J. Rakuno Gakuen Univ., **42** (2) : 105～113 (2018)

離散型 PSO を用いたジョブショップ・スケジューリング問題の解法

高　取　則　彦*

A discrete particle swarm optimization algorithm for the job-shop scheduling problem

Norihiko TAKATORI*
(Accepted 29 November 2017)

1．はじめに

スケジューリング問題は，組合せ最適化問題の一種である。ジョブショップ・スケジューリング問題は，スケジューリング問題の一種で，非常に解きにくいクラスの問題として知られている。一般に，この種の問題は，最適解を求めるのが極めて難しい。そのため，近似解法やメタヒューリスティクスと呼ばれる発見的方法を用いて，準最適解を求めることが多い。

これまで，メタヒューリスティクスとして，さまざまなものが提案されている。PSO（Particle Swarm Optimization）はその１つである。これは，人間や動物に見られる社会的行動から着想を得たもので，連続型最適化問題の発見的解法として提案されている。

この論文では，ジョブショップ・スケジューリング問題への PSO を用いた解法を提案する。従来の PSO は連続型問題を対象にしているので，これをもとにして組合せ最適化問題を扱えるような離散型 PSO を考え，ジョブショップ・スケジューリング問題に適用可能なアルゴリズムを構成する。ここでは，そのアルゴリズムと計算機実験の結果を報告する。

2．スケジューリング問題

2.1　スケジューリング問題

スケジューリングとは，複数の仕事について，処理の順序を決め，必要な資源を割り当てる時間を決めることである。これにより得られた結果がスケジュールである。スケジュールは，１つの問題に対して複数存在するのがふつうである。スケジューリング問題は，複数存在するスケジュールの中から，あらかじめ設定した評価基準のもとで最良のスケジュールを求める問題である[1]。

製造業では，部品や製品の製造が仕事であり，ジョブと呼ばれることが多い。工場内の生産設備である機械が資源である[2]。スケジューリング問題は，製造業のほかサービス業など実社会におけるさまざまな分野の業務の中にも見られる[3]。また，コンピューターのオペレーティング・システムにおけるタスク管理など，製造業などと全く異なるところでも同様の問題が見られる[4]。いずれの場合も，それぞれの設定に関して，制約条件や評価基準にさまざまなものがあり，問題の種類は多岐にわたっている。以下，スケジューリング問題を述べるとき，製造業で用いられる ‘ジョブ’ と ‘機械’ という用語を用いることにする。

2.2　ジョブショップ・スケジューリング問題

ジョブは，より小さな複数の作業から構成されることがある。この作業を通常のジョブと区別してタスクと呼ぶ。それぞれのジョブについて，それに含まれるタスクと，その処理順序と使用する機械が決められている。ジョブショップ・スケジューリング問題は，使用する機械の順序がジョブによって異なる問題である。

ジョブショップ・スケジューリング問題においても，さまざまな評価基準が用いられる。その代表的なものに，総処理時間 $C_{\max}$ すなわちすべてのジョブを完了する時間があり，これが最小となるスケジュールを求める問題がよく知られている。その多くは，ジョブはすべての機械を一度ずつ使用するとしている。つまり，ジョブには機械の台数と同じ数

* 酪農学園大学農食環境学群環境共生学類情報工学研究室
Computer Science, Department of Environmental and Symbiotic Science, College of Agriculture, Food and Environment Sciences, Rakuno Gakuen University, Ebetsu, Hokkaido, 069-8501, Japan

のタスクがあり，タスクはそれぞれ異なる１台の機械で処理される。さらに，以下を仮定している。

・ジョブを構成するタスクの間には，ジョブの中での先行関係が決められている。
・タスクの処理時間は，あらかじめ決まっていて変化しない。
・機械は，一度に１つのタスクだけを処理でき，中断されない。

ここでは，ジョブショップ・スケジューリング問題のうち $C_{\max}$ 最小化問題を対象とし，上と同様の仮定をおくことにする。

2.3 スケジューリング問題の解法

スケジューリング問題は，組合せ最適化問題の一種である。最適化問題は，複数の選択肢から最善のものを選ぶ問題であり，次のような解 $\boldsymbol{x}$ を求めることと表される。

目的関数 $f(\boldsymbol{x})$→最小化（最大化）

制約条件 $\boldsymbol{x} \in F$ （F：実行可能解の集合）

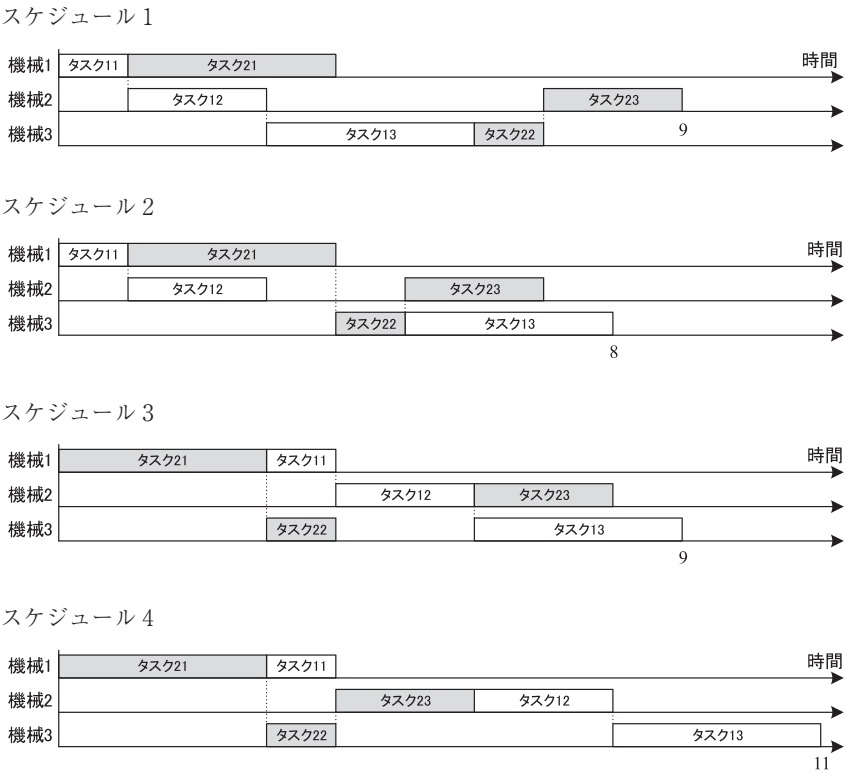
このうち，解が組合せ的・離散的なものを組合せ最適化問題という。この問題では，解は要素の順列や組合せで構成されることが多い[5]。

図１は，２ジョブ３機械のジョブショップ・スケジューリング問題の例である。(1)は，ジョブを構成するタスクの情報である。タスクが使用する機械と処理に要する時間は，あらかじめ決められている。(2)は，可能な解つまりスケジュールの一部である。各機械でのタスクの処理順序により，総処理時間 $C_{\max}$ が変わることがわかる。

スケジューリング問題のような組合せ最適化問題では，解の個数は有限である。それゆえ，理論的にはすべての解を列挙してその中から最良のものを選べば，問題を解くことができる。しかし，この例のような小さな問題でも可能な解は複数存在し，問題に含まれる要素の数が大きくなると，それは爆発的に増加する。そのため，このような列挙法的なアプ

	タスク（使用機械，処理時間）		
ジョブ１	タスク 11（1，1）	タスク 12（2，2）	タスク 13（3，3）
ジョブ２	タスク 21（1，3）	タスク 22（3，1）	タスク 23（2，2）

(1) ジョブとタスクの情報



(2) スケジュールの例

図１ ジョブショップ・スケジューリング問題の例（２ジョブ３機械）

ローチでは，現在最速のコンピューターを用いても，現実的な時間で解を得ることはできない。

このような理由から，組合せ最適化問題に対して，実用的には十分よい解を比較的短時間で求める解法が必要とされている。近年，メタヒューリスティクスと呼ばれる種類の解法が多数提案されている[6]。これに分類される解法は，近似解法の一種であるが，解に対して最適性の保証をもたないものが多い。しかし，計算時間を自由に設定でき，その中でよりよい解を求めることができる。このような実用に適した性質をもつため，実務でもよく用いられている。

3．PSO（Particle Swarm Optimization）

3.1　PSO の着想

鳥や魚など動物には，群れをつくって行動するものがある。このような動物は，群れをつくることによって，エサや仲間を探したり，捕食者を避けたりするなど，よりよい環境を見つけるための行動を，より効率的に行っていると考えられる。このときとる実際の行動は，どの方向へどのくらい移動するかといった物理的な動きとなる。人間にも同様のことが見られるが，人間の場合は物理的な動きよりもむしろ社会における振舞いに現れることが多い。たとえば，人間は，自分の属するグループのメンバーのようすを見て，考えや態度を変えることがある。このように，動物や人間が行動や態度などを決めるとき，自分の現状の認識と過去の記憶，さらに周囲からの情報が用いられていると考えられる。

PSO は，このような動物や人間の集団的行動に着想を得た，最適化問題に対するメタヒューリスティクスの 1 つである[7]。上述のような考察から，Kennedy らは，集団における個体は，

・評価
・比較
・模倣

という 3 つの原理にもとづいて行動すると考え，これらを最適化問題の解法に取り入れた。個体が問題の解に対応すると考え，複数の個体からなる集団を用いて，解の探索を行う。各個体は，自分の状態を評価し，他の個体と比較する。そして，自分よりよい個体を模倣することにより，自身の改善を試みる。さらに，個体は自分のそれまでの最良の状態を記憶しており，模倣のときこれも考慮に入れる。

個体間には，あらかじめ「つながり」が定められている。このつながり方は，解を探索している間保持される。情報交換ができるのは，つながりのある個体間に限られる。したがって，模倣する相手は，つながりのある個体から選ばれる。つながり方には，リング状すなわち個体 i が個体 $i-1$，$i+1$ とつながりを持つものや，つながりの範囲が個体 $i-Ne$，$i-(Ne-1)$，…，$i+(Ne-1)$，$i+Ne$ $(Ne>1)$ まで広がったものなど，さまざまなものがある。

3.2　PSO による最適化問題の解法

PSO による n 次元関数 $f(\boldsymbol{x})$ の最小化問題の解法について述べる[7]。個体 i は組 $(\boldsymbol{x}_i, \boldsymbol{v}_i)$ によって表される。$\boldsymbol{x}_i$ は位置と呼ばれ，問題の解そのものである，$\boldsymbol{v}_i$ は速度と呼ばれており，1 回の更新での $\boldsymbol{x}_i$ の移動ベクトルである。$\boldsymbol{v}_i$ の各成分 v_{ik} $(k=1, 2, \cdots, n)$ は，次式により更新される。

$$v_{ik} \leftarrow v_{ik} + r_1(x_{ik}^{\text{best}} - x_{ik}) + r_2(x_{ik}^{\text{local}} - x_{ik}) \quad \cdots (1)$$

ここで，$\boldsymbol{x}_i^{\text{best}}$ は，個体 i のこれまでで最良だった位置つまり $f(\boldsymbol{x}_i)$ が最小だった位置である。$\boldsymbol{x}_i^{\text{local}}$ は，個体 i とつながりのあるものの中で最良の個体の位置である。r_1，r_2 は一様乱数で，$r_1, r_2 \in [0, R]$ $(R>0)$ である。$\boldsymbol{v}_i$ は，$\boldsymbol{x}_i$ の修正量を表すベクトルであり，$\boldsymbol{x}_i^{\text{best}}$ と $\boldsymbol{x}_i^{\text{local}}$ の両方を用いて計算される。これは，過去の記憶と周囲からの情報を利用して，改善を試みることを意味している。v_{ik} は，あらかじめ定められた範囲 $[-V, V]$ $(V>0)$ を超えた場合，この範囲内に収められる。そして，$\boldsymbol{x}_i$ の各成分 x_{ik} $(k=1, 2, \cdots, n)$ は，$\boldsymbol{v}_i$ の各成分 v_{ik} $(k=1, 2, \cdots, n)$ を用いて，次式により更新される。

$$x_{ik} \leftarrow x_{ik} + v_{ik} \quad \cdots (2)$$

これらの処理は，各個体の各成分について，終了条件を満たすまで繰り返される。

以上より，全体の処理手順をまとめると，以下のようになる。Np は集団に含まれる個体数である。

Step 1　$i=1, 2, \cdots, Np$ について，$\boldsymbol{x}_i$ をランダムに生成する。$\boldsymbol{v}_i$ は $\boldsymbol{0}$ とするか，またはランダムに生成する。

Step 2　$i=1, 2, \cdots, Np$ について，$\boldsymbol{x}_i$ の評価値を求め，$\boldsymbol{x}_i^{\text{best}}$ を記録する。

Step 3　$i=1, 2, \cdots, Np$ について，以下を行う。

3-1　$\boldsymbol{x}_i^{\text{local}}$ を求める。

3-2　式(1)により $\boldsymbol{v}_i$ を更新する。

3-3　式(2)により $\boldsymbol{x}_i$ を更新する。

Step 4　終了条件を満たしていなければ，Step 2 へ戻る。

以上は $\boldsymbol{x}_i$ の成分が連続変数の場合であるが，Kennedy らは 2 値変数に対応した PSO も提案している[8]。それによれば，x_{ik} は式(2)ではなく

$$\text{if } \rho < g(v_{ik}) \text{ then } x_{ik}=1 \text{ else } x_{ik}=0 \quad \cdots (3)$$

のように更新される。ここで $g(v_{ik})$ はシグモイド関数

$$g(v_{ik})=\frac{1}{1+\exp(-v_{ik})} \quad \cdots (4)$$

で，ρ は $\rho \in [0, 1]$ の一様乱数である。このように，v_{ik} は $x_{ik}=1$ とする確率を与えるものとして用いられる。

4．離散型 PSO を用いた解法

4.1 解の表現

PSO を用いたスケジューリング問題の解法を考えるにあたり，まず個体すなわち解をどのように表現するかが問題となる。ここでは，解をジョブ番号の重複順列を用いて表す。この表現では，各ジョブの番号は，それが持つタスクの数だけ繰り返し現れる。k 回目に現れたジョブ番号は，そのジョブの k 番目のタスクを指す。このように解釈することにより，各タスクを順に所定の機械に割り付けて，解はスケジュールに変換される。解の評価は，スケジュールに変換した後に行う。ここでは，$C_{\max}$ の値を評価値として用いる。対象としている問題では，ジョブに含まれるタスクの個数は機械の台数に等しいとしている。それゆえ，個体の長さつまり個体に含まれる要素の個数は，ジョブ数×機械台数となる。

図 2 は，図 1 の問題例において，順列 1 1 2 2 1 2 を解読してスケジュールを作成するようすを示している。順列は，左から順に解読していく。最初の要素 ‘1’ は，ジョブ 1 の 1 番目のタスク 11 を表すので，機械 1 の時刻 0 に割り付ける。次の要素 ‘1’ は，ジョブ 1 の 2 番目のタスク 12 を表す。タスク 12 は，タスク 11 が終わってから処理可能となるので，機械 2 の時刻 1 に割り付ける。その次の要素は ‘2’ なので，ジョブ 2 の 1 番目のタスク 21 である。機械 1 はタスク 11 が終わってから使用可能になるので，タスク 21 は機械 1 の時刻 1 に割り付ける。このような処理をすべての要素について行うことにより，順列はスケジュールに変換される。この例からは，図 1(2) のスケジュール 2 が作られる。$C_{\max}$ は 8 なので，この個体の評価値は 8 となる。

4.2 提案する解法

提案する解法では，上述の順列すなわち離散的な解を直接扱うことができる，離散型 PSO を考えて応用する。このとき，模倣のプロセスすなわちどのように模倣の相手を選び，どのように模倣するかがポイントとなる。ここでは，次のような方法を用いる。

模倣相手の決定　個体 i は，$\boldsymbol{x}_i^{\text{best}}$ と $\boldsymbol{x}_i^{\text{local}}$ のいずれかを確率的に選んで模倣相手とする。前述の 2 値変数 PSO では，v_{ik} から確率を計算して x_{ik} の値として 0 または 1 を選んだ。2 つのうちのいずれかを選ぶという点では，2 値変数 PSO に似ているが，ここでは速度をベクトルではなくスカラー u_i とし，要素単位に値を決めるためではなく，個体を選ぶために用いる。具体的には，式(4)を用いて $g(u_i)$ を計算し，これを $\boldsymbol{x}_i^{\text{best}}$ を模倣する確率と考える。一様乱数 $\rho \in [0, 1]$ に対して $\rho < g(u_i)$ ならば $\boldsymbol{x}_i^{\text{best}}$ を，そうでなければ $\boldsymbol{x}_i^{\text{local}}$ を選ぶ。

u_i は，以下に定義する関数 h を用いて，次式により更新する。

$$u_i \leftarrow u_i + r \cdot h(\boldsymbol{x}_i, \boldsymbol{x}_i^{\text{best}}, \boldsymbol{x}_i^{\text{local}}) \quad \cdots (5)$$

ここで，r は $r \in [0, R]$ $(R>0)$ の一様乱数である。関数 $h(\boldsymbol{x}_i, \boldsymbol{x}_i^{\text{best}}, \boldsymbol{x}_i^{\text{local}})$ は，個体の評価関数 $f(\boldsymbol{x}_i)$ を用いて次のように定義する。

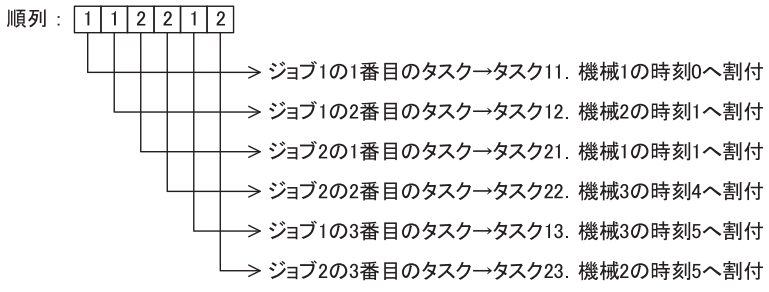


図 2　解の表現とスケジュールへの変換

$$h(\boldsymbol{x}_i,\ \boldsymbol{x}_i^{\text{best}},\ \boldsymbol{x}_i^{\text{local}})$$

$$=\begin{cases} 1\text{；}(f(\boldsymbol{x}_i)\geq f(\boldsymbol{x}_i^{\text{best}})\text{ または }f(\boldsymbol{x}_i)\geq f(\boldsymbol{x}_i^{\text{local}}))\\ \quad\text{かつ }f(\boldsymbol{x}_i^{\text{best}})<f(\boldsymbol{x}_i^{\text{local}})\text{ のとき}\\ -1\text{；}(f\ (\boldsymbol{x}_i)\geq f(\boldsymbol{x}_i^{\text{best}})\text{ または }f(\boldsymbol{x}_i)\geq f(\boldsymbol{x}_i^{\text{local}}))\\ \quad\text{かつ }f(\boldsymbol{x}_i^{\text{best}})>f(\boldsymbol{x}_i^{\text{local}})\text{ のとき}\\ 0\text{；それ以外のとき}\end{cases}$$

$$\cdots\ (6)$$

この関数 h は，$\boldsymbol{x}_i$ の現在の評価値より $\boldsymbol{x}_i^{\text{best}}$ もしくは $\boldsymbol{x}_i^{\text{local}}$ の評価値の方がよいとき，よい方を選ぶ確率がより高くなるように u_i の値を変える。$\boldsymbol{x}_i^{\text{best}}$ の方がよければ，u_i の値を増やして，$\boldsymbol{x}_i^{\text{best}}$ を選ぶ確率が高くなるようにする。一方，$\boldsymbol{x}_i^{\text{local}}$ の方がよければ，u_i の値を減らして，$\boldsymbol{x}_i^{\text{best}}$ を選ぶ確率が低くなるようにして，$\boldsymbol{x}_i^{\text{local}}$ を選ぶ確率が高くなるようにする。

u_i があらかじめ定められた範囲 $[-V,\ V]\ (V>0)$ を超えた場合は，連続型 PSO と同様にこの範囲内に収めるようにする。

模倣のしかた　模倣は，模倣相手の部分列すなわち 1 つ以上の連続した要素を選び，この部分列をそれと同じ $\boldsymbol{x}_i$ の位置に複写することにより行う。部分列の位置と長さは，ランダムに決定する。$\boldsymbol{x}_i$ の残りの部分は，現在の要素を先頭から順に複写する（図 3）。そのとき，各ジョブ番号がそのタスク数を超えて現れないようにして，実行不能なスケジュールを生じないようにする。

提案する解法の全体の処理手順は，以下のとおりである。

Step 1　$i=1,\ 2,\ \cdots,\ Np$ について，ジョブ番号の重複順列をランダムに生成して，$\boldsymbol{x}_i$ を初期設定する。$u_i=0$ とする。

Step 2　$i=1,\ 2,\ \cdots,\ Np$ について，$\boldsymbol{x}_i$ の評価値を求め，$\boldsymbol{x}_i^{\text{best}}$ を記録する。

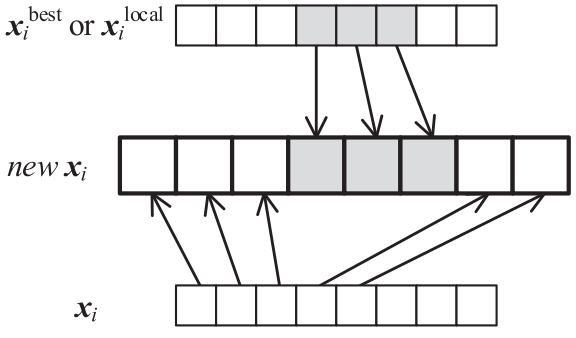


図 3　新しい解の生成

Step 3　$i=1,\ 2,\ \cdots,\ Np$ について，以下を行う。

3-1　$\boldsymbol{x}_i^{\text{local}}$ を求める。

3-2　式 (5) により u_i を更新する。u_i が $[-V,\ V]$ を超えたときは，この範囲内に収める。

3-3　u_i を用いて式(4)により確率を計算し，$\boldsymbol{x}_i^{\text{best}}$ と $\boldsymbol{x}_i^{\text{local}}$ から模倣相手を選ぶ。

3-4　模倣相手からランダムに部分列を選ぶ。これと $\boldsymbol{x}_i$ から新しい $\boldsymbol{x}_i$ を作成する。

Step 4　終了条件を満たしていなければ，Step 2 へ戻る。

5．計算機実験

5.1　実験と結果

前節で提案した解法をもとにプログラムを作成し，計算機実験を行った。実験に使用した PC は，CPU　Intel Core i3　2.93 GHz，メモリ 4 GB である。プログラムは，Visual C＋＋を用いて作成した。総処理時間 $C_{\max}$ の最小化問題を対象とし，よく知られているベンチマーク問題 'FT10'（10 ジョブ 10 機械，最適解 $C_{\max}=930$）を問題例として用いた。

今回は，つながりのある個体の範囲 Ne，式(5)で用いられる乱数の範囲 R，u_i の限界値 V の 3 つを変えて，結果を比較した。実験に用いた値は，$Ne\in\{1,\ 2,\ 4,\ 10\}$，$R\in\{0.1,\ 0.5,\ 1.0,\ 2.0,\ 4.0\}$，$V\in\{1.0,\ 2.0,\ 4.0\}$ である。その他，模倣する部分列の長さは，1 から個体の長さまでからランダムに選び，集団に含まれる個体数は $Np=200$ とした。終了条件は，Step 2〜4 の繰返しが指定した回数 Nr を超えたときに終了するものとし，$Nr=10000$ とした。

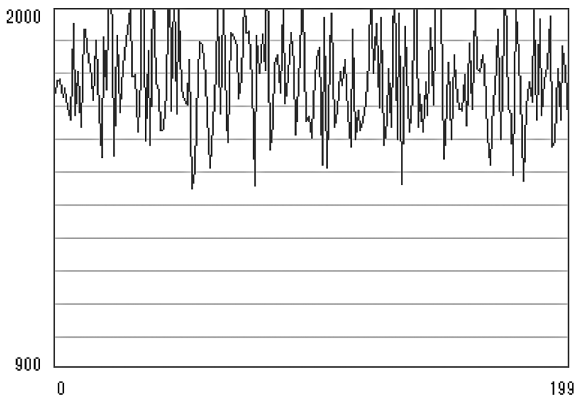
表 1 は，Ne，R，V のそれぞれの組合せについて，それぞれ 100 回ずつ実行したときの，$C_{\max}$ の最小値，最大値，平均，標準偏差を示している。1 回の実行に要した処理時間は，約 25 秒だった。$(Ne,\ R,\ V)=(4,\ 4.0,\ 1.0)$，$(10,\ 2.0,\ 4.0)$ のとき，全体の最小値 937 が得られた。これは，最適解より 0.8%大きい値である。その他の組合せについては，最適解との差は 5.3%以下である。

5.2　考察

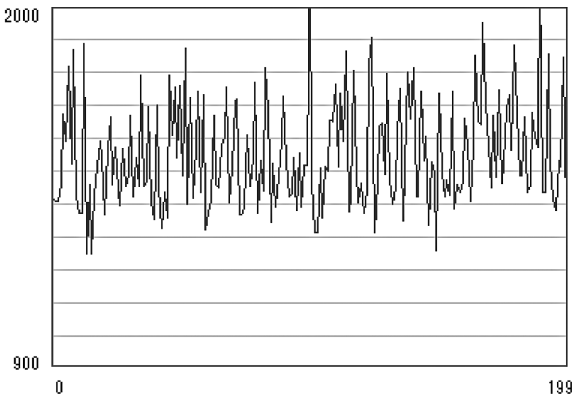
実行中の個体の評価値は，どのパラメータについても概ね図 4 のように推移していった。横軸は個体の番号，縦軸は個体の評価値を表している。最小化問題なので，値が小さいほど評価がよいことになる。初期状態では各個体さまざまな評価値を取っているが，処理が繰り返されるにつれて全体的に改善され

表 1 実験結果（*Ne*，*R*，*V* の各組合せについて 100 回実行）

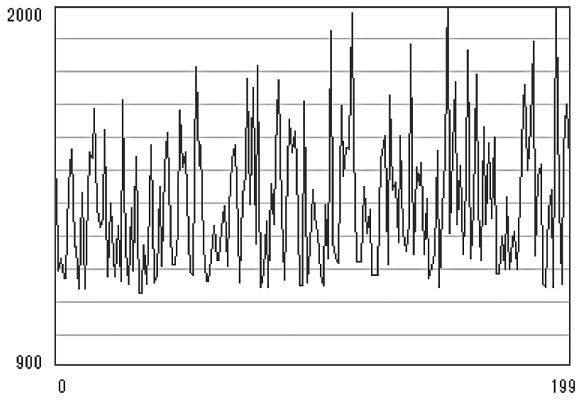
R	*V*	*Ne* = 1				*Ne* = 2				*Ne* = 4				*Ne* = 10			
		最小値	最大値	平均	標準偏差	最小値	最大値	平均	標準偏差	最小値	最大値	平均	標準偏差	最小値	最大値	平均	標準偏差
0.1	1.0	971	1046	1011.9	16.8	965	1051	1003.5	18.2	963	1055	1015.3	19.8	959	1080	1022.8	25.8
	2.0	972	1049	1014.4	14.3	961	1056	1010.2	18.1	967	1076	1015.8	22.6	979	1088	1023.8	25.2
	4.0	978	1058	1023.1	15.9	958	1046	1010.9	18.0	961	1066	1014.1	23.9	974	1106	1023.4	27.0
0.5	1.0	965	1045	1011.9	16.8	957	1044	1005.4	18.7	957	1066	1015.4	22.1	967	1100	1030.0	27.1
	2.0	974	1047	1013.8	15.2	962	1054	1010.5	20.5	970	1067	1019.2	22.0	972	1093	1029.0	24.3
	4.0	970	1057	1015.7	17.5	961	1071	1012.7	19.0	961	1079	1015.7	25.4	979	1108	1032.9	25.9
1.0	1.0	960	1048	1013.8	18.3	967	1067	1008.5	19.1	952	1057	1017.1	20.6	971	1077	1023.9	22.7
	2.0	971	1049	1012.6	15.2	946	1065	1009.4	21.0	971	1077	1020.6	20.9	979	1094	1026.1	23.0
	4.0	974	1044	1014.0	17.1	958	1048	1009.3	20.5	939	1106	1018.9	25.4	978	1125	1031.3	28.4
2.0	1.0	962	1045	1011.2	17.2	950	1048	1004.2	19.2	961	1078	1011.8	22.4	962	1099	1028.1	26.7
	2.0	973	1051	1008.7	18.3	946	1065	1013.4	19.6	970	1083	1020.7	22.4	961	1101	1028.7	26.4
	4.0	975	1052	1012.9	17.0	961	1063	1008.7	22.5	970	1075	1020.7	22.9	937	1124	1031.8	30.4
4.0	1.0	974	1056	1009.9	15.0	942	1050	1010.0	20.3	937	1081	1013.8	25.1	968	1088	1025.6	22.7
	2.0	952	1050	1014.5	17.2	962	1062	1012.0	21.1	967	1073	1020.7	22.3	971	1111	1038.7	29.8
	4.0	970	1052	1014.0	17.8	956	1061	1009.9	22.3	946	1073	1014.6	23.9	967	1119	1032.0	28.8



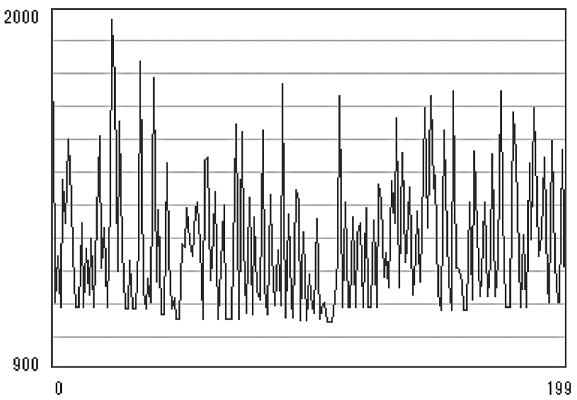
（1）初期状態



（2）繰返し 100 回後



（3）繰返し 1000 回後



（4）繰返し 10000 回後

図 4 個体の評価値の推移（横軸：個体番号，縦軸：評価値 C_{max}）

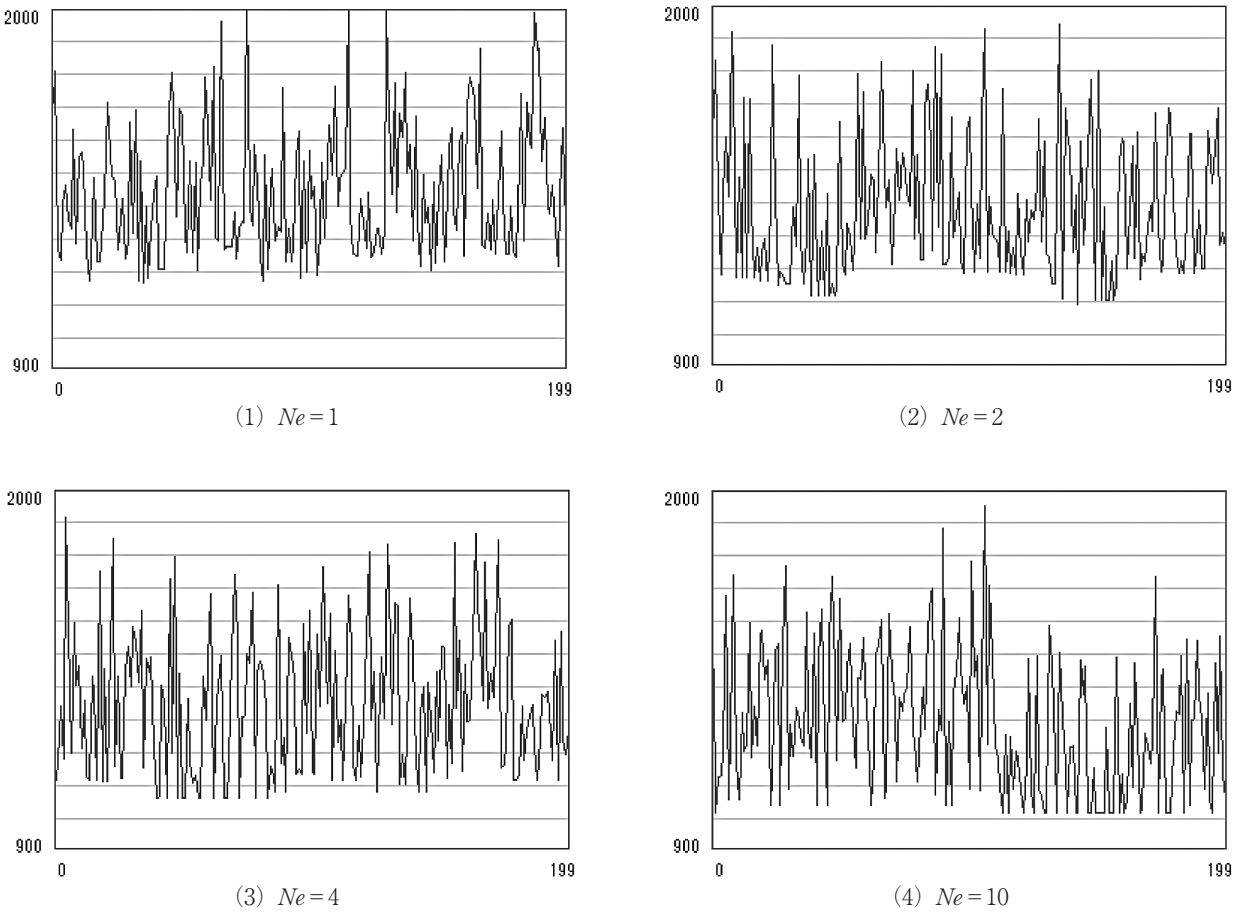


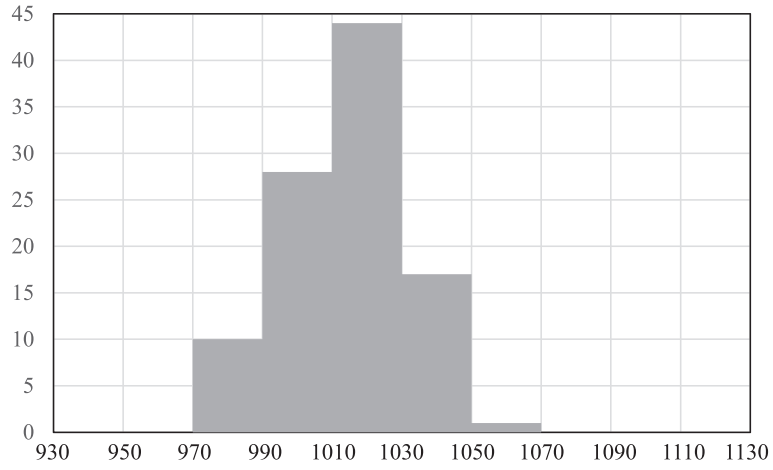
図 5　つながりの範囲の影響（横軸：個体番号，縦軸：評価値 $C_{\max}$）

ることがわかる。この方法では，個体は常に変更されるため，同じ評価値に留まることはまれである。評価値は常に変化し，グラフは振動し続けることが多い。これは，さまざまな評価値の解が残り続けること，つまり解の多様性が維持されることを示している。この解法では，多様性が失われると，それ以上解の改良が進まなくなる。しかし，多様な解が残っていると，さらに繰り返し実行することにより，解の更新が継続されて，よりよい解が見つけられる可能性がある。

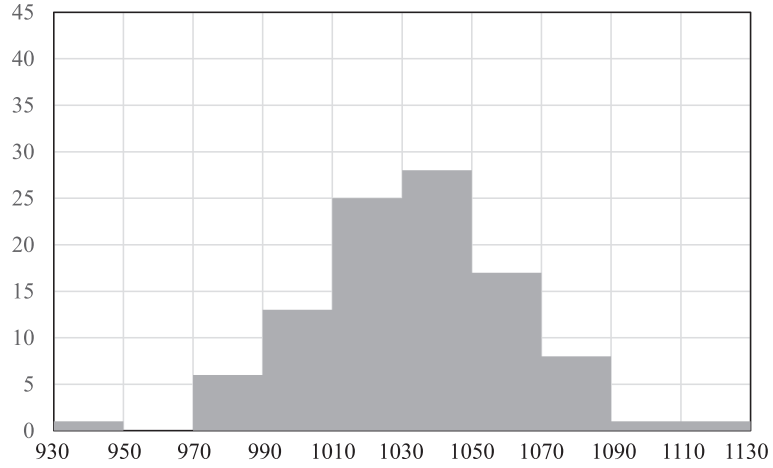
つながりのある個体の範囲 Ne の影響を考える。Ne の値が大きいほど，情報交換が可能な個体が増える。したがって，評価のよい個体の情報が，より広く周囲に伝わることになる。図 5 は，Ne＝1，2，4，10 について，繰返し回数 Nr＝500 のときの各個体の評価値を示している。Ne の値が大きいほど，最良値が同じである個体が早く増えて，そのグループが形成されやすい。グループの形成が早いと，周囲が同質の個体になるため，個体の変化はあまり進まなくなる。パラメータの各組合せについて調べたところ，Ne が大きいほど標準偏差は大きくなる傾向が見られた。これは，このようなことが反映されたためと考えられる。つまり，Ne が大きいと情報の拡散が速いため，グループ形成が早く進む。そのため，試行ごとに違うグループができやすくなり，評価の差が大きくなったと考えられる。

図 6 は，R＝2.0，V＝4.0 のときに得られた，典型的な解の分布を示している。Ne＝10 のとき，最小値は最適解に近いが，最大値はかなり大きく，広い範囲に分布していることがわかる。一方，Ne＝1 のときは，最小値は Ne＝10 のときに及ばないものの，最大値は小さく，分布の範囲は狭い。平均を比較すると，Ne＝1 の方がよい。他の R と V の組についても，ほぼ同様の傾向が見られた。

式(5)で用いられる乱数の範囲 R の値は，模倣相手の選択確率の変化に影響を与える。R が大きいと，u_i が大きく変わり，$\boldsymbol{x}_i^{\text{best}}$ もしくは $\boldsymbol{x}_i^{\text{local}}$ のいずれかを選ぶ確率が変わりやすい。一方，R が小さいと，u_i の変化は小さく，選択確率はあまり変化しない。今回の実験では，同じ Ne について，$R \geqq 1.0$ の



（1）$Ne=1$



（2）$Ne=10$

図 6 解の分布（$R=2.0$，$V=4.0$ のとき．横軸：評価値 C_{max}，縦軸：度数）

ときによい結果が得られることが多かった。

u_i の限界値 V の値は，模倣相手の選択確率の範囲を制限する。$V=1.0$，2.0，4.0 のとき，確率の範囲はそれぞれ 0.27〜0.73，0.12〜0.88，0.02〜0.98 となる。つまり，V が大きいと，各個体について模倣相手がどちらか一方に偏る可能性が上がる。V が小さいと，それは弱まる。今回の実験では，同じ Ne について，$V=1.0$ のときによい結果が得られることがやや多いようだが，それほど明瞭には見られなかった。

6．おわりに

この論文では，ジョブショップ・スケジューリング問題への PSO を用いた解法を提案した。この方法では，解をジョブ番号の重複順列で表した。解の更新は，よりよい解から部分列を複写することにより行った。複写元となる解は確率的に選ぶようにした。この確率は，よりよいものを選ぶように変化させた。

この方法をもとにプログラムを作成し，計算機実験を行った。問題例としてベンチマーク問題 'FT10' を用いて，この解法の基本的な性質を調べた。今回は，Ne，R，V の 3 つのパラメータについて，結果への影響を調べた。

この他に，PSO には集団サイズつまり集団に含まれる個体数 Np と繰返し回数 Nr というパラメータもある。いずれも，計算時間に直接影響を及ぼすものである。今後，これらの影響を調べる必要がある。いうまでもなく，どちらも小さい方が計算時間は短くなるので，短い計算時間でよりよい解を得ることが課題となる。

参考文献

［1］Pinedo, M.. Scheduling: Theory, Algorithms, and Systems. 2nd ed., Prentice-Hall, 2001, 586p.

［2］黒田　充, 村松健児編．生産スケジューリング．朝倉書店，2002，272p.

［3］Pinedo, M.. Planning and Scheduling in Manufacturing and Services, Springer, 2005, 506p.

［4］Blazewicz, J., Ecker, K. H., Pesch, E., Schmidt, G. and Weglarz, J.. Scheduling Computer and Manufacturing Processes. Springer, 1996, 491p.

［5］久野誉人，繁野麻衣子，後藤順哉．数理最適化．オーム社，2012，262p.

［6］久保幹雄，J.P. ペドロソ．メタヒューリスティクスの数理．共立出版，2009，227p.

［7］Kennedy, J. and Eberhart, R.. Particle Swarm Optimization. Proceedings of the IEEE International Conference on Neural Networks IV (ICNN 95), 1995, p. 1942–1948.

［8］Kennedy, J. and Eberhart, R.. A Discrete Binary Version of the Particle Swarm Algorithm. Proceedings of the 1997 Conference on Systems, Man, and Cybernetics, 1997, p. 4104–4108.

Abstract

Herein, a particle swarm optimization (PSO) algorithm for the job-shop scheduling problem has been proposed. The job-shop scheduling problem belongs to a hard class of combinatorial optimization problems. It is difficult to obtain an optimal solution for such problems. Thus, a near-optimal solution is obtained using heuristics. PSO is a heuristic approach for continuous optimization problems, which is inspired by the rules underlying the social behavior of animals, such as birds and fish. The particles of PSO are considered as the solutions of the target problem. The solutions of job-shop scheduling problems are discrete. Hence, we modified PSO to treat such problems. The particles in our algorithm are represented by repeated permutations of the job numbers. They are updated by copying a part of the permutation of other particles, which is chosen based on the probability. Then, the probability is changed in order to choose better particles. Computer experiments were performed with the famous benchmark problem “FT10.” Consequently, we obtained the best result, which is worse than the optimal solution by 0.8%.