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

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ひとつとされている．本論文では，確率的な局所探索法であるシミュレーテッドアニーリング（SA）法を用い，これに確定的（deterministic）な局所探索法である*shifting bottkeneck*（SB）法を組み合わせることによってJSSPの効率的な近似解法を提案する．現在のスケジュールに対して新たなスケジュールがクリティカルパス上の作業順序の入れ換えと，Giffler and Thompsonのアクティブスケジュール生成法を用いて生成され，SAによって確率的に受理される．さらに，受理されなかったスケジュールに対して，本方法のために変更を加えたSB法が適用され，スケジュールは修正される．修正されたスケジュールは改善が見られた場合に限って受理される．本方法をよく知られたいくつかのベンチマーク問題に適用した結果，解の品質において従来の近似解法を上回る結果を得ることができた．

Job-Shop Scheduling by Simulated Annealing Combined with Deterministic 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. This paper proposes a new method for solving JSSPs based on simulated annealing (SA), a stochastic local search, enhanced by shifting bottleneck (SB), a problem specific deterministic local search. In our method new schedules are generated by a variant of Giffler and Thompson's active scheduler with operation permutations on the critical path. SA selects a new schedule and probabilistically accepts or rejects it. The modified SB is applied to repair the rejected schedule; the new schedule is accepted if an improvement is made. Experimental results showed the proposed method found near optimal schedules for the difficult benchmark problems and outperformed other existing local search algorithms.

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．特にジョブショップスケジューリング問題（JSSP）は，複数の異なる仕事を処理するため，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である．

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呼ぶ．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*）と呼び，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と呼ぶ）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ひとつとされている．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

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

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的解法が提案されている．たとえば仕事の優先規則（priority rule）と，アクティブスケジュール生成に基づく方法は応用上よく用いられる[1]．Adams らは *Shifting Bottleneck*（SB）と呼ばれる確定的（deterministic）な近似解法を提案し，非常に効果的であることを示した[2]．これらは JSSP 専用の解法であるが，その他にもシミュレーテッドアニーリング（SA）や遺伝的アルゴリズム（GA），タブー探索などの，より汎用的な方法を用いた確率的（stochastic）な近似解法がいくつか提案されており，成果を収めている．山田らはクリティカルブロック上の仕事の移動に着目し，クリティカルブロック SA（CBSA）法を提案した[3),4]．CBSA 法は簡易さ，柔軟さなどの SA の長所を備え，しかも，難問のベンチマーク問題に対して，既存の SA 解法を上回る優れた解を求めることができた．しかし最近になって CBSA 法をさらに上回る近似解法も提案されている[5)~8]．

本論文の構成は次のようである．まず 2 章で JSSP が持つ基本的性質と SA で用いる近傍構造について説明する．3 章では JSSP の確率的近似解法として先に提案した CBSA 法について簡単に説明し，アクティブスケジュールと，相対受理確率を用いたその拡張を提案する．4 章では JSSP 専用の，確定的な近似解法として知られる SB 法について説明し，さらにこれを拡張し CBSA と組み合わせることによって得られるより優れた SA 解法である，CBSA+SB 法を提案する．また，mt10，mt20 および 10 tough problems と呼ばれる[9]，より大規模なベンチマーク問題を用いた詳細な計算結果を 5 章に示す．

2. 近傍構造

2.1 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*）がよく用いられる．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 1** に 3 × 3 問題を用いた選択グラフ

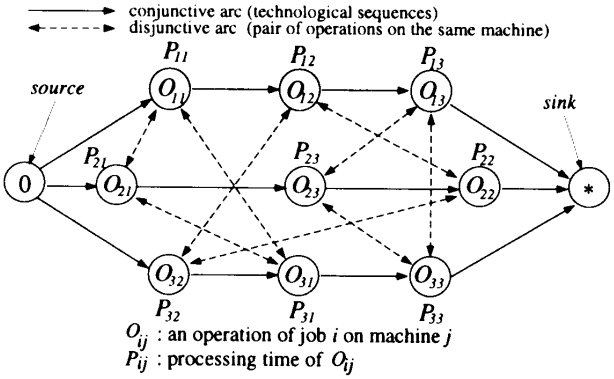

図 1 3 × 3 問題を用いた選択グラフ G の例
Fig. 1 The disjunctive graph G of a 3 × 3 problem.

の例を示す．図中円は節点，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すべて作業について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*) と呼ぶ．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 $G(S)=(V,C\cup S)$ がサイクル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*) と呼ばれる．与えられた完全選択と対応する実行可能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呼ばれ $\mathcal{P}$ で表す．$\mathcal{P}$ 上の作業はクリティカルな作業と呼ばれる．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呼ばれる．

2.2 クリティカルブロック近傍

SA 法では，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 $N(S)$ と呼ぶ．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 6), 10) 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[11]．この近傍を仮に AS（*adjacent swapping*）近傍と呼ぶ．

BAB 法における分枝操作として用いられる，もう 1 つの遷移操作が文献 5), 12) 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）と呼ぶ．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*）と呼ぶ．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．

SA による JSSP 解法において，AS 近傍，CB 近傍いずれも用いることができるが，CB 近傍の方がより強力であることが文献 4) において実験的に示されている．

2.3 アクティブ CB 近傍

しばしばスケジュールは完成後，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ることができる．どの作業もこれ以上このような繰り上げ操作ができないスケジュールをアクティブスケジュールと呼ぶ．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，最適化の際に，探索空間としてアクティブスケジュール全体を考えれば十分である．また，後に説明する SB 法によって生成されるスケジュールもアクティブスケジュールである．アクティブスケジュールは，Giffler and Thompson によって提案された，GT アルゴリズムによって生成される[13]．GT アルゴリズムの概要を**手続き 1** に示す．なおここで，作業 O の「最早完了時刻」($EC(O)$) とは，その作業を最優先で処理した場合の処理完了時刻のこととする．手続き 1 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．

上で説明したように，前候補，後候補は必ずしも実行可能ではない．また，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．そこで以下では，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，という性質を持つ新しい近傍を提案する．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 2** 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．この方法は一部文献 5) で用いられた考え方を拡張したものである．**図 2** に手続き 2 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新たな近傍 $AN^C(S)$ を次のように定義することにする：

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

手続き 1. GT アルゴリズム
(1) 未処理の作業中，最早完了時刻が最も小さい作業を求め O^* とする．すなわち，$O^* = \arg\min_{O:\text{未処理}}\{EC(O)\}$．$O^*$ を処理する機械を M^* とする．
(2) M^* 上，すでに $j-1$ 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 $C[M^*, j]$ を求める．$C[M^*, j]$ は，M^* 上の未処理の作業中，その処理が O^* と重複する作業の集合（O^* 自身も含む）である．
(3) M^* 上 j 番目に処理すべき作業を $C[M^*, j]$ より 1 つ選び O とする．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 2. 修正版 GT アルゴリズム
(1) 手続き 1 のステップ (1) を実行する．
(2) 手続き 1 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 CASE1 もしくは CASE2 を実行する．

CASE 1：$S_{M,p,k}$ の生成
- もし $k \le j \le p$ かつ $O_{p,M} \in C[M^*, j]$ なら $O_{p,M}$ を選択し，最優先で処理する．
- そうでなければ $C[M^*, j]$ 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 $p \le j \le h$ かつ $C[M^*, j]$ が $O_{p,M}$ 以外の要素を含めば，$C[M^*, j] \setminus \{O_{p,M}\}$ 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．
- もし $j = h$ または $C[M^*, j] = \{O_{p,M}\}$ なら，$O_{p,M}$ を選択し最優先で処理する．
- そうでなければ，$C[M^*, j]$ 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．

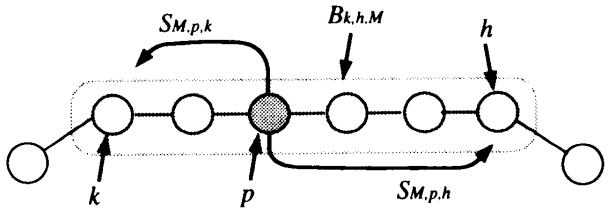

図 2 $S_{M,p,k}$ および $S_{M,p,h}$ の生成
Fig. 2 $S_{M,p,k}$ and $S_{M,p,h}$ generation.

手続き 3. CBSA メインループ
(1) $S = S_0$：ランダムな初期スケジュール，$L(S)$: S の総作業時間とする．カウンタ $k = 0$，初期温度 $T_{k=0} = T_0$ と設定する．
(2) 以下のステップを繰り返す：
(a) $AN^C(S)$ からランダムに S' を選択する．
(b) もし $L(S') \leq L(S)$ ならば無条件に S' を受理し，そうでなければ確率 $P(S) = e^{-(L(S')-L(S))/T_k}$ で S' を受理する．
(c) S' が受理された場合は $S = S'$ とする．
(3) $k = k + 1$ とし，現在の温度 T_k をアニーリングスケジュールに従って減少させる．
(4) もし $T_k > T_f$（T_f：終了温度）ならばステップ 2 へ，そうでなければ終了．

3. CBSA 法の拡張

CBSA（*Critical Block Simulated Annealing*）法[4]は前節で説明した近傍 $N^C(S)$ を用いた，JSSP のためのアニーリング法である．CBSA は**手続き 3** に示すような，中心となる繰返し処理のほかに，アニーリングの初期に温度を低い値から徐々に上昇させる過程を実行し，問題に応じて適応的に初期温度，終了温度を決定する「ウォームアップ処理」，十分長い遷移回数（これを再重点化頻度と呼び，R で表す）の後でもシステムの状態の改善が見られない場合，今まで得られた最良な状態へシステムをジャンプさせることによって，最良な状態の付近を重点的に探索する「再重点化戦略」から成り立っている．

本論文では，CBSA 法の近傍として $N^C(S)$ の代わりに $AN^C(S)$ を用いる．ただし，次に述べる工夫を施す．手続き 3 において，$AN^C(S)$ のどの要素に対しても受理確率が低い場合 S は局所最適解であって，システムの状態は長時間 S にとどまったままになり，なかなか他の解に遷移しない．本論文で扱う JSSP の探索空間は離散的であり，近傍 $AN^C(S)$ のサイズは有限で比較的小さい．したがってこのように局所最適解に陥った場合，近傍内の解はひととおりすべて評価された後も，どの解も受理されることなく，ひたすらステップ (2) の処理が繰り返される．このような状況は効率が悪い．そこで次のような相対受理確率 $\overline{P}(S_i)$ を導入し，もし近傍中のすべての解がすでに評価されていて受理確率も計算されている場合は，$AN^C(S)$ 中より確率 $\overline{P}(S_i)$ で要素 S_i を選択し，強制的に S_i に遷移させることにする．ここで $\overline{P}(S_i) = P(S_i)/\sum_{S_j \in AN^C(S)} P(S_j)$ である．このような考え方を用いて手続き 3 を修正したものが**手続き 4** である．

手続き 4. 修正版 CBSA メインループ
(1) $S = S_0$，$k = 0$，$T_{k=0} = T_0$ とする．また，$AN^C(S)$ 中ですでに一度選択され評価が済んでいる解の個数 $n = 0$ とする．
(2) 以下のステップを繰り返す：
(a) $AN^C(S)$ からランダムに S_i を選択する．もし $L(S_i)$ がまだ計算されていなければ $L(S_i)$ を計算し $n = n + 1$ とする．
(b) もし $L(S_i) \leq L(S)$ ならば無条件に S_i を受理し，そうでなければ確率 $P(S) = e^{-(L(S_i)-L(S))/T_k}$ で S_i を受理する．
(c) もし S_i が受理されれば $S = S_i$ とし $n = 0$ にリセットする．そうでない場合，もし $n = N$ ならば（N は $AN^C(S)$ の要素数），$AN^C(S)$ 中より確率 $\overline{P}(S') = P(S')/\sum_{S_j \in AN^C(S)} P(S_j)$ で要素 S_i を選択し $S = S_i$ とし，$n = 0$ にリセットする．
(3) $k = k + 1$ とし現在の温度 T_k をアニーリングスケジュールに従って減少させる．
(4) もし $T_k > T_f$ ならばステップ 2 へ，そうでなければ終了．

手続き 5. SBI
(1) $S = \emptyset$ とし，すべての機械を未スケジュールとする．
(2) 各未スケジュールの機械に対して部分問題である一機械問題を解き，最適な仕事の処理順序と，総作業時間を求める．得られた総作業時間の値の最も大きな機械（すなわちボトルネックの機械）に対する仕事の処理順序を S に組み込む．
(3) スケジュール済みの各機械を最適化し直す．
(4) S が完成したら終了．そうでない場合はステップ 2へ．

4. CBSA 法への SB 法の組み込み

Shifting Bottleneck（SB）法は，Adams らによって提案された JSSP の近似解法である[2]．SB 法では，各機械に対してもとの JSSP の部分問題である一機械問題を解き最適解を求める，という操作を繰り返すことによって JSSP の近似解を求める．これは JSSP では，最適解における各機械上の仕事の順序は，それぞれの機械に対して一機械問題を解いて得られる最適な仕事の処理順序と一致する場合が多い，という経験則に基づいている．SB 法は大きく 2 つの部分から構成される．前半（SBI）は一機械問題を解く操作の繰返しから構成されており（**手続き 5**），後半（SBII）は SBI を用いたバックトラックによる探索である．

このように SB 法は本来構成的（*constructive*）な解法であり，どの作業も処理されていない状態から出発して 1 つの実行可能スケジュールを生成する．しかしこのままでは SA 法に組み込むことができない．そこで本論文では，SA 法に組み込むため SB 法（SBI）の

表 1 MT ベンチマークを用いた CBSA と CBSA+SB の比較実験
Table 1 Comparisons between CBSA and CBSA+SB using MT benchmarks.

Prob	$n \times m$	CBSA ($R = 6,000$)				CBSA+SB			
		best	mean	std	BT	best	mean	std	BT
mt10	10 × 10	930	933.65	4.04	3 m 10 s	930	932.45	3.01	13 m 6 s
mt20	20 × 5	1178	1179.45	1.94	3 m 55 s	1165	1165.00	0.00	7 m 29 s

std：標準偏差，BT：最良解発見までの平均 CPU 時間

手続き 6. *BottleRepair*：逐次的 SBI
(1) S を完成された実行可能スケジュールとする．クリティカルでないすべての機械（クリティカルパスの一部を含まない機械）上の仕事の処理順序をすべてリセットし，未スケジュールとする．
(2) スケジュール済みの各機械を最適化し直す．
(3) リセットされた各機械に対して一機械問題を解き，各機械 M に対して得られた総作業時間の値の大きい順に次の (4), (5) を実行する．
(4) M に対し一機械問題を解き，結果を S に組み込む．この機械をスケジュール済みとする．
(5) スケジュール済みの各機械を最適化し直す．

手続き 7. CBSA への SB の組み込み
(b') もし S_i が受理されなければ S_i に対して *BottleRepair* を適用し，S_i^* を得る．もし $L(S_i^*) < L(S)$ ならば S_i^* を受理し，$S_i = S_i^*$ とおく．

手続きを部分的に変更し，ある完成されたスケジュール S を修正，改善する逐次的 (*iterative*) な手続きを構成する．これを *BottleRepair* と呼ぶ．**手続き 6** に *BottleRepair* の概要を示す．なお手続き 6 のステップ (2), (5) で用いる最適化の処理は手続き 5 のものと同様であり，要約すると，スケジュール済みの機械を 1 つ選びそのうえの仕事の処理順序をリセットし，一機械問題を解き直す，これを解が改善する限り繰り返す，という処理である．詳細は文献 2) を参照されたい．

手続き 6 における「クリティカルでない一部の機械を一時的に未スケジュールとし，最適化し直す」というこの方法の基本的な考え方は文献 2) で用いられており，*BottleRepair* はそれを発展させたものである．

手続き 4 で示したように，$AN^C(S)$ より選択された S_i は受理確率に応じて確率的に受理される．ここで S_i が受理されなかった場合に限り *BottleRepair* が適用される．この結果得られたスケジュール S_i^* は S と比較して改善が見られた場合，確定的に受理される．まとめると手続き 4 のステップ (2b) の直後に手続き 7 のステップ (b') を加えることになる．

5. 評価実験

5.1 SB 法組込みの評価

1963 年に Muth and Thompson によって提示された 10 仕事 10 機械からなる問題 (mt10)，および 20 仕事 5 機械からなる問題 (mt20) は JSSP の代表的ベンチマーク (MT ベンチマーク) としてよく知られている．この 2 つの問題に対し，SB 法を組み込んだ場合と組み込まない場合についてそれぞれ比較実験を行った．**表 1** は mt10, mt20 問題を用い，乱数の初期値を変えてそれぞれ 30 分の時間的制約のもと 20 回行った実験結果である．表 1 中の BT は最良解発見までの平均 CPU 時間であり，言い換えれば，BT 時間経過後は最良解の更新がいっさい見られなかったことを示している．ここで使用したマシンは SPARCstation 10 であり，プログラムはすべて C で書かれている．

mt10 問題に対して SB を組み込まず CBSA 法単独で用いた場合 (表 1 の CBSA)，最適解 $L = 930$ が求められたのは 20 回中 11 回であった．成功した 11 回について最適解が求められるまでの平均所要 CPU 時間は 3 分 10 秒，最も速く求められた場合が 1 分 21 秒である．最適解が求められたのは全体のうちほぼ半数についてのみであるが，その場合，最適解が求められるまでの CPU 所要時間は短い．この問題に関しては，温度スケジュールを変更し，温度をよりゆっくり下げることによって最適解が求められる割合をさらに増やすことは可能であるが，逆に所要 CPU 時間は長くなってしまう[4]．また，mt20 問題に対しては，SB を組み込まない CBSA では一度も最適解である $L = 1165$ は求められらず，ほとんどの場合，$L = 1178$ の局所解に陥ってしまった．この問題の場合，時間的制約を緩めて，温度スケジュールを変更してもこの傾向は変わらなかった．なお，再重点化頻度は $R = 6,000$ を用いた．

一方，SB を組み込んだ場合，mt10 問題に関して最適解が求められる割合は 20 回中 12 回と，少しだけ増加しているが，最適解が求められるまでの平均所要 CPU 時間は約 4 倍増加している．この結果より，mt10 問題に関しては SB を組み込まない CBSA 法で

表2 JSSPの10の難問を用いた実験結果
Table 2 Results of 10 tough JSSPs.

Prob	$n \times m$	LB	CBSA+SB best	CBSA+SB mean	CBSA+SB std.	CBSA+SB BT	Best	出典
abz7	20 × 15	655	*665	671.0	3.92	2 h 10 m 14 s	665	Taillard[6]
abz8	20 × 15	638	675	680.0	3.13	2 h 26 m 15 s	670	Aarts[14]
abz9	20 × 15	656	**686	698.6	7.42	2 h 25 m 49 s	691	Taillard[6]
la21	15 × 10	–	*1046	1049.3	3.32	6 m 1 s	1046	Vaessens[7]
la24	15 × 10	–	*935	939.2	1.99	1 h 43 m 46 s	935	Applegate[9]
la25	20 × 10	–	*977	979.3	1.62	1 h 8 m 37 s	977	Applegate[9]
la27	20 × 10	–	*1235	1242.4	6.15	2 h 10 m 5 s	1235	Carlier[16]
la29	20 × 10	1130	**1154	1162.4	7.10	1 h 30 m 34 s	1157	Balas[17]
la38	15 × 15	–	1198	1206.8	4.53	57 m 59 s	1196	Nowicki[8]
la40	15 × 15	–	1228	1230.2	2.32	55 m 31 s	1222	Applegate[9]

十分であることが分かる．逆に，mt20問題に関しては20回中すべての場合について最適解である $L = 1165$ の解を求めることができた．平均所要CPU時間は7分29秒，最も速く求められた場合が1分22秒である．mt20問題に関してはSBを組み込んだ効果がはっきりと現れている．ただし，探索の初期の段階で最適解か，それに近い解に達してしまうため，再重点化は用いていない．

以上の結果より，SBを組み込まないCBSA法が十分有効な場合は，さらにSBを組み込んでもあまり効果が期待できないが，単独のCBSA法ではうまく解けない問題に対しては，SBを組み込むことによって，処理は重くなるものの，それを補う解の品質の向上を期待できることが分かった．

5.2 提案法の性能評価

本論文で提案したCBSA+SB法を現在世界的に知られる10の難問[14]に適用した結果を**表2**のCBSA+SB欄に示す．なお，使用したマシンはHP735であり，SPARCstation 10の約1.5倍の性能を持つ．各実験は，それぞれ3時間の時間制約のもと，既知の最適解もしくは理論的下界に達した時点で終了させる．なお，再重点化頻度は $R = 1000$ を用いた．また参考のため，Best欄に現在知られる最良解の値と，その解を最初に求めた著者の名前およびその出典を記す．また，LB欄に理論的下界を示す[15]．我々は10問中7問について既知の最良解と同等（表中 ⋆）もしくは，それを上回る結果（表中 ⋆⋆）を得た．特にabz9，la29に関しては既知の最良解を更新した．その後la29に関しては我々が求めた結果 $L = 1154$ を初期解としてApplegateらのshuffleアルゴリズムを適用した結果 $L = 1153$ を持つ解が得られている[15]．

表2に示すように，CBSA+SB法は表にある10の難問のほとんどについて，得られた解の品質において他解法を上回る良好な結果を得ている．一方，計算時間に関しては，CBSA+SB法では $L = 1235$ の解を求めるのに2時間10分かかるのに対し，Vaessensらによると Applegateの方法ではla27に対し，SUN SPARC station ELC上で $L = 1269$ の解が10分程度で求められる[7]．ここでSUN SPARC station ELCはHP 730の約10倍遅いため，この結果だけを見るとCBSA+SB法はあまり計算効率が良いとはいえない．実際，CBSA+SB法の *BottleRepair* 処理は，解を改善せずに終了することが多く，この部分にはまだ改善の余地がある．

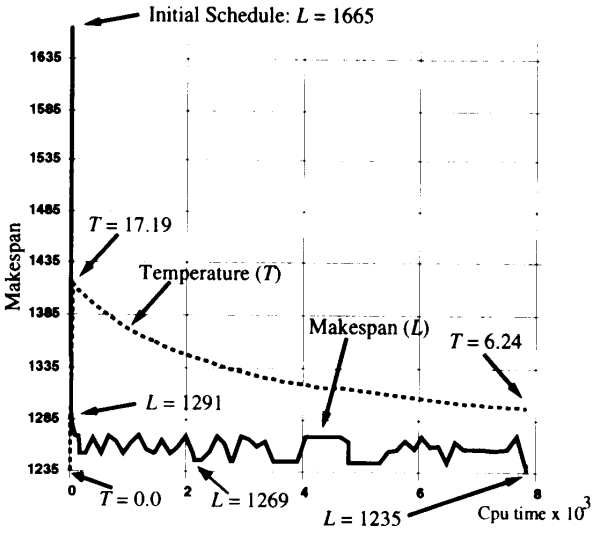

図3 la27を用いたCBSA+SBの収束の様子
Fig. 3 The time evolution of CBSA+SB trial for the la27 problem.

しかし，同じ実験においてCBSA+SB法は $L = 1269$ となる点を約2分40秒後に通過しているので（**図3**参照），$L = 1269$ を求めるまでの時間で比べると両者にあまり差はない．このように，表2に示す各文献では，解法ごとに実験に用いる計算機が異なり，また，文献によっては最良解に到達した時間の記載はあっても，最良解到達以降も探索を続け，これ以上解

の改善が望めないと判断して探索を打ち切るまで，全体としてどのくらいの時間を費やしたかが不明なことが多いため，計算時間の厳密な比較は難しい．

たとえば，Aartsらの SA によれば，$L = 1269$ の解を求めるのに VAX785 上で約 6.5 時間かかっている[14]．また，Nowicki らは TS を用いて $L = 1236$ の解を得ているが，所要時間は不明である[8]．一方，Balas らの近似解法によれば，$L = 1235$ の最適解は Sparc 330 上で 315 秒で求められる[17]．

5.3 提案法の性能に対する考察

BottleRepair はスケジュールのクリティカルパスに着目し，確定的な方法で一機械問題を繰り返し解くことによって作業の処理順序を変更するものであった．一般に *BottleRepair* によって S より優れたスケジュール S' が生成された場合，S' のクリティカルパスは S に比べて大きく変更されている．つまり SB は，解への大規模な変更によって，SA の時間のかかる，小刻みな遷移の積み重ねを省略し，探索の効率化を図る役割を果たす．一方，CBSA によるクリティカルパスの変更は確率的であり，1 回の変更の幅も比較的小さい．したがって SA は SB の生成する解に対して確率的摂動を与え，局所解からの脱出を助けるという効果を持つ．SB と SA の融合である本方法は，このような，いわば相乗効果を持っている．

例として la27 に対して CBSA+SB を適用し，最適解が求められたランの収束の様子を図 3 に示す．図中横軸は CPU 時間（秒）であり，縦軸は実線が解の評価値である総作業時間（L），点線が温度（T）である．この例では，実験開始時に総作業時間が $L = 1665$ であった後，開始直後のウォームアップ期間の間に $L = 1291$ まで急激に減少し，その後 12 回の再重点化を経て最終的に $L = 1235$ に達している．この実験においては，生成されたスケジュールの総数は 56059，受理されたスケジュール数は 8850，そのうち *BottleRepair* による受理数は 2204 であった．受理されたスケジュールは生成された全スケジュールの約 15 %であり，そのうちの 25 %が *BottleRepair* によって受理され，CBSA の効率化に寄与していることが分かる．また残りの 75 %は CBSA によって直接受理され，*BottleRepair* に確率的摂動を与える効果を持つ．総作業時間曲線に見られる振動は再重点化のためである．

本研究は，汎用的な最適化法である SA 法に，スケジューリング問題専用の近似解法である SB 法を組み合わせることによって，SA 法単独で用いる場合に比べどれだけ効果的であるかを調べたものである．したがって提案する CBSA+SB 法は，そこで用いられている SB 法が JSSP 専用の近似解法であるため，他の最適化問題にそのまま適用可能ではない．しかし他の最適化問題に対しても前章の手続き 7 において *BottleRepair* の代わりにその問題専用の近似解法を組み込むことによって同様な効果が期待できる可能性は十分にあると考えられる．

6. 結　び

本論文では，先に提案したクリティカルブロックの考え方に基づく SA 法である CBSA 法の近傍構造を GT アルゴリズムを用いて改良し，さらに確定的（deterministic）な近傍探索アルゴリズムである SB 法との融合を図ることによってより強力な解法である CBSA+SB 法を提案した．12 のベンチマーク問題に対して本方法を適用した結果，大部分の問題に対して実用的時間内に，解の品質において既存のアルゴリズムを上回る結果を得ることができた．今後は諸解法の間の関係を整理し，さらに高速化の研究を進めたい．

参考文献

1) Panwalkar, S. and Iskander, W.: A Survey of Scheduling Rules, *Oper. Res.*, Vol.25, No.1, pp.45–61 (1977).
2) 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).
3) Yamada, T., Rosen, B. and Nakano, R.: A Simulated Annealing Approach to Job Shop Scheduling using Critical Block Transition Operators, *Proc. IEEE Int. Conf. on Neural Networks (Orlando, Florida.)*, pp.4687–4692 (1994).
4) 山田武士，Rosen, B.E.，中野良平：クリティカルブロックシミュレーティドアニーリング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).
5) 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).
6) Taillard, E.: Parallel Taboo Search Techniques for the Job-shop Scheduling Problem, *ORSA J. on Comput.*, Vol.6, No.2, pp.108–117 (1994).
7) Vaessens, R., Aarts, E. and Lenstra, J.: Job Shop Scheduling by Local Search, Technical Report COSOR 94-5, Eindhoven University of Technology, Dpt. of Math. and CS (1994).
8) Nowicki, E. and Smutnicki, C.: A Fast Taboo

Search Algorithm for the Job Shop Problem, to appear, *Mgmt. Sci.* (1995).
9) 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).
10) Laarhoven, P.v., Aarts, E. and Lenstra, J.: Job Shop Scheduling by Simulated Annealing, *Oper. Res.*, Vol.40, No.1, pp.113–125 (1992).
11) Balas, E.: Machine Sequencing via Disjunctive Graphs: an Implicit Enumeration Algorithm, *Oper. Res.*, Vol.17, pp.941–957 (1969).
12) 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).
13) Giffler, B. and Thompson, G.: Algorithms for Solving Production Scheduling Problems, *Oper. Res.*, Vol.8, pp.487–503 (1960).
14) 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).
15) Vaessens, R.: Personal Communication (1995).
16) Carlier, J. and Pinson, E.: Adjustments of Heads and Tails for the Job-Shop Problem, *E. J. of Oper. Res.*, Vol.78, pp.146–161 (1994).
17) Balas, E. and Vazacopoulos, A.: Guided Local Search with Shifting Bottleneck for Job Shop Scheduling, Technical Report #MSRR-609, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania (1994).

(平成7年9月11日受付)
(平成8年2月 7 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．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．同年，日本電信電話公社（現NTT）入社．以来，統計解析，分散処理，データベース，人工知能の研究に従事．工学博士．現在，NTTコミュニケーション科学研究所主幹研究員．知能ロボット，人工脳の研究に興味を持つ．人工知能学会，神経回路学会，日本応用数理学会各会員．