進化的学習機構を用いたグラフ配置制約の自動抽出

Evolutionary Learning of Graph Layout Constraints from Examples

増井 俊之*

Summary. We propose a new evolutionary method of extracting user preferences from examples shown to an automatic graph layout system. Using stochastic methods such as simulated annealing and genetic algorithms, automatic layout systems can find a good layout using an evaluation function which can calculate how good a given layout is. However, the evaluation function is usually not known beforehand, and it might vary from user to user. In our system, users show the system several pairs of good and bad layout examples, and the system infers the evaluation function from the examples using genetic programming technique. After the evaluation function evolves to reflect the preferences of the user, it is used as a general evaluation function for laying out graphs. The same technique can be used for a wide range of user-adaptive interaction systems.

1 はじめに

遺伝的プログラミングを用いた進化的学習機構により，ユーザの示す例のみからグラフ配置の評価関数を抽出する枠組を提案する．評価関数の既知な問題の多くは遺伝的アルゴリズムのような統計的手法を使用することにより最適に近い解を求めることが可能であるが，グラフ配置問題のように評価関数が自明でない問題にこのような手法を適用することは困難であった．本論文では，ユーザが自分が美しい配置と感じる例と悪いと感じる例の組をシステムに与えることにより，システムが遺伝的プログラミングにより配置の良否を決定する評価関数を抽出するシステムを紹介する．本手法は適応インタフェースの基本的枠組としても使用することが可能である．

2 グラフ自動配置問題

多数の図形を限られた領域に適当な制約のもとに効果的に配置したいという要求は多い． 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このような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視覚化する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．

* Toshiyuki Masui, シャープ株式会社 ソフトウェア研究所

文書整形システム TEX もこのような配置システムの一種である．TEX は文字・単語・パラグラフ等をある評価関数にもとづいて箱のように並べながら 2 次元領域に配置していく．TEX のユーザは配置アルゴリズムのパラメタをある程度操作することはできるが，配置戦略はプログラムとしてシステム内に深く組み込まれているため根本的に配置手法を変更することはできない．TEX と同様に，自動配置システムのほとんどにおいて配置アルゴリズムをユーザが変更することは不可能である．

自動配置アルゴリズムは以下の理由において構築が困難である．まず，配置しなければならないデータ構造が複雑であるほどそれを効果的に配置することが難しくなるため複雑な自動配置のアルゴリズムが必要になり，その作成も修正も困難になる．また，インタフェースにおける自動配置の場合特に，そもそもどのような配置が効果的なのかという基準が明確でないことが多いうえに，矛盾する基準が複数存在してその調整方法がよくわからない場合もある．

最初の問題に対しては，遺伝的アルゴリズム (GA)[6][8] や焼きなまし法 (アニーリング法, Simulated Annealing)[10] のような統計的手法を使うことが有効である．これらの手法を用いると，配置アルゴリズムを考案しなくても，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，時間をかけて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，配置の評価関数が明確でかつ時間の制約が少ない VLSI 配置システムにおいて特によく使われている [2][22][23].

配置の評価基準がわからないという問題に対しては例示によるアプローチ [3] が有望である．もしシステムがユーザの示す例のみから評価基準を推論することができれば，ユーザは基準を陽にシステムに指示する必要がなくなる． Myers は WYSIWYG エディタにおいて文書整形マクロをユーザの示す例のみから作成するシステムを提案している [19]． Myers のシステムでは，例えばユーザがセクションタイトルの例をひとつ描くだけで，システムにセクションタイトルの整形マクロを推論させることができる． Hudson はグラフ配置システムに少数の例を示すだけでシステムが汎化を行ない配置アルゴリズムを計算するシステムを提案している [9]．ユーザの示す少数の例のみからユーザの望む配置規則を推論することは困難なので，システムは考えられる配置規則を複数作成し，それぞれの規則の適用結果をユーザに提示して正しいものを選択させることにより候補数の爆発を防いでいる．また宮下らの IMAGE システム [18] は，配置データと配置結果の間の双方向変換機構を使いながら， Hudson のシステムと同様の機構を用いてシステムとユーザが対話しながら配置規則の候補を絞り込んでいく．これらのシステムでは，比較的単純な配置規則をユーザの示す例のみから導くことが可能であるが，多数のヒューリスティクスが使われておりシステム自体が複雑であることに加え，複雑な配置の判断基準を推論することができないという欠点がある．

前述のようなアプローチに対し，我々はユーザの示す例を使用して進化的学習機構により配置規則や制約を抽出する枠組を提案する．ユーザは美しい配置と悪い配置の例の組をいくつかシステムに提示することにより，システムが遺伝的プログラミング [13] の手法を用いて配置の評価関数を自動的に作成する．配置の評価関数は木構造の計算式で表現され，これがユーザの嗜好を反映するように進化することにより適当な配置規則が得られる．ユーザの嗜好を反映する評価関数を一旦得ることができれば，前述のような統計的手法を用いた配置システムを用いて任意のデータに適用することが可能になる．本論文ではこの手法を有向グラフの配置問題に適用した例について述べる．

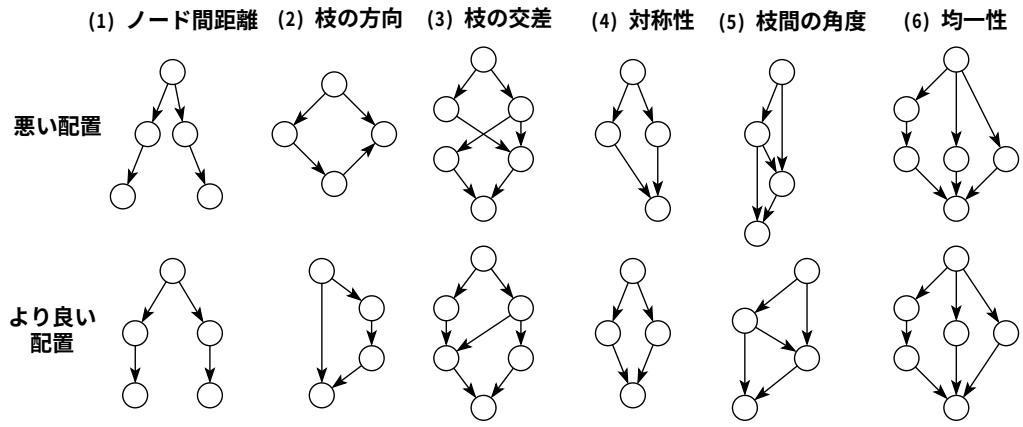


図 **1.** 有向グラフの配置における制約

3 有向グラフ配置問題

3.1 有向グラフ配置アルゴリズム

節点や枝の数が多いとき，グラフ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は図 1や以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 [24]．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 [5] 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．

このような手続き的配置手法では，配置の評価はアルゴリズムの奥深くに暗黙的に含まれているため，異なる評価基準に基づく配置を行なわせるためにはアルゴリズムを完全に書き換えなければならない．

3.2 グラフ配置問題への統計的手法の適用

近年，グラフ配置問題に GA のような統計的手法を適用する試みが盛んである [7][11][12][16][17][20]．これらのシステムでは，グラフ配置のよしあしを評価する関数は明示的に与えられ，ユーザは比較的簡単に評価関数を修正することができる．しかし適当な評価関数を決定することは容易ではない．例えば [16] では図 2に示す式が評価関数として用いられている．式が小さな値をとるほど良い配置を示す．

3000 * (上向きの枝の数) +
400 * (C_1 より短い枝の数) +
300 * (枝の交差数) +
400 * (枝間の角度が C_2 より小さいものの数)

図 **2.** [16] で使用されている評価関数

ここではいくつかの定数が使用されているが，これらを決定するまでには何回もの試行錯誤が必要になっており，また式の形や定数値を少し変更しただけでも得られる配置が大きく変わってくることがある．

評価関数の小さな違いにより配置結果が大きく異なる場合の例を図 3に示す．点 P を三角形 ABC の中の適当な位置に配置しようとするとき，式 $\overline{AP}+\overline{BP}+\overline{CP}$ を評価関数として用いてこれを最小にすべく GA を適用すると P は最終的に $\angle APB = \angle BPC = \angle CPA = 2\pi/3$ となる位置に落ちつくが，評価関数として $\overline{AP}^2+\overline{BP}^2+\overline{CP}^2$ を用いると P は ABC の重心に落ちつく．この例からわかるように，どのような配置が得られるかを評価関数から知ることは難しいし，好みの配置を得るための評価関数を作成することはさらに難しい．

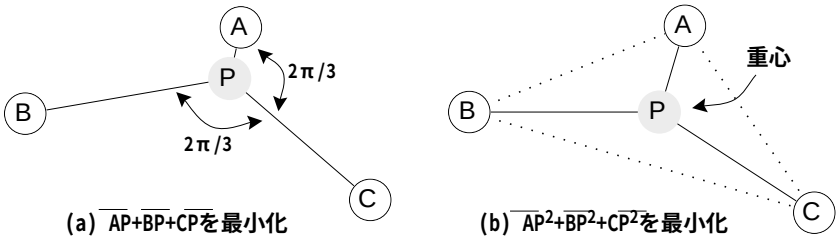


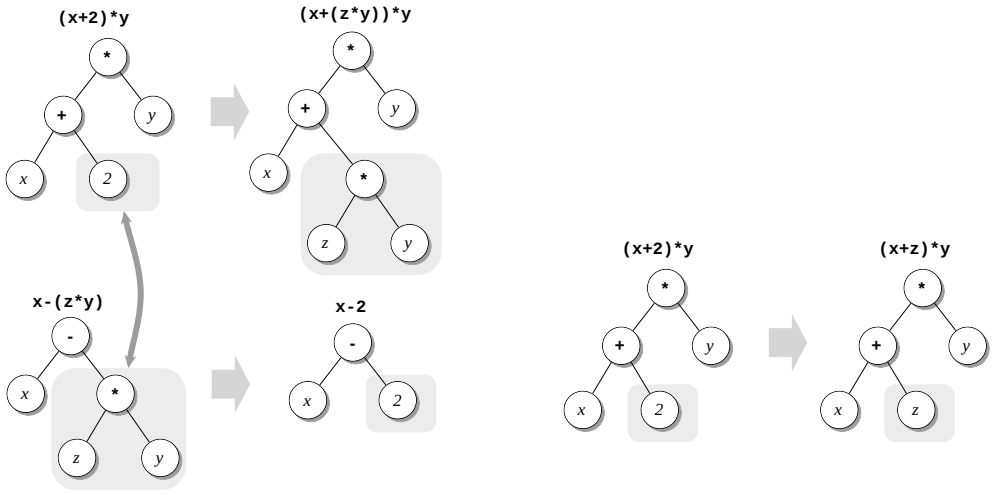
図 **3.** 2 種類の評価関数とその結果得られる配置

[16] のように対話的に遺伝的アルゴリズムを用いるシステムではシステムが好ましくない配置を出力してもユーザがそれを修正することができるが，全自動の配置システムではユーザは結果を受け入れるより仕方がないため大きな問題になる．いずれにしても，ユーザが何らかの方法により好みをシステムに伝えて評価関数を修正することができれば統計的配置システムはさらに有用になるはずである．

4 遺伝的プログラミングによる評価関数の自動生成

4.1 遺伝的プログラミング

遺伝的プログラミング [13] は遺伝的アルゴリズムをプログラムに適用したもので，ランダムに生成されたプログラム群がどれだけユーザの目的に近い動作をするかどうかを評価関数として淘汰による進化を行なわせることにより，最終的に目的のプログラムが得られるという手法である．プログラムは通常 Lisp の S 式のような木構造で表現される．最初はランダムにプログラム群の生成を行ない，それぞれのプログラムが目的の仕様とどれだけ近い動作をするかどうかを判定する．仕様に近い動作をするものほど次の世代に生き残る確率が高くなる．このようにして次世代に生き残るプログラムを選択した後，適当な確率で図 4(a) のようにプログラムの部分木の交換 (**交叉**演算) を行なう．また，いくつかのプログラムについては図 4(b) のように部分木を別のランダムなプログラム木で交換する (**突然変異**演算)．これらの操作を何世代も繰り返すうちに，与えた仕様に近い動作をするプログラムが最終的に生成される．



(a) 交叉演算　　(b) 突然変異演算

図 **4.** 遺伝的演算

4.2 ユーザの示す例からの好みの抽出

遺伝的プログラミングの手法を使うと，配置の評価関数をユーザの示す例のみから作成することが可能である．ユーザは同じ構造をもつグラフの美しい配置の例と悪い配置の例の組をシステムに与える．ある計算式を美しい配置のデータに適用した場合の値が悪い配置のデータに適用した場合よりも大きいとき，その式は美しい配置と悪い配置を見分けている可能性があるが，多く例の組に対して常にその式が美しい方の配置に対して大きな値を計算するようであればさらにその可能性は高くなる．ここでは N 個のグラフを使用し，各グラフ $i \in 1..N$ に対して美しい配置の例 G_i と悪い配置の例 B_i を与える．配置の評価関数 f を進化させるための評価関数として，$E(f) = \sum_{i=1}^{N} p(f,i)$ を使用する．ここで，$f(G_i) > f(B_i)$ のとき $p(f,i) = 1$ であり，それ以外では $p(f,i) = 0$

とする．もし全ての i に対して $f(G_i) > f(B_i)$ となれば $E(f) = N$ となる．遺伝的プログラミング手法を用いて，多くの例に対して $f(G) > f(B)$ を満たす関数 f をみつけることができれば，ユーザの好みを反映する関数 f を得られることが期待される．

5 実験

今回は評価関数作成の材料となる基本演算子及び引数として図 5に示したものを用いた．

演算子

add, sub, mul, div, abs,
sum, max, min, compare

引数

ノード位置の x/y 座標
枝の x/y 方向
交差している枝の数
ノード間距離
ノードから出る枝の間の角度の最小値

図 **5.** 評価関数に用いられる演算子と引数

最初はこれらの演算子や定数を用いて評価関数をランダムに生成させる．システムには 22 個の例の組を与えた ($N = 22$)．これらのいくつかを図 6に示す．全ての例 i に対し個体群中の全ての関数を適用して $f(G_i)$ と $f(B_i)$ を計算して $E(f)$ を求めている．

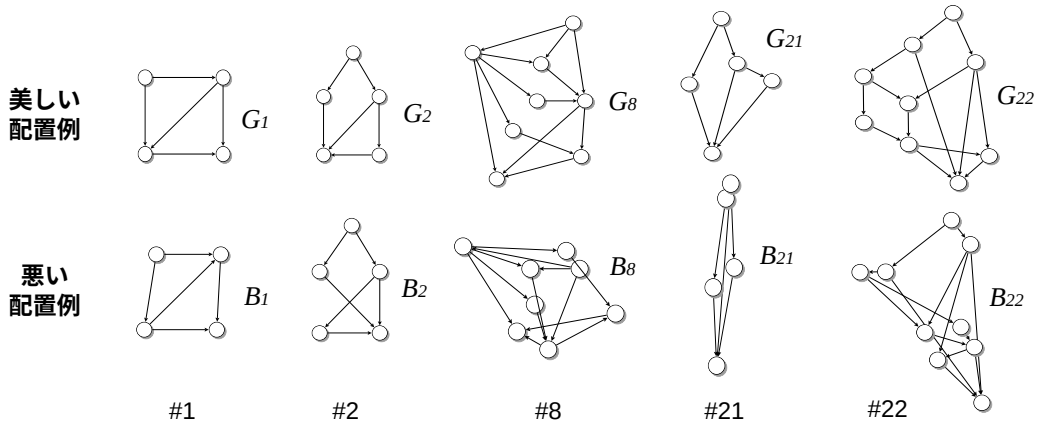


図 **6.** システムに与えられる美しい配置と悪い配置の組

世代の経過による $E(f)$ の平均と最高値の進化のようすを図 7に示す．ここで，個体数，交叉率，突然変異率はそれぞれ 600，0.8，0.005 である．交叉率や突然変異率は進化の効率に影響が大きい．個体数は多いほど少ない世代で進化がおこるが，それだけ世代毎の計算時間はかかる．

世代 1 では評価関数はランダムに生成されているので評価値の平均は 7 と低く最高でも 18 であるが，計算が進むにつれ評価関数は進化し，世代 102 において，全ての $i \in$

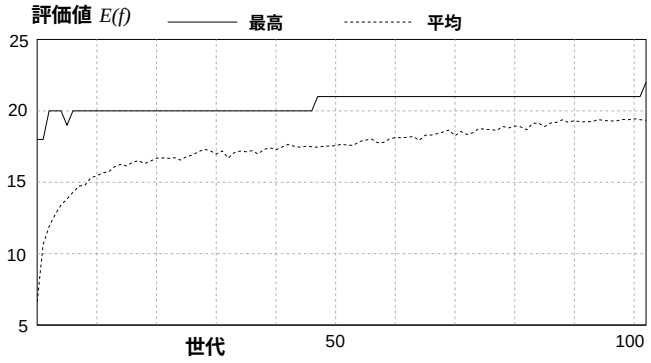


図 **7.** $E(f)$ の最高値と平均値

1..22 に対し $f(G_i) > f(B_i)$ となるような関数 f_b が得られている． f_b を Lisp プログラム風の型式で表現したものを図 8に示す.

(ADD (SUB (ADD (MUL (MUL (MUL (ADD (ADD (ADD SUM(diry) SUM(minangle)) (ADD 44.00 69.00)) (MUL 43.00 MIN(diry))) 5.00) (ADD (ABS MAX(minangle) MIN(dist)) (ADD (ABS 74.00 MIN(dirx)) (ABS 15.00 SUM(locx))))) SUM(minangle)) (MUL 12.00 (CMP (DIV 57.00 MIN(locx)) (CMP 94.00 MIN(intersec))))) (DIV (ABS (MUL (SUB SUM(locy) 27.00) (CMP 28.00 65.00)) 62.00) SUM(dirx))) (CMP (ABS (DIV 67.00 SUM(locy)) (CMP (ABS (ABS 73.00 (CMP 67.00 SUM(dist))) MIN(intersec)) (ABS MIN(dist) MIN(diry)))) MIN(diry)))

図 **8.** 計算された評価関数 f_b

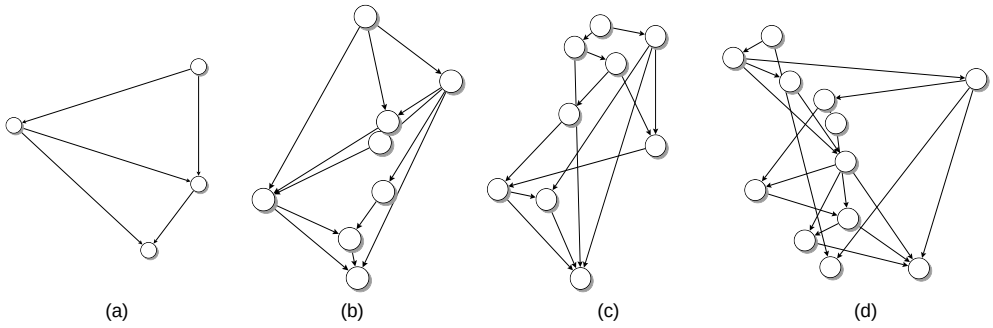


図 **9.** f_b を用いて計算した配置結果

得られた f_b を評価関数として用いて GA によるグラフ自動配置システム GALAPAGOS[16] によりグラフを自動配置した結果を図 9に示す． GALAPAGOS にはノードの接続情報及び f_b のみしか与えていないにもかかわらず，ユーザが例を通じて美しい配置の特徴として示したいくつかの特徴が結果に反映されていることがわかる．例えば図 9においてほとんど枝は下を向いているが，これは図 6において美しい配置の例として

示したグラフのほとんどの枝が下を向いていることを反映していると考えられる．

6 議論

6.1 適応型インタフェースの枠組

本システムでは，ユーザが例を沢山与えるほどシステムの性能が向上する．これは，適応型インタフェース [1][21] の構築に遺伝的プログラミングの手法が有望であることを示している．適応型インタフェースの構築の試みは，ユーザの操作を履歴情報やコンテキスト情報のみから予測しやすい場合のような単純な場合以外成功しているものが少ない [25]．さらに複雑なシステムではユーザモデルやアプリケーションモデルの知識を利用してシステムがユーザに適応することを試みているが，これらの知識はあらかじめて与えておかなければならないのでそれを構築することがそもそも難しいし，柔軟性も乏しい．これに対し，人工生命や機械学習の考えを用いて，システムがインタラクションを通じて徐々に進化 / 学習することにより適応型システムを実現しようという試みが近年盛んで有望である．例えば Maes の「学習型インタフェースエージェント」 [14] [15] はユーザの普段の行動や自分の提案に対するユーザの反応などをモニタしながらメモリベースの学習によりシステムが徐々にユーザに適応していく．このように，進化型システム / 学習型システムは適応インタフェースの重要な枠組として期待がもてる．

6.2 遺伝的プログラミングの有効性

遺伝的プログラミングの手法はランダムサーチに比べ本当に有効なのかという疑問が投げかけられることが多い． 5節の例では，目的の評価関数を得るのに 102 世代が必要であったが，ここで個体数は 600 だったので約 60,000 回評価関数の計算を行なったことになる．一方，遺伝的手法を使わず評価関数をランダムに生成して目的を満たすものが得られるまで繰り返すとすると，目的の関数が得られるまでの回数の期待値は $2^{22} \simeq 4{,}000{,}000$ となるので，少なくとも我々の例においては遺伝的手法により完全なランダムサーチよりもはるかに良い結果が得られたことになる．

6.3 対話的例示手法

システムへの例示により配置の基準を知らせる方式は，配置のアルゴリズムを作成することに比べれば簡単だとはいうものの，多くの例をシステムに提示するのは面倒である． 2節で紹介したように， Hudson のシステム [9] や宮下らの IMAGE システム [18] のように例の作成をシステムが支援することによりユーザは正しいものを選択するだけでよいというインタフェースが提案されているが，我々のシステムにおいても同様の手法が可能である．いくつかの例からシステムに評価関数を作成させた後，その評価関数と統計的配置システムにより別のグラフの配置を行なわせて，その結果とそれを人手でより良く修正したものの組を新たな例としてシステムに与えることにすればよい．実際，図 6の B_{21} は例 #1 から #20 までから作成された評価関数を用いて自動配置を行なったものであり， B_{22} は #1 から #21 までを用いて自動配置させたものである．

また，例の生成を完全にシステムにまかせて，ユーザは自分の好みのものを選択するだけというインタフェースも可能である．例えばシステムがランダムな基準を用いて配置したグラフを複数ユーザに提示し，ユーザはその中から最も自分の好みに近いものを選択するという作業を何度も繰り返すことにすればシステムはユーザの好みを抽出で

きるようになるであろう．このような手法は，面白い絵[1]や生物の形[2]などを進化的に生成させるための枠組としてよく用いられている．ランダムに近い配置から好みの配置を選ぶことはなかなか難しいと考えられるので今回このような方式は試みなかったが，分野によってはこのような手法が有効な場合があるかもしれない．

6.4 過適応と無意味な複雑化の防止

図 8の f_b は非常に複雑でありかつ無意味な計算要素をいくつも含んでいる．これは，評価関数を進化させるときにそれが与えた例を正しく判別できるかのみに着目して型式の単純さを考慮に入れなかったことが原因である． $E(f)$ に式の単純さの要素を加えれば複雑化はある程度防止できると考えられるが，目的の評価関数を得るまでの時間は余分にかかるようになる．

6.5 処理速度

本システムの最大の欠点は計算に長い時間がかかることである[3]．しかし遺伝的プログラミング / 遺伝的アルゴリズムは簡単に並列化可能であるし，計算機の処理速度の向上する速さや本システムの枠組の強力さから考えると，処理速度の問題は大きな問題ではないと考えられる．

7 結論

進化 / 学習型インタフェースシステムの例として遺伝的プログラミングを用いたグラフ配置制約抽出システムを提案した．現在のシステムはまだ単純で実用的ではないが，進化的手法が例示インタフェースや適応インタフェースを支援する枠組として有用であることは確かだと思われる．今後各種の進化的手法のインタフェースへの適用を試みていきたい．

参考文献

[1] D. Browne, P. Totterdell, and M. Norman, editors. *Adaptive User Interfaces*. Academic Press, London, 1990.

[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] Allen Cypher, editor. *Watch What I Do – Programming by Demonstration*. The MIT Press, Cambridge, MA 02142, 1993.

[4] Richard Dawkins. *The Blind Watchmaker*. W. W. Norton & Company, 1985.

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

[1] Karl Sims や畝見達夫らはこのような手法で芸術作品を生成する試みを行なっている．

[2] 生物 / 進化学者の R. Dawkins が著書「ブラインド・ウォッチメーカー」 [4] で紹介した BioMorph システムは，システムがランダムに生成した「虫」の形態の中からユーザが好みのものを選択することを繰り返すという人為淘汰によりそのユーザの好みの虫を生成するゲームである．各世代においてシステムは現在の虫を変異させた次世代の虫の侯補を 9 個ユーザに提示し，ユーザがそのうちひとつを選択することを繰り返すことによりだんだん虫の形態が複雑になっていく．

[3] 5節の例を 68040 (25MHz) NeXTstation の Objective-C で計算した場合，評価関数が求まるまでに数分を要する．

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

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

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

[9] Scott E. Hudson and Chen-Ning Hsi. A synergistic approach to specifying simple number independent layouts by example. In *Proceedings of ACM INTERCHI'93 Conference on Human Factors in Computing Systems (CHI'93)*, pp. 285–292. Addison-Wesley, April 1993.

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

[11] Corey Kosak, Joe Marks, and Stuart Shieber. New approaches to automating network-diagram layout. Technical Report TR-22-91, Center for Research in Computing Technology, Harvard University, Cambridge, MA 02138, 1991.

[12] 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, August 1991. Morgan Kaufmann Publishers.

[13] John R. Koza. *Genetic Programming*. The MIT Press, Cambridge, MA, 1992.

[14] Robyn Kozierok and Pattie Maes. A learning interface agent for scheduling meetings. In *Proceedings of the 1993 International Workshop on Intelligent User Interfaces*, pp. 81–88. ACM Press, January 1993.

[15] Pattie Maes. Learning interface agents. In *Proceedings of the 1994 Friend21 International Symposium on Next Generation Human Interface*, February 1994.

[16] Toshiyuki Masui. Graphic object layout with interactive genetic algorithms. In *Proceedings of the 1992 IEEE Workshop on Visual Languages*, pp. 74–80. IEEE Computer Society Press, September 1992.

[17] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.

[18] Ken Miyashita, Satoshi Matsuoka, Shin Takahashi, and Akinori Yonezawa. Interactive generation of graphical user interfaces by multiple visual examples. In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'94)*. ACM Press, November 1994.

[19] Brad A. Myers. Text formatting by demonstration. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'91)*, pp. 251–256. Addison-Wesley, April 1991.

[20] 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.

[21] M. Schneider-Hufschmidt, T. Kuhme, and U. Malinowski, editors. *Adaptive User Interface – Principles and Practice*. North-Holland, Amsterdam, 1993.

[22] 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.

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

[24] 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.

[25] 増井俊之. 適応 / 予測型テキスト編集システム. In *Proceedings of the Workshop on Interactive Systems and Software (WISS'94)*. ソフトウェア科学会, December 1994.