4ZC-02

図形表現を用いた可視化によるプログラム実行状況の解析
Analysis of Program Excution Using Diagram Animation

神山 拓哉 †　　六沢 一昭 ‡
KAMIYAMA Takuya　　ROKUSAWA Kazuaki

1. はじめに

本稿では, 図形表現を用いた可視化による, プログラム実行状況の解析について述べる.

本可視化は, 頂点の位置が変数の値に対応した図形の描画によって実現する. プログラムの実行によって変数の値が変化すると, 対応した頂点の位置も変化するため, 図形が変形する. このように, プログラムの実行状況は図形の変形によって表現される.

図形が変形していく様子は印象に残りやすく, プログラム実行状況の全体像を捉えやすくなると考えられる.

この可視化プログラムを用いて, 研究室で過去 10 年間で開発されたプログラムを可視化し, 解析を行った.

2. 可視化の実現

2.1 処理の流れ

可視化対象プログラムに, 変数の名前や値などの情報を出力するためのコードを追加し, ログ出力プログラムに変換する. このログ出力プログラムを実行することで, 実行ログが得られる. 可視化プログラムはこの実行ログを元に可視化を行う. 可視化の処理の流れを図 1 に示す.

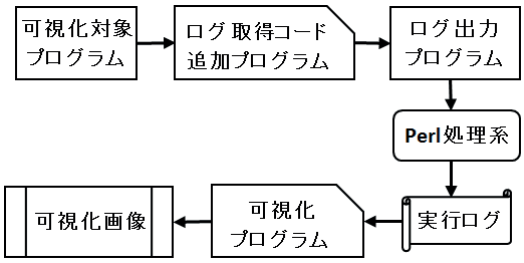


図 1: 処理の流れ

2.2 図形の形成

頂点の位置が変数の値に対応した図形を描画する.

頂点

頂点の原点からの距離は, 変数の値の対数値に比例する. また, 値の符号は頂点の色で表す. 値が文字列の場合は, 文字コードの総和を値とする.

2.3 可視化によって得られる情報

可視化によって描画される図形は, いつ, どの変数に, どのような値が入ったかを表す. そのため, 本可視化により, 変数の値の代入タイミング, 変数の更新タイミング, 更新の頻度について知ることができる.

3. 表示画面

可視化プログラムの表示画面例を図 2 に示す. 画面左側は可視化領域である. 画面の右側は変数情報領域と検査表示領域である.

可視化領域

変数の値に対応した頂点を持つ図形が描画される. 頂点に付随する番号は, 変数情報領域の番号と対応する.

変数情報領域

変数の番号, 変数名, 変数の値, 代入行の番号, 代入回数が表示される.

検査表示領域

本可視化プログラムには, 任意の位置で特定の条件を満たしているか調べる検査機能がある. この検査機能を使用していた場合にこの領域が表示され, 検査機能を指定した行番号, 条件式, 真偽が表示される.

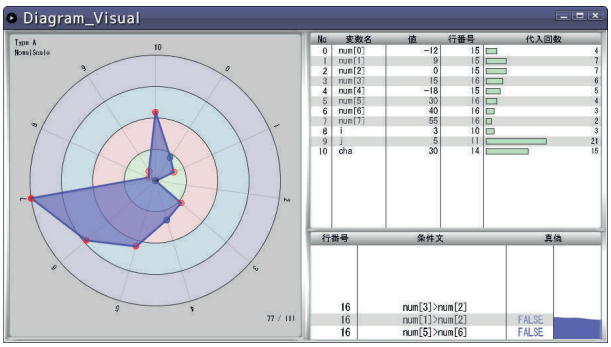


図 2: 可視化プログラムの表示画面例

4. 可視化プログラムの実行例

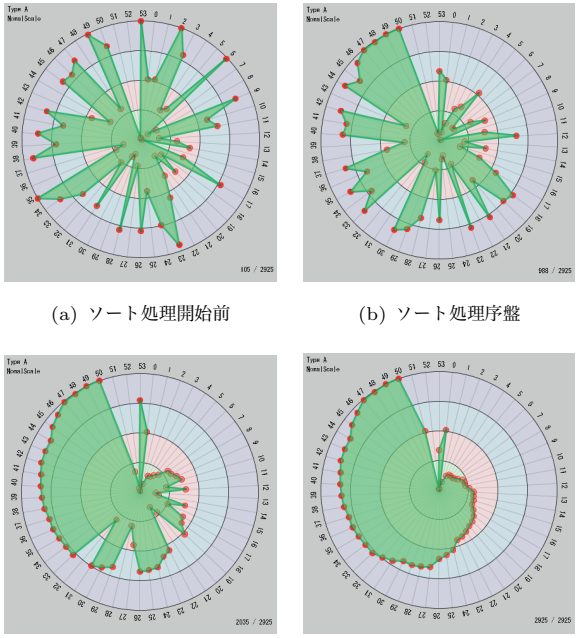
図 3 は, 50 個の値に対してソートを行うプログラムの実行状況を可視化したものである.

図 3(a) は, ソート処理開始前の状態であり, 棘が生えているような図形が描画されている. ソート処理が進むと図 3(b), 図 3(c) のように, 徐々に図形の棘がなくなり滑らかになっていき, ソート終了時は図 3(d) のように渦を巻くような図形となった.

5. 研究室で開発されたプログラムの解析

研究室で過去 10 年間に開発された 36 個のプログラム (表 1) に対して解析を行った.

†千葉工業大学 大学院 情報科学研究科 情報科学専攻
‡千葉工業大学 情報科学部 情報工学科



(a) ソート処理開始前　(b) ソート処理序盤

(c) ソート処理終盤　(d) ソート処理終了時

図 3: ソートプログラムの可視化例

5.1 対戦ゲームプログラム

可視化で見られた傾向を以下に示す.

- 実行の序盤に図形の外周が全て揃う.
- 図形の外周およそ 1/3 は形状変化がない.

可視化対象プログラムのソースコードと変数情報領域の情報から, 図形の変化がない部分は, ゲームの盤面情報を扱うための配列であることがわかった.

また, 代入処理の回数が最も多い変数は, for 文の制御変数であることがわかった.

5.2 プログラム実行状況の可視化プログラム

可視化で見られた傾向には以下の 2 種類があった.

パターン 1 実行の序盤に図形の外周が全て揃う.

パターン 2 実行に伴って徐々に図形が描かれていく.

パターン 1 のプログラムは, プログラムの実行状況の可視化を動画表現で行うものであった. パターン 2 のプログラムは, プログラムの実行状況の可視化を静止画や文字ベースで行うものであった.

また, 変数情報領域の情報から, 代入処理の回数が最も多い変数は, for 文の制御変数あるいは動画再生のための座標変数であることがわかった.

5.3 シミュレーションプログラム

可視化で見られた傾向を以下に示す.

- プログラム実行の序盤に図形の外周が全て揃う.
- 図形の外周およそ 1/2 は形状に変化がない.

形状に変化が見られない箇所は, 代入処理が 1 回のみの変数であった. この変数は, 可視化対象プログラムのソースコードと変数情報領域の情報から, 人の動きの設定や, 公園やテーマパークの情報を格納している変数であることがわかった.

また, 代入処理の回数が最も多い変数は, for 文の制御変数であることがわかった.

表 1: 解析対象プログラム

対象プログラム	代入回数	内容
対戦ゲーム (24 個)	数百万回 (一手辺り)	七並べ, 五目並べ, オセロ, 花札などの対戦プログラム
可視化プログラム (6 個)	数百〜数千回	プログラム実行時の変数や動作を可視するプログラム
シミュレーション (6 個)	数千〜数万回	公園やテーマパークでの人の行動シミュレーション

6. おわりに

図形表現を用いた可視化プログラムを作成し, 研究室で開発されたプログラムに適用した.

本可視化プログラムを用いて解析を行ったことで, 変数の値の代入や更新のタイミングが明らかになり, プログラムの振る舞いについて予想やイメージがしやすくなった. また, 図形が変形していく様子は印象に残りやすいく, プログラム実行状況の全体像を捉えやすくなる.

本可視化はプログラムの実行状況を理解する一助になることが期待される.

参考文献

[1] 神山 拓哉, 六沢 一昭, 変形する図形によるプログラムの実行状況の可視化, 第 15 回 情報科学技術フォーラム (FIT 2016), N-016, pp.319–320, 電子情報通信学会, 情報処理学会, 2016.

[2] 實川 祐児, 六沢 一昭, 変数の可視化と検査機能を持つプログラミングシステム, 第 6 回 情報科学技術フォーラム (FIT 2007), B-020, pp.125–126, 電子情報通信学会, 情報処理学会, 2007.

[3] Stephan Diel, Software Visualization, Springer, 2007.