

# Genetic Algorithm for Evolving Cellular Automata: Guide to Code and to HW 7

This document describes the GA-CA and CV packages for evolving and viewing the behavior of cellular automata. A caveat: Most of this code was written **a long time ago**, and was not written for teaching purposes, so the coding style and documentation may leave something to be desired. However, the program still works, at least on Linux machines.

## Package 1: *GA-CA*

This package consists of the following files, written in C:

Header files:

- definitions.h
- globals.h
- params.h
- prototypes.h

Program files:

- main.c
- init.c
- ga.c
- files.c
- packed-ca.c
- params.c
- stats.c

Other files:

- params
- .run\_num

The only files you will need to modify for this assignment are “params” and “ga.c”.

The file “params” gives the parameters for the GA and their default values. To change the value of a parameter, you need to edit this file; you do not need to recompile the program since this file is read in at runtime. The parameters are:

NUM\_GA\_RUNS: The number of runs to do.

NUM\_GENERATIONS: The number of generations per run.

CA\_LATTICE\_SIZE: The number of cells in the one-dimensional CA.

CA\_POP\_SIZE: The GA population size.

NUM\_ICS: The number of initial configurations (ICs) on which to test each individual for computing its fitness.

CA\_ITERATIONS\_FACTOR: For a given IC, a CA is iterated for this value times LATTICE\_SIZE.

SNAPSHOT\_INTERVAL: How often output is written out (in generations).

SHORT\_PRINT: If TRUE, the .short file is printed.

LONG\_PRINT: If TRUE, the .long file is printed. Usually this is set to FALSE, since it produces a very large file.

LAST\_GENERATION\_PRINT: If TRUE, the .last\_generation file is printed out.

CA\_MUTATION\_PROB: Per-bit probability of mutation.

CA\_CROSSOVER\_PROB: Probability (per pair of parents) for single-point crossover to occur.

CA\_TOURNAMENT\_SIZE: Size for tournament used in selecting individuals to be parents.

The file “ga.c” contains the code for running the GA. The file “main.c” contains a function “GA()”, which is a wrapper that calls functions in ga.c.

## Writing the code for your assignment

For implementing uniform crossover, you will need to modify the crossover part of the function “perform\_crossover\_and\_mutation” in ga.c. The function “knuth\_random()” returns a random double between 0 and 1.

All other experiments (except the optional coevolution experiment) will simply involve changing parameter values in the params file.

# Compiling and Running the Code

To compile the program: type “make”.

To run the program: type “ga-ca”.

Each run of the GA is assigned a unique identifying run number, `<run_num>`, that is read from the file `.run_num`. The number in that file is incremented by 10 each time the GA runs.

Running the GA with its default parameters will produce two output files:

`<run_num>.header`: lists the random number seed and values of the various parameters used in this run.

`<run_num>.short`: A trace of the GA run, in “short” format. The columns are (from left): Generation number; average fitness in the population; standard deviation of average fitness in the population; and best fitness found in the population at that generation.

If you would like to explore the behavior of the CAs you evolve, you can set the parameter `LAST_GENERATION_PRINT` to `TRUE` and run the GA. This will produce an additional output file:

`<run_num>.last_generation`: A list of every chromosome in the final population. The columns are (from left): Generation number; ID number of chromosome; chromosome bit string; chromosome fitness.

You can then use the *CA-Viewer* package to view CAs from this last generation.

## Package 2: *CA-Viewer*

The *CA-Viewer* is a package that uses the Linux “display” command as well as other Linux commands, so it will run only on Linux systems (as far as I know).

The *CA-Viewer* code is in the subdirectory “ca-viewer” in the source code directory.

To compile the program: go to this subdirectory and type “make”.

To run the program: type “cv”. This will prompt you for “Chromosome (as bit string):”. Cut and paste the chromosome you wish to view. A space-time diagram of its behavior, starting with a random initial configuration, should appear on the screen.