A-034

ジョブ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の近傍構造について
A Comparison of Neighborhood Structures of Iterated Local Search for the Job Shop Scheduling Problem

青山　優希[1)]　　片山　謙吾[1)]
Yuki Aoyama[1)]　Kengo Katayama[1)]

1　まえがき

ジョブショップスケジューリング問題 (Job Shop Scheduling Problem, JSSP)[1] は NP 困難として知られており，局所探索 (Local Search, LS) を導入したメタ戦略アルゴリズムが多く用いられる．LS で採用する近傍構造に応じて，LS の性能が左右されるため，近傍構造の研究が極めて重要視されている．JSSP における高品質な近傍構造として，スケジュール上で最も時間のかかるクリティカルパスに焦点をあてて近傍を生成する N5[2]，N6[1]，N7[3] に加え，2023 年には N8 近傍構造 [4] が提案されている．これら N 近傍を導入したタブー探索 (Tabu Search, TS) は，非常に良好な結果を算出することが報告されている．一方，他の代表的なメタ戦略として知られている反復局所探索法 (Iterated Local Search, ILS)[5] は、多くの組合せ最適化問題に対して良好な結果を示しているが，N8 近傍構造の発表以後，N8 近傍構造を導入した ILS の検証は十分でない．

本研究では，LS プロセスとして，N8 近傍構造を有する TS を導入し，探索の状況に応じて可変的な摂動処理を行う ILS を設計する．実験結果から，N7 近傍構造を用いた ILS との比較を通して，N8 きんぼうにもとづく ILS の有効性を示す．

2　ジョブショップスケジューリング問題

ジョブショップスケジューリング問題 (JSSP) は，m 台の機械 (マシン) $M_1,\ldots,M_m$，n 個の仕事 (ジョブ) $J_1,\ldots,J_n$，各ジョブにおいて処理する機械の順序 (技術的順序) と機械上での作業時間が与えられたとき，目的関数 f(本研究では総所要時間 (メイクスパン) とする) の最小化を目的としたスケジュールを作成する問題である [1]．ある仕事 J_j における機械 M_r 上での仕事を作業と呼び，O_{jr} と表す．各作業 O_{jr} は処理時間 P_{jr} を要し，中断されることなく処理される．ジョブは同時に 2 つ以上の機械で処理されることはなく，各機械も同時に 2 つ以上のジョブを処理することはない．

図 1 に，4 ジョブ 3 機械の場合の JSSP 解例を示す．ガントチャートの縦軸が機械番号，横軸は所要時間であり，図 1 の場合，メイクスパンは 25 となる．

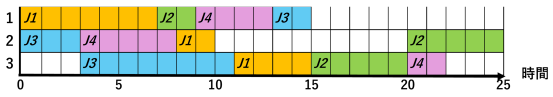

図 1　$n=4, m=3$ の場合の JSSP 解例ガントチャート

スケジュールの表現は一般にガントチャートが用いられるが，JSSP においては離接グラフ (Disjunctive Graph)[6] が用いられる．離接グラフは，あるジョブにおける機械の処理順序を指定する有向弧と，同一機械上で処理される作業の先行関係を決定するための離接弧によってスケジュールを表現する．

1)　岡山理科大学

3　局所探索と近傍構造

局所探索 (LS) は，ある解が与えられたとき，近傍構造に基づき，解に変化を加えることで生成される近傍解のうち，評価値が良い解に移動することで，より良い解を探索する．

JSSP に対する LS は，解の改善が見込まれる近傍に絞って効率的に探索するため，スケジュールのうち最も時間のかかる経路であるクリティカルパス上の，同じ機械による作業の連続であるクリティカルブロック (以後，単にブロックと呼ぶ) 内の作業に焦点をあてて近傍を生成する N 近傍構造 (N5，N6，N7) が提案されており，2023 年には N8 近傍構造 [4] が提案された．

N8 近傍構造を図 2 に示す．N8 近傍構造は N7 近傍構造を，N7 近傍構造は N6 近傍構造を内包する関係にある．N8 近傍構造は N7 までの近傍操作に加え，ブロックに含まれる作業をブロック外に移動する近傍操作も含む近傍構造である．また，N8 近傍は N7, N6 と比較して非常に大きな近傍であるため，評価値の改善が見込まれない近傍解を排除する近傍クリッピング [4] が導入された．

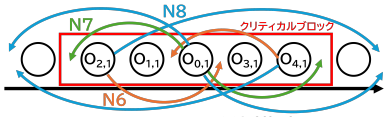

図 2　N8 近傍構造

4　JSSP に対する反復局所探索法

反復局所探索法 (ILS) は，一般的に，初期解生成，局所探索 (LS)，受入れ基準，摂動処理の 4 つの要素で構成される．LS は与えられた解に対して解の改善を試みる処理である．受入れ基準では，次反復における初期解を決定する．摂動処理は，与えられた解に突然変異 (Kick) を施すことで大局的に探索する．

4.1　反復局所探索法 (Iterated Local Search, ILS)

本研究における ILS の擬似コードを Algo.1 に示す．Line1 で，最初に初期解を生成し，現在解 x と $x_{ILSbest}$ を x_{init} で初期化する．Line7 で，x に対して LS のプロセスである TS を行う (Algo.2 参照)．Line8-10 で，解の評価を行う．$f(x_{TSbest})$ が $f(x_{ILSbest})$ よりも良ければ，$x_{ILSbest}$ を x_{TSbest} で更新する．Line11 で，x に摂動処理 (Kick) を行う (4.3 節参照)．終了条件を満たさなければ Line7 に戻り，再び LS プロセスを行う．

4.2　局所探索 (LS)

ILS における LS のプロセスで，タブー探索 (TS) を導入する．本研究における TS の擬似コードを Algo.2 に示す．Line4 で，現在解 x から生成可能な近傍を，近傍解集合 *NeighborSet* とする．Line5 で，*NeighborSet* のうち，評価値が良く，アスピレーション基準を満たし，タブーでない近傍解 x を選択する．もし条件を満たす解が存在しなければ，*NeighborSet* からランダムに解を選ぶ．Line6-10 で，解の評価を行う．$f(x)$ が $f(x_{TSbest})$ よりも良ければ，x_{TSbest} を x で更新する．Line11 で，

近傍移動に伴う情報をタブーリストに追加する．タブーリストのサイズは，Zhang らの方法 [3] に準じる．TS の終了条件を満たさなければ，Line4 からの一連の処理を繰り返す．

4.3 摂動処理

大局的に探索するための多様化処理として摂動処理 (Kick) を施す．本研究では，ILS の探索の状況に応じて摂動の強さを可変的に設定する．摂動は，現在解 x に対して TS と同じ近傍構造に基づき，多様化した近傍解集合を生成し，解集合から解をランダムに選択する．摂動処理で生成した近傍解の近傍移動に伴う情報をタブーリストに追加する．解が更新されない場合に，摂動の強さ ($KickSize$) を大きくする．$KickSize$ は 1 から始まり，Algo.1 の Line8 で解が更新されなかったとき，$KickSize$ をインクリメントする．解が更新されるか，$KickSize$ の最大値 ($MaxKickSize$) を超えたとき，$KickSize$ を 1 に戻す．$MaxKickSize$ は，機械数 m と任意に設定するパラメータ d により $KickSize = \lceil \frac{m}{d} \rceil$ で求める．

Algorithm 1 Iterated Local Search

1: x_{init} ←initial solution
2: $x_{ILSbest} \leftarrow x_{init}$ // Best solution found in ILS
3: $x_{TSbest} \leftarrow x_{init}$ // Best solution found in Tabu Search
4: $x \leftarrow x_{init}$ //Current solution
5: $T \leftarrow \emptyset$ // Initialize tabu list
6: **while** (stop criterion of ILS is not satisfied) **do**
7: $x, x_{TSbest}, T \leftarrow TabuSearch(x, T)$
8: **if** $f(x_{TSbest}) < f(x_{ILSbest})$ **then**
9: $x_{ILSbest} \leftarrow x_{TSbest}$
10: **end if**
11: $x, T \leftarrow Kick(x, T)$ // Perturbation
12: **end while**
13: **return** the best found solution $x_{ILSbest}$

Algorithm 2 Tabu search (x, T)

1: $x_{TSbest} \leftarrow x$ // Best solution in a Tabu Search
2: $noImpCount \leftarrow 0$ // Count no improve loop
3: **while** (stop criterion of TS is not satisfied) **do**
4: $NeighborSet \leftarrow$ generate neighborhood of x
5: $x \leftarrow \arg\min(f(x) : x \in NeighborSet \wedge (x$ is not tabu $\vee x$ satisfies aspiration criterion))
6: **if** $f(x) < f(x_{TSbest})$ **then**
7: $x_{TSbest} \leftarrow x, noImpCount \leftarrow 0$
8: **else**
9: $noImpCount \leftarrow noImpCount + 1$
10: **end if**
11: Update T
12: **end while**
13: **return** x, x_{TSbest}, T

5 実験結果

4 章で示した ILS の性能を評価するために実験を行った．評価する解法として，N8 近傍に基づく ILS(ILS-N8-固定 KickSize，ILS-N8-可変 KickSize)，N7 近傍に基づく ILS(ILS-N7-固定 KickSize，ILS-N7-可変 KickSize) の 4 解法を対象とする．JSSP のベンチマーク問題例として，良く知られているデータセット FT, ABZ, LA, TA から，計 5 題を選択した．各解は，法各ベンチマーク問題例について 10 回試行し，既知の最良解 (最適解) 算出時点，または反復回数の上限に達した時点の結果を出力し，解の精度を算出した．精度は得られた解 x^* と既知の最良解 x_{best} との誤差率 $\frac{f(x^*)-f(x_{best})}{f(x_{best})} * 100(\%)$ である．誤差率は，算出した解が既知の最良解に近いほど 0%に近づく．平均解精度が 0%となれば，全ての試行で既知の最良解が算出されたといえる．

計算機は，Ubuntu 24.04 を搭載する Intel(R) Xeon(R) E-2176G CPU @ 3.70GHz，RAM 32GB のマシンを用い，各解法のアルゴリズムは C++で実装し，g++コンパイラに-O3 オプションを付与してコンパイルした．

表 1 に N8 近傍構造に基づく ILS，表 2 に N7 近傍構造に基づく ILS-の結果をそれぞれ示す．表 1 と表 2 の各列は左から問題例名 Instance，各問題例で知られている最適解値 Opt，KickSize を 1 で固定した ILS の結果，4.3 節で述べた $d = 3$ として KickSize を可変的に設定した ILS の結果である．各解法における実験結果の各列は，左から，最良解精度 Best (%)，平均解精度 Avg (%)，最良解が求まった時点の処理時間 (sec) の平均である．

表 1 N8 近傍構造を有する ILS の実験結果

Instance	Opt	ILS-N8-固定 KickSize			ILS-N8-可変 KickSize		
		Best	Avg	Avg_Rtime	Best	Avg	Avg_Rtime
ft10	930	0.00	0.00	76.2	0.00	0.00	87.0
abz7	656	0.30	0.69	1,732.9	0.30	0.67	1,196.9
la24	935	0.00	0.00	220.6	0.00	0.00	284.2
la40	1222	0.16	0.16	26.8	0.00	0.13	206.0
ta01	1231	0.00	0.00	14.4	0.00	0.00	22.7
Average		0.09	0.17	414.2	0.06	0.16	359.4

表 2 N7 近傍構造を有する ILS の実験結果

Instance	Opt	ILS-N7-固定 KickSize			ILS-N7-可変 KickSize		
		Best	Avg	Avg_Rtime	Best	Avg	Avg_Rtime
ft10	930	0.00	0.00	258.0	0.00	0.00	236.6
abz7	656	0.91	1.07	1,464.1	0.91	1.11	1,228.1
la24	935	0.00	0.10	477.4	0.00	0.25	430.6
la40	1222	0.16	0.16	56.3	0.16	0.16	31.4
ta01	1231	0.00	0.00	35.0	0.00	0.00	18.3
Average		0.22	0.27	458.2	0.22	0.30	389.0

各表下部に，各列全問題例の平均値 Average を示している．表 1 の ILS-N8-固定 KickSize，表 2 の ILS-N7-固定 KickSize の下段に示す全問題例平均値 Average の結果から，N8 近傍構造を用いた ILS の方が優れた結果を，より少ない処理時間で算出していることから，N8 近傍構造を有する ILS の有効性が示された．また，表 1 の ILS-N8-固定 KickSize と ILS-N8-可変 KickSize の下段に示す Average の結果から，可変 KickSize を導入した ILS は，KickSize を固定した ILS の結果と比較して，同等かそれ以上に良好な解を算出可能であることを観測した．また，表 1 の可変 KickSize の結果は，表 2 の ILS-N7 の各結果と比較しても，より良い解を算出している．以上の結果から，N8 近傍構造および可変 KickSize を有する ILS の有効性を示した．

6 むすび

本論文では，LS プロセスに N8 近傍構造を有するタブー探索を導入し，探索の状況に応じて可変的な摂動処理を N8 近傍に基づき施す ILS を示した．実験結果から，N7 近傍構造を有する ILS との探索性能比較を通して，N8 近傍構造にもとづく ILS の有効性を示した．

参考文献

[1] E. Balas and A. Vazacopoulos. Guided local search with shifting bottleneck for job shop scheduling. *Management Science*, Vol. 44, No. 2, pp. 262–275, 1998.

[2] E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, Vol. 42, No. 6, pp. 797–813, 1996.

[3] C.Y. Zhang, P.G. Li, Z.L. Guan, and Y.Q. Rao. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research*, Vol. 34, No. 11, pp. 3229–3242, 2007.

[4] J. Xie, X. Li, L. Gao, and L. Gui. A new neighbourhood structure for job shop scheduling problems. *International Journal of Production Research*, Vol. 61, No. 7, pp. 2147–2161, 2023.

[5] H.R. Lourenço, O.C. Martin, and T. Stützle. Iterated local search. *Handbook of Metaheuristics*, pp. 320–353, Springer US, 2003.

[6] E.Balas. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, Vol. 17, No. 6, pp. 941–957, 1969.