Visual Analysis of Evolutionary Algorithms

Annie S. Wu[1], **Kenneth A. De Jong**[2], **Donald S. Burke**[3], **John J. Grefenstette**[4], and **Connie Loggia Ramsey**[5]

[1]Naval Research Laboratory, Code 5514, Washington, DC 20375, aswu@aic.nrl.navy.mil
[2]Computer Science Department, George Mason University, Fairfax, VA, kdejong@cs.gmu.edu
[3]Center for Immunization Research, Johns Hopkins University, Baltimore, MD, dburke@jhsph.edu
[4]Institute for Biosciences, Bioinformatics and Biotechnology, George Mason University, Manassas, VA, gref@ib3.gmu.edu
[5]Naval Research Laboratory, Code 5514, Washington, DC 20375, ramsey@aic.nrl.navy.mil

Abstract- The non-linear complexity of evolutionary algorithms (EAs) make them a challenge to understand. The difficulty in performing detailed analyses of an EA is in sorting through the large amount of of data that can be generated in a single run. This paper describes a visualization tool that facilitates navigation through the details of an EA run. The visualization tool organizes and displays EA data at various levels of detail and allows for easy transitions between related pieces of data.

1 Introduction

The large numbers of complex, non-linear interactions that compose evolutionary algorithms (EAs) make them difficult to analyze and a challenge to understand. The most common methods of evaluating EAs, which include evolutionary programming, evolutionary strategies, and genetic algorithms (GAs), focus on overall performance and gross population statistics. Such methods include comparing the number of function evaluations required to find an acceptable solution, the quality of the solution found, or the rate of increase of the population fitness. The effectiveness of specific mechanisms such as crossover may be estimated by comparing the overall performance of an EA with and without that mechanism turned on. While such measurements are important and provide useful top-down information about EA runs, focusing only on this information could cause us to miss other important details. For example, several studies have investigated the effects of non-coding regions on GA performance. These regions were expected to improve performance by providing a buffer against the disruptive effects of crossover; however, comparisons revealed little difference in overall GA performance (Forrest & Mitchell, 1992; Wu & Lindsay, 1995). A later study which examined the details of reproduction events found that non-coding regions did indeed reduce the disruptive effects of crossover in building blocks (Wu, Lindsay & Riolo, 1997). In fact, non-coding regions reduced crossover's total activity within building block regions, including crossover's ability to construct new building blocks. This decrease in construction appeared to cancel out any advantage gained from the expected decrease in disruption, resulting in little noticeable improvement in overall performance. Such complex, non-linear interactions exist in all types of EAs and suggest that, to fully understand how EAs work, we must examine not only the end result of EA runs but also the means to the end.

The difficulty in performing such analyses lies in sorting through the large amount of data that can be generated in a single EA run. While all of the data from an EA run can easily be saved into files, accessing and interpreting such a large amount of information is no trivial task. The development of tools for sorting, organizing and displaying such databases could greatly facilitate the access and analysis of such data. Desirable capabilities of such tools include the ability to do the following:

- to examine individuals and their encodings in detail
- to trace the source and survival of building blocks or partial solutions
- to trace family trees
- to examine the effects of genetic operators
- to examine populations for convergence, speciation, etc.
- to trace gross population statistics and trends
- to move freely in time and through populations.

One approach to this problem is off-line visualization. We have developed such a system, called VIS, to facilitate analysis of data from the Virtual Virus (VIV) project (Burke, De Jong, Grefenstette, Ramsey & Wu, 1999). VIS takes advantage of the effectiveness of graphical representations and the flexibility of a "clickable" links to provide a navigation tool for accessing and displaying data. In the rest of this paper, we will summarize previous approaches to visualizing EAs and describe the VIS system and how it addresses some of the desired capabilities listed above. Though the examples shown in this paper are primarily from the VIV project which is based on a GA, the VIS program can easily be extended to support other EA data.

2 Background

Graphical visualization techniques are some of the simplest and at the same time most powerful methods for analyzing and communicating information (Tufte, 1983). Well designed graphical elements can convey large amounts of information in very concise and compact formats. In addition, the human vision system is extremely sensitive to graphical patterns, making graphical representations an extremely useful analysis tool.

Visualization techniques have been used to study EAs both on-line and off-line. On-line systems allow users to closely follow and evaluate the progress of an EA and, in some cases, may allow users to interactively influence or guide the direction of evolution of the system (Collins, 1998; Jones, 1993). Off-line systems use visualization techniques to display information about an EA run after it is complete and may allow the display of data from multiple times of a run and movement both forwards and backwards in the evolutionary process (Shine & Eick, 1997). The most basic displays that have been used include population data matrices which simply present an entire population in raw text format and two-dimensional plots of individual aspects of the population (such as best or average fitness) with respect to time. More complex techniques have focused on how to display more substantial information about entire populations. Examples include methods for displaying the distribution of the population in the solution space (Collins, 1997; Collins, 1998; Nassersharif, Ence & Au, 1994; Shine & Eick, 1997), allele frequencies (Collins, 1997; Wu & Lindsay, 1996), the formation and variation of species (Spears, 1994), and the ancestry of individuals (Spears, 1994) from generation to generation. The multi-dimensional nature of EA systems makes this a balancing act between the clarity of the display and the amount of information that can be included.

3 Overview of system

VIS is an off-line visualization program developed to facilitate the examination and analysis of GA runs. This system was designed with two main goals in mind.

- Provide users with a tool with which they can examine the details of a GA run. The tool should provide easy access to desired information and easy transitions between related pieces of information.
- Develop novel methods and representations for displaying multi-dimensional data in a coherent and informative manner.

VIS is organized as a collection of windows that display data at varying levels of resolution. Within these windows, VIS combines graphical and textual displays to allow users to view "snapshots" in time from a GA run, to examine specific individuals and populations from a run, and to navigate forwards and backwards through a run. "Clickable" elements in the displays link related pieces of information and allow users to easily move through time and among the populations of a GA run.

Whereas most of the systems described in section 2 focus on ways to display the contents and evolution of entire populations and an EA's progress in the solution space, the goal of the VIS tool is to make the data from individual GA runs available and easily accessible for observation and analysis. As a result, VIS focuses less on developing abstract representations which may average, interpolate, or otherwise lose details, and more on displaying complete information and providing navigation capabilities for moving from one part of a run to another.

Because VIS is an off-line system, it is used only after a GA run is complete. Any GA run that is to be analyzed must generate a set of data files containing all of the information necessary for VIS to completely reconstruct the run. Details about files and formats are given in (Grefenstette, Burke, De Jong, Ramsey & Wu, 1997).

3.1 Representation of individuals

The most important elements of a GA are the individuals from the GA populations. These individuals represent potential solutions to the problem to be solved and are typically encoded as strings of characters or values. The ability to examine the formation and structure of individuals in a GA is one of the main purposes for developing the VIS tool. As a result, how we represent or display individuals is very important. VIS is able to display the individuals of a GA run in both textual and graphical representations. Graphical representations can enhance analysis of textual information for several reasons. Color blocks or strips are both easier to distinguish and require less space than individual characters or letters (allowing display of longer individuals). In addition, similarities and differences in color strip patterns are very easy for human vision system to detect, facilitating the comparison of multiple individuals.

VIS allows users to select from several different methods of representing individuals. While current representations focus on discrete alphabets – binary and multi-character – VIS can easily be extended to support floating-point representations and other problem-specific representations as needed. Table 1 shows examples of the currently available representations. The **Genotype** representation displays each individual as a string of characters. For binary alphabets, the characters will be either zero (`0`) or one (`1`). Alternative representations that use more than two values may be composed of other characters. This representation can be used for all alphabets. The **Zebra** representation works with binary alphabets and displays individuals as a series of black and white stripes. A black stripe represents a `0`; a white stripe represents a `1`. The **Neopolitan** representation also works with binary alphabets. This representation assigns one color to each pair of characters. There are four possible pairs of characters; an example color coding scheme would be: `00` = black, `11` = white, `01` = magenta, `10` = orange. This representation is especially sensitive to shifts, insertions, and deletions of one character. The **Color coded** representation works with multi-character alphabets and assigns a unique color to each letter of the alphabet. Individuals are represented as a series of multicolored

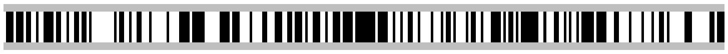
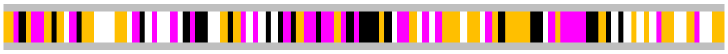
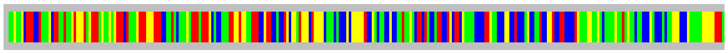
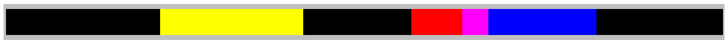
Name	Representation	Alphabet
Genotype	33123010011223023302303213030230001201	All
Zebra		Binary
Neopolitan		Binary
Four Color		Multi-character
Gene locations		Various

Table 1: Available VIS representations of individuals.

stripes. For example, the following color coding scheme was used for the VIV alphabet: `A` = blue, `C` = red, `G` = yellow, `T` = green. The **Gene location** works with problems in which groups of characters together compose building blocks or partial solutions. Each building block is displayed in a unique color on an individual. This representation is especially useful for tracing the construction, propagation, and disruption of building blocks.

3.2 Examining individuals

One of the most basic capabilities needed for studying GAs is easy and direct access to any individual of a run. Because individuals are the most basic elements of a GA, the ability to examine them in detail, in effect, gives us the ability to reconstruct events from any portion or all of a run. Because solutions are encoded as individuals, examining the fitness and composition of an individual essentially allows us to evaluate a GA's progress at a particular moment of a run. Parent and offspring comparisons can reveal the dynamics about reproduction events and how effectively information is constructed, propagated, and disrupted by various genetic operators. Easy access to the "family members" of an individual gives us the ability to trace the discovery and inheritance of information from generation to generation.

VIS provides some of these capabilities with its **Individual** window. An **Individual** window displays all relevant data associated with a given individual. Two formats are available: the **Data** format displays the vital statistics for an individual and the **Family** format displays a graphical representation of a complete family (an individual, its parents, and its offspring). Figure 1 shows an example of the **Data** format. The following information is included in this display.

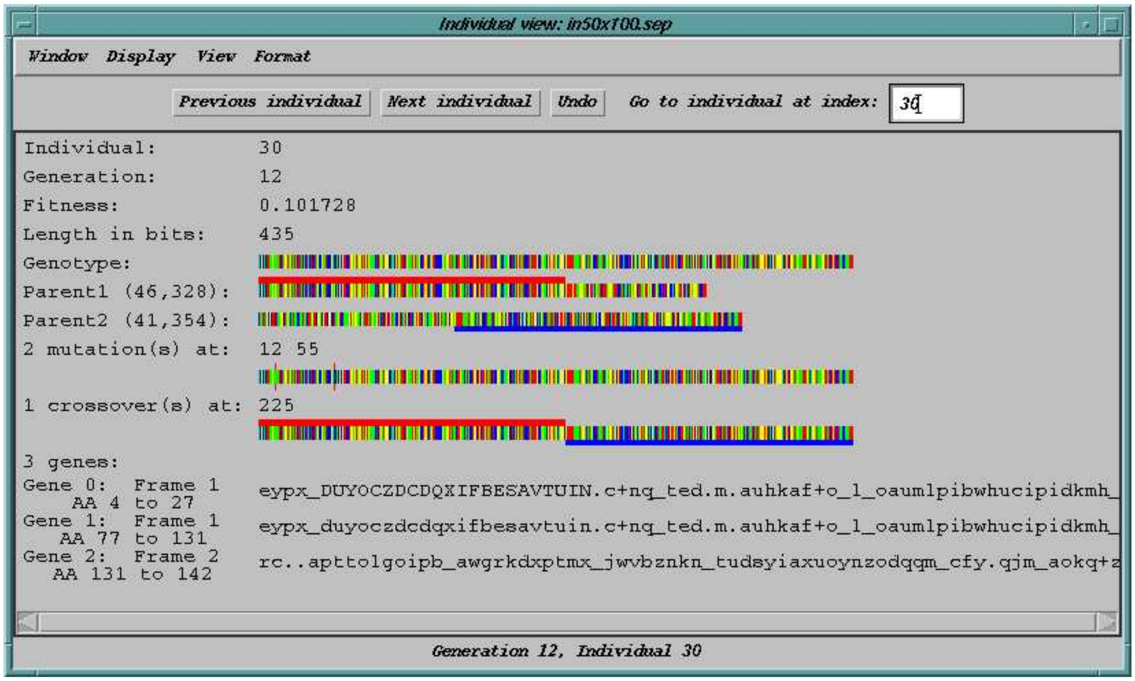
- The index of the individual. Each individual is arbitrarily assigned an index number to distinguish it from other individuals in the same generation.
- The generation to which the individual belongs.
- The fitness of the individual.
- The length of the individual in bits.
- The genotype of the individual. Available representations are shown in Table 1. The example in Figure 1 uses the **Color coded** representation.
- The genotype, index, and length of the individual's parents. Users may click on a parent genotype to open a new **Individual** window for that parent.
- The mutations, if any, involved in creating this individual. Mutation locations listed and are marked in color.
- The crossover points, if any, involved in creating this individual. If crossover occurred, the portion that each parent contributed to the individual is indicated in color. If crossover did not occur, the individual was cloned from `Parent1`.
- Any problem specific information such as genes or reading frames.

Figure 2 shows an example of the **Family** format. Clicking on a parent or offspring representation in either the **Data** or **Family** display opens a new **Individual** window for the selected individual.

3.3 Examining populations

In addition to examining individuals in isolation, it is also important to understand how individuals relate to other individuals in the population. Well designed graphical displays of a population can facilitate the detection of patterns or trends in the population that may suggest convergence or speciation. Overall characteristics such as diversity, convergence, and level of speciation can be important indicators of a GA's progress.

A **Population** window, shown in Figure 3, displays the individuals of a population in a scrollable window. allowing users to browse entire population. Three types of formats are available in **Population** windows: **Individual**, **Statistics**, and **Histogram**. The **Individual** format, shown in Figure 3, displays the individuals of a population and their index and fitness values. The **Statistics** format, shown in Figure 4, displays statistics for each individual in a population. For both of the above windows, users may click on a specific individual or its index to open a new **Individual** window for that individual. The **Histogram** format, shown in Figure 5, displays a histogram of the fitnesses of the individuals in the population. This display is particularly useful for examining the diversity of a population.



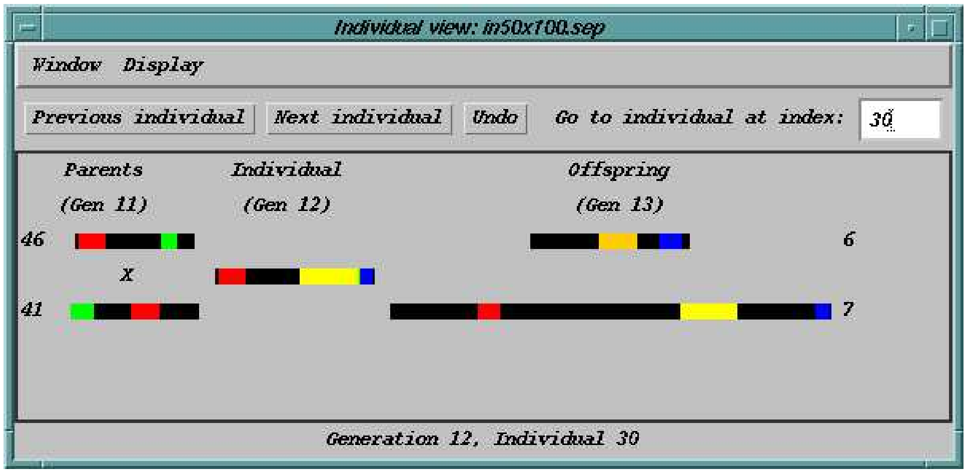
```
Individual:          30
Generation:          12
Fitness:             0.101728
Length in bits:      435
Genotype:
Parent1 (46,328):
Parent2 (41,354):
2 mutation(s) at:    12 55

1 crossover(s) at:   225

3 genes:
Gene 0:   Frame 1    eypx_DUYOCZDCDQXIFBESAVTUIN.c+nq_ted.m.auhkaf+o_l_oaumlpibwhucipidkmh_
   AA 4 to 27
Gene 1:   Frame 1    eypx_duyoczdcdqxifbesavtuin.c+nq_ted.m.auhkaf+o_l_oaumlpibwhucipidkmh_
  AA 77 to 131
Gene 2:   Frame 2    rc..apttolgoipb_awgrkdxptmx_jwvbznkn_tudsyiaxuoynzodqqm_cfy.qjm_aokq+z
  AA 131 to 142
```

Generation 12, Individual 30

Figure 1: An **Individual** window showing the **Data** format.



Generation 12, Individual 30

Figure 2: An **Individual** window showing the **Family** format.

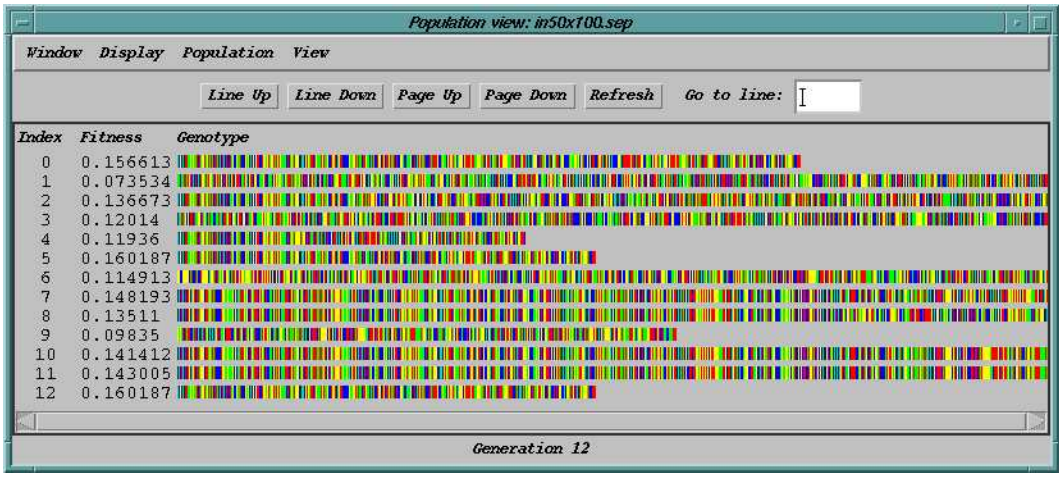
3.4 Examining runs

In addition to providing access to very specific details from a run, VIS also displays data relating to an entire run. Tracking gross population and run statistics provide a general idea of GA performance. Such data may contain useful information on trends through time as well as indications of areas (moments) that merit further investigation.

A **Run** window displays data over the entire run. The **Best** and **Median** formats display the best or median individuals, respectively, from each generation of a run. Figure 6 shows an example of a **Best** format. Users may click on an individual to open a new **Individual** window for that individual or click on a generation number to open a new **Population** window for that generation. The **Consensus** format, shown in Figure 7, displays statistics and a “consensus” individual for each generation of a run. The consensus individual shows the most common ordering of genes in a population.

4 Summary and future work

The VIS tool is an off-line visualization program developed to facilitate the examination and analysis of GAs. Instead of focusing solely on methods for displaying a GA’s progress in a solution space, VIS concentrates on ways to make all of the details of a run available and easily accessible. This tool allows users to examine snapshots of a GA run and investigate



Index	Fitness	Genotype
0	0.156613	
1	0.073534	
2	0.136673	
3	0.12014	
4	0.11936	
5	0.160187	
6	0.114913	
7	0.148193	
8	0.13511	
9	0.09835	
10	0.141412	
11	0.143005	
12	0.160187	

Figure 3: A **Population** window showing the **Individual** format.

questions such as how were the pieces of a solution assembled, when and why did a population converge, and what are the immediate effects of variation of parameters such as population size or selection method. We have found VIS to be an extremely useful tool for examining details of a GA run beyond the average and best fitness for each population. VIS allows us to focus in on specific details of interest, keeping related data easily accessible and all other data available. It has been especially useful in situations where we would otherwise need to print out unmanageable amounts of data just to find or examine a few specific examples.

The VIS tool played an integral role in our analyses of experiments from the VIV project. Using VIS, we were able to find specific examples to verify that, given the opportunity, a GA will retain backup copies of useful information and use this backup information if primary information is disrupted. In addition, we were able to examine the convergence of populations and the effects of genetic operators in detail. Full descriptions of these studies can be found in (Burke, De Jong, Grefenstette, Ramsey & Wu, 1999; Ramsey, De Jong, Grefenstette, Wu & Burke, 1998). We have also found the VIS tool to be extremely useful for developmental and verification purposes. In the development of new GA programs and applications, using a visualization tool to verify new representations and methods can be significantly easier and less time consuming than the alternative of printing out and verifying information on paper or on screen. In essence, VIS can be thought of as a debugger, but at the algorithmic level rather than the code level.

Future work on this project includes the continued development of effective displays of individual and population data and interactive sorting capabilities. We would like to extend graphical representations of individuals to support additional problem representations, including floating point values and possibly two-dimensional structural representations. In addition, we also plan to add automated data collection and statistical analysis options to collect, calculate, and plot data such as the number of offspring generated or the discovery and loss of partial solutions over entire runs or populations.

Acknowledgment

This research was conducted at the Naval Research Laboratory with support from the National Research Council and the Office of Naval Research.

References

Burke, D. S., De Jong, K. A., Grefenstette, J. J., Ramsey, C. L., & Wu, A. S. (1999). Putting more genetics into genetic algorithms. *Evolutionary Computation*, *6*(4), 387–410. (Winter 1998 issue).

Collins, T. D. (1997). Using software visualization technology to help evolutionary algorithm users to validate their solutions. In *Proceedings of the 7th International Conference on Genetic Algorithms*, (pp. 307–314).

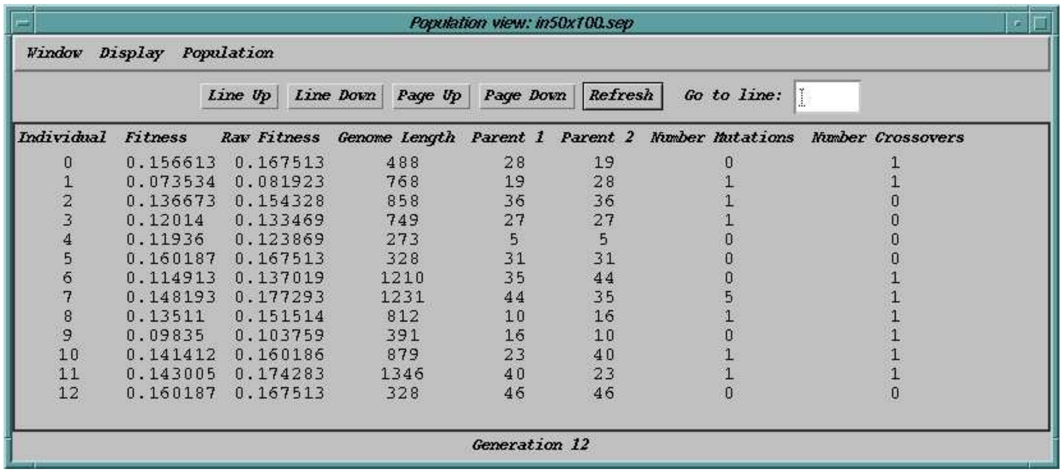
Collins, T. D. (1998). Understanding evolutionary computer: A hands on approach. In *WCCI-98*.

Forrest, S. & Mitchell, M. (1992). Relative building-block fitness and the building-block hypothesis. In *Foundations of Genetic Algorithms 2*, (pp. 109–126).

Grefenstette, J. J., Burke, D. S., De Jong, K. A., Ramsey, C. L., & Wu, A. S. (1997). An evolutionary computation model of emerging virus diseases. Technical Report AIC-97-030, Navy Center for Applied Research in Artificial Intelligence.

Jones, T. (1993). An introduction to SFI Echo. Santa Fe Institute working paper #93-12-074.

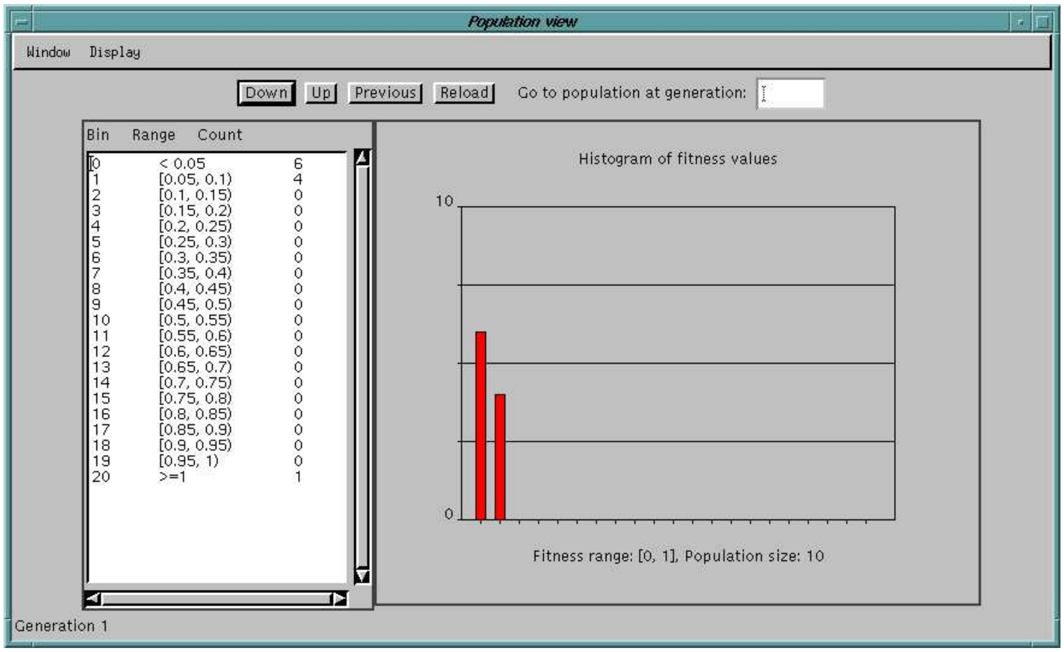
Nassersharif, B., Ence, D., & Au, M. (1994). Visualization of evolution of genetic algorithms. In *Proceedings of the World Congress on Neural Networks*, volume 1, (pp. 560–565).



Individual	Fitness	Raw Fitness	Genome Length	Parent 1	Parent 2	Number Mutations	Number Crossovers
0	0.156613	0.167513	488	28	19	0	1
1	0.073534	0.081923	768	19	28	1	1
2	0.136673	0.154328	858	36	36	1	0
3	0.12014	0.133469	749	27	27	1	0
4	0.11936	0.123869	273	5	5	0	0
5	0.160187	0.167513	328	31	31	0	0
6	0.114913	0.137019	1210	35	44	0	1
7	0.148193	0.177293	1231	44	35	5	1
8	0.13511	0.151514	812	10	16	1	1
9	0.09835	0.103759	391	16	10	0	1
10	0.141412	0.160186	879	23	40	1	1
11	0.143005	0.174283	1346	40	23	1	1
12	0.160187	0.167513	328	46	46	0	0

Generation 12

Figure 4: A **Population** window showing the **Statistics** format.


Population view
Window
Display
Down
Up
Previous
Reload
Go to population at generation:
Histogram of fitness values
10
0
Fitness range: [0, 1], Population size: 10

Bin	Range	Count
0	< 0.05	6
1	[0.05, 0.1)	4
2	[0.1, 0.15)	0
3	[0.15, 0.2)	0
4	[0.2, 0.25)	0
5	[0.25, 0.3)	0
6	[0.3, 0.35)	0
7	[0.35, 0.4)	0
8	[0.4, 0.45)	0
9	[0.45, 0.5)	0
10	[0.5, 0.55)	0
11	[0.55, 0.6)	0
12	[0.6, 0.65)	0
13	[0.65, 0.7)	0
14	[0.7, 0.75)	0
15	[0.75, 0.8)	0
16	[0.8, 0.85)	0
17	[0.85, 0.9)	0
18	[0.9, 0.95)	0
19	[0.95, 1)	0
20	>=1	1

Generation 1

Figure 5: A **Population** window showing a **Histogram** format.

Ramsey, C. L., De Jong, K. A., Grefenstette, J. J., Wu, A. S., & Burke, D. S. (1998). Genome length as an evolutionary self-adaptation. In *Parallel Problem Solving from Nature 5*, (pp. 345–353).

Shine, W. B. & Eick, C. F. (1997). Visualizing the evolution of genetic algorithm search processes. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, (pp. 367–372).

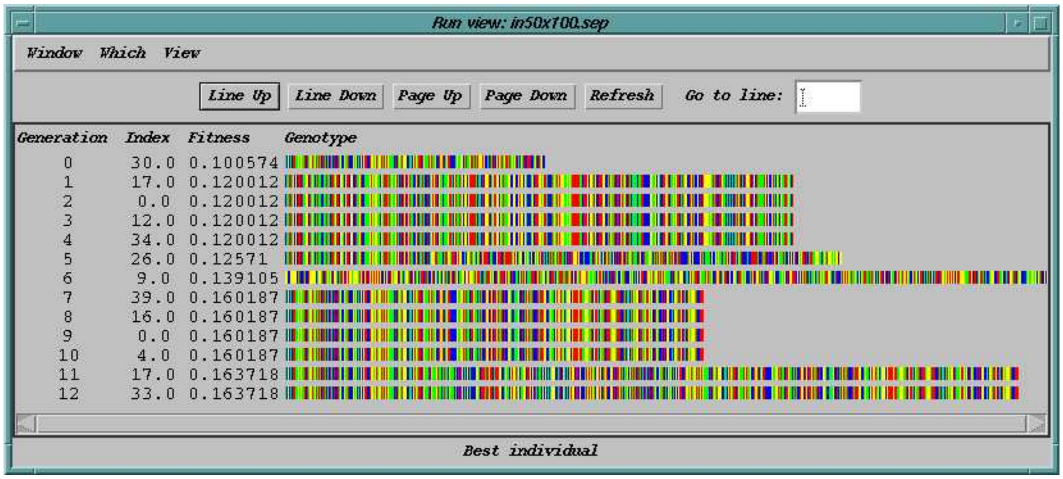
Spears, W. M. (1994). Visualizing genetic algorithms. Technical Report AIC-94-055, Navy Center for Applied Research in Artificial Intelligence.

Tufte, E. R. (1983). *The Visual Display of Quantitative Information*. Graphics Press.

Wu, A. S. & Lindsay, R. K. (1995). Empirical studies of the genetic algorithm with non-coding segments. *Evolutionary Computation*, *3*(2), 121–147.

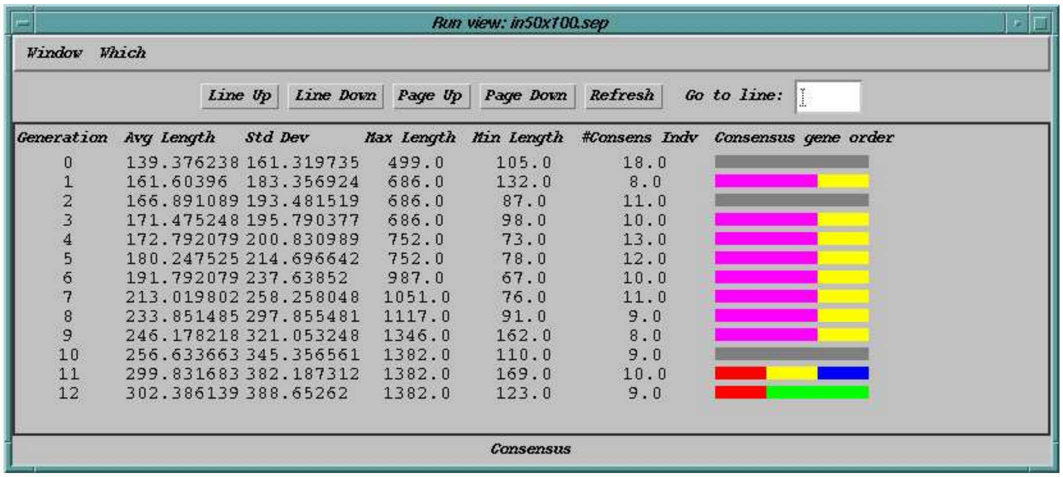
Wu, A. S. & Lindsay, R. K. (1996). A comparison of the fixed and floating building block representation in the genetic algorithm. *Evolutionary Computation*, *4*(2), 169–193.

Wu, A. S., Lindsay, R. K., & Riolo, R. L. (1997). Empirical observations on the roles of crossover and mutation. In Back, T. (Ed.), *Proceedings of the 7th International Conference on Genetic Algorithms*, (pp. 362–269).



Generation	Index	Fitness	Genotype
0	30.0	0.100574	
1	17.0	0.120012	
2	0.0	0.120012	
3	12.0	0.120012	
4	34.0	0.120012	
5	26.0	0.12571	
6	9.0	0.139105	
7	39.0	0.160187	
8	16.0	0.160187	
9	0.0	0.160187	
10	4.0	0.160187	
11	17.0	0.163718	
12	33.0	0.163718	

Figure 6: A **Run** window showing the **Best** format.



Generation	Avg Length	Std Dev	Max Length	Min Length	#Consens Indv	Consensus gene order
0	139.376238	161.319735	499.0	105.0	18.0	▨
1	161.60396	183.356924	686.0	132.0	8.0	🟪🟨
2	166.891089	193.481519	686.0	87.0	11.0	▨
3	171.475248	195.790377	686.0	98.0	10.0	🟪🟨
4	172.792079	200.830989	752.0	73.0	13.0	🟪🟨
5	180.247525	214.696642	752.0	78.0	12.0	🟪🟨
6	191.792079	237.63852	987.0	67.0	10.0	🟪🟨
7	213.019802	258.258048	1051.0	76.0	11.0	🟪🟨
8	233.851485	297.855481	1117.0	91.0	9.0	🟪🟨
9	246.178218	321.053248	1346.0	162.0	8.0	🟪🟨
10	256.633663	345.356561	1382.0	110.0	9.0	▨
11	299.831683	382.187312	1382.0	169.0	10.0	🟥🟨🟦
12	302.386139	388.65262	1382.0	123.0	9.0	🟥🟩

Figure 7: A **Run** window showing the **Consensus** format.