Hybrid Real-Coded Mutation for Genetic Algorithms Applied to Graph Layouts

Dana Vrajitoru
ISL, Computer and Informations Sciences
Indiana University South Bend
danav@cs.iusb.edu

Jason DeBoni
ISL, Computer and Informations Sciences
Indiana University South Bend
wanderung@yahoo.com

ABSTRACT

In this paper we introduce an application of real-coded genetic algorithms to the problem of consistent graph layout and exploring the role of mutation for this particular problem. We introduce several forms of mutation, some of which being specific to this problem, and show that the choice of this operator can have a great impact on the performance of the algorithm.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms

Keywords

real-coded mutation, graph theory

1. INTRODUCTION

The mutation has always played an important role for the genetic algorithms and its role is of clear interest. The mutation rate is usually kept low, a generally accepted heuristic being one over the chromosome length [5].

In the natural living systems, we can speculate that the rapid evolution of some organisms may be in part attributed to mutations which are not totally random, but favor changes that increase the fitness of the organism. In this paper we present a system in which the mutation can take various forms, some of which are based on heuristics using domain-specific knowledge that are likely to improve the individual's fitness. We are interested in the impact of the various mutation forms on the achieved fitness.

The problem we focus on is building consistent graph layouts for weighted graphs and requires a real gene encoding. Given a weighted graph, we must derive a spatial representation of the graph (a layout) such that the distances between the vertices are consistent with the weight of the edges. Extensive work has been accomplished on drawing unweighted graphs [4]. The best-known heuristic for generating graph layouts is certainly the spring algorithm [2].

We found that a real-encoded genetic representation is appropriate for the graph layout problem and this also gives us more flexibility regarding the mutation operator. Several real-encoded models for genetic algorithms have been proposed [3]. Our focus is on the mutation operator under these conditions.

2. FORCE BASED ALGORITHMS

Given an undirected and weighted graph, we must assign a geometric point to each of the vertices in the graph (a layout) such that for every edge, the distance between the vertices is equal to the weight of the edge [6].

Let us consider the error on an edge to be the difference between the weight of the edge and the Euclidean distance between the two points. If the error is positive, then the points are too close to each other. If the error is negative, the points are too far apart. We would like to find a layout that minimizes the total absolute error in the graph.

In the *breadth-first order* algorithm, each vertex is attracted or repelled by its neighbors according to the error on the edge. Non-adjacent vertices do not interact. The algorithm starts with a random layout that is adjusted in a number of iteration to obtain one that is consistent. At each iteration, the algorithm moves one vertex at a time on one of the outgoing edges, further away from the other vertex on the edge if the distance is smaller that the weight of the edge, and closer to the reference point if the distance between them is greater than the weight of the edge. A parameter ε controls the amount of the change.

An iteration of the *breadth-first order (BF)* algorithm starts with a randomly chosen vertex (origin), and it adjust all the other vertices in the graph starting from this origin with a breadth-first scanning method. By starting from a different origin at every iteration, we insure that the layout will not prematurely converge to a suboptimal configuration.

Let us suppose that we can construct a physical representation of the graph using interconnecting springs for the edges. Each spring corresponding to an edge has an initial length equal to the weight of the edge. When extended, the springs tend to contract to their initial length, and when compressed, they tend to extend. We can build a random graph layout using springs, and then let it evolve to an equilibrium state.

The *tension vector (TV)* algorithm starts with a random

GECCO'05, June 25–29, 2005, Washington, DC, USA.
ACM 1-59593-010-8/05/0006.

layout and readjusts it in several iterations to achieve an equilibrium. At each iteration, it computes the resulting tension forces in each vertex based on the current layout, then moves all of the points in the direction of the forces, again based on a configurable parameter ε.

3. REAL-CODED MUTATION

To apply the GAs to to our problem, we need to represent a graph layout as a chromosome. We consider that each of the layout points is composed of three genes taking real values. The genes are initially generated in a given boundary which is a 3D box of dimensions depending on the weights in the graph. The fitness function is based on the total error in the graph. We have chosen the uniform crossover with a swap probability of 0.45.

The genetic representation of a problem with real-coded genes offers more possibilities for defining the mutation. Several forms have been proposed in the literature, and some of our operators are inspired from them.

The *uniform* mutation replaces the value of the chosen gene with a uniform random value within the range specified by the user [3], eventually following a Gaussian distribution [1].

The *mirror* mutation replaces a gene with it's mirror value with respect to the middle point of the boundary interval for the gene. This is the closest operator to the binary bit-flipping usual mutation.

The *percentage* mutation replaces a gene with a random percentage of its value within the interval $[80\%, 120\%]$.

These three types of mutation are not specific to our problem. The following two forms of mutation represent a combination of the heuristics presented in Section 2.

The *edge* mutation is moving the 3 genes of a randomly selected point on a randomly selected edge starting from that vertex to reduce the error on that edge, as in the BF force-based method.

The *tension vector* mutation (TVM) is moving the 3 genes of a randomly selected point based on the tension vector resulting from all edges starting from that vertex as in the tension vector method.

4. EXPERIMENTAL RESULTS

We have conducted our experiences with 7 graphs with a number of vertices between 50 and 200 with existing solution. Table 1 shows the results of the five variations of the mutation operator on the ten graphs. The first column represents the number of vertices in the graph. All of these results represent an average over 50 trials with 1000 generations and a population of size 50. The ϵ parameter is equal to 0.005 for these results. The last two columns in this table represent the force-based algorithms.

We can notice that there is a considerable difference in the performance between the forms of mutation we have introduced. The percentage mutation is doing visibly better than both the mirror and uniform mutations, and the edge mutation is a further clear improvement. Last, the tension vector mutation is substantially better than even the edge mutation.

Both force-based methods converged to solutions very close to an exact one and they are still better than the use of genetic algorithms. Even so, the genetic algorithm cannot compete with the force-based method yet, the fitness function would allow us to embed more constraints into the derived layouts, like aesthetic qualities, which are harder to achieve by the traditional methods.

Table 1: Total error as percentage of the total weight in 1000 generations / iterations

Size	Uni	Mir	Per	Edge	TVM	BF	TV
50	93.6	94.5	68.1	60.4	21.0	0.01	0.02
70	94.1	94.7	74.0	65.0	08.8	9e-4	1e-3
100	94.0	94.5	75.0	64.2	11.2	5e-3	0.03
125	94.2	94.7	76.6	64.2	13.6	9e-7	0.01
150	94.2	94.7	79.1	66.9	14.2	1e-6	0.01
175	94.3	94.7	79.5	65.6	14.6	9e-7	0.02
200	94.3	94.7	81.1	67.1	22.9	1e-6	0.01

5. CONCLUSIONS

In this paper we have introduced a real-coded genetic algorithm applied to the consistent graph layout problem. The focus of this study has been the mutation operator, for which the real encoding of the chromosomes provides more choice than in the binary case.

The experiments have shown that the choice of the mutation operator can have a large impact on a real-encoded genetic algorithm. Thus, the uniform and mirror mutation show a similar performance, the percentage mutation constitutes an improvement over them, and the two problem-specific mutations have shown a substantially better performance than the general purpose operators.

6. ACKNOWLEDGMENTS

This work has been in part conducted under the IUSB Faculty Fellowship Grant, 2004.

7. REFERENCES

[1] L. Davis. Adapting operator probabilities in genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, San Mateo, CA, 1989. Morgan Kaufmann Publishes.

[2] P. Eades and D. Kelly. Heuristics for reducing crossings in 2-layered networks. *Ars Combin.*, 21.A:89–98, 1986.

[3] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading (MA), 1989.

[4] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. An Alan R. Apt Book. Prentice Hall, Upper Saddle River, NJ, 1999.

[5] G. Ochoa. Setting the mutation rate: Scope and limitations of the 1/l heuristic. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 495–502. Morgan Kaufmann Publishers, 2002.

[6] D. Vrajitoru and J. DeBoni. Consistent graph layout for weighted graphs. In *The 3rd ACS/IEEE International Conference on Computer Systems and Applications*, Cairo, Egypt, 2005.