論　文

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

山田武士 (NTT)
Bruce E. Rosen (University of Texas)
中野良平 (NTT)

Critical Block Simulated Annealing for Job Shop Scheduling

Takeshi Yamada (NTT), Bruce E. Rosen (University of Texas), †1Ryohei Nakano (NTT)

The Job Shop Scheduling Problem is one of the most difficult $\mathcal{NP}$-hard combinatorial optimization problems. This research investigates finding near optimal schedules using simulated annealing and a recently developed schedule permutation procedure. New schedules are generated by permuting operations within existing schedules. Simulated annealing (SA) probabilistically chooses one of the new schedules and probabilistically accepts or rejects it, allowing importance sampling search over the job shop schedule space. The initial and final temperatures are adaptively determined *a priori*, and a reintensification strategy improves the search by resetting the current temperature and state.

Experimental results show this method, possessing the simplicity and flexibility of SA, can find near optimal schedules for the difficult benchmark problems and can outperform the existing SA adjacent swapping approach as applied to job shop scheduling problems.

キーワード：シミュレーテッドアニーリング, ジョブショップスケジューリング, 組合せ最適化

1. はじめに

本論文で扱うジョブショップスケジューリング問題について簡単に説明する. 今, n 個の仕事 $J_1, \ldots, J_n$ を m 台の機械 $M_1, \ldots, M_m$ 上で処理することを考える. 各仕事を処理する機械の順序は仕事毎に予め与えられており, 技術的順序とよばれる. 各機械 M_r 上での仕事 J_j の処理のことを作業とよび, $O_{j,r}$ と表す. 各作業 $O_{j,r}$ は処理時間 $p_{j,r}$ をかけて機械 M_r 上で中断なく処理される. ここで各機械は全て異なり, 同時に二つ以上の作業を処理することができないとする. 全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, JSSP と記す) 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て, L を最小にするような各機械上での仕事の処理順序を全て決定することである.

JSSP は単に $\mathcal{NP}$-困難であるばかりか, 最適解を求めるのが極端に難しい問題とされている. 例えば, 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 [1]. 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 年近くにわたって盛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, そのほかにいくつかの近似的解法が提案されている. 例えば仕事の優先規則 (priority rules) と, アクティブスケジュール生成に基づく方法は応用上良く用いられる [2]. Adams らは *shifting bottleneck* 法とよばれる, より洗練された近似解法を提案し [3], 非常に効果的であることを示した. そのほかにも, シミュレーテッドアニーリング (SA : Simulated Annealing) 法 [4, 5] や, 遺伝的アルゴリズム (GA) [6], タブー探索法 [7] などによる近似解法もいくつか提案されており, 成果を収めている.

本論文では, ジョブショップスケジューリング問題の解法として, クリティカルブロック操作に基づく SA 法を提案し, ベンチマークを用いた実験を通じてその有効性を論じる. 2 節では背景技術として, クリティカルブロックと, その上の遷移操作, 近傍, SA 法の概要について説明する. 3 節では本論文が提案する解法について説明し, 4 節ではベンチマークを用いた実験結果を, 他の解法, 特に既存の SA 法と比較して述べ, 本解法の効果を論じる.

†1 1993 年 6 月 1 日から同年 8 月 31 日まで招聘研究員として NTT コミュニケーション科学研究所に滞在.

2. 背景技術

<2·1> クリティカルブロック　JSSPの解の表現法として選択グラフ(*disjunctive graph*)が良く用いられる. 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及び二つの特殊節点: ソース(o), 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 arc)の集合で, 同一機械上の作業の対を表す.

各作業$O_{j,r}$の処理時間$p_{j,r}$は各節点に付与された重み$p_{j,i}$によって表す. 図1に3×3問題を用いた選択グラフの例を示す. 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全ての作業について処理順序を全て決定することによって得られる.

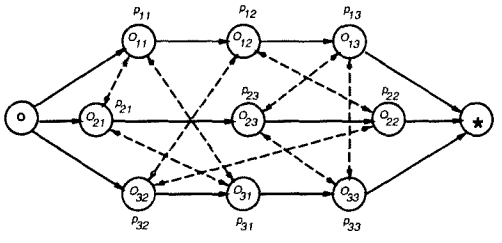


図1 3×3問題の選択グラフの例
Fig. 1. The disjunctive graph G of a 3×3 problem.

選択グラフモデルにおいて, このことはDの全ての無向(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*) S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$G(S)=(V, C\cup S)$がサイクルを持たない(*acyclic*)こととは同値である. このときSは「完全選択」(*complete selection*)とよぶ[†2]. 対応するスケジュールの総作業時間はo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$\mathcal{P}$と呼ばれる. クリティカルパス上の作業列Bは, 次の性質が成り立つときクリティカルブロック(または単にブロック)と呼ばれる.

1. Bの全ての作業は同一機械上にある.
2. Bの最初(最後)の作業の同一機械上での直前(直後)のの作業は(もし存在すれば)$\mathcal{P}$上にない.

<2·2> クリティカルブロック近傍　SA法では, 一つの解Sから一回の遷移によって到達可能な全ての解の集合を近傍$N(S)$と呼ぶ. 例えばLaarhoven等が用いているSA法では, クリティカルブロック内の連続する二つの作業の処理順序を反転させる遷移操作を考え, 近傍としてこの遷移の結果得られる解全体の集合を用いている[4, 5](仮にAS近傍[†3]と呼ぶ). 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[8].

BAB法における分枝操作として用いられる, もう一つの遷移操作が[9, 10]において提案されている. そこではブロック内の一番先頭, もしくは一番最後の作業を変更することによって遷移操作が定義される. この考え方に基づき, 新たな近傍$N^r(S)$を次のように定義する.

B_jをクリティカルパス$\mathcal{P}$上, 特定機械上の連続する作業$u_{i,j}$全体からなる, j番目のブロックとする. すなわち$B_j=(u_{1,j},\ldots,u_{i,j},\ldots,u_{l_j,j})$, ただし, l_jはB_j内の作業の個数である. このとき$\mathcal{P}$は$(\circ, B_1,\ldots,B_j,\ldots,B_k,\star)$のように表現される. $l_j>1$なるブロックB_jに対して, 次の二つの遷移操作$\mathcal{S}^b$および$\mathcal{S}^a$を考える.

- $\mathcal{S}^b(S,u_{i,j})\ (1<i\le l_j)$は, SのブロックB_j内の作業$u_{i,j}$をB_jの先頭に移動することによって得られる選択.
- $\mathcal{S}^a(S,u_{i,j})\ (1\le i<l_j)$は, SのブロックB_j内の作業$u_{i,j}$をB_jの最後に移動することによって得られる選択.

$\mathcal{S}^b$もしくは$\mathcal{S}^a$によって得られる選択はそれぞれ「前候補」(*before candidate*)もしくは「後候補」(*after candidate*)と呼ばれる. 図2に上記の遷移操作と, その結果生じるブロック内の作業の処理順序の入れ換えの様子を示す.

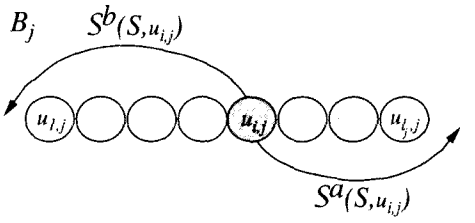


図2 ブロックB_jとS^b, S^aによって生じる作業の移動の例.
Fig. 2. Permutations of a block B_j when applying $\mathcal{S}^b$ or $\mathcal{S}^a$.

$\mathcal{S}^b$および$\mathcal{S}^a$によってSより得られる全ての選択の集合は次のように表される:

$$N'^C(S)=\{\mathcal{S}^b(S,u_{i,j})\mid l_j>1, i=2,\ldots,l_j, j=1,\ldots,k\}\bigcup\{\mathcal{S}^a(S,u_{i,j})\mid l_j>1, i=1,\ldots,l_j-1, j=1,\ldots,k\}$$

$N'^C(S)$の要素である選択のうちいくつかはサイクルを持つ可能性がある. その場合対応するスケジュールは実行可能ではない. したがってクリティカルブロック近傍(CB近傍)$N^C(S)$はそれらを除いて次のように与えられる:

$$N^C(S)\ =\ \{S'\in N'^C(S)\mid S'\text{は完全選択}\}$$

仮りにSが長さ2以上($l_j>1$)のブロックを一つも持たない場合Sは最適解である. 何故ならこのとき$\mathcal{P}$の弧は全て

[†2]簡略化のため, 有向弧の選択と対応するスケジュールは同じ記号Sを用いて表す.

[†3]Adjacent Swapping neighborhood

C の元からなり, $L(S)$ は $\mathcal{P}$ 上の全作業の加工時間の和であり, これは L の下界に等しい[5]. そうでない場合, S は少なくとも一つは長さ2以上のブロックを持ち, $N'^C(S)$ は空でない. さらに $N'^C(S)$ は実行可能でないスケジュールを含む可能性があるが, $N^C(S)$ は決して空にならない. 何故ならブロック内の連続する二つの作業の処理順序の入れ換えによって決してサイクルは生じず[5], したがって $\mathcal{S}^b(S, u_{2,j})$ および $\mathcal{S}^a(S, u_{l_j-1,j})$ に対応する少なくとも二つのスケジュールは常に実行可能である.

本論文では, この $N^c(S)$ に基づいた SA を提案する. $N^c(S)$ は次の意味で効果的である. まず, $\mathcal{P}$ 上でない遷移は決して L を減少させない. 何故なら遷移後のスケジュールにも $\mathcal{P}$ はパスとして存在するからである. 更に $\mathcal{P}$ 上の遷移に関しては, 次の事実が知られている (証明は[10]を参照).

補題 S, S' を $L(S') < L(S)$ となる二つの完全選択とする. このとき, $G(S)$ には少なくとも一つのブロック $B_j\,(l_j > 1)$ と, B_j 内の作業 $u_{i,j}$ が存在して, $G(S')$ において $u_{i,j}$ は B_j のほかのどの作業より先もしくは後で処理される.

従って, S よりも短いスケジュールに到達するためには必ず少なくとも一つのブロックに関してその上のある作業をそのブロックのどの作業よりも先, もしくは後に移動させなければならない.

しばしばスケジュールは完成後, 次のようにいくつかの作業を左方へ移動させることによって改善されることがある. 任意のスケジュールにおいて, 機械 M_r 上の二つの作業 $O_{j,r}, O_{k,r}$ を考える. いま仮に $O_{k,r}$ の処理が $O_{j,r}$ に先行し M_r 上, $O_{k,r}$ の処理以前に $p_{j,r}$ より長い遊休時間があるとすると, $O_{j,r}$ を $O_{k,r}$ の前に繰り上げることができる. これを *permissible left shift* と呼ぶ. どの作業も, これ以上 permissible left shift によって繰り上げること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[11]. 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. $N^c(S)$ の要素 S' がアクティブスケジュールでない場合, アクティブスケジュールに修正した結果が S と異なる場合に限って修正を行なう.

<2·3> シミュレーテッド・アニーリング 一般にシミュレーテッド・アニーリング (SA) 法とは次のような過程である[12]:

1. 適当な (十分大きい) 初期温度 T_0 とランダムな初期状態 x_0 を選び, $T_{k=0} \leftarrow T_0$, $x \leftarrow x_0$ とする.
2. 生成関数 g を用い x の近傍 $N(x)$ より, 新たな状態 x' を生成する. $x' \leftarrow g(x)$.
3. 評価関数 f を用い, x' の評価値を計算する. $E' \leftarrow f(x')$.
4. 受理関数 h の値に応じて確率的に x' を受理し, $x \leftarrow x'$, $E \leftarrow E'$ とする.
5. 温度を変更する. (例えば $T_{k+1} \leftarrow \gamma T_k$, $0 < \gamma < 1$).
6. 十分低い温度 T_f に対して $T_k < T_f$ なら x および E を出力して終了, そうでなければ $k = k + 1$ とし, ステップ2へ.

組合せ最適化問題の場合, g として一様分布が用いられることが多い. すなわち,

$$\mathrm{Prob}(g(x) = x_j) = 1/n,\ x_j \in N(x) \quad j = 1, \ldots, n. \quad (1)$$

ただし $n = |N(x)|$ は近傍 $N(x)$ のサイズを表す. 受理関数 h は現在の状態 x の評価値 $f(x)$ と, 新たな状態 x' の評価値 $f(x')$ に基づき次のように定義される.

$$h(x') = \min(1, e^{-(f(x')-f(x))/T}) \quad (2)$$

アニーリングにおける温度の変更は, 遷移回数 k の対数に反比例して温度を下げるのが理想的であるが, 実際には多大な時間がかかり実用的でない. したがって次式に示すように指数的に温度を下げることにする.

$$T_k = T_0 e^{(-ck)} \quad (3)$$

ここで c は一回の遷移で温度をどれくらい下げるかを決める正の定数である.

3. CBSA 法

本論文で提案する CBSA (*Critical Block Simulated Annealing*) 法は, JSSP の解空間を有効に探索するため以下に述べる工夫を盛り込んだ近傍探索法である. ここで状態 x はスケジュール S であり, 評価値 $f(x)$ は S の総作業時間 $L(S)$ によって与える. また, S の近傍としては **<2·2>** 節で定義された CB 近傍 $N^C(S)$ を用いる. したがって $S' \in N^C(S)$ の受理される確率は

$$P(S') = \min(1, e^{-(L(S')-L(S))/T}) \quad (4)$$

である.

<3·1> 再重点化戦略 SA による探索の過程において, システムが最適解からは程遠い, 見当違いな場所をさまよったり, 極端に低い温度において局所最適な状態に陥ったため, 近傍への遷移をほとんど受理せず, したがってこの状態から抜け出せなくなったりすることがある. そこで, 十分長い遷移回数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ことがしばしば有効である. 実際には, 十分大きい受理回数を経た後でも状態の改善が見られない場合, もしくは状態生成回数に対する受理回数の比率 (*AG* 比と呼ぶ) が予め定めた *AG* 比の閾値限界より低い場合, 現在の状態を今まで得られた最良の状態と入れ換える. 同時に最良の状態からの遷移を受理できるように, 温度パラメータを最良解とその近傍の各要素との総作業時間の差の自乗平均のルートをもとに再設定する. ただしこの変更はこの値が現在の温

度パラメータより大きい場合のみ行なう[†4]. このような再重点化戦略 (*reintensification strategy*) の手法は文献 [13, 14] において関数の最適化のために用いられた再アニーリング (*reannealing*) の手法を参考にした.

<3·2> アニーリングパラメータ　アニーリングスケジュールにおいて, 初期温度・最終温度 (T_0, T_f) および, 総遷移回数 (k_f) を適切な値に定める必要がある. 各 JSSP はそれぞれ異なる特徴, 制約, 問題の難しさを持っている. したがって, アニーリングのパラメータは問題に応じて個別に定めるのが望ましいが, これらの値を事前に決定するのは難しい. このための工夫としては例えば, 初期温度・最終温度 (T_0, T_f) をアニーリングの初期および終了時に解が悪化する方向への遷移をどの程度受理すべきかをもとに決定する手法がいくつか知られている [15], [16].

そこで本論文でも, 遷移によって生成 (*generate*) された解のうち元の解よりも悪化した場合 (*uphill* 方向) について, その総数に対する受理 (*accept*) 回数の比率 (μAG 比と呼ぶ) を考え, 初期温度・終了温度をこの μAG 比に基づいて決定する. これによって初期温度・最終温度を問題毎に適応的に定めることができる. まずアニーリングの初期には, *uphill* 方向に対しても受理確率を十分大きくする必要があり, μAG 比を大きくしなければならない. 一方アニーリングの末期においては, 解が改善する方向への遷移のみに焦点を絞るため, μAG 比を十分小さく設定しなければならない.

具体的には, アニーリングの初期に次のような短い「ウォームアップ期間」を導入することによって初期温度・終了温度を適応的に決定する. まず望ましい μAG 比の初期値および終了値を与える. 例えば μAG 比の初期値を 50% とする. これは現スケジュールより悪いスケジュールを 50% の割合で受理することに相当する. また例えば μAG 比の終了値を 0.2% とする. 次に温度 T を μAG 比が 0 になるような十分低い値 (例えば $T = 0$) に設定し, アニーリングとは逆に温度を徐々に上昇させる過程を実行する. これを「ウォームアップ期間」と呼ぶ. 実際の μAG 比が 0.2%, 50% に等しいか大きくなった時点での温度を記録し, その温度を終了温度, 初期温度とする.

一回の実験における総遷移回数 k_f は 1) 一回の温度の下げ幅は十分小さくなければならない, 2) あまり極端に長い時間をかけない (一つの実験がせいぜい数時間), 3) 一つの実験の間に十分たくさんの解を生成しなければならない, などを考慮に入れながら実験的に定めた.

再重点化戦略を適用する場合, 二つのパラメータを設定しなければならない. 一つは頻度 R であり, R 回の受理回数を経た後でも解の改善が見られない場合は再重点化を行なう. もう一つは AG 比の閾値限界であり, 現在の AG 比がこの値を下回る場合に再重点化を行なう.

[†4] 実際の実験ではこの条件を満たすことは稀であったので, この温度変更の詳細は省略する.

4. 実験結果

1963 年に Muth and Thompson によって提示された 10 仕事 10 機械からなるベンチマーク問題は, 20 年以上未解決のままであった難問である. この MT10×10 問題を用いて, 再重点化戦略および CB 近傍の有効性を検証するために行なった実験結果を表 1 に示す. 乱数の初期値を変えた各 10 回の実験によって CBSA 法と, AS 近傍を用いるアニーリング法 (ASSA 法) とを比較した. 各実験において, 再重点化頻度 ($R = 0$: 再重点化なし, $R = 3,000$, $R = 10,000$) 及び用いた近傍構造の違いを除いた他の条件は全て等しい. 各条件のもとでの, 最小値, 最大値, 平均, 標準偏差, 平均所要時間を記す. 実験の結果, どの条件のもとでも所要時間はほとんど変わらないものの, $R = 3,000$ とし, CBSA 法を用いた場合の結果が最も優れていることがわかる.

表 1　CBSA 法と ASSA 法との比較実験
Table 1. Comparison between CBSA and ASSA

Method	$R = 0$	$R = 3,000$	$R = 10,000$
CBSA	930　945	930　938	930　938
	937.8　4.2	930.8　2.4	933.1　3.8
	(41 m 13 s)	(44 m 36 s)	(42 m 44 s)
ASSA	938　972	930　951	934　970
	951.6　10.2	939.5　5.1	944.4　10.2
	(49 m　9 s)	(35 m 43 s)	(40 m 46 s)

左上: 最小値, 右上: 最大値,
左下: 平均値, 右下: 標準偏差, (平均所要時間)

表 2 に, $R = 3,000$ とし, CBSA 法を用いた場合についての詳細を示す. 初期温度は μAG 比によって適応的に設定されるので実験毎に異なる. 参考のため最適解が発見された時点における温度パラメータ (T_l) の値を付す. 実験の性質上, プログラムはすでに知られている最適解が発見された時点で終了する. したがって各実験終了時の温度パラメータの値は予め定めた T_f (平均 2.14, 標準偏差 1.51) より一般に大きく, ばらつきも大きい.

表 2　CBSA 法を用いた実験結果 ($R = 3000$)
Table 2. Ten Trials using CBSA Method ($R = 3000$).

No.	Best	$e \times 10^3$	$g \times 10^3$	T_0	T_l	Time
1	*930	481	548	51.84	6.19	38 m 0 s
2	*930	510	537	44.83	10.61	41 m 28 s
3	*930	507	580	47.04	6.20	40 m 8 s
4	*930	344	332	49.40	17.33	28 m 45 s
5	*930	367	404	44.76	7.89	28 m 40 s
6	*930	459	472	47.09	14.83	37 m 59 s
7	*930	372	405	38.69	9.07	29 m 8 s
8	*930	649	711	38.76	9.57	51 m 39 s
9	*930	317	352	36.83	8.41	25 m 0 s
10	938	459	1000	54.39	0.50	36 m 6 s

e: 総評価回数, g: 総遷移回数

また, 遷移が受理されなければ状態および近傍は変化しないので, 評価結果を保持することによって再利用が可能である. 表中, 評価総数と生成総数が異なるのはこのような再利

用の結果である. なお * 印は最適解を表す. 10 回の実験中, 9 回について最適解が求まっている.

全てのプログラムは C 言語で書かれ, 全実験を通じて総遷移回数は $k_f = 1,000,000$, 再重点化を行なう場合の AG 比の閾値限界は 10^{-3} とした. 実験の所要時間は (以下も全て) SUN SparcStation 2 上での値である.

図 3 に, 典型的な実験結果による, 再重点化戦略を用いた場合 ($R = 3,000$) と用いない場合における総作業時間の推移の様子を示す. 横軸は生成されたスケジュール総数, 縦軸は総作業時間と最適値との差分である. 再重点化戦略を用いない場合には, 一旦は最適解に近い解にたどり着きながら再び遠ざかってしまうのに対し, 再重点化戦略を用いると最良解の周辺を重点的に繰り返し探索するため, 最終的には最適解に到達する様子がわかる.

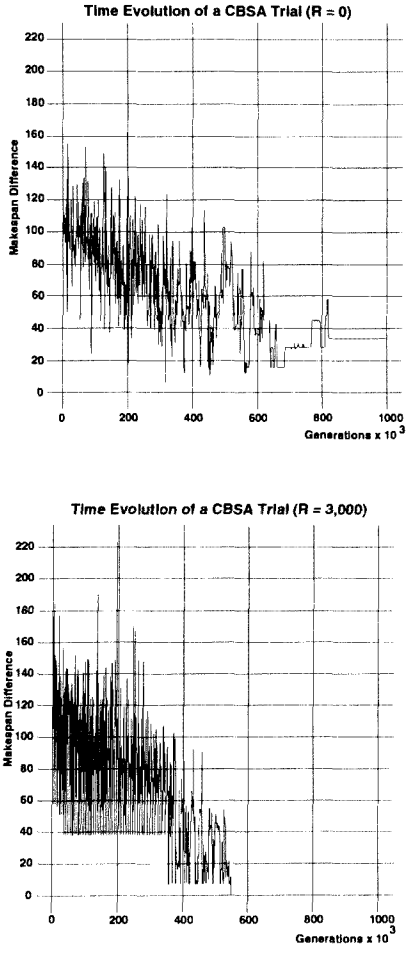


図 3　再重点化戦略あり ($R = 3,000$) と, 再重点化戦略なしにおける解の時間的推移の様子

Fig. 3. Time evolutions of the two trials: with reintensification ($R = 3,000$) and without reintensification.

図 4 は初期温度および総遷移回数を小さな値 ($T_0 = 0.01$, $k_f = 1,000$) に設定した CBSA 法および ASSA 法を, それぞれランダムな初期値を用いて 10,000 回の実験を行なった結果得られた解の分布である. このような条件下では解の改善する方向への遷移のみ受理するため欲張り (*greedy*) 探索となり, 短時間に局所解に陥って定常状態になる. 二つの解の分布を比較すると明らかに CB 近傍を用いた場合の方が優れている. これは近傍構造そのものの能力の違いによるものと思われる. なお得られた解の最小値・最大値はそれぞれ CB 近傍を用いた場合が 944, 1425, AS 近傍の場合が 971, 1491, 所要時間はともに 1 時間 40 分ほどであった.

本 CBSA 法を [17] に掲載されている 10 の難問に適用した結果を表 2 の CBSA 欄に示す. ただし, 各問題に対して 5 回の実験を行なった. また比較のために文献 [5], [4], [17], [7] による結果をそれぞれ Laar, Mats, Appl, Tail の欄に, 未解決の問題に対しては文献 [17] に記載されている理論的下界を LB の欄に付す. 再重点化頻度 R は先に MT10×10 問題において最も成績の良かった $R = 3,000$ を用いた. 本来はこの値も問題の特性に合わせて適応的に定めるべきである.

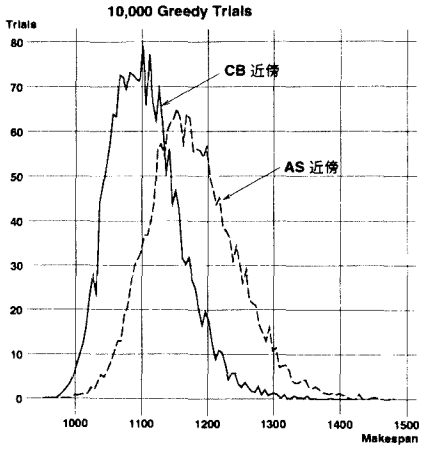


図 4　10,000 回の欲張り探索法により得られた (MT10×10) スケジュールの分布.

Fig. 4. Generated Makespans of 10,000 *Greedy* (MT10×10) Schedules.

5. 結び

本論文ではクリティカルブロック法に基づく 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 (CBSA 法) を提案した.

Aarts 等によれば, 近似解法が成功するか否かはアルゴリズムの性能, 実装の容易さ, 適応性および柔軟性にかかっている [15]. 一般にアニーリング法はアルゴリズムが単純で実装は容易である. CBSA 法ではアニーリング法が持つこれらの利点を生かしつつ従来の SA 解法である ASSA 法とほぼ同等の時間内に, より優れた品質の解を求めることができる. これは AS 近傍を用いた場合一回の遷移の結果生ずる解の変化が比較的小さいのに対し, CB 近傍は一回の遷移によ

表 3　ジョブショップスケジューリング問題の 10 の難問
Table 3. Ten difficult Benchmark Job Shop Scheduling Problems.

問題	$n \times m$	LB	CBSA 法				ASSA 法		参考	
			best	mean	std.	Time	Laar	Mats	Appl	Tail
abz7	20×15	654	671	679.00	6.57	3 h 35 m 25 s			668	665
abz8	20×15	635	679	689.20	7.36	3 h 58 m 34 s			687	676
abz9	20×15	656	701	712.00	6.07	3 h 39 m 57 s			707	691
la21	15×10	1040	1050	1054.60	3.01	1 h 7 m 2 s	1063	1071	1053	1047
la24	15×10	—	943	947.20	3.31	48 m 1 s	952	973	935	
la25	15×10	—	985	987.80	2.40	1 h 11 m 16 s	992	991	977	
la27	20×10	1235	1262	1266.00	2.76	2 h 7 m 2 s	1269	1274	1269	1240
la29	20×10	1120	1188	1193.20	4.66	2 h 31 m 10 s	1203	1196	1195	1170
la38	15×15	1184	1209	1214.00	2.76	2 h 3 m 59 s	1215	1231	1209	1202
la40	15×15	—	1235	1248.80	11.48	1 h 29 m 25 s	1234	1235	1222	

best: 最良解, mean: 平均, std.: 標準偏差, Time: 平均所要時間
(いずれも 5 回行なった実験での値)

って, 総作業時間を減少させる遷移をより多く (少なくとも同数) 含む, より大きな変化に対応する近傍となっているからであろう.

さらに, 再重点化戦略を用いることによってそれまでに得られた最良の解の周辺を重点的に探索する結果, 探索の効率を上げることができる.

MT 10×10 問題に関しては Applegate 等の提案する *shifting bottleneck* 法, *shuffle* 法を組み合わせた解法を用いることによって, 数秒から数十秒で最適解を求めることができる. したがってこの問題に関してはこれらの解法を用いる方が有利である. また, 表 3 の各問題に対する解の質に関しては, 彼らの方法とほぼ同等の結果を得ているものの彼らの方法は 1 時間程度であるのに対して CBSA 法は問題の規模に応じて 1 時間弱から 4 時間弱程度かかり, 速度の点では及ばない. しかし彼らの解法は問題の特性により依存しており, 実装も容易でない.

Taillard は AS 近傍に基づくタブー探索法を用いて短時間に, より優れた結果を得ているが [7], 本方法のように CB 近傍を用いる方法も有望であろう.

今後は, 遺伝的アルゴリズムの global sampling の能力との組み合わせなどによる能力向上を目指したい.

謝辞

本論文を丁寧に査読し, 貴重な情報およびコメントを与えて頂いた査読者に感謝いたします.

(平成 5 年 9 月 3 日受付, 同 5 年 11 月 24 日再受付)

文　献

[1] J.F. Muth and G.L. Thompson. *Industrial Scheduling*. Prentice-Hall, Englewood Cliffs, N.J., 1963.

[2] S. S. Panwalkar and Wafix Iskander. A survey of scheduling rules. *Oper. Res.*, 25(1):45–61, 1977.

[3] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Mgmt. Sci.*, 34(3):391–401, 1988.

[4] H. Matsuo, C.J. Suh, and R.S. Sullivan. A controlled search simulated annealing method for the general job-shop scheduling problem. *Working Paper, 03-04-88, Department of Management, The University of Texas at Austin*, 1988.

[5] P.J.M. van Laarhoven, E.H.L. Aarts, and J.K. Lenstra. Job shop scheduling by simulated annealing. *Oper. Res.*, 40(1):113–125, 1992.

[6] T. Yamada and R. Nakano. A genetic algorithm applicable to large-scale job-shop problems. In *Parallel Problem Solving from Nature (Brussels,Belgium)*, 1992.

[7] E. Taillard. Parallel taboo search technique for the job shop scheduling problem. *ORWP 89/11(Rev. 92/10), DMA, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*, 1989.

[8] E. Balas. Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Oper. Res.*, 17:941–957, 1969.

[9] J. Grabowski, E. Nowicki, and S. Zdrzalka. A block approach for single machine scheduling with release dates and due dates. *E. J. of Oper. Res.*, 26:278–285, 1986.

[10] P. Brucker, B. Jurisch, and B. Sievers. A branch & bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics (to appear).*

[11] B. Giffler and G.L. Thompson. Algorithms for solving production scheduling problems. *Oper. Res.*, 8:487–503, 1960.

[12] P. D. Wasserman. *Neural Computing: Theory and Practice*. Van Nostrand Reinhold, New York, 1989.

[13] L. Ingber. Very fast simulated re-annealing. *Mathl. Comput. Modeling*, 12(8):967–973, 1989.

[14] B. E. Rosen. Function optimization based on advanced simulated annealing. *Workshop on Physics and Computation, PhysComp 92*, 1992.

[15] E.H.L. Aarts and J.H.M. Korst. *Simulated Annealing and Boltzmann machines*. Wiley, Chichester, 1989.

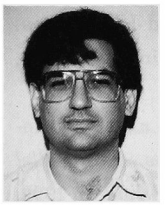
[16] D.S. Johnson, C.R. Aragon, L.A. Mcgeoch, and

C. Schevon. Optimization by simulated annealing: an experimental evaluation: part II, graph coloring and number partitioning. *Oper. Res.*, 39(3):378–406, 1991.

[17] D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *E. J. of Oper. Res.*, 3(2):149–156, 1991.

山田　武士　1964年10月2日生まれ. 83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.

Bruce E. Rosen　Bruce E. Rosen was born January 5, 1959. The author received his B.S. in Math / Computer Science and Economics in 1982 and Ph.D. in Computer Science in 1991 from the University of California, and currently is Assistant Professor of Computer Science at the University of Texas at San Antonio.



中野　良平　1947年生まれ. 1971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, 人工知能の研究開発に従事. 現在, NTTコミュニケーション科学研究所中野研究グループリーダ. 主幹研究員. 工博.