Appendix D

GONZO Example Applications

This Appendix contains the Lisp code used in Gonzo to produce the examples described in Chapter 7, Section 7.3. The code used in GeCo to produce online visualizations is given in section D.1. The code used in Gonzo to produce the offline visualizations of the maximum integer problem, the De Jong F1 test problem, and the royal road function, are given in Section D.2.

D.1 Online Visualization

The following annotated version of the GeCo EVOLVE method is used to produce online visualizations in Gonzo. The annotations made to the original GeCo EVOLVE method are shown here in bold.

```
(defmethod EVOLVE ((self ecosystem))

  (unless viz::*visualization-dialog* (viz:create-visualizations self))

  (evaluate self (plan self))

  (mapcar #'(lambda (view)

    (setf (current-generation-range view)

      (mapcar #'incf (current-generation-range view)))

    (setf (total-generation-range view)

      (list (first (total-generation-range view))

        (incf (second (total-generation-range view)))))))
```
D.2 Gonzo Example Problem Visualizations

This section presents the code used in Gonzo to produce the three example problem visualizations presented in Section 7.4.

D.2.1 The Maximum Integer Problem

The following Lisp code was used to produce the Gonzo visualization shown in Figure 7.9 of the maximum integer problem, see page 199.

```lisp
(defun maxint ()
  (test-plan \'run-1 \'maxint-plan) ;; GECO GA dataset run-1
  (create-visualizations run-1) ;; Gonzo create visualizations function
```

```lisp
(defun maxint ()
  (test-plan \'run-1 \'maxint-plan) ;; GECO GA dataset run-1
  (create-visualizations run-1) ;; Gonzo create visualizations function
```

This section presents the code used in Gonzo to produce the three example problem visualizations presented in Section 7.4.
(defmethod create-visualizations ((run ecosystem))

(setf *visualization-dialog* nil ;; list-of-dialog-items
     'my-navigator ;; device
     cg:*screen* ;; stream
     :name 'visualizer ;; name
     :pop-up-p nil ;; not a pop-up dialog
     :background-color cg:white ;; background colour
     :window-exterior (cg:make-box 30 50 1030 850) ;; window exterior box
     :title "Test") ;; title string for window

(create-fitness-versus-time-graph
  'fitness-graph-0 ;; name
  run ;; dataset
  *visualization-dialog* ;; parent-dialog
  (cg:make-box 400 600 1000 800)) ;; exterior-box

(create-fine-grained-chromosome-view
  'text-view-0 ;; name
  *visualization-dialog* ;; parent-dialog
  (cg:make-box 0 350 400 800)) ;; exterior-box

(create-search-space-visualization
  'scatterplot-view-0 ;; name
  run ;; dataset
  'GSM-D-circle ;; chromosome-mapping-technique
  *visualization-dialog* ;; parent-dialog
  (cg:make-box 400 0 1000 600)) ;; exterior-box
GONZO EXAMPLE APPLICATIONS

D.2.2 The De Jong F1 Test Problem

The following code was used to produce the example visualizations of a GA solving De Jong’s F1 test problem, as shown in Figure 7.12, see page 202. This code is virtually identical to that used to produce the visualizations of the maximum integer problem given in the previous subsection, the only differences being a change in the GA’s genetic plan, the image mapping used in the search.
space visualization, and the window dimensions of the schema highlight selector and fine grained chromosome view.

(defun dejong ()
  (test-plan 'run-1 'dejong-plan) ;; GEXO GA dataset run-1
  (create-visualizations run-1) ;; GONZO create visualizations function
)

(defun create-visualizations ((run ecosystem))
  (setf *visualization-dialog* nil) ;; visualization container dialog

  (open-dialog
    nil ;; list-of-dialog-items
    'my-navigator ;; device
    cg:*screen* ;; stream
    :name 'visualizer ;; name
    :pop-up-p nil ;; not a pop-up dialog
    :background-color cg:white ;; background colour
    :window-exterior (cg:make-box 30 50 1030 850) ;; window exterior box
    :title "Test") ;; title string for window

  (create-fitness-versus-time-graph
    'fitness-graph-0 ;; name
    run ;; dataset
    *visualization-dialog* ;; parent-dialog
    (cg:make-box 400 600 1000 800)) ;; exterior-box

  (create-fine-grained-chromosome-view
    'text-view-0 ;; name
    *visualization-dialog* ;; parent-dialog
(create-search-space-visualization
 'scatterplot-view-0 ;; name
 run ;; dataset
 'GSM-D-colour-rectangle ;; chromosome-mapping-technique
 *visualization-dialog* ;; parent-dialog
 (cg:make-box 0 400 1000 600) ;; exterior-box
 'D-GSM ;; coordinate-mapping-technique
 (list text-view-0)) ;; list-of-views

(create-movie-player
 'control-panel ;; name
 '(i < < i > 1 >> i) ;; list-of-labels
 '(start rewind back play-pause forward fforward end) ;; list-of-functions
 (list scatterplot-view-0) ;; list-of-views
 *visualization-dialog* ;; parent-dialog
 (cg:make-box 0 400 85)) ;; exterior-box

(create-generation-fitness-selector
 'view-range-window ;; name
 (list scatterplot-view-0) ;; list-of-views
 *visualization-dialog* ;; parent-dialog
 (cg:make-box 0 85 400 250)) ;; exterior-box

(create-schema-highlight-selector
 'schema-editor-window ;; name
 (list scatterplot-view-0) ;; list-of-views
 *visualization-dialog* ;; parent-dialog
 (cg:make-box 0 250 400 430)) ;; exterior-box
D.2.3 The Royal Road Problem

The royal road problem was the last example presented in Section 7.4. In order to produce this visualization, a new method was created to generate matrices of search space visualizations. This create-search-space-visualization-matrix method is presented here along with the code used to produce the example visualization of a GA solving the royal road problem.

(defun create-search-space-visualization-matrix
  (name-list (dataset ecosystem) chromosome-mapping-technique parent-dialog
  list-of-exterior-boxes coordinate-mapping-technique list-of-list-of-views
  list-of-projection-locus-orderings)

(mapcar #'(lambda (name exterior-box list-of-views loci-list)
  (create-search-space-visualization
    name
    dataset
    chromosome-mapping-technique
    parent-dialog
    exterior-box
    coordinate-mapping-technique
    list-of-views
    loci-list))
  name-list window-boxes list-of-list-of-views list-of-projection-locus-orderings)
)

This create-search-space-visualization-matrix method was applied as follows to produce the visualization shown in Figure 7.14, see page 205.
(create-search-space-visualization-matrix

'(scatterplot-view-0 scatterplot-view-1 scatterplot-view-2
   scatterplot-view-3 scatterplot-view-4 scatterplot-view-5
   scatterplot-view-6 scatterplot-view-7) ;; list-of-names

run-1 ;; dataset

'GSM-D-circle ;; chromosome-mapping-technique
*visualization-dialog* ;; parent-dialog

'((cg:make-box 400 204 619 402) (cg:make-box 619 204 839 402)
   (cg:make-box 839 204 1058 402) (cg:make-box 1058 204 1278 402)
   (cg:make-box 400 502 619 704) (cg:make-box 619 502 839 704)
   (cg:make-box 839 502 1058 704) (cg:make-box 1058 502 1278 704)) ;; list-of-exterior-boxes

'D-GSM ;; coordinate-mapping-technique

(list text-view-0) ;; list-of-views

'((0 1 2 3 4 5 6 7) (8 9 10 11 12 13 14 15) (16 17 18 19 20 21 22 23)
   (24 25 26 27 28 29 30 31) (32 33 34 35 36 37 38 39) (40 41 42 43 44 45 46 47)
   (48 49 50 51 52 53 54 55) (56 57 58 59 60 61 62 63)) ;; list-of-projection-locus-orderings