TM-1208

遺伝的ルールによる
並列ルールベースト・アニーリング

間藤　隆一、荒木　均
加藤　等、野島　晋二（松下）

August, 1992

© 1992, ICOT

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

遺伝的ルールによる並列ルールベースト・アニーリング

Parallel Rule-Based Annealing by Genetic rule

間藤 隆一　荒木 均　加藤 等　野島 晋二

Ryuuichi Mato　Hitoshi Araki　Hitoshi Kato　Shinji Nojima

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

我々は、シミュレーティッド・アニーリング(SA)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)を提案し、その方法が高速で局所的最小解に捕らわれにくい組合せ最適化の枠組であることを実験的に確認した[2]、[3]。

本研究では、さらに高速な枠組を目的として、疎結合型並列計算機におけるRAの並列処理方法(並列RA)を提案する。並列RAは各プロセッサ・エレメント(PE)がRAを実行する基本動作に加えて、遺伝的アルゴリズム(GA)の交差オペレータに相当する操作を遺伝的ルールとしてRAを自然に拡張することによって、各PEが探索した有効な組合せ情報を伝達して、協調的な探索を実行する並列処理方法である。

この並列RAをLSIのハイレベル合成問題に適用する方法も提案する。

1　はじめに

組合せ最適化問題の効率的な解法は、スケジューリング問題、配置問題や設計問題などをはじめ、情報処理の中心的な課題になっている。人工知能およびニューラルネット等多くの分野で研究されている問題であるが、処理時間の問題、局所的な最小解に陥る問題や実現性の困難さの問題からいづれも決定的な解法とはなっていない。実際は、一部の問題を除いて個々の問題や問題の規模に合わせて、ヒューリスティックなアルゴリズムにより解法しているのが現状である。

以下、SAとGAに分野を絞って、組合せ最適化問題に関する研究とその特徴について簡単に述べる。

最近、注目されているGA[4]、[5]は、組合せ最適化方法としても強力な手法である。

ビット列のデータ構造を遺伝子とみなし、遺伝子の集合を一世代とし、その集合の中で、良い形質をもつ遺伝子を利用して子供の遺伝子を生成し、悪い形質をもつ遺伝子が消滅し、次の世代の遺伝子の集合を生成する。GAは、この操作を繰り返すことにより、最適化を行う手法である。

実際の組合せ最適化問題では、単純なビット列のデータ構造では、組合せの状態を表現できなず、また、親遺伝子から子遺伝子を生成するとき、単純な交差や反転の操作では良い形質をもつ遺伝子が生成することができない。そのため、[6]では、巡回セールスマン問題(TSP)を解法するために、都市の順列を遺伝子とみなし、問題特有の知識を利用して、交差や反転のような操作を実現し、問題を解法するアルゴリズムを提案した。このように、GAを使用して、問題特有のデータ構造と遺伝子操作により、組合せ最適化問題を解法することができる。その他にもスケジュール問題を扱った[?]をはじめとして、多くの研究が報告されている。

GAの特徴の1つは、多くの従来の最適化手法が1つの組合せを少しずつ改善していたのに対し、複数の組合せを絶えず保持し、複数の組合せから新たに1つの組合せを生成していく点があげられる。このため、局所的最小解に陥りにくい性質をもっている。反面、多くの組合せを改善していく必要があるので、処理時間の増加を招きやすい。その解決策として、GAの場合、組合せの改善の処理が基本的に独立しているので、GAの並列処理が期待されている[5]。また、GAは、SAの温度パラメータようなものがなく、子遺伝子を生成するための親遺伝子の選択方法や形質の悪い遺伝子の消滅方法が最適化

の間に同じ方法で行うので、最適化の最初から局所解に収束する危険や最適化の最後に、良質の解が消滅してしまう危険がある。

一方、SA[11]も、組合せ最適化問題の有力な解法の1つである。SAは温度降下の方法や新しい組合せの生成の方法などの一定の条件を満たせば、理論的には、大域的な最小解を求めることができることが証明されている[13]。LSIの配置・配線問題では、実用的に使用されているアルゴリズムである。その反面、新しい組合せを生成する膨大な繰り返しを必要とし、処理時間が問題になっている。

SAの高速化手法として、新しい組合せを生成するとき、ランダムな組合せの要素の移動だけでなく、改善知識(repair knowledge)と呼ばれる知識を予め指定した順番に適合して新しい組合せを生成していくConstraint-Based Simulated Annealing[8]という方法が提案され、実用上、SAより高速であることが実験的に確認されている。

我々は、改善知識を単に順番に適用するのではなく、改善知識に相当するルールをルールが受理された確率やルールによりコストが減少した確率に基づき適用するルールベースト・アニーリング(RA)を開発し、SAより高速であることを実験的に確認した[2]、[3]。

SAの並列処理による高速化の研究も[1]、[14]をはじめとして、多くの研究が進められている。

SAとGAを融合する研究としては、複数の状態を保持し、GAの遺伝子操作後の解の選択を温度パラメータにより行い、フロアプラン設計問題に適用し、良質な解が得られたという報告がなされている[9]。

SAとGAを融合した並列処理の研究としては、[10]があり、本方法とは、異なる方法を提案している。

本報告での並列処理の目標は、GAの特徴である2つの組合せから1つ組合せを生成する操作をRAの枠組の延長として、遺伝的ルールとして取り込み、単に各PEが、RAを実行するのではなく、遺伝的ルールにより、各PEの組合せの情報を交換する協調的な機構により、高速な組合せ最適化の枠組を実現することである。

本報告では、組合せ最適化問題の例としてハイレベル合成問題[16]に関する遺伝的ルールを説明する。

以下、第2章で並列RAについて、並列処理方法を中心に述べる。第3章でハイレベル合成問題とこの問題における遺伝的ルールについて述べる。第4章で並列RAの特徴をまとめる。第5章で今後の課題を述べる。

2 並列RA

2.1 新しい組合せの生成

本節では、GAの遺伝子操作用のオペレータが、RAの新しい組合せの生成のためのルールとして位置付けられることをしめす。

本方法は反復改善法であり、ある組合せから始めて、その組合せに基づいて、新しい組合せを生成する必要がある。そのために、ルールと呼ぶ組合せ生成用の複数の知識を用意する。

巡回セールマン問題(TSP)を例として説明する。TSPは、空間上の座標が既定のN個の都市を1度づつ巡回する経路の内、最短の経路を見つける問題である。図1のように、ある巡回経路を都市名AからFの順列で表現する。(a)の現在の組合せでは、B、A、C、F、E、Dの順に都市を巡回する。

SAでは、最適解を保証するため、ランダムに次の組合せを生成する。(a)では、都市の中からランダムに2つの都市(AとF)が選択され、その2つの都市を交換することで新しい組合せが生成される。GAの反転オペレータや突然変異オペレータは、この範疇に属す。

RAでは、ランダムな方法のほかに、知識により新しい組合せを生成している。この方法を加えることは、最適解への収束が保証されなくなる反面、実際のSAでは、温度を離散的な値しかとれず、同じ温度での組合せ生成回数も上限あるため、ランダムな方法だけの方法より知識を加えた方が高速であることが実験的に確認されている[8]、[?]。知識による方法の例は、図1の(b)に示すように、ランダムにある部分経路(A、C、F)を選択し、その中を最短経路(A、F、C)に変換し、新しい組合せを生成する方法が考えられる。

さらに、並列RAでは、2つの組合せから1つの新しい組合せを生成する方法を追加した。これは、GAの交差オペレータに相当するものである。この方法の簡単な例を、図1の(c)に示す。ある親の巡回経路の半分の部分経路(B、A、C)を選択する。その部分経路の続きを別の親の部分経路をできるだけ親の経路を継承して、選択する(D、E、F)。並列RAでは、もう1つの親は、その他のPEから転送してもらう。

2.2 次の組合せの選択

上記の3つの方法により、現在の組合せから新しい組合せが生成される。

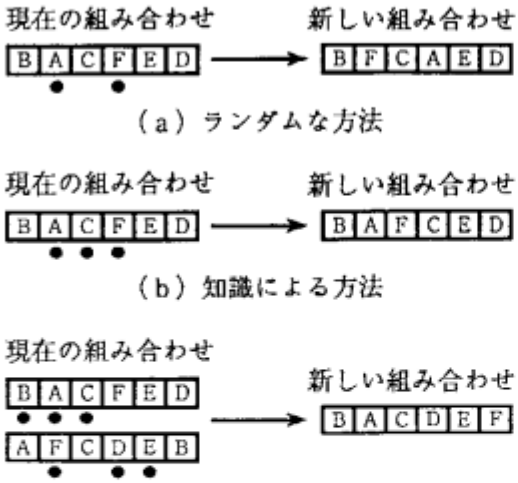


図 1: 新しい組合せの生成

GA では、前述したように、SA の温度パラメータようなものがなく、親遺伝子の選択方法や形質の悪い遺伝子の消滅方法が、最適化の間に変化がないので、最適化の最初から局所解に収束する危険や最適化の最後に、良質の解が消滅してしまう危険がある [10]。

しかし、RA では、SA で使用されている温度パラメータにより、最初に粗い探索を行ない、序々に最適解に近付けることが可能である。RA では、現在の組合せと新しい組合せのコスト差と温度パラメータにより、次の組合せを決定している。

2.3 ルールの選択

並列 RA では、上記の 3 つの新しい組合せの生成方法をルールとして、対等に扱っている。図 2に示すように、ランダムなルール、知識によるルールおよび遺伝的なルールとして、選択確率が決定される。直感的には、コストの低い組合せを生成できるルールの選択確率が高くなり、逆にルールを適用してもコストが高くなったり、ルールが発火しない場合は選択確率が低くなる。選択確率は、各温度ごとに動的に変更され、選択確率の決定方法は、

受理率による方法とコスト減少率による 2 通りの方法がある。ここでは、コスト減少率による方法を説明する。

2.3.1 コスト減少率によるルールの選択

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法では計算時間は考慮さ

番号	生成ルール	選択比率
1	ランダムなルール	0．15
2		0．05
3	知識によるルール	0．05
4		0．15
5		0．05
6		0．25
7		0．05
8	遺伝的なルール	0．15
9		0．10

図 2: ルールの選択確率

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法では、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。

本方法では、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 ($0.9 \leq B_i \leq 1$) 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。

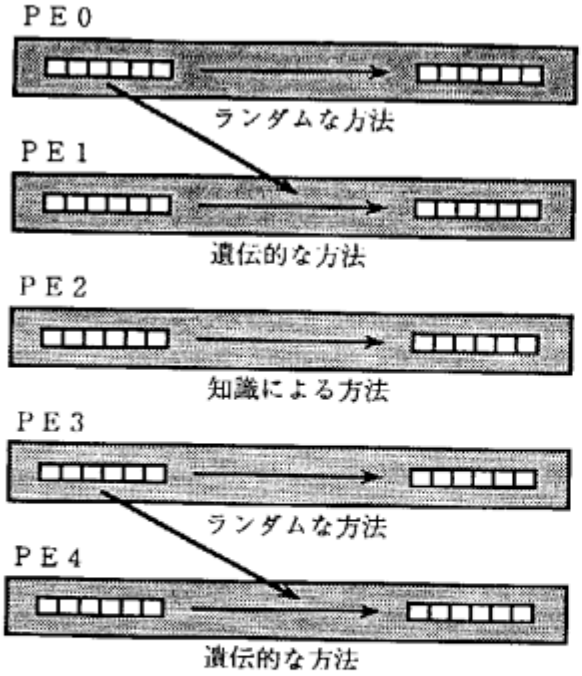


図 3: 並列処理の概念図

2.4 並列処理方法

図3は、以上の3点の方法による並列処理方法の概念図である。各PEは、1つの組合せを持ち、ランダムなルールと知識によるルールの場合は、PE自身の組合せに基づいて、新しい組合せを生成し、遺伝的ルールの場合は、隣接するPEから組合せを転送してもらい、新しい組合せを生成する。

図3では、PE0とPE3は、ランダムなルールにより、PE2は、知識によるルールにより新しい組合せを生成している。PE1とPE4は、それぞれPE0とPE3から転送してもらった組合せと自分のもっている組合せを親として、新しい組合せを生成する。

2.5 並列RAのアルゴリズム

並列RAは、ほとんどの疎結合型並列計算機で実現できるが、ここでは、（財）新世代コンピュータ技術開発機構が開発したMuti-PSI上で実現可能なアルゴリズムとして報告する。

並列RAアルゴリズムは、2種類の並列に実行されるアルゴリズムから構成される。主に温度管理を行うHOSTアルゴリズムと実際に最適化を行うRAアルゴリズムである。図4に示すように、HOSTアルゴリズムは、PE0に位置し、ストリームと呼ばれる通信路を各PEが持つRAアルゴリズムと接続する。隣接するRAアルゴリズム同志もまた、遺伝的ルールの組合せの情報を転送するためにストリームで接続される。

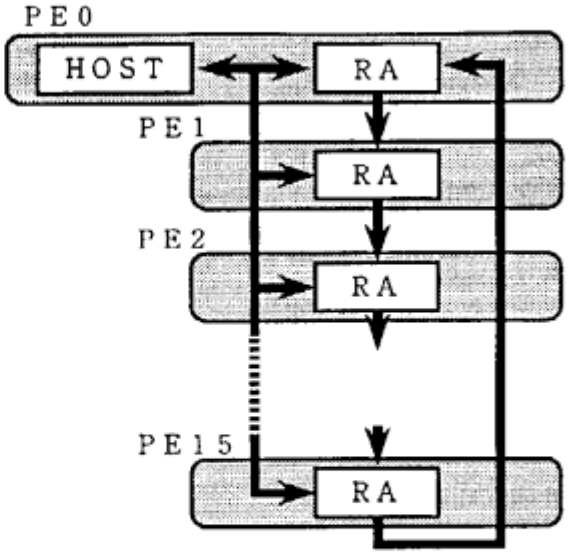


図 4: 並列処理の構成

以下に並列RAのアルゴリズムをHOSTとRAアルゴリズムにわけて、以下に示す。

```
HOST(j0 , T0 , R0) {
  S = j0; /* 初期解の設定 */
  T = T0;/* 初期温度の設定 */
  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 ) {
    RA(S,T,R);
        /* 各PEにTでRAを実行させる */
    while(すべてのPEから終了メッセージが
        かえってこない。) {
    }
  }
}

RA (S,T,R) {
  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);                 /* ホストに解を出力 */
}

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、並列RAアルゴリズムを簡単に説明する。まず、HOSTは、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し、各PEのRA分配する。Multi-PSIでは、並列タスクの実行が可能であるので自分自身にも分配する。ただし、自分自身のRAは、HOSTより実行優先順位を低く設定する。上記の記法では、明解に記述されていないが、初期解と初期選択比率は、最初にRAに転送された後、各RAは解と選択比率を独自に保持し、温度情報だけが、転送される。

次に、各RAは、ルール（ランダムなルール、知識によるルールおよび遺伝的なルール）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。遺伝的ルールの場合は、隣接するRAのもつ解を転送してもらう。しかし、このときRAに解が転送さるまでの待ち時間が生じるので、実際のインプリメントでは、各温度の処理の始めに選択確率に基づいて、隣接するRAに転送する解の頻度を決定し、ある変換回数ごとに送る側が主導で解を転送する。そして解が送られてきたRAは、そのとき遺伝的ルールを発火する。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??）、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。RAは、このような解の変換を十分繰り返した後、温度の終了メッセージと現在のコストをHOSTに伝え、自分自身は、同じ温度で変換を継続する。HOSTは、すべてRAからの終了メッセージを待ち、コスト変化がなければ、最適化の終了メッセージを各RAに伝える。コスト変化があれば、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ように、伝える。

以上のように、本方法ではRAの枠組の自然な拡張として、遺伝的ルールを取り込み並列処理方法を実現している。

3 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ハイレベル合成問題をとりあげ並列RAを適用した。そこで、本章ではハイレベル合成問題の概要と遺伝的ルールについて述べる。

3.1 ハイレベル合成問題

ハイレベル合成とは、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5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[16]。スケジュール表(図6)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並列RA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。

3.1.1 遺伝的ルール

ハイレベル合成問題におけるスケジュール表の遺伝的ルールを図7示す。その他の変換ルールについては、[3]で報告している。この問題では、式の縦方向の配置は、(time)と(register)コストに依存し、横方向は、(alu)、(bus)と(link)コストに依存している。そのため、縦方向にコストの低い配置の縦方向の式の並びと横方向にコストの低い配置の横方向の式の並びを組み合わせることで、よりコストが低い可能性がある配置を生成することができる。

図7の(a)は、縦方向にすぐれた配置である。実線がスケジュール表を示しており、点線内がスケジュール表から得られた縦方向の式の並びである。(1)、(2)、(3)、(4、5)、(7、8、6)の並びのコストが低いことになる。

(b)は、横方向にすぐれた配置である。点線内がスケジュール表から得られた横方向の式の並びである。(2、8)、(3、7)、(1、5、6)、(4)の並びのコストが低いことになる。

(c)は、(a)と(b)の配置から合成された配置である。この例では、合成したとき、式の重なりが生じていないが、式が重なったときは、同じ列の空白欄に移動するようになっている。

他のPEから配置を転送してきたとき、必ず、遺伝的ルールが発火するのではなく、転送されてきた配置が縦方向または横方向にすぐれた配置をもっているときのみ発火する。

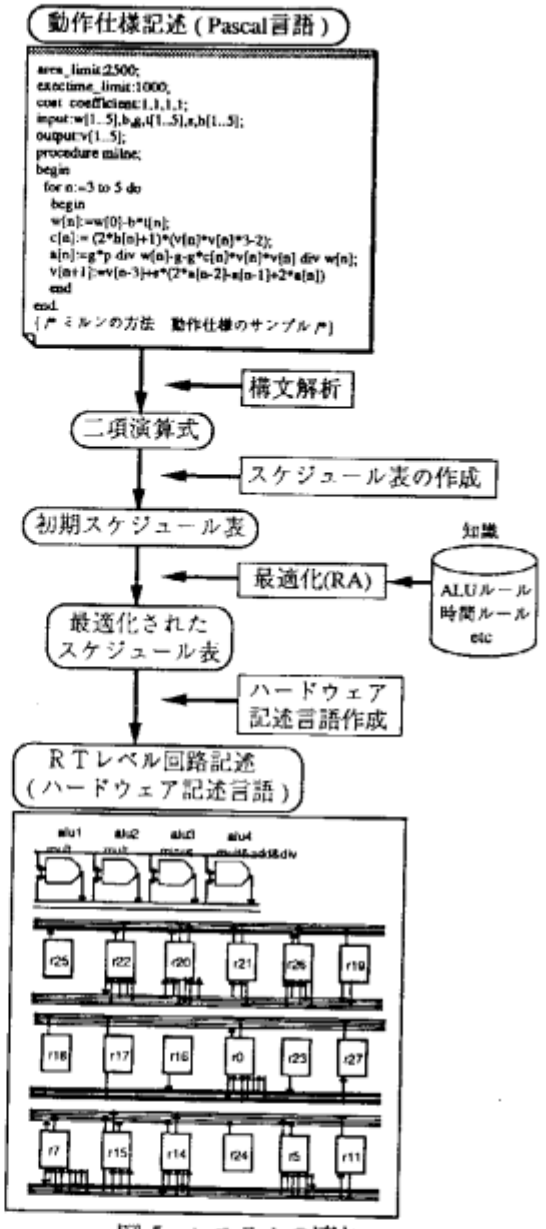


図5: システムの流れ

4 まとめ

我々は、RAの並列処理方法を提案した。逐次型RAは、以下の特徴をもっていた。

- 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。

	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 6: スケジュール表

- 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 に近づく。

並列 RA では、さらに次のような特徴を加えることができた。

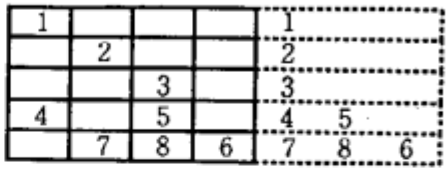
- 遺伝的ルールを RA のルールとした拡張として自然に取り込み、各 RA が持っている組合せの良質の部分を他の RA に伝達する協調的な枠組を作成した。
- 並列化することで、全体として複数の組合せをもつことになり、解の質がよくなる。
- 一般に交差オペレータは最適過程の終了時の Fine Tuning が困難である [10] という欠点を選択確率の動的な変化機構により、遺伝的ルールの適用回数が最適化の初期に多く終期に少なくなることが期待できる。

5 今後の課題

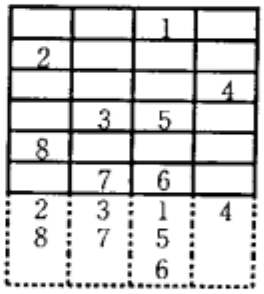
現在、Multi-PSI 上の KL1 言語を使用して、上記方法を試作中である。今後、定量的に上記の性質を評価する。また、我々が提案した別の並列処理方法との比較や共存方法も検討していきたい。

謝辞

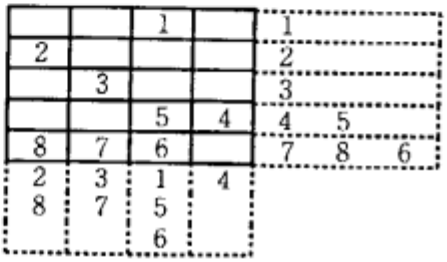
本研究は、(財)新世代コンピュータ技術開発機構 (ICOT) との再委託契約 (発住 S6202) に基づき行なわれたものである。新田克巳室長を始めとする ICOT 第 7 研究室の諸氏にご助言して頂いたことは本研究の推進に大変有用でありました。ここに記して感謝します。



(a) 縦方向にすぐれた配置



(b) 横方向にすぐれた配置



(c) 合成された配置

遺伝子ルール

図 7: 遺伝的ルール

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

[3] 加藤、荒木、野島、舘野、間藤: Rule-based Annealing、信学研報、AI92-33、1992.

[4] L.B.Booker, D.E.Goldberg and J.H.Hollad: *Classifier Systems and Genetic Algorithms* Artificial Intelligence 40, pp.235-282, 1989.

[5] David E.Goldberg: *Genetic Algorithms in Search, Optimization and Machine Learning* Addison Wesley 1989.

[6] John J. Grefenstette, *"Incorporating Problem Specific Knowledge into Genetic Algorithms"*, "Genetic Algorithms and Simulated Annealing", Pitman Publishing and Morgan Kaufmann Publishers, pp.42-60 (1987).

[7] Gilbet Syswerda, *"Schedule Optimization Using Genetic Algorithms"*, "Handbook of Genetic Algorithms", Van Nostrand Reinhold, pp.332-349 (1990).

[8] Monte Zweben, Eugen Davis and Michael Deale, *"Iterative Repair for Scheduling and Rescheduling"*, NASA Ames Research Center Report, (1991).

[9] S.Koakutsu, Y.Sugai and H.Hirata: "遺伝的要素を取り入れた改良型アニーリング法によるブロック配置手法 "信学論文, A, Vol.J73-A, No.1, pp.87-94, 1990.

[10] Donald E.Brown, Christopher L.Huntley and Andrew R.Spillane, *"A Parallel Genetic Heuristic for the Quadratic Assignment Problem*, ICGA89, pp.406-415 (1989).

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

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

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

[14] 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).

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

[16] 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).