An interactive genetic algorithm with co-evolution of weights for multiobjective problems

Helio J.C. Barbosa and **André M.S. Barreto**
Laboratório Nacional de Computação Científica
Av. Getúlio Vargas 333
25651 070 Petrópolis RJ, BRAZIL
{hcbm,andremsb}@lncc.br

Abstract

An interactive co-evolutionary genetic algorithm is proposed for a class of multi-objective optimization problems and is applied to the problem of graph lay-out generation taking into account the personal user's preferences.

1 INTRODUCTION

When using evolutionary algorithms (EAs) for solving a given problem, the user has to define a fitness function which will be used to indicate the quality of a given candidate solution. This requirement can be relaxed if the user somehow is able to rank the whole population of candidate solutions according to its quality. As a minimum requirement, given two candidate solutions, the user must be able to tell which one is the best.

In a number of applications for which EAs seem to be a suitable tool, it is very difficult to construct a fitness function which accurately reflects the user's perception of what is a good solution for the problem at hand. For instance, all artistic applications of EAs fall into this category where the fitness function is subjective.

The potential of *interactive* evolution was first demonstrated by Dawkins[1] and latter developed by Sims[2, 3]. Evolution strategies with subjective selection have been developed by Herdy[4, 5]. Additionally, a number of examples can be found in [6]. For musical applications, one is referred to [7, 8, 9, 10, 11] and references therein. Some hard engineering problems have also been approached in such a way that the user is included in the loop of the evolutionary process. For instance, Gruau and Quatramaram[12] have evolved neural nets for robot control by letting the user subjectively affect the fitness value of a given solution.

It is well known to the interactive evolutionary computation (IEC) practitioners that user fatigue is a key element to the success of an IEC application and, as a result, the construction of tools which are able to learn the fitness function/user's preferences and then replace the human in the loop is an attractive idea. Using data gathered during interactive run(s) with the user, neural networks have been trained with the objective of replacing the user and allowing for longer runs[9, 11].

Yet another very important situation where user preferences come into play is the arena of multi-objective optimization (MOO) problems, where genetic algorithms (GAs) are specially attractive and a large literature is available (see the site maintained by Coello: www.lania.mx/~ccoello/EMOO/EMOObib.html). Although GAs have been successfully applied to MOO problems, specially by using its population of solutions to approximate the whole Pareto front, the task of both the GA (in finding the Pareto front) and the user (in choosing a good compromise solution) grows in complexity as the number of objectives grows. One alternative solution is to weight the several objective functions in such a proportion that the resulting scalar objective function leads the GA to eventually find a solution which satisfies the user's preferences. Of course, the task of assigning such weights is not trivial and we believe that the idea of co-evolution can help the design of an interactive GA (IGA) for this class of problem.

One particular MOO application will be considered here, namely that of graph lay-out, in order to test the proposed procedure. An interactive co-evolutionary GA is proposed which maintains two populations: a graph lay-out population and a weight-set population. Each population evolves via an independent GA but their evolution is coupled by their fitness evaluation, which involves active user intervention. For each lay-out several aesthetical objectives are mathematically defined and the final fitness value is obtained by the current set of weights. However, the population of weight-sets also evolves according to a fitness defined as how well a given set of weights ranks the population of lay-outs as compared to the ranking periodically provided by the user. It is expected that the lay-out popu-

lation improves with respect to the current weight-set and also that the weight-set population evolves generating increasingly better weight-sets, i.e., sets that better reflect the user's preferences.

The paper is organized as follows: Section 2 describes the multi-objective optimization problem and the graph lay-out problem, Section 3 briefly reviews interactive evolutionary computations and Section 4 summarizes the idea of co-evolution and some previous work. Section 5 describes the co-evolutionary GA used here and Section 6 presents some computational experiments. The paper ends with conclusions and suggestions for further work.

2 MULTI-OBJECTIVE OPTIMIZATION

In an unconstrained multi-objective optimization (MOO) problem one seeks to optimize the m components of a vector of objective functions $F(x) = (f_1(x), f_2(x), \ldots, f_m(x))$ where $x \in \Omega \subset R^n$ is the vector of design (or decision) variables which belong to the admissible set Ω. The function $F : \Omega \mapsto \Lambda$ maps solution (or design) vectors $x = (x_1, x_2, \ldots, x_n)$ to vectors $y = (y_1, y_2, \ldots, y_m)$, with $y_i = f_i(x)$, in the objective function space. Without loss of generality it is assumed that each objective is to be minimized. Due to their non-commensurable and conflicting nature there is usually not a solution which minimizes all m objectives simultaneously. This motivates the concept of *dominance*: a vector $a = (a_1, a_2, \ldots, a_n)$ dominates a vector $b = (b_1, b_2, \ldots, b_n)$ if $a_i \leq b_i \forall i$ and there is j such that $a_j < b_j$. One then defines Pareto optimality: a solution $x \in D$ is said to be Pareto-optimal in D if and only if there is no $x' \in D$ such that its image $F(x')$ dominates $F(x)$. The Pareto-optimal set, $\mathcal{P}$, is thus the set of all $x \in D$ such that $F(x)$ is non-dominated in Λ. The Pareto front $\mathcal{P}_F$ is the image of the Pareto-optimal set in the objective function space.

The first application of GAs to MOO problems dates back to the mid-eighties[13, 14] and several papers have been published since then (see [15, 16, 17, 18]). An example of a *subjective* MOO problem is presented by Shibuya et all[19] where animations of human-like motions are to be computer generated. Another example is that of graph lay-out, which is discussed in the following section.

2.1 The graph lay-out problem

The graph lay-out or graph drawing (GD) problem considered here is the task of producing aesthetically-pleasing two-dimensional pictures of undirected graphs drawn with straight edges. Vertices will be drawn as points in the plane inside a rectangular frame and the problem thus reduces to finding the coordinates of such points, since the edges are defined by a given 0/1 connection matrix.

Automatic graph lay-out is a long-studied problem in computer science with a large literature (see [20], for a bibliography). Many aesthetic criteria can be conceived of and some generally accepted ones include: (i) uniform spatial distribution of the vertices, (ii) minimum number of edge-crossings, (iii) uniform edge length, (iv) exhibit any existing symmetric feature and (v) vertices should not be placed too close to edges. Of course one would strive for an algorithm which generates a solution fulfilling all the aesthetic criteria simultaneously. However, those criteria are non-commensurable and conflicting. As an alternative to a true multi-objective GA for approximating the Pareto front in such a high-dimensional space (see Table 1) one could "scalarize" the problem by assigning weights to each of those criteria in order to obtain a single objective function. The problem is then how to assign those weights in a way that a pleasant-looking lay-out emerges. In the graph lay-out problem it is observed that different sets of weights lead to different final solutions which display different aspects/properties of the graph at hand. The final choice is thus often a matter of personal preference.

3 INTERACTIVE EVOLUTIONARY COMPUTATION

When the fitness function is subjective, an interactive GA is implemented in a way that the quality of every candidate solution has to be externally provided by the user to the GA which then performs recombination and/or mutation of individuals in the current population in order to generate the next population. The new individuals must then be presented to the user which has the task to evaluate them according to his/hers preferences.

Caldwell and Johnston[21] have developed an IGA which allows for the interactive evolution of a face of a suspect seen by the user at the scene of a crime. An IGA has been applied by Louis and Tang[22] to the traveling salesman problem (TSP). First the user visually decomposes the TSP into clusters of cities and then a GA is applied to solve the TSP corresponding to each cluster. Finally, the user either (i) manually connects sub-tours in order to get a complete solution or (ii) defines a promising region between two clusters and then let the system perform an exhaustive search among edges to delete and add in this region such that the total tour length is minimized.

It is interesting to note that when the evolved artifact is represented in two- or three-dimensional space the individuals can be *simultaneously* presented (in small numbers though) to the user which can then rank them by comparison. However, for a musical solution, say a jazz solo, which develops itself in time, the user has to hear them carefully, one at a

time, and then proceed to the ranking stage.

It has been observed that the user's task often leads to fatigue and that an IGA should be carefully implemented if it is to be useful. One possible way to remove the human being from the IGA loop and thus speed up the evolutionary process is to substitute it by a neural network trained to reflect the user's preferences. That has been tried by Biles[9] in order to simplify his original procedure which consisted in presenting each solution (a jazz solo) to an audience for evaluation. Unfortunely good results were not obtained indicating that this option is also hard. In the work by Johanson and Poli[11] an interactive system allows users to evolve short musical sequences using interactive GP. As the user is the bottleneck in the process, the system takes rating data from a user's run and uses it to train a neural network based automatic rater which can then replace the user and allow for longer runs. The best pieces generated by the automatic raters were reported to be pleasant but were not, in general, as nice as those generated in user interactive runs.

An approach to the interactive development of programs for image enhancement using GP is presented by Poli and Cagnoni[23]. There is no fitness function and the user decides which individual should be the winner in the tournament selection. Good solutions to real-life problems are reported after only hundreds of evaluations. Also, a strategy to reduce user interaction is proposed: the choices made by the user in interactive runs are recorded and later used to build a model which can replace the user in longer runs.

There are other situations of scientific and technological importance where, although a reasonable fitness function can be designed, the evolutionary process would be improved if any additional user knowledge (perhaps hard to code into the fitness function) could be used in the evaluation or ranking of the population. In this case, one would profit from a *supervised* process where the user monitors the evolution and occasionally intervenes in order to guide the process towards what is perceived as a better region of the search space at least according to the user experience. An example of such approach has been presented recently by Boscheti[24], where IGAs are proposed for a class of geophysical inversion problems. For additional IEC applications and a survey, the reader is referred to Takagi[25, 26] and references therein.

As already mentioned, an important area where user preferences are essential is that of MOO problems, and we believe that the idea of co-evolution can bring some needed help to the design of an IGA for this class of problem.

4 CO-EVOLUTION

In a standard GA the fitness function does not change with time so that the population can be thought of as climbing a fitness landscape and gradually converging to one of its peaks. In contrast to these static landscapes, natural evolution happens in a dynamic fitness landscape where organisms are constantly adapting to each other and to their environment.

Artificial co-evolutionary algorithms have been used in the solution of practical problems and one idea is that of *cooperation*. Potter and De Jong[27] used it for function optimization in R^n where n subpopulations are maintained and the fitness of a given individual of a particular subpopulation is an estimate of how well it cooperates with other subpopulations to produce good solutions. These ideas were later applied to the co-evolution of neural networks[28] and sets of rules[29].

Another approach is that of *competition*. In the pioneering work of Hillis[30] it was applied to the problem of evolving minimal sorting networks for lists of a given number of elements. It was observed that the co-evolution of test cases along with the sorting networks results in an improved procedure. Two independent populations are maintained: one of sorting networks (or hosts) and the other of test cases (parasites). The fitness of each sorting network was measured by its ability to correctly solve test cases while the fitness of each test case was proportional to the number of times it was incorrectly sorted by the networks. Both populations evolved simultaneously, interacting through the fitness function.

Paredis[31] used a co-evolutionary approach to improve the genetic evolution of neural networks. Again two populations were maintained: one of neural networks and another of examples of the classification task which is submitted to the neural networks. The fitness of a neural network is defined as the number of successful classifications of the twenty most recently encountered examples. On the other hand, the fitness of an example is the number of times it was incorrectly classified by the neural networks it encountered most recently. However, the example population consists all the time of the same 200 pre-classified examples. To evolve decision trees, Siegel[32] also performed competitive co-evolution with fixed training examples.

Rosin and Belew[33] used competitive co-evolution for game-learning problems in which the fitness of an individual in a host population is based on a direct competition with individual(s) from a parasite population.

Inspired by the work of Hillis[30], a co-evolutionary GA has been proposed by Barbosa[34] in which two evolving populations are used to solve min-max problems. Several successful small scale applications were reported in [34]. Later[35] that approach was applied to a class of optimization problems stated as an adversary game between two players ("nature" and the designer), as well as to con-

strained optimization problems[36].

Competitive co-evolution was also used by Sebald[37] in the min-max design of neural net controllers for uncertain plants while Husbands[38] used a cooperative/competitive multi-population distributed co-evolutionary GA to handle a generalized version of job shop scheduling.

In this paper an interactive, co-evolutionary GA is proposed for MOO problems where the user's preferences are subjective, such as in the graph lay-out problem considered here. However, the procedure can not be classified neither as a competitive nor as a cooperative co-evolutionary GA.

5 THE CO-EVOLUTIONARY GA

In the co-evolutionary GA proposed here, two populations will be maintained: a graph lay-out population and a weight-set population. The lay-out population is composed of individuals that contain the coordinates of all vertices in the graph while the weights population is composed of individuals that contain, each one, a set of weights. An independent GA is applied to each population and the coupling of the evolutionary processes is made through the fitness evaluation. Each one of the metrics/objectives can be exactly computed for a given lay-out but the fitness of this lay-out depends on how – i.e. with which set of weights – those objectives are linearly combined.

A fitness function must also be defined for the weight population. Here it will be defined as follows. After the lay-out population evolves for a given number of generations (with its fitness being computed according to a fixed set of weights) a sample of lay-outs from the current population is displayed for user inspection. The user then ranks them according to his/hers personal preferences, which may be different from their current ranking in the population.

Now it is time for the weight population to evolve while the current lay-out population is kept "frozen". The fitness function value of a given set of weights is then computed as follows. Each individual (set of weights) evaluates the sample of lay-outs and produces its own ranking. The fitness of a given set of weights is higher, the closer its ranking approaches the ranking provided by the user. A set of weights that ranks the sample in a very different order from that chosen by the user is not very useful and has thus a low fitness.

The weight population is now evolved in order to search for that set of weights which produces a ranking the most closely resembling the one provided by the user. During a fixed number of generations, the lay-out population is kept "frozen".

After a (hopefully) better set of weights is found, the lay-out population is then allowed to evolve with its fitness evaluation being now performed using the newly found (best) set of weights.

The process is then repeated for a given number of cycles until a satisfactory graph lay-out emerges.

As a result, the fitness of an element in the lay-out population depends on the current set of weights, which depends on the evolution of the weight population whose fitness, in turn, depend on how well the current set of weights reflects the user's personal preferences concerning the graph lay-outs. In other words, the evolution of the lay-out population happens in a fitness landscape that changes every time a new set of weights is presented by the weights population. On the other hand, the fitness landscape of the weights population changes every time *the user* presents its ranking of a sample of the current lay-out population. Figure 1 displays the pseudo-code for the algorithm.

In order to minimize fatigue, the user only points and clicks the mouse on *some* of the displayed lay-outs following a decreasing order of his/her preference. The remaining lay-outs are ranked after those picked by the user and maintain their relative order.

```
Algorithm
    Initialize lay-out population randomly
    Initialize weight-set population randomly
    Compute each criterion for all graph lay-outs
    while not(user satisfied) do
        Display a sample from the lay-out population
        The user ranks the sample
        for i = 1, 2, ..., max_gen_w do
            Evaluate population of weights
            Generate new population of weights
        Pick the best set of weights
        for j = 1, 2, ..., max_gen_l do
            Compute each criterion for all graph lay-outs
            Evaluate population of lay-outs
            Generate new population of lay-outs
end
```

Figure 1: Pseudo-code for the algorithm.

Details of the GA used for each population are given next.

5.1 The genetic algorithm for the weights population

As suggested by an anonymous reviewer, a system of linear inequalities could be written defining the set of all feasible sets of weights, that is, weights that reproduce a given user's ranking. However, it cannot be proven that such a set is always non-empty: there is a human making choices. As a result, a generational GA with a rank-based selection scheme was adopted for the evolution of the weights population, where each chromosome is simply a vector of posi-

tive real numbers (the weights) whose sum is 1.

The fitness of a set of weights is evaluated by its capacity to rank the lay-out population in accordance with the user's preferences. The process can be explained as follows. When the user inputs his/her own rank, it can be seen as a vector U describing a specific ordering. The fitness of a set of weights S can be calculated by:

$$f(S) = \sum_{j=1}^{m} C_j(W_j - U_j)$$

where W is a vector like U, but describing the ordering generated by the individual S. C is a fixed vector of constants which allows one to emphasize the importance of the first individuals. In our tests, $m = 9$ and the elements of C are $\{9, 8, 7, 6, 5, 4, 3, 2, 1\}$.

The crossover operator used was the blend crossover - BLX[39] which generates one offspring by randomly generating for each weight a value in the range $[w_i^s - \delta, w_i^l + \delta]$ with $\delta = \alpha(w_i^l - w_i^s)$ and where w_i^s and w_i^l are the smallest and largest values respectively for w_i in the parents' chromosomes (α was set to 0.3).

Two mutation schemes are used, which will be referred to as weak perturbation mutation - WP and strong perturbation mutation - SP. First of all, a value is randomly chosen from the interval [0,1]. If it is lower than 0.85, mutation will occur. Next, the WP mutation will be applied with a probability of 0.7 and the SP will occur with probability of 0.3 (1.0 - 0.7). The mutation operators work as follows: (i) WP: One weight w_i of the set is randomly chosen and perturbed by an amount limited to the interval $[-0.3w_i, 0.3w_i]$; (ii) SP: One gene (a weight) is randomly chosen and has its value changed to another value randomly picked in the interval [0,1].

5.2 The genetic algorithm for the lay-out population

The GA used here is a standard generational one where each candidate solution is encoded as a real vector containing the coordinates (x_i, y_i) of each vertex of the graph. The initial population is randomly generated and the selection process is rank-based.

The GA for graph lay-out tries to optimize for several aesthetic criteria. One of them is the energy function defined by Kamada and Kawai[40]

$$\mathcal{E} = \sum_{i=1}^{|V|-1} \sum_{j=i+1}^{|V|} \frac{1}{2} k_{ij}(|p_i - p_j| - l_{ij})^2 \qquad (1)$$

where $|V|$ is the number of vertices, p_i is the position vector of vertex i, and k_{ij} and l_{ij} are respectively the spring force and the ideal distance between vertices i and j. The idea is to make the Euclidean distance between two vertices as close as possible to the ideal "graph theoretic" distance, which is defined as $l_{ij} = Ld_{ij}$ where d_{ij} is the distance between vertices i and j measured as the length of the shortest path between them and L is the ideal length of an edge in the graph, given by $L = L_0/\max\{d_{ij}\}$ where L_0 is the length of the side of the square display area. Finally, the spring constant in Eq. (1) is given by $k_{ij} = K/d_{ij}^2$ where K is a constant. Due to the observation that this energy function, which often leads to uniform edge length and uniform vertex distribution, is not always able to achieve a good lay-out, several other criteria have also been introduced. The objective function is thus composed by a weighted sum of those criteria and the energy function $\mathcal{E}$. All criteria adopted are listed in Table 1 with the complexity associated with their computation, where $|E|$ denotes the number of edges in the graph.

Only mutation operators are used in this GA. However, the approach adopted here is different from that of the standard GA; here, each *individual* has a probability of 0.85 to be submitted to mutation. The first mutation operator is the single vertex mutation – SV, which perturbs the coordinates of a randomly chosen vertex by an amount not larger than L. The second one is the exchange vertex mutation – EV, which exchanges two randomly pre-selected vertices of the same graph. Finally, a third one, called vertex replacement mutation - VR was also defined. Just like the SP mutation of the GA for the weight population, it simply replaces one gene for another value randomly chosen but feasible, i.e., the (x, y) coordinates of a vertex are replaced by a pair of values in the interval $[0, L_0]$. The relative probability of these 3 operators was set as 0.5, 0.2 and 0.3, respectively. One should note that both GA's have to adapt to – potentially drastic – landscape changes, so it seems a good idea to have a high mutation probability.

The reason for not using a crossover operator is the fact that the schemes tested not only did not bring any improvement but also made the process slower. This seems to be due to the difficulty known as the *competing conventions problem* in the area of evolutionary neural networks. In our case, it happens when two (or more) identical layouts of the same graph are either shifted, rotated or inverted. It is desirable for a crossover operator that, when combining two equivalent layouts, the offspring generated be similar to its parents. We intend to develop such an operator, using ideas from [41].

6 NUMERICAL EXPERIMENTS

In order to illustrate the feasibility and performance of the approach proposed here, the task of laying out a simple planar graph was undertaken.

Criterion	Abb.	Description	Complexity
Energy	En	Potential energy $\mathcal{E}$	$\Theta(\|V\|^2)$ [a]
Edge Crossings	Ec	Number of intersections between edges	$\Theta(\|E\|^2)$
Edge Deviation	Ed	Difference between edge lengths	$\Theta(\|E\|^2)$
Edge attraction	Ea	Distance between connected vertices	$\Theta(\|E\|)$
Vertices Repulsion	Vr	Inverse of distance between vertices	$\Theta(\|V\|^2)$
Centripetal Attraction	Ca	Distance between vertices and the centroid of lay-out	$\Theta(\|V\|)$
Central Uniformity	Cu	Difference between vertices distances to the centroid of lay-out	$\Theta(\|V\|^2)$
Vertex-edge Distance	VEd	Inverse of distance between vertices and edges	$\Theta(\|V\|\|E\|)$

[a] Assuming that the d_{ij} are previously calculated and stored.

Table 1: Aesthetics criteria adopted in the algorithm

Figure 2 shows 9 elements from the initial population which were presented to the user.

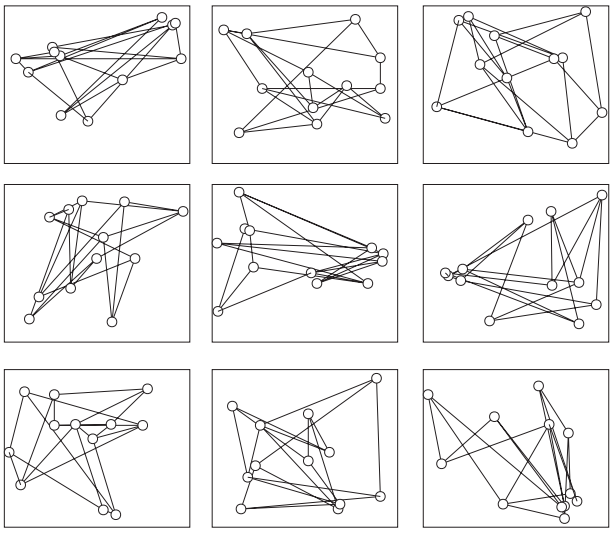

Figure 2: A sample from the initial population.

The first run was made by a user who did not care about the number of edge-crossings in the drawing.

The weights associated with the criteria used in this case, En, Ed, Ec, Ca and Vr then evolved respectively to 0.5394, 0.2854, 0.06554, 0.050776 and 0.05887 and a solution with many edge crossings was obtained but with an interesting spatial structure displayed as shown in Figure 3.

The second run was made by an user who wanted a solution with a minimum of edge-crossings. As expected, the weights evolved to quite different values: 0.0008176, 0, 0.6478, 0.02042 and 0.3310. It is easy to see that the (third) weight associated with edge-crossings now has evolved to a value ten times higher than that of the previous case and the one associated with the energy is almost $1/1000$ of the previous solution, indicating that this criteria goes "against" a planar solution in this problem. The graph obtained has no edge-crossings and is displayed in Figure 4.

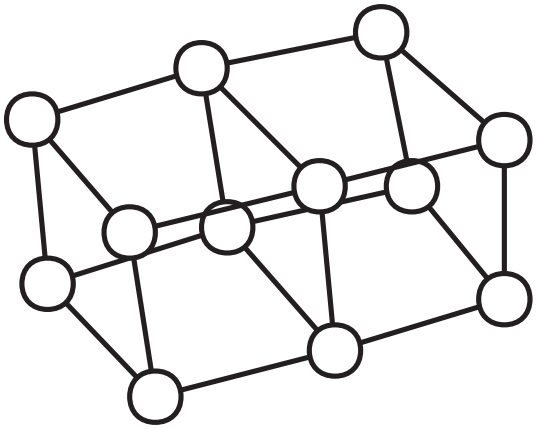

Figure 3: Solution for user 1.

7 CONCLUSIONS

The work presented here reports on an investigation being conducted concerning the design of an interactive co-evolutionary genetic algorithm for multi-objective optimization problems and its application to the graph lay-out problem. Comparing with previous interactive evolutionary algorithms, the proposed interactive co-evolutionary GA can use a larger population for the application at hand since the user only has to rank a sample from the corresponding population. A single paradigm – the genetic algorithm – is used for both tasks, namely, that of searching for a good solution of the multi-objective optimization problem (a graph lay-out, in the present application) and that of learning the user's subjective preferences.

Acknowledgements

This paper was written while the first author was visiting the Colorado State Artificial Intelligence Laboratory. The hospitality provided by Prof. Darrel Whitley is gratefully acknowledged. The authors would also like to thank the reviewers for comments and suggestions which helped improve the quality of the paper.

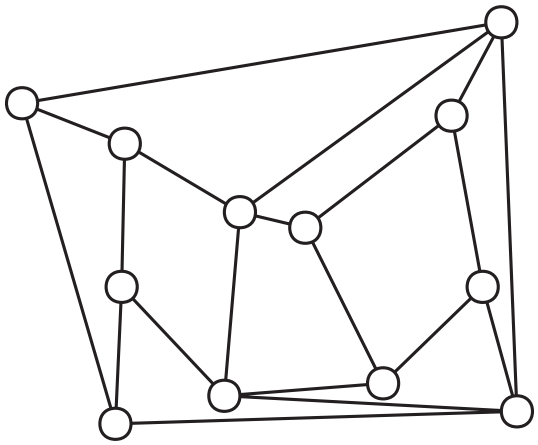

Figure 4: Solution for user 2.

References

[1] R. Dawkins. *The Blind Watchmaker*. Longman, Essex, 1986.

[2] K. Sims. Artificial evolution for computer graphics. *Computer Graphics*, 25(4):319–328, July 1991.

[3] K. Sims. Evolving 3D morphology and behaviour by competition. In R. A. Brooks and P. Maes, editors, *Artificial Life IV*, pages 29–39. MIT Press Cambridge, MA, 1994.

[4] M. Herdy. Evolution strategies with subjective selection. In W. Ebeling, H.-P. Schwefel, and H.-M. Voigt, editors, *Proc. of the Parallel Problem Solving from Nature - PPSN IV*, pages 22–31, Berlin: Springer, 1996.

[5] M. Herdy. Evolutionary optimization based on subjective selection - evolving blends of coffee. In *Proc. of the 5th European Congress on Intelligent Techniques and Soft Computing, EUFIT '97*, pages 640–644, 1997.

[6] P.J. Bentley, editor. *Evolutionary Design by Computers*. Academic Press Ltd., London, 1999.

[7] A. Horner and D. Goldberg. Genetic algorithms and computer-assisted music composition. In Richard K. Belew and Lashon B. Booker, editors, *Proc. of the Fourth International Conference on Genetic Algorithms and their Applications*. Morgan Kaufmann, San Mateo, CA, 1991.

[8] J.A. Biles. GenJam: A genetic algorithm for generating Jazz solos. In *International Computer Music Conference*, San Francisco, CA, 1994.

[9] J.A. Biles. Neural network fitness functions for a musical IGA. In *Soft Computing Conference*, 1996.

[10] B.L. Jacob. Composing with genetic algorithms. In *International Computer Music Conference*, San Francisco, CA, 1995.

[11] B.E. Johanson and R. Poli. GP-music: An interactive genetic programming system for music generation with automated fitness raters. Technical Report CSRP-98-13, University of Birmingham, School of Computer Science, May 1998.

[12] F. Gruau and K. Quatramaran. Cellular encoding for interactive evolutionary robotics. Technical Report Cognitive Science Research Paper CSRP425, School of Cognitive and Computing Sciences, University of Sussex, Brighton BN1 9QH, England, UK, 1996.

[13] J.D. Schaffer. *Some Experiments in Machine Learning using Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN, 1984.

[14] J.D. Schaffer. "Multiple objective optimization with vector evaluated Genetic Algorithms". In *Proc. of the 1st Int. Conf. on Genetic Algorithms*, pages pp. 93–100, 1985.

[15] C.M. Fonseca and P.J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, Spring 1995.

[16] Jeffrey Horn. *Multicriterion Decision Making*, volume 1, pages F1.9:1 – F1.9:15. IOP Publishing Ltd. and Oxford University Press, 1997.

[17] D.A. Van Veldhuizen and G.B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1998.

[18] C.A.C. Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, 1999.

[19] M. Shibuya, H. Kita, and S. Kobayashi. Integration of multi-objective and interactive genetic algorithms and its application to animation design. In *Proc. of IEEE Systems, Man, and Cybernetics*, volume III, pages 646–651, 1999.

[20] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. Annotated bibliography on graph drawing algorithms. *Computational Geometry: Theory and Applications*, 4(5):235–282, 1994.

[21] C. Caldwell and V.S. Johnston. Tracking a criminal suspect through face-space with a genetic algorithm. In Richard K. Belew and Lashon B. Booker, editors, *Proc. of the Fourth International Conference on Genetic Algorithms and their Applications*. Morgan Kaufmann, San Mateo, CA, 1991.

[22] S.J. Louis and R. Tang. Interactive genetic algorithms for the traveling salesman problem. In *Proc. of the Genetic and Evolutionary Computation Conference*, 1999.

[23] R. Poli and S. Cagnoni. Evolution of pseudo-colouring algorithms for image enhancement with interactive genetic programming. Technical Report CSRP-97-5, School of Computer Science, The University of Birmingham, B15 2TT, UK, January 1997.

[24] F. Boschetti and L. Moresi. Comparison between interactive (subjective) and traditional (numerical) inversion by genetic algorithms. In *2000 Congress on Evolutionary Computation*, pages 522–528, San Diego, CA, USA, July 2000. IEEE Service Center.

[25] H. Takagi. Interactive evolutionary computation: System optimization based on human subjective evaluation. In *IEEE Int'l Conf. on Intelligent Engineering Systems (INES'98)*, pages 1–6, Vienna, Austria, Sept. 1998.

[26] H. Takagi. Interactive evolutionary computation - Cooperation of computational intelligence and human *kansei*. In *5th Int'l Conf. on Soft Computing (IIZUKA'98)*, pages 41–50. World Scientific, Iizuka, Fukuoka, Japan, Oct. 1998.

[27] M.A. Potter and K.A. De Jong. A cooperative co-evolutionary approach to function optimization. In *Third Parallel Problem Solving from Nature*, Jerusalem, Israel, 1994.

[28] M.A. Potter and K.A. De Jong. Evolving neural networks with collaborative species. In *Proc. of the 1995 Summer Computer Simulation Conference*, Ottawa, Ontario, Canada, 24–26 July 1995.

[29] K.A. De Jong and M.A. Potter. Evolving complex structures via cooperative coevolution. In *Fourth Annual Conference on Evolutionary Computation*, San Diego, CA, 1–3 March 1995.

[30] W.D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D*, 42:228–234, 1990.

[31] J. Paredis. Steps towards co-evolutionary classification neural networks. In R. Brooks and P. Maes, editors, *Artificial Life IV*. MIT Press/Bradford Books, 1994.

[32] E.V. Siegel. Competitively evolving decision trees against fixed training cases for natural language processing. In K. Kinnear, editor, *Advances in Genetic Programming*. MIT Press Cambridge, MA, 1994.

[33] C.D. Rosin and R.K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In L.J. Eshelman, editor, *Proc. of the Sixth International Conference on Genetic Algorithms and their Applications*, Pittsburgh, PA, July 1995.

[34] H.J.C. Barbosa. A genetic algorithm for min-max problems. In E.D. Goodman, V.L. Uskov, and W.F. Punch III, editors, *EvCA'96 - First Int. Conf. on Evolutionary Computation and its Applications*, pages 99–109, Moscow, 24–27 June 1996. Institute of High Performance Computer Systems of the Russian Academy of Sciences.

[35] H.J.C. Barbosa. A coevolutionary genetic algorithm for a game approach to structural optimization. In Th. Baeck, editor, *Proc. of the Seventh International Conference on Genetic Algorithms and their Applications*, pages 545–552, East Lansing, MI, July 1997.

[36] H.J.C. Barbosa. A coevolutionary genetic algorithm for constrained optimization problems. In *Proc. of the 1999 Congress on Evolutionary Computation*, pages 1605–1611, Washington, DC, USA,, July 1999. IEEE Service Center.

[37] A.V. Sebald and J. Schlenzig. Minimax design of neural net controllers for highly uncertain plants. *IEEE Transactions on Neural Networks*, 5(1):73–82, 1994.

[38] P. Husbands. Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In T. Fogarty, editor, *Evolutionary Computing, Lecture Notes in Computer Science*, volume 865, pages 150–165. Springer-Verlag, 1994.

[39] L.J. Eshelman and J.D. Schaffer. Real coded genetic algorithms and interval schemata. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 187–202. Morgan Kaufmann, San Mateo, CA, 1993.

[40] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, April 1989.

[41] Jürgen Branke, Frank Bucher, and Hartmut Schmeck. A genetic algorithm for drawing undirected graphs. In *Proc. of the Third Nordic Workshop on Genetic Algorithms and their Applications (2NWGA)*, pages 193–206, 1997.