A-026

フローショップスケジューリング問題に対する反復局所探索法における摂動処理の改良

Improved Perturbation Processing in Iterated Local Search for the Flowshop Scheduling Problem

伊東　駿[1)]　井関　智也[1)]　小田　哲也[1)]　片山　謙吾[1)]
Shun Ito[1)]　Tomoya Iseki[1)]　Tetsuya Oda[1)]　Kengo Katayama[1)]

1　まえがき

総処理時間を最小化する生産スケジューリング問題の代表例として，順列フローショップスケジューリング問題 (Permutation Flowshop Scheduling Problem，PFSP) が挙げられる．PFSP に対する効率的な解構築法として NEH 法 [1] がある．NEH アルゴリズムはジョブを一つずつ総処理時間が最小となるように部分列に挿入し解を構築する．Ruiz らはこの処理を利用した反復局所探索法 (Iterated Local Search, ILS) [2] を提案し，より高性能な結果が得られることを示した．これらの NEH や ILS においては総処理時間が最小となるようにジョブを挿入するとき，総処理時間が同値になるような挿入箇所が複数存在する場合がある．そこで Fernandez-Viagas らはジョブを挿入するとき，同値となる候補から適切なジョブを選択するタイブレーク機構 TB_{FF} [3] を提案し，TB_{FF} を導入した ILS はさらに良好な解を算出できることを示した．この Fernandez-Viagas らの ILS は PFSP に対する最も高性能なメタ戦略アルゴリズムの一つとして広く知られている．

ILS は初期解生成，摂動処理，局所探索，解受入れ基準の 4 つのフェーズで構成される．従来 ILS の摂動処理においてはランダムな値（ジョブ）を取り除くシンプルな方法が採用されているが，摂動処理においては探索の状況に応じて取り除く適切なジョブが存在すると考えられる．本研究では，探索の状況に応じた摂動処理を行う ILS を提案し，従来の ILS との比較実験を通して提案 ILS の有効性を示す．

2　順列フローショップスケジューリング問題

順列フローショップスケジューリング問題は，複数の機械 m 台を用いて n 個の仕事 (ジョブ) を処理する問題である．様々な制約条件，目的関数のもとに成り立ち，PFSP は以下の制約条件のもとで目的関数である総処理時間 $C(\pi)$ が最小となるスケジュール解 π を求める問題である．式 (1) の $f_{i,j}$ は機械 j までのジョブ i の完了時間を示している．

＊各仕事は二つ以上の機械で同時に処理されない．
＊各機械は二つ以上の仕事を同時に処理しない．
＊全てのジョブで機械の順序は一定である．

$$C(\pi) = \max(f_{i,j}) \quad j \in M, i \in J \tag{1}$$

3　反復局所探索法（ILS）

Ruiz ら [2] が提案した反復局所探索法 (Iterated Local Search, ILS) は極めて効率的なメタ戦略である．図 1 に ILS の疑似コードを示す．

Line 1 では，初期解生成法として Nawaz らが提案した NEH アルゴリズム [1] を使用し解 π を生成する．NEH アルゴリズムは大きく 2 つの処理に分けられる．まず一つ目は優先度ルール (処理時間の平均) に基づいてジョブの順序付けを行い，二つ目にジョブ順序に基づいてジョブを一つずつ総処理時間が最小となるように部分列に挿入し解を構築する．Line3-14 までの 1 反復の処理に

```
     procedure Iterated Local Search
     d =Number of jobs to remove
     begin
1        generate initial solutions π;
2        π_b:=π;
3        repeat
4            π':= π
5            π':= Perturbation(π',d);
6            π'':=Local Search(π');
7            if C_max(π'') < C_max(π) then
8                π:= π'';
9                if C_max(π'') < C_max(π_b) then π_b:= π'';
10           elseif (random ≤ exp−(C_max(π'') − C_max(π))/Temperture then
11               π:= π'';
12           endif
13       until terminate := true
14       return π_b
     end
```

図 1　Iterated Local Search の疑似コード

おいては摂動処理，局所探索，解受入れ基準で構成され，これを終了条件を満たすまで繰り返す．Line 5 では摂動処理として構築された解からジョブをランダムに d 個取り除き，取り除いたすべてのジョブをそれぞれ最良の解の評価値が得られる位置に挿入する．Line 6 では以下の局所探索を行う．

1. π' に対してランダムかつ，局所探索内で選択していない一つのジョブを取り除く．
2. 取り除いたジョブを最良の解の評価値が得られる位置に追加する．
3. 1,2 を n 回繰り返す
4. 解が更新されていた場合 1 に戻る．
5. 改善された解 π'' を出力する．

Lines 7-13 では次の反復の対象となる解の決定を行う．$Temperture$ は以下の式によって計算される．

$$Temperature = T \cdot \frac{\sum_{i=1}^{m}(\sum_{j=1}^{n}(P_{ij}))}{n \cdot m \cdot 10} \tag{2}$$

4　TB_{FF}

ILS の初期解生成や局所探索などのフェーズにおいて，ジョブを暫定的に挿入し最小の総処理時間となる部分スケジュールを選択するとき，総処理時間が同値となる場合がある．Fernandez-Viagas らはジョブ挿入時の同値となる候補から適切なジョブを選択するタイブレーク機構 TB_{FF} を提案した [3]．この TB_{FF} は計算量を増加させることなく，ジョブ挿入時に総処理時間が同値となった時のみ実行される．実行時には機械の総待機時間 it を計算し，その値が最小となるスケジュールを部分スケジュールとする．図 2 は $J1$ と $J2$ の間 (2 番目) に新しいジョブを挿入した場合の機械の待機時間 it の一例を示している．l 番目に新しいジョブを挿入したときの機械の総待機時間 $it(l)$ は式 3 で計算される．ただし，$l = n$ の時，Δ'_{il+1} を考慮しない．

図 2　TB_{FF} におけるタイブレーク時の一例

$$it(l) = \sum_{j=2}^{m}(\Delta'_{il} + \Delta'_{il+1}) \tag{3}$$

1)　岡山理科大学

```
procedure Proposed Iterated Local Search
d : Number of jobs to remove
L = {L_1, ..., L_n} : Number of moves for each job in local search
begin
1    generate initial solutions π;
2    π_b:=π;
3    itecount := 0;
4    repeat
5        π' := π
6        π' :=Improved Perturbation(π',d,L);
7        π'' :=Local Search(π',L);
8        if C_max(π'') < C_max(π) then
9            π := π'';
10           if C_max(π'') < C_max(π_b) then π_b:= π'';
11       elseif (random ≤ exp−(C_max(π'') − C_max(π))/Temperture then
12           π := π'';
13       endif
14       itecount + +;
15       if (itecount = lc) then
16           itecount= 0;
17           level + +;
18           if (level = n − m) then level = 0;
19       endif
20   until terminate := true
21   return π_b
end
```

図 3　提案 Iterated Local Search の疑似コード

```
procedure Improved Perturbation
π': Solution
d : Number of jobs to remove
L = {L_1, ..., L_n} : Number of moves for each job in local search
begin
1    D := {∅};
2    L' := L
3    P :=Sort the jobs in descending order of L'_i;
4    for i := level to level + m do ;
5        D ∈ D ∪ P_i;
6    endfor
7    for i := 1 to d do ;
8        π_{r_i} := select one job at random from D ;
9        D := D\{π_{r_i}} ;
10   endfor
11   for i := 1 to d do ;
12       π' :=remove π_{r_i} from π';
13   endfor
14   for i := 1 to d do ;
15       π' :=best permutation obtained by inserting job π_{r_i} in all possible posi-
     tions of π';
16   endfor
17   return π'
end
```

図 4　Improved Perturbation の疑似コード

5　提案法 ILS

従来 ILS の摂動処理では図 1 の Line 5 でランダムにジョブを d 個取り除いている．本研究では，摂動処理において注目するジョブを探索の状況に応じて柔軟に選択することで，より効果的な探索を可能にするものと考えられる．そこで提案する ILS の摂動処理では局所探索においてジョブを移動させたとき，解の精度を向上させたジョブごとの移動回数に応じて取り除くジョブを選択する．提案 ILS の疑似コードを図 3 に示す．

提案 ILS の構成は図 1 に示した従来 ILS の初期解生成，摂動処理，局所探索，解受入れ基準と同様であるが，提案法の図 3 では摂動処理を改良している．それに併せて Line 7 の Local Search では解の精度向上に貢献した各ジョブの移動回数の記録を行う．具体的に Line 7 では取り除いたジョブ i を最良の解 π'' が得られる位置に追加したとき，追加前と異なる場所に挿入された場合，ジョブ i の移動回数を記録している L_i のカウントを増やす．Lines 15-19 では摂動処理の注目する領域の処理を行っている．Line 15 では Lines 4-20 の反復が繰り返されることで $itecount$ の数が 1 増加する．Lines 16-20 では $itecount$ の数が lc と同値になったときに $itecount$ の値を 0 に戻し，摂動処理の候補の選択に影響する変数 $level$ の値を増加させる．$level$ の値は $n-m$ の値を超えたとき $level$ を 0 とする．

図 4 は図 3 の Line 6 で示した提案する摂動処理である．Line 3 では L' を基準にジョブを並び替える．Lines 4-6 では P のうち $level$ から $level+m$ までを摂動処理の対象となるジョブ候補の集合 D に代入する．Lines 7-10 では D からランダムに摂動処理の対象となるジョブを d 個選択し，そのジョブは集合 D から取り除かれる．Lines 11-13 では解 π' から π_r を取り除く．Lines 14-15 では取り除いたすべてのジョブをそれぞれ最良の解の評価値が得られる位置に挿入する．提案 ILS に対して上述した TB_{FF} を導入したものを提案 ILS_{FF} とする．

表 1　Taillard ベンチマークに対する実験結果

Instance		従来 ILS_{FF}		提案 ILS_{FF}	
n	m	best	AVG	best	AVG
20	5	**0.000**	**0.016**	**0.000**	0.024
20	10	**0.000**	**0.000**	**0.000**	**0.000**
20	20	**0.000**	**0.000**	**0.000**	**0.000**
5	5	**0.000**	**0.000**	**0.000**	**0.000**
50	10	**0.000**	0.416	**0.000**	**0.397**
50	20	0.297	0.833	**0.249**	**0.777**
100	5	**0.000**	**0.000**	**0.000**	0.001
100	10	**0.000**	**0.055**	**0.000**	0.056
100	20	0.497	0.905	**0.451**	**0.798**
200	10	**0.000**	0.032	**0.000**	**0.029**
200	20	0.391	0.713	**0.355**	**0.666**
500	20	**0.000**	0.271	**0.000**	**0.246**
AVERAGE		0.099	0.270	**0.088**	**0.250**

6　実験結果

提案 ILS_{FF} の性能を評価するために従来の ILS に TB_{FF} を適用した従来 ILS_{FF} との比較実験を行った．各アルゴリズムのパラメータを $d=4, T=0.4$ とし，提案 ILS_{FF} の lc は $3\cdot n$ とした．各アルゴリズムは C 言語によってコード化し，コンパイラは最適化オプション-O3 を付与した gcc(Ver10.3.0) を使用した．すべての計算は，計算機 3.3GHz Intel(R) Xeon(R) CPU E3-1230 v3，RAM:8GB 上で実行した．対象とする問題例は PFSP の代表的なベンチマーク問題例である Taillard のベンチマーク 120 例題を使用した．1 例題に対して各解法を 10 回試行し，各アルゴリズムの計算打ち切り時間は $100\cdot n\cdot m$[ms] とした．表 1 は左から，ジョブ数 n，機械数 m，各解法の 10 試行の解の最良精度 (best)，それぞれの解の平均精度 (AVG) を示した．なお，得られた解 π の目的関数値の精度 (%) は $\frac{C(\pi)-C(\pi^*)}{C(\pi^*)}\times 100$ によって算出した．式中の $C(\pi)$ は得られた解 π の評価値，π^* は既知の最適解を表している．各解法のそれぞれの結果において，より良好な解の精度を太字で示した．表 1 において提案 ILS_{FF} は従来 ILS_{FF} より平均的に良好な結果を示した．特に $n=50, m=20$ や $n=100, m=20$，$n=200, m=20$ などのジョブ数に対する機械数が多い問題例に対して良好な結果を示した．以上より提案 ILS_{FF} は従来 ILS_{FF} に平均的に良好な解を算出可能なことを示した．

7　むすび

本研究では，探索の状況に応じた適切なジョブを摂動処理で選択する ILS を提案し，性能を評価した．実験結果から提案 ILS は高性能解法として知られている従来 ILS より良好な解を算出し，提案法の有効性を示した．今後の発展として，提案 ILS のパラメータの調整などが考えられる．

謝辞

本研究の一部は JSPS 科研費（基盤研究（C）19K12166）の助成を受けたものである．

参考文献

[1] M.Nawaz, E.Enscore and I.Ham, “A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem,” Omega, Vol.11, No.1, pp. 91-95, 1983

[2] R.Ruiz and T.Stutzle, “A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem,” European Journal of Operational Research, Vol.177, No.3, pp. 2033-2049, 2007

[3] V.Fernandez-Viagas, J.M. Framinan, “On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem,” Computers & Operations Research, Vol.45, pp. 60-67, 2014