The Application of Machine Learning to Problems in Graph Drawing A Literature Review

Raissa dos Santos Vieira
Hugo Alexandre Dantas do Nascimento
Wanderson Barcelos da Silva

Institute of Informatics
Federal University of Goiás
Goiânia - GO, Brazil
Email: {raissavieira,hadn,wandersonsilva}@inf.ufg.br

***Abstract*—Graph drawing, as a research field, is concerned with the visualization of information modeled in the form of graphs. The present paper is a literature review that identifies the state-of-the-art in applying machine learning techniques to problems in graph drawing. We focused on machine learning strategies that build up and represent knowledge about how to draw a graph. Surprisingly, only a few pieces of research can be found about this subject. We classified them in two main groups: the ones that extract knowledge from the user by human-computer interaction and those that are not based on data directly gathered from users. The study of these methods shows that there is still much to research and to develop regarding the application of machine learning to graph drawing. We suggest directions for future research on this area.**

Keywords–Graph Drawing; Human-Computer Interaction; Machine Learning.

I. INTRODUCTION

Graphs are mathematical models defined as a set of vertices and a set of edges. They are widely used to represent physical and abstract entities and their relationships. Often, it is necessary to draw a graph, that is, to construct a geometric representation of its vertices and edges [1]. For this aim, it is common to choose a standard graph-related convention (for example, drawing vertices as circles and edges as straight lines) and a set of aesthetic criteria (such as displaying edges with uniform orientation, minimizing edge crossings and presenting symmetry).

When a graph contains only a few vertices and edges, it can easily be drawn manually. However, as the size of the graph increases, manual drawing becomes more difficult and time consuming. The most common strategy for drawing medium to large-size graphs – ranging from hundreds to thousands of vertices and edges – is through the usage of automated techniques that incorporate a set of aesthetic criteria and apply algorithms for finding aesthetically pleasing drawings. A number of computational systems for drawing graphs exist based on this approach. Including, we have GraphViz [2] and Gephi [3].

Drawing a graph by a computational process, on the other hand, also creates many difficulties. One of them is that the search for drawings of good quality drawings with several aesthetic criteria is an NP-Hard problem [1]. In addition, some aesthetic criteria are in conflict, so that the improvement of a drawing in respect to one criterion may imply a reduction of other aesthetical aspects. Furthermore, the drawing of a graph is essentially a subjective task – some users may prefer to satisfy some particular aesthetic criteria, different from the preferences of other users.

For this reason, even with the use of heuristics, there is still a need for human intervention to assist in obtaining good quality graph drawings. This was perceived very early in the advent of the graph drawing research field, resulting in the inclusion of human-computer interactive resources in most graph drawing systems. Such resources help to tailor the drawing towards satisfying aesthetic criteria that are not fully treated computationally, and to escape from local minima, when the graph drawing process involves an optimization model.

In the current paper, we present a literature review of the use of machine learning techniques for graph drawing. This is a much more complex challenge than merely having a fully automatic graph drawing system or a system with a few simple interactive tools. We shall comment on computational approaches that attempt to acquire knowledge that can be used to help drawing graphs. As can be seen, there are few reports of research on this subject. However, some interesting ones have been found and their study may lead to promising lines of future research.

The remainder of the paper is organized as follows: in Section II, graph drawing definitions are presented; Section III provides an overview of the application of machine learning techniques to graph drawing; Section IV summarizes the characteristics of the existing approaches; finally, in Section V, we draw our conclusions and suggest future works.

II. GRAPH DRAWING

A *finite undirected graph* G is an ordered pair (V, E) of finite sets V and E, where V is a set of *vertices* representing a set of discrete objects, and E is a set of unordered pairs $\{x, y\}$ of distinct elements in V, termed *edges* between vertices x and y. In a *directed graph*, E is a set of ordered pairs of vertices such that an edge $e = (x, y)$ connects x to y, but the reverse is not true, unless there is an edge $e' = (y, x)$ in E. A *two-dimensional drawing* of a graph $G = (V, E)$ is a function that associates each vertex and edge of G with geomtric objects in a drawing space.

 ISBN: 978-1-61208-386-5

The graph drawing process begins with a computational representation of a graph (e.g., an adjacency matrix or a vertex-edge incidence list) and the selection of a *graphical convention*. Vertices are usually depicted as circles or squares while the edges are commonly represented by polygonal lines or arcs. Then, a set of aesthetic criteria are chosen in order to determine the aesthetic quality of any drawing of the graph. Aesthetic criteria, frequently employed as soft constraints, are:

- Minimizing the number of edge crossings;
- Displaying symmetries;
- Distributing the vertices evenly in the drawing area;
- Showing all edges with uniform length; and
- Arranging the edges in the same direction as much as possible (in the case of directed graphs).

In addition to the aesthetic criteria, soft constraints, hard "drawing constraints" can be imposed. While an aesthetic criterion may be neglected to a certain degree, all the hard drawing constraints must always be satisfied. Examples of drawing constraints are:

- Avoiding vertex over placement and
- Requiring some vertices and edges to be located in given fixed positions.

In the graph drawing research area, the drawing problem is studied according to the class of graphs considered, e.g. undirected, directed, planar or tree graphs. For each particular class, there are graph conventions, aesthetic criteria and hard constraints, all of relative importance.

A drawing of a given graph that reflects these details can be attempted. With a large drawing area, usually different drawings can be generated for non-trivial graphs. Not all of these drawings are of practical interest. The most desirable ones are those that best satisfy the aesthetic criteria, since they have a higher chance of being readable, that is, helping understand the inherent graph structure.

One of the difficulties of obtaining useful drawings, however, is that finding the best drawing for many aesthetic criteria is a NP-Hard problem [1]. Also, some aesthetic criteria conflict with the other. In this case, there may be no drawing that optimally satisfies two or more adopted aesthetic criteria. In such situations, there must be a trade-off between the criteria.

The definition of what is a "good" drawing can also be a very challenging task. The opinions of which aesthetic criteria to adopt may vary significantly between users. Sometimes, a common set of aesthetic criteria is desirable by all users, but they vary in their relative importance. For instance, one user may prefer edge crossing minimization over drawing symmetry, while another user may prefer the opposite. Many computational approaches to evaluate the quality of a graph drawing employ a weighted function of the given aesthetic criteria with weights provided in advance by the user.

In some situations, the user's preferences are imprecise and difficult to model mathematically. In this case, the users have to draw the graph manually, possibly in an incremental way until an acceptable quality of the drawing is reached.

Thus, we can conclude that human interaction is necessary in many current graph drawing systems because these systems do not automatically and adequately deal with all drawing issues that may be of concern to the users.

In a previous work, Nascimento and Eades [4] investigated interactive optimization systems for graph drawing. They also proposed a framework that combines graph drawing algorithms and human interaction in an optimization process. The main result of their study was that the combination of graph drawing algorithms with human interventions help to achieve better results than having the algorithms and the users working independently. Furthermore, the authors identified certain interactive actions already being performed by users that could be used as the basis for new graph drawing algorithms. They then suggested applying machine learning techniques to learn and automatize such actions.

The idea of using machine learning to graph drawing goes far beyond the traditional interactive graph-drawing approaches, as it tries to computationally empower existing algorithms while releasing users to carry out more suitable roles. The following section presents a review on existing approaches that, in some ways, pursue this idea.

III. Using Machine Learning for Graph Drawing

We performed a search of the relevant scientific literature in order to identify existing graph drawing methods that use machine learning techniques. The main literature databases investigated were: the bibliographic database of the Institute of Electrical and Electronics Engineers (IEEE), the database of the Association for Computing Machinery (ACM), Scopus, CiteSeer, Science Direct and the Web of Science. Our search string combined the terms: Graph Drawing, Graph Layout, Machine Learning, Expert Systems, Task Learning, Case-Based, Knowledge-Based, User Preferences and Artificial Intelligence.

The search identified 86 papers. Four other references that we previously knew of were added. We then analyzed all these papers and reduced the bibliography to only 11, all of which actually use machine learning techniques to help to produce good drawings of graphs. This final set of papers was divided into two main groups, presented next:

1) *Approaches that learn from human interaction* – these are approaches that learn information from users based on their interactions to a graph drawing system. They are presented in Section III-A;
2) *Approaches that are not based on human interaction* – we consider here approaches that gather and evolve knowledge about how to draw a graph from the results of other automatic graph drawing algorithms or from the graph structure itself. These approaches are presented in Section III-B.

We describe other details about the approaches that allows a more refined classification, later in the paper. Some aspects that we analyze include: the goal of the learning process (defining the quality of the drawing, or improving the convergence of an optimization process towards a good drawing), the class of graphs being drawn, the type of information interactively provided by the user for the learning process, the learning method, and whether or not the acquired knowledge can be reused to draw other graphs.

A. Approaches based on human-computer interaction

Since the choice of aesthetic criteria and their importance is an inherently subjective task, Neto [5] and Neto and Eades [6]

 ISBN: 978-1-61208-386-5

developed an approach for learning users' preferences. They were in the first to propose the automatic learning of human knowledge for drawing graphs. For this, they created an interactive system in which the parameters of an objective function (inferring the desirable aesthetics) and the setup parameters of a simulated annealing graph drawing method are automatically adjusted. As shown in Figure 1, the system consists of: a graph drawing editor, for human operators to draw the graph according to their preferences; a learning module that, implicitly, observes the users' actions and tries to learn and reproduce their results; and a knowledge base consisting of weights for a set of predefined aesthetic criteria (these weights appear in the objective function) and some parameters for initializing the simulated annealing process. Using the editor, the user provides a drawing of the graph. While doing this, the system performs two tasks autonomously: (1) it measures the aesthetic criteria of the user-generated drawing (2) and it runs a process for automatically learning all parameters necessary for reproducing the drawing. The learning process is also based on a simulated annealing strategy. The knowledge acquired by the system may help to recreate a drawing for other very similar graphs. However, the authors mentioned that it is not clear whether the aesthetic parameters can be meaningful to draw other graphs.

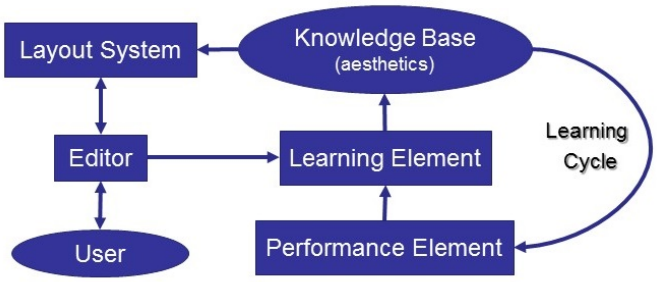

Figure 1. Learning model (adapted from [5]).

Some other approaches for the learning of aesthetic criteria are based on evolutionary computation techniques, such as genetic algorithms. This has been done by Masui [7], Rosete-Suárez *et al.* [8], Barbosa and Barreto [9], Bach *et al.* [10] and Spönemann [11]. In general, these authors present systems that collect information from human interaction that helps to clarify and to adjust a fitness function, as illustrated in Figure 2. Their goal is to refine the definition of the graph drawing problem (in the sense of identifying what kind of drawing addresses the subjective aesthetic criteria).

Following this line of thought, Masui [7] proposed having the user evaluating drawing examples, which are rated as "good" or "bad". This evaluation is done in a preliminary stage. Figure 3 illustrates some examples given by the user to the system. Such pieces of information then serve as a reference for the algorithm to infer the desirable aesthetic criteria. The approach was developed only for directed graphs. After setting the fitness function based on examples given by the user, an evolutionary process is run in order to produce drawings of these graphs.

Rosete-Suárez *et al.* [8] built a system that also acquires user preferences, but from the manual scoring of drawing

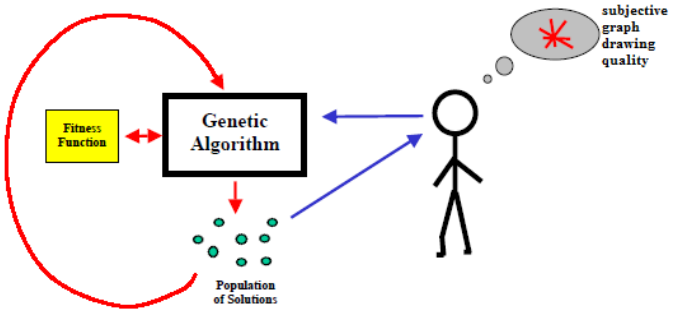

Figure 2. Interactive evolutionary computation: the user evaluates graph drawing solutions and helps to refine a fitness function (adapted from [4]).

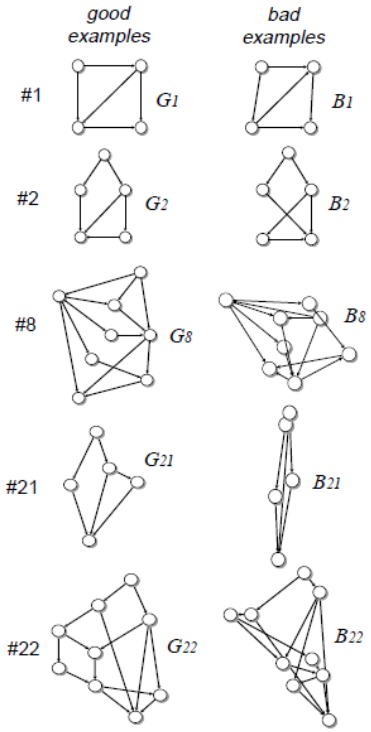

Figure 3. Good and bad example layout pairs given to the system reported in [7].

solutions in the population of a particular genetic algorithm. They worked with undirected graphs. When running the genetic algorithm, the system asks the user to give scores for six graph drawings from the current population. The learning of weights for a fitness function occurs from such an evaluation. The fitness function is adjusted with these weights to reflect the importance to the user of each aesthetic criterion. The interactive process also tries to improves the convergence of the genetic algorithm, since solutions with good aesthetic quality evolved more quickly than in some other evolutionary approaches without human interaction.

Likewise, Barbosa and Barreto [9] employed user evaluation to undirected graph drawings. However, they used a process called co-evolution, in which the set of weights of a fitness function is evolved indirectly using information from both the user and an internal evolutionary algorithm. Furthermore, the

 ISBN: 978-1-61208-386-5

user ranks the solutions in the current population instead of giving full scores. Figure 4 shows some drawings that were presented by their system to the user for evaluation.

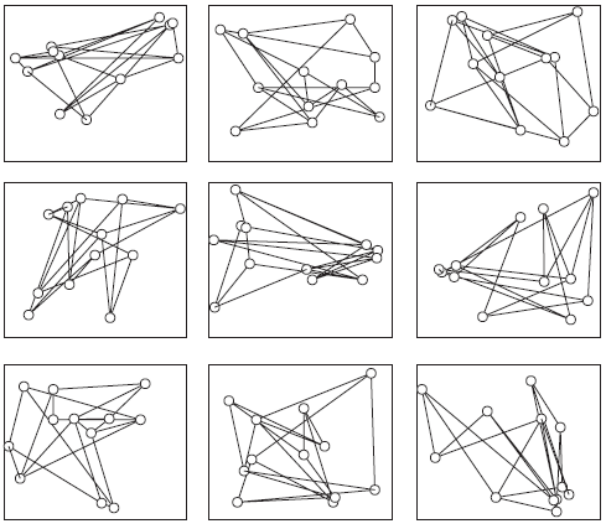

Figure 4. A sample from the initial population (from [9]).

Bach *et al.* [10] presented a system called "GraphCuisine", a random drawing generator for undirected graphs with a populational viewpoint. A random initial population is generated and an evolutionary algorithm improves the quality of this population. Then the user selects solutions they deem to be the best by visual inspection. When multiple graphs are selected, the system tries to infer what graph characteristics are important to the user and adjusts the fitness function to represent them. This interactive approach aims to refine the problem of how to produce good graph drawings and to improve the convergence of the genetic algorithm. The system interface to select the best drawings is shown in Figure 5. Note that there are options for the users to modify the generation interval in which they want to view the population, the population size and how many drawings should be presented for selection.

Lastly, Spönemann [11] makes use of two interactive human inputs with an evolutionary approach: the users can directly manipulate the weights of a fitness function by changing visual sliders, or they can select good drawings from the current population. Both actions result in the automatic adjustment of the weights in the objective function, representing aesthetic criteria. Figure 6 presents the user interface showing drawings for individuals of the current population. There is a checkbox below each graph drawing, which enables users to select the drawing for the purpose of the adaption of the weights. The system works with both undirected and directed graphs.

An advantage of using genetic algorithms for interactive optimization is that it naturally produces several alternative solutions for user evaluation and it is flexible in dealing with different aesthetics criteria and constraints.

B. Approaches not based on human-computer interaction

Other approaches exist that use well-known machine learning techniques for drawing graphs, but that do not take into consideration data provided by human operators.

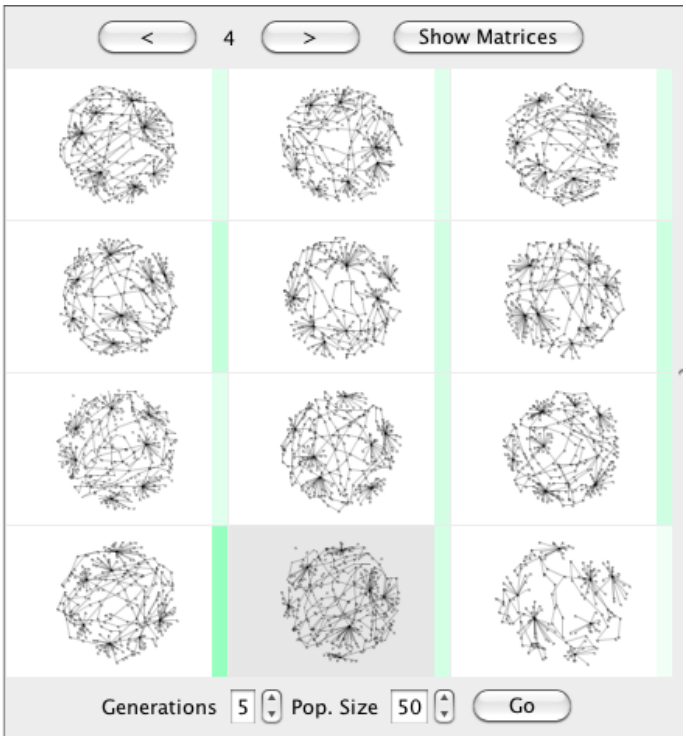

Figure 5. The population view, displaying representative graphs of the current population (adapted from [12]).

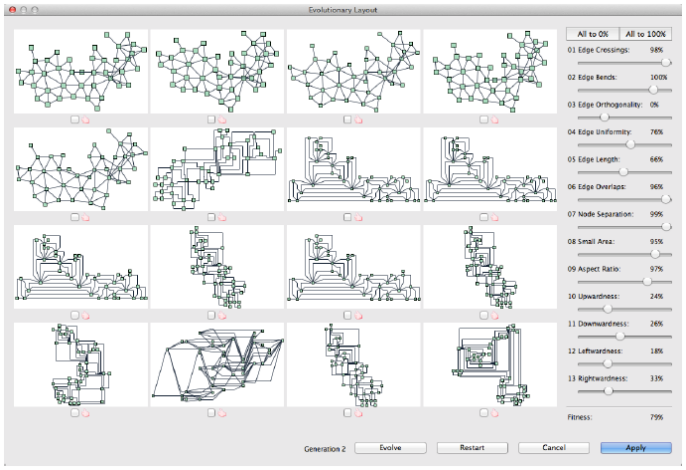

Figure 6. User interface for evolutionary meta layout (from [11]).

We start mentioning three reports of research that are based on neural networks.

Cimikowski and Shope [13] presented a parallel neural network algorithm that minimizes edge crossings in drawings of nonplanar graphs. They based their work on that of Takefuji and Lee [14] for computing a maximal graph planarization. The use of a neural-network representation for solving optimization problems was proposed by Hopfield and Tank [15]. After them, several developments followed.

Cimikowski and Shope dealt with graph drawings in the form of arc diagrams (or linear embeddings), as illustrated in Figure 7 for K_6, the complete graph on six vertices. In

their graphical convention, the vertices are placed along a horizontal line and the edges are drawn as semicircles in one of the two half planes bounded by the line. Their optimization approach consists of modeling each edge of the graph as two neurons, representing the possibility of the edge being drawn above or below the line. Every neuron can be set to state 0 or 1, and this indicates distinct configurations for the position of the edges (above or below the line, not placed at all or in an inconsistent condition). An energy system is then created, containing excitatory and inhibitory forces that push the neurons to a state that minimizes the number of edge crossings while ensuring that all edges are placed. Through an iterative process, the system converges from a randomly-defined initial state to a stable configuration with the minimal number of edge crossings. In terms of the quality of the final results and processing time, the approach proved to be more competitive than a conventional greedy algorithm for the same problem.

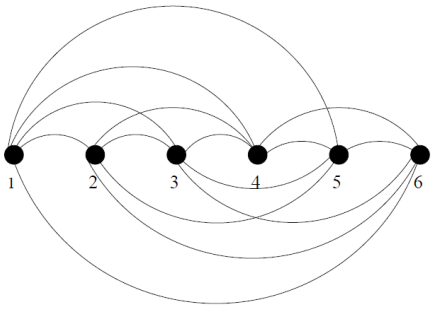

Figure 7. Linear embeddings of the complete graph K_6 with 3 crossings (from [13]).

Wang and Okazaki [16] improved the method proposed by Cimikowski and Shope [13]. They modified the internal dynamics of the neural network to permit temporary increases in the energy function in order to help the network to escape from local minima.

Finally, Meyer [17] presented an approached for graph drawing based on a competitive learning algorithm and on an extension of ideas used in unsupervised neural networks and self-organized maps. An association between a self-organized network and a graph drawing problem was established. The graph to be drawn is viewed as a neural network to be trained – every vertex of the graph is a neuron in the network; the position of a vertex v in the drawing is, in fact, described by the weight vector of the connections of v to other neurons. Drawing a graph is then treated as the problem of training its related network, that is, finding the proper sets of weights for the neurons. The training method consists of a competitive learning algorithm that updates the weights associated to a winner neuron v and to its neighbors so that they get closer to a random stimulus (to a point randomly chosen on a mesh over the drawing area). Ideally, the winner is the neuron whose weights are the closest to the stimulus. This process is iterated with new winners, with the changes being less significant at each time. The final result of the algorithm is a set of weights (coordinates) that visually highlights the structure of the graph. Figure 8 shows two drawings of K_6, obtained using a triangular mesh and a rectangular mesh for the random stimuli. The approach is also able to draw graphs in a three-dimensional space and to deal with some spatial aesthetics criteria.

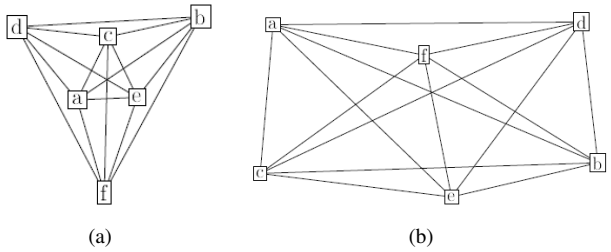

Figure 8. The complete graph K_6: (a) triangular layout, (b) rectangular layout (from [17]).

Despite neural networks and self-organized maps being very catchy names, we must point that these three related approaches deviate from the machine learning nature in which we are interested. They really do not build a general neural network that learns aesthetic criteria or hints about how to draw a graph. What their methods do is to model the graph structure and the drawing problem as a network coupled with an energy system. The methods then improve the state of the network in order to minimize the system energy, usually resulting in a good quality graph drawing. Unfortunately, there is not resultant learned knowledge that can be reused, and every new graph has to be drawn by repeating the whole optimization process from scratch.

A different and more interesting learning approach is the one proposed by Stolfi *et al.* [18], using asynchronous teams (A-teams) for drawing graphs. By experimenting with teams of graph drawing heuristics, they verified that some sequences of heuristics (alternating over a drawing of a graph) could produce better results than other sequences. They then termed such good sequences as "pedigrees" and built a system for finding and using them. The system operates in two modes: "playing" and "working". In the playing mode, graph drawing heuristics (coded as agents in an A-team) are applied at random in order to evolve a set of drawings of a given graph. A history describing what heuristic was used in sequence is kept for every drawing. When the A-team stops, the history for the best generated drawing is the pedigree that we are looking for. Next, in the working mode, the system produces drawings for new graphs by applying only the sequence of heuristics described by the pedigree.

The approach was tested using both undirected and directed graphs. The playing mode produces a high quality drawing, but is very timing consuming because it usually creates many intermediate solutions that have bad quality and are thus discarded. So its main advantage is to produce a pedigree. The working mode, on the other hand, does not create a perfect drawing, but is able to generate a reasonably good solution relatively quickly. The authors suggest that a pedigree could be applied as a recipe to draw similar graphs, but this was not fully tested.

At last, Niggemann and Stain [19] presented an approach for learning the best method for drawing clustered graphs. With this aim in mind, the authors used a set of graphs and a set of popular graph drawing methods. They applied clustering

 ISBN: 978-1-61208-386-5

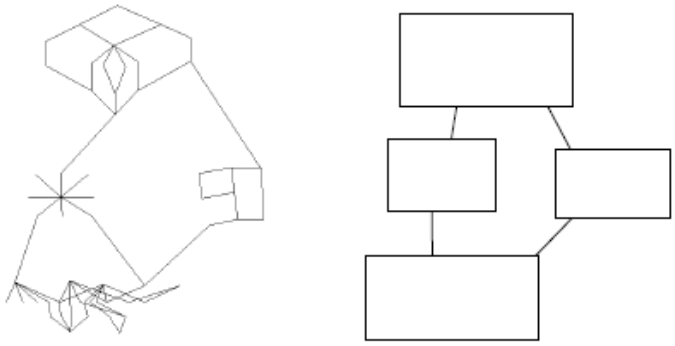

Figure 9. A graph, on the left, and the abstract view its clusters, on the right (from [19]).

algorithms to the graphs and, for each resultant subgraph (cluster), all drawing methods in the set were tested. Next, a knowledge base was created containing the characteristics extracted from all the subgraphs created and the method that generated the best drawing for them. After the learning step, the knowledge base could be used to draw any new graph. This involves partitioning the new graph into clusters, extracting their characteristics and generating a drawing for every cluster using the best method indicated by the knowledge base. The final step is to integrate the drawing in order to create a drawing of the whole graph. Figure 9 illustrates an input graph (the left image) and how it was clustered into four subgraphs (represented by rectangles on the right). Each subgraph has its own characteristics and is drawn by a particular method according to the knowledge base.

IV. CHARACTERISTICS OF THE APPROACHES

Table I summarizes eight aspects of the approaches described in the previous sections:

- *Year of publication*.
- *Class of graphs* – the type of graph that can be drawn by the approach. We use the letter "D" to represent directed graphs and "U" for undirected graphs.
- *Goal of the approach* – the existing approaches have two general goals: to *refine the graph drawing problem* (that is, to learn the user's preferences of drawing aesthetics and thus to provide a drawing that the user may like); and to *improve the convergence* of a graph drawing algorithm towards a high quality solution (in general, by helping to escape from local minima, focusing the algorithm on more important subproblems, tuning the algorithm's parameters and providing shortcuts).
- *Knowledge source* – this indicates the source from which the approach obtains information for the learning process. We consider here the main classification followed throughout the paper, regarding whether or not to use information coming from *human interaction*. However, we detail the second case by giving two options that better describe the non-interactive types of source: the *graph structure* alone or data collected from the usage of other *graph drawing methods*.
- *Type of interaction* – when the knowledge source is "human interaction", then this column specifies the form of interaction that can be performed by the user to provide the system with information. The types of human interaction that we found in the reviewed literature are: evaluating graph drawings (assigning scores or ranking the drawings), selecting some graph drawings from a list, performing manual adjustments (weights of aesthetic criteria or the drawing itself), and providing graph drawing examples with good or bad aesthetics.
- *Learning method* – this column indicates the learning method used in the approach. The following methods appeared in the reviewed literature: neural networks or related algorithms, case-based knowledge representation, and the adjustment of a graph-drawing evaluation function (that measures the quality of a graph drawing) used as a fitness function in an evolutionary graph drawing method.
- *Reusability* - this means that the approach acquires general knowledge that can be applied to the drawing of other graphs.

TABLE I. CHARACTERISTICS OF THE LEARNING APPROACHES.

			Goal		*Knowledge source*			*Type of interaction*				*Learning method*			
Paper	***Year of publication***	***Class of graphs***	*Refining the problem*	*Improving convergence*	*Data structure*	*Drawing method*	*Human interaction*	*Evaluation*	*Selection*	*Manual adjustment*	*Drawing examples*	*Neural network*	*Case-based know.*	*Fitness evolution*	***Reusability***
[5]	1994	U/D	X				X			X	X		X		X
[7]	1994	D	X				X				X			X	
[13]	1996	U		X	X							X			
[17]	1998	U		X	X							X			
[8]	1999	U	X	X			X	X						X	
[18]	1999	U/D		X		X							X		X
[19]	2000	U/D		X		X							X		X
[9]	2001	U	X				X	X						X	
[16]	2005	U		X	X							X			
[10]	2013	U	X	X			X		X					X	
[11]	2014	U/D	X				X		X	X				X	

From the table, we can see that machine learning techniques have not yet been fully explored to solve problems in graph drawing. There is a major concentration of approaches that use human-computer interaction to adjust a fitness function in an evolutionary graph drawing algorithm. However, reusability of the knowledge learned by the approach is still a weakness. Even when the knowledge is reusable, there is no guarantee that the results will be satisfactory. For example, the learning obtained in [18] can be reused for a new graph drawing task, but the quality of the drawings may not be as good as those obtained by drawing the graph via a complete process (in playing mode). Another aspect of the literature review is that we could not find publications about this theme between the years 2006 and 2012.

V. CONCLUSION AND FUTURE WORK

As far as we know, this is the first literature review to identify the state-of-the-art in the use of machine learning techniques to problems in graph drawing. As we could see from the review, only a few studies investigated such topic.

Considering that the graph drawing research area is well established, with an annual international symposium since 1994, this suggests that there is still much to investigate when the aim is to learn knowledge about how to draw a graph computationally.

The characteristics of the existing approaches presented in Table I can be useful for identifying possible research topics. For instance, case-based knowledge representation has been employed only marginally. There is also a lack of an effective method based on a general neural network for learning about how to draw graphs. Furthermore, the reusability of some described methods should be more deeply experimented with and evaluated. Finally, there are some questions on the current topic that still pose significant challenges. For example, "What other types of knowledge can be collected from a user in order to improve a graph drawing process?" and "How can the knowledge about drawing a particular graph be generalized and applied to other graphs?".

At the moment, we are investigating actions performed by humans when using interactive graph drawing systems, so that the perceptions and the decisions made by these users can be registered and eventually interpreted, modeled and transformed into algorithms.

ACKNOWLEDGMENT

The authors would like to thank the Brazilian state agency FAPEG for supporting this work with a master scholarship and other financial resources that allowed the necessary infrastructure for research. We also thank Professor Leslie Richard Foulds, National Senior Visiting Professor at INF-UFG, for revising this paper and suggesting important improvements.

REFERENCES

[1] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, "Algorithms for drawing graphs: an annotated bibliography," Computational Geometry, vol. 4, no. 5, 1994, pp. 235–282.

[2] "Graphviz - graph visualization software," URL: http://www.graphviz.org/ 2015.01.02 [accessed: 2015-01-02].

[3] "Gephi - the open graph viz platform," URL: http://gephi.github.io/ [accessed: 2015-01-02].

[4] H. A. D. do Nascimento, "User hints for optimization processes," Ph.D. dissertation, University of Sydney, Australia, November 2003.

[5] C. F. X. M. Neto, "A layout system for information system diagrams," Ph.D. dissertation, University of Queensland, Australia, March 1994.

[6] C. F. X. M. Neto and P. Eades, "Learning aesthetics for visualization," in Anais do XX Seminário Integrado de Software e Hardware, 1993, pp. 76–88.

[7] T. Masui, "Evolutionary learning of graph layout constraints from examples," in Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology, ser. UIST '94. New York, NY, USA: ACM, 1994, pp. 103–108.

[8] A. Rosete-Suarez, M. Sebag, and A. Ochoa-Rodriguez, "A study of evolutionary graph drawing," Laboratoire de Recherche en Informatique (LRI), Universite Paris-Sud XI, Tech. Rep. 1228, 1999.

[9] H. J. C. Barbosa and A. M. S. Barreto, "An interactive genetic algorithm with co-evolution of weights for multiobjective problems," in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), 2001, pp. 203–210.

[10] B. Bach, A. Spritzer, E. Lutton, and J.-D. Fekete, "Interactive random graph generation with evolutionary algorithms," in Graph Drawing, ser. Lecture Notes in Computer Science, W. Didimo and M. Patrignani, Eds. Springer Berlin Heidelberg, 2013, vol. 7704, pp. 541–552.

[11] M. Spönemann, "Evolutionary meta layout of graphs," Institut für Informatik, Christian-Albrechts-Universität zu Kiel, Deutsche Nationalbibliothek, Tech. Rep., 2014, in English, 21 pages.

[12] "Aviz: Visual analytics project - graphcuisine," URL: http://www.aviz.fr/Research/Graphcuisine [accessed: 2015-01-05].

[13] R. Cimikowski and P. Shope, "A neural-network algorithm for a graph layout problem," IEEE Transactions on Neural Networks, vol. 7, no. 2, 1996, pp. 341–345.

[14] Y. Takefuji and K.-C. Lee, "A near-optimum parallel planarization algorithm," Science, vol. 245, no. 4923, 1989, pp. 1221–1223.

[15] J. J. Hopfield and D. W. Tank, ""neural" computation of decisions in optimization problems," Biological Cybernetics, vol. 52, no. 3, 1985, pp. 141–152.

[16] R.-L. Wang and K. Okazaki, "Artificial neural network for minimum crossing number problem," in Proceedings of 2005 International Conference on Machine Learning and Cybernetics, vol. 7, 2005, pp. 4201–4204.

[17] B. Meyer, "Self-organizing graphs - a neural network perspective of graph layout," in Graph Drawing, ser. Lecture Notes in Computer Science, S. Whitesides, Ed. Springer Berlin Heidelberg, 1998, vol. 1547, pp. 246–262.

[18] J. Stolfi, H. A. D. do Nascimento, and C. F. X. M. Neto, "Heuristics and pedigrees for drawing directed graphs," Journal of the Brazilian Computer Society, vol. 6, 07 1999, pp. 38 – 49.

[19] O. Niggemann and Benno, "A meta heuristic for graph drawing: learning the optimal graph-drawing method for clustered graphs," in Proceedings of the working Conference on Advanced Visual Interfaces, ser. AVI '00. New York, NY, USA: ACM, 2000, pp. 286–289.

 ISBN: 978-1-61208-386-5