14

Interactive Evolutionary Algorithms

This chapter discusses the topic of **interactive evolution**, where the measure of a solution's fitness is provided by a human's subjective judgement, rather than by some predefined model of a problem. Of course, the world around us is full of examples of human intervention in biological evolution, in the form of pets, garden flowers, food crops and farm animals. Applications of **Interactive Evolutionary Algorithms** (IEAs) range from capturing aesthetics in art and design, to the personalisation of artefacts such as medical devices. When including humans 'in the loop' we must consider their peculiar characteristics. On one hand, they can provide insight and guidance beyond simply selecting parents for breeding. On the other, they can be inconsistent, and are prone to fatigue and loss of attention. These factors make it inappropriate to use the 'traditional' model of an EA generating possibly thousands of candidate solutions. This chapter describes and explains some of the major algorithmic changes that have been proposed to cope with these issues.

14.1 Characteristics of Interactive Evolution

The defining feature of IEAs is that the user effectively becomes part of the system, acting as a guiding oracle to control the evolutionary process. As a starting point, consider agricultural breeding, where human interference changes the reproductive process. Based on functional (faster horses, higher yields) or aesthetic (nicer cats, brighter flowers) judgements, a supervisor selects the individuals that are allowed to reproduce. Over time, new types of individuals emerge that meet the human expectations better than their ancestors. From this familiar process we can now start to distinguish some particular features that impact on the design of IEAs.

A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*,
Natural Computing Series, DOI 10.1007/978-3-662-44874-8_14

14.1.1 The Effect of Time

Many crop plants have annual life cycles, and farm animals may take years to reach maturity. Thus in plant breeding and animal husbandry we are used to the idea of many humans playing successive roles, in a process that may take centuries. Although both of these are nonstationary problems, since fashions in flowers, pets, and food change, the relative infrequency of intervention, and the importance of making the right choice, makes human input fairly reliable.

However, when we are considering simulated evolution, then many decisions may be required fairly rapidly from a single user. Also, each individual decision may seem less important, since we are used to the idea that we can re-run computer programs. In these cases human fatigue, and its effect on the consistency of evaluation, becomes a major factor, even if the person is attempting to apply a well-understood standard. People have a natural limited attention span, and, like fatigue, a loss of engagement with the process has been shown to lead to increasingly erratic decisions as the user performs more and more evaluations [76]. Thus, from the perspective of evolutionary time, there is a need to avoid lengthy evolution over hundreds or thousands of generations, and instead focus on making rapid gains to fit in with human needs.

Even taking this into account, from the perspective of wall-clock time IEAs can be slow to run, as it usually takes much longer for a person to make a decision than it does to calculate a mathematical fitness function. The net effect of time constraints and human cognitive limitations is that typically only a small fraction of the search space is considered. This is the major challenge facing IEA designers, and it is common for successful IEA applications to employ one or more different approaches to alleviate this issue. A sense of progress can be achieved by evaluating fewer solutions per generation — either via small populations, or by only evaluating some of a large population. Frustration can be reduced by not losing good solutions. More generally, engagement can be increased, and the onset of fatigue delayed, by allowing the user more direct input into the process.

14.1.2 The Effect of Context: What Has Gone Before

Closely related to the issue of time, or the length of the search process, is that of context. By this we mean that human expectations, and ideas about what is a good solution, change in response to what evolution produces. If (as we hope) people are pleasantly surprised by what they see, then their expectations rise. This can mean that a solution might be judged average early in a run, but only sub-standard after more solutions have been presented. Alternatively, after a few generations of viewing similar solutions, a user may decide that they are in a 'blind alley'. In that case they might wish to return to what were previously thought to be only average solutions to explore their potential. Both of these imply a need to generate a diverse range of solutions — either by increasing the rate of mutation or by a restart mechanism, or by maintaining some kind of archive.

14.1.3 Advantages of IEAs

Despite, or perhaps because of, the issues identified above, it is worth reiterating the potential advantages of incorporating humans directly into the evolutionary cycle. These were summarised by Bentley and Corne in [50]:

- Handling situations with no clear fitness function. If the reasons for preferring certain solutions cannot be formalised, no fitness function can be specified and implemented within the EA code. Subjective user selection circumvents this problem. It is also helpful when the objectives and preferences are changeable, as it avoids having to rewrite the fitness function.
- Improved search ability. If evolution gets stuck, the user can redirect search by changing his or her guiding principle.
- Increased exploration and diversity. The longer the user 'plays' with the system, the more and more diverse solutions will be encountered.

14.2 Algorithmic Approaches to the Challenges of IEAs

Having outlined the issues confronting IEA designers, we now discuss some of the ways in which they have been addressed.

14.2.1 Interactive Selection and Population Size

In general, the user can influence selection in various ways. The influence can be very direct, for example, actually choosing the individuals that are allowed to reproduce. Alternatively, it can be indirect — by defining the fitness values or perhaps only sorting the population, and then using one of the selection mechanisms described in Sect. 5.2. In all cases (even in the indirect one) the user's influence is named **subjective selection**, and in an evolutionary art context the term **aesthetic selection** is often used.

Population size can be an important issue for a number of reasons. For visual tasks, computer screens have a fixed size and humans need a certain minimum image resolution, which limits how many solutions can be viewed at once. If the artefacts being evolved are not visual (or if they are, but the population cannot be viewed simultaneously), then the user has to rely heavily on memory to rank or choose individuals. Psychology provides the rule of thumb that people only hold around seven things in memory. Another aspect is that if a screenful of solutions are being ranked, the number of pairwise comparisons and decisions to be made grows with the square of the number onscreen. For all these reasons, interactive EAs frequently work with relatively small populations. Also, small populations can help keep the user engaged by providing them with a sense of progress.

Since the mid-2000s a number of authors have investigated ways of dealing with problems that exhibit a mixture of quantitative and qualitative aspects.

The usual approach is to use a multi-objective EA as described in Chap. 12, and the increased solution space usually requires bigger population sizes. However, not all of these need be evaluated by the user, since some can be ruled out on the basis of inferior performance on quantitative metrics.

14.2.2 Interaction in the Variation Process

However controversially, advances in biology have meant that agricultural breeding can now act not just on the selection process, but also on variation operators — for example, inserting genes to produce genetically modified crops. Similarly, Interactive Artificial Evolution can permit direct user intervention in the variation process. In some cases this is implicit - for example allowing the user to periodically adjust the choice and parameterisation of variation operators. Caleb-Solly and Smith [76] used the score given to an individual to control mutation — so that 'worse' individuals were more likely to have values changed, or to have values changed by a larger amount. As well as noting performance benefits, they also reported an effect which is typical of human interactions with Artificial Intelligence. Specifically, users' behaviour changed over time, as they got used to the system and developed a perception of how it would respond to their input. This manifested as users alternating between *exploiting* promising regions (awarding high scores to cause small changes), and *exploring* new regions (awarding low scores as a kind of reset button).

More explicit forms of control may also be used. For instance, interactive evolutionary timetabling might allow planners to inspect promising solutions, interchange events by hand and place the modified solutions back to the population. This is an explicit Lamarckian influence on variation.

Both types of algorithmic adaptation have in common the desire to maximise the value of each interaction with a user. Methods for directly affecting variation are usually designed with the goal of increasing the rate of adaptation towards good solutions by removing some of the black-box elements of search. However, in practice a nice synergy has been observed – in fact the ability to guide search actually increases user engagement, and so delays the onset of fatigue [335]. Again, this mimics results elsewhere in interactive AI.

14.2.3 Methods for Reducing the Frequency of User Interactions

The third major algorithmic adaptation attempts to reduce the number of solutions a user is asked to evaluate, while simultaneously maintaining large populations and/or using many generations. This is done by use of a **surrogate fitness function** which attempts to approximate the decisions a human would make. In fact this approach is also used to reduce the wall-clock time of evolutionary algorithms working with heavily time-intensive fitness functions.

Typically the surrogate fitness models are adaptive and attempt to learn to reflect the users' decisions. Normally the model is used to provide fitness for all solutions, and only occasionally is the user asked to evaluate a solution. This input can then be used as feedback to update the parameters of the model used. Hopefully, over time, the fitness values predicted by the surrogate model match the users' input better and better. Models used range from the simple (offspring receive the mean fitness of their parents) to the use of advanced machine learning algorithms such as Support Vector Machines and Neural Networks. The crucial decisions from an algorithmic perspective are: how complex a model is necessary; how often should the real fitness function be invoked (in this case a human evaluation); how should solutions be chosen to be properly evaluated; and how should the surrogate model be updated to reflect the new information obtained. Full discussion is beyond the scope of this book, but the interested reader will find a good review of this type of approach in [235] and a discussion of recent developments and the current research issues in [236].

14.3 Interactive Evolution as Design vs. Optimisation

Interactive evolution is often related to evolutionary art and design. It can even be argued that *evolution is design*, rather than optimisation. From this conceptual perspective the canonical task of (natural or computer) evolution is to design good solutions for specific challenges. Many exponents have arrived at a view that distinguishes parameter optimisation and exploration [48, 49], the main underlying difference being the representation of a solution.

Many problems can be solved by defining a parameterised model of possible solutions and seeking the parameter values that encode an optimal solution. This encoding is 'knowledge-rich' in the sense that the appropriate parameters must be chosen intelligently – if there is no parameter for a feature that influences solution quality that feature can never be modified, different values cannot be compared, and possibly good solutions will be overlooked. *Design optimisation* typically uses this type of representation. Propagating changes across generations, an EA acts as an optimiser in the parameter space.

An alternative to parameterised representations is component-based representation. Here a set of low-level components is defined, and solutions are constructed from these components. This is a 'knowledge-lean' representation with possibly no, or only weak, assumptions of relationships between components and the ways they can be assembled. IEAs for evolutionary design commonly use this sort of representation. Also known as **Generative and Developmental Systems**, these quite naturally give rise to exploration, aiming at identifying novel and good solutions, where novelty can be more important than optimality (which might not even be definable). In this paradigm, evolution works as a *design discovery* engine, discovering new designs and helping to identifying new design principles by analysing the evolved designs [303]. The basic template for interactive evolutionary design systems consists of five components [50]:

1. A phenotype definition specifying the application-specific kind of objects we are to evolve.
2. A genotype representation, where the genes represent (directly or indirectly) a variable number of components that make up a phenotype.
3. A decoder, often called growth function or **embryogeny**, defining the mapping process from genotypes to phenotypes.
4. A solution evaluation facility allowing the user to perform selection within the evolutionary cycle in an interactive way.
5. An evolutionary algorithm to carry out the search.

This scheme can be used to evolve objects with an amazing variety, including Dawkins' pioneering Biomorphs [101], coffee tables [47], images imitating works of the artist M.C. Escher [138], scenes of medieval towns [405], music [307], and art such as the 'Mondriaan Evolver' [437] or collaborative online 'art breeder' systems.[1] The basics of such evolutionary art systems are the same as those of evolutionary design in general: some evolvable genotype is specified that encodes an interpretable phenotype. These might be visual (two-dimensional images, animations, and three-dimensional objects) or audible (sounds, ringtones, music). The main feature that distinguishes the application of IEAs to art from other forms of design lies in the intention: the evolved objects are simply to please the user, and need not serve any practical purpose [355].

14.4 Example Application: Automatic Elicitation of User Preferences

Pauplin et al. [335] described the development of an interactive tool used by quality control engineers. The task is to create customised software that automatically detects and highlights defects in images of processed items. The context is a flexible manufacturing environment where changes in equipment (cameras, lighting, machinery), or in the product being created, mean that the system frequently needs reconfiguring. The concept exploited here is that interactive evolution provides a means to automatically elicit user preferences. This can replace the time-consuming, and often costly, process where a series of interviews are followed up by an image-processing expert implementing a system, which then needs iterative refinement. Instead, a good image processing system is evolved interactively. The user is shown a series of images of products containing various defects. Candidate image processing systems are applied to segment these images and the results are shown by coloured lines. Each different candidate solution will segment images in a different way, so the question asked of the user by the interface in Fig. 14.1 becomes: "Which of these sets of images has the lines drawn to separate out things you might be interested in, and how well do they do?"

[1] `http://picbreeder.org` or `http://endlessforms.com/`

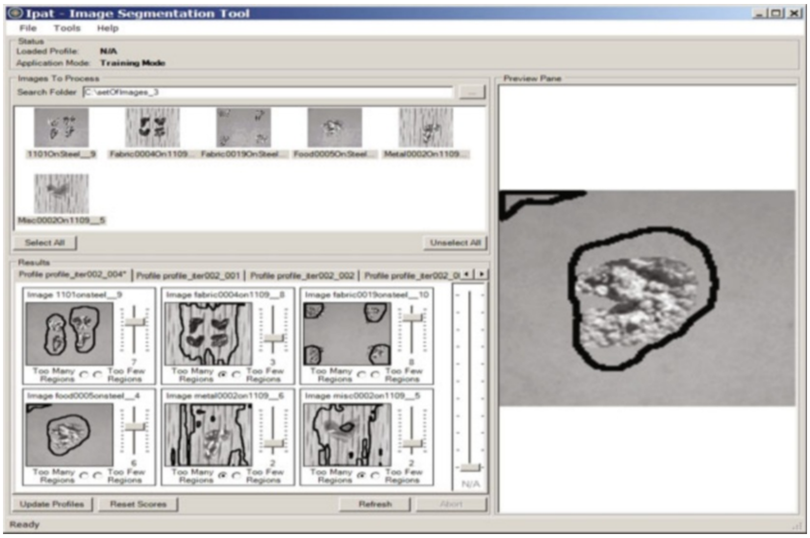

Fig. 14.1. User interface for the IPAT tool. Top window shows 'raw' images, bottom window shows segmentation results for six different images processed by current solution, right-hand side shows enlarged preview of image under cursor

In order to reduce the influence of human preconceptions, the component-based approach is used. A candidate solution, i.e., an image processing system, is composed of a variable number of image processing kernels. Each kernel consists of a module drawn from a large library of routines, together with any relevant parameter values. Kernels may be composed in a range of ways (parallel/sequential) and may be inverted to produce the final segmented version of an input image.

The algorithm, and in particular the detailed graphical user interface (GUI) design, were based on the twin principles of reducing the number and complexity of the decisions that the user had to make at each stage, and maximising the information extracted from those decisions. The user's concentration is treated as a limited resource that should be used sparingly, and is reduced by every mouse click, whereas attention is reinforced when the system appears to be working in collaboration with them. Thus, for example, the interface shown in Fig. 14.1 uses widgets that change the focus and images displayed in the right-hand 'preview' window according to mouse position rather than clicks.

Typically a session will examine several images at once, containing different types of defects, and try to find a compromise that segments them all. The interface handles this multi-objective search in two ways. Users can assign a partial fitness for each image segmented by the current solution (in which case these are averaged), or can assign an overall fitness for the solution. One mouse click allows them to switch between comparing all images segmented by a solution, or all solutions' segmentation of a single image.

Behind the interface, the principal algorithmic adaptations were:

- To reduce the number of decisions required, users were only asked to provide a score for the best solution in each generation, and only six images were presented. This necessitated a $(1 + 5)$ selection strategy.
- Only 10 different fitness levels were used to reduce the cognitive burden of scoring.
- In response to videos showing users' surprise and frustration when evolution appeared to 'forget' good solutions, one copy of the chosen solution was always shown unchanged. Comparative results showed that this evaluation regime increased the number of user interactions and the quality of the final solution (according to subjective and objective criteria).
- The applied mutation rate was determined by the users' scoring, decreasing from a high of 50% for a score of 0 to 0% for a 'perfect' solution. Results demonstrated that users' behaviour changed over time as they gained experience — so that effectively they used a low score as a reset button to counteract the high selection pressure of the $(1 + 5)$ strategy.
- Hint buttons were provided to allow the user to directly guide the direction of search. In this case domain-specific knowledge was applied to translate user input such as 'too many regions segmented' into biases in the mutation rate applied to parts of the solutions.

The most obvious component not present in this tool was the use of a surrogate fitness function. Nevertheless, the results showed that naive users were able to create image processing systems that accurately segmented a range of images, from scratch, in fewer than 20 iterations.

For exercises and recommended reading for this chapter, please visit `www.evolutionarycomputation.org`.