岡山理科大学紀要　第59号 A pp. 63-71 (2023)

フローショップスケジューリング問題に対する拡張 NEH 法

伊東　駿・小田　哲也[1)]・片山　謙吾[1)*]

岡山理科大学大学院工学研究科情報工学専攻

1) 岡山理科大学工学部情報工学科

(2023年10月31日受付、2023年11月29日受理)

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 の総処理時間を最小化するとき，機械の台数が 2 台の場合 Johnson のアルゴリズム[10]によって多項式時間で厳密に解くことが可能である．しかしながら，機械の台数が 3 台以上の場合，NP 困難[9]であることが知られている[7]．PFSP において実行可能解はn！通りあり，全ての解を列挙できれば総処理時間が最小になる解を得られるが，nが大きい場合，現実的な時間ではすべての解を列挙できない．そのため現実的な時間で近似解を算出する様々な近似解法が提案されている．Campbell らは Johnson のアルゴリズムを拡張した CDS[5]を提案した．CDS は複数の機械を 2 台に仮想的に置き換え Johnson のアルゴリズムを適用するアルゴリズムである．また Palmer[1]は，各ジョブの処理時間に基づいて，slope index を計算し，slope index に基づいてジョブをスケジューリングするアルゴリズムを提案した．その後，Gupta[6]は，slope index を異なる方法で算出したアルゴリズムを提案した．これらの様々な近似解法の中でも，NEH[8]は PFSP に対して最も効率的な近似解法として広く知られている．NEH の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)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．特に一つ目のジョブの順序付けは解の精度に大きく影響を与えるため様々な優先度ルールが提案されてきた．Li ら[13]は，処理時間の偏差を考慮した新しい優先ルールを提案した．Dong ら[12]は，処理時間の平均と不偏標準偏差をジョブの順序付けに使用することで，優先度ルールを改良した NEH-D を提案し，オリジナルの NEH よりも良好な結果を得た．Liu ら[11]が提案した PR_{SKE} は，処理時間の平均と不偏標準偏差の 2 つのモーメントで行うジョブの順序付けでは十分でないと考え，NEH-D の優先度ルールに第 3 のモーメントである歪度を追加した．これらの優先ルールの変更により，NEH の計算量を増やすことなく，得られる解の質が向上した．多くの問題例において PR_{SKE} は NEH や NEH-D よりも良い解を得ることが期待できるが，問題例によって PR_{SKE} は NEH や NEH-D よりも良好な結果が得られない場合がある．このように様々な問題例に対して単一で有効な優先度ルールは存在しないと考えられるため，本論文では 3 つのモーメントの適切な組み合わせからなる複数の優先ルールを段階的に用いる，NEH の拡張アルゴリズム Extend NEH（E-NEH）を示す．また Taillard と VRF ベンチマーク問題例に対して適用した性能結果を示す．

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)を以下の制約条件のもと処理する問題である．

＊各ジョブは複数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．

＊全てのジョブは 1, 2,..., mの順序でm個の工程を経て完了される．

PFSP の中にはより強い制約を持つものがある．より現実的な制約条件の例として，各ジョブを処理する前に機械の段替時間を要する問題がある．このような問題を段替時間付きフローショップスケジューリング問題と呼ぶ．また，製造業では，同一の製品を複数作る場合，ロット単位で製造する場合がある．これを考慮した問題をロット付きフローショップスケジューリング問題と呼ぶ．このように，PFSP は様々な形で拡張されてきた．これらの PFSP の目的は次のような例である．

＊総処理時間の最小化

＊各ジョブの取り掛かりから完了までの時間の和を最小化

＊全てのジョブの完了期限である納期からの遅れを最小化
＊納期に間に合わない製品の最小化

本論文における PFSP の目的は，最も一般的に研究されている総処理時間を最小化することである．本稿で使用する目的関数といくつかの表記を以下に示す[11].

$$Min: F = C_{[n],m}$$

n :ジョブの総数
m :機械の総数
i :ジョブのインデックス，$1 \leqq i \leqq n$
k :機械のインデックス，$1 \leqq k \leqq m$
$[i]$: i番目にスケジュールされたジョブ
$t_{i,k}$:ジョブ i の機械 k での処理時間
$C_{[i],k}$:機械k上のi番目のジョブの完了時間
AVG_i :ジョブiの処理時間の平均
STD_i :ジョブiの処理時間の不偏標準偏差
SKE_i :ジョブ i の処理時間の歪度

３．従来法

３－１　NEH

PFSP に対して効率的な近似解法である NEH において，全ての機械における合計処理時間が長いジョブは，合計処理時間が短いジョブよりも優先してスケジュールするべきであるという仮定に基づいている．以下に Nawaz ら[8]の NEH アルゴリズムの手順を示す.

STEP1　n個の各ジョブiについて，m台の機械での合計処理時間$T_i = \sum_{k=1}^{m} t_{i,k}$を計算する．ここで，$t_{i,k}$は機械$k$上のジョブ$i$の処理時間である.

STEP2　T_iの降順にジョブを並べたジョブ順列をリスト L とする.

STEP3　ステップ 2 のリスト L の 1 番目と 2 番目の位置から 2 つのジョブを取り出し，2 通りの解の総処理時間を計算することによって，これら 2 つのジョブの最適な解を見つける．残りのステップでは，これら 2 つのジョブの相対的な位置は変更されない．ここで，$i = 3$ とする.

STEP4　ステップ 2 で生成されたリスト L のi番目の位置にあるジョブを選択し，既に割り当てられているジョブの互いに対する相対的な位置を変更することなく，前のステップで見つかった部分解の挿入可能なi箇所に暫定的に挿入し，その中から最良の解を選択する.

STEP5　$n = i$であればアルゴリズムを終了し，そうでなければ$i = i + 1$とし以降 STEP4,5 を繰り返す.

３－２　NEH-D 及び PR_{SKE}

NEH の STEP1 では，各ジョブの合計処理時間を計算し，ジョブの順序付けを行う．しかし，合計処理時間は同じ値になる可能性があり，これらの同値の解決の仕方は解の質に影響を与える．そこで，Dong ら[12]は，ジョブの順序付けに合計処理時間の平均AVGに不偏標準偏差STDを加えることで，これを改善した．不変標準偏差のモーメントを適切に扱う上で，合計処理時間の代わりにその平均値を使用した．しかし，Liu ら[11]はこれらの 2 つのモーメントだけでは，複雑な問題においてはジョブを区別するには不十分であるとしている．そこで，Liu ら[11]はより高次元のモーメントである歪度を用いて，さらにジョブを区別している．このように，第 3 のモーメントを追加することで，計算量を増やすことなく，また同値になることを避け，処理時間分布を正確に特徴付けることができる．この優先度ルールは，複数のジョブが同じ値のAVGとSTDを持つ場合，歪度の絶対値が大きいジョブを優先すべきであるという新しい仮説に基づいて提案された．ジョブの処理時間分布が歪んでいる場合，正規分布のジョブよりも処理時間が平均から外れることになる．従って，歪度が大きいジョブは優先度が高くなる．この仮説では，分布の正の歪度と負の歪度は同

じ影響を持つので，歪度の絶対値を使用する．3 つのモーメントであるAVG，STD，SKEは以下のように定義される．

$$AVG_i = \frac{T_i}{m}$$

$$STD_i = \sqrt{\frac{1}{m-1}\sum_{k=1}^{m}(t_{i,k} - AVG_i)^2}$$

$$SKE_i = \frac{\frac{1}{m}\sum_{k=1}^{m}(t_{i,k} - AVG_i)^3}{\left(\sqrt{\frac{1}{m}\sum_{k=1}^{m}(t_{i,k} - AVG_i)^2}\right)^3}$$

４．優先度ルールの比較実験

上述したように，NEH の優先度ルールは合計処理時間が基準であり，NEH-D の優先度ルールは合計処理時間の平均 AVG に不偏標準偏差 STD を加えている．また，PR_{SKE} では，NEH-D の優先度ルールに SKE を追加することで，優先度ルールが同値になることを回避し，得られる解の質を向上させている．PR_{SKE} の優先度ルールは 3 つのモーメントを用いているが，それぞれのモーメントの比率がすべての問題例において最適かどうかは不明である．そこで本節では，PR_{SKE} で用いる 3 種類のモーメントの重みを変更することによる解の精度の影響を観測するために実験を行った．

モーメントの重みは AVG に対し STD，SKE の比率を 0, 1, 2, 3 倍に変化させた．これらの NEH は C 言語によってコーディングし，コンパイラは gcc(Ver8.1.0)を使用した．計算機は 3.5GHz Intel(R) Core(TM) i7-7567U CPU，RAM:16GB 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．このベンチマークには 12 種類のセットがあり，与えられた種類ごとに 10 個の問題例がある．表 1 は左から順に，ジョブ数n，機械数m，それぞれの解の平均精度を示している．なお，得られた解πの目的関数値の誤差率（RPD）は$RPD_p = \frac{HS_p - UB_p}{UB_p} \cdot 100[\%]$を用いる．HSp は問題例 p に対して得られた解 π の評価値を表し，UBp は Taillard[3]によって与えられたそれぞれの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，より良好な解の誤差率を太字で示した．実験の結果，AVERAGE において PR_{SKE} は他の優先度ルールに基づく NEH に比べて平均的に良好な結果を示した．しかし，$n = 20$，$m = 5$の問題例では，AVG+STD+SKE×3 の優先度ルールが最も良好な結果を示し，$n = 50$，$m = 5$の問題例では，AVG+SKE×2 の優先ルールが最も良い性能を示した．このことから比較に用いたルール中では問題例に対して有効な優先度ルールは存在しないということを示した．また，AVG+SKE×3 や AVG+STD×3+SKE×3 など，STD や SKE の重みが大きすぎると性能が悪くなる傾向がみられる．

表 1　優先度ルールの比較実験結果

Instance		AVG (NEH) [8]	AVG + SKE	AVG+ SKEx 2	AVG+ SKEx 3	AVG+ STD (NEH-D) [12]	AVG+ STD+ SKE (PR_{SKE}) [11]	AVG+ STD+ SKEx 2	AVG+ STD+ SKEx 3	AVG+ STDx 2	AVG+ STDx2 + SKE	AVG+ STDx2 + SKEx2	AVG+ STDx2 + SKEx3	AVG+ STDx 3	AVG+ STDx3 + SKE	AVG+ STDx3 + SKEx2	AVG+ STDx3 + SKEx3
n	m																
20	5	3.3	3.46	3.21	3.13	2.7	2.71	2.74	**2.67**	3.3	3.32	3.59	3.61	3.2	3.26	3.43	3.3
20	10	4.6	4.35	4.68	4.68	4.08	3.68	3.6	**3.59**	3.78	4.27	3.83	3.84	4.72	4.42	4.64	4.57
20	20	3.73	3.53	3.75	3.87	3.82	**2.91**	3.24	3.55	3.79	3.81	3.75	3.64	3.5	3.56	3.49	3.77
50	5	0.73	0.8	**0.53**	0.72	0.89	0.88	0.89	0.91	1.02	1.35	1.08	0.92	1.16	1.03	0.88	1.14
50	10	5.07	4.94	5.36	5.26	4.9	4.84	4.77	4.69	5.06	**4.46**	5	5.06	4.52	5.03	5.18	4.67
50	20	6.65	6.1	6.53	6.25	6.12	6.42	6.45	6.56	6.21	6.26	**5.78**	6.18	6.09	6.64	6.3	6.32
100	5	0.53	0.45	0.51	0.5	0.41	0.54	0.56	0.52	0.39	**0.37**	0.55	0.62	0.49	0.65	0.66	0.58
100	10	2.21	2.12	2.4	2.25	2.16	2.24	**1.97**	1.99	2.55	2.36	2.29	2.56	2.63	2.5	2.54	2.53
100	20	5.34	5.48	5.32	5.47	5.65	**4.99**	5.25	5.42	5.57	5.46	5.48	5.55	5.65	5.76	5.55	5.68
200	10	1.26	1.45	1.44	1.4	**1.24**	**1.24**	1.39	1.36	1.34	1.31	1.25	1.32	1.37	1.3	1.26	1.33
200	20	4.44	4.14	4.49	4.4	4.57	4.15	4.32	4.38	4.32	**4.1**	4.21	4.15	4.52	4.28	4.57	4.54
500	20	2.07	2.3	2.2	2.08	2.13	2.13	**1.91**	2.2	2.03	2.02	2.09	2.08	2.15	2.09	2.13	2.21
AVERAGE		3.33	3.26	3.37	3.34	3.22	**3.06**	3.09	3.15	3.28	3.26	3.24	3.3	3.33	3.38	3.39	3.39

５．E-NEH

上述した通り，単一の優先度ルールでは，様々な問題例に対して良い結果を得ることはできない．そこで，本論文で示す Extend NEH(E-NEH)は 1 つの問題例に対して複数の優先度ルールを適用し，適切なジョブ順序を用いるアルゴリズムである．E-NEH では，表 1 の中で良好な結果を示した優先度ルールを用いる．図 1 に E-NEH の概要を示す．E-NEH はまず AVG，AVG+STD，AVG+STD+SKE，AVG+SKE の順に①で囲った 4 種の優先度ルールを実行する．AVG または AVG+SKE の解が他の優先ルールよりも良い解を算出した場合，アルゴリズムを終了する．これは，AVG が①の 4 つの優先度ルールの中で最良の解を示した場合，STD と SKE を付与する必要がないと判断するためである．また，AVG+SKE が①の 4 つの優先度ルールの中で最適解を示す場合，STD のモーメントは不要と考え，優先度ルールの拡張を行わない．SKE の重みをさらに増加させる優先度ルール AVG+SKE×2 は，表 1 で良好な結果を示さなかったため考慮しない．AVG+STD または AVG+STD+SKE が①の 4 つの優先度ルールの中で最良の解を得た場合，新たな優先ルールを試す．例えば，AVG+STD がより良い解を示した場合，AVG に STD を追加することは有効であり，STD は必要であると考え，②の STD を追加した AVG+STD×2 の優先度ルールを試す．AVG+STD×2 が AVG+STD より良い解を得た場合，③の AVG+STD×2+SKE の優先ルールを試す．AVG+STD×2+SKE が AVG+STD×2 より良い解を得た場合，④の AVG+STD×2+SKE×2 を試み，アルゴリズムを終了する．優先度ルール AVG+STD+SKE が最初に試した①の 4 つの優先度ルールの中で最良の解を示した場合，AVG+STD+SKE×2 と AVG+STD×2+SKE を試す．AVG+STD×2+SKE が AVG+STD+SKE と AVG+STD+SKE×2 よりも良い解を得た場合，AVG+STD×2+SKE×2 を試し，アルゴリズムを終了する．以上説明した探索で見つかった最良の解を出力 する．

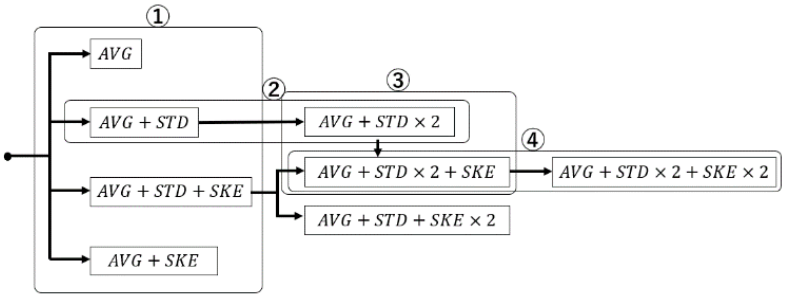

図 1　E-NEH の概要図

６．実験結果

提案する E-NEH の性能を評価するために，Taillard と VRF ベンチマーク問題例[2][3][4]を用いて比較実験を行った．VRF のベンチマークは Large と Small の 2 種類のセットがあり，従来法との性能差を顕著に示すため，より大規模な問題例である 24 種類の問題例のみを使用し，各種類には 10 例題が含まれる．比較解法は AVG を優先度ルールとする NEH，AVG+STD を優先度ルールとする NEH-D，AVG+STD+SKE を優先度ルールとする PR_{SKE}，ALL である．ALL は E-NEH で用いた 8 種類の優先度ルールを全て試し，その中から最良解を返す方法である．表 2 と表 3 は，左から順にジョブ数nと機械数m，各アルゴリズム（NEH，NEH-D，PR_{SKE}，E-NEH，ALL）の平均精度と計算時間[s]を示している．各アルゴリズムの解の精度を測定するために，性能指標として RPD を用いた．

表 2 に Taillard のベンチマーク問題例に対する結果を示す．NEH，NEH-D，PR_{SKE}，E-NEH，ALL の解の精度は，それぞれ AVERAGE において 3.33，3.22，3.06，2.58，2.47 であった．E-NEH は，AVERAGE において NEH，NEH-D，PR_{SKE} よりも良い結果を示した．特に，E-NEH は$n = 50$，$m = 20$のようなジョブ数に対して機械数が多い問題例において，他の NEH 基準の手法よりも良好な結果を示した．NEH，NEH-D，PR_{SKE}，E-NEH，ALL の計算時間は，それぞれ AVERAGE で 0.01，0.01，0.01，0.04，0.09 である．E-NEH の計算時間は AVERAGE において他の NEH 基準の手法の約 4 倍であるが，これは E-NEH が常に少なくとも 4 つの異なる優先度ルールを試すのに対し，他の NEH ベースの手法はそれぞれ 1 つの優先度ルールを試行するためであると考えられる．単純なアルゴリズムの ALL と比較して，E-NEH の平均解精度は約 0.11[%]悪いが，計算時間は約半分程度である．

表 2 Taillard ベンチマーク問題例に対する実験結果

Instance		NEH [8]		NEH-D [12]		PR_{SKE} [11]		E-NEH		ALL	
n	m	avg	time[s]	avg	time[s]	avg	time[s]	avg	time[s]	avg	time[s]
20	5	3.30	0.00	2.70	0.00	2.71	0.00	2.09	0.00	1.97	0.00
20	10	4.60	0.00	4.08	0.00	3.68	0.00	3.16	0.00	2.75	0.00
20	20	3.73	0.00	3.82	0.00	2.91	0.00	2.68	0.00	2.66	0.00
50	5	0.73	0.00	0.89	0.00	0.88	0.00	0.40	0.00	0.36	0.00
50	10	5.07	0.00	4.90	0.00	4.84	0.00	3.97	0.00	3.78	0.01
50	20	6.65	0.00	6.12	0.00	6.42	0.00	5.15	0.01	5.09	0.01
100	5	0.53	0.00	0.41	0.00	0.54	0.00	0.25	0.00	0.21	0.02
100	10	2.21	0.00	2.16	0.00	2.24	0.00	1.66	0.01	1.61	0.03
100	20	5.34	0.00	5.65	0.00	4.99	0.00	4.69	0.02	4.61	0.06
200	10	1.26	0.01	1.24	0.01	1.24	0.01	1.05	0.03	1.01	0.07
200	20	4.44	0.01	4.57	0.01	4.15	0.01	3.93	0.07	3.79	0.12
500	20	2.07	0.08	2.13	0.08	2.13	0.08	1.91	0.38	1.80	0.75
AVERAGE		3.33	0.01	3.22	0.01	3.06	0.01	2.58	0.04	2.47	0.09

表 3 に VRF のベンチマーク問題例に対する結果を示す．NEH，NEH-D，PR_{SKE}，E-NEH，ALL の解精度は，AVERAGE においてそれぞれ 3.33，3.24，3.21，2.96，2.89 であった．E-NEH は，AVERAGE において NEH，NEH-D，PR_{SKE} よりも良い結果を示した．NEH，NEH-D，PR_{SKE}，E-NEH，ALL の計算時間は，それぞれ AVERAGE で 0.18，0.21，0.19，0.88，1.16 である．E-NEH は他の NEH ベースの手法より約 4.5 倍時間がかかり，ALL に対しては約 0.75 倍短縮された．Taillard のベンチマークの場合と比較して，E-NEH の計算時間が増加した．しかし，VRF のベンチマークに対する解の精度は，他の NEH ベースの手法よりも優れている．以上より従来法にくらべ処理時間を若干要するものの全ての問題例において良好な結果を算出しその有効性を示した．

7．むすび

本研究では様々な問題例において共通して有効な優先度ルールはないという考えのもと，順列フローショップのための拡張 NEH(E-NEH)と呼ばれる NEH ベースの手法を示した．E-NEH は，AVG，STD，SKE のモーメントを段階的に組み合わせることで，適切な優先ルールを採用するアルゴリズムである．Taillard と VRF のベンチマークでの実験結果は，E-NEH が他の NEH ベースの手法よりも多少の時間はかかるが良好な解を算出し，E-NEH の有効性を示した．

表 3 VRF ベンチマーク問題例に対する実験結果

Instance		NEH [8]		NEH-D [12]		PR_{SKE} [11]		E-NEH		ALL	
n	m	avg	time[s]	avg	time[s]	avg	time[s]	avg	time[s]	avg	time[s]
100	20	5.71	0.01	5.61	0.01	5.44	0.01	4.84	0.02	4.82	0.04
100	40	5.47	0.01	5.31	0.01	5.25	0.01	4.78	0.04	4.60	0.06
100	60	4.95	0.01	4.51	0.01	4.70	0.02	4.28	0.06	4.23	0.09
200	20	4.23	0.02	4.02	0.01	4.17	0.02	3.72	0.08	3.60	0.11
200	40	4.71	0.03	4.66	0.04	4.56	0.04	4.27	0.13	4.19	0.22
200	60	4.60	0.06	4.35	0.07	4.45	0.08	4.03	0.22	3.99	0.37
300	20	3.00	0.04	3.03	0.04	3.00	0.05	2.71	0.13	2.45	0.22
300	40	4.08	0.07	3.90	0.07	3.97	0.07	3.65	0.28	3.56	0.44
300	60	3.93	0.12	3.91	0.11	3.85	0.11	3.55	0.50	3.52	0.73
400	20	2.56	0.06	2.43	0.05	2.44	0.05	2.17	0.24	2.04	0.33
400	40	3.66	0.13	3.51	0.11	3.57	0.11	3.28	0.50	3.12	0.72
400	60	3.56	0.20	3.47	0.19	3.36	0.17	3.17	0.93	3.14	1.16
500	20	2.27	0.08	2.23	0.11	2.00	0.08	1.86	0.36	1.77	0.50
500	40	3.20	0.18	3.11	0.21	3.00	0.18	2.90	0.99	2.86	1.10
500	60	3.12	0.27	3.20	0.34	3.10	0.28	2.82	1.54	2.79	1.79
600	20	1.57	0.12	1.64	0.13	1.61	0.11	1.50	0.58	1.47	0.72
600	40	3.13	0.23	2.98	0.36	2.97	0.24	2.78	1.42	2.69	1.57
600	60	2.93	0.40	2.94	0.44	2.91	0.38	2.73	1.68	2.68	2.54
700	20	1.40	0.16	1.23	0.16	1.30	0.16	1.13	0.74	1.11	0.96
700	40	2.77	0.34	2.60	0.46	2.65	0.34	2.49	1.55	2.43	2.13
700	60	2.75	0.54	2.71	0.59	2.69	0.53	2.55	2.65	2.53	3.51
800	20	1.23	0.20	1.15	0.26	1.13	0.19	1.08	0.90	1.07	1.25
800	40	2.46	0.45	2.50	0.49	2.38	0.54	2.32	2.16	2.24	2.76
800	60	2.71	0.68	2.67	0.73	2.57	0.73	2.52	3.46	2.47	4.43
AVERAGE		3.33	0.18	3.24	0.21	3.21	0.19	2.96	0.88	2.89	1.16

参考文献

1) D.S. Palmer, Sequencing jobs through a multi-stage process in the minimum total time-a quick method of obtaining a near optimum, Journal of the Operational Research Society, Vol.16, pp.101–107, 1965.

2) E. Taillard, Benchmarks for basic scheduling problems. European Journal of Operational Research, Vol.64, pp.278–285, 1993.

3) E. Taillard, Summary of best known lower and upper bounds of Taillard's instances, http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/flowshop.dir/best_lb_up.txt (accessed 4.18.22).

4) E. Vallada, and R. Ruiz, and J. M. Framinan, New hard benchmark for flowshop scheduling problems minimising makespan, European Journal of Operational Research, Vol.240, pp.666-677, 2015.

5) H. G. Campbell, R. A. Dudek, and M. L. Smith, A heuristic algorithm for the n job, m machine sequencing problem, Management science, Vol.16, No.10, pp.B-630, 1970.

6) J. N. D. Gupta, A functional heuristic algorithm for the flowshop scheduling problem, Journal of the Operational Research Society, Vol.22, pp.39-47, 1971.

7) A. H. G. Rinnooy Kan, Machine Scheduling Problems: Classification, Complexity and Computations, Martinus Nijhoff, The Hague, The Netherlands, 1976.

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

9) M. R. Garey. and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York, 1979.

10) S. M. Johnson, Optimal two-and three-stage production schedules with setup times included, Naval research logistics quarterly, Vol.1, No. 1, pp.61-68. 1954.

11) W. Liu, Y. Jin, and M. Price, A new improved NEH heuristic for permutation flowshop scheduling problems, International Journal of Production Economics, Vol.193, pp.21–30, 2017.

12) X. Dong, H. Huang, and P. Chen, An improved NEH-based heuristic for the permutation flowshop problem, Computers & Operations Research, Vol.35, No. 12, pp.3962-3968, 2008.

13) X. Li, Y. Wang, and C. Wu, Heuristic algorithms for large flowshop scheduling problems. In: Fifth World Congress on Intelligent Control and Automation, Vol.4, pp.2999–3003, 2004.

An Extended NEH based Method for Permutation Flowshop Scheduling Problem

Shun Ito, Tetsuya Oda[1)] and Kengo Katayama[1)*]

Graduate School of Engineering, Okayama University of Science

1) Department of Information and Computer Engineering, Faculty of Engineering, Okayama University of Science

1-1 Ridai-cho, Kita-ku, Okayama 700-0005, Japan

(Received October 31, 2023; accepted November 29, 2023)

The Permutation Flowshop Scheduling Problem (PFSP) is an important manufacturing scheduling problem where n jobs have to be processed on m machines, with each job following the same order at the machines. Since the problem to minimize makespan has been proven NP-hard when the number of m is larger than three, a number of heuristic methods have been developed for finding high quality solutions in a reasonable computation time. NEH heuristic is one of the most efficient and effective constructive heuristic methods for PFSP. NEH algorithm consists of two key steps 1) order jobs based on priority rule (the original one is the sum (average) processing times of each job); 2) insert jobs one by one according to the initial job order to construct a sequence (solution) of n jobs by testing all possible inserting positions for unscheduled jobs. Since the first step is crucial for the resulting solution quality, many different priority rules have been investigated. It is well known that the priority rule based on two moments of the average and standard deviation of processing times to order jobs is effective. Furthermore, a more effective priority rule including a third moment called skewness has been developed recently. In this paper, we present an extended NEH based method called Extended NEH (E-NEH) for PFSP in which the appropriate combination of moments is selected from the three moments step by step, depending on the search situation. Computational results on benchmark problem set showed that E-NEH obtained better results, or at least the competitive ones, than the original NEH and other NEH based methods including the rules based on the three moments although more computation times are required.

Keywords: flowshop scheduling problem； constructive heuristic; combinatorial optimization;