Experiments with Genetic Algorithms for Displaying Graphs

A. Márkus

Computer and Automation Institute, Hungarian Academy of Sciences
1502 Budapest, POB 63, Hungary

Abstract

The paper first outlines a set of propositions on displaying graphs of labeled and weighted nodes and arcs. These graphs have neither a natural spatial mapping, nor special structure. After a look-out to the centering and local modification techniques, the core of the paper discusses the use of genetic algorithms: the combination of imperfect layouts by simulating natural selection processes. Finally, further display methods are proposed: a rule-based approach, rendering on a globe and tessellation.

1. Introduction

This paper does not try to push the limits of the representation of conceptual structures, it is accepted as a de facto standard that conceptual structures are represented as a set of atomic entities, and relations restricted to binary ones, either symmetric or not. Instances of the relations are not considered as hypermodes, i.e. no relations are allowed between them. Instances of the nodes and relations have number(s) attached, referring to weight(s) (e.g. of the importance of the node, or the similarity of the concepts etc.). Accordingly, in simplest practical terms, our central aim is to display undirected graphs, with no restriction of their being trees, or planar ones, or of a limited size etc. As for a recent survey of related work we refer to [4].

Experiments reported here deal with cases where the very nature of the data suggests no distinguished, natural mapping of the whole structure to 2D or 3D space. Moreover, we do not suppose that there exists a natural partitioning of the task so one can not synthesize good partial solutions into a good total solution: accordingly, the divide-and-conquer approach of [7] is not viable here.

Structures considered here are static: as soon as we have a feasible layout of a large structure, small, step by step modifications could better be made interactively until the need of some major reorganization arises.

The efforts reported here belong to the preliminary phase of a larger project that aims at developing a pattern matching machinery of conceptual structures. Our opinion, supported by programmers' practice in various kinds of animated program tracing is that half of the battle is over as soon as we find a nice picture of the structures and their dynamic behavior. Accordingly, if we would like to match patterns, first we have to browse the maze of their interconnections.

2. Preliminaries

Our experiments with the rendering of conceptual structures have suggested the following pieces of advice that largely determine the methods applicable here:

2.1. Remain in 2D

Having made lengthy but fruitless experiments with using the third dimension in rendering, our very basic supposition is that—at least on a black and white display—it is better to refrain from the third dimension for rendering these abstract structures. The reason is that we have found no way of showing the depth of the elements definitely enough: it is additional information that may truly resolve the ambiguity when we see 2D or 3D objects. Empirical evidence shows that, as for adult persons, it is next to impossible to find our way in 3D seen on the screen of computers: while it is easy to program a turtle with the usual commands, a corresponding 3D creature that changes the directions in space can hardly be understood. So we are afraid that one could be lost not only in hyperspace but in 3D as well (notwithstanding sceneries furnished with everyday objects). Most probably, what we can do in 3D has to be connected somehow to manipulating with our hands, but as creatures we mostly exist on the surface of the Earth. So it may be more than a present-day technical limit that it seems to be worth preferring pure 2D for the rendering of abstract, conceptual structures.

2.2. Do not show too much regularities

The display of the graphs must not suggest more relations than truly present in the data. This requirement has turned out to be a rather strong one: there are as many (mis)interpretations as people seeing such pictures. Although we claim that there are patterns more easily
recognizable than others (and that this issue is not as simple as saying that people prefer symmetries and fractions close to the golden cut), cultural and personal imprints may cause large differences of what one prefers to find as a pattern within an abstract picture. Moreover, overly ordered pictures are less challenge for the human interpreter in finding the hidden features of the data.

In order to avoid unintended regularities the most obvious choice is to use an abundance of random elements in the algorithms. Although by preferring randomness one may not aim at best solutions, on the other hand, a variety of unlike, suboptimal solutions may suggest further ideas concerning deterministic or interactive final phases.

2.3. Have a pragmatic notion of clarity

A central issue of these algorithms is that we should find a realistic criterion of a picture being well-balanced and clear. Moreover, this criterion has to be effectively calculated: recomputation should be proportional to the change of the picture. Accordingly, a very simple measure of clarity has been chosen: each intersection of the arcs of the graph counts as an add-on term varied by the weights of the nodes and/or arcs. (By the way, the near to uniform density distribution of the points has been provided by the algorithms without further ado.)

As for other, more sophisticated measures (e.g. a small angle of intersecting arcs is more disturbing than one around 60 degrees; or an intersection near to the nodes is worse than one near to the middle of the arc), our present standpoint is that it is hopeless to incorporate all these features into a single objective function and too much conflicting terms might lead to an evaluation too flat.

A further requirement is that the rendering algorithms should work with limited resources: after presenting preliminary results and allowing the user to make minor adjustments, the algorithm should be able to run further.

2.4. Let annotation come last

The above criterion of clarity of pictures does not deal with additional elements, e.g. with labels attached to the nodes/arcs. Our opinion is that this belongs to a separate phase: having created a picture clear as it can be, the annotations can be assigned to their proper places within a second round. The labeling task is far easier, since the conflicts here are more local and the changes rarely have far-reaching effects: repositioning a colliding label may cause further collisions etc., but these chains of effects are not long.

3. Methods
3.1. Centering of the nodes

Centering is the far most obvious way of generating a clear layout of these graphs: in order to produce a well-comprehensible picture, each node should be positioned in the middle of the nodes that are related to it. Unfortunately, the result of this algorithm largely depends on the initial configuration and on the order of letting the nodes move to the centre. In order to avoid cumulative bias, successive runs of the centering algorithms should use new random permutations of the nodes. Since centering tends not only to shrink the graph smaller and smaller but to make it collapse into a oblique line, rescaling and an appropriate rotation should be applied between centering cycles. Further improvement can be achieved by readjusting the positions of the nodes so that they be positioned onto a rectangular grid in such a way that the ordering of the nodes be kept intact when projected onto the axes.

Centering could be considered as a process of energy minimization: if nodes are zero mass joints, links are springs, central positions of the nodes correspond to a state of minimal energy within the corresponding mechanical system. Using this metaphor, the question emerges whether a more elaborate analogy could help in finding better solutions: having added an inertial component to the joints, could this result in better convergence?

The main advantages of centering is that it is very effective in the first phase of smoothing out the maze of interconnections. Disadvantages are that the results depend on the original, random layout, and the process may be stuck to local optima. Moreover, centering alone is not efficient at disentangling quasi independent parts of the structures.

3.2. Local modifications driven by simulated annealing

Simulated annealing [5] is a random local optimization technique that uses a thermodynamic analogy of cooling physical structures: good moves of the particles are accepted any time, the odds of expecting bad moves change with the value of an exponential function depending on the value of the step (this is the analog of changing energy) and a control parameter (i.e. this is the analog of temperature). In qualitative terms, while simulated time flies, temperature gets lower and smaller bad moves are likely to be accepted.

Defining the energy of our pictures is straightforward: it should be a monotone function of the clarity of the picture. The effectiveness of the strategy depends on two factors: first on the annealing schedule, second on the definition of the actual moves. Here we do not want to go into details, our only claim is that the above strategy can be used as a driver of a domain-dependent local optimization process, e.g. of centering steps as discussed before. Similar efforts have been reported in [1].

3.3. The use of genetic algorithms

The central idea of genetic algorithms (GAs) [3] is that, unlike other optimization techniques that focus on a single candidate solution, an ever-changing population of
individual solutions is to be operated on. New generations of the population are created from the previous ones by using an artificial genetic machinery that borrows more or less from the features of the genetic mechanism of Nature. The aim of using the genetic machinery is to produce a sequence of populations that tends to grow better as time flies. Generations contain individuals that have been produced as most valuable ones in the past; at the same time, individuals should be able to drive the search for even better solutions in the future.

The key concepts of GAs are genetic material and operations. Genetic material is a representation of the individuals’ relevant characteristics. Genetic operations create the genetic material of new individuals, getting that of either from one ancestor (mutation) or from two parents (crossover). Genetic material and operations are so closely related to each other that requirements can be directed only towards the genetic machinery as a whole. Some of the most important requirements towards this machinery, grouped into conflicting pairs, are as follows:

1. Be able to create all those individuals that can be described by the given representation, but do not create many individuals that, after some further processing, turn out to be inviable.
2. Pass a large portion of the relevant characteristics to the offspring, i.e. fitness of the offspring be related to the parents’ fitness, but if parents are dissimilar, offspring be produced with a larger variability.
3. Support the survival of advantageous structures stored in the genetic material, but allow the free combination of any segments of genetic material.
4. Specify a mutation rate that helps to avoid premature uniformity of the population, but do not allow mutation to oppress the inheritance of useful substructures.
5. Start the genetic machinery with a random population, but prepare the initial population in such a way that it may incorporate domain specific knowledge.

The cast of our GA application is as follows:

- Individuals are layouts, fitness of the individual is the clarity of the picture. Initial population is generated randomly.
- Genetic material is the picture itself, i.e. it contains the exact actual positions of the nodes. (Although we have made experiments with more restricted genetic material, so that to allow more place for adaptation of the individuals—i.e. for forming phenotypes of the same genotype—the simplicity of this genetic material is more efficient.)
- For the crossover operation, there are two competing alternatives:

1. Inherit each point independently from each other: This crossover destroys the coupled inheritance of connected structures. On the other hand, this crossover is more “creative”, especially when the individuals' adaptation process is allowed to have enough time to produce its results.
2. Inherit groups of connected points: This approach is much closer, although not identical, to the usual setting of genetic algorithms that use chromosomes: this process randomly picks up two points, one for a “maternal center” and another for a “paternal center”.

Then, following the links of the graph, maternal vs. paternal areas are ground around the centers until the areas come into conflict with each other.

There is not much difference in the early generations if made by this or that crossover operation. But later on, as the structure gets discovered and disentangled, the maternal and paternal regions of the layout tend to be isolated, and one sees a remarkable stability the good regions. Actually, there is a simple message of the two crossover operators: as soon as there have been found some chunks of information worth passing to the offsprings together, prefer to pass them as a chunk.

New generations have been created with using a sampling rate that gives larger chance of mating individuals with higher fitness. Mutation, the other genetic operation, has been random repositioning or centering of some points of the picture. Mutation rate has been tuned so that if the population seems to turn too homogeneous, the mutation rate of better individuals is increased to produce a larger diversity of the population.

It may be worth mentioning that neither crossover nor mutation alone has been able to drive the evolution process quickly enough, but if working together, departing from a randomly generated initial population, the genetic process converged into a population which could not further be improved by mutation or crossover. Although final populations consisted of individuals of near to equal fitness, there were not at all uniform with respect of their structure.

### 3.4. Sample results

Methods discussed in the previous points have been integrated into a prototype system that consists of the following modules: (1) A small graph-editor with the usual interactive facilities (e.g. setting, moving, deleting and dragging points, setting and removing arcs). (2) An automatic graph display module that implements (2a) centering, (2b) random local modification driven by simulated annealing and (2c) the GA driven refinement of the pictures.

Having started with random layouts, the examples on Fig. 1-2. are about concepts related to inflation and the humour of renaissance Italy. Fig. 3. shows a map of wandering over nodes in a hypertext object. As for different kinds of browsing strategies, among them "wandering", see [6]. The layouts have been produced by the GA incorporating centering and random modification as its mutation element.
Fig. 1. Concepts related to inflation.

Fig. 2. Concepts related to the humour of renaissance Italy.

Fig. 3. Wandering in hyperspace.
For all the above examples, the genetic algorithm outperformed stand-alone centering and random changes. A convincing proof of this could be given by watching the changes of the structures on the screen: the most fertile individuals are those where the crossover operation finds and keeps such smaller or bigger substructures of the graph that have already been laid out better than others. (This is in obvious contrast with the centering methods where the level of granularity is one point only.) The key factor of the power of GAs is just this: they are able to find components, even if a formal (still less a productive) definition of the components is not known.

4. Further experiments with rendering conceptual structures
4.1. Tessellation

Having got a clear layout of the graph, further interpretation of the underlying conceptual structure could be supported by discarding the links that have driven the process up to this point. Allowing that the nodes have weights, a kind of Voronoi diagrams could be used for this purpose.

Supposing that point A has weight $w_A$, B has $w_B$, those points $P$ should make the border of the regions where:

$$w_B \cdot \text{distance}(A,P) = w_A \cdot \text{distance}(B,P).$$

Accordingly, border points of the regions produce circular arcs if the weights are different and line segments if the weights are equal. Pictures got in this way have high explanatory power, because both quantitative and qualitative changes of the structure correspond to well comprehensible changes in the topology of the picture.

4.2. Rendering on a globe

A flat piece of drawing is a really adequate medium for displaying graphs if and only if there are nodes in central and others in peripheral roles. If many nodes may be central then the surface of the globe, since it has no boundaries, is a more appropriate medium (this idea is closely related to the fisheye views, see [2]). Taking this approach, nodes are points on the surface of a globe, links are arcs of great circles. Regions of the surface should be projected to the screen by well-known geographic mappings.

Layout on the globe has many advantages: (1) it is suitable for richly connected structures, since subsets of the links may be directed towards the hemisphere not seen, and links that cross the contour circle may be omitted in order to increase the clarity of the picture; (2) it avoids the unintentional effects of something being on the upper part of the screen, or near to the left border (psychology claims that these positions are suggesting more importance than those close to the right margin).

The disadvantage of this approach is the large computational demand of the basic operations (e.g. drawing the perspective image of a circular arc) which may be prohibitive in computing environments not dedicated to these purposes.

4.3. The reasoning alternative

This approach starts with an initial layout, generated by an arbitrary method. Let agents be implemented that see this layout and search for specific configurations they are
able to repair (e.g., the task of an agent may be that if two points are too close to each other, shift away the one which is more loosely connected to other parts of the picture). Let the agents bid for the elements of the picture (i.e., both for the atomic ones and, maybe for complex structures as well). The amount of the bid be derived by the agent using its own judgment of the situation (patterns may be more or less fitting to the agent's task). Let a control program allow the agent with the highest bid to change the picture.

The above activities are similar to the recognize-act cycle of a data-driven, rule-based system. The only problem here is that there may be too many alternatives and one hardly knows beforehand what kind of rules will be useful and how can their activities be kept within manageable limits (e.g., there may be too many competing bids, cyclic modifications etc.). The above scheme can be made far more effective with implementing a learning component, i.e., the agents should receive payments for their involvement in producing the final picture. Agents that have set the stage in an earlier phase could be honored by having the payments propagated back; see the bucket brigade algorithm of classifier systems [8].

5. Conclusion

In order to produce clear layouts of graphs we need more efficient rendering strategies that can be incorporated as modules into visual programming systems. The metaphor of genetic algorithms, even if it may be a relatively weak method for well-defined problems, offers a practical method of dealing with this problems. Other methods (local modifications, centering) can be integrated as useful modules within this framework.

Acknowledgment

Many thanks for Judit Boros, art historian, for the useful remarks not influenced by temptations and prohibitions of present day computing practice. Three anonymous referees have provided valuable comments and suggestions.

References


