TM-1209

ルールベースト・アニーリング

加藤　等、荒木　均、野島　晋二
館野　峰夫、問藤　隆（松下）

August, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

ルールベースト・アニーリング
Rule-Based Annealing

加藤 等　　荒木 均　　野島 晋二
Hitoshi Kato　Hitoshi Araki　Shinji Nojima

舘野 峰夫　　間藤 隆一
Mineo Tateno　Ryuuichi Mato

松下電器産業（株）情報通信東京研究所
Tokyo Information And Communications Research Lab.
Matsushita Electric Industrial Co., Ltd.

概要

本研究では、シミュレーティッド・アニーリング（SA）法とルールベース・システムを融合したルールベースト・アニーリング（RA）法を提案する。RA法は、汎用的な最適化手法であるSA法をベースにし、ルールベース・システムなどで用いられるヒューリスティックな知識を付加し、有効な知識を優先的に利用することによって効率的な探索を実現する。また、SA法の確率的山登り機構によって局所的最適値を脱出でき、高速に精度のよい解を求めることができる。本手法を論理回路設計における機能合成問題に適用したところ、通常のSA法に比較して10倍以上の高速化が実現された。

1　はじめに

組合せ最適化問題は、巡回セールスマン問題[2]、ナップザック問題[2]をはじめ、各種のスケジューリング問題、最適設計問題など広い応用範囲を持ち、AIの分野でも代表的な問題として研究されてきた。組合せ最適化問題を効率的に解くことは、今後さらに重要になってくるであろう。本研究では、組合せ最適化問題を実用的な時間で解くために有効な手法として、ルールベースト・アニーリング（RA）法を提案する。RA法の特徴として、以下のものが挙げられる。

- 経験的知識をもとにしたルールベースを利用して効率的な探索をする。
- SA法の確率的山登りの枠組により、局所的最適解に陥りにくい機構を持つ。
- 効果的な知識が優先的に用いられるよう動的に調整する機構を持つ。
- システムの動作は、与えられた最適化時間が短いほど貪欲なルールベース・システムに近づき、十分な時間が与えられると確率的なSA法に近づく。

また、組合せ最適化問題の例として論理回路設計における機能合成問題[8]をとりあげ、SA法と比較して本手法の有効性を示した。

以下、第2章で組合せ最適化問題の種々の解法とその問題点について述べ、第3章でRA法の枠組および有効な知識を優先して用いるための2つの手法について述べる。また、第4章で機能合成問題とRA法の適用について述べる。第5章で実験結果とその評価、第6章でまとめと今後の課題を述べる。

2　組合せ最適化問題の解法

組合せ最適化問題を解くために厳密解法によってしらみつぶし探索をすると組合せ爆発を起こしてしまう。そのため、一般的には適用する問題の性質によって探索を枝刈りしたヒューリスティック解法を用いて近似解を求めることが多い。問題に適した貪欲解法、局所探索法[2]、緩和法など各種の近似解法が開発・利用されている。

一般に、貪欲なヒューリスティック解法を用いれば短時間である程度の近似解を得ることができる。

しかし、非常に単純な問題でない限り最適またはそれに近い解を求めることは困難なので、解の精度の点で問題がある。スケジューリング問題における ASAP (As Soon As Possible) [13] などがその一例である。同法では、各作業を開始時刻の早い順に配置していくが、作業資源がある日時に集中してしまうなど、あまりよいスケジュールが得られない。山くずし法 [4] など他の手法を用いてさらに最適化を図る必要がある。

AI の研究分野でも組合せ的な問題は数多く扱われており、TMS[9]、ATMS[11]、CSP[14] などがその手法であるが、これらは制約を満たす解を求めることが主な目的であり、最適化に関してはあまり考察されていない。

局所探索法 [2] は、何らかの手段で得られた近似解または適当な初期解から探索を開始し、微小な変更を加えることで生成される解に順次置き換えていく方法である。このとき解が改善される方向にだけ置き換える方法を山登り法と呼び、貪欲な解法としてよく用いられている。しかし、山登り法は現在の解から良くなる方向にしか探索が進まないため、解の精度は初期解に依存し、局所的最適値に陥る可能性が大きい。また、近年よく研究されている方法として遺伝的アルゴリズム (Genetic Algorithm)[7] があげられるが、遺伝的アルゴリズムでは一つの解を遺伝子（例えば 2 進のビット列）として表現する必要があり、実用的な問題ではその符合化法や制約条件を満たす子孫を作る交差方法などが難しい。

シミュレーティッド・アニーリング (SA) 法 [10] は局所探索法の一つであるが、山登り法と異なり目的関数が悪くなる方向への変換を確率的に許しており、貪欲性が小さい。うまくこの確率を制御すれば局所的最適解から脱出し大域的最適解へ収束することができる。このような探索を確率的山登り法 [15] という。ただし、通常の SA 法は現在の解をランダムに変形して次の解を生成するため無駄になってしまう探索が多く、最適解に収束するためには非現実的なほど膨大な計算時間が必要である。SA 法の高速化についてはさまざまな手法が研究されており、代表的なものとして並列処理による高速化があげられる [6][1] が、短い時間に近似解を得る場合には貪欲解法の方が有利である。

以上のことから、解の精度より計算時間の制約が大きいときには貪欲なヒューリスティック・アルゴリズムを用いて解を求め、十分な計算時間が与えられる場合には SA 法などの確率的で貪欲性が小さい手法を用いるというように使い分けるのが妥当だと考えられる。しかし、一般的には、許される有限時間内で最も精度のよい解が求められる必要がある。本稿で提案する RA 法は、長時間計算すれば最適に近い解へ収束するという特徴を持つ SA 法と、短い時間で速く近似解を求めるという貪欲なヒューリステック解法の両特徴を備えており、与えられた時間内にできる限り精度のよい解を求める機構を持っている。

3 RA 法

SA 法は、各温度で平衡状態に達するまで解をランダムに変換することを繰り返した後に温度をゆっくりと下げなければならない。これは、大域的最適解への収束を保証するためであるが、変換はランダムであるため無駄な変換が多くなり探索効率が悪くなっている。また、各温度で十分に変換をすることは実際には時間の制約からほとんど不可能であるため、その場合は大域的最適値への収束性は保証できなくなる。ただし、実用的には求める解が大域的最適解でなければならないことは少なく、最適解に近い解で十分であることが多い。そこで、大域的最適値への収束は保証されなくても、効果的な解の変換を増やし無駄な変換を削減することが優れた解を高速に求めるために有効な手段であると考える。

本稿で提案する RA 法では、ランダムな変換だけでなく、ルールベース・システムで利用される経験的な知識を用いることによって効率的な探索を実現し、高速化を図る。システムは経験的な知識による解の変換法（変換ルールと呼ぶ）を複数用意しており、これらの変換ルール（ランダム変換を含む）のうち一つを適用して現在の解から新しい解を生成することによって探索を進める（図 1）。また、各変換ルールは選択比率と呼ばれる値を保持しており、選択比率にしたがって 1 つの変換ルールが確率的に選ばれるようになっているため、局所的最適値に陥りにくく、かつ選択比率を動的に変化させることによって有効な知識を優先的に用いることができる。

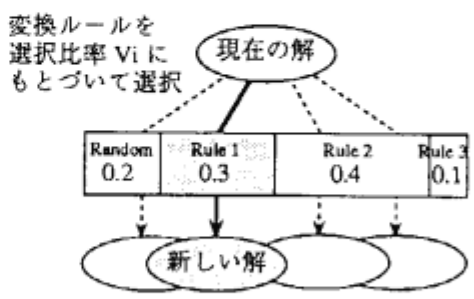

図 1: 変換ルールによる解の変換

この選択比率をうまく調整することにより、貪欲なルールを優先させたり、ランダム変換を優先させたりできるため、与えられた最適化時間によってシステムの動作を貪欲なルールベース・システムに近づけたり、確率的なSA法に近づけることが可能である。

3.1 RA法アルゴリズム

RA法のアルゴリズムを以下に示す。SA法と同様に、1つの解に対して一意にコストが定義されているかまたは計算によって求まるものとし、システムはコストが最小の解を探索するものとする。

```
RA (j0 , T0 , R0) {
  T = T0;          /* 初期温度の設定 */
  S = j0;          /* 初期解の設定 */
  R = R0;          /* 初期選択比率の設定 */
  while( 温度が十分小さくなっていない ) {
    reset(A);      /* データのリセット */
    while(十分な回数繰り返していない) {
      i = select(R, i);       /* ルール選択 */
      j = generate(S, i);     /* 新解を生成 */
      if(accept(c(j),c(S),T) {
        S = j;                /* 新解を受理 */
        A = count(A);         /* データ更新 */
        }
      }
      T = update(T);          /* 温度更新 */
      R = change(R, A);       /* 選択比率調整 */
    }
    return(S);                /* 解を出力 */
  }
accept(c(S') , c(S) , T) {
        /* T は温度パラメータ。 */
  Δc = c(S') - c(S);      /* コスト値の差 */
  y = f(Δc, T);           /* 受理関数 */
  r = random(0,1);        /* [0,1) の一様乱数 */
  if(r < y)
    return(1);            /* 受理 */
  else
    return(0);            /* 受理しない */
  }
```

ここで、受理関数 $f(\Delta c, T)$ は、

$$f(\Delta c, T) = \begin{cases} e^{-\frac{\Delta c}{T}} & \text{if } \Delta c > 0 \\ 1 & \text{otherwise} \end{cases}$$

である。

以下に、RAアルゴリズムを簡単に説明する。まず、初期アニーリング温度、初期解、ルールの初期選択比率を設定する。次に、変換ルール（ランダム変換、ヒューリスティック・ルールがある）の中から1つを選択比率に基づいて確率的に選び、現在の解に適用して新しい解候補を作る。このとき、各変換ルールの選択される確率はそのルールが持っている選択比率の値に等しく（図1）、選択比率は温度ごとに調整される。その後、コスト評価関数(accept)によって新しい解候補を受理するかどうかを決定する。すなわち、コストが減少した場合は確率1で新しい解候補を次の解とし、コストが増加した場合は増分を ΔC、そのときの温度パラメータを T としたとき確率 $e^{\frac{-\Delta C}{T}}$ で新しい解候補を次の解とする。このような解の変換を十分繰り返した後、温度をある規則(update)にしたがって下げ、各ルールの選択比率を調整して(change)さらに解の変換を繰り返す。温度が十分に小さくなったら、解を出力する。本稿では、コスト評価関数および温度更新規則にはSA法との比較のため、同じものを用いている。

本方式ではルールベースによる解の変換ルールを実装することにより知的に解を変換するため、効率的な解候補の生成をすることができる。ただし、変換ルールをあらかじめ用意する必要がある。また、アルゴリズムからもわかるように、SA法と同様の温度スケジュールを持ち、複数の変換ルール（ランダムな変換を含む）から一つを確率的に選ぶことにより、局所的最適解に陥るのを防いでいる。さらに、各変換ルールの選択比率を動的に調整し、有効な変換ルールを優先的に選択することによって、短い時間で精度のよい解に収束させる。調整の方法については後述するが、最も単純なものとしては、各変換ルールともすべて等しい初期選択比率を与えておき、温度ごとの更新をしない方法があり、この方法ではいつも各変換ルールからランダムに一つ選ぶことになる。選択比率をうまく調整することによって、貪欲なシステムにしたり、SA法に近い確率的なシステムにしたりすることが可能である。

3.2 変換ルールの選択制御

本RA法は、経験的知識による変換ルールを用いて解を変換するため、どの変換ルールを用いるかによってシステムの動作が異なってくる。どの変換ルールを用いて解を変換していけば最も効果的なのかは、一般に明らかではない。変換ルールを選択する方法として最も単純なものとしては、全ての各変換ルールからいつもランダムに一つ選ぶという方法

が考えられる。しかし、各変換ルールには有効に働く状況（現在の解の性質）があり、一定の確率で選択されてもいつも有効に働くわけではない。各変換ルールの効果は、現在の解の性質のほか、温度パラメータの値との関連がかなり大きいことがわかっている [3]。

本 RA 法では、効果の大小によって変換ルールの選択確率を温度ごとに動的に更新して、効果が大きいルールを優先して用いて効果が小さい変換ルールを次第に用いないようにする機能がある。ここでは、2 つの選択確率の更新方法を挙げる。1 つは受理率に着目した方法であり、新しく生成した解が受理される率が高い変換ルールを優先させる。もう 1 つはコスト減少率と呼ばれる貪欲性を表す値に着目した方法であり、これを用いればシステムを貪欲なシステムにしたり、SA 法に近い確率的なシステムにしたりすることが可能なため、計算時間が短いときは貪欲的な変換ルール、計算時間が長いときはランダムな変換を実際に優先させている。なお、両方式とも選択確率の調整は温度ごとに行なわれる。

3.2.1 受理率によるルール選択制御

受理率によるルール選択制御方式では、新しく生成した解が受理される率が高い変換を有効な変換ルールと考えて優先して選択する。これは、受理されない変換は解を置き換えることができないのであるから効果がなかったと見なし、よく受理される変換は解を置き換えることが多いので効果的であるという考えに基づく。本方式では、同一温度のアニーリングにおいて各変換ルールが選ばれた回数（選択回数）と生成した解が受理された回数（受理回数）をカウントし、温度が更新されるときに各変換ルールの受理率（受理回数／選択回数）を求め、その値の大小にもとづき、次の温度における各変換ルールの選択確率を決定する [3]。

選択比率の計算式は以下に示すとおりで、W は増減幅パラメータであり、この値が大きいほど選択比率が大きく増減する。k 番目の温度 $T(k)$ における各変換ルール i の選択比率を $V_i(k)$ とすると、次の温度 $T(k+1)$ における選択比率は、

$$V_i(k+1) = \frac{(1 + R_i(k)) \cdot V_i(k)}{\sum_i ((1 + R_i(k)) \cdot V_i(k))}$$

$$V_i(0) = \frac{1}{N}$$

である。ここで増減率 $R_i(k)$ は、

$$R_i(k) = W \cdot (F_i(k) - \frac{\sum_i F_i(k)}{N})$$

とした。ここで、$F_i(k)$ は変換ルール i の温度 $T(k)$ における受理率、N は変換ルール数である。各温度において変換ルール i の受理率が全ルールの平均受理率よりも大きい場合、選択比率 $V_i(k+1)$ が大きくなるので変換ルール i は次の温度において選ばれやすくなる。また、逆に平均受理率より小さい場合は選択比率 $V_i(k+1)$ が小さくなるので変換ルール i は次の温度において選ばれにくくなる。初期選択比率は各変換ルールとも等しく $1/N$ とした。

3.2.2 コスト減少率によるルール選択制御

最適化のためには、ある有限な計算時間内にできるだけ精度のよい解を求める必要があるが、受理率によるルール選択制御方式では計算時間は考慮されていない。また、受理されたかどうかだけでなくコストをどれだけ減少させたか、または増加させたかを考慮することにより、さらにきめ細かい制御が行なえる。そこで、コスト減少率によるルール選択制御方式では、以下のような基本的な考えを実現することを考えた。

- 与えられた計算時間が短いとき、より貪欲な変換ルールが多く適用され、速く近似解を求める。
- 与えられた計算時間が十分長いとき、ランダムに解を変換するルールが多く適用され、より最適に近い解を求める。

これらを実現するため、コスト減少率を定義する。コスト減少率は、各変換ルールが同一温度で何度か適用され、受理されたときのコスト減少値の総和をそのルールが選択された回数と消費した時間で割ったものであり、単位時間にどれだけコストを減少させたかを示す。コスト減少率は、その変換ルールの貪欲さを表すものであると考えられ、貪欲な変換ルールほど大きい値に、貪欲性の小さいランダムなどは小さい値になる。

本方式では、各温度のアニーリングの終了時に、コスト減少率とシステムに与えられた計算時間にもとづき、次の温度における各変換ルールの選択確率を決定する。選択比率の計算式は以下に示すとおりである。ここで W は増減幅パラメータであり、この値が大きいほど選択比率が大きく増減する。また、$Time_c$ は標準計算時間であり、この時間より短い計算時間が与えられたときは貪欲なルールを優先して利用する。変換ルール i のコスト減少率を g_i、システムに与えられた計算時間を $Time_{ex}$、k 番目の温度における各変換 i の選択比率を $V_i(k)$ とすると、

次の温度における選択比率は、

$$V_i(k+1) = \frac{(1+R_i(k))\cdot V_i(k)\cdot B_i}{\sum_i((1+R_i(k))\cdot V_i(k))\cdot B_i}$$

$$V_i(0) = \frac{1}{N}$$

である。ここで増減率 $R_i(k)$ は、

$$R_i(k) = A\cdot(g_i(k) - \frac{\sum_i g_i(k)}{N})$$

また、

$$A = W\cdot \ln\frac{Time_c}{Time_{ex}}$$

とした。ここで、N は変換ルール数である。

各変換ルールの選択比率 V_i は温度ごとに調整されるが、システムに与えられた計算時間によって調整のしかたが異なる。与えられた計算時間が小さいとき ($Time_{ex} < Time_c$) はコスト減少率が平均より大きい変換ルール（貪欲なルール）の選択比率が大きくなり、計算時間が大きくなると ($Time_{ex} > Time_c$) コスト減少率が平均より小さい変換ルール（ランダム変換など）の選択比率がやや大きくなる。これによって本節冒頭の機能を実現する。また、上式における B_i はルール i が現在までのアニーリングにおいて、解を変換できなかった回数から計算される係数 ($0.9 \le B_i \le 1$) であり、選択されたのにもかかわらず解を変換できなかった変換ルールは選択比率が少し減少する。これは、貪欲な変換ルールほどアニーリングの進行に伴い解を変換できないことが多くなって効率が落ちることを防ぐためであり、かわりに確率的な変換ルールがやや優先されることになる。

4 論理回路設計への適用

最適化問題の例として論理回路設計の上流工程である機能合成問題をとりあげ RA 法を適用した。そこで、本章では機能合成問題と実現した変換ルールについて述べる。

4.1 機能合成問題

機能合成とは、LSI の設計過程において、設計者が与える高級言語の動作仕様からレジスタ・トランスファ（RT）レベルの回路を生成する技術である。システムは、図2に示すように、動作仕様 (Pascal 言語) を構文解析して二項演算式をアニーリング可能なスケジュール表にマッピングする [8]。スケジュール表 (図3) はそれぞれの演算器 (ALU) が各動作ステップごとにどの式を実行するかを示している。システムは、与えられたスケジュール表を初期解として RA 法を実行することによって最適化する。その結果出力されたスケジュール表をもとにハードウェア記述言語によって記述された論理回路を生成する。なお、最適化の指標となるコスト c は任意のスケジュール表に対して一意に決まり、ALU、レジスタ、バスによるチップ面積、動作ステップ数を考慮した

$$c = (alu) + (register) + (bus) + (link) + (time)$$

で表され、システムはコストが最小のスケジュール表を探索する。

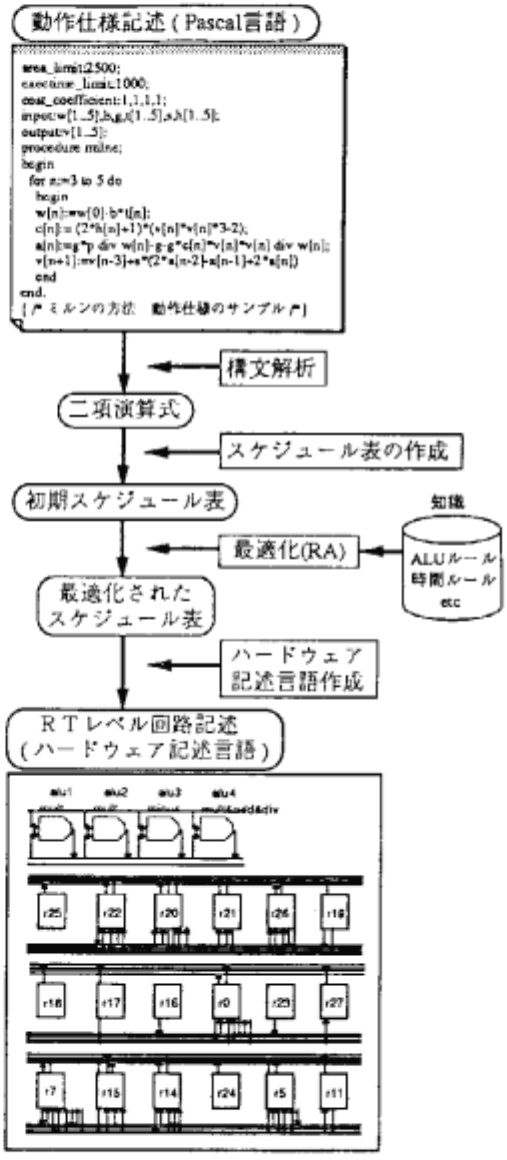

図 2: システムの流れ

	ALU0 *	ALU1 +, *	ALU2 −	ALU3 /
time0	bd=ad*cc	k=k+1		
time1				cc=n/k
time2		sc=sc+bd		
time3		kk=2*k		
time4			a4=ad−c	
time5				sc=ad/kk

図 3: スケジュール表

4.1.1 機能合成問題における変換ルール

本研究で実験に用いた機能合成問題におけるスケジュール表の変換ルールを示す。ここでRandomはランダム変換であり、通常のSA法ではこの変換だけを繰り返し適用することによって最適解を求める。その他のルールは知識にもとづく変換であり、コストcを効率的に下げる変換である。かなり貪欲的なものや比較的確率的なものも実現した。現在、ランダム変換を含めて9個の変換ルールが実現されており、これらは、選択比率にもとづいて確率的に選択される。また、各変換ルールは必要に応じて複数の式を移動させる。

1. (Random) 式をランダムに一つ選びランダムな場所に移動する。式の重なりを許す。
2. 式をランダムに一つ選び他の式と重ならないようにランダムな場所に移動する。
3. 式をランダムに一つ選び同じ演算を持つALUに移動する。
4. 式をランダムに一つ選びなるべく動作ステップとALUコストを増やさないように移動する。
5. 2つ以上の演算子を実行するALUのうち、式数が少ない演算子の式を他のALUに移動する。
6. 式数が最も少ない動作ステップの式を動作ステップが増加しないように移動する。
7. 式数が少ない動作ステップの式をALUコストが増加しないように移動する。
8. 式数が最も少ない動作ステップの式を、ランダムに他の動作ステップに移動する。
9. 式数が少ないALUの式を他のALUへ移動する。

5 実験結果と評価

上記のようなRA法を（財）新世代コンピュータ技術開発機構（ICOT）のPSI上に並列論理型言語KL1を用いて実現した。対象とした問題は論理回路設計における機能合成であり、用いた動作仕様はエリプティック・フィルタ（二項演算式32個）[8]である。実現した変換ルールはランダム変換を含む9個である。通常のSA法との比較実験のため、各パラメータは以下のように設定した。

初期温度$T(0)$	2764°
温度更新規則	$T(k+1) = 0.9 \cdot T(k)$
終了条件	各温度の最終コストが5回連続同じ

ただし、$T(0)$は受理率$\chi_0 = 0.9$として、次式により求めたものである[12]。ここで、$\overline{\Delta C}^{(+)}$はランダムウォークの平均コスト増加値である。

$$T_0 = \frac{\overline{\Delta C}^{(+)}}{\ln(\chi_0{}^{-1})}$$

一定温度での変換回数を変化させたときのアニーリング終了時の計算時間とそのときのコスト値を測定した実験結果を表1、表2、表3、表4に示す。各実験で乱数の種を変えて15回試行した。各表はそれぞれ通常のSA法、等確率に変換ルールを選ぶRA法、受理率によるルール選択制御を行なったRA法、コスト減少率によるルール選択制御を行なったRA法による結果を示す。受理率によるルール選択制御を行なったRA法におけるパラメータは$W = 0.05$とした。また、コスト減少率によるルール選択制御を行なったRA法におけるパラメータはW=0.003、$Time_c$=1200(sec)とし、システムに与える最適化時間$Time_{ex}$は、一定温度での変換回数から近似的に計算した。その他、各ルールが変換の際に消費する時間は前もって測定しておいた近似値を用いた。

表 1: SA法による実験結果

変換回数	時間(sec)	コスト
16	21.8	2584.7
32	49.3	2378.0
64	94.7	2254.7
128	195.3	2169.0
256	383.2	2041.3
512	799.1	1985.0

各表から明らかなように、ランダム変換のみによって解の変換を行なうSA法よりも各RA法が短

い時間でよい解（小さいコスト）に収束していることがわかる。表1と表2において、等しいコストに収束する時間を比較することにより、その高速化の効果は約10倍であることがわかる。

表 2: RA 法 (等確率) による実験結果

変換回数	時間 (sec)	コスト
16	18.9	2226.3
32	34.8	2047.0
64	76.9	2030.3
128	164.0	1949.3
256	347.4	1939.0
512	856.5	1889.3

表 3: RA(受理率) 法による実験結果

変換回数	時間 (sec)	コスト
16	19.2	2122.3
32	41.1	1996.0
64	85.0	1976.0
128	176.1	1917.3
256	386.9	1882.1
512	873.9	1882.7

表 4: RA(コスト減少率) 法による実験結果

変換回数	時間 (sec)	コスト
16	19.0	2050.0
32	44.1	1986.7
64	91.3	1973.7
128	201.0	1909.0
256	417.0	1886.7
512	921.4	1832.7

ルール選択制御を行なった2つの RA 法は、いずれも等確率に選択する RA 法よりもさらに高速になった。その高速化の効果は、等しいコストに収束する時間を比較することにより、約2倍であることがわかる。等確率に選択する RA 法と受理率によるルール選択制御を行なった RA 法では、計算時間が長くなっていったときに得られるコストがあまりよくなっていない（512 回）。これは、局所的最適値に陥っているためと考えられる。コスト減少率によるルール選択制御を行なった RA 法は、受理率によるルール選択制御の RA 法によって得られるコスト値と比較して全体的にあまり差がなかったが、計算時間が短いときと長いときにより低いコストの解を得ることができた。

6 おわりに

本稿では、SA 法と経験的知識によるルールベースとの融合によって高速化を図る RA 法について述べた。本方式は、経験的な知識を用いて高速に組合せ最適化を実現する、SA 法の持つ確率性によって局所的最適解を避ける、有効な知識（ルール）を優先的に利用するという特徴を持つ。さらに、本方式を論理設計における機能合成問題に適用してその有効性を示したが、他の問題についても変換ルールを変更することによって適用可能である。すでに巡回セールスマン問題に対して、等確率で各ルールを選択する RA 法、受理率によるルール選択制御を実施した RA 法を実際に適用して有効であることを確認している。ただし、RA 法の効果は変換ルールの質にある程度依存しているため、質のよい変換ルールを実現することが重要である。

今後の課題としては、今回の実験に用いた問題より大きいサイズの問題で RA 法の有効性を確認することと、一般的なスケジュール問題など、他の組合せ最適化問題に適用することを検討している。

謝辞

本研究は、(財) 新世代コンピュータ技術開発機構（ICOT）との再委託契約（発仕 S6202）に基づき行なわれたものである。新田克己室長を始めとする ICOT 第7研究室の諸氏にご助言して頂いたことは本研究の推進に大変有用でありました。ここに記して感謝します。

参考文献

[1] 荒木、舘野、加藤、間藤: 疎結合並列計算機上でのシミュレーティッド・アニーリング, 情処研報, 91-AI-77-2, pp.7-14, 1991.

[2] 石畑: アルゴリズムとデータ構造, 岩波講座ソフトウェア科学 3, P.486, 1989.

[3] 舘野、荒木、加藤、間藤: Rule-based Annealing, 情処研報, 91-AI-78-2, 1991.

[4] 刀根 監修: PERT 講座 第1巻 基礎編, 東洋経済新報社, P.254, 1966.

[5] 原、湯上、吉田: データ依存関係を用いた組合せ最適化手法, 情処研報, 91-AI-76-4, 1991.

[6] E.H.L.Aarts, F.M.J.Bont, J.H.A.Habers, P.J.M.vanLaarHoven, *"Parallel implementations of the statistical cooling Algorithm"*, Integration, journal 4, pp.209-238 (1986).

[7] L.Davis (Editor), *"Genetic Algorithms and Simulated Annealing"*, Pitman Publishing and Morgan Kaufmann Publishers, P.216 (1987).

[8] S.Devadas and A.R.Newton, *"Algorithms for hardware allocation in data path synthesis"*, IEEE Trans. on CAD, vol.8, no.7, pp.768-781 (1989).

[9] J.Doyle, *"A Truth Maintenance System"*, Artificial Intelligence, Vol.12 (1979).

[10] S.Kirkpatrick, C.D.Gelatt and M.P. Vecci, *"Optimization by simulated annealing"*, *Science*, vol.220, no.4598, pp.671-683 (1983).

[11] J.de Kleer, *"An Assumption-based TMS"*, Artificial Intelligence, Vol.28, No.2 (1986).

[12] P.J.M. van Laarhoven and E.H.L.Aarts, *"Simulated Annealing : theory and applications"*, Kluwer Academic Publishers (1987).

[13] M.C.MacFarland, A.C.Parker, R.Camposano, *"Tutorial on High-Level Synthesis"*, Proc. ACM/IEEE DAC, pp.330-336 (1988).

[14] A.K.Mackworth, E.C.Frender, *"The complexity of some polynomial network consistency algorithms for constraint satisfaction problems"*, Artificial Intelligence, Vol.25, No.1 (1985).

[15] F.Romeo and A.Samgiovanni-Vincentelli, *"Probabilistic hill climbing algorithms: properties and Applications"*, Proc. Chapel Hill Conference on VLSI, pp.393-417 (1985).