Automatic Evolution of Java-written Game Heuristics

Michael Orlov, Carmel Bregman and Moshe Sipper

orlovm, carmelbr, sipper@cs.bgu.ac.il

Department of Computer Science
Ben-Gurion University, Israel

SSBSE 2011
September 12, Szeged, Hungary

GP: Programs or Representations?

"While it is common to describe GP as evolving **programs**, GP is not typically used to evolve programs in the familiar Turing-complete languages humans normally use for software development. It is instead more common to evolve programs (or expressions or formulae) in a **more constrained** and often **domain-specific language**."

A Field Guide to Genetic Programming
[Poli, Langdon, and McPhee, 2008]

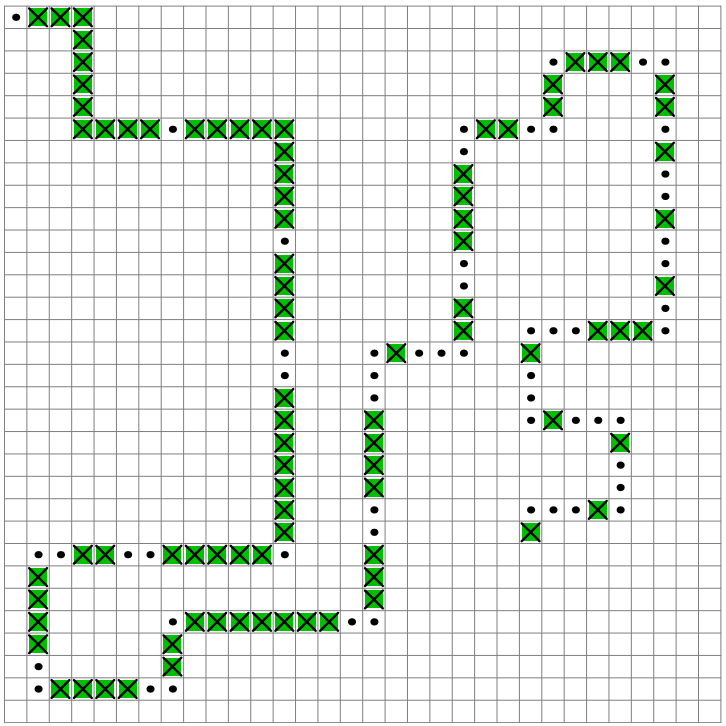
FINCH:
Fertile Darwinian Bytecode Harvester

- A system for evolving Java (bytecode).
- Employs a sophisticated compatible crossover operator.
- **Always** produces correct (compilable) bytecode.
- Which we can deploy directly or decompile back to Java for perusal.
- All you need: A seed and a wish (= fitness function).
- If the wish is good—even a **bad** seed will eventually blossom!

Garbage in... Goody out!

Evolved program solving Artificial Ant problem:

```
void step() {
  if (foodAhead()) {
    move(); right();
  }
  else {
    right(); right();
    if (foodAhead())
      left();
    else {
      right(); move();
      left();
    }
    left(); left();
  }
}
```



. . . and out

Evolved program solving Intertwined Spirals problem:

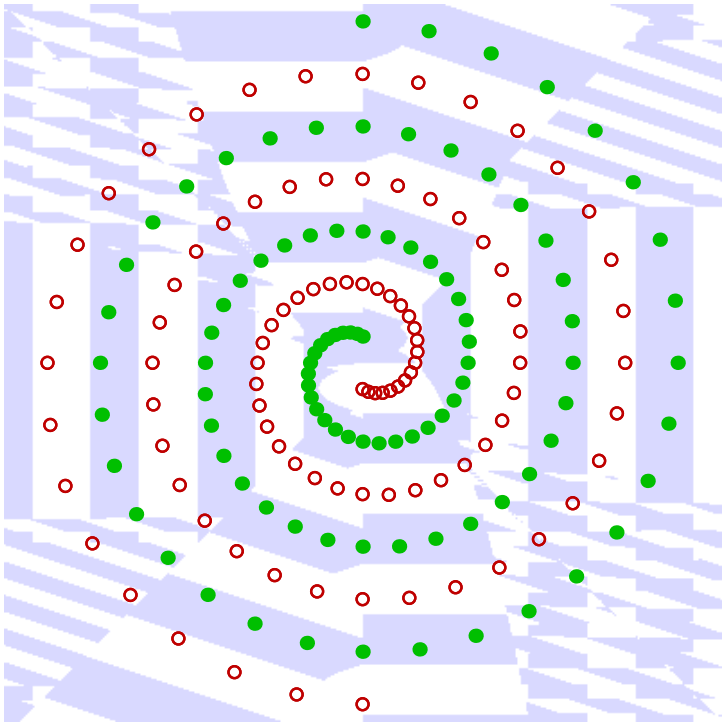
uses sign of $\sin\left(\frac{9}{4}\pi^2\sqrt{x^2+y^2} - \tan^{-1}\frac{y}{x}\right)$

```
boolean isFirst(double x, double y) {
  double a, b, c, e;
  a = Math.hypot(x, y);  e = y;
  c = Math.atan2(y, b = x) +
    -(b = Math.atan2(a, -a))
    * (c = a + a) * (b + (c = b));
  e = -b * Math.sin(c);
  if (e < -0.005612648701876277) {
    b = Math.atan2(a, -a);
    b = Math.atan2(a * c + b, x);  b = x;
    return false;
  }
  else
    return true;
}
```

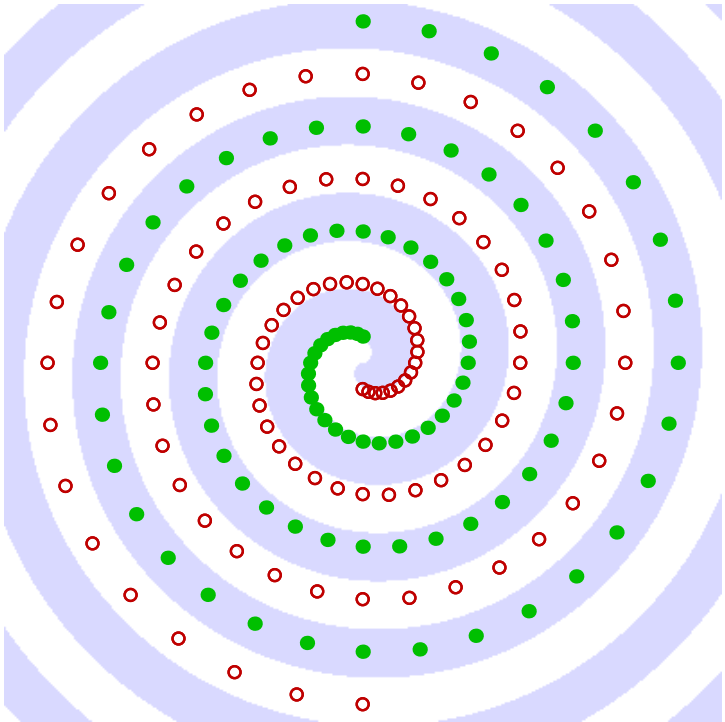
Tree GP vs. FINCH

And compare the phenotypes:

Koza's:



Ours:



The Insidious Errors

```
1  int negamaxAB(TicTacToeBoard board,
2        int alpha, int beta, boolean save) {
3    Position[] free = getFreeCells(board);
4    // utility is derived from the number of free cells left
5    if (board.getWinner() != null)
6      alpha = utility(board, free);
7    else if (free.length == 0)
8      alpha = 0   save = false ;
9    else for (Position move: free) {
10     TicTacToeBoard copy = board.clone();
11     copy.play(move.row(), move.col(),
12                     copy.getTurn());
13     int utility =  -   (removed) negamaxAB(copy,
14                     -beta, -alpha,  false   save );
15     if (utility > alpha) {
16       alpha = utility;
17       if (save)
18         // save the move into a class instance field
19         chosenMove = move;
20       if ( alpha >= beta   beta >= alpha )
21         break;
22     }
23   }
24   return alpha;
25 }
```

Why Evolve Software?

- Our work aims at one of the hardest problems known to (and created by) man: software design.

- Given the size and importance of the software industry, any step taken toward automating the programmer's task could impact society in ways far outreaching the boundaries of evolutionary computation.

Why Evolve Java Bytecode?

- We have taken another major step forward.
- Ours is the first approach that allows viable evolution of extant, real-world software in a mainstream programming language (Java is one of the 2 most popular programming languages).
- Moreover, FINCH is not limited to Java:
  Scala, Groovy, Jython, Kawa, JavaFX Script, Clojure.

Evolving Game Heuristics

- A row of cells: k black pieces, empty cell, k white pieces.
- Pieces move towards the opposite direction, striving to reverse the initial board situation.
- Pieces can move one step towards the opposite direction, or jump over one complementary-color piece.
- FINCH successfully evolved a `getMove` method, solving the problem consistently and effortlessly.
- Additionally, we had significant progress evolving heuristic evaluation functions for the game of *Connect Four*.

Evolving Game Heuristics

Evolved program solving the k-empty-k game:

```
Move getMove(Board board) {
    int i = board.findEmpty();
    Piece left1  = board.getPlace(i-1);
    Piece left2  = board.getPlace(i-2);
    Piece right1 = board.getPlace(i+1);
    Piece right2 = board.getPlace(i+2);
    if (left1  == Piece.BLACK && left2 == Piece.WHITE)
        return Move.RIGHT;
    if (right2 == Piece.BLACK)
        return Move.LEFT;
    if (left1  == right2)
        return Move.RIGHT;
    if (right1 == Piece.BLACK)
        return Move.LEFT;
    return Move.RIGHT;
}
```

Darwinian Software Development

- "Judiciously used, digital evolution can substantially augment the cognitive limits of human designers and can find novel (possibly counterintuitive) solutions to complex ... system design problems."

  (Recent study by US DoD on futuristic systems)

- FINCH represents a significant step on the (long) path toward full-fledged **Darwinian Software Development**.

References

M. Orlov and M. Sipper. **Genetic programming in the wild: Evolving unrestricted bytecode.** Proceedings of GECCO 2009.

M. Orlov and M. Sipper. **Flight of the FINCH through the Java wilderness.** IEEE Transactions on Evolutionary Computation, 15(2), pp. 166–182 (April 2011).