人　工　知　能　76－4
（1991．5．22）

データ依存関係を用いた組合わせ最適化手法

原 裕貴, 湯上 伸弘, 吉田 裕之
(株) 富士通研究所
hara@flab.fujitsu.co.jp

概要

仮定に基づく組み合わせ最適化システム (Assumption based Combinatorial Optimizaion System) を提案する．本手法では, 目的関数に関する極小支持 (minimal support) という概念を用いることによって効率的に近傍探索を行う. 極小支持を利用することにより, 近傍の数を大幅に減らすことができる． また探索の枝刈り, ループチェックを効率的に行うことができ, 局所最適の問題が生じない．ジョブショップスケジューリング問題に対して適用し, 反復改善法や分枝限定法と比較して, その有効性を実証した.

Assumption based Combinatorial Optimization System

Hirotaka Hara, Nobuhiro Yugami, Hiroyuki Yoshida
Fujitsu Laboratories LTD.
1015, Kamikodanaka Nakahara-ku Kawasaki 211, Japan
hara@flab.fujitsu.co.jp

Abstract

In this paper, We propose Assumption based Combinatorial Optimization System. It is one of local search methods. It uses the minimal support assumption set of the objective function. The feature of it is that: (1) it reduces the number of neighbors. (2) it performs loop check and pruning effectively, (3) it never stop at a local minimal solution. We apply the system to the jobshop scheduling problems and evaluate our system comparing with the iterative improve method and the branch and bound method.

1 まえがき

組み合わせ最適化は, 巡回セールスマン問題, ナップザック問題, スケジューリング問題, 最適配置問題等広い応用範囲を持つ重要な問題であり, 古くからORの分野で研究されてきた. 厳密な解法としては整数計画法, 分枝限定法などが用いられるが, 組み合わせ最適化は一般にNP-completeの難しい問題であることが多いため, 近傍探索法や緩和法などの各種の近似的手法が開発されている.

一方, AIの分野でも組み合わせ的な問題は古くから扱われており, TMS [1] やATMS [4] もそのような手法の一つと考えることができる. 近年では, このような問題がCSP [5] として定式化され, 各種の手法が研究されている [6] [8]. しかし, AIの分野では制約を満たす解を求めることが主な目的でであり, 最適化を明に扱うことはなかった.

本稿では, 組み合わせ最適化問題に対して, 仮定に基づく組み合わせ最適化システム (Assumption based Combinatorial Optimization System) を提案する. 本手法では, 目的関数に関する極小支持 (minimal support) という概念を用いることによって効率的に近傍探索を行う. ジョブショップスケジューリング問題に適用した結果についても報告する.

2 近傍探索法

近傍探索法 (local search method) は, なんらかの方法で得られた近似最適解に対して, その近傍, つまり少し変更を加えることで得られる他の解を調べて置き換えていく方法である. 特に目的関数が改善される場合のみに置き換えを行う場合を反復改善法または山登り法と呼び, 広く用いられる. 巡回セールスマン問題におけるLK法が有名である.

反復改善法の問題点としては次のようなことがある. まず第1に, 問題のサイズが大きくなると近傍の数も増えて解が改善される近傍を捜す効率が悪くなることである. 第2に, 近傍により良い解が存在しない, いわゆる局所最適解に陥る可能性があることである. これらの問題点は近傍の定義に依存する. 効率をあげるためには近傍の数を減らせば良いが, それはまた局所最適解を増大させることにつながるため, この二つの問題は相反する課題である.

近年, 近傍探索法のひとつとしてしてシミュレーテッドアニーリング法 [13] が注目されている. これは, 確率的に目的関数が悪くなる方向への変更も許す探索法で, この確率をうまく制御することによって局所最適解から脱出する手法である. シミュレーテッドアニーリングの問題点としては収束するまでに非常に時間がかかることがあげられている. 逆に収束のスピードをあげようとすると局所最適の問題が生じる. そういった意味では反復改善法の問題点を本質的には解決していないともいえる.

これらの手法の問題点は, いずれも探索の制御を目的関数のみによって行っていることである. このため手法が汎用的である反面, 問題固有の性質や構造を生かすことができない.

3 仮定に基づく組み合わせ最適化システム

我々のアイデアは, 組み合わせ最適化問題を仮定 (assumption) によって定式化し, 目的関数の値を決定する最小の仮定の集合である極小支持 (minimal support) の概念を利用することである. この手法の利点は二点ある.

まず第一に, 極小支持を利用することによって, 近傍探索法における近傍の数を大幅に減らすことができ, 探索が効率化される.

第二に, 極小支持を利用することにより, 探索の枝刈り, ループチェックを効率的に行うことができる. したがって, 反復改善法やアニーリングのように目的関数によって動きを制御する必要がなく (目的関数の値が悪くなる方向に変更しても構わない), 局所最適の問題を自然な形で

解決することができる.

まず, 解は仮定 (assumption) の集合として表現されるものとする. たとえば, 巡回セールスマン問題の場合, ある都市からある都市へのパスをはることがひとつの仮定となる. すなわち目的関数を最適にする仮定の集合を見つけることが目的となる. 以下では解 S といえば, 仮定の集合を意味するものとする.

システムは変数や目的関数の値に対する極小支持 (minimal support) を計算, 保持しながら探索を進める. 極小支持は仮定の集合から成り, 変数の値を保証する極小の仮説の集合である. 極小支持は, 解を構成する仮定の集合の部分集合となる.

極小支持の計算は TMS や ATMS と同様に行う. 以下ではデータとその極小支持を [data, minimal support] と記述する. 以下に極小支持計算の例をあげる.

例 1 $[x = 3, X], [y = 2, Y], z = x + y \Rightarrow [z = 5, X \cup Y]$

例 2 $[x \geq 3, X], [y = 2, Y], z = \max(x, y) \Rightarrow [z \geq 3, X]$

解 S における目的関数値に対する極小支持のことを, 解 S の極小支持と呼ぶ. 本システムでは, 生成された解の極小支持は NG 集合に蓄えられ, 解の更新のチェックに用いられる. この機構は, 探索のループを防ぐとともに, 枝刈りの機能も持つ. システムの動作アルゴリズムを以下に示す.

アルゴリズム

1. NG 集合を空とする.
2. 初期解を生成し, S とする.
3. S の目的関数値およびその極小支持 MS を計算する.
4. MS を NG 集合に追加する.
5. S に変更を加えて $MS \subseteq S'$ ではない新しい解 S' を生成する.
6. NG 集合中に S' の部分集合となるような NG が存在すれば, ステップ 5 に.
7. S := S'
8. ステップ 3 に.

ステップ 5 においては, 条件から変更すべき仮定を MS の中から選ぶことになる. これによって, 探索する近傍の数を減らすことができる. 変更した結果がステップ 6 で受理されない場合は, ステップ 5 に戻って別の変更をためすことになる. どのような変更を先に行うかは全体の効率に重要な影響を与え, 適切なヒューリスティクスを用いることになる.

どのようなヒューリスティクスが有効であるかについては, 問題に依存するが, 必ずしも解全体を改善する (すなわち目的関数を改善する) 必要はなく, 現在の極小支持の中の一部分が改善されることを保証するようなヒューリスティクスであれば良いと考えられる.

ステップ 6 では解の変更を受理するか否かのチェックが行われるが, ここで目的関数値によるチェックは行われない. その代わりに, NG 集合 (今までに生成された極小支持の集合) とのチェッ

クが行われる．これによって，同じ解を生成してしまうループを防ぐとともに，探索の効果的な枝刈りが行われる．なぜなら，極小支持は解の部分集合であるから，以前生成された解と全く同じ状態ではなくても，同じ極小支持を含むような解は受理されないからである．

4 ジョブショップスケジューリング問題

この章では，組み合わせ最適化問題の例として，ジョブショップスケジューリング問題をとりあげ，我々の手法の適用方法を示す．

ジョブショップスケジューリング問題は，n個のジョブを m個の機械に割り当てる問題である．各ジョブはm個の作業からなりたっていて，各機械を一回づつ利用する．利用する機械の順番は各ジョブごとに決まっている．機械sを利用するジョブjのk番目の作業を O_{jks} と記述する．

スケジューリングとは，各機械における作業順序を決定することである．表1，表2にジョブショップスケジューリング問題の例をあげる．この例は文献 [12] よりとりあげたものである．

表 1: ジョブショップ問題の例 (使用機械)

作業	*1*	*2*	*3*
ジョブ1	1	2	3
ジョブ2	3	1	2
ジョブ3	2	3	1

表 2: ジョブショップ問題の例 (処理時間)

作業	*1*	*2*	*3*
ジョブ1	5	8	2
ジョブ2	7	3	9
ジョブ3	4	11	7

この問題に対する一つのスケジュールの例を図1 [12] に示す．

本手法を適用するためには，まず仮定を定義する必要がある．スケジューリング問題に対しては同一機械で連続して実行される作業をひとつの仮定とすることができる．作業O'の次に作業Oを実行するという仮定を $O' \prec O$ と記述する．図1の例は，次のような仮定の集合から構成されている．

$$\begin{aligned}\{\quad & O_{111} \prec O_{311}, O_{311} \prec O_{221}, O_{221} \prec O_{431}, \\ & O_{412} \prec O_{122}, O_{122} \prec O_{232}, O_{232} \prec O_{332}, \\ & O_{213} \prec O_{133}, O_{133} \prec O_{423}, O_{423} \prec O_{323}\}\end{aligned} \tag{1}$$

ジョブショップスケジューリングにおける目的関数としては，総所要時間，重み付き終了時刻和，最大遅れなどが用いられるが，本稿では総所要時間を目的関数とする．作業 O_{jks} の実行時間を e_{jks}，終了時刻を x_{jks} と表すと，総所要時間最小という目的関数は次のように記述できる．

$$\min \quad z$$

$$z \geqq \max_{1 \leq j \leq n} x_{jms}$$

また, 各作業の開始時刻は, その作業の先行作業の終了時刻と機械 s における直前の作業の終了時刻の大きい時刻になる.

$$x_{jks} = \max(x_{j(k-1)s'}, x'_{jks}) + e_{jks}$$

ただし, ここで x'_{jks} は $O' \prec O_{jks}$ なる作業 O' の終了時刻である. このような O' が存在しない (すなわち, O が機械 s で最初に実行される作業である) 場合は $x'_{jks} = 0$ とする. 同様に, $k = 1$(すなわち, O がジョブ j における最初の作業である) 場合は $x_{j0s'} = 0$ とする.

このように定式化すると, 解に対する極小支持を計算することができる. 図 1 の解に対する極小支持計算の一部を以下に示す.

$$[x_{111} \geq 5, \{\}], [x'_{122} \geq 4, \{O_{412} \prec O_{122}\}], x_{122} = \max(x_{111}, x'_{122}) + e_{122}$$
$$\Rightarrow [x_{122} \geq 13, \{\}]$$
$$[x_{122} \geq 13, \{\}], [x'_{133} \geq 7, \{O_{213} \prec O_{133}\}], x_{133} = \max(x_{122}, x'_{133}) + e_{133}$$
$$\Rightarrow [x_{133} \geq 15, \{\}]$$
$$[x_{412} \geq 4, \{\}], [x'_{423} \geq 15, \{O_{133} \prec O_{423}\}], x_{423} = \max(x_{412}, x'_{423}) + e_{423}$$
$$\Rightarrow [x_{423} \geq 26, \{O_{133} \prec O_{423}\}]$$
$$[x_{311} \geq 6, \{O_{111} \prec O_{311}\}], [x'_{323} \geq 26, \{O_{133} \prec O_{423}, O_{423} \prec O_{323}\}],$$
$$x_{323} = \max(x_{311}, x'_{323}) + e_{323}$$
$$\Rightarrow [x_{323} \geq 33, \{O_{133} \prec O_{423}, O_{423} \prec O_{323}\}]$$
$$[x_{323} \geq 33, \{O_{133} \prec O_{423}, O_{423} \prec O_{323}\}],$$
$$[x'_{332} \geq 22, \{O_{122} \prec O_{232}, O_{232} \prec O_{332}\}], x_{332} = \max(x_{323}, x'_{332}) + e_{332}$$
$$\Rightarrow x_{332} \geq 43, \{O_{133} \prec O_{423}, O_{423} \prec O_{323}\}]$$
$$[x_{133} \geq 15, \{\}], [x_{232} \geq 22, \{O_{122} \prec O_{232}\}],$$
$$[x_{332} \geq 43, \{O_{133} \prec O_{423}, O_{423} \prec O_{323}\}], [x_{431} \geq 33, \{O_{133} \prec O_{423}\},$$
$$z = \max(x_{133}, x_{232}, x_{332}, x_{431})$$
$$\Rightarrow [z \geq 43, \{O_{133} \prec O_{423}, O_{423} \prec O_{323}\}]$$

したがって, この解の極小支持は $\{O_{133} \prec O_{423}, O_{423} \prec O_{323}\}$ となる. これは, もちろん解を構成する集合 (1) の部分集合になっている. この極小支持は, これらの仮定のうち少なくとも一つを変更しないと, 解が改善されることはないことを保証している.

さて, ジョブショップスケジューリング問題に対して近傍探索法を適用する場合, 近傍の定義として最も単純なものは, 作業をひとつ選んで別の位置に移動することである. 本稿でも以下ではこの定義を採用する. この場合, 取り出す作業の総数が mn, 移動場所が n で, 近傍の数は mn^2 になる.

しかし, このような変更のうち, 極小支持中の仮定を変更しないような変更によっては解が改善されないことが保証されている訳であるから, 極小支持の情報を用いることによって無駄な近

傍探索を大幅に押さえることができる．作業の総数 mn に対して，極小支持に含まれる作業の数は最大でも n であり，通常，これよりも少ない．

以上のように本手法では，極小支持の中から仮定を選んで変更を行う．例えば，O_{423} と O_{323} の順序を変更する．これによって，$O_{423} \prec O_{323}$ という仮定は成立しなくなるので，上の極小支持も成り立たなくなる．この新しい状態を図 2 に示す．目的関数も改善されていることが分かる．

しかし，このような変更によって目的関数が改善されるとは限らない．反復改善法ではそのような変更は受理されないが，本手法では NG 集合とのチェックに成功すれば変更を行う．

第 3 章で述べたように，変更に際してどのような変更から先に試すかというヒューリスティクスが効率に重要な影響を与える．我々がジョブショップスケジューリング問題において用いたヒューリスティスクは，ある機械における一連の仮定における最後の作業を早めるような変更を優先するというものである．図 1 においては，O_{323} を早めるような変更である．なぜなら，この変更によって O_{332} の開始時刻が早まることが保証されているからである．ただし，この変更によっていろいろな影響がでるわけであるから (たとえば O_{423} の開始時刻は遅くなる)，全体として改善されることが保証されている訳ではない．しかし，このように局所的に改善が保証されたヒューリスティックを本手法に対して用いることによって，第 6 章で述べるように良い結果を得ることができた．

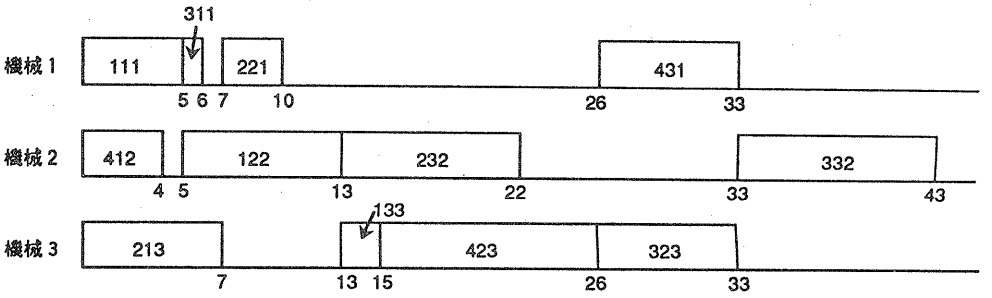

図 1．スケジュールの例

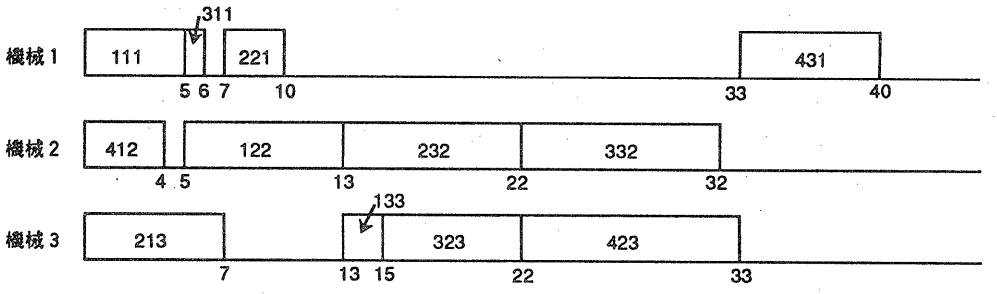

図 2．変更後のスケジュール

5　ジョブショップスケジューリング問題に対する各種解法

ジョブショップスケジューリング問題は，フローショップ問題などのスケジューリング問題の中でも最も一般的で難しい問題であり，NP-complete であることが知られている．本章では，ジョブショップスケジューリング問題に対するこれまでの厳密解法，近似解法について簡単に述べる．

5.1 整数計画法

ジョブショップスケジューリング問題は, 整数計画問題として定式化できる [12]. この場合, 同一機械における任意の二つの作業のペアに対してどちらを先に行うか? という選択がひとつの整数変数になるので, m機械nジョブの場合, 整数変数の数は, $mn(n-1)$ にもなる. このため, ジョブ数が大きくなると厳密解を求めるのは難しい. 我々がSun上で試作したプログラムでも, 5機械5ジョブの場合で完了までに3時間以上かかり, これ以上大きな問題に対しては完了することができなかった.

5.2 分枝限定法

ジョブショップスケジューリング問題の最適解は, アクティブスケジュールであることが知られている. アクティブスケジュールとは, それぞれの機械における作業順序をかえることを含めて, いずれの作業もそれ以上早く開始することが不可能なスケジューリングである. 図1の例はアクティブスケジュールである. すべてのアクティブスケジュールを生成するアルゴリズムが提案されており, これに対して分枝限定法を用いることができる [12]. この手法の詳細についてはここでは述べないが, 基本的には前から順番に配置可能なジョブを配置していく手法である. 機械やジョブに着目して残っている作業の和を求めて下界値を求め, 限定操作が行われる. この手法でも, ジョブの数の増加とともに探索範囲は大幅に増え, 完了まで実行することは不可能となるが, 打ち切って近似解として用いることはできる. 第6章の測定では, 深さ優先探索を用いた分枝限定法の結果についても述べる.

5.3 近似解法

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はいくつも提案されているが, いずれもなんらかのヒューリスティクスを用いて作業を順に決定的に配置していくものである [10]. 反復改善法などの近傍探索法を利用した報告は発見できなかった. このような近似解法のひとつで簡単なものは, 開始可能時刻が早いものから順に配置していくというものである. 第6章ではこの手法で求まった解を初期解として, 我々の手法を適用した結果についても述べる.

6 評価

ジョブショップスケジューリング問題を対象として, (1) 単純な反復改善法, (2) 極小支持を利用した反復改善法, (3) 本手法 (4) 分枝限定法の比較を行った. (2) の極小支持を利用した反復改善法は, 近傍探索の際に変更する作業を極小支持の中から選ぶようにしたものである. 以下に測定の条件を述べる.

- 測定は, Sun4/110上のQuintus Prologを用いて行われた. 測定における時間の単位は全て秒である.
- 評価で用いられるデータにおける作業の大きさは乱数を用いて, ランダムに生成されたものである. 最小値を2, 最大値を10としている.

まず, 5機械10ジョブ50作業問題と10機械20ジョブ200作業問題に対して反復改善法, 反復改善法+極小支持および本手法を適用した. 初期解はランダムに生成された解を共通に用いている (表3, 表4).

改善率は測定中見つかった最も良い解の初期解に対する改善率であり, 変更回数はそれまでの変更の回数, CPU時間1はそれまでに要した時間である. CPU時間2は全測定時間である.

表 3: 5 機械 10 ジョブ 50 作業問題

手法	変更回数	改善率	*CPU*時間 *1*	*CPU*時間 *2*
反復改善法	3	6%	140	7200
反復改善法 + 極小支持	3	6%	21	201
新手法	350	29%	426	7200

表 4: 10 機械 20 ジョブ 200 作業問題

手法	変更回数	改善率	*CPU*時間 *1*	*CPU*時間 *2*
反復改善法	0	0%	0	7200
反復改善法 + 極小支持	2	2%	1392	7200
新手法	131	22%	3026	7200

50 作業問題に対しては, 反復改善法に極小支持を利用することによって効率が良くなっているが, 改善が 3 回起きたところで局所最適解に陥って探索が終了してしまった.

問題の規模が大きくなり 200 作業問題になると, 反復改善法の効率は非常に悪くなり 2 時間動かしても改善が起きなかった. 極小支持を用いても 2 時間後の改善率は 2% に留まった.

これに対して, 本手法はどちらの問題に対しても 20% 以上の改善が行われた. 問題の規模の拡大に対して効率, 改善率ともにあまり悪化しないのが特徴である. また, 探索の初期に良い解が早く見つかるのも特徴であり, 200 作業問題の例では, 80 秒で 9%, 309 秒で 16% の改善が達成されている.

次に最早開始可能時刻順に配置する近似手法によって求められた解を初期解として用いた場合について測定を行った. ここでは同時に深さ優先の分枝限定法との比較も行った. 分枝の際に最早開始可能時刻優先のヒューリスティクスをを用いたので, 最初に求まる解は同一の解になる. なお, ここで用いた問題は表 4 における問題と同一の問題である (表 5).

表 5: 10 機械 20 ジョブ 200 作業問題 (近似初期解利用)

目的関数値	*137*	*136*	*135*	*134*	*133*	*132*	*124*	*finish*
分枝限定法	47	64	123	144	318	-	-	7200
反復改善法	18	-	-	-	-	-	-	7200
反復改善法 + 極小限定	18	34	170	452	617	1183	-	3467
新手法	18	24	30	37	54	103	3191	7200

横軸は具体的な目的関数値である. 表の中の数値は, その目的関数値が求まるまでの時間を示している. 横軸最後の finish は測定終了時刻である. 反復改善法では改良が起きなかった. 反復改善法 + 極小支持では 5 回の改善が起きた後, 局所最適解に陥って終了した. 分枝限定法は 133 という解が求められた後, それ以上の解を 2 時間以内には発見できなかった.

これに対して, 本手法では 1 分以内で 132 という解を発見し, さらに 1 時間以内で 124 という非常に良い解を発見することができた. ちなみに, 同じ問題に対して, 初期解生成に近似手法を用いなかった表 4 の測定では, 初期解が 169 で発見された最も良い解が 131 であった. 本手法を良

い初期解生成手法と組み合わせることによって，性能が改善されることがわかった．

図３にこの問題における本手法の目的関数の推移を示す．横軸は変更の回数，縦軸は初期値を１としたときの目的関数の比率である．

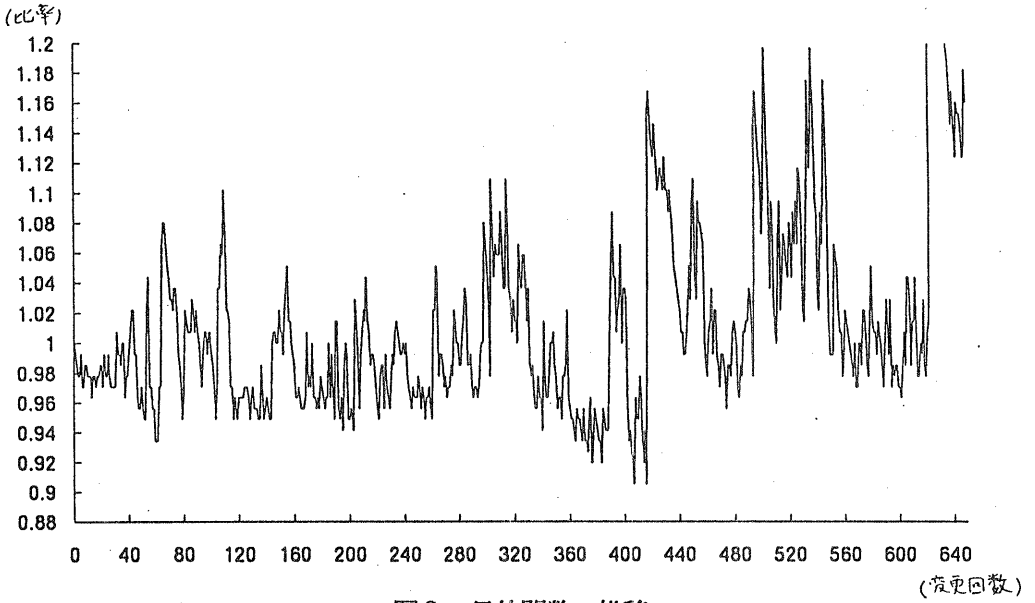

図３．目的関数の推移

7　考察

7.1　極小支持

ジョブショップスケジューリング問題に対しては本手法は非常に良好な結果を導くことができた．しかし，本手法がすべての組み合わせ最適化問題に対して有効なわけではない．本手法が有効かどうかは，極小支持が解を表現する仮定に対してどの程度小さく押さえられるかに依存する．たとえば，巡回セールスマン問題のような場合，極小支持は解全体となってしまう．

一般的に，どのような問題に対して，仮説をどのように定義することによって，極小支持が小さくなるかという問題については，さらに研究が必要である．

7.2　インプリメンテーション

最初に本シテスムを作成したときに，効率上のネックになるのはNG集合の増加によるNGチェックであろうと予測し，極小支持における仮定をソートしておきNG集合全体を木状に管理した．このためか，NGチェックのオーバヘッドはほとんど問題にならなかった．NGチェックによって不受理が起きた回数は全改善数のほぼ半分であって，この割合は探索中にほとんど変わらなかった．

むしろ，効率上の問題となったのは変更を行った際に次の解の目的関数値や極小支持を計算する部分であった．現在は，新しい解が生成されるごとにすべて計算しなおしているため効率が悪い．変更前の解の情報を用いて部分的に変更をすることによってさらに高速化できると考えられる．その際にはreteネットなどの技術が使えるであろう．

8 おわりに

本手法は, (1) 規模の大きな問題に対しても適用可能である, (2) 良い解が早い時期に求められる, (3) 近似解を初期解としてさらに良い解を求められる, などの利点があり, 現実的な利用価値の高い手法であると考える. 今回の報告では, 十分な測定を行えなかったので, 現在さらに詳細な測定を行い, 本手法の有効性の確認を行っている.

今後は, より現実的な問題への適用を進めるとともに, 極小支持の計算法や現在ヒューリスティクスを用いている変更の仕方などに関して, さらに理論的な研究を行う予定である.

参考文献

[1] J. Doyle. A Truth Maintenance System. Artificial Intelligence, Vol.12, 1979.

[2] M. S. Fox, N. Sadeh, and C. Baykan. Constrained heuristic search. In Proceedings IJCAI-89, 1989.

[3] M. S. Fox. ISIS: A Constraint-directed Reasoning Approach to Job Shop Scheduling. Technical Report, Carnegie-Mellon University, 1983.

[4] J. de Kleer. An Assumption-based TMS. Artificial Intelligence, Vol.28 No.2, 1986.

[5] A. K. Mackworth, E. C. Frender. The complexity of some polynomial network consistency algorithms for constraint satisfaction problems. Artificial Intelligence, Vol.25(1), 1985.

[6] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method. In Proceedings AAAI-90, 1990.

[7] R. Reiter and J. de Kleer. Foundations of Assumption based Truth Maintenace System. In proceedings AAAI-87, 1987.

[8] M. Zweben and M. Eskey. Constraint satisfaction with delayed evaluation. In Proceedings IJCAI-89, 1989.

[9] 茨木. 探索の高速化とその限界. 人工知能学会誌 Vol6. No.1, 1991.

[10] 石井. スケジューリング問題の近似解法. オペレーションズリサーチ, 31, 1986.

[11] 奥野. ATMS の高速化技法とその応用. 人工知能学会誌 Vol6. No.1, 1991.

[12] 今野, 鈴木. 整数計画法と組み合わせ最適化. 日科技連.

[13] 中野, 中西. 組み合わせ最適化に対する Simulated Annealing 法. オペレーションズリサーチ, 31, 1986.

[14] 西川, 三宮, 茨木. 最適化. 岩波講座情報科学 19, 1982.

[15] 湯上, 原, 吉田. スケジューリング問題への ATMS の適用. 情報処理学会研究報告, 90-AI-72, 1990.