遺伝的局所探索法による ジョブショップスケジューリング問題の解法

山田武士† 中野良平†

ジョブショップスケジューリング問題（JSSP）は$\mathcal{NP}$-困難な組合せ最適化問題の中でも特に難しい問題の1つとされている．本論文では，JSSPの近似解法として，局所近傍探索と遺伝的アルゴリズム（GA）との組合せによる多段階探索交叉GA（MSXF-GA）を提案する．MSXF-GAは，アクティブスケジュールのクリティカルブロック（CB）上の作業の移動に基づく近傍であるアクティブCB近傍に基づく近傍探索と，問題空間の距離と近傍構造に基づく交叉である多段階探索交叉（MSXF）を用いたGAから構成される．さらに，アクティブスケジュールに対し「スケジュールの反転」という概念を導入し，与えられた問題に適した方向を適応的に選択する機構を取り入れて，解法の高速化を図る．MSXF-GAを含むいくつかの解法をジョブショップスケジューリング問題（JSSP）のベンチマーク問題に適用したところ，高品質の解が高速かつ安定して求まるという，MSXF-GAの優れた性能が明らかになった．

Job-Shop Scheduling by Genetic Local Search

TAKESHI YAMADA† and RYOHEI NAKANO†

The job-shop scheduling problem (JSSP) is one of the most difficult $\mathcal{NP}$-hard combinatorial optimization problems. In this paper, an approximation method for JSSP called MSXF-GA is proposed using local neighborhood search and evolutionary computation, especially Genetic Algorithms (GA). MSXF-GA is composed by a neighborhood search that uses active CB neighborhood based on an active scheduler and operation permutations on the critical path, and Multi-Step Crossover Fusion (MSXF) that utilizes a neighborhood structure and a distance measure in the search space. The "reversal of the direction" of an active schedule is also introduced and MSXF-GA can adaptively select an appropriate direction for a given problem. Experimental results showed such a promising performance of MSXF-GA that it quickly found near optimal schedules in most cases.

1. はじめに

スケジューリングとは，ある目的を達成するため共通に使われる資源の時間配分を決定することである．ジョブショップスケジューリング問題とは，複数の異なる仕事（製品の組立てなど）を処理するために，共通資源である機械群の時間的な割当てを決定する問題ということができる．本論文で扱うジョブショップスケジューリング問題は次のように記述できる．n 個の仕事を m 台の機械で処理することを考える．各仕事を処理する機械の順序（技術的順序）は仕事ごとにあらかじめ与えられている．各機械上での仕事の処理のことを作業とよぶ．各作業は与えられた処理時間をかけて各機械上で中断なく処理される．ここで各機械はすべて異なり，同時に2つ以上の作業を処理することができないとする．すべての仕事を完成させるまでの時間を総作業時間（*makespan*）とよび，L で表す．このとき，$n \times m$（総作業時間最小）一般ジョブショップスケジューリング問題（以下単にJSSPとよぶ）とは，各仕事の技術的順序と，各作業の処理時間が与えられたもとで，L を最小にするような各機械上での仕事の処理順序をすべて決定することである．

JSSPは$\mathcal{NP}$-困難な組合せ最適化問題の中でも特に難しい問題の1つとされている．たとえば，1963年にMuth and Thompsonによって提示された 10×10 問題は20年以上未解決のままであった．JSSPは，その難解さ，そして産業上の応用分野の広さから，オペレーションズリサーチ（OR）の分野で30年近くにわたってさかんに研究されてきた．その主な解法は分枝限定（BAB: Branch and Bound）法によるものであるが，その他にいくつかの近似解法が提案されている．

† NTT コミュニケーション科学研究所
NTT Communication Science Laboratories

たとえば仕事の優先規則（priority rule）と，Giffler and Thompsonによるアクティブスケジュール生成手続き（GTアルゴリズム）[12]などに基づく方法は応用上よく用いられる[20]．Adamsらは *Shifting Bottleneck*（SB）とよばれる効果的な近似解法を提案した[3]．

JSSPの遺伝的アルゴリズム解法として今までに提案されている方法には，スケジュールをビット列にコード化しSimple GAを用いる方法[18]，仕事名の順列によってスケジュールを表現し順列上に作用する交叉を用いる方法[7),14]，アクティブスケジュール生成法に基く方法[11),28]などがある．しかしこれらの解法の多くは問題固有の性質をあまり用いていないため解法としては単純明解で汎用性が高いものの，小規模な問題のみに有効である．その他に，SB法と組み合わせた方法[11]なども知られているが，いずれも比較的大規模な問題に対しては十分効果的とはいえない．一方，局所探索法がJSSP解法として有効であることはいくつかの研究事例が報告されている．たとえばシミュレーテッドアニーリング（SA）による解法は文献15), 16), 31), 32)などに見られる．最近ではやはり局所探索法の1つであるタブー探索による効率的な解法もいくつか提案されており，成果を収めている[10),19]．さらにGA自体，局所探索法と組み合わせることによって性能が向上することはよく知られている[24),27]．GAと局所探索法の組合せによる効果的なJSSP解法としては文献9)があげられる．

本論文ではGAを確率的な局所近傍探索法と組み合わせることによる効果的なJSSP解法を提案する．JSSPに局所近傍探索を適用する際に用いる近傍としては，クリティカルパス上の仕事の移動に基づくものがいくつか提案されているが，ここでは文献32)で用いられ，その有効性が明らかになっているアクティブスケジュールに基づく近傍を用いる．そこに本論文で新たに提案する**スケジュールの作業処理順序の反転操作**を導入する．さらにGAと近傍探索との組合せである新たなオペレータ：**多段階探索交叉**（Multi-Step Crossover Fusion: MSXF）を提案する．比較的小規模なベンチマーク問題を題材にして，これら個々の導入効果を明らかにするとともに，多段階探索交叉を用いたGA（MSXF-GA）を，さらに難問とされるベンチマーク問題に適用してその優れた性能を明らかにする．

本論文の構成は次のようである．まず2章では局所近傍探索の基本的事項およびJSSPの近傍構造と距離について説明し，3章ではスケジュールの反転について説明する．4章では，進化型計算と局所探索を組み合わせたJSSP解法として，交叉を用いない集団近傍探索と，多段階探索交叉を用いた解法について説明する．5章では，Muth and Thompsonベンチマーク問題および，10難問とよばれる[4]より大規模なベンチマーク問題を用いた評価結果を示す．

手続き1 近傍探索アルゴリズム

Initialize 探索の初期値を選ぶ：$x = x_0 = x_{best}$.
do (1) $y \in N(x)$ を $V(y)$ の値に基づくあらかじめ与えられたある選択基準のもとに選び，$x = y$ とする．
(2) もし $V(x) < V(x_{best})$ ならば $x_{best} = x$ と置く．
until ［与えられたある終了条件を満たす］

2. 近傍探索と近傍構造

2.1 近傍探索

局所近傍探索（もしくは単に近傍探索，以下このようによぶ）において，解は探索空間上の点 x として表現される．今 x の近傍を $N(x)$ とし，目的関数 $V(x)$ を最小にする近傍探索の概要を**手続き1**に示す．ただし手続き中の終了条件としては，たとえば「繰返し回数がある定数 $I_{\max}$ に達するまで」などを用いる．

近傍探索は手続き1のステップ(1)で用いられる選択基準によって特徴づけられる．たとえば $V(y) < V(x)$ となるような $y \in N(x)$ をランダムに選ぶという最も単純な方法は「降下法」（descent method）とよばれる．上のような y のうち最小のものを選ぶ場合は「最良降下法」（best descent method）とよばれる．また次のような確率的な方法（Metropolis法）もよく知られている．すなわちランダムに選ばれた $y \in N(x)$ を，$V(y) < V(x)$ ならば確率1で選択し，そうでない場合は

$$P(y)=\exp\left(-\frac{\Delta V}{T}\right) \quad \text{ただし } \Delta V=V(y)-V(x) \tag{1}$$

で定義される確率 $P(y)$ で選択する．ここで $P(y)$ は受理確率とよばれる．シミュレーテッドアニーリング（SA）とよばれる方法では[1]，温度パラメータ T の値がアニーリングスケジュールに従って，繰返し回数の増加とともに減少する．その他，タブーリストとよばれる解のリストを用意し，近傍の要素でそのリストに登録されていないもののうち最も優秀な個体に遷移するタブー探索とよばれる方法もある[13]．このように近傍探索の考え方は非常に単純であり実装も容易であるが，一方で局所解に陥って抜け出せなくなってしまうか，さもなければ最適解に到達するためには非常に時間がかかるという欠点がある．

2.2 JSSP の近傍構造と距離

本論文では JSSP の近傍構造として文献 32) で提案されたアクティブ CB 近傍を用いる．アクティブ CB 近傍とは，クリティカルブロック上の作業の移動に基づく近傍で，しかも生成されたスケジュールがつねにアクティブスケジュールとなるよう，GT アルゴリズムに基づく特別な工夫をしたものである．JSSP に関する基本的な事項の説明および，GT アルゴリズム，アクティブ CB 近傍の構成法などは文献 32) もしくは，本論文の付録を参照されたい．

各機械上で処理される仕事の処理順序の違いに基づいて 2 つのスケジュール S,T 間の距離を定義することができる[18]．すなわちスケジュールを選択グラフ上で表現し（付録参照），それぞれ対応する選択グラフ上，S と T とで向きの異なる選択弧の個数を S,T 間の距離とする．このようにして定義された距離のことを DG 距離とよぶことにする．

3. スケジュールの反転

GT アルゴリズムをはじめとするアクティブスケジュール生成手続き，したがってアクティブ CB 近傍生成の手続きでは，いずれの場合も各作業を時間的順序が早いものから順に処理順序を確定する（詳しくは付録参照）．これは各作業の前方に余分な空き時間を作らない，というアクティブスケジュールの性質上当然のことである．しかしここでは以下のように考えることによって逆方向，すなわち時間的順序が遅いものから順に処理順序を確定することを考える．

一般に与えられたスケジュール問題の各仕事の技術的順序をまったく逆に反転した問題は元の問題と等価である．すなわち一方の問題に対する任意の実行可能解は，その処理順序を反転するだけで他方の問題の実行可能解となり，両者のクリティカルパス，および総作業時間はともに変わらない．しかし一方の問題に対するアクティブスケジュールは，反転後もアクティブスケジュールであるとは限らない．したがって「アクティブ」という性質はこの反転によって保存されない．

例として **図1** に示すような 2 機械，2 仕事の簡単な問題を考える．**図2** (1) はこの問題の 1 つの解であり，すでにアクティブスケジュールであるため，他の作業の処理開始時刻を遅らせることなくある作業を繰り上げる操作である left shift（付録参照）もしくは付録の手続き 6を用いてもこれ以上改善することはできない．ここで 図 2 (1) を反転させると 図 2 (2) が得られる．これはアクティブスケジュールではなく，図 2 (3) のように機械 2 上で仕事 1 を仕事 2 の前へ left shift することによって解を改善できる．図 2 (4) に図 2 (3) を再び反転した解を示す．この解は最適解である．結果的には 図 2 (1) の機械 2 上で仕事 1 を仕事 2 の後へ移動させることによって改善されたスケジュール 図 2 (4) を得たことになる．

仕事	技術的順序（処理時間）	
J_1	$M_2(3)$	$M_1(4)$
J_2	$M_2(2)$	$M_1(4)$

図1 簡単な 2 × 2 問題の例
Fig. 1 A simple 2 × 2 problem.

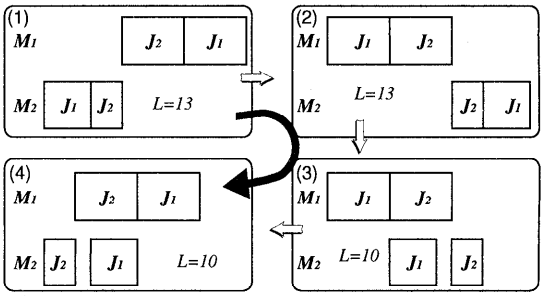


図2 スケジュールの反転およびアクティブ化
Fig. 2 Schedule reversal and activation.

アクティブスケジュールへの修正は多くの場合解の改善に結び付くため有効であるが，その修正で用いる left shift は対象となる作業の前方の空き時間しか考慮しないため，実は簡単な操作で解を改善できるのにそれを見逃す場合があることが以上の考察から分かる．後の章で実験的に示すように，解くべき問題によっては反転してから解いたほうが元の問題に比べて易しいということがあり得る．そこで，反転したアクティブスケジュールを同時に考えることが有効であると考えられる．しかし一般に近傍探索では探索中にたくさんの解を生成するため，新たな解を生成するたびに解を反転させアクティブスケジュールに修正するのは計算時間の点で効率が悪い．そこで一連の解生成過程において定期的に解を反転させるなどの工夫が必要である．

4. 遺伝的局所探索

遺伝的アルゴリズム（GA）は自然淘汰と遺伝の仕組みにヒントを得た最適化手法の 1 つである．GA の特徴は複数の解の集合である個体集団を用いた集団的探索戦略にあるということができる．GA を組合せ最適化問題に用いる場合，GA の局所探索能力の弱さを補うために他の局所探索法（LS），主に近傍探索法と組み合わせて用いることによってより効果的な解法となる場合が多い．このような組合せは「遺伝的局所探

索」(Genetic Local Search: GLS) とよばれる[24]. 本章では JSSP の近傍であるアクティブ CB 近傍を用いた近傍探索を GA に組み込んだ GLS 解法を提案する.

本章ではまず個体集団中の各個体が互いに独立に繰り返し局所探索する探索モデルである交叉なしの遺伝的局所探索：集団近傍探索を考える. 交叉を用いないため個体間の相互作用/情報交換は存在しないが, GA の定常状態モデル[22),26)]を用いることによって優秀な個体に対する淘汰圧がかかり, より短い総作業時間を持つスケジュールがより多く存在する領域に探索のバイアスがかかる.

なおここで用いる局所探索は, 特に 2.1 節で述べた確率的な近傍探索である. 確率的な局所探索の代表的なものに SA があるが, 遺伝的局所探索 1 回の試行中に何度も繰り返し SA を走らせるのは時間がかかりすぎ実用的ではない. そこで SA において温度パラメータを定数に制限する ($T = c$) 方法を用いる. このため手続き 1 の受理確率を次のように変更したものを用いる.

$$P_c(y) = \exp\left(-\frac{\Delta V}{c}\right) \quad (c \text{ は定数}) \qquad (2)$$

次に各個体間の相互作用/情報交換の手段として, 多段階探索交叉とよぶ交叉オペレータを導入する. 通常の交叉とは異なり, 多段階探索交叉はそれ自体一種の局所探索としてデザインされている. 実際 1 回の多段階探索交叉では, 一方の親個体を初期値とする近傍探索が行われ, 探索の結果得られた最も優秀な解が子個体として用いられる. ただしこの近傍探索において, 現在の解の近傍から新しい解が生成される過程でもう一方の親に関する情報が用いられる.

通常の GA 解法では解は遺伝子とよばれるある汎用のデータ構造で表現される. これは遺伝子コーディングとよばれる. たとえば JSSP の場合, 先に述べたようにスケジュールをビット列[18)]や仕事名の順列[7),14)]などにエンコードする方法がよく用いられる. これは, 一度汎用なデータ構造で表現してしまえば, その上に作用する汎用な交叉オペレータ, 突然変異オペレータを考えることができるためである. しかしこのような汎用的な方法では, 交叉の結果が実行可能解である保証がなくなるなどの問題もある. 一方多段階探索交叉はそれ自体近傍探索であるので, 近傍構造と距離の定義さえ与えれば, 個体のデータ構造とは独立に議論できる. 実際には我々が JSSP 解法に用いるアクティブ CB 近傍では, スケジュールを各機械上の仕事名の順列として表現するのが実装上は便利であるが, 他の問題への適用性も考慮してここでは近傍構造と距離の概念のみを用いて説明することにする.

4.1 交叉なしの遺伝的局所探索

手続き 2 に交叉なしの遺伝的局所探索による JSSP 解法の概要を示す. ここでは交叉, 突然変異を用いず, 代りに各個体は一定期間の近傍探索を行う. 以降この解法を次の交叉を用いる方法と区別するために, 集団近傍探索とよぶ.

淘汰, もしくは自然選択の基本モデルとしてここでは文献 22), 26) で提案されている GA の定常状態モデル (steady state model) を用い, また適応度に関しては, 総作業時間の短いスケジュールほど適応度は高いものとする. すなわち, つねに個体集団内の個体を総作業時間の短い順にソートし, 総作業時間の最も短い個体の順位が最も高くなるように順位付けをする. そしてこの順位の高さをその個体の適応度とする. 手続き 2 のステップ (1) において, 各個体の選ばれる確率がその順位の高さ, すなわち適応度に比例するようにランダムに 1 個体を選択する. ステップ (2) で確率的に, 3 章で説明した解の反転を行い, ステップ (3) で一定短期間の近傍探索を行った後, ステップ (4) で個体集団を更新する. ステップ (1), (4) を合わせたものが通常の GA の淘汰, もしくは自然選択に相当する. また個体集団のサイズが比較的小さい場合でも初期収束を回避できるよう, ステップ (4) において個体集団内にすでに同一の総作業時間を持つ個体が存在する場合はその個体を採用しないものとする.

ここで 3 章の解の反転を行う前の個体を l-個体, 反転後の個体を r-個体とよぶことにし, 手続き 2 において, 開始時の初期個体集団中には l-個体, r-個体を同数生成しておく. 処理が進むとともにステップ (4)

手続き 2 集団近傍探索による JSSP 解法

Initialize 同数のランダム l-個体, r-個体で構成される初期個体集団を生成し, 集団の各要素に対し局所探索を実行する. 各個体をその総作業時間の小さいもの順にソートし, 順位付けしておく.

do (1) 個体集団より, 総作業時間の短い優秀な個体ほど選ばれやすくなるよう, 順位の高さに比例する確率でランダムに個体 p を選択する.

(2) ある確率 P_r (たとえば $P_r = 0.1$) で p を反転させる.

(3) 個体 p に対し受理確率として式 (2) を用いた手続き 1 により一定期間の近傍探索を行い, その結果新たなスケジュール q を得る.

(4) もし q の総作業時間が個体集団中の総作業時間の最大値より小さく, 個体集団内に同一の総作業時間を持つ個体が存在しなければ, 最大値をとる個体を q で置き換え, 全体をソートしなおす.

until [与えられたある終了条件を満たす]

end 個体集団中最良の個体を出力して終了.

において優秀な個体が選抜されるため，個体集団には優秀な個体が多く存在するようになるが，同時に l-個体と r-個体の存在比も変化し，与えられた問題に適応してその問題に適した方向性を持つ個体が集団中に多く存在するようになる．

4.2 多段階探索交叉を用いた遺伝的局所探索

前節で用いた集団近傍探索では，各個体が行う近傍探索は互いに独立であり，各個体間での相互作用もしくは情報交換は存在しなかった．一方従来の遺伝的局所探索では，個体集団から確率的に選ばれた2個体に対し，交叉の適用によって新たな個体を生成する．さらにその個体を出発点とする局所探索を行い，最寄りの局所最適解を求め，これを子個体として次世代に用いる．したがって，交叉オペレータが各個体間での相互作用の役割を担っている．ただしここで交叉と局所探索とは互いに独立した操作であり，両者の間の関連性はあまりない．

Reevesはビット列を扱うSimple GAに基づく遺伝的局所探索として，局所探索と交叉を統一的に扱う近傍探索交叉：NSX（Neighborhood Search Crossover）を提案した[21]．そこではまず任意の2個体 x,z と個体 y に対して $d(x,z)=d(x,y)+d(y,z)$ となるとき，y を x,z の「中間（intermediate）」の個体と定義する．ただし x,y,z はビット列であり，$d(x,y)$ は x,y のハミング距離である．この時 $x \diamond y \diamond z$ と記す．次に二個体 x,z の「k 次2近傍」$N_k(x,z)$ を $N_k(x,z)=\{y \mid x \diamond y \diamond z$ かつ $(d(x,y)=k$ または $d(y,z)=k)\}$ と定義する．k 次2近傍は x,z の近傍の和集合の中から，両者の中間の個体に注目してそこに制限したものである．これらを用いて「k 次の近傍探索交叉：NSX_k」とは，両親 p_1,p_2 に対し $N_k(p_1,p_2)$ の要素すべてを吟味し，その中で最も優れた個体を選択して子個体とする操作，と定義される．すなわちこれは，近傍を親二個体の中間の領域に制限した上での2.1節の近傍探索（最良降下法）の解更新の操作そのものであり，近傍探索の拡張として交叉が位置付けられている．

本節ではこのようなNSXの考え方を発展させ，JSSPのような複雑な問題にも適用可能な交叉：多段階探索交叉（Multi Step Crossover Fusion: MSXF）を提案する．MSXFはNSXに比較して次のような特徴を持つ．ただしここで2つの親個体を p_1, p_2 とする．

- NSXが1回の解更新操作であったのに対し，MSXFは一方の親個体 p_1 を初期値とする一定短期間の近傍探索であり，探索の結果得られた最も優秀な解を子個体とする．
- MSXFはビット列に限らず，JSSPのようなより一般的な解表現と近傍構造を扱うことができる．
- MSXFで用いる近傍探索は前節で用いた確率的探索法に基づく．
- NSXにおいて近傍を両親の中間の領域に制限したのに対し，MSXFでは近傍中，他方の親個体 p_2 との距離が近い要素を確率的に優先して選択する．

多段階探索交叉の具体的な内容を**手続き3**に示す．これは手続き1の拡張である．ここで2個体 x,y 間の距離をハミング距離に限定せず一般に $d(x,y)$ で表す．

JSSPの場合には $d(x,y)$ はDG距離を表す．多段階探索交叉ではまず $x=p_1$ とし，$N(x)$ の各要素 y_i を p_2 までの距離 $d(y_i,p_2)$ の小さい順に並び替え，番号を付け替える．次にステップ(1)において，番号 i の値に反比例した確率で y_i をランダムに選ぶ．したがって番号 i の値が小さいほど選ばれる優先順位が高くなる．選ばれた個体はステップ(3)のように確率的に受理され，受理されると現在の解 x と入れ代わる．受理されなかった場合はその個体の番号を付けかえて（ステップ(4)），次回再びステップ(1)で選ばれる優先順位を下げる．以上のような優先順位の導入により，$N(x)$ 中に評価値がほぼ等しい複数の個体が存在する場合には p_2 に近いほうが選ばれやすくなる．したがって p_1 を初期値とする近傍探索は，特に優秀な解が見つかるまでは p_2 に近づく方向を優先し，近傍中に優秀な解が見つかるとそちらを優先して探索する（図**3**参照）．以上の操作を与えられたある終了条件を満たすまで繰り返す．ここで2.1節の手続き1と同様に，終了条件としてはたとえば，「繰り返し回数が

手続き3 多段階探索交叉（MSXF）

- p_1, p_2 を親である2つの解とする．

Initialize $x=p_1=q$ とおく．

do
- 各要素 $y_i \in N(x)$ に対して $d(y_i,p_2)$ を計算する．
- 各 $y_i \in N(x)$ を $d(y_i,p_2)$ の値の小さい順にソートし，番号を付け替える．

do (1) 番号 i の値に反比例する確率で $N(x)$ の要素 y_i をランダムに選ぶ．
(2) もし y_i が評価済みでなければ $V(y_i)$ を計算する．
(3) もし $V(y_i) \leq V(x)$ なら確率1で y_i を受理．そうでない場合は確率 $P_c(y_i)$ で受理．
(4) y_i の添字番号を i から n に変更し，y_k ($k \in \{i+1,i+2,\ldots,n\}$) の添字番号を k から $k-1$ に変更する．

until [y_i が受理]

- $x=y_i$ とおく．
- もし $V(x)<V(q)$ ならば $q=x$ とおく．

until [与えられたある終了条件を満たす]

- q を子個体として採用する．

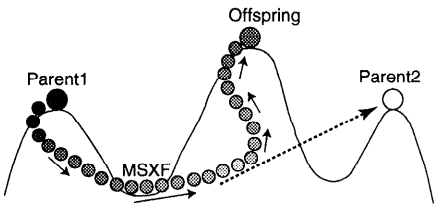


図3 MSXF による両親（Parent1, Parent2）から子個体（Offspring）の生成
Fig. 3 Offspring generated from Parents 1 and 2 by MSXF.

ある定数 $I_{\max}$ に達するまで」などを用いる．終了後，最終的に得られた最も優秀な解（ステップ（3）の q）が子個体として採用される．

手続き 3 において $N(x)$ をソートする際，$N(x)$ のすべての要素 y_i に対してあらかじめ $d(y_i, p_2)$ を計算しなければならない．一見これは非効率的であるが，たいていの場合，$d(y_i, p_2)$ は x から y_i への遷移操作および $d(x, p_2)$ を用いて簡単に計算でき，たとえばこの計算のためだけにわざわざ y_i を生成，評価する必要はない．

また $N(x)$ をソートする際，$d(y_i, p_2)$ の代わりに符号のみを重視した次式：

$$sgn(d(y_i, p_2) - d(x, p_2)) + r_\epsilon \qquad (3)$$

を用いることも考えられる．ここで $sgn(x)$ は x の符号を表す，すなわち $x > 0$ ならば $sgn(x) = 1$，また $x = 0$ ならば $sgn(x) = 0$，その他の場合は $sgn(x) = -1$ である．また r_ϵ は同一符号を持つ要素間にランダムな順位を付けるための微小な乱数である．後のベンチマーク実験ではこちらを用いる．

もし与えられた親個体 p_1, p_2 間の距離が小さく両者が互いに似かよっている場合，p_1 を初期値とし，p_2 に近づく方向を優先するという MSXF の枠組みはうまく働かない．したがってそのような場合は，手続き 3 とほぼ同じで，ステップ（3）においてソートする順序を逆にした点だけが異なる手続きを用いる．これを近傍探索変異（Multi Step Mutation Fusion: MSMF）とよぶ[29]．

MSXF（および MSMF）を用いた GA である MSXF-GA は，手続き 2 に変更を加えることによって**手続き 4** のように記述される．

手続き 4 MSXF-GA による JSSP 解法

Initialize 同数のランダムな l-個体，r-個体で構成される初期個体集団を生成し，集団の各要素に対し局所探索を実行する．各個体をその総作業時間の小さいもの順にソートし，順位付けしておく．

do （1） 個体集団より，総作業時間の短い優秀な個体ほど選ばれやすくなるよう，順位の高さに比例する確率でランダムに 2 個体 p_1, p_2 を選択する．

（2） ある確率 P_r（たとえば $P_r = 0.1$）で p_1 を反転させる．

（3） ある確率 P_c（たとえば $P_c = 0.5$）でステップ（3a）か，ステップ（3b）のどちらかを選び実行する．

（a） もし p_1, p_2 間の DG 距離があらかじめ定めたある値 d_m より小さい場合は，p_1 に対して MSMF を実行し q を生成する．
そうでない場合，p_1, p_2 にアクティブ CB 近傍，DG 距離を用いた MSXF（手続き 3）を適用し，新たなスケジュール q を生成する．

（b） 個体 p_1 に対し受理確率として式2を用いた手続き 1 により一定期間の近傍探索を行い，その結果新たなスケジュール q を得る．

（4） もし q の総作業時間が個体集団中の総作業時間の最大値より小さく，個体集団内に同一の総作業時間を持つ個体が存在しなければ，最大値をとる個体を q で置き換え，全体をソートしなおす．

until ［与えられたある終了条件を満たす］

end 個体集団中最良の個体を出力して終了．

5. 実験結果

前章で提案した解法の有効性を評価するために 2 種の計算機実験を行った．

5.1 MT ベンチマークを用いた比較評価

アクティブ CB 近傍，スケジュール反転，集団近傍探索，多段階探索交叉など，これら個々の要素の導入効果を検証するために，Muth and Thompson（MT）ベンチマーク[17]を用いた計算機実験を行った．**図 4**，**図 5** に，さまざまな実験条件のもとでの実験結果を，実験を打ち切るまでの CPU 時間の比較によって示す．ここで L, R, LR はそれぞれ l-個体のみ，r-個体のみ，両個体混合による集団近傍探索を，LRX は LR に多段階探索交叉を組み込んだ MSXF-GA による解法である．各々の解法は乱数の初期値を変えて 100 回ずつ実験を行った．ただし L, R, LR, LRX に関しては個体集団のサイズ 10 を用い，各実験は既知の最適解に至るか，最大経過時間 10 分を超えた時点で終了させた．1 回あたりの近傍探索および多段階探索交叉における繰返し回数の上限 $I_{\max} = 1000$ とし，また LR, LRX については $P_r = 0.1$，LRX については $P_c = 0.5$，$d_m = 0.25nm$（n：仕事数，m：機械数）を用いた．すべての実験は DEC Alpha 600 5/266 上で行いプログラムはすべて C 言語で書かれている．各図の y 軸は各実験を打ち切るまでの CPU 時間（秒）を，x 軸は実験の番号を表し，CPU 時間が短い順に並べ替えてある．CPU 時間が 600 である実験は制限時間内に最適解が求まらなかったことを意味する．図から，アクティブ CB 近傍を用いたこれらの実験では，mt 20×5 問題に対して l-個体のみを用いた場合を除き，ほとん

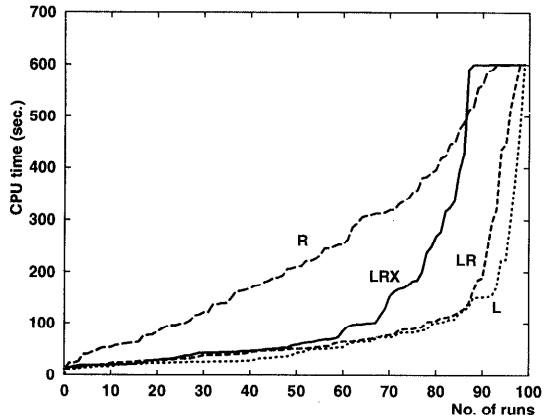


図4 CPU 時間の比較（mt 10 × 10 問題）
Fig. 4 CPU time comparisons using mt 10 × 10 problem.

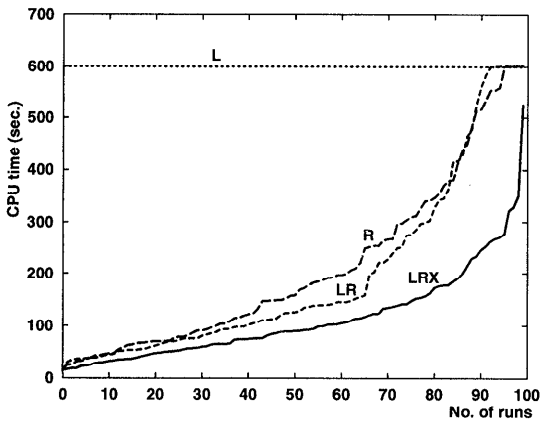


図5 CPU 時間の比較（mt 20 × 5 問題）
Fig. 5 CPU time comparisons using mt 20 × 5 problem.

どの場合10分という制限時間内に最適解が求まるが，最適解が求まるまでの時間には解法によって差があることが分かる．たとえば図4から，この問題に関しては解法L, LR < LRX < Rの順に短時間で最適解が求まる．

図4，図5の両図を総合的に見ると，まず解法Lと解法Rのどちらの解法が適しているかは解くべき問題に依存することが分かる．実際，mt 20 × 5 問題の場合，解法Lでは時間内に最適解を求めることはできないのに対し，解法Rではほぼ毎回最適解が求まる．一方mt 10 × 10 問題の場合，逆に解法Lのほうが解法Rより短時間で最適解を求めることができ，成績が良い．また l-個体，r-個体を併用する解法LRはどちらの問題に対しても効率的な解法となっている．解法LRにおいて最終的に得られた個体集団を調べると，mt 10 × 10 問題の場合はほとんどすべての個体が l-個体に，またmt 20 × 5 問題の場合は r-個体になっており，LRを用いることによって適応的に問題に適した方向性が選択されていることが確認できる．

次に両問題ともに適した解法となっているLRと，LRにMSXFを組み込んだ解法であるLRXを比較すると，mt 10 × 10 問題の場合，解法LRXのほうがわずかではあるが解法LRに比べて結果が悪化している．一方，mt 20 × 5 問題の場合は逆に解法LRXのほうが解法LRよりかなり有利である．この問題に対しては解法LRXがランダムな初期値によらず高速で安定した解法となっていることが分かる．

なお，参考のためアクティブCB近傍の代りにセミアクティブなCB近傍（付録参照）を用いた場合の集団近傍探索による実験の結果を**表1**に示す．ここでは他の実験より有利になるよう，個体集団のサイズ30を用い，各実験は既知の最適解に至るか，最大経過時間40分を超えた時点で終了させるものとした．表中SEM欄はセミアクティブなCB近傍による集団近傍探索の結果を，またアクティブCB近傍による解法を代表してLRX欄に図4，図5のLRXによる結果を再掲する．表中bst, wst, avgはそれぞれ100回の実験中求まったスケジュールの総作業時間の最良値，最悪値，平均値を表す．SEMに関しては長時間かければなんとか最適解が求まる場合があるものの，個体数，探索時間ともにハンディを与えて他より有利に設定しているにもかかわらず，LRXの結果に比べかなり悪いことが分かる．

表2にMTベンチマークに関する我々の結果を含めた過去のGA解法の結果をまとめておく．この表は文献7)掲載のものに，さらにいくつかの結果を加えたものである．表中MTベンチマークの2つの問題に対しともに最適解が求まっているのは我々の解法および，やはりGAと局所探索と組合せによる解法であるMattfeldらによる解法のみである．表から分かるように，既存のGA解法の多くはMTベンチマークとして知られるこの2つの問題を両方ともうまく扱うことは依然として難しい．Mattfeldらの解法との性

表1 アクティブCB近傍とセミアクティブCB近傍における性能比較
Table 1 Performance comparisons between semi-active and active CB neighborhood.

Alg.	bst	wst	avg
mt 10 × 10 problem			
SEM	930	955	943.14
LRX	930	938	930.89
mt 20 × 5 problem			
SEM	1165	1218	1181.64
LRX	1165	1165	1165

表2 MT ベンチマークによる各種 GA 解法の比較
Table 2 Comparisons between various GA methods on the MT problems.

			10 × 10	20 × 5
1963	Muth-Thompson	Test problems		
1991	Nakano/Yamada	Conventional GA [18]	965	1215
1992	Yamada/Nakano	Giffler-Thompson GT-GA [28]	930	1184
	Dorndorf/Pesch	Priority-Rule based P-GA [11]	960	1249
	Dorndorf/Pesch	Shifting-Bottleneck SB-GA [11]	938	1178
1994	Mattfeld/Kopfer/Bierwirth	Diffusion GA [9]	930	1165
1995	Kobayashi/Ono/Yamamura	Subsequence Exchange Crossover [14] SXX-GA	930	1178
1995	Bierwirth	Generalized-Permutation GP-GA [7]	936	1181
——	Our results	MSXF-GA	930	1165

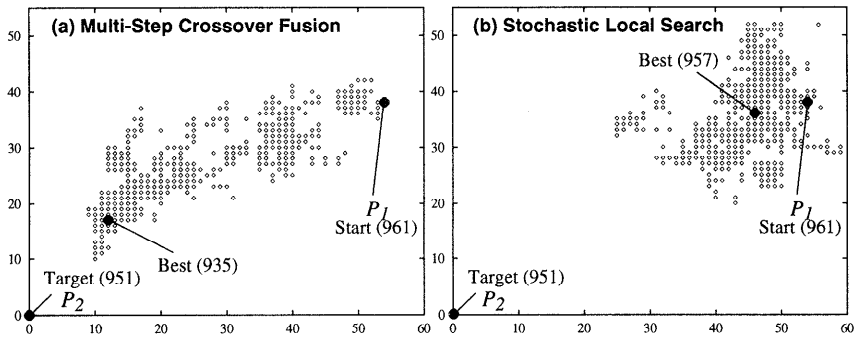


図6 それぞれ (a) 1 回の MSXF, (b) 短期間の局所探索, で得られた解の分布
Fig. 6 Distribution of solutions generated by an application of (a) MSXF and (b) a short-term stochastic local search.

能比較に関しては, 5.2 節で再び述べる.

図 6 (a) に 1 回の MSXF で生成された解の分布を, 図 6 (b) に (a) と同等の計算量による確率的な局所探索によって生成された解の分布を示す. (a), (b) ともに同一の初期値 (すなわち p_1) を用いている. 図の x 軸, y 軸はそれぞれ, 奇数番号, 偶数番号の機械上での p_2 との向きの異なる選択弧の個数, すなわち限定された DG 距離を表す. (b) の探索点がより狭い範囲にほぼ一様に分布しており, p_1 の周辺のきめ細かい探索であるのに対し, (a) においては p_2 の方向に探索のバイアスがかかる結果, p_1 と p_2 の間のより広い領域の, 多少きめの粗い探索になっていることが分かる. MSXF-GA では粒度の異なるこの 2 つの探索方法の併用によってより安定した解法になっていると考えられる.

5.2 10 難問ベンチマークの結果

前節の MT ベンチマークを用いた実験により, スケジュールの反転を組み込んだ解法 LR を用いることによって, どの問題にも適した効率的な集団近傍探索が構成できることが実験的に示された. さらに LR に多段階探索交叉を組み込んだ MSXF-GA を用いた実験を行ったが, MSXF の効果については, mt 20 × 5 問題では効果があるが, mt 10 × 10 問題では効果がないという結果となった. これは mt 10 × 10 問題は規模が小さいため, MSXF の導入による効果よりも解法 LR の局所探索の効果のほうが大きいためと考えられる. これを検証し, さらにより規模の大きな問題に対しても本論文で提案する方法が有効であることを示すために, より規模が大きく難しいとされている 10 個の問題[4]に対し MSXF-GA を適用する実験を行った.

表 3 に, 解法 LRX をそれぞれの問題に対しそれぞれ乱数の初期値を変え 30 回行った実験の結果を示す. ただし, 個体集団のサイズ 20 を用い, 各実験は既知の最適解に至るか, 最大経過時間 40 分を超えた時点で終了させる. その他の条件は MT ベンチマークのときと同様である. 表 3 の各列はそれぞれ問題名 (prob), 現在知られている最適解の下限および上限 (lb, ub), 得られた最良値 (bst), 平均値 (avg), 分散 (var), 最悪値 (wst) を表す. また * で表されるのは最適解であり, 最適解が得られた実験に関しては最適解が得られた回数および, 最適解発見までの平均時間 (秒) をそれぞれ n_{opt}, t_{opt} 欄に記す. なおこれら問題データ, 最適解の下限および上限などの情報は OR-library[6] より入手できる. 表 3 より実験の結

表3 JSSPの10難問を用いた実験結果
Table 3 Results of 10 tough JSSPs.

prob	(lb,ub)	bst	avg	var	wst	n_{opt}	t_{opt}
abz7	(656,656)	678	692.5	0.94	703	–	–
abz8	(645,669)	686	703.1	1.54	724	–	–
abz9	(661,679)	697	719.6	1.53	732	–	–
la21	–	*1046	1049.9	0.57	1055	9	687.7
la24	–	*935	938.8	0.34	941	4	864.1
la25	–	*977	979.6	0.40	984	9	765.6
la27	–	*1235	1253.6	1.56	1269	1	2364.75
la29	(1142,1153)	1166	1181.9	1.31	1195	–	–
la38	–	*1196	1198.4	0.71	1208	21	1051.3
la40	*1222	1224	1227.9	0.43	1233	–	–

表4 10難問による各種近似解法の比較
Table 4 Comparison with various heuristic methods on the 10 tough problems.

prob	our bst	Nowi	Dell	YN96	Aarts	Matt	Appl
abz7	678	–	667	665	668	672	668
abz8	686	–	678	675	670	683	687
abz9	697	–	692	686	691	703	707
la21	*1046	1047	1048	*1046	1053	1053	1053
la24	*935	939	941	*935	*935	938	*935
la25	*977	*977	979	*977	983	*977	*977
la27	*1235	1236	1242	*1235	1249	1236	1269
la29	1166	1160	1182	1154	1185	1184	1195
la38	*1196	*1196	1203	1198	1208	1201	1209
la40	1224	1229	1233	1228	1225	1228	*1222

果得られた問題ごとの最良値は，最適解か現在知られる最も優れた解に非常に近いことが分かる．また実験間のばらつきを表す分散も小さく，得られる解の品質は安定している．特に10問のうち5問について最適解が得られており，la27を除く4問については最適解の得られる頻度も多く，最適解にいたる時間も十分短い．

表4はこれらの問題に対する既存の近似解法との比較を表す．表中Nowi, Dellはそれぞれ文献19), 10)のタブー探索による解法，YN96, Aartsはそれぞれ文献30), 32), 2)のSAによる解法，Mattは文献9)のGAによる解法，Applは文献4)による解法をそれぞれ表す．ここで表2と重複する文献はMattのみであるが，これは他のGAの文献がこれらの問題を扱っていないためである．表より，MSXF-GAによる解法は全体的に他の近似解法に比べてほぼ互格かそれ以上の結果を示していることが分かる．また特にMSXF-GAと同様な近傍構造を用い，問題専用の近似解法であるSB法と組み合わせたSA解法であるYN96のSA+SB法と比べると，これら2つの解法が互いに相補的な関係にあることは興味深い．すなわち，abz7-9およびla29の各問題に関してはSA+SB法が優れているが，SA+SB法で最適解が求まらなかったla38問題に対しMSXF-GAでは頻繁に最適解が求まっており，またSA+SB法の結果が文献10)や文献4)の結果を下回っているla40問題に対し，MSXF-GAは比較的良好な結果を示している．なお比較のためここで取りあげた文献は一部の代表的なものだけである．さらに広範囲な文献調査に基づくより詳細な実験結果については文献25)を参照されたい．

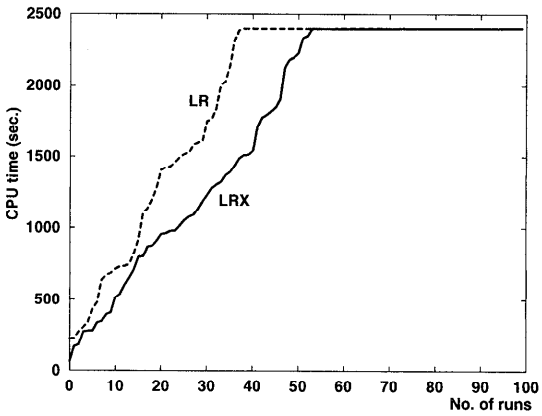


図7 CPU時間の比較（la38 15 × 15 問題）
Fig. 7 CPU time comparisons using la38 15 × 15 problem.

表3の中でも最適解の得られる回数が多かったla38問題に対し，前節の図4，図5同様にLRとLRXを比較するために初期値を変え100回ずつ行った実験のCPU時間の比較を**図7**に示す．前と同様，図のy軸は各実験を打ち切るまでのCPU時間（秒）を，x軸は実験の番号を表し，CPU時間が短い順に並べ替えてある．なお，CPU時間が2400である実験は制限時間内に最適解が求まらなかったことを意味する．図からMSXFを用いたLRXのほうがLRよりも最適解の得られる頻度，CPU時間（最も速い場合約63秒で最適解が求まった）において優れていることが分かる．図5の結果とあわせて，局所探索のみでは手に負えない複雑な問題に対しては多段階探索交叉が特に有効であることが実験的に明らかになったといえる．

6. 結　論

本論文では，まず，ジョブショップスケジューリング問題に対する近似解法として，アクティブスケジュールのクリティカルブロック上の作業の移動に基づく近傍であるアクティブCB近傍を用いた集団近傍探索を提案した．またスケジューリング問題の特質に着目して，解の反転という概念を導入し，問題によってはこの反転によって高品質の解が高速かつ安定して求まることを示した．さらにGAの淘汰機構を利用し，与えられた問題に適した解方向を適応的に選択し，問題に

依存しない優れた解法が構成できることをMuth and Thompsonベンチマークを用いた実験によって検証した.

次に,探索空間の近傍構造と距離を利用した交叉である多段階探索交叉(MSXF)と,MSXFを上記集団近傍探索に組み入れたMSXF-GAを提案した.1回のMSXFは,両親個体の一方を初期値として用い,もう一方の親を探索方向の道案内役に用いる短期の局所探索から構成される.MSXFでは両親の形質をそれぞれ継承した優れた解を求めて,問題空間内の特に両親個体の間の部分空間に重点を置いた探索を行う.MSXF-GAを10難問とよばれるより規模の大きなベンチマーク問題に適用した結果,最適解を含む非常に質の高い解が短時間に得られ,MSXF-GAが比較的規模の大きな問題に対しより高速で安定した解法であることが分かった.

参考文献

1) Aarts, E. and Korst, J.: *Simulated Annealing and Boltzmann Machines*, Wiley, Chichester (1989).
2) Aarts, E., van Laarhoven, P., Lenstra, J. and Ulder, N.: A Computational Study of Local Search Algorithms for Job Shop Scheduling, *ORSA J. on Comput.*, Vol.6, No.2, pp.118–125 (1994).
3) Adams, J., Balas, E. and Zawack, D.: The Shifting Bottleneck Procedure for Job Shop Scheduling, *Mgmt. Sci.*, Vol.34, No.3, pp.391–401 (1988).
4) Applegate, D. and Cook, W.: A Computational Study of the Job-shop Scheduling Problem, *ORSA J. on Comput.*, Vol.3, No.2, pp.149–156 (1991).
5) Balas, E.: Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm, *Oper. Res.*, Vol.17, pp.941–957 (1969).
6) Beasley, J.: OR-Library: Distributing Test Problems by Electronic Mail, *E. J. of Oper. Res.*, Vol.41, pp.1069–1072 (1990).
7) Bierwirth, C.: A Generalized Permutation Approach to Job Shop Scheduling with Genetic Algorithms, *OR Spektrum*, Vol.17, pp.87–92 (1995).
8) Brucker, P., Jurisch, B. and Sievers, B.: A Branch & Bound Algorithm for the Job-shop Scheduling Problem, *Discrete Applied Mathematics*, Vol.49, pp.107–127 (1994).
9) Mattfeld, D.C., Kopfer, H. and Bierwirth, C.: Control of Parallel Population Dynamics by Social-like Behavior of Ga-individuals, *3rd PPSN* (1994).
10) Dell'Amico, M. and Trubian, M.: Applying Tabu Search to the Job-shop Scheduling Problem, *Annals of Operations Research*, Vol.41, pp.231–252 (1993).
11) Dorndorf, U. and Pesch, E.: Evolution Based Learning in a Job Shop Scheduling Environment, *Computers Ops Res*, Vol.22, pp.25–40 (1995).
12) Giffler, B. and Thompson, G.: Algorithms for Solving Production Scheduling Problems, *Oper. Res.*, Vol.8, pp.487–503 (1960).
13) Glover, F.: Tabu Search – Part 1, *ORSA J. on Comput.*, Vol.1, No.3, pp.190–206 (1989).
14) Kobayashi, S., Ono, I. and Yamamura, M.: An Efficient Genetic Algorithm for Job Shop Scheduling Problems, *6th ICGA*, pp.506–511 (1995).
15) van Laarhoven, P., Aarts, E. and Lenstra, J.: Job Shop Scheduling by Simulated Annealing, *Oper. Res.*, Vol.40, No.1, pp.113–125 (1992).
16) Matsuo, H., Suh, C. and Sullivan, R.: A Controlled Search Simulated Annealing Method for the General Jobshop Scheduling Problem, Department of Management, The University of Texas at Austin, Vol. Working Paper, 03-04-88 (1988).
17) Muth, J. and Thompson, G.: *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ (1963).
18) Nakano, R. and Yamada, T.: Conventional Genetic Algorithm for Job Shop Problems, *4th ICGA*, pp.474–479 (1991).
19) Nowicki, E. and Smutnicki, C.: A Fast Taboo Search Algorithm for the Job Shop Problem, Institute of Engineering Cybernetics, Technical University of Wroclaw, Wroclaw, Poland., Vol. Preprinty nr 8/93 (1993).
20) Panwalkar, S.S. and Iskander, W.: A Survey of Scheduling Rules, *Oper. Res.*, Vol.25, No.1, pp.45–61 (1977).
21) Reeves, C.R.: Genetic Algorithms and Neighbourhood Search, *Evolutionary Computing, AISB Workshop (Leeds, U.K.)*, pp.115–130 (1994).
22) Syswerda, G.: Uniform Crossover in Genetic Algorithms, *3rd ICGA*, pp.2–9 (1989).
23) Taillard, E.: Parallel Taboo Search Techniques for the Job-shop Scheduling Problem, *ORSA J. on Comput.*, Vol.6, No.2, pp.108–117 (1994).
24) Ulder, N., Pesch, E., van Laarhoven, P., Bandelt, H.J. and Aarts, E.: Genetic Local Search Algorithm for the Traveling Salesman Problem, *1st PPSN*, pp.109–116 (1994).

25) Vaessens, R., Aarts, E. and Lenstra, J.: Job Shop Scheduling by Local Search, Technical Report, Eindhoven University of Technology, Dept. of Math. and CS (1994).
26) Whitley, D.: The Genitor Algorithm and Selection Pressure: Why Rank-based Allocation of Reproductive Trials is Best, *3rd ICGA*, pp.116–121 (1989).
27) Yagiura, M. and Ibaraki, T.: *Genetic and Local Search Algorithms as Robust and Simple Optimization Tools*, Kluwer Academic Publishers, MA, USA (1996).
28) Yamada, T. and Nakano, R.: A Genetic Algorithm Applicable to Large-scale Job-shop Problems, *2nd PPSN*, pp.281–290 (1992).
29) Yamada, T. and Nakano, R.: A Genetic Algorithm with Multi-step Crossover for Job-shop Scheduling Problems, *GALESIA '95*, pp.146–151 (1995).
30) Yamada, T. and Nakano, R.: *Job-shop Scheduling by Simulated Annealing Combined with Deterministic Local Search*, Kluwer Academic Publishers, MA, USA (1996).
31) 山田武士, Rosen, B.E., 中野良平：クリティカルブロックSA法によるジョブショップスケジューリング問題の解法, 電気学会論文誌, Vol.114-C, No.4, pp.476–482 (1994).
32) 山田武士, 中野良平：確率的探索と確定的探索の組合せによるジョブショップスケジューリング問題の解法, 情報処理学会論文誌, Vol.37, No.4, pp.597–604 (1996).

付録：クリティカルブロックに基づくJSSPの近傍構造

クリティカルブロック

JSSPの解の表現方法として選択グラフ（*disjunctive graph*）が広く用いられている．JSSPは選択グラフ $G=(V,C\cup D)$ を用いて次のように表現される．

- V は節点の集合であり，作業に対応する節点および2つの特殊な節点：ソース (0) とシンク $\star$ からなる．
- C は節点間を結ぶ有向弧（*conjunctive arc*）の集合で，技術的順序を表す．
- D は選択弧（*disjunctive graph*）の集合で，同一機械上の作業の対を表す．

各作業の処理時間は対応する節点に付与された重みによって表す．**図8** に 3×3 問題を用いた選択グラフの例を示す．図において円は節点，すなわち作業を表す．また実線は有向弧，破線は選択弧を表す．

完全なスケジュールは同一機械上の作業の処理順序をすべて決定することによって得られる．選択グラフモデルにおいて，このことは D のすべての無向（選択）弧を有向弧に変えることに対応する．このとき D より得られた有向弧の集合を「選択」（*selection*）とよぶ．ある選択 S が実行可能スケジュールを表現していることと，有向グラフ $G(S)=(V,C\cup S)$ が閉路（cycle）を持たないことは同値である．このとき S は「完全選択」（*complete selection*）とよばれる．

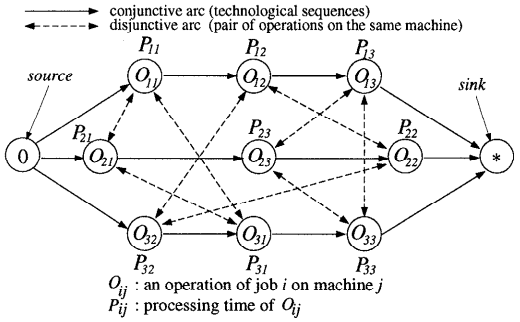


図8 3×3 問題を用いた選択グラフ G の例
Fig. 8 The disjunctive graph G of a 3×3 problem.

1つの完全選択から対応するスケジュールを生成する際，各作業を可能な限り早く加工して一意に得られるスケジュールをセミアクティブスケジュールとよぶ．完全選択と対応するセミアクティブスケジュールは同一視できるため，区別せず同一の記号 S を用いて表すことにする．S の総作業時間 $L(S)$ は (0) から $\star$ に至る最も長い重みつきパスの長さによって与えられる．このパスはクリティカルパスとよばれ $\mathcal{P}$ で表す．$\mathcal{P}$ 上の作業はクリティカルな作業とよばれる．同一機械上で連続して処理されるクリティカルな作業全体はクリティカルブロックとよばれる．

セミアクティブ近傍

近傍探索において，1つの解 S から1回の遷移，すなわち解への小さな摂動によって到達可能なすべての解の集合を近傍 $N(S)$ とよぶ．JSSPにおいて，たとえばクリティカルブロック上の隣接する作業の処理順序を入れ換える遷移と，この遷移を用いた近傍が文献15), 23)において用いられている．この考え方はもともとBalasによってBAB法の分枝操作として導入されたものである[5]．この近傍をここではAS（*adjacent swapping*）近傍とよぶ．

BAB法における分枝操作として用いられるもう1つの遷移操作が8), 10)において用いられている．そこではクリティカルブロック内の作業をそのブロックの一番先頭，もしくは最後へ移動させる遷移を考える．この遷移を用いた近傍をクリティカルブロック近

傍（CB近傍）とよぶ．一般にスケジュール S においてクリティカルブロックの作業をそのブロックの先頭，もしくは最後に移動させることによって新たに得られるスケジュールをそれぞれ前候補（*before candidate*），後候補（*after candidate*）とよぶ．ただしこれらは必ずしも実行可能とは限らない．したがって，S のCB近傍 $N^C(S)$ とは，S の前候補，後候補全体の集合から実行可能でないものを除いたものである．

アクティブCB近傍

セミアクティブスケジュールにおいて，ある作業の処理順序を，他の作業の処理開始時刻を遅らせることなく，繰り上げて早められる場合がある（これをleft shiftとよぶ）．どの作業もこれ以上left shiftできないスケジュールをアクティブスケジュールとよぶ．セミアクティブスケジュールのアクティブ化はしばしば解全体の改善につながる．最適スケジュールは明らかにアクティブスケジュールであるので，探索空間を初めからアクティブスケジュールに制限して最適化することで解法の効率化が期待できる．Giffler and Thompsonによるアクティブスケジュールを生成する手続き（GTアルゴリズム）[12]の概要を**手続き 5**に示す．なおここで，作業 O の最早開始時刻，最早完了時刻（$ES(O), EC(O)$）とは，その作業を最優先で処理した場合の処理開始時刻，処理完了時刻のことである．手続き5をすべての作業が処理されるまで繰り返すことによってアクティブスケジュールが1つ得られる．ステップ（3）で作業を選ぶ際，すべての可能性を考えることによって全アクティブスケジュールが生成されるが，その数は膨大である．与えられたセミアクティブスケジュール S を定義に従ってアクティブスケジュールに修正するためには，可能な限りleft shiftを繰り返すことになるが，より効率的な方法が手続き5を一部変更することによって得られる．これを**手続き 6**に示す．

前節で説明したAS近傍，CB近傍の要素は必ずしもアクティブスケジュールではない．そこで文献32)では，CB近傍の考え方を拡張し，その構成要素はすべて実行可能，しかもアクティブであり，元になった前候補，後候補に近い，という性質を持つ近傍を提案した．今 S をアクティブスケジュールとし，$B_{k,h,M}$ を S の機械 M 上のクリティカルブロックで，先頭および最後の作業が M 上それぞれ処理順序が k 番目，h 番目の作業であるものとする．また $O_{p,M}$ を $B_{k,h,M}$ に属する作業で，M 上処理順序が p 番目の作業とする．**手続き 7**は作業 $O_{p,M}$ を $B_{k,h,M}$ の先頭（もしくは最後）か，あるいはそれができない場合はできるだけ近い位置に移動させることによって新たなアクティブスケジュール $S_{M,p,k}$（もしくは $S_{M,p,h}$）を得る方法を示している．

手続き 5 GTアルゴリズム

（1） 技術的順序上，次に処理可能な作業の全体を G とし，G 中最早完了時刻が最も小さい作業を O^* とする．すなわち，$O^* = \arg\min\{O \in G \mid EC(O)\}$．$O^*$ を処理する機械を M^* とする．

（2） M^* 上，すでに $i-1$ 個の作業が処理されているとするとき，コンフリクト集合 $C[M^*, i]$ を求める．ただし，$C[M^*, i] = \{O \in G \mid ES(O) < EC(O^*)\}$．

（3） M^* 上 i 番目に処理すべき作業を $C[M^*, i]$ より1つ選び O とする．

（4） O を作業開始時刻が $ES(O)$ に等しくなるように処理する．

手続き 6 セミアクティブスケジュール修正アルゴリズム

（1） 手続き5のステップ（1）を実行し，O^* を求める．

（2） 手続き5のステップ（2）を実行し，$C[M^*, i]$ を求める．

（3） S の機械 M^* 上 i 番目の作業 O_{i,M^*} に対し，$O_{i,M^*} \in G$ かつ $EC(O) < ES(O_{i,M^*})$ となる $O \in C[M^*, i]$ が存在すればこれを選択し，そうでない場合，$O_{i,M^*} \in C[M^*, i]$ ならば O_{i,M^*} を O とし，それ以外の場合は $C[M^*, i]$ 中 S で最も早く処理されている作業を O とする．

（4） O を，作業完了時刻が $EC(O)$ に等しくなるように処理する．

手続き 7 修正版GTアルゴリズム

（1） 手続き5のステップ（1）を実行する．

（2） 手続き5のステップ（2）を実行する．

（3） $S_{M,p,k}, S_{M,p,h}$ のどちらを生成するかに応じて次のCASE 1もしくはCASE 2を実行する．

CASE 1: $S_{M,p,k}$ の生成

- もし $k \le i \le p$ かつ $O_{p,M} \in C[M^*, i]$ なら $O_{p,M}$ を選択し，最優先で処理する．
- そうでなければ $C[M^*, i]$ 中 S で最も早く処理されている作業を選択し最優先で処理する．

CASE 2: $S_{M,p,h}$ の生成

- もし $i = h$ または $C[M^*, i] = \{O_{p,M}\}$ なら，$O_{p,M}$ を選択し最優先で処理する．
- そうでなければ，$C[M^*, i]$ 中 $O_{p,M}$ 以外の要素のうち S で最も早く処理されている作業を選択し最優先で処理する．

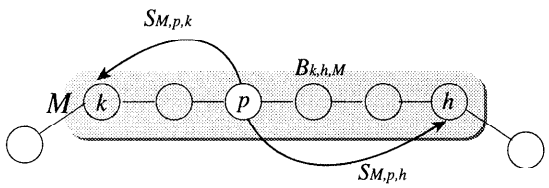


図9 $S_{M,p,k}$ および $S_{M,p,h}$ の生成
Fig. 9 $S_{M,p,k}$ and $S_{M,p,h}$ generation.

図9に手続き7での作業 $O_{p,M}$ の移動と $S_{M,p,k}$, $S_{M,p,h}$ の生成の様子を示す．S のすべてのクリティカルブロックに対し，可能なすべての $S_{M,p,k}, S_{M,p,h}$

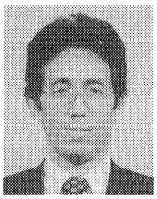
を考え，これらのうちから S に等しいものを除いて，実行可能なアクティブスケジュールの集合であるアクティブ CB 近傍 $AN^C(S)$ を次のように定義する：

$$AN^C(S)=\bigcup_{B_{k,h,M}}\Big(\bigcup_{k<p<h}\{S_{M,p,k},S_{M,p,h}\}\Big)\setminus S.$$

(平成 8 年 8 月 23 日受付)
(平成 9 年 4 月 3 日採録)

山田 武士（正会員）

昭和 39 年生．昭和 63 年 3 月東京大学理学部数学科卒業．同年 NTT 入社．現在，NTT コミュニケーション科学研究所所属．主として遺伝的アルゴリズム，シミュレーテッドアニーリング等の研究に従事．1996 年 9 月より 1 年間，英国コヴェントリー大学客員研究員．人工知能学会，電子情報通信学会各会員．



中野 良平（正会員）

昭和 22 年生．昭和 46 年東京大学工学部計数工学科卒業．工学博士．同年，日本電信電話公社（現 NTT）入社．以来，統計解析，データベース，人工知能，神経回路網の研究に従事．現在，NTT コミュニケーション科学研究所主幹研究員．電気通信普及財団賞（テレコムシステム技術賞）受賞（平成 9 年）．人工知能学会，神経回路学会，日本応用数理学会各会員．