ヒューマンインタフェース 41－6
（1992. 3. 2）

遺伝子アルゴリズムを用いた対話型図形配置

増井俊之

シャープ株式会社 技術本部

情報技術開発センタ

masui@shpcsl.sharp.co.jp

多数の図形を2次元平面上にある制約をもって配置するアルゴリズムは多数考案されているが、一般にこれらは複雑・単機能で多くの発見的手法を含んでおり、制約や配置結果をユーザが修正することはできない。このような問題を解決するため、ユーザが対話的に制約や配置結果を修正しながら自動的に図形の配置を行なうシステムGALAPAGOSを作成、評価した。GALAPAGOSは遺伝子アルゴリズムを対話的に拡張したアルゴリズムを用いて図形の配置を行なう。ユーザはシステムに配置の良否の評価基準を与えるだけで、配置方法を指定することなく自動的に配置が行なわれる。またユーザは配置の途中経過を監視して対話的に制約を修正することによって、完全自動によるよりも好ましい配置を得ることができる。

Interactive Graphic Object Layout with Genetic Algorithms

Toshiyuki MASUI

Information System R&D Center

Corporate Research and Development Group

SHARP Corporation

masui@shpcsl.sharp.co.jp

Conventional automatic graphic object layout algorithms are complicated, single-purpose, and often full of heuristics. They do not allow users to change their behavior during computation. We developed a general-purpose interactive graphic layout system GALAPAGOS based on genetic algorithms. Genetic algorithms are stochastic algorithms which simulate evolution through natural selection. GALAPAGOS is general-purpose because graphic objects are laid out not by specifying how to lay them out, but just by specifying the preferences for the layout. GALAPAGOS can not only lay out complicated graphs automatically, but also allow users to modify the constraints at run time so that they can tell the system their own preferences.

1 はじめに

複数の物体を空間内に適当な制約のもとに配置したいという要求は様々な場面で発生する。VLSIのレイアウト・建築や都市計画のレイアウト・布地の効率的裁断などはすべて配置問題の一種である。計算機のユーザインタフェースにおいても配置手法は重要である。複雑なデータ構造や大量のデータをユーザに示すためには計算機が人間にわかりやすい配置を計算しなければならないし、ウィンドウシステムのようなグラフィック画面を使用した対話的環境では常にユーザが認識・操作しやすい配置を画面に出力する必要がある。このような要求にこたえるため、計算機を使用して複雑な配置問題を解く様々な手法が従来より研究されてきている。また、制約を指定することは制約を解くことに比べるとはるかに楽な場合が多いため、"文字入力部はウィンドウの中心"といった制約記述により対話画面の配置が決定されるようなユーザインタフェースツールの研究が盛んである [14] [15] [22]。

制約解決の複雑さは制約の性質により大きく異なる。前述の例のような線型の制約は簡単に解くことができるため、ユーザインタフェースツールのようにユーザの操作に高速に反応しなければならないシステムでも使用することができるが、グラフの美しい配置のような複雑な制約をもつ問題においては、最適解を求めるアルゴリズムが存在しなかったり解くのに膨大な時間かかったりするのが普通である。このように最適解を実際上求めることが不可能な配置問題に対しては発見的手法(ヒューリスティクス)を使用して近似解を求める手法が従来よく用いられてきた。しかしこれには以下のような欠点がある。

- 手法の発見がむずかしい
- 制約がほんの少し変化しただけでも以前の手法が有効でなくなる場合が多く、柔軟性に欠ける

このため、制約を直接解こうとしないで統計的に解を得る手法が近年注目を集めている。統計的手法としてよく知られたものに焼きなまし法(アニーリング法, Simulated Annealing) [9] と遺伝子アルゴリズム (Genetic Algorithms) [4] [7] がある。これらの手法を用いると、配置手続きを考案しなくても、配置結果においてどの程度制約が満足されているか評価することさえできれば、制約の種類によらず、繰り返し計算を行なうことにより徐々に最適に近い解を計算していくことができる。

統計的手法は強力である反面、配置手続きを直接指定しないため、結果にユーザの意図を反映させることがむずかしい場合がある。得られた配置がユーザの気に入らない場合は制約を修正して再計算させなければならないが、その結果望む配置が得られる保証は無く、何度も試行錯誤しなければならない。

一方、美的感性が必要になるような配置問題では、完全自動配置は手動による配置に及ばない場合がある。

このような問題点を解決するため、遺伝子アルゴリズムを拡張し、ユーザが対話的に制約を変更しつつ配置計算を行なうことにより望ましい最終解を得ることのできるシステム GALAPAGOS (Genetic ALgorithm And Presentation-Aided Graphic Object layout System) を作成、評価した。本システムを使用すると、単体の遺伝子アルゴリズムによる自動配置では満足できる結果が得られない場合でも、得られた解を直接修正したり制約を計算途中で修正したりすることにより、自動配置と手動配置の両者の特長をもった望ましい解を得ることができる。まず大体の配置をシステムに計算させ、ユーザがそれを修正し、細かな最終処理を再度システムに処理させるなどの処理が可能になる。

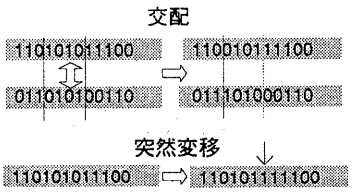

図 1: 遺伝子操作関数

2 遺伝子アルゴリズム

2.1 遺伝子アルゴリズム概要

遺伝子アルゴリズム (以降 GA と略記) は自然淘汰による遺伝子進化を模擬するアルゴリズムで、近年様々な最適化問題や機械学習の分野で幅広く応用されてきている。

遺伝子アルゴリズムにより最適化問題を解く手順は以下のようになる。まず計算に使用する一定長の文字列 (遺伝子) から解空間への写像を適当に決め、解の候補となるランダムな文字列の集合を用意する。要素の文字列それぞれについてその表現する解を計算し、最適化したい評価関数によりその評価値を求める。評価値の良いものは残し、悪いものは捨てるようにして残った文字列の間でその部分文字列の交換 (交配) 及びランダムな文字置換 (突然変移) を実行する。こうして得られた新たな文字列の集合について同じ操作を繰り返すうちに評価の良いものが蓄積されて最適に近い解を表現する文字列が残る。部分文字列の交換、文字置換のような操作を遺伝子操作関数と呼び、繰り返しの1周期を世代と呼ぶ。遺伝子操作関数を図 1に示す。

2.2 配置問題への適用

GA を配置問題へ適用した例としては VLSI のレイアウト [2] [17]、無向グラフの二次元平面上への配置 [11] [12] などが報告されている。前者においてはレイアウトの総面積や配線の長さなどが評価基準となり、後者においては交差する枝の数や対称性といった審美的要素が評価基準として使用されている。

単純な例として、GA を木構造の配置に適用した結果を図 2に示す。左側中段のグラフの X 軸は世代を示し、Y 軸はその世代における平均評価値を示している。配置を示す

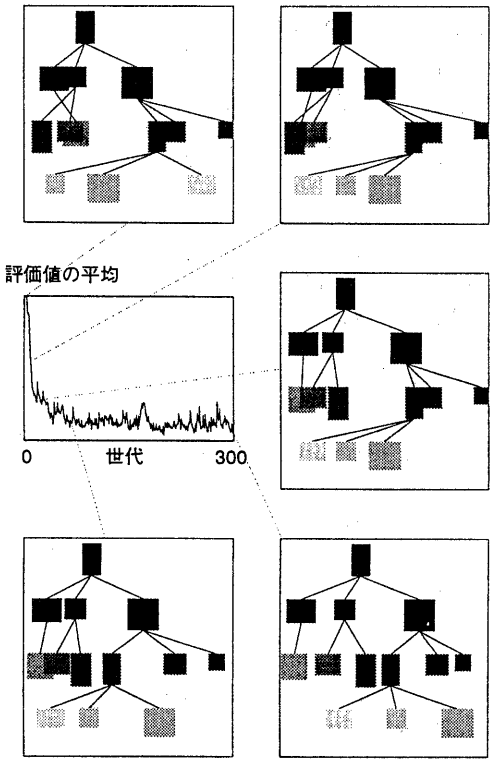

図 2: 木構造の配置への GA の適用

5個の図は、それぞれの世代における最適配置を示している。ここでは、1) 節点は重なりあわない, 2) 子節点の左右の順番が正しい, 3) 親節点の位置が自分の右端の子と左端の子の中間, という 3 個の制約を評価関数で使用している。図 2において、薄い色の節点が濃い色の節点よりも左側に位置するのが正しい順番である。

```
int evaluategene(char *gene)
{
    int val = 0;
    節点の位置,大きさの情報をgeneから取り出す;
    for(全ての節点){
        if(隣の節点と重なっている){
            val += (重なり幅) * 100;
        }
    }
    for(全ての節点){
        val += sqr(節点の位置 -
                   子節点の中心位置);
    }
    /* 他の評価基準の計算 */
    return val;
}
```

図 3: 木構造配置の評価関数

評価関数を図 3に示す。制約がよく満たされているほど小さい値が返される。

このように、制約を解くためのアルゴリズムを陽に指定しなくても求める結果を得ることができるのが GA の特長である[1]。しかし配置がシステムまかせでありユーザの要求を細かく反映させることができないという点は発見的手法を用いた場合と同様である。

3 対話的図形配置システム GALAPAGOS

GALAPAGOS は配置結果や制約をユーザが実行時に変更することのできる GA 配置システムである。遺伝子アルゴリズムのもつ特長に加え、ユーザが実行時に対話的に配置結果や制約を変更することがきるため、自動配置と手動配置の両者の特長を兼ね備えた良好な配置結果を得ることができる。

システムはまず予め与えられた制約にもとづいて計算を開始し、各世代における最適配置を表示する。ユーザは適宜その配置を修正したり制約を修正/追加したりすることができる。例えば、ある対象が好ましくない位置に配置されようとしていることにユーザが気付いた場合、ユーザはそれを別の望ましい位置に移動させて固定させることができる。また、あるふたつの対象の Y 座標を一致させた方が美しい配置になるとユーザが感じた場合、それらのが同じ Y 座標をもつという新しい制約を追加することができる。交配率や突然変移率のような GA のパラメタも途中で変更することができる。

GALAPAGOS は GENESIS システム [5] と同じアルゴリズムを使用している。GALAPAGOS の中心アルゴリズムを以下に示す。

```
initialize();
for(gen=0; gen < maxgen; gen++){
    checkevent();   // ユーザ入力チェック
    selectgene();   // 使用遺伝子選択
    mutate();       // 突然変移
    crossover();    // 交配
    evaluategene();// 遺伝子の評価
    measure();      // 最悪値の更新
    display();      // 結果の表示
```

selectgene() は Baker のアルゴリズム [1] にもとづいて遺伝子の選択を行なう。ここでは評価の最悪値が設定されており、それよりも評価の悪い遺伝子は捨てられて次の世代における計算に使用されない。遺伝子は selectgene() の後においてその数が評価値と最悪値の差に比例するように選択される。最悪値は各世代において更新される。

[1]木構造を美しく配置する問題はそう難しいものではなく、高速な配置アルゴリズムも各種提案されているが [10] [13] [21]、美しさの評価基準が変わった場合でも評価関数を変更するだけですむということが GA を使用する利点である。

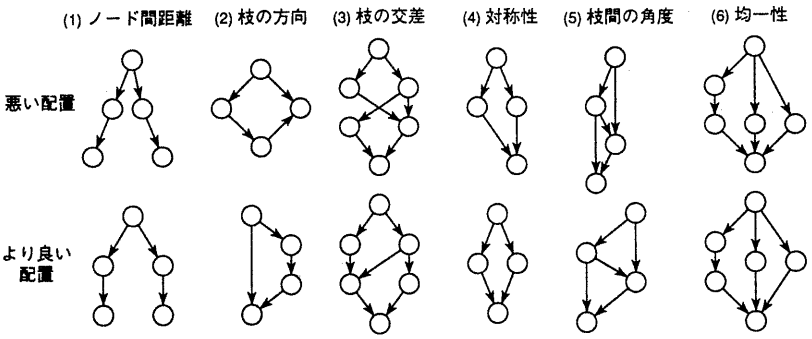

図 4: 有向グラフの配置における制約

checkevent () においてGALAPAGOSはユーザの要求をチェックし、要求がある場合は制御をユーザに戻して、配置結果やevaluategene () で使用される制約を修正できるようにしている[2]。ユーザはそのまま処理を続行してもよいし、修正後続行してもよい。配置結果を修正した場合は、新しい配置を示す遺伝子が遺伝子集合全体の1/3に戻されて処理が続行される。これにより修正結果が生き残るようになっている。

4 GALAPAGOSによる有向グラフの配置

本節ではGALAPAGOSを有向グラフの配置問題に適用した例について解説する。

4.1 有向グラフの配置問題

有向グラフとは節点の集合Nと枝の集合Eで構成されるグラフで、枝は節点を要素とする順序対(n, m)で表現される。図5は4個の節点と5本の枝をもつ有向グラフの例である。

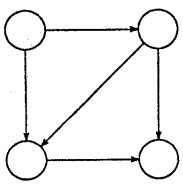

図 5: 4個の節点と5本の枝を持つ有向グラフ

節点や枝の数が多いとき、これらを見やすく紙や画面のような2次元空間に配置することは非常に困難である。見やすい配置を得るために例えば図4や下に示す制約を考えることができる。

1. 節点の間に充分な間隔があること
2. 枝の先頭は後尾よりも下方にあること
3. 枝の交差の数がなるべく少ないこと
4. なるべく図が対称であること
5. 同じ節点から出るふたつの枝の間の角度が小さすぎないこと
6. 節点が配置画面上に適当に分散しているいこと

これらの制約を解くための各種の配置アルゴリズムが提案されている[20]。しかし枝の交差の数を最小にするような節点の配置を求める問題はNP困難であり、他の多くの制約についても同様であるため、提案されているアルゴリズムはなんらかの発見的手法を使用している。例えばEadesと杉山[3]による方法では以下の手順で配置を行なう。

段階 1 節点を枝の向きでソートする
段階 2 グラフの頂上から下端の間の“層”の数の最小値を計算する
段階 3 同じ層内のふたつの節点間に枝が存在しないようにすべての節点をどれかの層に割り当てる。このとき節点がなるべく均一に各層に割りあてられるようにする。
段階 4 層をまたいだ枝が存在するときはそこにダミーの節点を導入し、そのような枝が無くなるようにする。
段階 5 各層内で節点を移動させ、枝の交差の数が最小になるようにする

ここで段階2, 3, 5はNP困難であり、いくつかの発見的手法が用いられている。

4.2 GALAPAGOSにを使用した有向グラフ配置

4.2.1 標準GAアルゴリズムの変更

GALAPAGOSでは、有向グラフの配置において、標準GAアルゴリズムに若干の変更を加えている。第1に、2.1節で述べたような文字列表現のかわりに整数表現を使用している。各節点の座標が整数配列に格納されており、これが遺伝子として使用される。交配は部分配列の交換、突然変移は乱数の配列要素への代入で実現される。第2に、反転演

[2]GALAPAGOSとGENESISのアルゴリズムの違いはcheckevent ()の有無のみである。

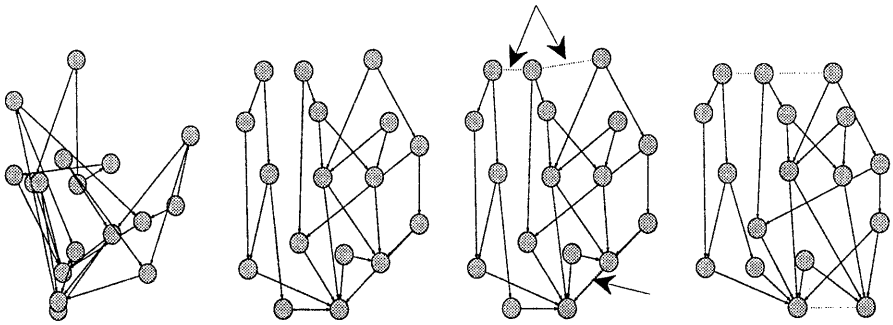

図 7: 配置の実行例

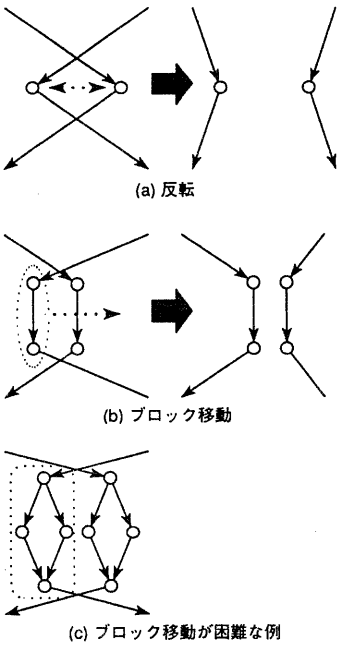

図 6: 反転とブロック移動

算が遺伝子操作関数に追加されている。反転演算とはふたつの節点の位置の交換である。図 6(a) は反転演算が配置においてうまく働く様子を示している。図 6(b) に示されるようなブロック移動も配置問題に適用可能であるが、図 6(c) に示されるようにうまくいかない場合も多く反転演算ほどの効果はないため使用していない。

4.2.2 配置に使用される制約

以下の制約がデフォルトで設定されている。

1. 枝の先頭は後尾よりも下に位置する
2. すべての節点間の距離がある定数値より大きい
3. 枝の交差の数がなるべく少ない
4. 節点を共有する枝の間の角度がある定数値より大きい

それぞれの制約に重みを示す定数が割り当てられており、$\sum_{\text{制約の種類}}$(制約の違反数 × 重み) が評価関数の値となる。以下に示す例では、重みの値として (3000, 400, 300, 400) が使用されている。例えば、枝の交差ひとつごとに評価関数の値に 300 が加算される。

これらの組み込みの制約の他に、以下の制約をユーザが対話的に追加することができる。

5. 指定した位置に節点を固定
6. ふたつの節点の X 座標が等しい
7. ふたつの節点の Y 座標が等しい

4.2.3 配置の実施例

現在の実装では、GA の実行及び得られた最適配置を表示するプログラム (GA Visualizer) と図形エディタは別プログラムになっている。初期データは図形エディタまたは他のプログラムにより GA Visualizer に渡される。GA Visualizer は GA を実行し、各世代における得られた最適配置を表示する。その結果を見たユーザが制約を追加したいと思った場合は再び図形エディタを使って制約を追加し、GA Visualizer に新しいデータを返す。

図 7に GALAPAGOS を使用した配置の流れの例を示す。まず有向グラフの接続情報がシステムに与えられ、システムは GA の実行を開始する。図 7(a) がシステムが最初に計算した配置である。何世代か経過した後かなり改善された配置 (b) が得られる。ここでユーザは、グラフの上部の 3 個の節点の Y 座標が等しく、下部の 2 個の節点の Y 座標も等しければさらに美しい配置となると考え、(c) の大きな矢印で示された 3 個の制約を追加する。新しい制約は点線で示

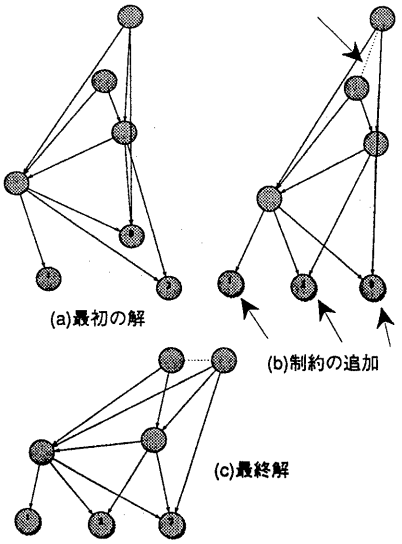

図 8: 位置の固定

されている。この後計算を続行することにより最終的に (d) の結果が得られる。

図 8に別の例を示す。ユーザは最初に得られた配置 (a) を修正し、3 個の節点につけられた番号が並ぶように (b) のように節点を固定する。固定された節点には影がついている。同時に上部のふたつの節点の Y 座標が等しくなるような制約も加えられている。この後計算を続行し、最終結果 (c) が得られる。

両方の例において同じ遺伝子数 (= 200)、交配率 (= 0.8)、突然変移率 (= 0.006) が使用されている。

5 評価

5.1 GALAPAGOS の特長

GALAPAGOS は、一般的な GA の特長に加え、ユーザの制約操作による細かい指示が可能であるという特長を持っている。これらをまとめると以下のようになる。

- 配置アルゴリズムが不要
  図 3のような評価関数を与えるだけで自動的に配置が得られるため、複雑な制約に対しても配置のためにアルゴリズムを考案する必要がない
- ユーザの好みを反映した配置
  制約を実時間に対話的に変更できるため、自動配置に手動配置の要素を加えてより望ましい配置を得ることができる。
- 制約の強さを指定できる
  4.2.2節において有向グラフ配置のための制約の重み付けの一例を示したが、この値を変化させると得られる結果は変化する。このように、重み付けの変更により得られる配置を微調整することができる。発見的手法によるアルゴリズムなどでは制約の強弱を扱うことは困難である。
- 柔軟である
  GALAPAGOS では矛盾した制約を指定した場合でさえ何らかの意味のある配置を得ることができる。これに対し、線型制約の解決システムにおいては矛盾する制約があると単に解けないという結果が得られるだけである。

5.2 異なる GA アルゴリズムの使用

GALAPAGOS で採用した GENESIS のアルゴリズムは基本的な GA であるが、現在様々な GA の改良手法が提案されている[3]。GALAPAGOS の GA 計算部は通常の GA に対話処理部を追加するだけであるため、より優れた GA に対応した GALAPAGOS を作成することは容易である。

5.3 評価関数の選択

制約の重みの変更の場合と同様、評価関数の作りかたにより挙動が大きく変化し、ユーザにとって意外な配置結果が得られることがある。これは一見問題のようにも見えるが、以下の理由で特長とも考えることができる。

- ユーザが想定していなかった意外な好ましい配置が得られることがある。
- いろいろな制約を動的に変化させて結果を試すことができるので、それらの関係から“好ましい”配置とはどういうことかについて新たな次元からの解析ができる。

もちろん評価関数が明確なものに関しては問題はおこらない。

6 関連研究と今後の課題

GA を有向グラフの配置に適用した例としては [6] [16] などがあるが、良好な結果は得られていない。無向グラフの配置に適用した例として [11] [12] がある。[11] ではコネクションマシン上に実装した並列 GA を使用し、4.2.1節で解説したブロック移動などの演算を追加して良好な結果を得ている。近年 VLSI レイアウトに GA が広く応用されつつあり [2] [17]、焼きなまし方と融合した方式についても研究が行なわれている [8] [23]。VLSI レイアウトへの応用に関しては [18] に詳しい。しかし以上のものはすべて全自動による配置手法であり、制約や配置結果を動的に変更することはできない。

TRIP2[19] は自動配置された結果を修正してもとの制約にフィードバックすることのできる制約グラフィック自動配置システムである。TRIP2 は配置結果をユーザが修正し

[3]遺伝子長を可変にしたもの・性を導入したもの・複数の独立進化集団を使用するもの・共生進化を模擬するものなど各種のものがある。

て定義にフィードバックできるという点でGALAPAGOSに似ているが、高速配置のためとフィードバックを可能にするために単純な制約しか使用することができず、また変更後の配置は再び最初から全自動配置をやり直さなければならない。

今後の課題として以下のようなものがあげられる。

- 現在は修正可能な制約として本文中で述べた3種類のもしか使用できないので、任意の式を指定できるように拡張する
- 図形エディタのGALAPAGOSへの組み込み
- より性能の良いGAアルゴリズムの使用
- 広い範囲の配置問題への適用、評価

7 結論

GAを核として、ユーザが配置結果を見ながら対話的に制約を修正することのできるシステムGALAPAGOSを作成し評価した。GALAPAGOSは、ユーザの意向を結果に反映させることのできない全自動配置システムと異なり、複雑な図形配置におけるGAの有用性及び対話処理の融合により、全自動配置と手動配置の両者の特長をあわせもった配置を行なうことが可能である。

謝辞

遺伝子アルゴリズムの有効性を紹介し、適切な助言をいただいた日本電気の北野宏明氏と、本研究を援助していただいているシャープ株式会社情報技術開発センタの河田亨所長及び千葉徹部長に感謝いたします。

参考文献

[1] James E. Baker. Reducing bias and inefficiency in the selection algorithm. In John J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pp. 14–21, Hillsdale, NJ, July 1987. Lawrence Erlbaum Associates, Publishers.

[2] J. P. Cohoon and W. D. Paris. Genetic placement. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 422–425, 1986.

[3] P. Eades and K. Sugiyama. How to draw a directed graph. *Journal of Information Processing*, Vol. 13, No. 4, pp. 424–437, 1990.

[4] David E. Goldberg. *Genetic Algorithm in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[5] John J. Grefenstette and Nicol N. Schraudolph. *A User's Guide to GENESIS 1.1ucsd*. Computer Science & Engineering Department, University of California, San Diego, La Jolla, CA, August 1990.

[6] L. J. Groves, Z. Michalewicz, P. V. Elia, and C. Z. Janikow. Genetic algorithms for drawing directed graphs. In Z. W. Ras M. Zemankova M. L. Emrich, editor, *Methodologies for Intelligent Systems 5, Proceedings of the Fifth International Symposium*, pp. 268–276. North-Holland, October 1990.

[7] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[8] Youngtak Kim, Youngjo Jang, and Myunghwan Kim. Stepwise-overlapped parallel annealing and its application to floorplan designs. *Computer Aided Design*, Vol. 23, No. 2, pp. 133–44, March 1991.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, No. 220, pp. 671–680, 1983.

[10] Kevin Knight. Treeprint: A program that draws parse trees. Technical Report CMU-CMT-87-MEMO, Center for Machine Translation, Carnegie Mellon University, September 1987.

[11] Corey Kosak, Joe Marks, and Stuart Shieber. A parallel genetic algorithm for network-diagram layout. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 458–465, UCSD, California, July 1991. Morgan Kaufmann Publishers.

[12] A. Markus. Experiments with genetic algorithms for displaying graphs. In *Proceedings of 1991 IEEE Workshop on Visual Languages*, pp. 62–67, Kobe, Japan, October 1991. IEEE Computer Society, IEEE Computer Society Press.

[13] Sven Moen. Drawing dynamic trees. *IEEE Software*, Vol. 7, No. 4, pp. 21–28, July 1990.

[14] Brad A. Myers, Brad Vander Zanden, and Roger B. Dannenberg. Creating graphical interactive application objects by demonstration. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pp. 95–104, November 1989.

[15] Dan R. Olsen and Kirk Allan. Creating interactive techniques by symbolically solving geometric constraints. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pp. 102–107, October 1990.

[16] Donald P. Pazel. A graphical interface for evaluating a genetic algorithm for graph layout. Technical Report RC14348, IBM Research Division, T.J. Watson Research Center, 1989.

[17] Khushro Shahookar and Pinaki Mazumder. A genetic approach to standard cell placement using meta-genetic parameter optimization. *IEEE Transaction on Computer-Aided Design*, Vol. 9, No. 5, pp. 500–511, May 1990.

[18] Khushro Shahookar and Pinaki Mazumder. Vlsi cell placement techniques. *ACM Computing Surveys*, Vol. 23, No. 2, pp. 143–220, June 1991.

[19] Shin Takahashi, Satoshi Matsuoka, Akinori Yonezawa, and Tomihisa Kamada. A general framework for bi-directional translation between abstract and pictorial data. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pp. 165–174, October 1991.

[20] Roberto Tamassia and Peter Eades. Algorithms for drawing graphs : an annotated bibliography. Technical Report CS-89-09, Brown University, Department of Computer Science, October 1989.

[21] John Q. Walker. A node-positioning algorithm for general trees. *Software – Practice & Experience*, Vol. 20, No. 7, pp. 685–705, July 1990.

[22] Bradley T. Vander Zanden. Constraint grammars - a new model for specifying graphical application. In *Conference on Human Factors in Computing Systems (CHI 89)*, pp. 325–330. ACM SIGCHI, Addison-Wesley, May 1989.

[23] 小坏 成一, 須貝 康雄, 平田 廣則. 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. 電子情報通信学会論文誌, Vol. J73-A, No. 1, pp. 87–94, January 1990.

付録

NeXT社のワークステーション上に実装されたGALAPAGOSのGA Visualizerの画面を図9に示す。ユーザは右下の“Stop”及び“Start”ボタンを押すことにより任意の時点で計算を中断、再開することができる。(図9では“Cont”と表示されているが、これは“Stop”を押した後の同ボタンの状態である。) GAの種々のパラメタは画面右のテキスト枠への入力で変更することができる。制約の修正は、まず現在の配置をファイルへセーブし、図形エディタで修正を行なった後もう一度それを読み込むという手順で行なわれる。

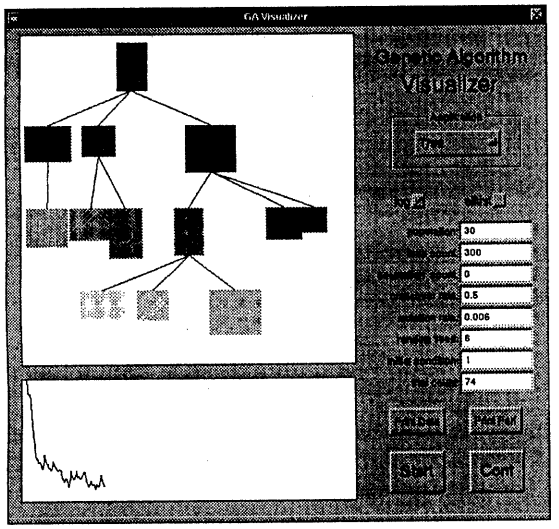

図 9: GA Visualizer