数

連続関数として厳密に定義することは困難なことが多い. そこで, 本節では, 意思決定者の主観的な満足度にできるだけ一致したメンバーシップ関数を, 簡単な補間手法により設定する方法をいくつか提案する. なお, 上述のように, 3章以降で示す解法がメンバーシップ関数の形状を仮定しない手法であるので, ここで述べる設定方法以外の方法でメンバーシップ関数を設定してもよい.

まず, 最も簡単な線形のメンバーシップ関数は, 次のような手順で設定することができる.

[設定方法 A]

Step 1: ジョブの加工終了時間に関して, “これより早ければ完全に満足”という納期 t_d と “これより遅ければ完全に不満足”という納期 t_f を設定する.

Step 2: 図2のように式(2.8)を用いて2点 d と f の間を補間し, メンバーシップ関数を設定する.

この設定方法では, 全てのジョブに対して, 2点 d と f との間で単純な線形補間が用いられている(図2参照). しかし, ジョブごとに納期 t_d の重要度に違いがある場合では, 非線形関数による補間が有効となる. 例えば, 納期 t_d が非常に重視されるジョブでは, 点 d を過ぎれば急激に満足度が低下することが考えられる(図3). 一方, 納期 t_d よりも納期 t_f が重視されるジョブでは, 点 f 近くでも満足度が高いことが考えられる(図4). すなわち, 納期 t_d が重視されるジョブに対しては, 図3のような下に凸の補間を行い, 納期 t_f が重視されるジョブに対しては, 図4のような上に凸の補間を行うことが有効であると考えられる.

図3や図4のメンバーシップ関数は, 次のように書くことができる.

$$\mu_j(t) = \begin{cases} 1; & 0 < t \leq t_d \\ \{1 - (t - t_d) / (t_f - t_d)\}^u; & t_d < t \leq t_f \\ 0; & t_f < t \end{cases} \quad (2.9)$$

ここで、 u は正の実数である。点 d と点 f の間の補間は、 $0 < u < 1$ のときは上に凸となり、 $1 < u$ のときは下に凸となる。例えば、図3と図4は、それぞれ、 $u=2$ と $u=0.5$ のときのメンバーシップ関数を図示したものである。なお、式(2.8)のメンバーシップ関数は、式(2.9)において $u=1$ と設定したものである。したがって、式(2.9)のメンバーシップ関数は、式(2.8)のメンバーシップ関数を一般化したものと考えることができる。

図3や図4のような非線形のメンバーシップ関数を設定する手順をまとめれば、次のようになる。
[設定方法 B]

Step 1: 設定方法 A と同じ。

Step 2: 正の実数 u の値を設定する。

Step 3: 図3や図4のように、式(2.9)を用いて与えられた2点 d と f を補間し、メンバーシップ関数を設定する。

設定方法 B における u の値は、各ジョブにおける納期 t_d の重要度と解釈することも可能である。すなわち、納期 t_d が重視されるジョブでは、 u の値を大きく設定することにより、図3のような下に凸のメンバーシップ関数が得られる。逆に、納期 t_d よりも納期 t_f が重視されるジョブでは、 u の値を小さく設定することにより、図4のような上に凸のメンバーシップ関数が得られる。

図3 非線形補間の一例(下に凸)

図4 非線形補間の一例(上に凸)

設定方法 B では、正の実数 u を直接指定する方法を用いたが、次のように満足度が 0.5 となる点を指定することにより、 u の値を間接的に指定することもできる。

[設定方法 C]

Step 1: 設定方法 A と同じ。

Step 2: 2点 t_d と t_f の間に、満足度が 0.5 となる加工終了時間 t_e を設定する。さらに、 t_e において式(2.9)のメンバーシップ関数が 0.5 になるように、 u の値を求める。すなわち、次式が成り立つような u の値を求める。

$$\mu_j(t_e) = \{1 - (t_e - t_d) / (t_f - t_d)\}^u = 0.5 \quad (2.10)$$

Step 3: 設定方法 B と同じ。

この設定方法では、図5に示すように、満足度が 0.5 となる納期 t_e を設定することにより、非線形のメンバーシップ関数が得られる。なお、設定方法 C において

$$t_e = (t_d + t_f) / 2 \quad (2.11)$$

と設定すれば、 $u=1$ となり、図2のような線形のメンバーシップ関数が得られる。

さらに、加工終了時間が早すぎても満足度が減少するような状況を考えれば、ファジィ納期のメンバーシップ関数は、例えば、図6のようになる。図6は、点 a, b, c と点 d, e, f の補間を設定方法 C と同様の方法で行うことにより、メンバーシップ関数の設定を行ったものである。したがって、図6のメンバーシップ関数は、次のように書くことができる。

図5 設定方法 C によるファジィ納期のメンバーシップ関数

$$\mu_j(t) = \begin{cases} 0; & 0 < t \leq t_a \\ \{(t-t_a)/(t_c-t_a)\}^{u'}; & t_a < t \leq t_c \\ 1; & t_c < t \leq t_d \\ \{1-(t-t_d)/(t_f-t_d)\}^u; & t_d < t \leq t_f \\ 0; & t_f < t \end{cases} \quad (2.12)$$

ここで、 u' と u は、正の実数である。このようなメンバーシップ関数は、次のような手順により設定することができる。

[設定方法 D]

Step 1: 図6のように、ジョブの加工終了時間に関して、満足度が1となる区間 $[t_c, t_d]$ と満足度が正となる区間 (t_a, t_f) を設定する。さらに、 t_a と t_c の間および t_d と t_f の間に、満足度が0.5となる加工終了時間 t_b と t_e を設定する。

Step 2: t_b と t_e で式(2.12)のメンバーシップ関数が0.5になるように、 u' と u の値を求める。すなわち、次式が成り立つような u' と u の値を求める。

$$\mu_j(t_b) = \{(t_b - t_a) / (t_c - t_a)\}^{u'} = 0.5 \quad (2.13)$$

$$\mu_j(t_e) = \{1 - (t_e - t_d) / (t_f - t_d)\}^u = 0.5 \quad (2.14)$$

Step 3: 図6のように、式(2.12)を用いて、2点 a と c および2点 d と f を補間し、メンバーシップ関数を設定する。

設定方法 D において、 $t_a = t_b = t_c = 0$ とすれば、設定方法 C と同様に図5のようなメンバーシップ関数を設定することができる。さらに、このとき、式(2.11)を用いて t_e を設定すれば、図2のような線形のメンバーシップ関数を設定することもで

きる。設定方法 B は設定方法 C と同じものと考えることができるので、設定方法 D は、設定方法 A~C を含んだ最も一般的な方法である。

最後に、特殊なタイプのメンバーシップ関数を用いた場合、2.2で定式化したファジィ・フローショップ・スケジューリング問題が、総処理時間最小化を目的とした最も一般的なフローショップ・スケジューリング問題と一致することを示す。いま、全てのジョブに対して全く同じメンバーシップ関数 $\mu(t)$ が用いられると仮定する。この場合では、2.2で定式化したファジィ・フローショップ・スケジューリング問題は、次のようになる。

$$f(\mathbf{x}) = \min\{\mu(t_c(m, x_k)) \mid k=1, 2, \dots, n\} \rightarrow \text{最大化} \quad (2.15)$$

ここで、メンバーシップ関数 $\mu(t)$ が図7に示すような単調減少型であり、各ジョブの加工終了時間に較べて t_e や t_f の値が十分に大きいと仮定する。この場合では、式(2.15)の $f(\mathbf{x})$ を最大化することは、次のように定義された $h(\mathbf{x})$ を最小化することと一致する。

$$h(\mathbf{x}) = \max\{t_c(m, x_k) \mid k=1, 2, \dots, n\} \rightarrow \text{最小化} \quad (2.16)$$

この $h(\mathbf{x})$ は、通常のフローショップ・スケジューリング問題における総処理時間である。このように、全てのジョブに対して同じファジィ納期を与え、そのメンバーシップ関数を図7のような単調減少型に設定した場合は、本研究で定式化したファジィ・フローショップ・スケジューリング問題は、総処理時間最小化を目的とした通常のフローショップスケジューリング問題と一致する。なお、図7のようなメンバーシップ関数は、設定方

図6 6点a~fにより決定されるファジィ納期のメンバーシップ関数

図7 単調減少型のメンバーシップ関数

法 D において, $t_a=t_b=t_c=t_d=0$ とすることにより設定することができる(もちろん, 設定方法 $A \sim C$ を用いても設定可能)。

3. SA 法を用いた解法

3.1 SA 法アルゴリズム

SA 法は, アルゴリズムが単純であるためにプログラム化が容易であることと, 問題領域に対する深い知識が不要であることから, 多くの組合せ最適化問題に応用されている[20,21]. SA 法は, 目的関数の値が悪くなる解への状態推移を確率的に行う局所探索法と考えることができる. いま, 目的関数 $f(\mathbf{x})$ の最大化を考えれば, $f(\mathbf{x}) \leq f(\mathbf{y})$ の場合すなわち目的関数の値が改良される場合では, 現在の解 $\mathbf{x}$ から候補解 $\mathbf{y}$ への状態推移は確率 1 で行われる. 一方, $f(\mathbf{x}) > f(\mathbf{y})$ の場合すなわち目的関数の値が悪くなる場合では, 次のような確率により, 現在の解 $\mathbf{x}$ から候補解 $\mathbf{y}$ への状態推移が行われる.

$$p(\mathbf{x} \rightarrow \mathbf{y}) = \exp(-\Delta f/c) \quad (3.1)$$

ここで,

$$\Delta f = f(\mathbf{x}) - f(\mathbf{y}) \quad (3.2)$$

は目的関数の値の悪化量である. また, c は物理的な焼きなまし (Annealing) における温度に対応した制御変数である. 式 (3.1) における Δf および c と $p(\mathbf{x} \rightarrow \mathbf{y})$ との関係を示すと, 図 8 のようになる.

図 8 状態推移確率 $p(\mathbf{x} \rightarrow \mathbf{y})$

図 8 からわかるように, c の値が大きい状態では, ほとんど全ての状態推移が受理され, c の値が非常に小さい状態では, 目的関数の値が減少する

(悪化する)状態推移はほとんど受理されない. SA 法では, 局所解に陥ることなく最適解に近い良好な解を得るため, 高温の状態から c の値を徐々に小さくするという冷却スケジュールが用いられる. SA 法の性能は, この冷却スケジュールの選び方に大きな影響を受ける. 例えば, 制御変数を急激に冷却した場合では, SA 法は局所探索法に似た振舞いを示し, 局所解に陥り易くなる. 逆に, 制御変数を高温の状態に保った場合では, 目的関数の値が悪化する解への状態推移が盛んに行われ, 良好な解に収束することは困難になる.

SA 法は, 無限回の状態推移を考えれば, 確率 1 で最適解に収束することが知られている[22,23]. しかし, 実際には無限回の状態推移を行うことはできないので, 制御変数 c の有限個の減少列を与えることにより, 近似解法として SA 法を組合せ最適化問題に適用することになる.

制御変数の減少列 $c_1, c_2, \dots, c_N$ が与えられた場合, 最大繰り返し数 N を停止基準とした SA 法アルゴリズムは, 次のように表現することができる.

[SA 法アルゴリズム: 繰り返し数 N]

- Step 1 初期解 $\mathbf{x}$ をランダムに設定し, $i := 0$ とする.
- Step 2 現在の解 $\mathbf{x}$ の近傍 $S(\mathbf{x})$ の中から候補解 $\mathbf{y}$ をランダムに選び出す. さらに, $i := i+1, c = c_i$ とする.
- Step 3 $f(\mathbf{y}) \geq f(\mathbf{x})$ ならば, $\mathbf{x} := \mathbf{y}$ とする. $f(\mathbf{y}) < f(\mathbf{x})$ ならば, 式 (3.1) で与えられる確率で, $\mathbf{x} := \mathbf{y}$ とする.
- Step 4 $i \geq N$ ならば, アルゴリズムの実行中に得られた最も良い解を最終的な解とし, 計算を終了する. 一方, $i < N$ ならば, Step 2 へ戻る.

SA 法アルゴリズムでは, 近傍構造 $S(\mathbf{x})$ と制御変数の減少列 (すなわち冷却スケジュール) の選び方が, 最終的に得られる解の性能に大きな影響を与える. 本研究では, 近傍構造 $S(\mathbf{x})$ として, 通常

のフローショップ・スケジューリング問題に対して Osman & Potts[16]や Taillard[17] により用いられているシフトと呼ばれる近傍構造を用いた。シフトと呼ばれる近傍構造は、任意の i, j に対して、 i 番目のジョブ x_i を j 番目のジョブ x_j の位置に挿入し、以下のようなシフト操作を行うことにより定義される。すなわち、 $i < j$ の場合では、 $i+1$ 番目から j 番目までのジョブが左へシフトし、現在の解 x から次のような解が生成される。

$$(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_j, x_i, x_{j+1}, \dots, x_n)$$

逆に、 $j < i$ の場合では、 j 番目から $i-1$ 番目までのジョブが右にシフトし、次のような解が生成される。

$$(x_1, \dots, x_{j-1}, x_i, x_j, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

なお、1回のシフト操作により現在の解 x から生成可能な解の総数(すなわち近傍構造の濃度)は、ジョブ数が n の場合では、 $(n-1)^2$ となる[17]。

冷却スケジュールとしては、Kirkpatrick ら[9]により提案された次のような減少関数を用いた。

$$c_i = c_1 \alpha^{t-1} \quad \text{for } (t-1)L < i \leq tL, \\ t = 1, 2, \dots, N/L \quad (3.3)$$

すなわち、 L 個の c_i が等しい値を取り、階段状に制御変数 c_i が減少する。なお、本研究の数値実験では、 L の値を、 $N/L=100$ となるように設定した。また、 α の値は、制御変数の初期値 c_1 および最終値 c_N を用いて、

$$\alpha = (c_N/c_1)^{1/99} \quad (3.4)$$

と設定した。これは、 $i=N$ および $t=N/L=100$ を式(3.3)に代入して導出したものである。

3.2 数値実験

数値実験を行うため、10 工程 20 ジョブのファジィ・フローショップ・スケジューリング問題を、乱数を用いて 10 問生成した。このとき、各ジョブの各工程における加工時間 $t_p(i, j)$ は、区間[1, 99]内の整数としてランダムに与えた。また、各ジョブの納期として、図 5 および図 6、図 7 に示すような 3 種類のファジィ納期を考えた。すなわち、次のような 3 種類のタイプのファジィ納期を、乱

数を用いて生成し、それぞれ、30%、40%、30%の確率で各ジョブの納期として与えた。

$$\begin{aligned} \text{タイプ I (図 5): } & t_a = t_b = t_c = 0 \\ & t_d = P \cdot \text{rnd}[0, 1] \\ & t_e = t_d + P \cdot \text{rnd}[0, 1] \\ & t_f = t_e + P \cdot \text{rnd}[0, 1] \end{aligned}$$

$$\begin{aligned} \text{タイプ II (図 6): } & t_a = 0 \\ & t_b = 0.5 \cdot P \cdot \text{rnd}[0, 1] \\ & t_c = t_b + 0.5 \cdot P \cdot \text{rnd}[0, 1] \\ & t_d = t_c + P \cdot \text{rnd}[0, 1] \\ & t_e = t_d + P \cdot \text{rnd}[0, 1] \\ & t_f = t_e + P \cdot \text{rnd}[0, 1] \end{aligned}$$

$$\begin{aligned} \text{タイプ III (図 7): } & t_a = t_b = t_c = t_d = 0 \\ & t_e = 2 \cdot P \cdot \text{rnd}[0, 1] \\ & t_f = t_e + 2 \cdot P \cdot \text{rnd}[0, 1] \end{aligned}$$

ここで、 $\text{rnd}[0, 1]$ は区間[0, 1] 内の乱数であり、 P は次のように定義された定数である。

$$P = \sum_{i=1}^m \sum_{j=1}^n t_p(i, j) / m \quad (3.5)$$

このようにして生成された 10 個の例題に対し、次のような条件で SA 法を適用した。

アルゴリズムの繰り返し数 N : 10000
 制御変数の初期値 c_1 : 0.1, 1
 制御変数の最終値 c_N : 0.0001, 0.0002, 0.0005, 0.001, ..., 0.5

10 個の例題に対する数値実験の結果得られた目的関数値の平均を、図 9 と図 10 に示す。図 9 は、制御変数 c の初期値 c_1 を $c_1 = 1$ と設定した場合の結果であり、図 10 は、 $c_1 = 0.1$ と設定した場合の結果である。

比較のため、同じ問題に対して、次のような局所探索法を適用した。

[局所探索法アルゴリズム]

Step 1 $i := 0$ とする。
 Step 2 初期値 x をランダムに設定する。
 Step 3 x の近傍 $S(x)$ のなかで最も良好な解を y^* とする。すなわち、次式を満たす y^* を求める。

$$f(y^*) = \max\{f(y) \mid y \in S(x)\} \quad (3.6)$$

- Step 4 $i := i + (n-1)^2$ とする. $i \geq N$ ならば, アルゴリズムの実行中に得られた最も良い解を最終的な解とし, 計算を終了する.
- Step 5 $f(y^*) > f(x)$ ならば, $x := y^*$ として Step 3 へ戻る. $f(y^*) \leq f(x)$ ならば Step 2 へ戻る.

このアルゴリズムでは, 探索が局所解に陥るごとに新たな初期解から探索を行っている. また, 前節の SA 法アルゴリズムと計算量を同一にするために, 目的関数 $f(x)$ の計算回数 N を局所探索法アルゴリズムの停止基準とした. このアルゴリズムを 10 個の例題に適用した結果得られた目的関数値の平均は, 0.060 であった. このような結果と図 9 および図 10 の結果を比較することにより, 次のようなことがわかる.

図 9 $c_1 = 1$ とした場合の数値実験の結果

図 10 $c_1 = 0.1$ とした場合の数値実験の結果

- (1) 制御変数 c の初期値 c_1 と最終値 c_N を適切に設定すれば, SA 法により局所探索法よりも良い結果が得られる.

- (2) 制御変数 c の初期値 c_1 と最終値 c_N の設定が不適切であれば, SA 法により得られる結果は非常に悪くなる.

4. 目的関数の修正による解法の改良

4.1 目的関数の修正

2 章で定式化したファジィ・フローショップ・スケジューリング問題の目的関数 $f(x)$ は, 次のように定義されていた.

$$f(x) = \min\{\mu_{x_k}(t_c(m, x_k)) \mid k=1, 2, \dots, n\} \rightarrow \text{最大化} \quad (4.1)$$

上式からわかるように, ファジィ納期に対する満足度が 0 となるジョブが一つでもあれば, 目的関数 $f(x)$ の値も 0 となる. したがって, $n!$ 通りの可能な加工順序 x のなかで, ほとんどの加工順序に対して $f(x)$ が 0 となることも考えられる. 実際, 3 章で設定した 10 個の例題では, 平均して 99.93% の加工順序に対して $f(x)$ が 0 となっている (なお, この数値は各例題に対してランダムに 10000 通りの加工順序を生成することにより得られたものである).

表 1 SA で得られた解における目的関数の値 ($c_1=1$, $c_N=0.001$ の場合)

問題番号	目的関数	問題番号	目的関数
問題 1	0.000	問題 6	0.000
問題 2	0.000	問題 7	0.000
問題 3	0.310	問題 8	0.000
問題 4	0.000	問題 9	0.000
問題 5	0.441	問題 10	0.000

SA 法では, 現在の解 x と候補解 y における目的関数の値の差に基づき探索 (状態推移) が行われるので, ほとんど全ての解で目的関数が 0 となれば, 効率的に探索を行うことができない. 実際, 10 個の例題に対して SA 法で得られた解の一例を示すと表 1 のようになり, 8 個の例題で探索が有効に行われていないことがわかる.

そこで本章では, SA 法において効率的な探索が可能になるように, 目的関数を修正する方法を示す. まず, 各ジョブのファジィ納期を表すメン

バーシップ関数 $\mu_j(t)$ を用いて, 新たに図 11 に示すような関数 $\nu_j(t)$ を設定する. 図 11 の $\nu_j(t)$ は, メンバーシップ関数 $\mu_j(t)$ を用いて次のように設定されたものである.

$$\nu_j(t) = \begin{cases} (t-t_a)q; & 0 < t < t_a \\ \mu_j(t); & t_a \leq t \leq t_f \\ (t_f-t)q; & t_f < t \end{cases} \quad (4.2)$$

ここで q は正の定数であり, $t < t_a$ および $t_f < t$ の領域での $\nu_j(t)$ の傾きを表わしている. このように設定された関数 $\nu_j(t)$ は, ファジィ納期を表すメンバーシップ関数 $\mu_j(t)$ を, 負の値を取り得る関数に拡張したものとみなすことができる.

図11 関数 $\nu_j(t)$

ジョブ j ($j=1, 2, \dots, n$) に対するファジィ納期を表すメンバーシップ関数 $\mu_j(t)$ から生成された関数 $\nu_j(t)$ を用いて, SA 法を用いて最大化されるべき目的関数を次のように設定する.

$$g(\mathbf{x}) = \min\{\nu_{x_k}(t_c(m, x_k)) \mid k=1, 2, \dots, n\} \quad (4.3)$$

式(4.1)で与えられる修正前の目的関数 $f(\mathbf{x})$ と式(4.3)で設定した修正後の目的関数 $g(\mathbf{x})$ との関係は次のようになる.

$$f(\mathbf{x}) = \max\{0, g(\mathbf{x})\} \quad (4.4)$$

すなわち, 式(4.3)の目的関数は, $g(\mathbf{x}) > 0$ の領域では 2 章で定式化した目的関数 $f(\mathbf{x})$ と全く同じ値を取る.

4.2 数値実験

式(4.3)の目的関数を用いて, 3.2 節と全く同じ条件で数値実験を行った. なお, 式(4.2)における傾き q の値は, $q=1$ とした.

このような数値実験の結果を図 12 と図 13 に示

す. 図 12 は制御変数の初期値 c_1 を $c_1=1$ とした場合に得られた解における $f(\mathbf{x})$ の値の平均であり, 図 13 は $c_1=0.1$ とした場合での結果である. 図 12, 図 13 を, それぞれ, 図 9, 図 10 と比較することにより, 目的関数の修正を行った結果, SA 法の性能が向上していることがわかる. なお, 式(4.3)を用いて局所探索を行った結果得られた解における $f(\mathbf{x})$ の値の平均は, 0.131 であった.

図12 目的関数の修正を行った結果
($c_1=1$, 傾き $q=1$)

図13 目的関数の修正を行った結果
($c_1=0.1$, 傾き $q=1$)

さらに, 式(4.2)における傾き q の値と SA 法の性能との関係を調べた. このときの数値実験の結果を, 図 14, 図 15 に示す. なお, 傾き q が 0 の場合は, 式(4.3)で定義した修正後の目的関数 $g(\mathbf{x})$ は, 2 章で設定した目的関数 $f(\mathbf{x})$ と同じものとなる. このような数値実験の結果から, 次のようなことがわかる.

- (1) 目的関数の修正を行う ($q > 0$ とする) ことにより, SA 法の性能が向上する (図 14, 図 15).
- (2) SA 法の性能は, 制御変数 c の設定方法の影響を強く受ける. ただし, 制御変数 c の設定を適切に行えば, SA 法により局所探索法よりも良好な結果が得られる (図 12, 図 13).

修正後の目的関数 $g(x)$ を用いた SA 法により得られた解の一例を具体的に示すと、表 2 のようになる。この結果を、修正前の目的関数 $f(x)$ を用いた表 1 の結果と比較すると、目的関数の値が 0 であった 8 個の問題のなかで、5 個の問題に対して正の値が得られていることがわかる。すなわち、これらの問題では、SA 法による探索が有効に行われていることがわかる。

図14 傾き q と SA 法の性能との関係
($c_1=1$, $c_N=0.001$)

図15 傾き q と SA 法の性能との関係
($c_1=0.1$, $c_N=0.001$)

表 2 目的関数 $g(x)$ を用いた SA 法により得られた解における $f(x)$ の値
($q=1$, $c_1=1$, $c_N=0.001$ の場合)

問題番号	$f(x)$	問題番号	$f(x)$
問題 1	0.146	問題 6	0.000
問題 2	0.000	問題 7	0.376
問題 3	0.289	問題 8	0.170
問題 4	0.000	問題 9	0.002
問題 5	0.441	問題 10	0.002

修正後の目的関数 $g(x)$ を用いた SA 法による探索の軌跡を問題 7 に対して示すと、図 16 のようになる。図 16 に示すように、 $g(x)$ が負の領域、すなわち修正前の目的関数 $f(x)$ が 0 の領域でも探索が行われ、最終的に正の最小満足度が得られて

いる。一方、修正前の目的関数 $f(x)$ を用いた場合での探索の軌跡を示すと、図 17 のようになり、 $f(x)=0$ の初期状態から全く解が改善されていないことがわかる。なお、図 17 では、式(3.1)における Δf が常に 0 となるので、解の推移は近傍のなかで全くランダムに行われている。

図16 目的関数の修正を行った後の探索の軌跡
($q=0.001$, $c_1=1$, $c_N=0.001$)

図17 目的関数の修正を行う前の探索の軌跡
($q=0$, $c_1=1$, $c_N=0.001$)

5. 改良型 SA 法を用いた解法

5.1 改良型 SA 法アルゴリズム

3 章や 4 章の数値実験で示したように、SA 法の性能は制御変数 c の設定方法に大きく依存する。特に、制御変数 c の値を高く設定した場合には、図 18 に示すように、目的関数の値が悪化する方向への状態推移も高い確率で受理され(図 8 参照)、良好な解へ探索が収束しない。なお、図 18 は、図 16 と同じ問題に対して、制御変数の最終値

c_N を $c_N=0.5$ と設定した場合での探索の軌跡を示したものである(図16では $c_N=0.001$).

このように、制御変数 c の設定方法が不適切な場合には、通常のSA法の性能はたいへん悪化する. これに対してIshibuchiら[11,12]では、制御変数 c の設定方法に関してロバストな性能を持つ、次のような改良型SA法アルゴリズムが提案されている.

図18 制御変数の設定が不適切な場合での探索の軌跡 ($q=0.001$, $c_1=1$, $c_N=0.5$)

[改良型 SA 法アルゴリズム]

- Step 1 初期解 $\mathbf{x}$ をランダムに設定し, $i := 0$ とする.
- Step 2 近傍 $S(\mathbf{x})$ の中から K 個の解を, 重複を許してランダムに選び出し, 最も良好な解を $\mathbf{y}$ とする. さらに, $i := i + K$, $c = c_i$ とする.
- Step 3 $g(\mathbf{y}) \geq g(\mathbf{x})$ ならば, $\mathbf{x} := \mathbf{y}$ とする. $g(\mathbf{y}) < g(\mathbf{x})$ ならば, 式(3.1)で与えられる確率で, $\mathbf{x} := \mathbf{y}$ とする.
- Step 4 $i \geq N$ ならば, アルゴリズムの実行中に得られた最も良い解を最終的な解とし, 計算を終了する. 一方, $i < N$ ならば, Step 2 へ戻る.

このアルゴリズムの繰り返し数は N/K 回であり, 目的関数 $g(\mathbf{x})$ の計算回数は, 3章で示したSA法アルゴリズムと同様に N 回である. 改良型SA法では, 現在の解 $\mathbf{x}$ の近傍 $S(\mathbf{x})$ から K 個の解をサンプルし, 最も良好な解を状態推移の候補解 $\mathbf{y}$ としている. したがって, 良好な解ほど高い生成確率を持つことになる(通常のSA法では生

成確率は解の性能に依らず近傍内の全ての解で同一). このような修正により, 制御変数 c の値が大きい高温状態でも良好な解への収束性が高まり, 制御変数の設定に関してロバストな性能が得られる. 例えば, $K=5, 10, 20$ と設定し, 図18と同じ問題に対して, 同じ条件で数値実験を行った結果を図19~図21に示す. これらの図から, K の値が大きいほど収束性が高まることがわかる. た

図19 改良型 SA 法を用いた場合での探索の軌跡 ($q=0.001$, $K=5$, $c_1=1$, $c_N=0.5$)

図20 改良型 SA 法を用いた場合での探索の軌跡 ($q=0.001$, $K=10$, $c_1=1$, $c_N=0.5$)

図21 改良型 SA 法を用いた場合での探索の軌跡 ($q=0.001$, $K=20$, $c_1=1$, $c_N=0.5$)

だし、 K の値が大きすぎる場合では、探索の収束性が高くなりすぎ、局所解に陥り易くなる場合もある。なお、種々の設定での数値実験により、 K の値をジョブ数 n と同じ程度に設定すれば改良型 SA 法の性能が高くなるという結果が得られている。例えば、 $K=n$ に設定した場合では、近傍 $S(x)$ に含まれる $(n-1)^2$ 個の解のなかで、約 $1/n$ の解が 1 回の状態推移のためにサンプルされることになる。

なお、Step 2 における K 個の解の生成と目的関数値の計算は独立に実行することができるので、改良型 SA 法は並列計算に適したアルゴリズムである [11]。

5.2 数値実験

4 章の図 12、図 13 と同じ条件で、10 個の例題に対して、 $K=20$ ($K=n$) と設定した改良型 SA 法を適用した。この結果を図 22 と図 23 に示す。図 22、図 23 を、図 12、図 13 と比較することにより、改良型 SA 法が制御変数 c の設定に関してロバストな性能を持つことがわかる。また、図 22 と

図22 改良型 SA 法を用いた結果 ($c_1=1$)

図23 改良型 SA 法を用いた結果 ($c_1=0.1$)

図 23 では、制御変数 c の設定方法に依らず、同じ計算量の局所探索法(得られた解の平均値: 0.131)よりも良好な結果が得られている。

なお、通常の SA 法により得られた解と改良型 SA 法により得られた解を比較すると、表 3 と表 4 のようになる。表 3 では、制御変数の設定が不適切であるため探索が効果的に行われておらず(図 18 参照)、得られた解の性能は良くない。一方、改良型 SA 法(表 4)では、制御変数の設定が不適切である場合でも、適切に制御変数の設定を行った通常の SA 法(表 2)と同程度の解が得られている。

表 3 制御変数の設定が不適切な SA 法により得られた解における $f(x)$ の値
($q=1, c_1=1, c_N=0.5$ の場合)

問題番号	$f(x)$	問題番号	$f(x)$
問題 1	0.000	問題 6	0.000
問題 2	0.000	問題 7	0.000
問題 3	0.069	問題 8	0.000
問題 4	0.000	問題 9	0.000
問題 5	0.156	問題 10	0.000

表 4 制御変数の設定が不適切な改良型 SA 法により得られた解における $f(x)$ の値
($K=20, q=1, c_1=1, c_N=0.5$ の場合)

問題番号	$f(x)$	問題番号	$f(x)$
問題 1	0.146	問題 6	0.000
問題 2	0.000	問題 7	0.361
問題 3	0.302	問題 8	0.170
問題 4	0.000	問題 9	0.002
問題 5	0.440	問題 10	0.002

比較のため、同じ問題に対して、ファジィ納期の満足度が 0 となる t_f の値(図 2 ~ 図 7 参照)の小さい順にジョブを加工するというヒューリスティック手法を適用した。このような手法により、表 5 のような解が得られた。また、得られた解における $f(x)$ の値の平均は、0.017 であった。このような結果から、納期の順に加工するという単純なヒューリスティック手法では、SA 法(表 2)や改良型 SA 法(表 4)よりも良い解が得られていないことがわかる。

さらに比較のため、ランダムサンプリング法(無作為に複数個のスケジュールを生成し、そのなか

の最良解を求めるという方法)を10個の例題に適用し、サンプル数と得られた解の平均値との関係を調べた。サンプル数を10000, 20000, ..., 100000とした場合での、サンプル数と解の性能との関係を図24に示す。図24を図22, 図23と比較することにより、目的関数の計算回数が10000回の改良型SA法により、計算回数が100000回のランダムサンプリング法(得られた解における $f(x)$ の平均値: 0.049)よりも良好な解が得られていることがわかる。

表5 ヒューリスティック手法により得られた解における $f(x)$ の値

問題番号	$f(x)$	問題番号	$f(x)$
問題1	0.017	問題6	0.000
問題2	0.000	問題7	0.004
問題3	0.074	問題8	0.010
問題4	0.000	問題9	0.000
問題5	0.065	問題10	0.000

図24 ランダムサンプリング法を用いた結果

本研究における数値実験は全て、ソニーワークステーション NES 3460 (17 MIPS) を用いて行った。各例題に対する各手法の1回の実行(繰り返し数10000)に必要な計算時間を示すと表6のようになり、各手法とも約20秒程度の計算時間で実行できることがわかる。なお、本研究の数値実験では、繰り返し数を10000に設定したが、SA法(改良型SA法や局所探索法でも以下同様)では、繰り返し数を自由に設定することができる。また、SA法の計算時間は繰り返し数に比例する。このようなことから、SA法では、利用可能な計算時間と計算機の性能に応じて繰り返し数を設定することに

より、多様な計算時間の制約に対応することができる。例えば、石渕ら[24]では、繰り返し数を200~50000に設定したSA法を、総処理時間最小化を目的としたフローショップ・スケジューリング問題に適用した結果が示されている。

表6 計算時間の比較

手 法	計算時間
$f(x)$ を用いたSA法	21.73 (秒)
$g(x)$ を用いたSA法	22.26 (秒)
$g(x)$ を用いた改良型SA法	19.17 (秒)
$f(x)$ を用いた局所探索法	17.09 (秒)
$g(x)$ を用いた局所探索法	17.44 (秒)

5.3 最適解との比較

これまでの数値実験では、20ジョブ10工程問題を用いて、近似解法としてのSA法の性能について考察した。ここでは、最適解を得ることが可能な小規模な問題を用いて、最適解法としてのSA法の性能について考察する。

3.2節の数値実験と同様にして、10ジョブ5工程問題を乱数を用いて50問作成した。次に、完全列挙法を用いて最適解を求めた(実行可能解の総数: $10! = 3,628,800$ 個)。なお、最適値が0となった問題に対しては乱数を用いて再度問題を作成した結果、最適値が正となる50個の例題とその最適値が得られた。

このような問題に対して、 $g(x)$ を用いた改良型SA法を、次のような条件で適用した。

目的関数の計算回数 N の最大値: 3000

制御変数 c (一定): 0.001, 0.01, ..., 100

傾き q : 1

サンプル数 K : 1, 5, 10

50個の問題のなかで、最適解が得られた問題数を表7に示す。表7より、3,628,800個の実行可能解の0.1%以下である3000個の実行可能解のみに対する目的関数の値を調べることににより、適切な条件のもとでは、50個全ての問題に対して最適解が得られていることがわかる。特に、改良型SA法におけるサンプル数 K の値をジョブ数と同じ10に設定した場合では、制御変数 c の値に依ら

ず、非常に高い確率で最適解が得られることがわかる。

表7 最適解が得られた問題数

目的関数の 計算回数 N		1000	2000	3000
$K=1$	$c=0.001$	45	46	46
	$c=0.01$	48	50	50
	$c=0.1$	32	34	35
	$c=1$	26	31	32
	$c=10$	26	30	32
	$c=100$	16	21	23
$K=5$	$c=0.001$	46	47	47
	$c=0.01$	47	49	49
	$c=0.001$	45	46	46
	$c=0.1$	44	46	49
	$c=1$	40	44	46
	$c=10$	37	40	45
$K=10$	$c=100$	38	44	48
	$c=0.001$	47	48	48
	$c=0.01$	49	50	50
	$c=0.1$	46	50	50
	$c=1$	47	48	50
	$c=10$	46	47	49
	$c=100$	45	48	50

6. 結言

本研究では、フローショップ・スケジューリング問題にファジィ納期を導入することにより、ファジィ・フローショップ・スケジューリング問題を定式化し、SA法を用いた解法を示した。本研究のファジィ・フローショップ・スケジューリング問題は、ファジィ納期に対する満足度の最小値を最大化する問題として定式化されている。さらに、SA法の性能を向上させるために、目的関数の修正を行う方法を示した。修正された目的関数では、ファジィ納期を表すメンバーシップ関数が、負の値を取り得るように拡張されている。また、このような定式化および解法の提案を行うと同時に、ランダムに生成された10個の10工程20ジョブ問題に対する数値実験により、以下のことを示した。

- (1) SA法により、局所探索法よりも良好な結果が得られる。
- (2) 目的関数の修正を行うことにより、SA法の性

能が向上する。

- (3) 改良型SA法により、制御変数 c の設定方法に関してロバストな性能が得られる。

参 考 文 献

- [1] R.E.Bellman and L.A.Zadeh: Decision Making in a Fuzzy Environment, Management Sciences, Vol.17, pp.B141-B164 (1970)
- [2] 田中, 奥田, 浅居: ファジィ数理計画法, 計測自動制御学会論文集, Vol.9, pp.607-613(1973)
- [3] H.J.Zimmermann: Description and Optimization of Fuzzy Systems, International Journal of General Systems, Vol.2, pp.209-215 (1976)
- [4] 坂和正敏: ファジィ対話型多目的計画法, 日本ファジィ学会誌, Vol.4, No.1, pp.11-20(1992)
- [5] 乾口雅弘: 確率計画問題とファジィ数理計画問題, 日本ファジィ学会誌, Vol.4, No.1, pp.21-30 (1992)
- [6] 乾口雅弘: ファジィ数理計画問題の可能性理論に基づく体系的定式化法, 大阪府立大学博士論文 (1991)
- [7] 石井博昭: ファジィ組合せ最適化, 日本ファジィ学会誌, Vol.4, No.1, pp.31-40 (1992)
- [8] H.Ishii, M.Tada and T.Masuda: Two Scheduling Problems with Fuzzy Duedates, Fuzzy Sets and Systems, Vol.46, pp.339-347 (1992)
- [9] S.Kirkpatrick, C.D.Gelatt Jr. and M.P.Vecchi: Optimization by Simulated Annealing, Science, Vol.220 pp.671-680(1983)
- [10] V.Cerny: Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, Journal of Optimization Theory and Applications, Vol.45, No.1, pp.41-51 (1985)
- [11] H.Ishibuchi, S.Misaki and H.Tanaka: Parallel Algorithm of Simulated Annealing with Modified Generation Mechanism, Proceedings of IJCNN'91(November 18-21, Singapore), pp.2434-2439(1991)
- [12] H.Ishibuchi, S.Misaki and H.Tanaka: Simulated Annealing with Modified Generation Mechanism for Flow Shop Scheduling Problems, Proceedings of International Symposium on Robotics, Mechatronics and Manufacturing

- Systems(September 16-20, Kobe, Japan), pp. 21-26(1992)
- [13]坂和, 岩佐, 宇高, 五百旗頭: Simulated Annealing 法とファジィモデリングによる電気炉の最適運用計画, 第8回ファジィシステムシンポジウム講演論文集, pp.149-152(1992)
- [14]坂和, 澤田, 銅木: ニューラルネットワークによるファジィ多目的組合せ最適化, 第8回ファジィシステムシンポジウム講演論文集, pp.145-148(1992)
- [15]M.Widmer and A.Hertz: A New Heuristic Method for the Flow Shop Sequencing Problem, European Journal of Operational Research, Vol.41, pp.186-193 (1989)
- [16]I.H.Osman and C.N.Potts: Simulated Annealing for Permutation Flow-Shop Scheduling, OMEGA, Vol.17, No.6, pp.551-557 (1989)
- [17]E.Taillard: Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem, European Journal of Operational Research, Vol. 47, pp.65-74 (1990)
- [18]J.C.Ho and Y.L.Chang: A New Heuristic for the n-Job, M-Machine Flow-Shop Problem, European Journal of Operational Research, Vol.52, pp.194-202 (1991)
- [19]R.A.Dudek, S.S.Panwalkar and M.L.Smith: The Lessons on Flowshop Scheduling Research, Operations Research, Vol.40, No.1, pp.7-13 (1992)
- [20]P.J.M Van Laarhoven and E.H.L.Aarts: Simulated Annealing: Theory and Applications, Kluwer Academic Publishers, Dordrecht (1987)
- [21]E.H.L.Aarts and J.Korst: Simulated Annealing and Boltzmann Machines, John Wiley & Sons, Chichester (1989)
- [22]S.Geman and D.Geman: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.PAMI-6, No.6, pp.721-741 (1984)
- [23]B.Hajek: Cooling Schedules for Optimal Annealing, Mathematics of Operations Research, Vol.13, No.2, pp.311-329(1988)
- [24]石渕, 田村, 田中: シミュレーティッド・アニーリング法によるフロー・ショップ・スケジューリング, システム制御情報学会論文誌, Vol.4, No. 3, 111-117 (1991)

(1992年8月21日 受付)

(1992年12月16日 再受付)

(1993年3月8日 再々受付)

[問い合わせ先]

〒593 大阪府堺市学園町1

大阪府立大学工学部経営工学科

石渕 久生

☎: 0722-52-1161(内)2377

☎: 0722-59-3340

著者紹介

石渕 久生 (いしおち ひさお)

大阪府立大学 工学部 経営工学科

'85年京都大学工学部機械系精密工学科卒業。'87年同大学大学院修士課程精密工学専攻修了。同年大阪府立大学工学部経営工学科助手, '93年同講師となり現在に至る。博士(工学)。ファジィ理論とニューラルネットの応用に関する研究に従事。日本経営工学会, 電子情報通信学会, JNNS, INNS, IFSAなどの会員。

御前 伸太 (みさき しんた)

大阪府立大学 工学部 経営工学科

'91年大阪府立大学工学部経営工学科卒業。'93年同大学大学院工学研究科博士前期課程経営工学専攻修了。同年日立製作所入社。在学中, シミュレーティッド・アニーリング法に関する研究に従事。システム制御情報学会, 日本ファジィ学会の会員。

田中 英夫 (たなか ひでお)

大阪府立大学 工学部 経営工学科

'62年神戸大学工学部計測工学科卒業。同年ダイキン工業(株)入社。'69年大阪市立大学大学院工学研究科博士課程(電気工学専攻)修了。工学博士。同年大阪府立大学工学部経営工学科助手。'87年7月同大学教授。現在に至る。この間、'72年カリフォルニア大学電気・計算機学科、'76年アーヘン工科大学OR学科、'81年カンサス州立大学化学工学科研究員。ファジィ理論とその応用に関する研究に従事。IEレビュー貢献賞授賞。日本ファジィ学会、日本経営工学会、日本人間工学会などの会員。