IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 April 2023, accepted 23 May 2023, date of publication 29 May 2023, date of current version 8 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3281260

RESEARCH ARTICLE

Experiments and a User Study for Hierarchical Drawings of Graphs

PANAGIOTIS LIONAKIS, GIORGOS KRITIKAKIS, AND IOANNIS G. TOLLIS
Department of Computer Science, University of Crete, 70013 Heraklion, Greece

Corresponding author: Panagiotis Lionakis (lionakis@csd.uoc.gr)

This work was supported in part by the University of Crete and the publication of the article in OA mode was financially supported by HEAL-Link.

ABSTRACT We present a detailed hierarchical graph drawing technique that is based on the *Path Based Framework (PBF)*. Extensive edge bundling is applied to draw all edges of the graph and the height of the drawing is minimized using compaction. We present experimental results and a user study for hierarchical drawings of graphs that show the usefulness of the new drawing technique. The drawings produced by the new framework are compared to drawings produced by the well known Sugiyama framework in terms of area, number of bends, number of crossings, and execution time. The new algorithm runs very fast and produces drawings that are readable and more efficient in terms of area and number of bends. Furthermore, since there are advantages (and disadvantages) to both frameworks, we performed a user study. The results show that the drawings produced by the new framework are well received in terms of clarity, readability, and usability. Hence, the new technique offers an interesting alternative to drawing hierarchical graphs, and is especially useful in applications where user defined paths are important and need to be highlighted.

INDEX TERMS Hierarchical graph drawing, edge bundling, experimental results, user study.

I. INTRODUCTION

Hierarchical graphs are very important for many applications in several areas of research and business because they often represent hierarchical relationships between objects in a structure. They are directed (often acyclic) graphs and their visualization has received significant attention recently [1], [2], [3]. An experimental study of four algorithms specifically designed for (Directed Acyclic Graphs) DAGs was presented in [4]. DAGs are often used to describe processes containing some long paths, such as in project management applications, model-based engineering, constraint graphs, and biological networks, protein-protein interactions networks, and more, see for example [3], [5], [6], [7]. The paths can be either application-based, e.g., critical paths, user defined, or automatically generated paths. Furthermore, DAGs are used to model causal graphs. They provide a simple way to graphically represent and understand key concepts of causal relationships such as in epidemiology and in clinical studies [8], [9].

The associate editor coordinating the review of this manuscript and approving it for publication was Walter Didimo.

The Sugiyama Framework [10] has been extensively used in research [3] and in industrial systems [11], [12] to visualize hierarchical graphs over the past decades. A new framework to visualize directed graphs and their hierarchies was introduced in [13] and [14]. It is based on a path/channel decomposition of the graph and is called *(Path-Based Framework or PBF)*. It computes readable hierarchical visualizations in two phases by *hiding* (abstracting) some selected edges, while maintaining the complete reachability information of the graph. However, these drawings are not satisfactory to users that need to visualize all the edges of the graph.

In this paper, we extend the hierarchical graph drawing framework (PBF) of [13] and [14] in order to draw all the edges of the graph using extensive edge bundling. We also minimize the height of the drawing using a compaction technique. The width of the drawing of the DAG is minimized by applying algorithms similar to task scheduling. The total time required for the aforementioned extensions is $O(m + n \log n)$, where m is the number of edges and n the number of nodes of a graph G, which means that all problems (except cycle removal) are solved in polynomial time. The main

contributions of our work are summarized as follows: a) we present a polynomial technique for drawing a DAG with small area and few bends, b) we present experimental results that show that our technique is faster and produces better drawings than the ones produced by the most updated (state of the art) implementation [15] of the Sugiyama framework [10], and c) we present results from an extensive user study that confirm that the drawings produced by our technique are clearer, more understandable, and more user friendly. Figure 1 shows an example DAG drawn by the two frameworks.

Another advantage of our technique is that the edges of G are naturally split into three categories: *path edges, path transitive edges,* and *cross edges*. Path edges connect consecutive vertices in the same path, see Figure 1(a) (see also Figure 2b). Path transitive edges connect non-consecutive vertices in the same path and can be easily bundled, see the blue edges in Figure1(a) (see also Figure 2c. Cross edges connect vertices that belong to different paths, see the edges between the left and right path in Figure 1(a) (see also Figure 2d). Figure 2 shows a small DAG and the three categories of edges.

The Sugiyama Framework consists of four phases, and for each phase various methods and variations have been proposed in the literature over the years, see [3] for more recent information regarding this framework. A comparative study [4] concluded that the Sugiyama-style algorithms performed better in most of the metrics with respect to other alternatives available at the time. Commercial software such as the Tom Sawyer Software TS Perspectives [11] and yWorks [12], use this framework in order to offer automatic visualizations of directed graphs. Even though it is very popular, the Sugiyama Framework has several important limitations: most problems and subproblems that are used to optimize the results in various steps of each phase have turned out to be NP-hard. Additionally, the insertion of dummy vertices increases the size of the input graph. Hence, the overall time complexity of this framework (depending upon implementation) can be $O((nm)^2)$, or even higher if one chooses algorithms that require exponential time. Another important limitation of this framework is the fact that heuristic solutions and decisions that are made during previous phases (e.g., crossing reduction) will influence severely the results obtained in later phases. Nevertheless, previous decisions cannot be changed in order to obtain better results.

By contrast, the path-based framework [14] departs from the typical Sugiyama Framework as it consists of only two phases: (a) Cycle Removal, (b) the path/channel decomposition and hierarchical drawing step. Furthermore, most problems of the second phase can be solved in polynomial time. It is based on the idea of partitioning the vertices of a graph into node disjoint paths/channels, where in a channel consecutive nodes are connected by a path but not necessarily connected by an edge. In the rest, we only use the term ''path'' but of course our algorithms work also for ''channels.'' The vertices in each path are drawn vertically aligned on some x-coordinate; next the edges between vertices that belong to different paths are drawn. Note that the computation of a path/channel decomposition of minimum cardinality takes polynomial time [16], [17], [18], [19].

If a path decomposition contains k paths, the number of bends introduced is at most $O(kn)$ and the required area is at most $O(kn)$. We also show how to draw optimally the *path transitive edges* that are not drawn in the framework of [14]. We present experimental results comparing drawings obtained by the extended-PBF to the drawings obtained by running the hierarchical drawing module of Open Graph Drawing Framework (OGDF) [15], which is based on the Sugiyama Framework, and is a quite active research software that implements this framework. The results show that PBF runs much faster, and produces drawings with better area and less bends. OGDF produces drawings with less crossings, especially for sparse graphs. Furthermore, the two frameworks produce vastly different drawings.

Therefore, we decided to perform a user study between these two drawing frameworks, in order to obtain feedback from users using these drawings. The users had to perform a set of tasks on the drawings of some DAGs. The tasks include determining if two given vertices are connected, finding the length of a shortest path, and determining if some vertices are successors of a given vertex. The users' answers were correct above 90% for PBF and above 84% for the Sugiyama Framework (implemented in OGDF). The users were also asked to express their preference, in terms of clarity and readability, between the two frameworks. 58.3% of the users showed a clear preference to using drawings produced by PBF. Hence, this technique offers an interesting alternative to drawing hierarchical graphs, especially if there are user defined paths that need to be clearly visualized. Section IV describes a detailed analysis of the user study and the experimental results.

II. OVERVIEW OF THE TWO FRAMEWORKS

The two hierarchical drawings shown in Figure 1 demonstrate the significant differences between the two frameworks: Part (a) shows a drawing of G computed by our algorithms that customize PBF. Part (b) shows the drawing of G computed by the Sugiyama Framework (as implemented in OGDF). The graph consists of 20 nodes and 31 edges. The drawing computed by our algorithms has 12 crossings, 18 bends, width 10, height 15, and area 150. On the other hand, OGDF computes a drawing that has 5 crossings, 22 bends, width 18, height 15 and area 270. Clearly, the two frameworks produce vastly different drawings with their own advantages and disadvantages.

The original Path Based Hierarchical Drawing Framework follows an approach to visualize DAGs that hides some edges and maintains their reachability information [14]. As discussed before this framework is based on the idea of partitioning the vertices of the graph G into (a minimum number of) *channels/paths*, called *channel/path decomposition* of G, which can be computed in polynomial time. By contrast, the

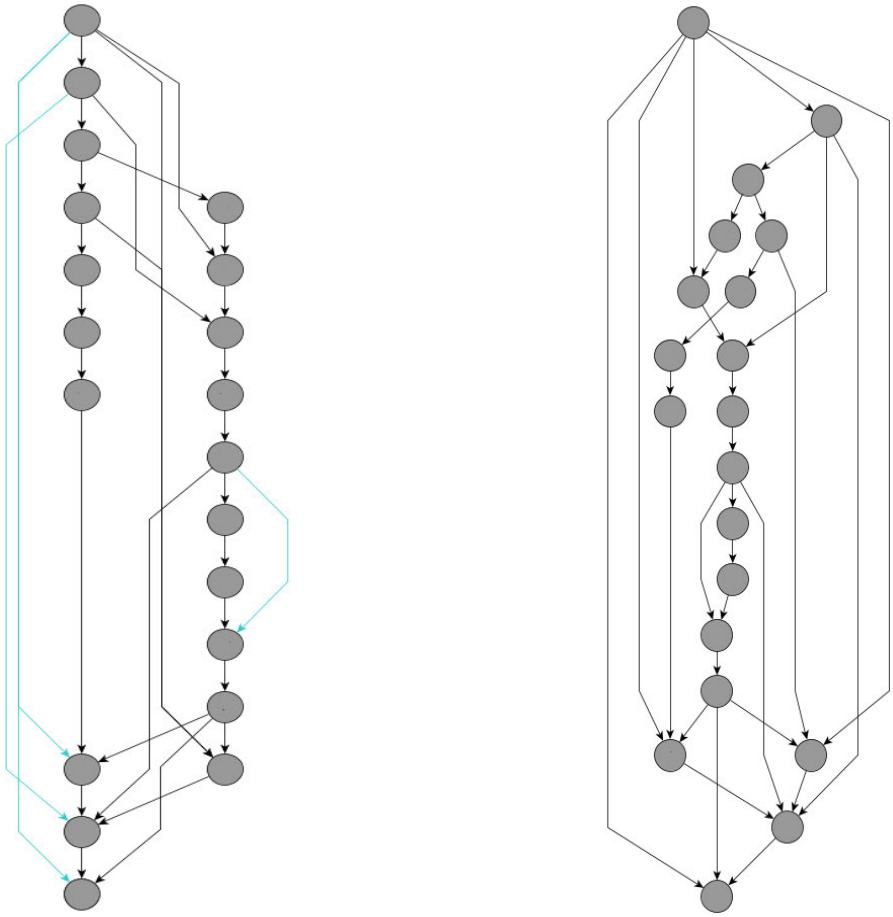

FIGURE 1. Example of a DAG *G* drawn by our proposed framework (left). Same DAG drawn by the Sugiyama framework as implemented in OGDF (right).

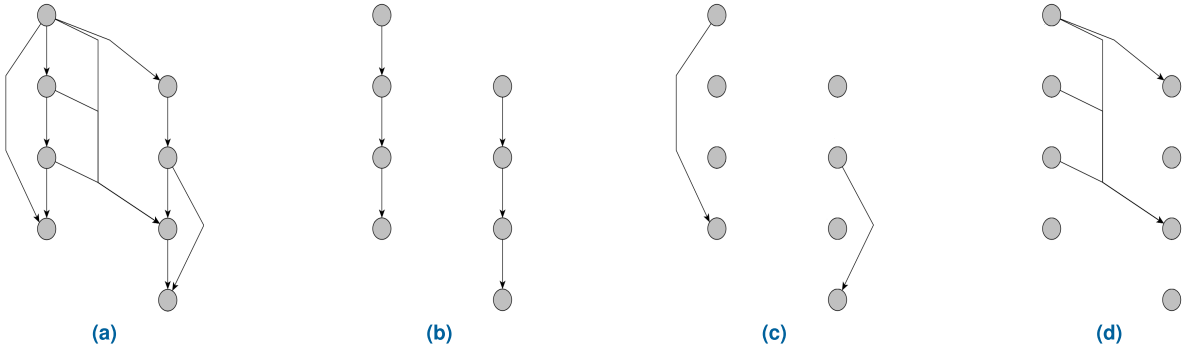

FIGURE 2. Figure 2a shows a graph *G* drawn by PBF. Figure 2b shows the *path edges* of *G*. Figure 2c shows the *path transitive edges* of *G*. Figure 2d shows the cross edges of *G*.

Sugiyama framework performs a horizontal decomposition of a graph, even though the final result is a vertical (hierarchical) visualization. In this sense PBF is *orthogonal* to the Sugiyama framework since it is a vertical decomposition of G into (vertical) paths/channels. Thus, most resulting problems are *vertically contained*, which makes them simpler, and reduces their time complexity. This framework does not need to introduce any dummy vertices and keeps the vertices of a path *vertically aligned*, which is important for specific applications (such as visualizing critical paths in Program Evaluation Review Technique (PERT) diagrams [5]).

Let $S_p = \{P_1, \ldots, P_k\}$ be a path decomposition of G such that every vertex $v \in V$ belongs to exactly one of the paths of S_p. Another advantage of PBF is that any path decomposition naturally splits the edges of G into: (a) *path edges* (b) *cross edges* and (c) *path transitive edges*, that are drawn differently.

Given any path decomposition S_p of a DAG G with n vertices and m edges, the main algorithm of [14], draws the vertices of each path P_i in S_p *vertically aligned* on some x-coordinate depending on the order of path P_i. The y-coordinate of each vertex is equal to its order in any topological sorting of G. Hence the height of the resulting drawing is $n-1$. It is also important to highlight that the Path-Based Framework works for any given path decomposition. Therefore, it can be used in order to draw graphs with user-defined or application-defined paths, as is the case in many applications, see for example [5], [6]. If one desires automatically generated paths, there are several algorithms that compute a path decomposition of minimum cardinality in polynomial time [16], [17], [18], [19]. Since certain critical paths are important for many applications, it is extremely important to produce clear drawings where all such paths are vertically aligned, see [14]. In the rest of this paper, we assume that a path decomposition of G is given as part of the input to the algorithm.

OGDF is a self-contained C++ library of graph algorithms, in particular for (but not restricted to) automatic graph drawing. The hierarchical drawing implementation of the Sugiyama Framework in OGDF is implemented following [20], [21] and it uses the following default choices: For the first phase of Sugiyama, it uses the *LongestPathRanking* (to assign vertices into layers) which implements the well-known longest-path ranking algorithm. This is where many dummy vertices are introduced, which increases the size of the original graph. Next, it performs crossing minimization using the *BarycenterHeuristic*. This module performs two-layer crossing minimization and is applied during the top-down and bottom-up traversals [15]. The crossing minimization is repeated 15 times, and it keeps the best configuration. Finally, the final-coordinates (drawing) are computed, attempting to reduce the number of bends and final crossings, with *FastHierarchyLayout* layout of OGDF.

III. COMPUTING COMPACT AND BUNDLED DRAWINGS

Now we present an extension of the framework of [14] by (a) compacting the drawing in the vertical direction, and (b) drawing the path transitive edges that were not drawn in [14]. Since the edges of G are split into three categories, it clearly adds new possibilities to the understanding of the user and allows a system to show the different edge categories separately, without altering the user's mental map.

A. COMPACTION

Let $G = (V, E)$ be a DAG with n vertices and m edges. Following the framework of [13] and [14] the vertices of V are placed in a unique y-coordinate, which is specified by a topological sorting. Let T be the list of vertices of V in ascending order based on their y-coordinates. We start from the bottom and visit each vertex in T in ascending order. For every vertex v in this order we assign a new y-coordinate, $y(v)$, following a simple rule that compacts the height of the drawing: "If v has no incoming edges then we set its $y(v)$ equal to 0, else we set $y(v)$ equal to $a+1$, where a is the *highest* y-coordinate of the vertices that have edges incoming into v."

The produced drawings as shown in Figure 3, have the following simple properties:

Property 1. Two vertices of the same path are assigned distinct y-coordinates.

Property 2. For every vertex v with $y(v) \neq 0$, there is an incoming edge into v that starts from a vertex w such that $y(v) = y(w) + 1$. Based on the above algorithm and property the height of the compacted drawing of graph G is at most L and it can be computed in $O(n+m)$ time.

B. DRAWING THE PATH TRANSITIVE EDGES

An important aspect of our work is the preservation of the mental map of the user that can be expressed, in part, by the reachability information of a DAG. Recall that path transitive edges are not drawn by the framework of [13] and [14]. In this subsection we show how to bundle and draw these edges while preserving the user's mental map of the previous drawing. Additionally, one may interact with the drawings by hiding the path transitive edges at the click of a button without changing the user's mental map of the complete drawing. We describe an algorithm that bundles and draws the path transitive edges using the minimum extra width (minimum extra number of columns) for each (decomposition) path as shown in Figure 4. The steps of the algorithm are briefly described as follows:

1) For every vertex of each decomposition path we compute the indegree and outdegree based only on path transitive edges.
2) If all indegrees and outdegrees are zero the algorithm is over, if not, we select a vertex v with the highest indegree or outdegree and we bundle all the incoming or outgoing edges of v, respectively. These bundled edges are represented by an *interval* with starting and finishing points, the lowest and highest y-coordinates of the vertices, respectively.
3) Next, we insert each interval on the left side of the path on the first available column such that the interval does not overlap with another interval.
4) We remove these edges from the set of path transitive edges, update the indegree and outdegree of the vertices and repeat the selection process.
5) The intervals of the rightmost path, are inserted on the right side of the path in order to avoid potential crossings with cross edges.
6) A final, post-processing step can be applied because some crossings between intervals/bundled edges can be removed by changing the order of the columns containing them.

The above algorithm can be implemented to run in $O(m + n\log n)$ time by handling the updates of the indegrees and outdegrees carefully, and placing the appropriate

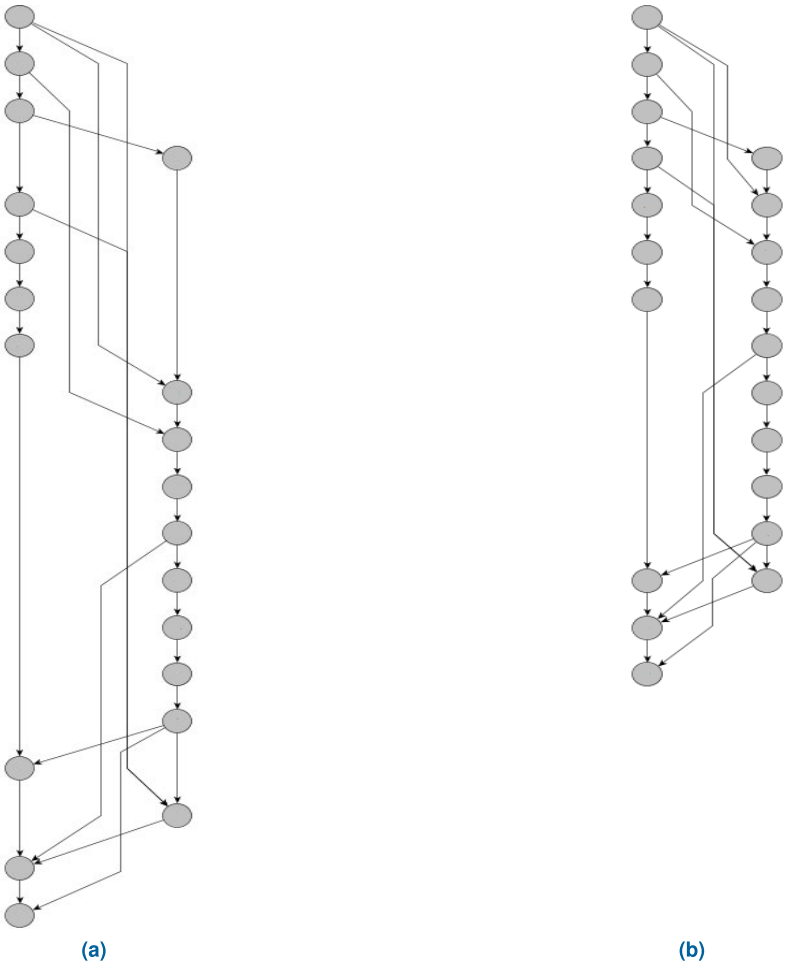

FIGURE 3. A DAG G drawn without its path transitive edges: In 3a, drawing Γ_1 is computed, without applying compaction; in 3b, drawing Γ_2 is computed applying the compaction step.

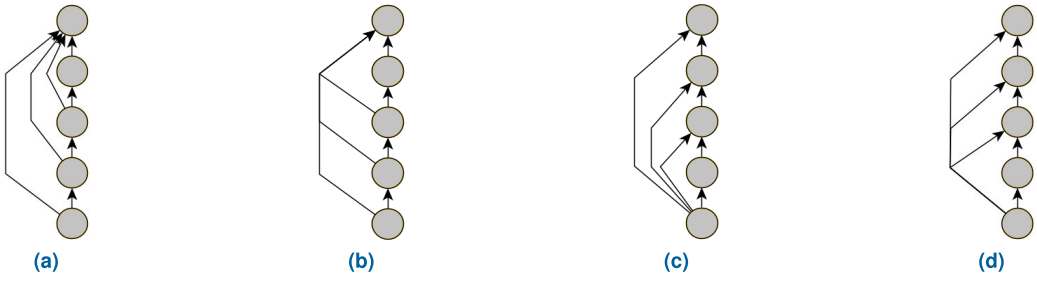

FIGURE 4. Examples of Bundling path transitive edges from left to right: (a) incoming edges into the last vertex of the path, (b) bundling the incoming edges, (c) outgoing edges from the first vertex of the path, (d) bundling the outgoing edges.

intervals in a (Max Heap) Priority Queue. As expected, the fact that we draw the path transitive edges increases the number of bends, crossings, and area, with respect to not drawing them.

For each decomposition path, suppose we have a set of b intervals such that each interval I has a start point, s_I, and a finish point f_I. The starting point is the position of the vertex of the interval with the lowest y-coordinate. Similarly, the finish point f_I is the position of the node of the interval with the highest y-coordinate. We follow a greedy approach in order to minimize the width (number of columns) for placing the bundled edges. The approach is similar to Task Scheduling [22], for placing the intervals. It uses the optimum number of columns and runs in $O(b \log b)$ time, for each path with b intervals. Since the sum of all b's for all paths in a path decomposition is at most n we conclude that the algorithm runs in $O(n \log n)$ time. For details and proof of correctness see [22].

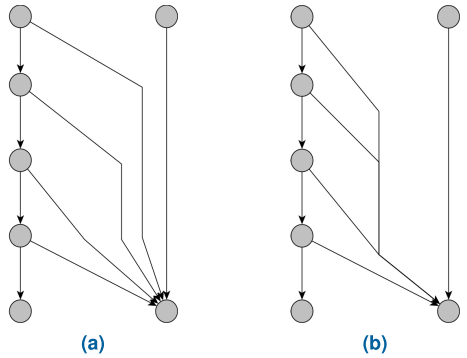

FIGURE 5. Bundling of cross edges: Figure 5a shows the cross edges before bundling while Figure 5b shows the cross edges after bundling.

C. DRAWING AND BUNDLING THE CROSS EDGES

Cross edges connect vertices that belong to different paths. The number of bends of any cross edge (u, v) depends on the vertical distance of its incident nodes, u and v, and is determined as follows. An example before and after bundling of cross edges is shown in Figures 5a and 5b, respectively.

1) Two bends if the vertical distance between u and v is more than two.
2) One bend if the vertical distance between u and v is two.
3) The edge is a straight line segment (no bend) if the vertical distance between u and v is one.

We bundle all incoming cross edges for each vertex whose vertical distance is more than one unit from the target. The bundled cross edges are placed between the paths/channels using the same technique we used for path transitive edges, i.e., using a technique that relies on task scheduling, as above.

IV. EXPERIMENTAL EVALUATION AND USER STUDY

In this section we present experimental results obtained by the extended path-based framework and we compare them with the respective experimental results obtained by running the hierarchical drawing module of OGDF. In order to evaluate the performance, we used the following standard metrics:

- Number of crossings.
- Number of bends.
- Width of the drawing: The total number of distinct x coordinates that are used by the framework.
- Height of the drawing: The total number of distinct y coordinates that are used by the framework.
- Area of the drawing: The area of the enclosing rectangle.

Based on these metrics, we conducted a number of experiments to evaluate the performance of the two hierarchical drawing frameworks using a dataset of 20 randomly generated DAGs. The dataset was created using the Path-Based DAG Model [23]. In this model, graphs are randomly generated based on a number of predefined but randomly created paths. Additionally, the metrics of PBF could vary depending on the path/channel decomposition algorithm we use and the ordering of the columns.

TABLE 1. Results on the *number of crossings, bends, width, height* and *area* for *PBF* and *OGDF* for all the DAGs created by the generator.

n=50	m=62		m=87		m=150		m=250		m=500	
	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF
Crossings	17	6	126	92	839	703	2469	2585	8061	14479
Bends	15	25	22	69	54	188	91	380	176	863
Width	12	36	13	59	18	116	24	206	33	442
Height	13	16	17	21	20	23	21	28	24	33
Area	156	576	221	239	360	2668	504	5768	792	14586
n=100	m=125		m=175		m=300		m=500		m=1000	
	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF
Crossings	105	29	705	430	3749	3366	13068	12890	42934	62695
Bends	29	50	59	143	108	388	194	757	324	1737
Width	18	60	20	103	26	230	36	414	51	912
Height	22	27	22	32	26	30	27	28	38	45
Area	396	1620	440	3296	676	6900	972	11592	1938	41040
n=200	m=250		m=350		m=600		m=1000		m=2000	
	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF
Crossings	594	278	3094	1929	16357	12490	52095	49278	209446	266260
Bends	48	100	128	288	226	763	350	1519	597	3498
Width	23	107	32	216	37	450	52	830	83	1813
Height	27	46	33	48	39	40	38	42	49	60
Area	621	4922	1056	10368	1443	18000	1976	34860	4067	108780
n=500	m=625		m=875		m=1500		m=2500		m=5000	
	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF
Crossings	2746	1501	15474	11221	102195	81537	389241	327017	1486777	1636057
Bends	123	246	280	730	544	1916	911	3909	1482	8802
Width	41	260	51	531	71	1142	96	2103	138	4565
Height	42	78	39	71	50	57	64	74	79	90
Area	1722	20280	1989	37701	3550	65094	6144	155622	10902	410850

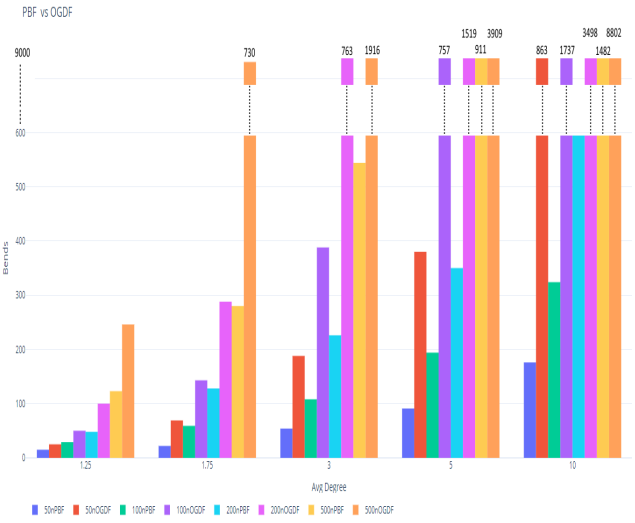

FIGURE 6. Results on the number of bends for *PBF* and *OGDF* for all the DAGs created by the generator.

Table 1 shows the results of our experiments based on these metrics for *PBF* as implemented in TS Perspectives [11] compared to the results produced by OGDF. In order to be consistent with the experimental settings of OGDF, we used the default parameters. In the experiments that we present in this section, we see that in all cases our approach gives better results than the ones produced by OGDF with respect to the number of bends, width, height, and as expected the total area of the drawings. For the number of bends we observe that our proposed technique produces bends that are a small fraction of n, whereas OGDF produces bends that are proportional to m. The bar charts (see Figure 6) show how the number of bends grows as the DAGs grow in size and average degree and provide a clear evidence that the number of bends for *PBF* is significantly lower than OGDF in all cases. On the other hand, the drawings of OGDF have a lower number of crossings when the input graphs are relatively sparse. However, when the graphs are a bit denser (e.g., average degree higher than

TABLE 2. Results on the number of crossings, bends, width, height and area for PBF and OGDF for a sample of DAGs taken from www.graphdrawing.org.

n=20	Average Degree=1.6									
	graph I		graph II		graph III		graph IV		graph V	
	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF
Crossings	12	5	16	5	17	4	17	3	16	4
Bends	14	22	17	22	14	21	14	24	17	22
Width	9	18	8	18	11	19	8	16	10	14
Height	14	14	15	15	13	13	12	12	15	15
Area	126	252	120	270	143	247	96	192	150	210
n=50	Average Degree=1.6									
	graph I		graph II		graph III		graph IV		graph V	
	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF	PBF	OGDF
Crossings	119	39	103	40	101	35	135	39	136	40
Bends	36	59	33	55	38	58	36	58	42	64
Width	24	51	19	40	21	45	21	48	19	57
Height	27	27	25	25	31	31	23	23	31	31
Area	648	1377	475	1000	651	1395	483	1104	589	1767

five) the PBF drawings have less crossings. Since the two frameworks use a different coordinate system, for a fair comparison between them we chose to count as height of a drawing the number of different layers (or different y-coordinates) and as width the number of different x-coordinates of nodes and bends, used by each system. In other words, we normalize the two coordinate systems by mapping them on a ''grid''.

In general, our experiments showed that PBF produces readable drawings. Additionally, it clearly partitions the edges into three distinct categories, and vertically aligns certain paths, which may be user/application defined. This may be important in certain applications. The results showed that our approach differs from the Sugiyama Framework completely, since it examines the graph vertically. The extended PBF performs bundling very efficiently and computes the optimal height of the graph. In most cases, the drawings based on PBF need less area than OGDF and contain fewer bends. On the other hand, OGDF generally has fewer crossings than PBF. This is expected since OGDF places a major computational effort into the crossing minimization step, whereas PBF does not perform any crossing minimization. Figures 7, 8 show that the time for PBF grows linearly in contrast to ODFG where its time complexity seems to be cubic. We observe that comparing quantitative metrics alone does not lead to a concrete conclusion. Hence, we decided to perform a user study in order to evaluate the readability and clarity of PBF comparing it with the Sugiyama framework from the perspective of a user.

A. USER STUDY

In order to proceed with our user evaluation, we conducted another series of experiments in order to further validate the previous statements with respect to the graphs that will be chosen for the user study. To this respect, we used the benchmarks found in www.graphdrawing.org. For the user evaluation we chose appropriate graphs in terms of size and density. Namely, we selected rather small and sparse graphs so the users can perform the series of tasks required for the evaluation. The archive consisted of graphs with 10 to 100 nodes with average degree about 1.6. The results of these additional experiments were similar to the results reported

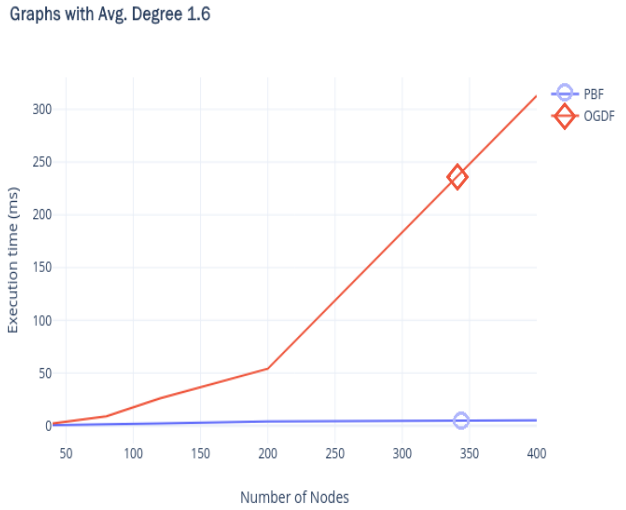

FIGURE 7. Execution time of *PBF* and *OGDF* on Graphs with average degree 1.6.

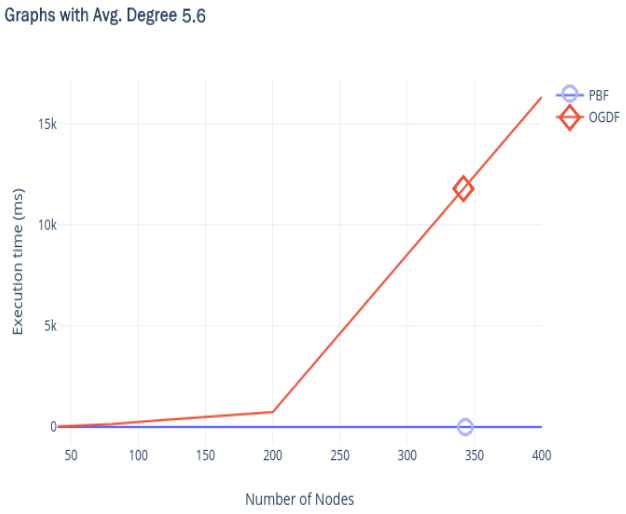

FIGURE 8. Execution time of *PBF* and *OGDF* on Graphs with average degree 5.6.

above, see Table 2. They also highlighted the fact that the two frameworks focus on different aspects of the graphs and produce vastly different drawings (notice that for such small and sparse graphs both techniques produce drawings of optimum height). Finally, from these graphs, we chose four graphs (two with 20 vertices and two with 50 vertices) that were most appropriate to be used by the users for the user study.

Evaluation of visualizations [24] and analysing data is a difficult research challenge [25], [26]. Motivated by the work of [27], we designed an evaluation with users, with a set of tasks that focus on revealing any usability issues. To this respect, we chose a set of ''cognitive tasks'' [28] where we evaluated the two systems based on attributes of clarity and readability as expressed by the reachability

TABLE 3. Graphs dataset.

Graph	Number of nodes	Number of edges	Graph Sizes
Graph1	20	31	Small Graphs
Graph2	20	31	
Graph3	50	82	Medium Graphs
Graph4	50	79	
Graph5	100	163	Large Graphs
Graph6	100	169	

TABLE 4. The set of tasks participants had to answer for each of the 2 different graph drawing frameworks over various graphs.

ID	Task Description
1	Is there a path between the two highlighted vertices?
2	How long is the shortest path between the two highlighted vertices?
3	Is there a path of length at most 3 that connects the two highlighted nodes?
4	Are all of the green vertices successors of the red vertex?

information within a drawing. Moreover, in order to design our evaluation we took into account the various restrictions of [28] where the authors emphasized the basic guidelines in that particular direction. In our case, the participants had to choose an answer among ''Yes'',''No'', or ''Do Not Know''. Even though there was no time limit, the participants were expected to answer ''Do Not Know'' if a question was too difficult or too time-consuming to answer. The purpose of this assumption was to focus on tasks that can be easy also for non-expert users in order to detect any usability issues.

1) USERS

We recruited 72 participants. In order to have more accurate and sophisticated results, we selected an audience that was familiar with graph theory and graph drawing styles. More specifically, 35% were software developers and researchers and 65% were postgraduate and advanced undergraduate students.

2) TRAINING

We created a Google form and we invited all participants to fill it in. Initially, the users were asked to watch a short video used for training: the tutorial gave a short description of the two hierarchical drawing frameworks (the video is available at https://youtu.be/BWHc2xO4jmI).

3) DATASETS

We experimented with a dataset of 3 graph categories with different number of nodes (20 nodes, 50 nodes and 100 nodes, i.e., small, medium and large graphs) with average degree around 1.6 (Table 3).

4) TASKS

We asked the users to answer a set of questions for the two different drawings and carry out a sequence of basic tasks. Similar to previous user studies (see, e.g., [29], [30], [31], [32]), we decided to choose tasks involving graph reading which are easily understandable also to non-expert users. Moreover, we also took into account that the purpose was to evaluate hierarchical drawings and as expected some tasks such as counting incoming or outgoing edges were rather simple and they would not produce useful insights. Thus, we considered the tasks shown in Table 4.

For questions on Task 2, the participants had to choose a number as an answer. We did not require numeric answers for all the tasks. More specifically, for Tasks 1, 3 and 4 the participants had to choose an answer among ''Yes'', ''No'', or ''Do Not Know''. There was no time limit, although participants were expected to answer ''Do Not Know'' if a question was too difficult or too time-consuming to answer. Each of the previous tasks was repeated for each drawing framework model 4 times: i.e., 2 using small size graphs and 2 using medium size graphs. Note that for each question of the same task we used different highlighted nodes. In total, the number of tasks was 32 i.e., 16 questions for PBF drawings and 16 for OGDF drawings. The questions on small graphs (20 nodes) preceded those on medium graphs (50 nodes). Finally, to counteract the learning effect, the questions appeared in a randomized order. Figure 9 shows a snapshot for the question ''Is there a path between the two highlighted vertices?'' for both drawings.

5) TASK BASED COMPARISON

For the results, we recorded the total *number of correct answers* for each question of the 2 different drawing frameworks, for all participants. Also note that the ''Do Not Know'' answer was considered incorrect. More specifically, regarding the first graph (i.e., *graph1*) for all tasks, we have that the users had the same performance in terms of correct answers for both *PBF* and *OGDF* drawings. By examining the rest of the results the average percentage revealed that for all the graphs the performance of both drawing frameworks was not significantly different, although it was slightly better for *PBF* drawings for almost all tasks and drawings. We observed the same when comparing the average percentage for each task: the numbers for PBF drawings were consistently better than the numbers for OGDF drawings, but the differences were small. We showed a comprehensive visualization of these results in Figure 10. It shows the ratio of participants that answered *"Yes"*, *"No"* and *"Do Not Know"* on the various tasks for each of the two drawing frameworks, over the different graphs where we observed again that *PBF* is slightly better than *OGDF* for all cases (Tables of Figures 13, 14). We observed that although the numbers are very low for both, the number of users that answered *"Do Not Know"* for OGDF is often double the corresponding number for PBF, which may imply that some OGDF drawings may be more confusing to some users.

In general, the performance of the participants is slightly better when they are working with PBF drawings than with OGDF drawings. Since the differences are rather small we

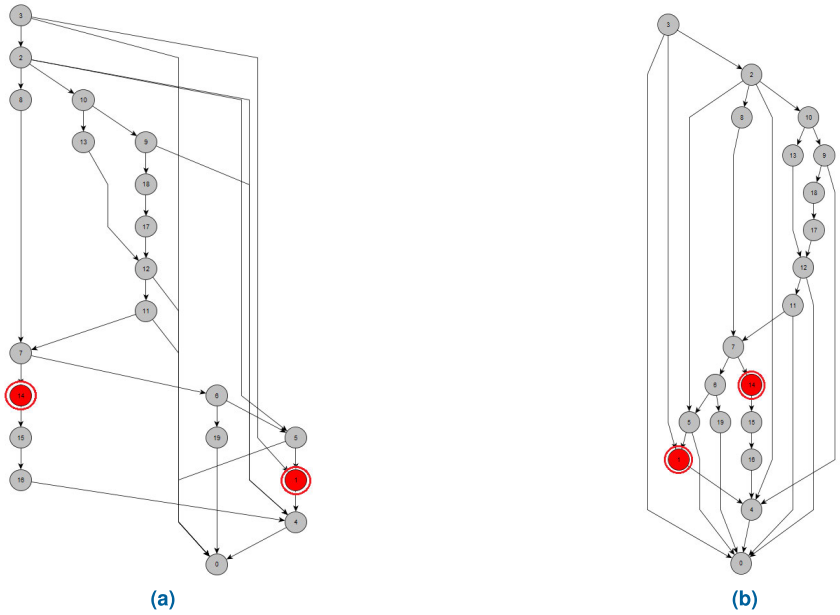

FIGURE 9. Snapshots of drawings of the same graph used in the user study for the question *"Is there a path between the two highlighted vertices?".*

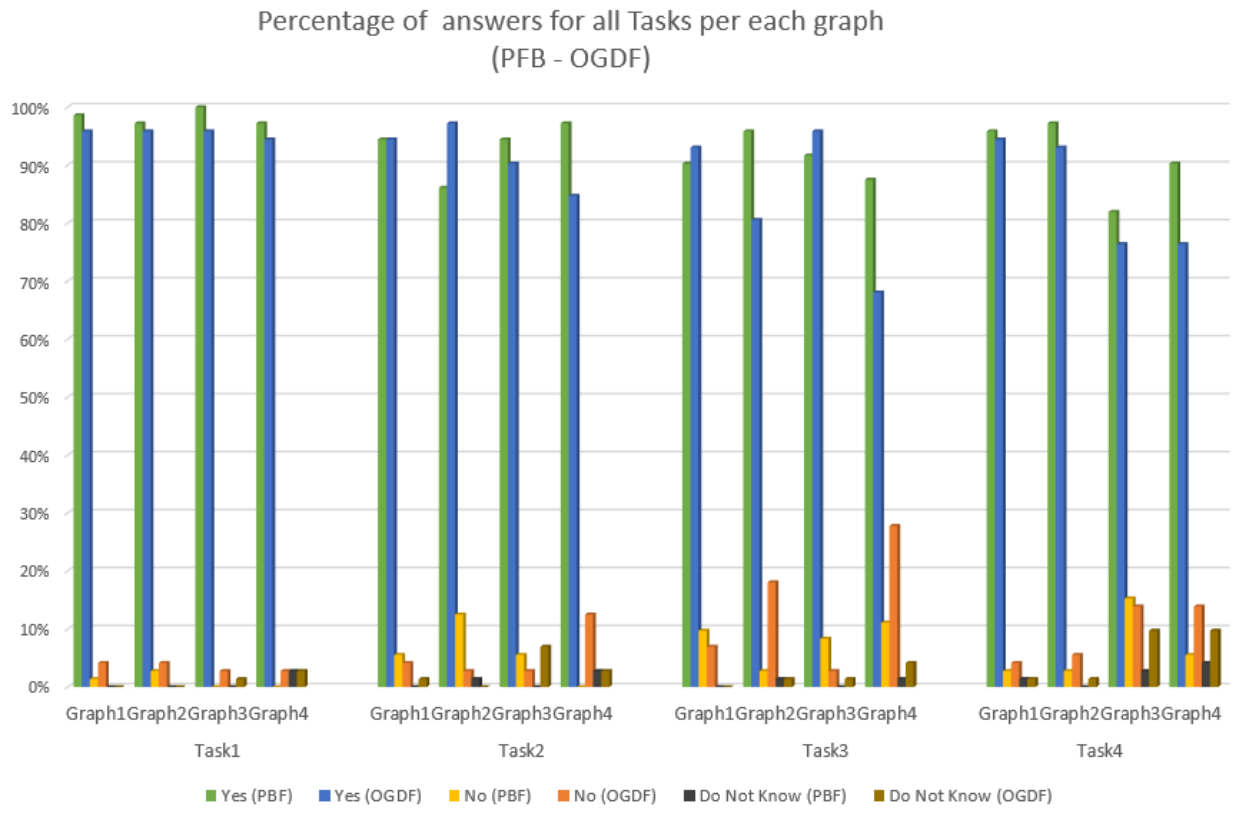

FIGURE 10. Results *(the ratio of participants that answered "Yes", "No" and "Do Not Know" over the total number of answers)* on the various tasks for each of the drawing framework *(PBF), (OGDF)* over different graphs.

cannot extract a concrete conclusion as to which is better for the users. However, it has become clear that the path-based framework is an interesting alternative to the Sugiyama Framework for visualizing hierarchical graphs. Furthermore, for specific applications, that require to visualize specific paths (such as critical paths) it would be the preferred choice since the nodes of each path are placed on the same x-coordinate.

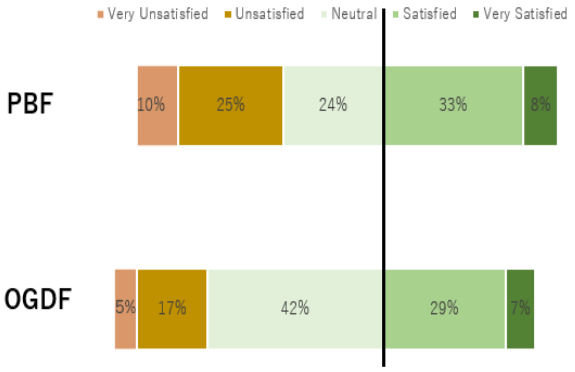

FIGURE 11. Percentage results for PBF and OGDF for the question *I*, over graph5.

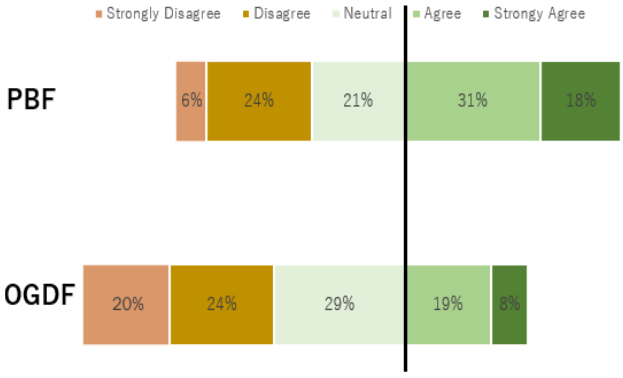

FIGURE 12. Percentage results for PBF and OGDF for the question *II*, over graph6.

	PBF					OGDF				
Graph	Avg per graph	Task1	Task2	Task3	Task4	Avg per graph	Task1	Task2	Task3	Task4
1	95%	71/72	68/72	65/72	69/72	94%	69/72	68/72	67/72	68/72
2	94%	70/72	67/72	69/72	70/72	92%	69/72	70/72	58/72	67/72
3	92%	72/72	68/72	66/72	59/72	91%	69/72	65/72	69/72	55/72
4	93%	70/72	70/72	63/72	65/72	80%	68/72	61/72	49/72	55/72
Avg per task		98%	93%	91%	91%	Avg per task	95%	91%	84%	85%

FIGURE 13. Results *(number of correct answers)* on the various tasks for each of the drawing framework *(PBF), (OGDF)* over different graphs.

	PBF					OGDF				
Graph	Avg per graph	Task1	Task2	Task3	Task4	Avg per graph	Task1	Task2	Task3	Task4
1	0.3%	0/72	0/72	0/72	1/72	0.6%	0/72	1/72	0/72	1/72
2	0.6%	0/72	1/72	1/72	0/72	0.6%	0/72	0/72	1/72	1/72
3	0.6%	0/72	0/72	0/72	2/72	4.8%	1/72	5/72	1/72	7/72
4	2.7%	2/72	2/72	1/72	3/72	4.8%	2/72	2/72	3/72	7/72
Avg per task		0.6%	1%	0.6%	2%	Avg per task	1%	2.7%	1.7%	5.5%

FIGURE 14. Results *(number of "Do Not Know" answers over the total number of all answers)* on the various tasks for each of the drawing framework *(PBF), (OGDF)* over different graphs.

6) DIRECT COMPARISON OF THE TWO FRAMEWORKS

As a second-level of analysis, we perform a direct comparison of the two drawing frameworks. We used the two (large) graphs of 100 nodes. To this respect we asked the participants to rate each of the 2 models by answering the following questions:

1) On a scale of 1 to 5, how satisfied are you with the following graph drawings?
2) Do you believe it would be easy to answer the previous tasks for the following graph?
3) Which of the following drawings of the same graph do you prefer to use in order to answer the previous tasks?

Similar to the previous experiments, we had two PBF drawings and two OGDF drawings. Since the objective of this section was to evaluate the usability of both drawing frameworks, using the System Usability Score (SUS) [33], we asked the users to answer questions *I* and *II*, by giving a rate using the following scale *<Very Unsatisfied, Unsatisfied, Neutral, Satisfied, Very Satisfied>* and *<Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree>* respectively. The results show that for Question *I*, 41% of the participants rated *PBF* from scale 4 to 5, in contrast to 36% for *OGDF*, as shown in Figure 11. Notice that the answers scale 1 and 2 are worse for

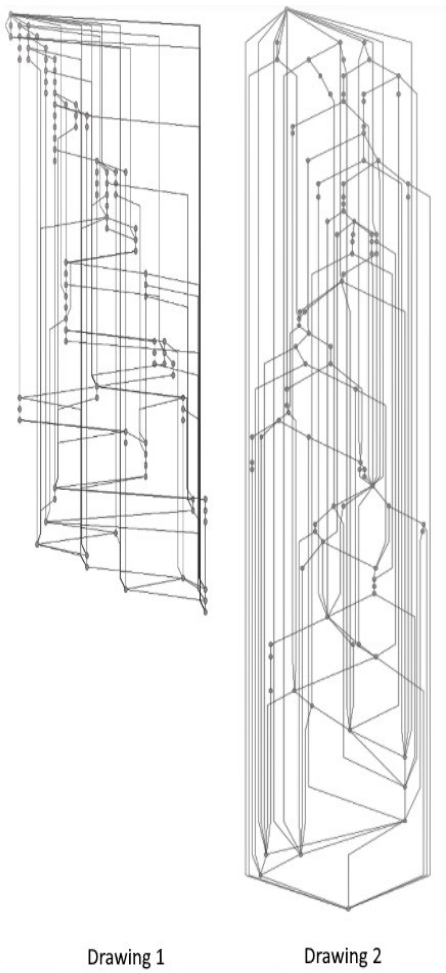

FIGURE 15. Snapshots of the same graph as used in our survey. Drawing 1 is the one computed by *PBF* and Drawing 2 is the one as produced by *OGDF*.

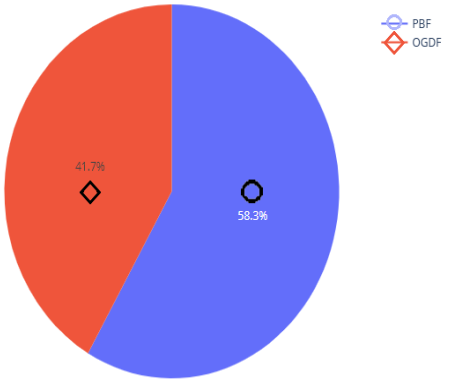

FIGURE 16. Percentage results for the task *"Which of the following drawings of the same graph do you prefer to use in order to answer the previous tasks."*

PBF. This probably signifies that the users are not familiar with this new hierarchical drawing style. For Question *II*, almost 50% of the participants rated *PBF* from scale 4 to 5, in contrast to less than 30% for *OGDF*, see Figure 12.

At the end of this user study, we asked the participants to perform a direct comparison of the two drawing frameworks for the same graph, by answering this question: "Which of the following drawings of the same graph do you prefer to use in order to answer the previous tasks?" (Figure 15). The results as shown in Figure 16 highlight that 58.3% of the participants stated that they prefer the drawing produced by *PBF* over the *OGDF* drawing. In terms of statistical significance, the exact (Clopper-Pearson) 95% Confidence Interval (CI) is (48%, 72%) indicating that PBF drawings are preferred by the users over OGDF drawings. However, given the small differences, we conclude that *PBF* is a significant alternative to the Sugiyama Framework for visualizing hierarchical graphs.

V. CONCLUSION

We present a detailed general-purpose hierarchical graph drawing framework that is based on the Path Based Framework (PBF) [14]. We apply extensive edge bundling to draw all the path transitive edges, and cross edges of the graph and we minimize its height using compaction. The experiments revealed that our implementation runs very fast and produces drawings that are efficient and readable. We also evaluated the usability of this new framework compared to *OGDF* which follows the Sugiyama Framework. The experimental results show that the two frameworks produce drawings that differ considerably. Generally, the drawings produced by our algorithms have lower number of bends and are significantly smaller in area than the ones produced by OGDF, but they have more crossings for sparse graphs. Thus the new approach offers an interesting alternative for visualizing hierarchical graphs, since it focuses on showing important aspects of a graph such as critical paths, path transitive edges, and cross edges. For this reason, this framework may be particularly useful in graph visualization systems that encourage user interaction. Moreover, the user evaluation shows that the performance of the participants is slightly better in PBF drawings than in OGDF drawings and the participants prefer PBF in overall rating compared to OGDF.

REFERENCES

[1] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.
[2] M. Kaufmann and D. Wagner, *Drawing Graphs: Methods and Models*. Springer, 2003.
[3] N. S. Nikolov and P. Healy, "Hierarchical drawing algorithms," in *Handbook of Graph Drawing and Visualization*, R. Tamassia, Ed. Boca Raton, FL, USA: CRC Press, 2014, pp. 409–453.
[4] G. Di Battista, A. Garg, G. Liotta, A. Parise, R. Tamassia, E. Tassinari, F. Vargiu, and L. Vismara, "Drawing directed acyclic graphs: An experimental study," in *Proc. Int. Symp. Graph Drawing (GD)*, in Lecture Notes in Computer Science, vol. 1190, S. C. North Ed. Berkeley, CA, USA. Berlin, Germany: Springer, Sep. 1996, pp. 76–91.
[5] G. Di Battista, E. Pietrosanti, R. Tamassia, and I. G. Tollis, "Automatic layout of PERT diagrams with X-PERT," in *Proc. IEEE Workshop Vis. Lang.*, Jan. 1989, pp. 171–172.
[6] D. L. Fisher and W. M. Goldstein, "Stochastic PERT networks as models of cognition: Derivation of the mean, variance, and distribution of reaction time using order-of-processing (OP) diagrams," *J. Math. Psychol.*, vol. 27, no. 2, pp. 121–151, Jun. 1983.
[7] C. Bachmaier, U. Brandes, and F. Schreiber, "Biological networks," in *Handbook of Graph Drawing and Visualization*, R. Tamassia, Ed. Boca Raton, FL, USA: CRC Press, 2014, pp. 621–651.
[8] A. M. Lipsky and S. Greenland, "Causal directed acyclic graphs," *JAMA*, vol. 327, no. 11, pp. 1083–1084, 2022.
[9] T. C. Williams, C. C. Bach, N. B. Matthiesen, T. B. Henriksen, and L. Gagliardi, "Directed acyclic graphs: A tool for causal studies in paediatrics," *Pediatric Res.*, vol. 84, no. 4, pp. 487–493, Oct. 2018.
[10] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical system structures," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 2, pp. 109–125, Feb. 1981.
[11] *Tom Sawyer Software—Perspectives*. [Online]. Available: https://www.tomsawyer.com/perspectives
[12] *yEd—Java Graph Editor*. [Online]. Available: http://www.yworks.com/en/products_yed_about.htm
[13] G. Ortali and I. G. Tollis, "Algorithms and bounds for drawing directed graphs," in *Proc. 26th Int. Symp. Graph Drawing Netw. Vis. (GD)*, Barcelona, Spain. Springer, Sep. 2018, pp. 579–592.
[14] G. Ortali and I. G. Tollis, "A new framework for hierarchical drawings," *J. Graph Algorithms Appl.*, vol. 23, no. 3, pp. 553–578, 2019.
[15] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel, "The open graph drawing framework (OGDF)," in *Handbook of Graph Drawing and Visualization*, vol. 2011, 2013, pp. 543–569.
[16] E. J. Hopcroft and M. R. Karp, "An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs," *SIAM J. Comput.*, vol. 2, no. 4, pp. 225–231, 1973.
[17] A. Kuosmanen, T. Paavilainen, T. Gagie, R. Chikhi, I. A. Tomescu, and V. Mäkinen, "Using minimum path cover to boost dynamic programming on dags: Co-linear chaining extended," in *Proc. 22nd Annu. Int. Conf. Res. Comput. Mol. Biol. (RECOMB)*, Paris, France, Apr. 2018, pp. 105–121.
[18] J. B. Orlin, "Max flows in O(nm) time, or better," in *Proc. Symp. Theory Comput. Conf. (STOC)*, Palo Alto, CA, USA, Jun. 2013, pp. 765–774.
[19] C. P. Schnorr, "An algorithm for transitive closure with linear expected time," *SIAM J. Comput.*, vol. 7, no. 2, pp. 127–133, May 1978.
[20] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo, "A technique for drawing directed graphs," *IEEE Trans. Softw. Eng.*, vol. 19, no. 3, pp. 214–230, Mar. 1993.
[21] G. Sander, "Layout of compound directed graphs," 1996.
[22] M. T. Goodrich and R. Tamassia, *Algorithm Design and Applications*, 1st ed. Hoboken, NJ, USA: Wiley, 2014.
[23] P. Lionakis, G. Ortali, and I. G. Tollis, "Constant-time reachability in DAGs using multidimensional dominance drawings," *Social Netw. Comput. Sci.*, vol. 2, no. 4, p. 320, Jul. 2021.
[24] H. Lam, E. Bertini, P. Isenberg, C. Plaisant, and S. Carpendale, "Empirical studies in information visualization: Seven scenarios," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 9, pp. 1520–1536, Sep. 2012.

[25] J. Heer, F. V. Ham, S. Carpendale, C. Weaver, and P. Isenberg, "Creation and collaboration: Engaging new audiences for information visualization," in *Information Visualization*. Berlin, Germany: Springer, 2008, pp. 92–133.
[26] C. Plaisant, "The challenge of information visualization evaluation," in *Proc. Work. Conf. Adv. Vis. Interfaces*, May 2004, pp. 109–116.
[27] E. Wall, M. Agnihotri, L. Matzen, K. Divis, M. Haass, A. Endert, and J. Stasko, "A heuristic approach to value-driven evaluation of visualizations," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 1, pp. 491–500, Jan. 2019.
[28] R. Amar, J. Eagan, and J. Stasko, "Low-level components of analytic activity in information visualization," in *Proc. IEEE Symp. Inf. Vis. (INFOVIS)*, Oct. 2005, pp. 111–117.
[29] E. D. Giacomo, W. Didimo, G. Liotta, F. Montecchiani, and I. G. Tollis, "Exploring complex drawings via edge stratification," in *Proc. Int. Symp. Graph Drawing*. Cham, Switzerland: Springer, 2013, pp. 304–315.
[30] M. Ghoniem, J.-D. Fekete, and P. Castagliola, "A comparison of the readability of graphs using node-link and matrix-based representations," in *Proc. IEEE Symp. Inf. Vis.*, Oct. 2004, pp. 17–24.
[31] H. C. Purchase, J. Hamer, M. Nöllenburg, and S. G. Kobourov, "On the usability of Lombardi graph drawings," in *Proc. Int. Symp. Graph Drawing*. Berlin, Germany: Springer, 2012, pp. 451–462.
[32] W. Didimo, E. M. Kornaropoulos, F. Montecchiani, and I. G. Tollis, "A visualization framework and user studies for overloaded orthogonal drawings," *Comput. Graph. Forum*, vol. 37, no. 1, pp. 288–300, Feb. 2018.
[33] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *J. Usability Stud.*, vol. 4, no. 3, pp. 114–123, 2009.

PANAGIOTIS LIONAKIS received the B.Sc., M.Sc., and Ph.D. degrees in computer science from the University of Crete, Greece, in 2013, 2017, and 2023, respectively. He was involved extensively as a Researcher and participated in a number of national and European projects with the Institute of Computer Science (ICS), Foundation for Research and Technology-Hellas (FORTH), University of Crete, and in research and development in software companies. His research is related to visualization and analysis of very large graphs.

GIORGOS KRITIKAKIS received the bachelor's and master's degrees in computer science from the University of Crete, in 2020 and 2022, respectively. In 2019, he was an Associate Researcher with the CARV Laboratory, Institute of Computer Science (ICS), FORTH, and he extended SCOOP source to source compiler. From 2020 to 2022, he was an Associate Researcher with the Graph Drawing and Information Visualization Laboratory, University of Crete, where he presented his work "Analysis and Visualization of Hierarchical Graphs." He accomplished his military service with the IT Department, Greek Army, in December 2022.

IOANNIS G. TOLLIS received the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, in January 1988. From 1988 to 2004, he was a Professor of computer science with The University of Texas at Dallas. From 2005 to 2010, he was with the Institute of Computer Science, FORTH (ICS-FORTH), where he was the Head of the Biomedical Informatics Laboratory (BMI Lab.). From 2012 to 2016, he was the Director of the Center for ICT, University of Crete (UoC), Greece, and the Data Processing Laboratory. He is currently a Professor of computer science with UoU. Furthermore, he has significant industrial experience and is interested in employing efficient graph technology to solve new challenges. He has two U.S. patents and several of his projects have been licensed by companies for commercial distribution. He has published eight books, and almost 200 journal and conference papers. His research interests include design, analysis, and visualization of networks, data analytics and security, graph drawing, information visualization, the modeling and visualization of biomedical data and networks, and algorithms and applications. His research has been funded by numerous funding agencies and companies. He has given more than 75 invited lectures worldwide.