人　工　知　能 90－8
（１９９３．９．２８）

Simulated Annealingによる大規模生産計画問題の解法

湯上　伸弘　　原　裕貴

富士通研究所

Simulated Annealing（SA）は、非常に強力な組合せ最適化問題の解法であるが、現実の問題への適用を考えるといつくかの問題点がある。ひとつは問題解決に比較的長時間必要な点である。これは大規模な問題を比較的短時間で解かなくてはならないような場合に問題となる。もうひとつの問題点は、目的関数を一つしか扱うことができないという点である。現実の組合せ最適化問題はほんどの場合複数の評価基準を持ち、それら全てをバランス良く準最適化する必要がある。本論文ではSAのこれらの短所を克服するために、高速に解の評価を行うための伝播アルゴリズムVP2およびSAの多目的最適化問題への拡張であるSA/Mを提案し、その効果を工場の操業計画問題を使って検証する。我々は、SA/MとVP2を用いることにより、専門家が２日かけて立案する計画よりも良い操業計画を約４０分で立案することができた。

Solving a Large-Scale Production Scheduling by Extended Simulated Annealing

Nobuhiro Yugami　　Hirotaka Hara

Fujitsu Laboratories

Simulated Annealing (SA) is one of the best methods for combinatorial optimization problems. However, SA has some difficulties for solving practical problems. One is that SA needs relatively a long time for problem solving. This makes it difficult to apply SA to large-scale problems. Another difficulty is that SA can only deal with one objective function. In practical problems, there are many criteria for estimating one solution and we must obtain a solution which is good for all criteria. To resolve these difficulties, we propose two methods. One is VP2 which calculates the values of objective functions by using dependency between variables. Another is SA/M which is an extension of SA for optimization problems with multiple objectives. We applied VP2 and SA/M to a large-scale production scheduling problem. We could get a good solution in 40 minutes which was better than that of an expert who solved in 2 days.

1　はじめに

本論文では、工場の操業計画問題に代表されるような、大規模な組合せ最適化問題にシミュレーテッドアニーリング法（SA）[1,2]を適用する際の問題点とその解決手段について議論する。

近年計画問題や設計問題などの合成型問題解決システムへのニーズが高まっている。しかし、殆どの合成型問題は組合せ最適化問題の一種であり、NP完全ないしはそれ以上の難易度を持つため、問題の規模が極小規模な場合を除いては厳密な最適解を求めることはできない。そのため、現実的な時間で良い解を求めるための近似解法が重要となる。組合せ最適化問題の近似解法は様々な手法が提案されているが[3,4]、SAは、山登り法とは異なり局所最適解からの脱出が可能であること、また、負荷が計算量の面から見ても必要なメモリ量の面から見ても軽いことなどのの点で優れている。しかし問題点もある。ひとつは問題解決に必要な時間が比較的長いという点である。実際的な問題では頻繁に条件が変更されるような問題も多く、特に大規模な問題を解くさいに十分な時間を割くことができない場合がある。もうひとつの問題点は、目的関数をひとつしか扱えないということである。現実的な問題では、解の評価基準（目的関数）は複数あるのが普通であり、全ての評価基準にたいして良い値となるようなバランスのとれた解を求める必要がある。このような多目的最適化問題をSAで扱うためには全ての目的関数を線型結合等によって合成して仮の目的関数をつくり、この仮の目的関数を最適化することになる。ところが、このような目的関数のもとでは、ある目的関数の値は悪いけれども、他の目的関数の値が非常に良いような解の評価値が非常に良くなりうるので、結果としてバランスの悪い解が得られる可能性がある。

この論文ではこれらの問題点を解決するためにいつくかの提案を行う。まず問題解決の速度に関しては、2つの方法により改善をおこなう。ひとつは近傍を生成する際に適当なルールを用いることにより（準）最適解に到達するまでに調べる解の個数を減らすことである。もうひとつは解の評価、すなわち目的関数値を計算する際に変数間の依存関係を利用した再計算アルゴリズムVP2により、ひとつの解を評価するのに必要な時間を減らすことである。

また、SAを複数の目的関数を同時に扱えるように拡張したSA/Mを提案する。SA/Mでは、ある解からその近傍への遷移確率の計算方法を目的関数が複数ある場合に拡張することにより多目的最適化問題を解く。SA/Mでは、遷移確率を、他の目的関数値がどれだけ改善されても、ある目的関数が悪化する場合には小さな値をとるように定義することによりバランスの悪い解が得られるのを避ける。

これらの手法の効果を、工場の月間の操業計画問題に適用することにより検証する。この操業計画問題は、機械の数が約７０台で製品数が約２００であるような、単一の問題としてはかなり大規模な最適化問題である。

2　生産計画問題

この論文で扱うのは、化学繊維工場の月間の操業計画問題である。計画は一日単位で立案される。すなわち、計画立案とは各機械が各日に何を生産するかを決めることである。この工場では、月に一度各製品毎の生産要求が与えられる。この要求はほとんどの場合実際の生産能力を越えているため、要求からの不足量を最小とするような計画を立案することが目的である。

図１はこの工場における生産の流れである。この工場の製品は、基本的には２つの工程を経て生産されるが、第１工程のみで出荷される製品もある。以下では、実際に出荷される製品を最終製品、最終製品を生産するために必要となる製品（第１工程における生産物）を中間製品と呼ぶ。ひとつの機械が生産可能な製品の集合は機械毎に異なる。機械の１日当りの生産量は機械と製品の両方に依存する。すなわち、同じ製品を生産する場合でも、生産を行う機械が異なれば生産量はことなるし、逆に同じ機械でも製品が異なれば生産量は異なる。

この工場の操業計画問題における制約は全部で１００種類を越えるが、重要な制約は次の３種類の制約である。

（１）中間製品の在庫量の下限

この制約は、中間製品の生産と最終製品の生産の時間的な関係を表すための制約である。最終製品の生産が、その最終製品が必要とする中間製品が必要なだけ生産されていない状態で行われると、中間製品の在庫が負になる。すなわち、この制約が満足されない。逆にこの制約が満足されているならば、最終製品の生産は中間製品が必要なだけ生産されている状態で行われていることを示す。

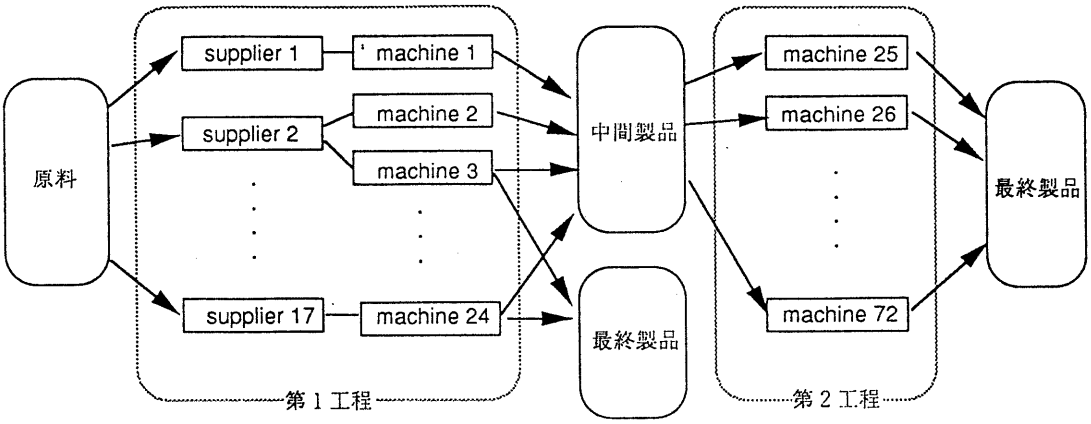

図１　工場における生産の流れ

（２）１日当りの切替回数の制限

ある機械で前日と異なる製品を生産する場合には、生産の前に切替と呼ばれる作業を行う必要ある。切替にはある時間が必要となるので、１日に行い得る切替作業の回数はある一定回数以下でなければならない。

（３）同時生産制約

第１工程用の機械には、それぞれ原料を供給するための供給器がついている。供給器のうち１部は単一の機械へ原料を供給するのに使われるが、いくつかの供給器は２つの機械に同時に原料を供給する（図１）。そのため供給器を共有するような２台の機械では、同じ原料を使う製品しか同時に生産することができない。

近傍探索法では、制約充足は、制約違反度を表す関数を定義し、その関数を最小化することにより達成される。そのため、この問題には元々の目的関数である生産の不足量の他に上の３種類の制約に対応する３種類の目的関数がある。各制約の違反度は次のように定義する。

（１）在庫制約の違反度（図２）

$$\sum_{p}\sum_{d}\max(0,\ -\mathrm{stock}(p,d))$$

（２）切替制約

$$\sum_{d}\max(0,\ \mathrm{setup\_times}(d) - \mathrm{setup\_upper\_bound}(d))$$

（３）同時生産制約違反

$$\sum_{s}\sum_{d}(1 - \delta(\mathrm{material\_of}(p(m_1(s),d)), \mathrm{material\_of}(p(m_2(s),d))))$$

ここでp(m,d)は機械mが日dに生産している製品であり、全ての機械と全ての日に対してp(m,d)の値を決めることが操業計画を立案することである。stock(r,d)およびsetup_times(d)はそれぞれ製品rの日dにおける在庫量および日dにおける切替回数である。また、m1(s), m2(s), material_of(p) は定数であり、それぞれ第１工程において原料供給器sを共有する２個の機械と、中間製品pを生産するための原料を表す。stock(r,d)や

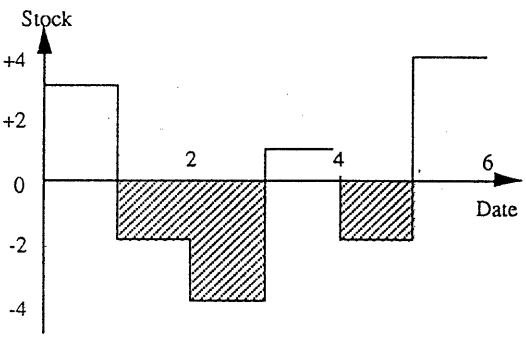

図２　在庫制約の違反度

ある製品の在庫違反度は図の斜線部分の面積

set up_times(d)の値は、p(m,d)の値が全て決れば、決めることができる。このような意味でp(m,d)のことを独立変数、stock(r,d)やset up_times(d)のことを従属変数と呼ぶことにする。すなわち従属変数は独立変数および自分以外の従属変数の関数であり、目的関数も従属変数の一種である。従属変数の値は独立変数の値から計算できるから、目的関数を直接独立変数を使って定義することもできる。ところが、ある程度構造を持った問題では、目的関数中に従属変数に対応するような独立変数の組合せが何度も現れることになり、独立変数だけで目的関数を記述すると目的関数の定義が非常に長くなる。この操業計画問題では、４つの目的関数を合計するとその長さ（現れる独立変数の総数）は独立変数の数が約2,000であるにもかかわらず1,000,000を越える。しかし、stock(r,d)などの適当な従属変数を使えばそれを1/10以下に減らすことができる。このことは、目的関数値をそれだけ高速に求めることができることを意味するから、この問題のように大規模な問題に対しては非常に重要な意味を持つ。

この問題の規模は次の通りである。機械の数は７２台、製品の種類が２０６種類、機械１台が生産可能な製品の種類の平均は約２０種類、計画期間は１カ月である。独立変数p(m,d)は、機械mが日dになにを生産するかを表すから、変数の数は約２０００個で、変数の取り得る値の数すなわち定義域の大きさの平均は約２０個になる。

現在この工場では、１人の専門家が毎月の操業計画を立案している。専門家は上に揚げた制約を全て満足し、要求量からの生産量の不足が約５％（６０トン）の解を約２日かけて立案している。

3　シミュレーテッドアニーリング

SAは近傍探索法の一種であり、生成された近傍を受理するかどうかの判定に目的関数値を用いるという点では山登り法と似ている。しかし、山登り法では目的関数値が良くなるような近傍のみを受理するのに対して、SAでは目的関数値が悪化するような近傍も確率的に受理する。

以下にSAのアルゴリズムを示す。なお以後は簡単のため最適化＝最小化として説明する。

```
procedure SA
  S←initial solution;
  n←0;
  while the finish condition is not satisfied do begin
    while the temperature change condition
                                  is not satisfied do begin
      S'← a neighbor of S;
      P_transition ← min( 1,exp(-(f(S') - f(S)) / T(n)) )
      r← random number ∈ [ 0, 1 );
      if r ≤ P_transition  then S ← S';
    end;
    n←n+1;
  end;
```

ここでT(n)はアニーリングスケジュールと呼ばれる関数で、正の値を取り、nに対して単調に減少する。また、T(n)の値を温度と呼ぶ。目的関数を悪化させるような近傍を受理する確率は、目的関数の変化量（悪化量）と温度との比率で決り、変化量が温度よりも小さい場合には十分大きな値となる。そのため、上のアルゴリズムの２番目のwhile-loopのなかでは、温度T(n)以下の凹凸を無視した最適化が行われることになる。言い替えると、深さがT(n)以下であるような局所最適解から脱出できることを示している。

SAで良い解を得るためには、どのようなアニーリングスケジュールを用いるかが重要になる。Gemanらは、次のようなアニーリングスケジュールを用いれば必ず最適解を得ることができることを証明した[5]。

$$T(n) = \frac{T_0}{\ln(n+2)}$$

ここでT_0は最も深い局所最適解の深さである。しかし、この関数はなかなか０に収束せず、探索が終了するまでに非常に時間がかかるため、現実的ではない。そこで、通常はより速く収束する関数が使われる。例えば

$$T(n) = T_0 * R^n$$

$$T(n) = \frac{T_0}{n+1}$$

というような関数である。これらのアニーリングスケジュールでは最適解を得るという保証はないが、十分良い解を得ることができる。

4　多目的最適化への拡張：SA/M

実際の計画問題や設計問題では、ほとんどの場合解の評価基準は複数ある。また、SAなどの近傍探索法では制約充足は制約違反度を表す関数の最小化としてあつかわれる場合が多いので、制約も一種の目的関数といえるから、例え本来の目的関数が一つでも、制約が存在する場合は目的関数が複数個になる。このように、実際の最適化問題は本来は多目的最適化問題である。ところが、SAでは目的関数を一つしか扱うことができない。そのため、適当な重みを用いた線型結合等により仮の目的関数を作り、それを最適化するということを行う。しかし、この方法では、ある目的関数の値は非常に良いけれども、別の目的関数の値が悪いというような解が得られる可能性がある。このような解は現実の問題においては望ましい解ではない。望ましい解とは、全ての目的関数の値はバランスよく良くするような解である。このような解を得るためにSAの多目的最適化問題への拡張であるSA/Mを提案する。

目的関数を$f_1(S)$, ..., $f_N(S)$とすると、SA/Mは以下のアルゴリズムにより、最適化を行う。

```
procedure SA/M
  S←initial solution;
  n←0;
  while the finish condition is not satisfied do begin
    while the temperature change condition
                          is not satisfied do begin
      S'← a neighbor of S;
      P_i ← min( 1, exp( - (f_i(S') - f_i(S)) / T_i(n) ) )
      P_transition ← ∏_{i=1}^{N} P_i
      r← random number ∈ [ 0, 1 );
      if r ≤ P_transition  then S ← S';
    end;
    n←n+1;
  end;
```

これと3節の通常のSAのアルゴリズムとの差は、解Sからその近傍S'への遷移確率の定義である。この遷移確率の特徴は、ある目的関数の値が悪くなる場合は、他の目的関数がどれだけ改善されていても小さな値をとるということである。このため、極端にバランスが悪い解を避けることができる。

SA/Mのもうひとつの特徴は、アニーリングスケジュールを目的関数毎に別々に設定できるという点である。これにより、各目的関数の重要度を探索中に変更することができる。例えば、ある目的関数$f_i(S)$に対しては、十分高い温度から出発し、急速に温度が下がるようなアニーリングスケジュールを用い、他の目的関数$f_j(n)$には低い温度からスタートするがなかなか温度が低くならないようなアニーリングスケジュールを用いると、探索の初期には$f_i(S)$は殆ど考慮されずに$f_j(S)$の最適化が行われるが、探索が進むにつれて$f_i(S)$の相対的な重要度が増加するため、徐々に$f_i(S)$の値が改善されていく。しかし$f_j(S)$の温度も低いので、$f_j(S)$が極端に悪化することはない。これを利用すると、制約がある最適化問題において、探索の初期には制約充足をあまり考えずに目的関数の改善を行い、その後、目的関数をできるだけ悪化させないように制約充足を進めるということができる（図3）。逆に探索の前半で制約充足を行い、後半で制約を充足する領域内での（本来の）目的関数の改善を行うこともできる。問題の性質に合わせて各目的関数毎にアニーリングスケジュールを設定することにより、単一の（合成された）目的関数を用いるよりも良い解を求めることができる。

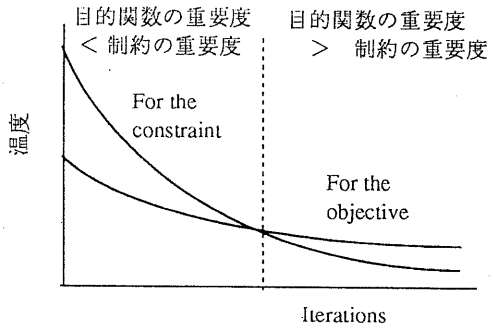

図3　アニーリングスケジュールと重要度の変化

5 SAの高速化

SAなどの近傍探索法による探索時間は、調べる解の数と1つの解を調べるのに必要な時間で決る。よって、探索時間を短縮するためには、調べる解の数を減らすか、1つの解を調べるのに要する時間を短縮する必要がある。

5-1 ルールによる近傍生成

調べる解の数を減らすには2つの方法がある。ひとつは出来るだけ目標とする解に近い初期解を使うことである。近傍探索法における初期解の重要性についてはこれまでにもかなり議論がなされている[7]ので、ここでは述べない。もうひとつの方法は、できるだけ寄り道をせずにまっすぐに目標に向って進むことである。これは近傍を生成する際の制限や優先順位をルールという形で与えることにより実現できる。ここで重要なのは、生成された近傍を受理するかどうかはSAの探索アルゴリズムが判定するために、ルール自身は必ずしも良い近傍のみを生成する必要はないという点である。言い替えればルールの健全性はあまり要求されない。これは問題に依存しない、非常に一般的なルールが使えることを意味している。そのため、知識獲得の困難はほとんどない。もちろん問題固有のルールが比較的簡単に獲得できるのならばそれを用いればより効率を上げることができる。

5-2 依存関係を利用した再計算アルゴリズム VP2

大規模な組合せ最適化問題では、解の目的関数値を計算することは、かならずしも容易ではない。例えば2節で述べた操業計画問題で、ある解が与えられたとき4個の目的関数の値を計算するためには約15,000個の従属変数の値を計算しなければならない。しかし、SAなどの近傍探索法では、近傍は小数の独立変数の値を変更することによって生成されるから、従属変数のなかでその値が変更されるものの割合も比較的小さいことが期待できる。そのため、全ての従属変数の値を計算する必要はなく、値が変わる可能性のある従属変数だけを再計算すればよい。この再計算は従属変数の依存関係を利用した、単純な伝播を用いて実行することができる。

```
procedure VP1
  Q ← set of dependent variables which directly depend
        on independent variables whose values are changed
  while Q ≠ φ do begin
    select and remove a variable u from Q;
    calculate and renew the value of u;
    if the value of u is changed then Q←Q∪forward(u);
  end;
```

ここで、forward(u)は、変数uに直接依存するような、すなわち値を計算するための関数中にuが現れるような従属変数の集合である。このアルゴリズムは、2つの変数が相互に依存しあっているような場合には停止性が保証されない。しかし、通常の問題は相互依存するような変数の組合せが現れないように問題を定式化することが可能である。

このアルゴリズムのもうひとつの問題点は、その効率である。このアルゴリズムでは、ある変数の値が複数回計算される可能性がある。このような冗長な計算は、Qからの変数の選択時に適当な選択基準を用いれば避けることができる。そのためにまず、全ての従属変数について以下の条件を満足する非負の整数レベルを計算する。

(c1) level(u) ≤ level(u') if u' depends on u
(c2) level(u) = level(u') iff u depends on u' and u' depends on u

レベルは、有効グラフの2重連結成分を列挙するアルゴリズムとトポロジカルソート[6]によって比較的簡単に計算でき、その計算量は依存関係を表すグラフの辺の数のオーダーである。レベルを用いると、以下のようなアルゴリズムによりVP1における冗長な計算を避けることができる。

```
procedure VP2
  Q ← set of dependent variables which directly depend on
        independent variables whose values are changed
  while Q ≠ φ do begin
    select and remove a variable u whose level is
      smallest in Q;
    calculate and renew the value of u;
    if the value of u is changed then Q←Q∪forward( u );
end;
```

forward(u) に含まれる従属変数のレベルは、(c1)よりuのレベルと等しいか、あるいは大きい。よって、Qから選択される従属変数のレベルは単調に増加する。もし、相互依存しているような変数の組が存在しないならば、すなわち、(c2)より全ての従属変数のレベルが異なる場合には、変数は自分自身には依存しないから、Qから選択される変数のレベルは狭義に単調増加する。このことから、ある変数の値の計算は高々一回しか行われない。すなわちVP1とは異なり、冗長な計算は行われない。

6 実験

6-1 実験の設定：近傍の定義と初期解

実験は2節で述べた化学繊維工場の月間操業計画問題を用いて行った。実験結果に入る前に、2節の補足として、SAを用いる際に、近傍をどのように生成したかと、探索の初期解について述べる。

最も単純な近傍の定義は独立変数を1つ選択して、その値を変更することである。この問題では、機械と日を選択して、その機械でその日に生産する製品を変更することがこれに対応する。しかしこのような近傍の生成では非常に多くの局所最適解が生じ、良い解を得ることが非常に難しくなる。そこで、次の7つのオペレータを用いて、近傍を生成する。

O1: 機械と日を選択し、その機械のその日の製品を変更する。
O2: 機械と日を2組選択し、（もし可能なら）そこで生産されている製品を交換する。
O3: タスクを生成する。
O4: 2つのタスクを選択し、（もし可能なら）それらの製品を交換する。
O5: タスクを2つのタスクに分割する。
O6: 同じ機械上の連続する2つのタスクを選択し、その順序を変える。
O7: タスクの長さを伸ばす（縮める）。

ここで、タスクとは、同じ機械上で同じ製品を生産している領域のことである。一般に工場などの生産計画問題では、タスクをベースとしたオペレータにより近傍を生成すると良い解が得られる。

実験では、上のオペレータをランダムに使って近傍を生成する場合と、ルールを用いてオペレータの適用を制限したり優先順位をつけたりした場合との比較を行う。ただし、ルールはこの問題に固有のルールではなく、以下のように、操業計画問題一般に使えるようなごく一般的なルールである。

rule-1 : 在庫が負になっている中間製品があるなら、その中間製品を生産しているタスクの長さを伸ばす。

次に、SAを行う際の初期解について述べる。この実験では初期解は簡単な欲張り法で生成した。ここで用いたヒューリスティックは、生産が不足している製品を可能な限り生産し続けるというもので、在庫制約と同時生産制約は完全に満足し、切替制約には若干違反するような解を作り出す。ただし、生産不足量はかなり残る。この初期解生成は2分程度で終了した。

6-2 実験結果

本論文で提案した方式について、2節で述べた操業計画問題に適用し、その評価を行った。表1は目的関数値を計算するための伝播アルゴリズムVP2の効果に関する実験の結果である。比較対象は、全ての変数の値を計算しなおした場合と、変数の順序付を行わない伝播アルゴリズムであるVP1である。実験は、ある解に6-1で述べた7個の近傍生成用の演算子をランダムに適用して得られた10,000個の近傍を用いて行った。1個の近傍は、平均的には、2,000個の独立変数のうち約6個の独立変数の値を変更することにより生成されている。表1は、1個の近傍を評価ためにおこなわれた、変数の値の計算回数および計算時間の平均である。実験はSPARCstation2上でCを用いて行った。これはこの他の実験でも同様である。VP2は、全てを計算しなおす場合に比べて約20倍、VP1と比べても約2倍高速であった。

次に、実際に2節の操業計画問題を解く際にSA/Mや近傍生成用のルールがどの程度効果があるかを実験した。表2と表3はその結果である。表2は、4個の目的関数の線型結合により仮の目的関数をつくり、それを通常のSAで最適化した場合と、4節で提案した多目的最適化問題への拡張であるSA/Mとの比較である。また表3はSA/Mで近傍生成にルールを使用した場合としない場合との比較である。なお、近傍の生成や受理するかどうかの判定に乱数を用いているので、全てのケースについて乱数の系列を変えて3回づつ測

	計算回数	cpu time (sec)
全ての変数の値を計算	15383.0	2.05
VP1	593.4	0.18
VP2	263.6	0.09

表1　近傍の評価速度の比較

	cpu time (sec)	在庫制約違反（トン・日）	切替回数違反（回）	同時生産制約違反（日）	生産不足量（トン）
Traditional SA 1	4493	0.0	0.0	0.0	83.8
Traditional SA 2	2602	0.7	0.3	1.6	69.4
SA/M	2257	3.8	2.3	0.0	51.3
Expert	(2 days)	0.0	0.0	0.0	60.0

表2　仮の目的関数を線型結合で生成して、それを最適化した場合とSA/Mとの比較
Traditional SA 1 : 制約重視
Traditional SA 2 : 最良の結果を与えた重み

	cpu time (sec)	在庫制約違反（トン・日）	切替回数違反（回）	同時生産制約違反（日）	生産不足量（トン）
without rules	3834	5.6	2.7	0.0	80.5
with rules	2257	3.8	2.3	0.0	51.3

表3　SA/Mにおけるルールの効果

定を行い、表にはその平均を示した。また、この実験では、表1の実験でVP1の効果が証明されたので、近傍の評価にはVP2を用いている。表2のTraditional SA 1は、制約充足解を求めるために、3種類の制約の重みを比較的大きく取って線型結合をした場合の結果であり、実際に制約が完全に満足されている解が求められた。しかし、生産不足量は非常に大きい。なお、ここでは、近傍生成にルールを用いている。Traditional SA 2は、何通りかの重みの組合せのなかでベストのものであり、若干制約違反が起きているものの、SA1よりはかなり生産不足量を減らすことができた。しかし、専門家と比べるとまだ生産不足量が多い。SA./Mを用いた場合には、やはり制約違反が若干起きているが、生産不足量は専門家よりも約20%少なくすることができた。表3では、SA/Mにおいてルールを用いた場合と用いなかった場合とを比較している。この結果から分かるように、ルールを用いることにより、生産不足量を約40%減らすことができた。

7 まとめ

本論文では、多目的最適化問題へのシミュレーテッドアニーリングの拡張であるSA/Mと、変数間の依存関係を利用して高速に近傍の目的関数の値を計算する伝播アルゴリズムVP2を提案し、その効果を工場の月間の操業計画問題を用いて評価した。その結果、SA/MとVP2を用いることにより、約40分で専門家が2日かけて立案した計画よりも良い操業計画を得ることが出来た。今後は他の分野の問題に適用し、SA/MやVP2が、工場等の生産計画問題だけでなく、他の問題に対しても有効であることを示していく予定である。

References

[1] Kirkpatrick, S. et el. : Optimization by simulated annealing, Science, Vol.220, pp.671-680 (1983)

[2] 喜多 : Hopfield型ニューラルネットワークとシミュレーテッドアニーリング, 人工知能学会誌, Vol.7, pp.38-47 (1992)

[3] Yoshida, H., Yugami, N. and Hara, H. : How to use the ATMS for scheduling, Proceedings of the 4th UNB AI symposium, pp.613-623(1991)

[4] Hara, H., Yugami, N. and Yoshida, H. : An Assumption-based Combinatorial Optimization System, Proceedings of CAIA-92, pp.120-126 (1992)

[5] Geman, s. and Geman, D. : Stochastic relaxation, Gibbs distributions and the Baysiean restoration of images, IEEE trans., Vol.PAMI-6, pp.721-741 (1984)

[6] Knuth, D. E. : The Art of Computer Programming, vol. 1, Addison-Wesley (1973)

[7] Minton, S. et al. : Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems, Artificial Intelligence, Vol. 58, pp.161-205 (1992)