“viewPP”: グラフ構造とアニメーション表現に基づくプログラム実行の視覚化

“viewPP”:Visualization of program exectuion based on the animated graph structure

南雲 淳† 田中 二郎††
Jun NAGUMO Jiro TANAKA

†筑波大学 大学院修士課程理工学研究科
Master’s program in Science and Engineering Univ. of Tsukuba
††筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics Univ. of Tsukuba

概 要

並列論理型言語の実行を分りやすく可視化するには，グラフ構造とアニメーション表現が有効である．本研究では，これらの概念を用い，さらにグラフ自動描画アルゴリズムのアニメーション的表現向けの改良を行って，並列論理型言語の実行を可視化するビジュアルプログラミングシステム “viewPP” を作成し，グラフ構造のアニメーション的変形による実行の表現の有用性を確認した．本論文では，アニメーション的表現向けのグラフ描画アルゴリズムの改良と，大きなプログラムを見やすく表現するためのフィッシュアイ表示の組み込みについて述べる．

1 はじめに

論理型言語を 2 次元や 3 次元のグラフィックを用いて可視化する試みが数多く行われている [CP85] [EB88] [Tan94b] [STTS96].

我々は以前から，無向グラフ構造を用いて並列論理型言語 GHC [K.U85]を視覚化するビジュアルプログラミングシステムである “PP” [Tan94a] を作成してきている．

本論文では，GHC のプログラム実行の視覚化を行うシステムである “viewPP” の作成について述べ，並列論理型言語の実行の可視化へのグラフ構造の変形のアニメーション的表現の利用およびその実現のための無向グラフ自動描画アルゴリズムの改良について述べる．

また，ビジュアルプログラミングシステムで大きなプログラムを実行する際の問題点について述べ，解決法を示す．

2 “viewPP” の制作

“viewPP” は，並列論理型言語の実行をグラフ構造を用いて可視化するシステムである (**図 1**).

並列論理型言語を無向グラフ構造として可視化するには，グラフの要素 (辺および節) に，アトムや変数と言ったプログラムの要素を対応づける必要があるが，基本的には「述語を節に，論理変数を辺に」対応づけるのが良い [NT94].

“viewPP” において実際に行った対応づけを**表 1**に示す.

この対応づけによって，並列論理型言語のプログラムはすべてグラフ構造へと変換して視覚化することが可能である．

ここで，論理変数を辺で表現することは，特にメリットが大きい．従来のテキストベースの表記では全ての変数に変数名を付ける必要があり，多数の述

語を変数で接続してデータの受渡しなどを行うスタイルである並列論理型言語のプログラムの可読性を下げる原因の一つとなっていた．しかし，これを辺という図形で表現することによって，述語同士の単純な接続として，視覚的に表現できるようになる．

静的なプログラムはこれにより視覚化が可能であるが，述語のリダクションやユニフィケーションをグラフ構造の変形によって表現すれば，プログラムの実行を視覚化することも可能になる．本研究では，この変形をスムースなアニメーション的表現を用いることによって実際にプログラム実行の視覚化を行った．

無向グラフ構造を見易い形に配置するためには，既存のグラフ自動描画アルゴリズムを使うことができる．“viewPP” の作成にあたっては，無向グラフ自動描画アルゴリズムの一つである Eades スプリングモデル [Ead84] を改良したものを用いた．改良点は，Eades のアルゴリズムの反復法の計算過程で得られる節や辺の配置を順次表示していくことにより，そのままグラフ変形のスムースなアニメーション的表現が得られるようにした点である．

Eades のアルゴリズムでは，各節の間に引力あるいは斥力を定義した力学系を考え，その力学系の釣合を反復法で求めることによって無向グラフの配置を求めるが，斥力の定義式を変更することによって計算過程を順次表示すればそれがそのままスムースなアニメーション的表現となるようにした．

グラフ描画アルゴリズムの改良について，具体的な手法を述べる．

まず，従来の Eades のアルゴリズムでは，以下のようにしてグラフの配置を求める．

まず，ある節に注目して，その節以外の全ての節との間に働く「力」をそれぞれ計算し，合計をとる．その力は，辺によって直接接続された節の間を f_s，そうでない節の間を f_r とすると，

$$f_s(d) \quad = \quad C_s \log \frac{d}{d_0}$$

表 1　グラフ構造の要素とプログラムの要素との対応付け

グラフ構造の要素	プログラムの要素
辺	論理変数
節	アトム・述語・ファンクタ

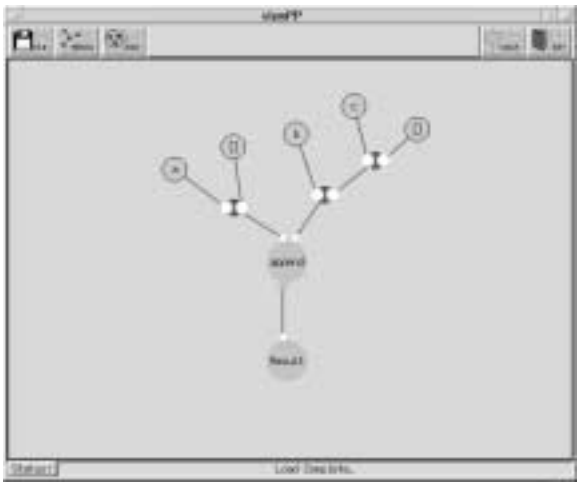

図 1　viewPP 画面

$$f_r(d) \quad = \quad C_r \frac{1}{d^2}$$

のように定義される．これの斥力定義関数を以下の通りに変更する．

$$f_r(d) = \begin{cases} C_{r'} & (d < d_0) \\ C_r \frac{1}{d^2} & (\text{Otherwise}) \end{cases}$$

従来の Eades のアルゴリズムでは一ステップあたりの節の移動量が大きくなりがちであるせいでアニメーション的表現に適したものでなかったものが，この改良によってステップ毎の節移動量を小さく押さえることができ，計算の経過を順次表示することによってスムースなアニメーション的表現ができるようになっている．

各ステップ毎の各節の移動量平均を改良前のアルゴリズムと改良後のアルゴリズムで比較した結果を**図 2**に示す (右上に示された 6 つの節を持つグラフについて計測した結果である)．

グラフの横軸は移動開始からのステップ数である．今回のアルゴリズムの改良は，移動開始からの数ステップの間の節の移動の不自然さを解消するのが目的であるので，グラフは移動開始から 15 ステップまでについて示してある．縦軸は，各ステップ毎の節の移動量の平均 (単位: ピクセル) である．

初期配置は，各節を半径 20 ピクセルの円周上に置くものとし，節間理想距離 $d \quad = \quad 150/R$(R : グラフの半径) として計測を行なった．

グラフを見ると，改良前では移動開始から数ステップの間の各節の移動量が大きいが，改良した後ではこの部分が平均化されており，アニメーション的表現として改良前より適切なものになっていると

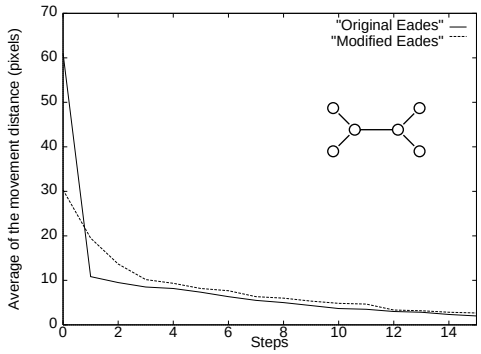

図 2　各ステップ毎の節の移動量の比較

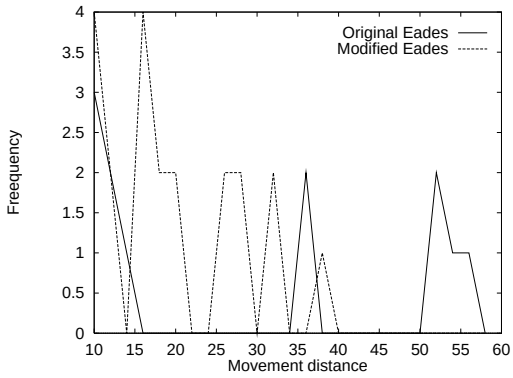

図 3　各ステップ毎の移動距離の度数分布

言える．

このグラフに対する各節のステップ毎の移動量について度数分布を調べた結果を**図 3**に示す．グラフの横軸は 1 ステップ毎の移動量 (単位: ピクセル) であり，縦軸が度数である．

アニメーション的表現の見栄えに対する影響が大きい 10 ピクセル以上の移動についての度数を示している．

改良前のアルゴリズムでは，一度に 40 ピクセル以上動く場合が多いが，改良後ではそのような動きはなく，全ての移動が 1 ステップ当り 40 ピクセル以内になっていることが読みとれる．この結果は改良の効果が現れてよりアニメーション向けのアルゴリズムになっていることを示している．

3　問題点とその解決

現在の “viewPP” において解決すべき問題として考えられているのは，大きく以下の 2 点である．

- 現在のままでは，大きなプログラムへの適用が困難である
- 入力やデバッグの手法をどうするか

今回は特に，前者についてグラフ描画アルゴリズムの改良により表示を工夫するというアプローチによって解決を図る．

3.1　大きなプログラムへの適用

現在の “viewPP” では，述語の一つ一つでは小さな仕事しかできないので，これを組合せて大きな仕事をしようとすると画面内の節や辺の数が膨大になったり，辺の交差が増えたりすることから，見づらいものになってしまう．

一般に，ビジュアルプログラミングシステムでは「プログラムの規模が大きくなると，必要なスペースも大きくなり，見づらくなる」という問題がある．

この問題を解決するためには，以下の 2 つのアプローチが考えられる．

- 述語一つ一つを高機能化する
- 画面内の部分的な拡大 / 縮小により，注目されるべき部分を強調表示するような表示方式の工夫

前者の方法では，一つ一つの述語を高機能化し，画面内に現れる述語数を減らすことによって，見やすさの向上を図るものであり，後者の方法は，画面内のプログラムの中で注目したい部分を選びその部分を強調し，同時に他の部分の表示を簡略化することによって見やすさを向上させるものである．

前者の方法を用いて述語一つ一つの抽象度を上げることにすれば画面内の節や辺などの図形の数を減らすことは可能ではあるが，抽象度を上げすぎるとその述語がどのように動いているかユーザが知りたくなったときにある述語を展開してしまえば結局節や辺の数は増えてしまう．

後者の方法，つまり，注目部分を拡大してそうでない部分の表示を簡略化する場合には，フィッシュアイ方式 [W.F86]などの手法を取り入れることが考えられる．

フィッシュアイを用いることによって，全体を見ることができるようにする全体性と，自分の見たい部分を詳しく見る詳細性を両立することが可能になる．

先述の通り，グラフ変形のアニメーション的表示のためにグラフ自動描画アルゴリズムの改良を行なったが，これにさらにフィッシュアイによる変形・表示を自動的に行えるようなアルゴリズムを組

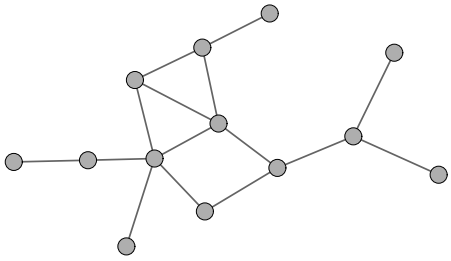

図 4　従来の Eades での再配置

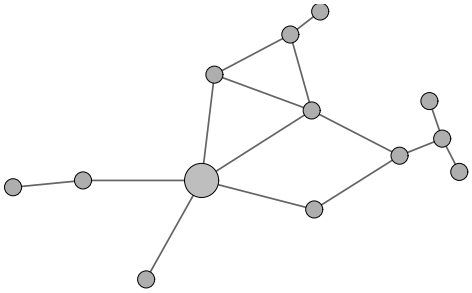

図 5　改良後のアルゴリズムでの再配置

みこむことにより，大きなプログラムに対応した表示の改善を図る．

先に上げた Eades のアルゴリズムにおいて，節の間の理想的な距離を定める d_0 を変更する．

今注目したい節 (拡大表示したい節) と再配置のために移動させようとしている節の間の距離 (2 つの節の間にある辺の数の最小値) を l，注目する節に接続する辺の長さを $d_{\max}$ ，さらに a 「辺の長さの減少の度合を表す」定数，b をグラフの半径に比例した定数として

$$d_0 \quad = \quad d_{\max}\left\{1 - \frac{1}{1+e^{-a(l-b)}}\right\}$$

のようにする．注目したい節に接続する辺は長くなり，そこから遠い節に接続する辺は短くなる．これにより，擬似的にフィッシュアイが実現できる．従来の Eades のアルゴリズムで再配置したグラフを**図 4** に，フィッシュアイを擬似的に組みこんだアルゴリズムによるものを**図 5**に示す ($a = 1.5, b = 2$ とした)．注目している節 (図では少し大きく表示されている) の周辺の辺が長く，そこから遠くの節の間の辺は短くなっており，擬似的なフィッシュアイ表示になっていることがわかる．

4　まとめ

ビジュアルプログラミングシステム “viewPP” の制作を通して，並列論理型言語の実行の可視化に「述語やアトムを節，変数を辺とする」無向グラフ構造と，その変形のアニメーション的表現が有効であることを確認し，さらにグラフ構造の変形のスムースな表現のためのアルゴリズムの改良を行って実際のシステムへの組み込みを行った．

また，多数の節や辺を見やすく表示するためにフィッシュアイ表示に注目し，グラフ描画アルゴリズムへの擬似的なフィッシュアイ表示の組み込みを試みた．

今後は，グラフ描画アルゴリズムへのフィッシュアイの組み込みについて有効性の検証を行い，実際に “viewPP” へ組込み，単一フィールド上での直接操作によるビジュアルプログラミングシステムへと発展させる．

参考文献

[CP85] P T. Cox and T. Pietrzykowski. Lograph: A graphical logic programming language. In *Proceedings of IEEE COMPINT 85*, pp. 145–151, 1985.

[Ead84] P. Eades. A heuristics for graph drawing. *Congressus Numeranitium*, Vol. 42, pp. 149–160, 1984.

[EB88] M. Eisenstadt and M. Brayshaw. The transparent prolog machine (tpm): an execution model and graphical debugger for logic programming. *Journal of Logic Programming*, Vol. 5, No. 4, pp. 277–342, 1988.

[K.U85] K.Ueda. Guarded horn clauses. Technical report, ICOT Technical Report TR-103, 1985.

[NT94] 中野勝次郎, 田中二郎. ビジュアルプログラミングシステムにおけるモデルの視覚化アルゴリズム. 竹内彰一（編）, インタラクティブシステムとソフトウェア II 日本ソフトウェア科学会 WISS'94, pp. 205–214. 近代科学社, 1994.

[STTS96] 志築文太郎, 豊田正史, 高橋伸, 柴山悦哉. ビジュアル並列プログラミング環境 klieg: プロセスネットワークパターンを利用した再利用性の向上と実行表示の効率化. 田中二郎（編）, インタラクティブシステムとソフトウェア IV 日本ソフトウェア科学会 WISS'96, pp. 81–90. 近代科学社, 1996.

[Tan94a] Jiro Tanaka. Visual programming system for parallel logic languages. In *Workshop on Parallel Logic Programming and its Program Environments, the University of Oregon*, pp. 175–186, 1994.

[Tan94b] 田中二郎. 並列論理型言語 ghc のビジュアル化の試み. 竹内彰一（編）, インタラクティブシステムとソフトウェア I 日本ソフトウェア科学会 WISS'93, pp. 265–272. 近代科学社, 1994.

[W.F86] G W.Furnas. Generalized fisheye views. In *Proc. CHI '86*, pp. 16–23, 1986.