Conflict Resolution and a Framework for Collaborative Interactive Evolution

Sean R. Szumlanski, Annie S. Wu, Charles E. Hughes

School of EECS
University of Central Florida
Orlando, FL 32816-2362
{seansz, aswu, ceh}@cs.ucf.edu

Abstract

Interactive evolutionary computation (IEC) has proven useful in a variety of applications by combining the subjective evaluation of a user with the massive parallel search power of the genetic algorithm (GA). Here, we articulate a framework for an extension of IEC into collaborative interactive evolution, in which multiple users guide the evolutionary process. In doing so, we introduce the ability for users to combine their efforts for the purpose of evolving effective solutions to problems. This necessarily gives rise to the possibility of conflict between users. We draw on the salient features of the GA to resolve these conflicts and lay the foundation for this new paradigm to be used as a tool for conflict resolution in complex group-wise human-computer interaction tasks.

Introduction

Increasingly, we find ourselves faced with the challenge of implementing systems that afford utilization by multiple users simultaneously. This has led to a great need within the field of human-computer interaction for novel and effective approaches to conflict resolution, particularly for systems that provide only a single display for multiple users, or Single Display Groupware (SDG) (Stewart *et al.* 1999). To facilitate this conflict resolution in this sort of collaborative environment, we turn to the genetic algorithm, but with a set of modifications that constitute a new approach within the field: collaborative interactive evolution.

We will chart the evolution, so to speak, of genetic algorithms into interactive evolutionary computation (IEC), and examine some of the strengths of IEC. We will then present what we see to be a natural extension of IEC into collaborative interactive evolution (CIE), articulating its framework and defining the domain of problems for which we expect it to be applicable. We will then describe the system we implemented to test the CIE framework, which engages multiple users simultaneously with a task of creativity and complexity. After discussing the results of experimentation, we will outline the possibilities for the direction of future research in CIE.

Background

In his original work on the genetic algorithm (GA), Holland (1975) presented a general-purpose approach to problem solving in which potential solutions to a problem are encoded as (typically binary) strings called *chromosomes*. When the evolutionary algorithm begins, a collection of chromosomes is initialized randomly, together forming a *population* on which the algorithm operates.

The GA employs *selection* to determine which chromosomes in the population should be given an opportunity to propagate their information. Selection is guided by a *fitness function*, a black box component of the algorithm responsible for determining the viability of potential solutions represented by the chromosomes of the population. Chromosomes with higher fitness values typically have a greater chance of being selected for mating.

A variety of operators may be applied to the chromosomes selected from the initial population, thereby modifying them and giving rise to a new *generation* – a separate population of chromosomes that is the result of an application of genetic operators to the previous generation's most fit individuals. Primary among the operators used in GAs today are *crossover* and *mutation*. Crossover allows for an exchange of information between two chromosomes, and, ideally, exploits information within the population that contributes to high fitness values in the chromosomes selected for crossover. In contrast, mutation traverses a chromosome and determines, based on a specified *mutation rate*, whether each gene should remain the same or be mutated (e.g., in the case of binary representations, whether a given bit should be flipped). In this way, mutation may give rise to genes that are not present within the previous generation of evolution.

This process continues until a predetermined number of generations or some maximum fitness value is reached. At that point, the process terminates and the individual with the highest fitness, as assigned by the fitness function, is

presented as the most promising candidate for a solution to the problem in question.

In pseudocode, the algorithm is as follows:

```
Procedure GA {
    Initialize population;
    While termination condition not satisfied {
      Select parents from population;
      Apply genetic operators;
      Evaluate offspring using fitness function;
      Replace parent population with offspring;
    }
}
```

Figure 1. Standard GA pseudocode.

We saw an emergence in the 1980s of interactive evolutionary computation (IEC) – the application of genetic algorithms in which the fitness function of the GA was replaced by a human who evaluated solutions by hand.

The GA pseudocode, modified for IEC, thus becomes:

```
Procedure IEC {
    Initialize population;
    While termination condition not satisfied {
      Select parents from population;
      Apply genetic operators;
      Human evaluates offspring subjectively;
      Replace parent population with offspring;
    }
}
```

Figure 2. GA modified for interactive evolutionary computation.

Initially, IEC was used for creative and artistic applications. In the original work on IEC, Richard Dawkins (1989) evolved biomorphs, or computer generated graphics that resembled insects. Smith (1991) later expanded this work by employing the use of genetic crossover and maintaining a larger population in his experimental setup. Computer graphics was one of the first domains to draw on the power of IEC, and has continued to make important use of the technique. Prominent in this field is Karl Sims, who has been particularly active in the area of evolutionary art (Sims 1991).

Since its inception, IEC has also seen a rise in popularity for non-artistic applications. Graf and Banzhaf (1995) used IEC to simulate and study natural evolution. More recently, IEC has been applied to more practical problems such as evolving blends of coffee interactively (Herdy 1997) and producing believable models of human walking patterns (Lim and Thalmann 2000). Takagi (2001) provides a comprehensive and fairly up-to-date survey of the state of the art for IEC.

These latest applications are representative of the type of problem domain in which IEC has been shown to be most effective. It is widely recognized that IEC is most appropriate for problems with poorly defined search spaces that are not easily quantified, but which have solution candidates that lend themselves nicely to subjective human evaluation. Takagi (2001) further suggests that IEC is best suited for search spaces where there is no single optimal solution, but where there is instead a large global optimal area associated with the problem, which requires human exploration. But it is in (Takagi 1998) that this idea is expounded most clearly by the argument that the success of IEC is owed largely to its ability to incorporate into its search human *KANSEI*, or the motives, psychology, and aesthetic preferences of human users, which cannot be adequately represented by traditional fitness functions.

If IEC is so effective at navigating broad globally optimal areas of fitness with the help of a single user, it seems natural to extend this idea to incorporate the guidance of multiple users. The rationale here is twofold. First, the fitness landscape of a problem in which multiple users are seeking a solution will necessarily have an array of optima as diverse as the users guiding the evolutionary process. Secondly, the optima will necessarily arise in terms of vague areas of optimality, as opposed to individual points.

These two properties of a fitness landscape that incorporates the *KANSEI* of multiple users fit the criteria for problem domains in which IEC seems to perform its best. Thus, there seems to be reason to believe that systems taking advantage of input from many users might benefit from IEC. However, these two properties also rely upon the assumption that the fitness of solutions is open to subjective interpretation by the users. Thus, we expect interactive evolution with multiple users to function best when the problem being approached is of a highly subjective nature.

But in order to utilize interactive evolution with multiple users, we must first establish a framework for such by modifying the existing algorithm. Here, we establish the new idea of collaborative interactive evolution (CIE). Specifically, we must decide how the input from multiple users (which will inevitably conflict at some point) should be combined. We are guided by a lesson from the greatest problem with IEC: human fatigue (Takagi 2001). When users are forced to subjectively evaluate the output of every generation of evolution, the process becomes tedious, and users can only be expected to hold up for a small number of generations. As such, we opt to gather user input only once in the CIE framework, at the onset of evolution. This gives users the opportunity to define the fitness landscape with their subjective preferences. From there, we rely on the salient features of the GA, including a certain degree of randomness coupled with its exploratory power, to give adequate coverage of the resulting landscape.

To put this strategy into effect, we still must combine the user input. For this purpose, we implement the use of fitness function templates. These are generic templates that inherently reflect conflict resolution policy, and govern the combination of user input as they define the fitness landscape. The fitness function template is left undefined to keep the framework as generic as possible, and can be

crafted for each implementation, just as the fitness function is left open to definition in the standard GA. We do, however, describe two specific fitness function templates in the following section.

This leaves us with the following pseudocode for CIE:

```
Procedure CIE {
    Users provide subjective input;
    Initialize population;
    While termination condition not satisfied {
      Select parents from population;
      Apply genetic operators;
      Evaluate offspring using fitness function
        templates and user input;
      Replace parent population with offspring;
    }
}
```

Figure 3. GA modified for collaborative interactive evolution.

Experimental Setup

In order to test the CIE framework, we implement a system of the sort described above, in which multiple users are engaged in a task that is both creative and complex and relies upon their subjective evaluation. In the experimental setup, users log into a web-based system each day and design the physical and emotional makeup of a fictional character. Once all users have submitted their data for the day, CIE is used in conjunction with two different fitness function templates to evolve new characters. The two best individuals produced by CIE with each fitness function template are placed in a tournament pool along with two other characters derived from functional computations, for a total of six characters in the tournament pool.

Of the two characters derived from simple functions, one is comprised of the most frequently requested phenotype for each character trait in the users' input. The other character is the one whose phenotype for each gene corresponds to the genotype given by averaging all user input for that gene on that day.

Using the web-based system, participants examine each of the six characters in the tournament pool and then vote for the one they think is the best, by whatever subjective measures they see fit. In order to motivate users to create and vote for believable and interesting characters, those characters are used to tell an on-going interactive story, the direction of which is determined by the types of characters evolved by the users. This process is outlined below in Figure 4.

The experiment unfolds in two phases. In Phase 1, a consistent group of ten users logs into the web-based system each day for five days, giving rise to a total of five characters. In Phase 2 a different group of ten users logs into the system over the course of seven days, giving rise to seven characters. However, these users are permitted to skip days as their schedules require, thereby giving rise to a set of data more likely to represent the kinds of conditions one might encounter in the real world, where some users remain focused on the system while others provide only intermittent feedback and input.

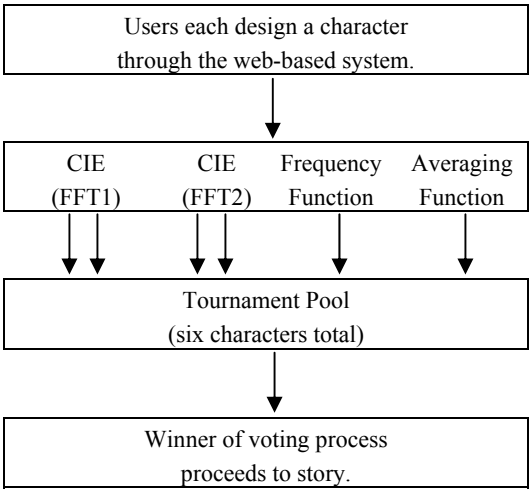

Figure 4. Experimental procedure followed each day.

Chromosomes

Users define six physical traits and six psychological attributes for each character they create as a prelude to evolution each day. Within the GA, each trait is represented as a single gene on the chromosome. Because each trait can take on eight values, and encoding is binary, each gene is comprised of three bits. The chromosome, therefore, is a string of 36 bits (Figure 5).

Physical Characteristics						Psychological Features					
001	011	010	101	011	010	111	000	001	010	101	011
1	2	3	4	5	6	7	8	9	10	11	12

Phenotypes for Genes as Numbered:

1. Skin Color
2. Face Shape and Sex
3. Hair Style
4. Hat
5. Eye Color
6. Glasses
7. Happiness
8. Sadness
9. Anger
10. Suspiciousness
11. Curiosity
12. Ego

Figure 5. Chromosome structure for encoding a character's physical and personality traits.

GA Parameters

The GA uses fitness proportional selection with a .01 mutation rate and single-point crossover. Evolution takes place over the course of 100 generations. In order to simulate the performance of a real-time interactive system operating under tight time constraints, we perform only five runs. Despite the small number of runs, the tendency is

toward homogeneity among the final solutions produced by the same fitness function template.

Fitness Function Template 1

For each gene on a chromosome, Fitness Function Template 1 (FFT1) awards a single point of fitness for each user who requests that specific gene at the time of character creation each day. In this way, FFT1 models a most-frequently-requested-gene function, but the limited number of generations and complexity of the fitness landscape prevent it from evolving the exact same character as the actual frequency computation method.

Fitness Function Template 2

Fitness Function Template 2 (FFT2) awards a certain number of fitness points for each user, based on the number of genes in the chromosome that match that user's hand-crafted character (Figure 6).

Number of Gene Matches

1	2	3	4	5	6	7	8	9	10	11	12
9	8	6	4	5	6	7	8	9	10	11	12

Fitness Reward

Figure 6. For each user, FFT2 awards a number of points based on the number of genes matched.

In order to encourage the GA to match as many genes in as many users' hand-crafted characters as possible, a high reward is given any time a chromosome contains a large number of the genes requested by a given user. If a chromosome matches all 12 traits requested by a user, then it is awarded 12 fitness points, but if it only matches 4 of the requested traits, it only gains 4 points toward its fitness.

However, a large fitness reward is also given when only a few genes match the request of a given user, with the hope of encouraging the GA to match at least one trait requested by each user rather than allowing any one user's input to be totally ignored if a dominant subgroup of participants designs remarkably similar characters. Therefore, if a chromosome only matches 1, 2, or 3 traits requested by a certain user, it gains a fitness reward of 9, 8, or 6, respectively.

Results

We use a variety of mechanisms to determine whether users favor the characters evolved with CIE over the characters produced by averaging and frequency computation.

In this section, the performance of averaging and frequency computation for input combinations are combined to give an idea of just how effective the functional computations are in comparison to the CIE methods of combining user input. Even with those two functional methods combined, they do not surpass the success of either GA method.

Voting

After all user input is combined each day and candidate solutions are evolved using CIE with the two separate fitness function templates, as well as averaging and frequency computation, users examine the resulting characters and chose the one each favors the most. In this way, users directly indicate which of the methods for combining their (sometimes conflicting) input performs the best.

We see that in both phases of the experiments, users prefer characters produced by CIE at least twice as much as the characters produced by averaging and frequency computation. In the first phase of experiments, FFT1 garners 36% of all votes, compared with the 30% of the votes that go toward the functional combination methods. This contrast is starker when examining the number of wins; FFT1 produces three winning characters in the first phase of the experiment, while the functional methods only produce one, as does FFT2.

In the second phase of experiments, however, FFT2 emerges as the most successful method for combining input, with 42% of the votes and four winning characters overall, as opposed to the functional methods, which only garner 33% of the votes and produce two winning characters, and FFT1, which is even less successful with 25% of the votes and a single winning character. (See Table 1.)

Combination Method	Phase 1		Phase 2	
	Votes	Wins	Votes	Wins
CIE with FFT1	18 (36%)	3	13 (25%)	1
CIE with FFT2	17 (34%)	1	21 (42%)	4
Functional	15 (30%)	1	17 (33%)	2

Table 1. Number of votes cast and number of winning characters for each input combination method.

Time Spent Viewing

A less explicit method of evaluating users' interest in the output of the system is to track how long they spend examining that output. The web-based system used in this research facilitates this task nicely. Users are able to read about the personality traits of the characters they are voting on, look at images of those characters, and rotate 3D models of them. We track how long each user spends examining characters in this way. In order to ensure the integrity of this data, we do not inform users that their interactions with the characters are being tracked for timing purposes.

We see once again that in both phases of experiments, users spend more time examining characters produced by CIE than those produced by averaging and frequency computation in all cases but one. In Phase 1, characters

produced by FFT1 are viewed for approximately 450 seconds longer than any other characters in the tournament pool, while in Phase 2, FFT1 characters are viewed for 3000 seconds more than those produced by FFT2, which in turn are viewed 3000 seconds longer than those produced by averaging and frequency computation. (See Table 2.)

Combination Method	Phase 1	Phase 2
CIE with FFT1	2800 seconds	9423 seconds
CIE with FFT2	2318 seconds	6762 seconds
Functional	2387 seconds	3403 seconds

Table 2. Amount of time spent viewing characters generated by different experimental methods.

Phase 2 sees the emergence of what might be perceived as a dramatic positional effect, but the success of FFT2 over FFT1 in terms of voting suggests that no significant biasing occurs as a result of any positional effect.

Reloading

In the web-based system, users have the opportunity to spend as much time examining any one character as they would like, and they may choose to come back to any character as many times as they wish before casting their votes each day. This provides another implicit measure of users' interest in the characters produced both by evolution and by the averaging and frequency functions. Again, we guard the integrity of this measure by not informing participants that we are tracking the number of times they reload a character for examination. In Phase 1, FFT1 is most successful by this measure, with 94 reloads, followed by the functional combination methods with 87 reloads and FFT2 with only 80 reloads. In Phase 2, FFT1 is again the most successful with 67 reloads, compared with 50 reloads for FFT2 and only 43 reloads for averaging and frequency computation combined. (See Table 3.)

Combination Method	Phase 1	Phase 2
CIE with FFT1	94 reloads	67 reloads
CIE with FFT2	80 reloads	50 reloads
Functional	87 reloads	43 reloads

Table 3. Number of reloads by input combination method.

Collaborating

Another indication of the success of CIE is whether or not the method encourages users to work together towards a common goal, thereby minimizing conflict within the system. While this initially seemed to be beyond the scope of our system, two users at one point requested permission to contact each other and work together in attempts to gain greater influence over the system. Ultimately, the two users who conspired were not successful in dominating the tournament pool. At most, they only managed to get 75% of their genetic material into the tournament pool, and on average, four other individuals had also happened to select the same genes as the two collaborating users, so those two users were not entirely responsible for the propagation of those genes. This is a favorable scenario, as it shows that collaboration yields rewards, yet it does not allow total domination of the system, even when a high percentage of the users provide similar input.

Conclusions

We have developed a framework for collaborative interactive evolution and shown that it has potential as a tool for mitigating conflict in multi-user systems and evolving complex output in a creative domain. In experimentation, CIE performed better than frequency and averaging functions combined.

While any interpretation of why users preferred certain characters of the tournament pool over others is beyond the scope of this paper, there are a number of salient features of the GA that make it effective as a method for combining input from multiple users. We have already seen that, in domains where the fitness of a solution is a matter of great subjectivity and multiple users contribute to that complexity, fitness landscapes tend to contain multiple broad areas of optimality, rather than a single global optimum. As we have discussed, such problems are ideal candidates to benefit from interactive forms of evolution.

But the GA itself also facilitates the search of such a space. The great exploratory power of the GA, combined with its stochastic nature, tend to give rise to solutions with a broad spectrum of unpredictable characteristics. This is in comparison with the tendency of averaging to give rise to the same middle-of-the-road values when used to combine input, and the predictability of frequency functions, which may allow as few as two collaborating users to completely dominate a system.

One of the most exciting results of experimentation was the spontaneous use of collaboration among users. While a novel method for input combination should effectively resolve conflicts between users, it is also desirable to encourage users to work together, and the system proved effective in doing just that.

In this way, we propose that CIE may not only be an effective method for input combination and conflict resolution, but may actually serve as a useful educational implement, as well. This notion is not new, however. Smith (1991) commented on the potential for traditional IEC to serve as an educational tool by pointing out that, when a user is engaged by his interactive evolutionary system and sees individual biomorphs evolve and cross over with other biomorphs, "an intuitive understanding of some features of GA operation" is developed. This underlying educational aspect of interactive evolution makes IEC and CIE particularly attractive candidates for the control of interactive educational exhibits.

Future Work

We have taken care to define a general class of problems for which we expect CIE to perform well. Specifically, these are applications with complex output that draws heavily on creative and subjective human input. There is a great deal of room for CIE to be applied to additional problems of this type, with comparisons of its performance to other common methods of input combination. Appropriate applications of CIE will likely be aesthetic, creative, and generally highly interdisciplinary in nature.

We would like to see this work extended further into the domain of direct conflict resolution. The framework would benefit from a comparative study of the performance of CIE in systems that lend themselves to empirical measurement of conflict and resolution, such as the collaborative workspace environments of Single Display Groupware (Stewart *et al.* 1999).

The framework leaves room for exploration of appropriate fitness function templates, as well. While experimentation has already shown that CIE is a viable means of combining user input, a more rigorous and comparative evaluation of different FFTs is in order.

The idea has arisen of using traditional IEC to evolve one-time use fitness function templates for a group of users who will be collaborating over the course of many iterations of CIE. This approach may be used to produce, in the form of an FFT, a conflict resolution policy intrinsic to the group of users interacting with the system. At the same time, it addresses the problem of human fatigue by only engaging users in intensive subjective analysis once (during the evolution of the FFT itself), and frees them from the problem in subsequent iterations of CIE.

References

Dawkins, R. 1989. The evolution of evolvability. Artifical Life: *Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*.

Graf, J., and Banzhaf, W. 1995. Interactive evolution for simulated natural evolution. In *Artificial Evolution. European Conference* (pp. 259-272).

Herdy, M. 1997. Evolutionary optimization based on subjective selection -- evolving blends of coffee. In *Fifth European Congress on Intelligent Techniques and Soft Computing* (pp. 640-644).

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

Lim, I. S., and Thalmann, D. 2000. Solve customers' problems: interactive evolution for tinkering with computer animation. In *Proceedings of the ACM Symposium on Applied Computing* (pp. 404-407).

Louis, S. J., and Tang, R. 1999. Interactive genetic algorithms for the traveling salesman problem. In *Genetic and Evolutionary Computation Conf.* Vol. 1, pp. 1043-1048.

Sims, K. 1991. Artificial evolution for computer graphics. *Computer Graphics (Proceedings of ACM SIGGRAPH).* Vol. 25, pp. 319-328.

Smith, J. R. 1991. Designing biomorphs with an interactive genetic algorithm. In *Fourth International Conference on Genetic Algorithms* (pp. 535-538).

Stewart, J., Bederson, B., and Druin, A. 1999. Single Display Groupware: A Model for Co-present Collaboration. In *Proceedings of the CHI* (pp. 286-293).

Takagi, H. 1998. Interactive evolutionary computation: cooperation of computational intelligence and human KANSEI. In *Proceedings of the Fifth International Conference on Soft Computing* (pp. 41-50).

Takagi, H. 2001. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. In *Proceedings of the IEEE*. Vol. 89, pp. 1275-1296.