

INTERACTIVE EVOLUTION FOR SYSTEMATIC EXPLORATION OF A PARAMETER SPACE

DARYL H. HEPTING

Department of Computer Science
University of Regina
Regina, SK, S4S 0A2, CANADA
dhh@cs.uregina.ca

ABSTRACT

Interactive evaluation of a fitness function for a genetic algorithm through direct manipulation is known as interactive evolution. Because it removes the need to specify a fitness function prior to exploration, the user can change evaluation criteria over time. This is especially important when the fitness function is unknown or not easily specified beforehand. This paper describes a system that allows the user to evaluate and evolve small collections of candidate solutions that represent the range of available solutions in order to explore parameter ranges of interest. Although users may be expert with respect to their particular tasks, some may be novices in relation to their software. For them, interactive evolution removes syntactic barriers to the specification of alternative candidate solutions and navigation amongst them. Expert users benefit from exposure to candidate solutions outside of their prior experience and so may find solutions that better meet their needs.

INTRODUCTION

Consider that every parameter under consideration has some permissible values, either discrete or continuous. The space defined by these parameters is the Cartesian product of all values for all parameters. For N parameters, each N -tuple of values is a distinct point in the parameter space, as follows:

$$\langle v_1, v_2, \dots, v_N \rangle \in P_1 \times P_2 \times \dots \times P_N$$

One might easily encounter what is called the “curse of dimensionality” or a “combinatorial explosion” when exploring this space. If each of 3 parameters have just 10 values, there are 1000 distinct tuples in this space, making an exhaustive search overwhelming. It is also fruitful to consider this parameter space as a solution space (Simon, 1977), since the expectation exists that a solution to the current problem can be found amongst the combinations of parameters values.

The computer has the promise to democratize exploration of these parameter spaces, as suggested by Jessup (1992), for example. The promise, however, will not be fully realized until software addresses the issue of access. In general, the relationship between humans and computers can be viewed as either automatic, manual, or augmented (Kochhar et al., 1991). Automatic systems, like Mackinlay’s APT [A Presentation Tool] (1986), are generally prescriptive and work with very little input from the user to determine the visual representation for the user. Manual systems, like AVS [Advanced Visualization

System] described by Upson et al. (1989), provide the building blocks but require the user to take responsibility to assemble them. Augmented systems encompass a wide range of designs, including examples such as VISTA (Senay and Ignatius, 1994) and Design Galleries (Marks et al., 1997), and are intended to allow the computer to support the user's activities by working with the user.

Although many, including Foley and Ribarsky (1994) consider that automatic systems are the only feasible solution to the curse of dimensionality, these systems presuppose a complete, or at least adequate, articulation of the problem. Winograd and Flores (1985), however, contend that such a definitive articulation is not possible but it is rather an ongoing process. This paper describes an augmented system, called *cogito*, that employs interactive evolution to allow for an initially incomplete articulation of the problem to be refined as the interaction progresses. The rest of this paper is organized as follows. The next section details the need for a new approach to parameter space exploration. The following section details the design of the present software system, called *cogito*. The implementation section describes some details of the prototype version now in use and relates some of the experience gained thus far. Finally, the last section describes the conclusions drawn and areas for future work.

MOTIVATION

In all cases when a user of computer software wants to find a solution, either he or she knows what is needed or wants to explore what is available. If the user is familiar with computers in general and the software specifically, then it may be an easy matter specify a particular solution. However, this user may also be unaware of better solutions that exist outside of his or her experience, content with a local rather than global maximum. When exploration is indicated, it may involve considerable work even for an expert user. In the domain of data plotting, Bertin (1983) rightly indicated that it would take "more patience than imagination to generate 100 DIFFERENT FIGURES from the same data." That patience may give out before the best solution may be found is a serious issue.

If a complete articulation could be done for any one instance of the problem, that solution would not have to be located again. And, there would be an algorithmic mapping of the problem to the solution. However, there are many issues that work to create new instances of the problem and solution: task and individual experience being just two examples (Casner, 1991). If the solution must be adapted for the individual and the circumstances, there must be a means by which this adaptation can be achieved to support Hadamard's (1954) view that, "in the end, I know only one inventor, that is myself."

For users who do not have sufficient expertise to operate the particular software, the translation between the semantics of the problem and the syntax of the software can become more onerous. Norman (1988) uses the term "gulf of execution" to describe the distance between the user's concept of what is to be achieved and how to realize that goal using the software. A similar difficulty arises when evaluating the output of the software, in an effort to reconcile the computer's state with the original goal state. This is what Norman called the "gulf of evaluation." Most often, an iterative process is required to narrow these gulfs in order to reach a satisfactory outcome, the length of the process reflects the difficulty. From the perspective of solution spaces, Perkins (1995) classifies

spaces depending on the amount of clues available: Klondike spaces are clue-poor and homing spaces are clue-rich. Ideally, tools would enable users to transform a Klondike space into a homing space for themselves.

Beyond the specifics of the commands, the translation of the user's internal representation to one that may be communicated externally may also involve translation (Paivio, 1986), specifically when the goal is visual in nature. A user's internal representation of his or her goal must be converted to the programmer's internal representation and then that converted to the software. Even if the user has a mental picture of what is to be achieved, it can still be difficult to communicate verbally. There is increasing evidence, in part reported by Schooler and Engstler-Schooler (1990), which indicates that attempts to verbalize descriptions of non-reportable phenomena may overshadow the original information. This is particularly important for people who do not have a strong graphical vocabulary and this extra difficulty can keep people from finding what they need. If a programmer is employed, translation errors between the user and programmer may also become a serious issue.

DESIGN

Although it is difficult for a user to generate hundreds of alternatives, it is not so for the computer. If it is possible to parameterize the form of the solution, all available solutions that fit that model can be computed automatically. Instead of quickly returning to the problem of the combinatorial explosion, a different approach is used. The parametric representation of the solution makes it well-suited to a genetic algorithm (Goldberg, 1989), but the standard implementations may be too automatic. Other automatic systems (APT), and some augmented ones (Design Galleries) also automate this initial review by having the computer first evaluate each solution. Such a specification can unwittingly eliminate desirable alternatives. Even though Design Galleries presents the user with a sampling of the full range of solutions, based on the results of the initial evaluation, specification of the metric *a priori* may represent a significant hurdle especially when the requirements are hard to specify, or even articulate.

The involvement of users is important from the perspective of personalization. Therefore, the design of *cogito* relies on the user beginning with the whole space and making decisions at the time of inspection instead of beforehand. Rather than attempt an exhaustive search for the optimal solution, the focus is to find satisficing solutions instead (Simon, 1977). Yet, if not all solutions are likely to be directly evaluated, the system must provide an effective means to choose the combinations that will be reviewed, since all will be available. This is done by allowing the space to be partitioned and having each partition sampled.

The paradigm of interactive evolution is well-suited to a system that is designed to encourage individual involvement. The user is able to see alternatives without having the burden of directly specifying them. This sort of approach, which has the benefit of a low syntactic burden for the user by virtue of its direct manipulation style, has been used to great effect by Sims (1991), for example.

With respect to Perkins, the visual interface style of interactive evolution allows users to explore and locate landmarks within the solution space. In this

regard, Boden (1996) was critical of Sims' system since its genetic programming component (that permitted the functionality to also evolve) made it insufficiently deterministic, compared to others like that of Todd and Latham (1992), which allowed more directed exploration. This determinism is also a goal of the *cogito* system.

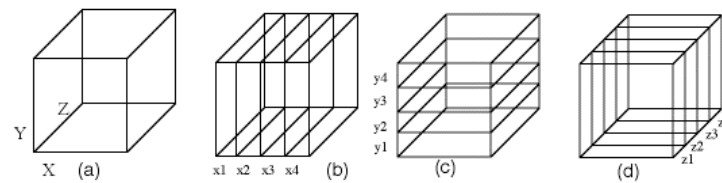


Figure 1: Consider a three-dimensional parameter space, depicted in (a), with parameters X, Y, and Z (each with 4 values). Organizing the space in terms of any of those 3 axes leads to other states shown (b-d). For example, as in (b), if space is partitioned according to values in X, those 4 can be shown sequentially while values from Y and Z can be chosen pseudo-randomly.

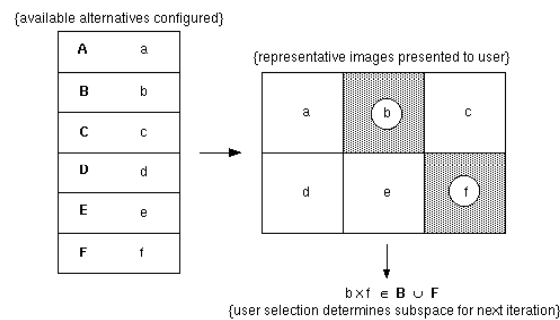


Figure 2: Schematic view of the interface: the space of available alternatives is grouped according to user-specified criteria (see Figure 1). Each group (A-F) has a representative element (a-f) that is displayed to the user. The subspace for the next iteration is based on the user selection (b and f).

As the solution space is explored and landmarks are found, the user is able to develop a mental model of the solution space just as one develops a model of an application's interface (Rosson and Carroll, 2002). This process helps the user to gradually articulate his or her needs in a particular situation. Despite the fact that a complete articulation, especially before any exploration, is not possible (Winograd and Flores, 1985), Wegner (1997) asserts that this interactive approach is also much more powerful.

Based on the preceding two figures, consider the following simple example. Imagine that the user has organized the space according to the X parameter, and subsequently chosen samples from the x1 and x3 partitions. If these selections were in fact $\langle x1, y3, z2 \rangle$ and $\langle x3, y4, z1 \rangle$, the newly generated subspace consistent with those selections would also contain $\langle x1, y3, z1 \rangle$, $\langle x1, y4, z2 \rangle$, $\langle x1, y4, z1 \rangle$, $\langle x3, y3, z2 \rangle$, $\langle x3, y3, z1 \rangle$, and $\langle x3, y4, z2 \rangle$. The new subspace contains 8 of the original 64 available combinations.

The interaction is not unlike the notion of computational steering (Bullock et al., 2002), in the sense that the user can guide the computation and focus interest on areas of the solution space that he or she finds particularly important. Because users evaluate the computational results for a few points in the solution space without ever seeing the whole space, it is expected that users will not become overwhelmed by the total number of available alternatives.

The benefit of the genetic algorithm in searching the solution space is its global character that allows global extrema to be found over local ones. The common genetic operations of crossover (illustrated in the earlier example) and mutation provide very powerful means for exploring the space. Crossover, for example, allows similar solution-space points to be located. Mutation, which allows unexpected values to be added to combinations, could be used to balance the coverage of a space (for example, by periodically adding infrequently-visited values to the mix). This approach benefits the user by encouraging deeper understanding of the problem through involvement and exploration, and ultimately providing direct access to a variety of solutions. It provides better access to the solution space than the popular spreadsheet paradigm (Jankun-Kelly and Ma, 2001, and Chi, 1998) because there the user must still exercise great caution in choosing the parameters and values to explore, all done two at a time.

IMPLEMENTATION AND EXPERIENCE

This system was implemented in C++ under the IRIX operating system using X Windows for the interface and OpenInventor (Wernecke, 1994) for the graphics. It permitted a generic approach to interactive evolution. The first applications written for the system were visual, where each valid combination of values from the parameters was realizable as a graphic (see Figure 3 below). Compatibilities were defined to describe rules for combining values (code modules) from different parameters. The executable code to realize the graphics was primarily stored in dynamic shared objects that were loaded at run-time. A text file enabled a description of the application and the space in terms of parameters and values to the *cogito* system that managed the exploration of that solution space by the user.

Rather than complete particular solutions, this approach allowed the programmer to create specific modules to correspond with each value for particular parameters (but the number of modules may be less than the number of values, since the same code could be reused with different arguments.)

This interface allows the user to organize his or her view of the space according to different parameters, creating a hierarchical organization of the space best suited to the user. In addition to allowing the search space to be narrowed, it can also be expanded by re-introducing all values for a particular parameter, for example.

Figure 3 shows the first two screens of an interaction where the user moves towards a desired visual representation. From a very large initial space that could contain over 25 million points, a user can quickly select interesting visual representations and see the system display alternatives consistent with those selections, through application of a crossover operation.

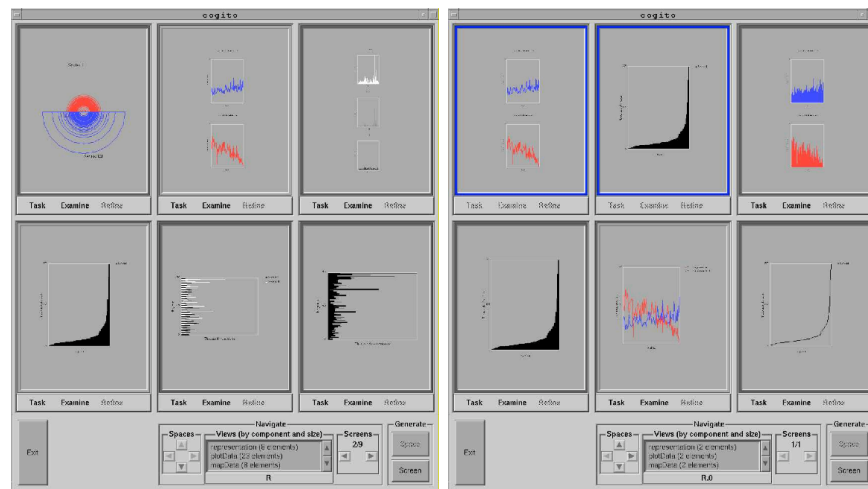


Figure 3: Two screens in an interaction with the *cogito* system. In the left screen, the user selects the top middle and bottom left cells. In the subsequent screen on the right, the system first displays the two “parent” selections then other alternative visual representations that are consistent with those selections, as determined by the application of a crossover operation. In this example, the user has chosen to proceed by selecting the bottom middle cell, although he or she may view more alternatives before making any selections. Through this system interface, the user is also able to direct the search by reintroducing values for specific parameters before continuing with selections.

A preliminary user study asked participants to choose plots to answer questions about a set of data. The *cogito* system (with the interactive evolution interface shown in Figure 3) was tested against the same system with an interface more typical of traditional visualization systems. The study, described elsewhere (Hepting 2002, Hepting 1999), found a preference for the interactive evolution interface, in particular when accessing alternatives.

CONCLUSIONS AND FUTURE WORK

This new paradigm for access to a large, complex information space has proven effective for exploration. The user study indicated the need for both the global search provided by the interactive evolution interface and the local control available through the more typical interface. Therefore, the appropriate integration of those two access methods is now being examined. Although the user may have a conceptual map of the solution space, a graphical representation of this map is also being developed for the next version of the software. Such a map, and a more detailed history of explorations, will be valuable for individual users over time and also to facilitate collaboration between users. Overall, the system is being redesigned to accommodate a wider range of applications.

REFERENCES

- Bertin, J. Semiology of graphics: diagrams, networks, maps. University of Wisconsin Press, 1983.
- Boden, M. A., Creativity. In M. A. Boden, editor, *Artificial Intelligence: Handbook of Perception and Cognition*, Chapter 9. Academic Press, second edition, 1996.
- Bullock, S. et al., Prospects for Computational Steering of Evolutionary Computation. E. Bilotta et al. (Eds.). *Workshop Proceedings of the Eighth International Conference on Artificial Life*, Sydney, Australia, December 8-13, pages 131-137, MIT Press, Cambridge, MA, 2002.
- Casner, S. M. A task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics*, 10(2), April, 1991.
- Chi, E. H., Principles for information visualization spreadsheets. *IEEE Computer Graphics and Applications*, July/August 1998.
- Foley, J. & Ribarsky, W., Next-generation data visualization tools. In L. Rosenblum et al., editors, *Scientific Visualization*, Academic, 1994.
- Goldberg, D. E., Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading, MA, 1989.
- Hadamard, J., *The Psychology of Invention in the Mathematical Field*. Dover, 1954.
- Hepting, D. H., A New Paradigm for Computer-Aided Exploration, Ph.D. Thesis, Simon Fraser University, Canada, 1999.
- Hepting, D. H., Towards a visual interface for information visualization. Proceedings of the 6th International Conference on Information Visualization, IEEE Press, pp. 295-302, 2002.
- Jankun-Kelly, T. J. & Ma, K.-L., Visualization Exploration and Encapsulation via a Spreadsheet-Like Interface, *IEEE Transactions on Visualization and Computer Graphics*, 7(3), pages 275-287, 2001.
- Jessup, M.E., Scientific visualization: Viewpoint on collaborations of art, science, and engineering. *SIGBIO Newsletter*, February, pages 1-9, 1992.
- Kochhar, S. et al., M., Interaction paradigms for human-computer cooperation in graphical-object modelling. In S. MacKay and E. M. Kidd, editors, *Proceedings of Graphics Interface '91*, 1991.
- Mackinlay, J., Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2), 1986.
- Marks, J. et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Computer Graphics: SIGGRAPH '97 Conference Proceedings*, 1997.
- Norman, D.A. *The psychology of everyday things*. Basic Books, New York, 1988
- Paivio, A. *Mental representations: A dual-coding approach*. Oxford University Press, 1986.
- Perkins, D. N., Insight in minds and genes. In R. J. Sternberg and J. E. Davidson, editors, *The Nature of Insight*, pages 495-533. MIT Press, Cambridge, MA, 1995.
- Rosson, M. B. & Carroll, J. M., Usability Engineering: Scenario-based development of human-computer interaction. Morgan-Kaufmann, 2002.
- Schooler, J.W. & Engstler-Schooler, T.Y. Verbal overshadowing of visual memories: Some things are better left unsaid. *Cognitive Psychology*, volume 22, pages 36-71, 1990.
- Senay, H., & Ignatius, E., A knowledge-based system for visualization design. *IEEE Computer Graphics and Applications*, 14(6), 1994.
- Simon, H., *Models of Discovery*, Reidel, 1977.
- Sims, K., Artificial evolution in computer graphics. In R. J. Beach, editor, *Computer Graphics: SIGGRAPH '91 Conference Proceedings*, pages 319-328. ACM Press, 1991.
- Todd, S. and Latham, W., *Evolutionary art and computers*. Academic Press, London, 1992.
- Upson, C. et al., The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4), pages 30-42, 1989.
- Wegner, P., Why Interaction Is More Powerful Than Algorithms, *Communications of the ACM*, 40(5), pages 80-91, 1997.
- Wernecke, J., *The Inventor Mentor*. Addison-Wesley, 1994.
- Winograd, T. and C. F. Flores. *Understanding Computers and Cognition*. Ablex, Norwood, New Jersey, USA, 1985.