GDGA 2013

Workshop on Graph Drawing and Graph Algorithms
Dhaka, Bangladesh, 2013

Proceedings

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

HEQEP

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)

Program Schedule

Day 1: January 26, 2013

08:30am - Rest of the day	:	**Registration**
09:00am - 09:45am	:	**Inauguration** (Chief Guest: Prof. Dr. M. Muhibur Rahman, Member, University Grant Commission, Bangladesh Special Guest: Prof. Dr. Subhas Chandra Nandy, ISI, Kolkata, India)
09:45am - 10:00am	:	**Tea Break**
10:00am - 12:00pm	:	**Invited Talk** (Professor Dr. Subhas Chandra Nandy)
12:00am - 01:00pm	:	**Open Problem Session** (Introduction to Open Problems)
01:00pm - 02:00pm	:	**Prayer and Lunch Break**
02:00pm - 04:00pm	:	**Invited Talk with open problem discussions** (Professor Dr. Md. Rezaul Karim)
04:00pm - 04:30pm	:	**Prayer and Tea Break**
04:30pm - 06:00pm	:	**Paper Presentations**

Md. Iqbal Hossain and Md. Saidur Rahman, *Straight Line Monotone Grid Drawings of Series-Parallel Graphs*

Naima Khan, Nazifa Karima, Md. Saidur Rahman, and Md. Iqbal Hossain. *Orthogonal Grid Pointset Embeddings of Maximal Outerplanar graphs*

Md. Sazzadur Rahaman, Tousif Ahmed Eshan, Sad Al Abdullah, and Md. Saidur Rahman. *Antibandwidth Problem for Itchy Caterpillar*

Md. Akter Hossain. *Applications of Shortest Route Algorithms*

Day 2: January 27, 2013

08:30am - 10:30am	:	**Invited Talk with open problem discussions** (Professor Dr. M. Kaykobad)
10:30am - 11:00am	:	**Tea Break**
11:00am - 01:00pm	:	**Invited Talk with open problem discussions** (Professor Dr. Masud Hasan)
01:00pm - 02:00pm	:	**Prayer and Lunch Break**
02:00pm - 03:30pm	:	**Paper Presentations**

Muhammad A. Adnan, Ifat Afrin, Masud Hasan, Tahmina Khanam, and Mir M. Al-Mahmud. *Zonohedra, Zone Graphs, and Their Linear Area Straight Line Grid Drawing*

Shaheena Sultana and Md. Saidur Rahman, Arpita Roy, and Suraiya Tairin. *Algorithm for Bar 1-Visibility Drawings of Graphs*

Aftab Hussain and Md. Saidur Rahman. *A New Clustering Technique using (k,w)-Core Decomposition for Restructuring Software Functions*

Kh. Md. Mominul Haque. *Irregular Total Labelling of Mobius Ladder*

03:30pm - 05:00pm	:	**Open Problem Brainstorming, Prayer, and Tea Break**
05:00pm - 05:30pm	:	**Presentation on Outcome of Open Problems**
05:30pm - 06:00pm	:	**Concluding Remarks**

Message from the Chief Guest

It is a great pleasure for me to welcome all the participants of the Workshop on Graph Drawing and Graph Algorithms (GDGA 2013), being organized by the Department of Computer Science and Engineering (CSE), Bangladesh University of Engineering and Technology (BUET) under the HEQEP subproject "CP # 2082: Enhancement of Graduate Studies Facilities for the Department of CSE, BUET".

To meet the challenges of globalization, it is essential to raise higher education quality to international standard. Accordingly, the Ministry of Education, Government of Bangladesh, has undertaken the Higher Education Quality Enhancement Project (HEQEP), under an agreement with the World Bank which will provide about 88% of the project cost as soft loan. The project aims at improving the quality of teaching-learning and research capabilities of the universities of the country through encouraging both innovation and accountability and by enhancing the technical and institutional capacity of the higher education sector. The University Grants Commission (UGC) of Bangladesh is the implementing agency of the project. A HEQEP unit has been established in the UGC for implementation, management, monitoring and evaluation of the activities.

Workshops on advanced topics can play an important role in enhancing the quality of higher education. As Bangladesh is now advancing towards a digital era, we need to establish an adequate ICT infrastructure as well as ensure a timely and useful implementation of ICT enabled applications and systems in every sector of our life: from private to public, from education to medical sector, from services to commerce, and so on. Conferences and workshops can provide us opportunities both for sharing and dissemination of knowledge and learning from others. GDGA 2013, I believe, will give the young researchers of Bangladesh a unique opportunity to come closer to the frontier of world-class computer research and enable their potentials to flourish.

I wish all the success for this workshop and sincerely hope that GDGA will serve as an international platform for presenting excellent research contributions in Graph Drawing and Graph Algorithms.

Finally I congratulate the Department of CSE, BUET, in particular Sub-Project Manager Professor Dr. Md. Saidur Rahman and his team, for organizing the workshop as part of the activities of the sub-project CP # 2082, and wish its successful implementation.

[illegible]

January 26, 2013

Prof. Dr. M. Muhibur Rahman
Member, University Grants Commission of Bangladesh

Message from the Head, CSE, BUET

The Department of Computer Science and Engineering(CSE) of BUET provides the highest quality of research. There are a large variety of research areas with several active groups utilizing attractive facilities. A number of highly-cited articles are being published from the areas such as graph drawing, artificial intelligence, network management, bioinformatics, computational biology, database etc. Students of this department are exhibiting research excellences through publication of a number of papers in reputed domestic and international venues.

To this extent Workshop on Graph Drawing and Graph Algorithms (GDGA 2013) is going to be one step forward towards the practice of ongoing research knowledge, sharing of ideas, and assembly of researchers around the globe. On the eve of this occasion, I am very glad to have the honor of hosting such a great event in this department. This workshop welcomes worldwide inter-disciplinary participation. Four distinguished experts in the field of Graph Drawing and Graph Algorithms from different parts of Bangladesh and abroad will give their valuable talks on this occasion. Besides, experts and interested individuals from different universities in Bangladesh and abroad are participating in this grand occasion. I hope that this event will act as a congregation of intellectuals and scientists in the field of graph drawing and graph algorithms. At the same time, it will provide a great opportunity to share valuable knowledge and recent ideas among the community. I would like to express my gratitude to Professor Dr. M. Muhibur Rahman, Member, University Grant Commission (UGC) for giving his kind consent for attending this event as the chief guest. I also heartily thank the World Bank and UGC for their cordial co-operation and financial support for arranging this workshop.

[illegible]

January 26, 2013

Dr. Abu Sayed Md. Latiful Hoque
Head
Department of CSE
Bangladesh University of Engineering and Technology

Message from Program Chair

I am very fortunate to observe how persistent efforts by Professor Md. Saidur Rahman have resulted in some standard structures of research this time through HEQEP project. Professor Rahman has been very determined in continuing with nontrivial research in the field of Graph Drawing and Graph Algorithms since his return from Japan. His initiatives resulted in a world class Workshop named "International Workshop on Algorithms and Computation" (WALCOM) that are being held in the soil of Bangladesh and India since 2007, and whose proceedings are being published by the famous Springer, and in special issues of reputed journals. This publication of WALCOM proceedings has opened new opportunities for Bangladeshi researchers to lift themselves up to the world standard in publishing proceedings of workshops and conferences held in Bangladesh. GDGA is opening a different culture of brainstorming over a set of unresolved problems for two days. I hope that this will create a lot of enthusiasm in our young researchers to be effective in getting results in a short time. Should it be successful, we should create more and more such opportunities before them. In line with WALCOM, GDGA opens up possibilities for our researchers in the field of graph drawing to do nontrivial research and present them in a knowledgeable forum so that further research and results in the field warrant output for the archival.

I thank GDGA committees whose hard work has resulted in organizing the workshop. Four invited papers along with eight papers will be enlightening our researchers. I take this opportunity to thank Professor Subhas C. Nandy to take the trouble of traveling to Bangladesh and help us strengthen the culture of research in the field. I also thank UGC, in particular Professor M. Muhibur Rahman for encouraging us through his presence in the workshop.

I have the best wishes for success of GDGA.

[illegible]

January 26, 2013

Prof. Dr. M. Kaykobad
Professor
Department of CSE
Bangladesh University of Engineering and Technology
&
Program Chair, GDGA 2013

Message from Organizing Chair

It is a great pleasure for me to welcome all the participants of Workshop on Graph Drawing and Graph Algorithms (GDGA 2013), being organized by Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET) under the HEQEP subproject "CP # 2082: Enhancement of Graduate Studies Facilities for Department of CSE, BUET". GDGA 2013 is providing an excellent opportunity for the academicians and researchers in this field to be familiar with the cutting edge research and to exchange their views.

For quality enhancement of graduate studies and research, people often give attention only on development and improvement of infrastructural facilities. But development of qualified academicians and researchers is very much essential as well. With this end in view, we need to create facilities for academicians and researchers to update their knowledge by interacting with the experts in their fields. Under this HEQEP subproject, a number of workshops on advanced topics of Computer Science and Engineering will be held in which senior undergraduate students, graduate students, faculties of CSE Dept. BUET as well as those of other universities of Bangladesh will get opportunity to enrich their knowledge in the respective fields. GDGA 2013 is the second workshop under this subproject, and the success of this workshop will be a step forward towards achieving the goal of the subproject.

I thank Prof. Dr. M. Muhibur Rahman, Honorable Member of University Grants Commission of Bangladesh for accepting our invitation to inaugurate GDGA 2013 as the chief guest. I also thank the invited speakers Prof. Dr. Subhas Nandy, Prof. Dr. M. Kaykobad, Prof. Dr. Md. Rezaul Karim and Prof. Dr. Masud Hasan for presenting their talks on recent research areas of graph drawing and graph algorithms. I would like to extend our thanks to the members of the program committee led by Prof. Dr. M. Kaykobad for their wonderful job and to the members of advisory committee for their support. Members of organizing committee worked hard for the success of the workshop; I also thank them. I am grateful to university authority and to Prof. Dr. A.S.M. Latiful Hoque, Head, Department of Computer Science and Engineering, BUET for their prompt organizational support.

I wish the workshop all the success and sincerely hope that GDGA 2013 will be a recognized platform for exchanging views of academicians and researchers in the field of graph drawing and graph algorithms.

Saidur

January 26, 2013

Dr. Md. Saidur Rahman
Professor, Dept. of CSE, BUET
Organizing Chair, GDGA 2013
&
Subproject Manager, HEQEP CP # 2082

Table of Contents

Invited Talks

Prune-and-Search for Geometric Optimization Problems in Read-only Set-up

Minati De[1], Subhas C. Nandy[1] and Sasanka Roy[2]
[1] Indian Statistical Institute, Kolkata, India
[2] Chennai Mathematical Institute, Chennai, India
e-mail: {minati_r, nandysc}@isical.ac.in, sasanka@cmi.ac.in

I. Introduction

Designing algorithmic tools with fast and limited size memory for fast processing of massive data is a challenging field of research [1], [2], [3]. The problem becomes much more difficult if the input data is given in a read-only array. In this paper, we show that the general *prune-and-search* technique can be implemented where the objects are given in read-only array. As examples we consider convex-hull and minimum enclosing circle problem in 2D.

Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of points in 2D, the problem of designing sub-quadratic time algorithm for computing convex hull and minimum enclosing circle for P with sub-linear extra work-space is an important problem and is being studied for a long time. Bronnimann et al. [5] showed that Graham's scan algorithm for computing convex hull of a planar point set of size n can be made in-place maintaining $O(n \log n)$ time complexity. Here, the extra workspace required is $O(1)$, and the output is available in the same input array. In the same paper they also showed that (i) the convex hull of a point set in 2D can be computed in an in-place manner in $O(n \log h)$ time and with $O(1)$ extra workspace where h is the number of hull vertices, and (ii) the linear programming in 2D with n constraints can be solved in $O(n)$ time. Very recently, Vahrenhold [12] showed that the prune-and-search algorithm by Kirkpatrick and Seidel [9] for computing the convex hull of a planar point set can also be made in-place maintaining the $O(n \log h)$ time complexity and using only $O(1)$ work-space. All these algorithms permute the input array after the execution. If a planar point set P is given in a read-only array, then the well-known Jarvis March algorithm computes the convex hull in $O(nh)$ time with $O(1)$ extra space. The problem of designing a sub-quadratic algorithm for computing convex hull in a read-only environment with sub-linear work-space is an open problem for a long time.

Recently, Asano et al. [1] proposed an $O(n^2)$ time and $O(1)$ extra-space algorithm for computing all the vertices of the farthest point Voronoi diagram of the point set P, where the members in P are given in a read-only array. Needless to say, the same time complexity holds for computing the minimum enclosing circle. In the same paper they mentioned the possibility of finding the minimum enclosing circle in sub-quadratic time as an open problem.

In this work, we first address the open problem related to the convex hull problem in 2D. We show that if the points in P are given in a read-only array then the convex hull can be computed in $O(n^{\frac{3}{2}+\epsilon})$ time and $O(n^{\frac{1}{2}})$ extra space. Next, we consider a restricted version of the convex hull problem, where the input points are given in sorted order of their x-coordinates. Here, we can apply *prune-and-search* technique of Kirkpatrick and Seidal [9] in the read-only set-up to compute the convex hull of P in $O(n^{1+\epsilon})$ time and $O(\log n)$ space, where ϵ is a constant satisfying $\sqrt{\frac{\log \log n}{\log n}} < \epsilon < 1$. This extensively uses the median finding algorithm in read-only set-up by Munro and Raman [10] that runs in $O(n^{1+\epsilon})$ time and $O(1)$ space. We also show that similar technique works for solving the minimum enclosing circle for the point-set in P are given in a read-only array. The worst case time and extra space requirement of the proposed algorithm are $O(n^{1+\epsilon})$ and $O(\log n)$ respectively, where $\sqrt{\frac{\log \log n}{\log n}} < \epsilon < 1$.

II. Convex hull for arbitrary point set

Given a set P of n points in 2D in a read-only array, let $CH(P)$ be the convex hull of P. We now describe the method of reporting the vertices of the upper hull of $CH(P)$; the lower-hull of $CH(P)$ can be computed in a similar manner. We use three arrays, namely A, B and C, each of size $O(\sqrt{n})$ as the work-space. We use $P_{(i)}$ to denote the set of points whose x-coordinate lies between $x_{(\lfloor i\sqrt{n}\rfloor+1)}$ to $x_{(\lfloor (i+1)\sqrt{n}\rfloor)}$, and P_i to denote the set of points whose x-coordinate is less than $x_{(\lfloor (i+1)\sqrt{n}\rfloor)}$, where $x_{(k)}$ denotes the k-th smallest x-coordinates among the points in P. The upper hull of $P_{(i)}$ and P_i are denoted by $CH_{(i)}$ and CH_i respectively. Our algorithm executes in two passes. Each pass consists of $\lceil\sqrt{n}\rceil$ stages. In the i-th stage of the first pass, we pick up the points in $P_{(i)}$ in the array A. We assume that $(i-1)$ stages are complete; the vertices of $CH_{(i-1)}$ in the convex hull CH_{i-1} are stored in the array B. The j-th element of the array C (denoted by $C[j]$) contains the first and last hull-vertices (f_j, ℓ_j) among the points in $P_{(j)}$ in the convex hull CH_{i-1}, $j \leq i-1$. If no such hull-vertex exists then $C[j]$ contains $(-1,-1)$. We first compute the upper hull $CH_{(i)}$ of the points in A using the in-place convex hull algorithm of [5]. Next, we merge $CH_{(i)}$ with CH_{i-1} to form CH_i. If the tangent of $CH_{(i-1)}$ and $CH_{(i)}$ touches $CH_{(i-1)}$ at its left-most vertex, then we recompute

$CH_{(i-2)}$, and consider its portion from f_{i-2} to ℓ_{j-2}. The procedure continues until we get a convex chain $CH_{(j)}$, $j < i$ whose at least two vertex lies in the convex chain CH_i. The first pass terminates after processing the points $P_{(\sqrt{n})}$. In the second pass, we compute $CH_{(i)}$ for all the blocks i such that $C[i] \neq (-1,-1)$, and report only the portion from f_i to ℓ_i.

Theorem 1: [7] Given a set P of n points in 2D in a read-only array, the convex-hull of P can be correctly computed in $O(n^{\frac{3}{2}+\epsilon})$ time using $O(n^{\frac{1}{2}})$ extra-space, where $\sqrt{\frac{\log\log n}{\log n}} < \epsilon < 1$.

III. Convex hull for sorted point set

Given a set P of n points in 2D sorted with respect to their x-coordinates in a read-only array, the objective is to report the edges of the convex hull of the points in P. We will show how Kirkpatrick and Seidel's [9] deterministic prune-and-search algorithm for computing convex hull can be implemented in this framework.

The algorithm in [9] computes upper-hull and lower-hull separately and report them. It follows follows divide-and-conquer paradigm, and uses a procedure for computing the bridge between the convex hull of two sets of points separated by a straight line using prune and search paradigm. The details of this procedure in the read-only environment is given in [7]. The following theorem summarizes the result obtained on this problem.

Theorem 2: Given a set of n sorted points P of 2D in a read-only array, the convex-hull of P can be computed in $O(n^{1+\epsilon})$ time using $O(\log n)$ extra-space, where $\sqrt{\frac{\log\log n}{\log n}} < \epsilon < 1$.

IV. Minimum enclosing circle

Let $P[0,\ldots,n-1]$ be an array containing the n input points. We first describe Megiddo's linear time algorithm for the MEC problem for the point set P. It is an iterative prune-and-search algorithm. Let π^* be the center of the desired MEC. Each iteration of this algorithm determines a pair of horizontal and vertical lines such that the quadrant in which π^* lies can be identified, and a constant fraction of points in P can be deleted.

The algorithm works as follows. If the number of non-deleted points is less than or equal to 3, it computes the minimum enclosing circle in $O(1)$ time. Otherwise it uses a pruning procedure that discards $\frac{n}{4}$ points and execute the next iteration on the remaining point set. The details of the implementation of Megiddo's algorithm when input is given in a read-only array is given in [6]. In each iteration it determines the pair of orthogonal lines as pruning criteria using the median finding algorithm of Munro and Raman in read-only set-up [10]. This needs $O(n^{1+\epsilon})$ time using $O(1)$ extra space, where $\sqrt{\frac{\log\log n}{\log n}} < \epsilon < 1$. Since it recursively prunes points and no mark bit is used for the deleted point set, we need to maintain a stack for storing the pruning criteria, which is a pair of orthogonal lines; it needs $O(1)$ space. Again, since in each iteration, a constant fraction of points are deleted, the number of iteration is $O(\log n)$ in the worst case. Thus, we have the following result:

Theorem 3: The minimum enclosing circle of a set of points in 2D given in a read-only array can be found in $O(n^{1+\epsilon}\log^4 n)$ time and $O(\log n)$ space, where ϵ is a positive constant less than 1.

References

[1] T. Asano, W. Mulzer, G. Rote, and Y. Wang, *Constant-work-space algorithms for geometric problems*, JoCG, vol. 2(1), pp. 4668, 2011.

[2] H. Blunck and J. Vahrenhold, *In-place algorithms for computing (layers of) maxima*, Algorithmica, vol. 57(1), pp. 121, 2010.

[3] P. Bose, A. Maheshwari, P. Morin, J. Morrison, M. H. M. Smid, and J. Vahrenhold, *Space-efficient geometric divide-and-conquer algorithms*, Comput. Geom., vol. 37(3), pp. 209227, 2007.

[4] L. Barba, M. Korman, S. Langerman, K. Sadakane, and R. I. Silveira, *Space-time trade-offs for stack-based algorithms*, CoRR, abs/1208.3663, 2012.

[5] H. Bronnimann, J. Iacono, J. Katajainen, P. Morin, J. Morrison, and G. T. Tousosaint, *In-place planar convex hull algorithms*, in Proc. LATIN, pp. 494507, 2002.

[6] M. De, S. C. Nandy, and S. Roy, *Minimum enclosing circle with few extra variables*, in Proc. FSTTCS, 2012.

[7] M. De, S. C. Nandy, and S. Roy, *Convex Hull and Linear Programming with few extra variables*, Tech. Report No. CoRR abs/1212.5353, 2012.

[8] T. M. Chan and E. Y. Chen, *Multi-pass geometric algorithms*, Discrete & Computational Geometry, vol. 37(1), pp. 79102, 2007.

[9] D. G. Kirkpatrick and R. Seidel, *The ultimate planar convex hull algorithm?*, SIAM J. Comput., vol. 15(1), pp. 287299, 1986.

[10] J. I. Munro and V. Raman, *Selection from read-only memory and sorting with minimum data movement*, Theor. Comput. Sci., vol. 165(2), pp. 311323, 1996.

[11] N. Megiddo, *Linear-time algorithms for linear programming in $\mathcal{R}^3$ and related problems*, SIAM J. Comput., vol. 12(4), pp. 759–776, 1983.

[12] J. Vahrenhold, *On the space efficiency of the ultimate planar convex hull algorithm*. in Proc. CCCG, pp. 131136, 2012.

Research on Graph Drawing Algorithms

Md. Rezaul Karim
Dept. of Computer Science and Engineering
University of Dhaka
Email: rkarim@cse.univdhaka.edu

***Abstract*—We present survey of different graph drawing algorithms and open problems in this field. There are different graph drawing styles - planar drawing, straight-line drawing, grid drawing, orthogonal drawing, convex drawing, Polyline drawing, rectangular drawing, box-rectangular drawing, visibility drawing, etc. A graph has an infinite number of drawings. Some drawings are better than others in conveying information on the graph. We concentrate our talk on area efficient drawing, minimum segment drawing, point set embedding. Our approach is to define the problem, provide existing results as well as interesting open problem of related field.**

I. Introduction

A graph is an abstract structure that is used to model information. Many real-world situations can conveniently be described by means of graphs. Graphs may be used to represent any information that can be modeled as objects and connections between those objects. Thus graph drawing addresses the problem of constructing geometric representations of conceptual structures that are modeled by graphs. For example, graphs are used to represent social network where each node represents an actor (generally a person or an object or an organization, etc.) and each edge represents a relationship or a communication between the actors. Other than social networks, automatic graph drawings have important applications to key computer technologies such as software engineering (data flow diagrams, subroutine-call graphs, program nesting trees), real-time systems (state-transition diagrams), artificial intelligence (knowledge-representation diagrams) etc. Further application can be found in other science and engineering disciplines, such as medical science (concept lattices), chemistry (molecular drawings), civil engineering (floorplan maps), cartography (map schematics).

There are different graph drawing styles. The drawing styles are - planar drawing, straight-line drawing, grid drawing, orthogonal drawing, convex drawing, Polyline drawing, rectangular drawing, box-rectangular drawing, visibility drawing, etc. In the different drawing styles, vertices are represented by small circles, or points, or rectangular boxes, or horizontal line segments; and the edges are represented by straight-line segments, or chain of horizontal and vertical line segments, or horizontal line segments, or vertical line segments etc.

A graph has an infinite number of drawings. Some drawings are better than others in conveying information on the graph. Various criteria have been proposed in the literature to evaluate the aesthetic quality of a drawing. They are - area, edge crossing, angular resolution, aspect ration, shape of faces, symmetry of drawing.

The important topics of the talk are given below.

A. Area Efficient Drawing

Area of a drawing means the area of the smallest rectangle that encloses the drawing. Smaller area of a drawing increases the readability of the drawing. Compact drawing of a circuit is preferable for VLSI fabrication since a compact drawing helps us to avoid wasting of valuable wafer space. In 1990, de Fraysseix *et al.* [1] and Schnyder [2] showed by two different methods that every planar graph of $n \geq 3$ vertices has a straight-line drawing on an integer grid of size $(2n-4) \times (n-2)$ and $(n-2) \times (n-2)$, respectively. A natural question arises: what is the minimum size of a grid required for a straight-line drawing? de Fraysseix *et al.* showed that, for each $n \geq 3$, there exits a plane graph of n vertices, for example nested triangles, which needs a grid size of at least $\lfloor 2(n-1)/3 \rfloor \times \lfloor 2(n-1)/3 \rfloor$ for any grid drawing [1], [3]. It has been conjectured that every plane graph of n vertices has a grid drawing on a $\lceil 2n/3 \rceil \times \lceil 2n/3 \rceil$ grid, but it is still an open problem. For some restricted classes of graphs, more compact straight-line grid drawings are known. For example, a 4-connected plane graph G having at least four vertices on the outer face has a straight-line grid drawing with area $(\lceil n/2 \rceil - 1) \times (\lfloor n/2 \rfloor)$ [4]. Garg and Rusu showed that an n-node binary tree has a planar straight-line grid drawing with area $O(n)$ [5], [6]. Although trees admit straight-line grid drawings with linear area, it is generally thought that triangulations may require a grid of quadratic size. Hence finding nontrivial classes of planar graphs of n vertices richer than trees that admit straight-line grid drawings with area $o(n^2)$ is posted as an open problem in [7]. Karim and Rahman [8] showed that Doughnut graph as well as its spanning subgraph admit straight-line grid drawing of linear area. Karim et.al [9] showed that a subclass of outerplanar graph admits straight-line grid drawing $O(n \log n)$

B. Minimum Segment Drawing

Minimum segment drawing uses the minimum number of line segments among all possible straight line drawings of the planar graph. Planar graphs with few slopes and segments presetned by Dujmovic et.al [10]. Durocher et.al. proved that it is NP-complete to determine whether a plane graph G has a straight-line drawing with at most k segments, where

$k \geq 3$. Samee et al. [11] developed an algorithm for computing minimum segment drawings of series-parallel graphs with the maximum degree three. Biswas et al. [12] gave an algorithm for minimum-segment convex drawings of 3-connected cubic plane graphs.

C. Point Set Embedding

A point-set embedding of a plane graph G with n vertices on a set S of n points is a straight-line drawing of G, where the vertices of G are mapped to distinct points of S. Cabello [13] proved that it is NP-complete in general to decide whether a given 2-connected plane graph admits a point-set embedding on a given set of points. Durocher and Mondal [14] proved that the problem remains NP-complete for 3-connected planar graphs. Nishat et al. [15] developed an algorithm for point set embedding of plane-3 trees.

REFERENCES

[1] J. P. H. de Fraysseix and R. Pollack, "How to draw a planar graph on a grid," *Communications of the ACM*, vol. 10, pp. 41–51, 1990.
[2] W. Schnyder, "Embedding planar graphs on the grid," in *Proc., First ACM-SIAM Symp. on Discrete Algorithms*, 1990, pp. 138–148.
[3] M. Chrobak and S. Nakano, "Minimum-width grid drawings of a plane graphs," *Comp. Geom. Theory and Appl.*, vol. 11, pp. 29–54, 1998.
[4] K. Miura, S. Nakano, and T. Nishizeki, "Grid drawings of four-connected planar graphs," *Discrete and Computational Geometry*, vol. 26, pp. 73–78, 2001.
[5] A. Garg and A. Rusu, "Straight-line drawings of binary trees with linear area and arbitrary aspect ration," in *Proc. of Graph Drawing 2002*, vol. 2528. LNCS, 2002, pp. 320–331.
[6] ——, "A more practical algorithm for drawing binary trees in linear area with arbitrary aspect ratio," in *Proc. of Graph Drawing 2003*, vol. 2912. LNCS, 2003, pp. 159–165.
[7] F. Brandenburg, D. Eppstein, M. T. Goodrich, S. Kobourov, G. Liotta, and P. Mutzel, "Selected open problems in graph drawing," in *Proc. of Graph Drawing 2003*, vol. 2912. LNCS, 2003, pp. 515–539.
[8] M. R. Karim and M. S. Rahman, "On a class of planar graphs with straight-line grid drawings on linear area," *Journal of Graph Algorithms and Applications*, vol. 13(2), pp. 153–177, 2009.
[9] M. R. Karim, M. J. Alam, and M. S. Rahman, "Straight-line grid drawings of label-constrained outerplanar graphs with $o(nlogn)$ area," *Journal of Graph Algorithms and Applications*, vol. 15(3), pp. 437–456, 2011.
[10] V. Dujmovic, D. Eppstein, M. Suderman, and D. R. Wood, "Drawings of planar graphs with few slopes and segments," *Computational Geometry: Theory and Application*, vol. 38(3), pp. 194–212, 2007.
[11] M. A. H. Samee, M. J. Alam, M. A. Adnan, and M. S. Rahman, "Minimum segment drawings of series-parallel graphs with the maximum degree three," in *Proc. of Graph Drawing 2008*, vol. 5417. LNCS, 2009, pp. 408–419.
[12] S. Biswas, D. Mondal, and R. I. Nishat, "Minimum-segment convex drawings of 3-connected cubic plane graphs," *Journal of Combinatorial Optimization*, pp. 1–21, 2011.
[13] S. Cabello, "Planar embeddability of the vertices of a graph using a fixed point set is np-hard," *Journal of GraphAlgorithms and Applications*, vol. 10(2), pp. 353–363, 2006.
[14] S. Duropcher and D. Mondal, "On the hardness of point-set embeddability," in *Proc. of WALCOM 2012*, vol. 7157. LNCS, 2012, pp. 148–159.
[15] R. I. Nishat, D. Mondal, and M. S. Rahman, "Point-set embeddings of plane 3-trees," in *Proc. of Graph Drawing 2010*, vol. 6502. LNCS, 2011, pp. 317–328.

Majority Spanning Trees, Cotrees and Their Applications

Mohammad Kaykobad
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
www.buet.ac.bd
Email: kaykobad@cse.buet.ac.bd

***Abstract*—We show that in any digraph on a connected underlying graph with non-negative weights on its edges, there is a so-called *Majority Spanning Tree* for which sum of weights of edges of a cutset running along the edge of the spanning tree determining the cutset is not less than sum of those running in opposite direction. Similarly, existence of *majority Cotrees* and simultaneous existence of majority spanning trees and majority cotrees have been established. We have shown how these structures are important in scheduling transports of equal frequencies by minimizing sum of weighted waiting times in connections, ranking round robin tournaments by minimizing number of upsets, in settling multiple debts and in construction of transport networks with unbalanced road capacity.**

I. Introduction

In this paper we consider directed graphs $G = (V, E)$ whose underlying graph is connected. Let $w_{ij} \geq 0 \; \forall (i, j) \in E$. That every connected graph has a spanning tree was an important discovery. Spanning trees have become a very important structure in solving problems of various fields like optimization, networks and clustering. Minimum spanning trees are very sought after structures for construction of which there are many algorithms like [1] and [2]. Let $T = (V, E')$ be a spanning tree of G. $T \setminus \{e\}$ divides V into two disjoint components V_1, V_2 such that $V_1 \cap V_2 = V, \; V_1 \cup V_2 = \emptyset$. Let

$$W(V_1, V_2) = \sum_{a_k=(l,m)\in K_k^+ | l \in V_1, m \in V_2} w_{lm}$$

$$\text{where} \quad a_k = (i, j) \in T$$

and

$$W(V_1, V_2) = W[V_1, V_2] - W[V_2, V_1]$$

$[V_1, V_2]$ is said to be a fundamental cutset determined by the edge $e \in T$. In the same way an edge $e \in \bar{T}$ together with T produces a unique cycle called fundamental cycle.

A spanning tree T of G is said to be a mojority spanning tree if

$$\forall (i, j) \in T, \quad i \in V_1, \quad j \in V_2, \; W[V_1, V_2] \geq 0$$

. In the same manner we define majority cotree. $\bar{T} = G \setminus T$ is called a cotree when T is a spanning tree of G. Each edge $a_k = (i, j) \in \bar{T}$ together with edges of T includes a unique cycle C, not necessarily directed whose orientation is in the orientation of edge a_k in cycle C. Let us define weight of a cycle as follows:

$$W(C) = \sum_{(i,j)\in C_k^+} w_{ij} - \sum_{(i,j)\in C_k^-} w_{ij}$$

where C_k^+ are the edges in C that are in the same orientation as $a_k = (i, j) \in \bar{T}$

A cotree $\bar{T}$ is said to be a majority cotree if

$$\forall a_k \in \bar{T}, \; W(C_k) \geq 0$$

It is not very clear that every directed graph with non-negative weights on its edge must have a majority spanning tree or majority cotree. However, in the next section, we shall answer this question affirmatively. Not only that we will also show that for every directed graph $G = (V, E)$ there is a majority spanning tree T whose complement $\bar{T} = G \setminus T$ is also a majority cotree.

II. Main Results

Before formulating the main theorems and their proof let us note down the following important observations. Detailed proofs of the following results can be found in Kaykobad [3].

Observation 1: Two edges that are in the same orientation in a cutset are in different orientation in the cycle containing them and vice versa.

Observation 2: If we subtract minimum edge weight of all edges of a cutset in that orientation, and add that value to edges in the opposite orientation, then weight of cycles do not change. Similarly, if a positive weight is subtracted from weights of edges in a cycle i in one orientation and the same figure is added to edges in opposite orientation then the weigths of cutrsets will remain unchanged.
The above two results allow us to manipulated edge weights keeping weights of cycles fixed while changing along a cutset or weights of cutsets fixed while changing along a cycle. In the following we shall use this technique to construct majority spanning trees and majority cotrees. We shall start proving the following theorem by construction.

Theorem 1: Every digraph $G = (V, E)$ with non-negative weights on its edges must have a majority spanning tree.

Proof. So long as there is a cycle with positive weights on all its edges, subtract the minimum weight of an edge from edges in the same orientation in the cycle, and add the same weigth to all edges in the opposite orientation. Then at least one edge will be of weight 0. In this way so long as there arre positive weighted cycles carry out the same process in that cycle. Ultimately positive weighted edges and some edges of 0 weight will constitute a spanning tree T that must be a majority spanning tree. In this process, no cutset weight has been changed. Now in each fundamental cutset only weight of the edge of spanning tree determining the fundamental cutset is positive(non-negative) other weights being 0. Hence weight of the correpsonding fundamental cutset is at least non-negative.

Theorem 2: Every digraph $G = (V, E)$ with non-negative weights on its edges must have a majority spanning tree.
Proof: Proof goes in the same way. Only here weights are changed along cutsets keeping weights of cycles unaffected. Then edges of positive weights together with some edges of 0 weights constitute a cotree $\bar{T}$ which is a majority cotree, and weight of each fundamental cycle equals the weight of the edge of the cotree determining the fundamental cotree. Now we have the following important result.

Theorem 3: Every digraph $G = (V, E)$ with non-negative weights on its edges has a majority spanning tree T such that $\bar{T}$ is a majority cotree, and vice versa.

In order to prove the above theorem let us first formulate the following linear programming problem in which each equation corresponds to a fundamental cycle of a tree with coeficient +1 for those edges that appear in the positive orientation of the cycle (orientation of the edge of the cotree that determines the cycle) multiplied by variable y_{ij} corresponding to the edge (i, j) For edges appearing in opposite orientation coefficients are -1, and for edges that do not appear in the fundamental cycle coefficients are 0s. Right hand side equals weight of the cycle. Remember, variables corresponding to edges of the cotree form a basic solution since they do not appear in other equations/fundamental cycles. In objective function variables corresponding to edges of the tree T will appear with coefficients equal to weight of the fundamental cutset detrmined by the edge of T. So linear porgramming peoblem will look like

$$\begin{aligned} \min\ r_T^t y_T \\ Ly = s \\ y \geq 0 \end{aligned} \tag{1}$$

where r_T is a vector whose kth component equals weight of the fundamental cutset determined by the edge $a_k = (i, j) \in T$, s is yet another vector whose kth compoent equals weight of the edge $a_k \in \bar{T}$. That such a system has non-negative solution is evident from the fact that $y_{ij} = w_{ij}$ is a non-negative solution of the system. Any basic feasible solution of the system will correspond to a majority cotree. However, if we formulate the dual of this problem and remember that $-L_T^t = Q_{\bar{T}}$ we get that basic feasible solution of dual problem is a majority spanning tree. So optimal solution to the primal problem simultaneously determines a majority cotree (corresponding to basic variables), and edges correpsonding to non-basic variables constitute a majority spanning tree.

This proves the theorem on simultaneous existence of majority spanning trees and majority cotrees in any digraph with non-negatives weights on its edges.

III. Applications

The structures introduced in previous sections appear to be useful in a variety of unrelated applications. Let us briefly mention them below. Detailed applications have been shown in Kaykobad [3], Datta, Hossain and M Kaykobad et al [4] and Kaykobad et al. [5].

1. Minimum Connection Time Problem: In a transportation system where each trip is generated with equal frequency since most often each pair of destination is not served by a single transport; there is waiting time in connection. Australia, being a vast largely uninhabited continent, where capital cities are very far it is not possible to have direct rail connections between every pair of state capitals. The problem of scheduling trains arises so that total weighted waiting time in connections is minimized. If trips are denoted by vertices and connections by edges it can be shown that optimal scheduling will have 0 waiting times in connections that constitute a majority spanning tree of the digraph.

2. Ranking of round robin tournament: If we want to rank players of round robin tournament by minimizing number of upsets then optimal solution represented by placing nodes corresponding to successively ranked players together with edges from a player directed towards the player ranked immediately after will correspond to a directed hamiltonian path. However, this can be shown that optimal solution will correspond to a majority spanning tree. Not only that optimal solutions corresponding to a suset of consecutively ranked players will be a majority spanning tree on the subdigraph induced by vertices corresponding to those players.

3. Balancing both way roads: Imagine that for a changed traffic load the road system is not balanced. We are given extra traffic on some sets of road segments and cost of constructing them. Now optimal sets of road segments for costruction will correspond to a majority spanning tre of the corresponding digraph. 4. Settling multiple debts: If there is a set of borrowers and loan givers and we want to settle debts in an optimal way then then settling transactions will correspond to directed edge from borrower to loan giver with edge weight equal to the amount borrowed. Optimal solution will again correspond to a majority spanning tree in a related digraph.

IV. Conclusion

The same structure of majority spanning tres and cotrees have been found useful in solution of totally unrelated problems. Possibly further investigation will lead to discovery of other properties of these structures.

Acknowledgment

I would like to thank my supervisor Dr FJM Salzborn for introducing me to the beautiful area of graph theory that I wholeheartedly disliked in my earlier life.

References

[1] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem,” vol. 7, pp. 48–50, 1956.

[2] R. Prim, “Shortest connection networks and some generalizations,” *Bell System Technical Journal*, vol. 36, pp. 1389–1401, 1957.

[3] M. Kaykobad, “Minimum connection time and some related complexity problems,” *Unpublished PhD Thesis, The Flinders University of South Australia*, 1986.

[4] A. Datta, M. Hossain, and M. Kaykobad, “A new algorithm for ranking players of a round-robin tournament,” *International Journal of Computers and Operations Research*, vol. 22, pp. 221–226, 1995.

[5] M. Kaykobad, Q. N. U. Ahmed, A. S. Khalid, and R. Bakhtiar, “A modified algorithm for ranking players of a round-robin tournament,” *International Journal of Computer Mathematics*, vol. 85, pp. 1–7, 2007.

Future Computing and The Steiner Tree Problem

Masud Hasan, Nagib Meshkat, Md.Masudur Rahman
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Dhaka-1000, Bangladesh
masudhasan@cse.buet.ac.bd, diko007@gmail.com, masud.12cse@yahoo.com

January 13, 2013

Abstract

The class of NP-complete problems contain many useful problems in computer science and mathematics, such as the traveling salesman problem, Hamiltonian cycle, Steiner tree, etc. No polynomial time solutions are known to solve these problems in today's computers. In this talk we shall see what other alternatives scientists are thinking of to solve these problems as well as to bypass the impasse of NP-completeness. One of the alternatives is to solve the Steiner tree problem by soap bubble experiment. In this talk we shall physically demonstrate this experiment and discuss the challenges to realize it as a solution of NP-completeness.

Contributed Talks

Straight-Line Monotone Grid Drawings of Series-Parallel Graphs

Md.Iqbal Hossain, Md.Saidur Rahman
Graph Drawing and Information Visualization Laboratory,
Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology
Email: {mdiqbalhossain,saidurrahman}@cse.buet.ac.bd

***Abstract*—A monotone drawing of a graph G is a straight-line drawing of G where a monotone path exists between every pair of vertices of G in some direction. In this paper we study monotone drawings of series-parallel graphs in a variable embedding setting. We show that a series-parallel graph of n-vertices has a straight-line planar monotone drawing on a grid of size $O(n) \times O(n^2)$.**

I. Introduction and Preliminaries

A path P in a straight-line drawing of a graph is *monotone* if there exists a line l such that the orthogonal projections of the vertices of P on l appear along l in the order induced by P. A straight-line drawing of a graph is *monotone* if it contains at least one monotone path for each pair of vertices. Recently monotone related drawings of graphs have been discovered as a new standard for visualizing graphs [1], [2], [3], [4].

In this paper, we investigate this problem for a non-trivial subclass of planar graphs called series-parallel graphs. We show that every series-parallel graph admits a straight-line monotone drawing on $O(n) \times O(n^2)$ grid which can be computed in $O(n \log n)$ time.

Let p be a point in the plane and l a half-line starting at p. The slope of l, denoted by $slope(l)$, is the angle spanned by a counter-clockwise rotation that brings a horizontal half-line starting at p and directed towards increasing x- coordinates to coincide with l. Let Γ be a drawing of a graph G and let (u, v) be an edge of G. The half- line starting at u and passing through v, denoted by $d(u, v)$, is the *direction of* (u, v). The direction of an edge e is denoted by $d(e)$ and slope of e is denoted by $slope(e)$.

II. Monotone Grid Drawing

In this section we give an $O(n \log n)$ time algorithm to find a straight-line planar monotone grid drawing of a series-parallel graph on an $O(n) \times O(n^2)$ grid. If the input series-parallel graph G is not biconnected then by definition of series-parallel graph G is a connected graph. Furthermore, G has a single source and a single sink. In this case we add an edge between the source and the sink to make it biconnected. Let $\mathcal{T}$ be the SPQ-tree [5] of G with respect to edge st. So the root of $\mathcal{T}$ is a Q-node r and the only child of r is a P-node x. Now we re-rooted $\mathcal{T}$ at x. Let $\mathcal{T}'$ be an ordered SPQ-tree obtained from $\mathcal{T}$ as follows.

We traverse each P-node of $\mathcal{T}$; if any Q-node exists as child of a P-node we put the Q-node as the leftmost child of the P-node. The rest of the children of the P-node are S-nodes and we draw them from left to right according to increasing order of the number of vertices in the subtree rooted at the respective S-node.

We now assign a slope to each node of $\mathcal{T}'$. Let $1/1, 2/1, \ldots, n/1$ be n slopes in increasing order. Initially we assign the slope $1/1$ to the root of $\mathcal{T}'$. We then traverse $\mathcal{T}'$ to assign a slope to each node x of $\mathcal{T}'$. We first consider the case where x is a P-node. Let the slope assigned to x be $\mu/1$, and let $x_1, x_2, \ldots, x_k$ $(k < n)$ be the children of x in left to right order. We assign the slope $\mu/1$ to the leftmost child x_1. We next assign the slope $(\mu_{i_{max}} + 1)/1$ to x_{i+1} where $\mu_{i_{max}}/1$ is the largest slope assigned in the subtree rooted at x_i. We now consider the case where x is an S-node. Let the slope assigned to x be $\mu/1$, and let $x_1, x_2, \ldots, x_k$ $(k < n)$ be the children of x in left to right order. We assign the same slope $\mu/1$ to x_i $(i \leq k)$. Thus the largest slope assigned to a vertex can be at most $(n-1)/1$. We are now ready to draw G on a grid using the slope assigned to the each node of $\mathcal{T}'$.

We first give a drawing algorithm for a P-node. Let x be a P-node with slope $\mu/1$ assigned to it and let s and t be the source and the sink of x, respectively. Let $x_1, x_2, \ldots, x_k$ be the children of x in left to right order. Let $\mu_1/1, \mu_2/1, \ldots, \mu_k/1$ be the slopes assigned to $x_1, x_2, \ldots, x_k$, respectively.

If x is not the root of $\mathcal{T}'$, let x' be the parent node of x, and let x'' be the parent node of x'. Clearly x' is an S-node and x'' is a P-node. Let $\mu/1$ and $\mu'/1$ be the slope assigned to x' and x'', respectively. Let s'' and t'' be the source and the sink of x'', respectively.

We denote the position of a vertex u by $p(u)$; $p(u)$ is expressed by its x- and y-coordinates as $(p_x(u), p_y(u))$ on a grid. Let $p(x)$ be a point on the grid. We place the source vertex of the P-node x on $p(x)$. Let A_x be a set of points where the neighbors of t in $pert(x)$ are to be drawn. We now have the following two cases to consider.

Case 1: $t \neq t''$. We place the source vertex s on $p(x)$. If x_1 is a Q-node we leave it for now, and we draw the respective edge after placing the sink t of x. We add the $p(x)$ to A_x. Otherwise all x_i are S-nodes and we follow the drawing algorithm of S-node to draw each x_i.

After drawing all x_i, we elongate the l_1 (the equation of l_1 is $y = \mu_1 \times x + p_x(x)$) line up to the point $p(x_{end}) = (p_x(x) + n(x), p_y(x) + \mu_1 \times n(x))$ and place the sink t of x on $p(x_{end})$. Since $n(x_1) \leq n(x_2) \leq n(x_3) \ldots \leq n(x_k)$ and the slope $\mu_1/1 < \mu_2/1 < \ldots < \mu_k/1, p(x_{i_{end}})$ is visible from $p(x_{end})$. We connect t to all points in A_x using straight line segments and we call each of these edges a *P-node closing edge*. Note that the slope of edge $(p(x_{i_{end}}),\ p(x_{end}))$ holds $\pi/2 > slope(p(x_{i_{end}}), p(x_{end})) > -\pi/2$.

Case 2: $t = t''$. Let $p(Y'')$ be the largest x-coordinate used in the drawing of x''. If $p_x(x) < p_x(Y'')$, we set $p(x) = (p_x(Y''), \frac{p_x(Y'')-p_x(x'')}{\mu})$. We then place the s on $p(x)$. We now draw all the S-nodes according to the S-node drawing algorithm. Since the sink vertex of x and x'' are same, we do not draw t in this step. All the end vertices of x_i will be connected at the drawing of the sink of x''. If x_1 is Q-node, we add the $p(x)$ in A_x.

We now describe an algorithm for drawing an S-node. Let x be an S-node of $\mathcal{T}'$ with the assigned slope $\mu/1$. Let x' be the parent node of x and let s' be the source vertex of x'. Let $p(x')$ be the point where s' has already been placed. Clearly, x' is a P-node. Let l be a straight-line such that the equation of l is $y = \mu/1 \times x + p_x(x')$. Assume first that all the children of x are Q-nodes. Then the $pert(x)$ is a path. In this case we place corresponding vertex of $pert(x)$ on the line l and sequentially on the integer point. The last vertex of $pert(x_i)$ lies on the point $p_{end}(x) = ((p_x(x) + n(x),\ p_y(x) + \mu/1 \times n(x))$ $(slope(l) = \mu/1)$. Then we add the $p_{end}(x)$ in $A_{x'}$. Assume now that some of the children of x are P-node. We traverse left to right subtrees of x. If x_i is a Q-node, we place corresponding vertices on line the l. If x_i is a P-node, we set $p(x_i) = ((p_x(x) + i,\ p_y(x) + \mu/1 \times i)$ when the source vertex of x_i is not s' otherwise $p(x_i) = p(x')$. Then we use the drawing algorithm for P-nodes. We call the algorithm described above Algorithm *Monotone-Draw*. We now have the following theorem.

Theorem 1. *Algorithm Monotone-Draw finds a monotone drawing of a series-parallel graph on a grid of size $O(n) \times O(n^2)$ in $O(n \log n)$ time.*

Proof: Let Γ be the drawing of G constructed by Algorithm Monotone-Draw. We now show that Γ is a monotone drawing of G. To prove a such claim, we show that, a monotone path exists between every pair of vertices in Γ.

Let s and t be the source and the sink of G in Γ. In fact we will prove that a monotone path exists every pair of vertices of G in the drawing of $G \setminus (s, t)$. Let v be a vertex in Γ such that $v \notin \{s, t\}$. We first show that the path $P(v, s)$ and $P(v, t)$ are monotone. One can easily observe that a path $P(v, s)$ exists such that no P-node closing edge contain in $P(v, s)$ and each edge $e \in P(v, s)$ holds $5\pi/4 \geq slope(e) \geq 3\pi/2$. So the path $P(v, s)$ is monotone because $range(P(v, s)) < \pi$. On the other hand a path $P(v, t)$ exists such that $s \notin P(v, t); (s, t) \notin P(v, t)$ and some P-node closing edge might exist in $P(v, t)$. The path $P(v, t)$ is monotone because each edge $e \in P(v, t)$ holds $\pi/2 < slope(e) < -\pi/2$.

We now show that every two vertices $u, v \in G$ $(v \notin \{s, t\}$, $u \notin \{s, t\})$, a monotone path between u and v exists in Γ. Let $e_1 = (u, u')$ and $e_2 = (v, v')$ be the two edges lie on the path $P(u, s)$ and $P(v, s)$, respectively where no P-node closing edge exist in $P(u, s)$ and $P(v, s)$.

Let M and N be the two Q-nodes in $\mathcal{T}'$ such that $(u, u') \in pert(M)$ and $(v, v') \in pert(N)$, and let W be the lowest common ancestor of M and N in $\mathcal{T}'$. Let U and V be the children of W and ancestors of M and N, respectively. Let μ_W, μ_U, μ_V, μ_M, μ_N be the slopes assigned to the nodes W, U, V, M and N, respectively. Since e_1 and e_2 are not P-node closing edge, the slopes of $d(e_1)$ and $d(e_2)$ are $-\mu_M$ and $-\mu_N$, respectively. Now we consider the following two cases.

Case 1: W is a P-node. Without loss of generality we may consider $\mu_M > \mu_N$. So according to the slope assignment $\mu_M \geq \mu_U > \mu_V \geq \mu_N$. Let w and w' be the source and sink vertices of W in Γ. The path $P(u, v)$ $(w' \notin P(u, v))$ is composed of path $P(u, w)$ and of path $P(w, v)$. Clearly, each edge $e \in P(u, w)$, and $e' \in P(w, v)$ hold $\mu_M \geq slope(e) \geq \mu_U$ and $\mu_N \geq slope(e') \geq \mu_V$, respectively. So we have $range(P(u, w)) \cap opp(P(w, v)) = \emptyset$. Then by [1] (Lemma 1,) $P(u, v)$ is a monotone path.

Case 2: W is an S-node. If M and N are children of the same S-node then the case is straight-forward, the path $P(u, v)$ lies on a straight-line. Otherwise the path $P(u, v)$ could have some P-node closing edge. let W' be the parent of W. Clearly W' is a P-node. Let w' be the source vertex of W' in Γ. Then the path $P(u, v)(w' \notin P(u, v))$ is monotone with respect to a horizontal half-line.

Thus we prove Γ is a monotone drawing of G.

We are placing the sink of each P-node on the $x = n(x)$ line and y-coordinate can be $O(n^2)$. Thus the total grid size will be $O(n) \times O(n^2)$. We construct SPQ-tree on linear-time and $O(n \log n)$ time is required to sort. Thus the overall time complexity of the algorithm is $O(n \log n)$. ■

References

[1] P. Angelini, E. Colasante, G. Di Battista, F. Frati, and M. Patrignani, “Monotone drawings of graphs,” *Journal of Graph Algorithms and Applications*, vol. 16, no. 1, pp. 5–35, 2012.

[2] P. Angelini, W. Didimo, S. Kobourov, T. Mchedlidze, V. Roselli, A. Symvonis, and S. Wismath, “Monotone drawings of graphs with fixed embedding,” in *Proceedings of the 19th international conference on Graph Drawing*, ser. GD’11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 379–390.

[3] E. M. Arkin, R. Connelly, and J. S. Mitchell, “On monotone paths among obstacles with applications to planning assemblies,” in *Proceedings of the fifth annual symposium on Computational geometry*, ser. SCG ’89. New York, NY, USA: ACM, 1989, pp. 334–343.

[4] M. A. H. Samee and M. S. Rahman, “Upward planar drawings of series-parallel digraphs with maximum degree three,” in *WALCOM’07*, 2007, pp. 28–45.

[5] A. Garg and G. Liotta, “Almost bend-optimal planar orthogonal drawings of biconnected degree-3 planar graphs in quadratic time,” in *Graph Drawing*, ser. Lecture Notes in Computer Science, J. Kratochvyl, Ed. Springer Berlin / Heidelberg, 1999, vol. 1731, pp. 38–48.

Orthogonal Grid Pointset Embeddings of Maximal Outerplanar Graphs

Naima Khan, Nazifa Karima, Md. Saidur Rahman, Md. Iqbal Hossain
Graph Drawing and Information Visualization Laboratory,
Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology
Email: {naimakhan076,nazifakarima}@gmail.com {saidurrahman,mdiqbalhossain}@cse.buet.ac.bd

***Abstract*—Let G be a planar graph with n vertices, and let S be a set of n prescribed grid points on a grid. An orthogonal grid pointset embedding of G on S is an orthogonal drawing of G such that each vertex of G is drawn as a point in S and each edge is drawn as a sequence of alternate horizontal and vertical line segments without edge crossing. In this paper we give linear-time algorithms for finding three variants of orthogonal pointset embeddings of a maximal outerplanar graph of maximum degree 4. We also give a linear-time algorithm to determine whether an outerplanar graph can be triangulated to a maximal outerplanar graph of the maximum degree 4 and find such a triangulation if it exists.**

I. Introduction

An *orthogonal drawing* of a planar graph G is a drawing of G where each vertex is drawn as a point and each edge is drawn as a chain of horizontal and vertical line segments. Orthogonal drawings of planar graphs is a well-studied area in graph drawing literature [1], [2], [3], [4], [5], [6]. We study orthogonal grid pointset embeddings of maximal outerplanar graphs and find the following results: Let G be a maximal outerplanar graph of the maximum degree 4. a) We show that G has an orthogonal grid pointset embedding with two bends per edge on a 2-spaced pointset. b) We show that G has an orthogeodesic pointset embedding with at most two bends per edge on a diagonal pointset and axis parallel pointset. c) We give a linear-time algorithm for checking if an outerplanar graph can be triangulated to a maximal outerplanar graph of the maximum degree 4 by adding edges.

II. Preliminaries

A graph is *planar* if it can be embedded in the plane so that no two edges intersect geometrically except at a vertex to which they are both incident. A *plane* graph is a planar graph with a fixed embedding in the plane. A plane graph divides the plane into connected regions called *faces*. The unbounded region is called the *outer face* and each bounded region is called an *inner face*. A maximal outerplanar graph of six vertices shown in Fig. 1(b) is addressed as *outerplanar octahedron* throughout this paper.

Let S be a pointset on a grid and let (x_{p_i}, y_{p_i}) be the coordinate of point p_i. A 2-*spaced pointset* is such a set S of points on a grid, where the minimum difference of

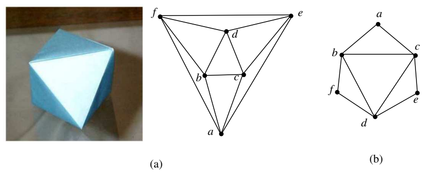

Fig. 1. (a) Octahedron and (b) outerplanar octahedron.

x-coordinates and y-coordinates of any two points p_i and p_j in S, is greater than or equal to 2. A set of points all having equal value for either x-coordinates or y-coordinates is called an *axis-parallel pointset*. A *diagonal pointset* is a 2-spaced pointset where y coordinates increases or decreases monotonically along the x-axis.

The following properties of a maximal outerplanar graph are known.

Lemma 1. *[7] Let G be a maximal outerplanar graph of $n(\geq 3)$ vertices. Then the following $(i) - (iv)$ hold. (i) G has $2n-3$ edges; (ii) G has $n-3$ inner edges; (iii) G has $n-2$ inner faces and each inner face is a triangle and, (iv) G has at least two vertices of degree 2.*

Lemma 2. *[8] Let n_2 be the number of vertices of degree 2 and t be the number of internal triangles in a maximal outerplanar graph with n vertices. If $n \geq 4$, then $t = n_2 - 2$.*

We now have the lemma on the following properties of a maximal outerplanar graph with the maximum degree 4 whose proof is omitted.

Lemma 3. *Let G be a maximal outerplanar graph of three or more vertices with the maximum degree 4. Assume that G is not an outerplanar octahedron. Then the following (a)-(d) hold. (a) G has no internal triangle. (b) The inner dual of G is a path. (c) If G has four or more vertices, then G has exactly two pair of consecutive vertices x and y such that degree of x is 2 and degree of y is 3, and G has exactly $n-4$ vertices of degree 4. (d) If G has more than four vertices, then every inner edge of G is incident to a vertex of degree 4 in G.*

The outerplanar octahedron is an exception to the three properties in Lemma 3(a)-(c) above. Furthermore the following lemma holds whose proof is omitted.

III. Orthogonal Grid Pointset Embeddings

In this section we give a linear-time algorithm to embed a maximal outerplanar graph G of the maximum degree 4 on a given 2-spaced pointset in Theorem 1. We then show that G has an othogeodesic pointset embedding with at most two bend per edge on a diagonal pointset and axis parallel pointset in Theorem 3 and Theorem 3, respectively.

Theorem 1. *Let G be a maximal outerplanar graph of the maximum degree 4 except an outerplanar octahedron. Then an orthogonal grid pointset embedding of G with two bends per edge on a 2-spaced pointset can be found in linear time, where the drawing of G is planar but not outerplanar.*

Proof: Let G be a maximal outerplanar graph of n vertices with $\Delta(G) = 4$. Let S be a 2-spaced pointset of n points and $p_1, p_2, p_3, \cdots, p_n$ be the points in S sorted according to x-coordinate where p_1 has the smallest x-coordinate and p_n has the largest x-coordinate. Let the coordinate of p_i be (x_{p_i}, y_{p_i}).

We assume that $n \geq 4$. According to Lemma 3(c), G has a pair of adjacent vertices x and y such that the degree of x is 2 and the degree of y is 3. Let $w_1 = x, w_2, \cdots, w_n = y$ be the vertices on the outer cycle of G in this order. We now draw vertices $w_1 = x, w_2, \cdots, w_n = y$ on the points $p_1, p_2, p_3, \cdots, p_n$, respectively. We first draw each edge (w_i, w_{i+1}) where $1 \leq i < n$ by three line segments $L_{i_1}, L_{i_2}, L_{i_3}$ where L_{i_1} is a horizontal line segment from (x_{p_i}, y_{p_i}) to $(x_{p_i} + 1, y_{p_i})$, L_{i_2} is a vertical line segment from $(x_{p_i} + 1, y_{p_i})$ to $(x_{p_i} + 1, y_{p_{i+1}})$, L_{i_3} is a horizontal line segment from $(x_{p_i} + 1, y_{p_{i+1}})$ to $(x_{p_{i+1}}, y_{p_{i+1}})$ as the edges $(a, h), (h, g), (g, f), (f, e), (e, d), (d, c), (c, b)$. Let the highest and lowest y-coordinate of the points $p_1, p_2, p_3, \cdots, p_n$ in S be y_{max} and y_{min} respectively. We then connect w_1 to w_n using three line segments L_{n_1}, L_{n_2} and L_{n_3} where, L_{n_1} is a vertical segment from (x_{p_n}, y_{p_n}) to $(x_{p_n}, y_{max} + \lfloor (n-1)/2 \rfloor)$, L_{n_2} is a horizontal segment from $(x_{p_n}, y_{max} + \lfloor (n-1)/2 \rfloor)$ to $(x_{p_1}, y_{max} + \lfloor (n-1)/2 \rfloor)$ and L_{n_3} is another vertical segment from $(x_{p_1}, y_{max} + \lfloor (n-1)/2 \rfloor)$ to (x_{p_1}, y_{p_1}) as the edge (a, b).

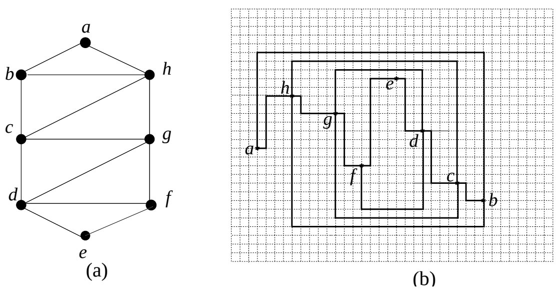

Fig. 2. Illustration for orthogonal pointset embedding on a 2-spaced pointset.

Then we draw the inner edges as illustrated in the Figure 2(b). Clearly each edge receives at most two bends, and the drawing can be found in linear time. ■

Theorem 2. *Every maximal outerplanar graph of the maximum degree 4, except an outerplanar octahedron, has an orthogeodesic pointset embedding with atmost two bends per edge, on a diagonal pointset which preserves the outerplanar embedding. Furthermore, the embedding can be found in linear time.*

Theorem 3. *Every maximal outerplanar graph of the maximum degree 4 except outerplanar octahedron, has an orthogonal grid pointset embedding with at most two bends per edge on an axis parallel pointset.*

The proofs of Theorems 2 and 3 are omitted in this short version.

IV. Triangulating an Outerplanar Graph

We know the inner dual of G is a path. Let $P = f_1, f_2, \cdots, f_k$, be the inner dual of G. Since P is a path, degree of f_1 and f_k is 1. Let F_i be the face of G corresponding to f_i of P. Let (a, b) be the common edge of face F_1 and F_2 and (u, v) be the common edge of face F_{k-1} and F_k such that a, u, v, b appear on the outer face of G in clockwise order. Let $X = \{x_0 = a, x_1, \cdots, x_{p-1}, x_p = u\}$ be the set of vertices on the outer cycle of G in clockwise order from a to u. Let $Y = \{y_0 = b, y_1, \cdots, y_{q-1}, y_q = v\}$ be the set of vertices on the outer cycle of G in counter clockwise order from b to v.

Therefore, we can only add an edge between a vertex in X and a vertex in Y. Thus to get the desired triangulation we need to realize one of the zigzag paths $P_1 = \{x_0, y_1, x_1, y_2, \cdots, x_{p-1}, y_q\}$ and $P_2 = \{y_0, x_1, y_2, x_1, \cdots, y_{q-1}, x_p\}$ by adding missing edges such that every vertex has degree 4 or less. It is not difficult to show that if none of P_1 and P_2 is realizable then G has no such triangulation.Thus, the following theorem holds.

Theorem 4. *Let G be an outerplanar graph of the maximum degree 4. Then one can determine in linear time whether G can be triangulated to maximal outerplanar graph with the maximum degree 4 or not. Furthermore such a triangulation can be found in linear time if it exists.*

References

[1] R. Tamassia, "Embedding vertices at points: Few bends suffice for planar graphs," *SIAM Journal on Computing*, vol. 16, no. 3, pp. 421–444, 1987.

[2] A. Garg and R. Tamassia, "A new minimum cost flow algorithm with applications to graph drawing," in *Graph Drawing*, 1996, pp. 201–216.

[3] ——, "On the computational complexity of up- ward and rectilinear planarity testing," *SIAM Journal on Computing*, vol. 31, no. 2, pp. 601–625, 2001.

[4] A. Morgana, C. P. de Mello, and G. Sontacchi, "An algorithm for 1-bend embeddings of plane graphs in the two-dimensional grid," *Discrete Applied Mathematics*, vol. 141, no. 1-3, pp. 225–241, 2004.

[5] Y. Liu, A. Morgana, and B. Simeone, "A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid," *Discrete Applied Mathematics*, vol. 81, no. 1-3, pp. 69–91, 1998.

[6] T. C. Biedl, "New lower bounds for orthogonal drawings," *Journal of Graph Algorithms and Applications*, vol. 2, no. 7, pp. 1–31, 1998.

[7] S.-M. Lee, M. Kitagaki, and J. Young, "On Edge-Graceful and Edge-Magic maximal Outerplanar Graphs," *Journal of Combinatorial Mathematics and Combinatorial Computing*, vol. 59, pp. 119–129, 2006.

[8] K. F. Jao and D. B. West, "Vertex Degrees in Outerplanar Graphs," 2010, preprint (2010), available at `www.math.uiuc.edu/ west/pubs/outerpl.pdf`.

Antibandwidth Problem for Itchy Caterpillar

Md. Sazzadur Rahaman, Tousif Ahmed, Sad Al Abdullah, Md. Saidur Rahman
Graph Drawing and Information Visualization Laboratory,
Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology
Email: {sazzad114,eshan077,siam9090}@gmail.com {saidurrahman}@cse.buet.ac.bd

***Abstract*—The antibandwidth problem is to label the vertices of a graph of n vertices by $1, 2, 3, \cdots, n$ bijectively, such that the minimum difference of labels of adjacent vertices is maximized. The antibandwidth problem is known as NP-hard. In this paper we give the lower bound of antibandwidth problem for a special class of tree called itchy caterpillar and also give the upper bound of antibandwidth problem for itchy caterpillar with hair length 1, which is the first non-trivial upper bound for this problem.**

I. Introduction

The antibandwidth problem is a popular vertex labeling problem, where we have to label the vertices in such way that the minimum diffrence between two vertices is maximized. More formally, let G be a graph on n vertices. Given a bijection $f : V(G) \rightarrow \{1, 2, 3, \ldots, n\}$, if $|f| = \min\{|f(u) - f(v)| : uv \ \epsilon \ E(G)\}$ then the antibandwidth of G is the maximum $\{|f|\}$ over all such bijections f of G.

The antibandwidth problem is NP-hard [1]. It is known to be polynomially solvable for the complements of interval, arborescent comparability and treshhold graphs [2], [3]. Exact results and tight bounds for paths, cycles, grid, meshes, hypercubes, complete binary tree found in [4], [5], [6], [7]. For complete k-ary tree [8], the antibandwidth is $\frac{n+1-k}{2}$, when k is even. Interestingly, still there is no results for any class of graphs, for which the upper bound of antibandwidth is non-trivial.

II. Antibandwidth for Itchy Caterpillar

We solve antibandwidth problem for a special class of trees which we call "Itchy Caterpillar". A Caterpillar T is an *itchy caterpillar* if T can be decomposed into vertex disjoint paths $P_0, P_1, P_2, \cdots, P_k$ such that (a) exactly one end of a path P_i, $1 \leq i \leq k$ is adjacent to a vertex on P_0, (b) each P_i, $1 \leq i \leq k$, has equal number of vertices and (c) the number of paths among P_i, $1 \leq i \leq k$, adjacent to each vertex on P_0 is equal.

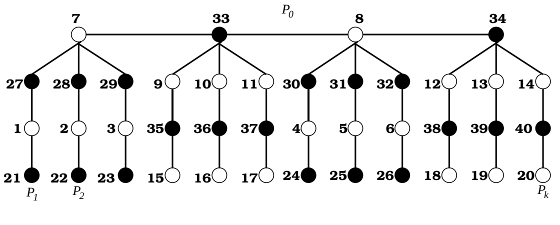

Fig. 1. An Itchy caterpillar with antibandwidth labeling

P_0 is called *spine* and P_i is called *hair*. In Figure 1, an itchy caterpillar having 4 spine vertices with 3 hairs (of length 3) per spine vertex, is shown with antibandwidth labelling. Later in this section, we will provide the upper bound and exact result of itchy caterpillar with hair length 1 and also provide lower bounds for all other itchy caterpillars. The following two theorems state our main results.

Theorem 1. *Let G be an itchy caterpillar, p be the number of spine vertices of G, q be the number of hairs per spine vertices of G, r be the length of hair of G, n be the number of vertices of G, then the antibandwidth of G is $\frac{n}{2}$ for p is even and $\lceil \frac{n-q}{2} \rceil$ for p is odd, when $r = 1$.*

Theorem 2. *Let G be a regular caterpillar with p spine vertices, and let q be the number of hairs linked to each spine vertices and r be the length of each hair. Then G admits a vertex labeling Γ, such that the difference of the labelings of any two adjacent vertices is at least,*

$$\begin{cases} \frac{n}{2} & \textit{if } p \textit{ is even} \\ \lfloor \frac{n}{2} \rfloor & \textit{if } p \textit{ is odd and } r \textit{ is even} \\ \lceil \frac{(n-q)}{2} \rceil & \textit{if } p \textit{ is odd and } r \textit{ is odd} \end{cases}$$

We need the following lemmas to prove our results.

Lemma 1. *Let G be an itchy caterpillar, p be the number of spine vertices of G, q be the number of hairs per spine vertices of G, r be the length of hair of G, n be the number of vertices of G. Then G admits a vertex labeling Γ, such that the difference of the labelings of any two adjacent vertices is at most $\left\lceil \frac{(n-q)}{2} \right\rceil$, when $r = 1$ and p is odd.*

Proof: To prove this lemma by contradiction, it is sufficent to prove that for an itchy caterpillar G, the minimum labelling difference $\left\lceil \frac{(n-q)}{2} \right\rceil + 1$ can not be acquired. To label the vertices of G, we have a set of labels, $N = \{1, 2, 3, \cdots, n\}$ with cardinality n.

Now to label G, first we split N into two sets N_1 and N_2 such that one of the set (let, N_1) has $\left\lceil \frac{(n-q)}{2} \right\rceil + 1$ number of labels and the other (let, N_2) has $n - \left\lceil \frac{(n-q)}{2} \right\rceil - 1$ number of labels like in Figure 2. In Figure 2, the lower consecutive half of N are chosen for N_1 and the upper consecutive half of N are considered to be in N_2. Let, b is the number of spine vertices, for those the labels are chosen from N_1 and $1 \leq b \leq p$. Now for adjacent vertices of the spine vertices, labeled from N_1, we are bound to choose labels from set N_2. So the labels of N_1 can be distributed like bellow :

- b number of labels are assigned to the corresponding number of spine vertices of G.
- rest of the labels are assigned to the leaves of the other $p - b$ spine vertices (spine vertices for those, the labels will be assigned from N_2) and the number of such leaves is $(p - b) * q$.

The labels of N_2 are distributed like bellow :

- $p - b$ number of labels are assigned to the corresponding number of remaining spine vertices of G.
- $b * q$ number of labels are assigned to the leaves of other b spine vertices (spine vertices for those, the labels are assigned from N_1).

If we choose the labels in the above mentioned way, then no two neighboring vertices will be assigned with the labels from N_1. Now considering the values that b can take, which corresponds to the arbitrarily choosing the number of labels for spine vertices from any of the halves of N, the following two cases can arise.

Case I: $\lceil \frac{p}{2} \rceil \leq b$.

In this case, the number of labels for the spine vertices chosen from N_1 is at least $\lceil \frac{p}{2} \rceil$. Now let $b = \lceil \frac{p}{2} \rceil + i$, such that $0 \leq i \leq \lfloor \frac{p}{2} \rfloor$. Let the number of labels needed to be in N_2 is $|N_2|'$. We can easily verify that the value of $|N_2|'$ must be,

$$\begin{aligned} &= (p - b) + (b * q) \\ &= (p - \left\lceil \frac{p}{2} \right\rceil - i) + \left(\left\lceil \frac{p}{2} \right\rceil + i \right) * q \\ &= \frac{n + q - 3}{2} + i(q - 1) + 1. \end{aligned}$$

But the value of $|N_2|$ is,

$$\begin{aligned} &= n - \left\lceil \frac{n - q}{2} \right\rceil - 1 \\ &= \frac{n + q - 3}{2}. \end{aligned}$$

But we see $|N_2|' - |N_2| \geq 0$, which means the number of labels in N_2 is less than the number of labels needed for the vertices G, from N_2 and thus contradicts the initial assumption.

Case II: $\lceil \frac{p}{2} \rceil > b$.

Similarly this case corresponds to the number of labels for the spine vertices chosen from N_1 is at most $\lceil \frac{p}{2} \rceil$. Now let $b = \lceil \frac{p}{2} \rceil - i$, such that $1 \leq i \leq \lceil \frac{p}{2} \rceil$. The total number of labels needed to be in N_1 is $\left[\left(\lceil \frac{p}{2} \rceil - i \right) + \left(\lfloor \frac{p}{2} \rfloor + i \right) * q \right]$ which is greater than the cardinality of N_1 and the total number of labels needed to be in N_2 is $\left[\left(\lfloor \frac{p}{2} \rfloor + i \right) + \left(\lceil \frac{p}{2} \rceil - i \right) * q \right]$

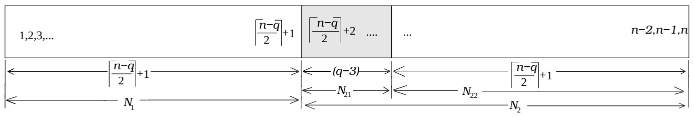
1,2,3,...
$\lceil\frac{n-q}{2}\rceil+1$
$\lceil\frac{n-q}{2}\rceil+2$ ....
...
n−2,n−1,n
$\lceil\frac{n-q}{2}\rceil+1$
(q−3)
$\lceil\frac{n-q}{2}\rceil+1$
N_{21}
N_{22}
N_1
N_2

Fig. 2. Partitioning N into N_1 and N_2 (also N_2 into N_{21} and N_{22}).

which is less than the cardinality of N_2. As N_2 has extra labels, we can divide N_2 into N_{21} and N_{22} like shown in Figure 2 and the labels from N_{21} can be added with N_1 to meet up the shortage. We see from Figure 2, that if this redistribution is possible then, still no violation of the assumption is done. But we will show that such assignment is not possible.

Let the number of labels needed to be in N_{21} is $|N_{21}|'$. We can easily verify that the value of $|N_{21}|'$ must be,

$$\begin{aligned} &= b + (p - b) * q - \left\lceil \frac{(n - q)}{2} \right\rceil - 1 \\ &= (\left\lceil \frac{p}{2} \right\rceil - i) + \left(\left\lceil \frac{p}{2} \right\rceil + i \right) * q - \frac{(n - q + 1)}{2} - 1 \\ &= i(q - 1) - 1. \end{aligned}$$

But the value of $|N_{21}|$ is,

$$\begin{aligned} &= n - 2 \left(\left\lceil \frac{n - q}{2} \right\rceil - 1 \right) \\ &= q - 3. \end{aligned}$$

We see $|N_{21}|' - |N_{21}| \geq 0$, which means the number of labels in N_{21} is less than the number of labels needed for the vertices G, from N_{21} violating the initial assumption. Hence our proof is done. ■

Lemma 2. *Let G be an itchy caterpillar, p be the number of spine vertices of G, q be the number of hairs per spine vertices of G, r be the length of hair of G, n be the number of vertices of G, Then G admits a vertex labeling Γ, such that the difference between two neighbour vertices is at least $\frac{n}{2}$ for p is even and $\lceil \frac{n-q}{2} \rceil$ for p is odd, when $r = 1$.*

Proof of Theorem 1: For an itchy caterpillar, when p is even and $r = 1$, Lemma 2 says that the antibandwith is at least $\frac{n}{2}$, which is the trivial upper bound of antibandwith for any class of graphs. Considering this with Lemma 1 (non-trivial upper bound for itchy caterpillars, when p is odd and $r = 1$) along with Lemma 2 concludes the proof of Theorem 1.

The proofs of Lemma 2 and Theorem 2 are omitted in this short version.

REFERENCES

[1] J. Leung, O. Vornberger, and J. Witthoff, “On some variants of the bandwidth minimization problem,” *SIAM JOURNAL*, vol. 13, pp. 650–667, 1984.
[2] S. Donnely and G. Isaak, “Hamiltonian powers in treshold and arborescent comparability graphs,” *Discrete Mathematics*, vol. 202, pp. 33–44, 1999.
[3] G. Isaak, “Powers of hamiltonian paths in interval graphs,” *Discrete Mathematics*, vol. 28, pp. 31–38, 1998.
[4] L. Yixun and Y. JinJiang, “The dual bandwidth problem for graphs,” *Electronic Notes in Discrete Mathematics*, vol. ED 35, pp. 1–5, 2003.
[5] L. Palios, “Upper and lower bounds for the optimal tree partitions,” *Technical Report GCG*, p. 46, 1994.
[6] A. Raspaud, H. Schroder, O. Sÿkora, L. Török, and I. Vrťo, “Antibandwidth and cyclic antibandwidth of meshes and hypercubes,” *Discrete Mathematics.*, vol. 309, pp. 3541–3552, 2009.
[7] Y. Weili, Z. Ju, and L. Xiaoxu, “Dual bandwith of some special trees,” *J. Zhengzhou Univ. Nat.Sci.*, vol. ED 35, pp. 1–5, 2003.
[8] T. Calamoneri, A. Massini, L. Török, and I. Vrťo, “Antibandwidth of complete k-ary trees,” *Electronic Notes in Discrete Mathematics*, vol. 24, pp. 259–266, 2006.

Interval Shortest-Route Problem

Hossain, Md. Akter
Department of CSE
IBAIS University
http://www.ibais.edu.bd
Email: akter_h@yahoo.com

***Abstract*—As our world grows in complexity, it becomes increasing difficult to make informed and intelligent decisions. Often, these decisions are made with less than perfect knowledge and the presence of considerable uncertainty. Yet solutions to these problems are essential to our well being and ever ultimate survival. The determination of shortest route between a source node and a destination node or two different locations or nodes on a large-scale real network is an important task in many applications. In this paper, we focus on the shortest route problem when the parameters of the given network are uncertain.**

I. Introduction

There are many reasons why network models, methods and algorithms are widely used, for instance, it exactly represents the real world systems, it facilitates extremely efficient solution to large real problems, it can solve problems with significantly more variables and constraints than can be solved by other optimization techniques, etc [12].
The Shortest-Route Problem (SRP) is a classical network problem, and it is the most popular algorithm among all network algorithms. The shortest route problem is concerned with determining the shortest route from an origin to a destination through a connecting network, given nonnegative distances associated with the respective arcs of the network [12-14].

A network is called acyclic, if it does not have any loop. If a network contains any loop, it will be cyclic. The acyclic algorithm is easier than the cyclic algorithm, because it yields fewer computations. And cyclic is more general in the sense that it subsumes the acyclic case [14].

Consider a connected network $G = (N, A)$, where $N = \{1, ..., n\}$ is the set of the nodes and $A = \{(i, j), (k, l), ..., (y, z)\}$ is finite set of arcs joining nodes in N. Let $d(i)$ is the distance label and i ∈ N; $d(i, j)$ is the generalized length (distance, time, cost, etc.) of arc $(i, j) \in A$ [6, 7].

In literature, Dijkstra algorithm [1] is considered as classical algorithm for SRP, and the last five decades hundreds of different variants of Dijkstra algorithm have been developed. It is the most implemented shortest route algorithm. It is simple, easy to understand and implement, and highly efficient algorithm [2, 9]. The basic idea of Dijkstra algorithm is as follows: it assigns to all nodes labels, which are either $temporary$ or $permanent$. At the beginning, each node receives a temporary label, which is further converted into a permanent label. This conversion is performed when it becomes evident that the last obtained distance in the label is the shortest distance to the node. In Dijkstra algorithm, in each iteration one temporary label is converted into permanent label [2, 8, 12, 14].

The interval shortest route problem [3-5, 10, 11] is concerned with determining the interval shortest route from an origin to a destination through a connecting network, given the interval generalized distance between nodes i and j is a non-negative, interval number represented by D_{ij} , D_{ij} = $[\overline{d_{ij}}, \underline{d_{ij}}]$.

II. INTERVAL SHORTEST-ROUTE ALGORITHMS

In many practical cases, the parameters of the network models are not exactly known, they are uncertain. A typical way to express these uncertainties in the edge weights is to utilize tools based on probability theory, interval mathematics, fuzzy sets theory, etc.

An interval algorithm is proposed for solving network problems under parametric uncertainty in [3]. The exact values of the parameters of a given network are unknown, but upper and lower limits within which the values are expected to fall are considered. The interval algorithm is developed on the base of midpoint and half-width representation of intervals. Considerable unification and simplification are obtained by using the mean-value lemma. This interval algorithm is applicable when the parameters of a given network are interval and non-interval. The interval algorithm is applicable when the given network is acyclic. The complexity of this algorithm is not evaluated.

Interval algorithms are presented for solving the shortest route problem for acyclic and cyclic networks in [4, 5, 11]. The formulation of the interval shortest route algorithms is an interval extension of classical shortest route algorithm given in [1]. The complexity of these algorithms is not evaluated.

Interval acyclic method and algorithm are proposed in [10], using midpoint and half-width representation of intervals and the mean value lemma. The complexity of this algorithm is evaluated.

Interval cyclic method and algorithm which represent interval extension of the Dijkstra algorithm are proposed in [10], using midpoint and half-width representation of intervals and the mean value lemma. The complexity of this algorithm is evaluated.

Interval acyclic method and algorithm and Interval cyclic method and algorithm are more efficient than the interval methods and algorithms that could be obtained by using traditional interval description and comparison, and the complexity of such an algorithm will be too high from the point of view of computation and practical applications.

A real application of interval cyclic algorithm is given in [10]. Using the interval cyclic algorithm, the authors obtained the interval shortest route from source node to any destination node for special transportation services in Sofia transportation network, for example, Police car, Fire service, Ambulance, etc. It would the possible to calculate the interval shortest route in Sofia transportation network from source node to any destination node for ordinary (public and private) transportation services. For ordinary transportation service, the author have to consider some additional parameters, for instance, traffic jam, direction of the route segments, allowed speed of the route segments, etc.

III. Conclusion

The network models are the most widely used models among all mathematical programming models. Since late 1950, the network models, methods, and algorithms are growing intensively from the point of view of both applications and theory. In real situation, it is not possible to know the parameters of the network models exactly in advance. It is not always possible to estimate travel cost in a network exactly, because it depends on many factors, for example, traffic jam, price of fuel, accident, etc. This type of parametric uncertainty is handled within the framework of probability theory or interval analysis or fuzzy sets theory.

Most of the available shortest route algorithms are for the deterministic case, when the parameters of the given network are certain. The deterministic algorithms are not workable in uncertain and/or risk situation.

There are some new algorithms available for uncertain parameters, for example, the robust algorithms but these algorithms are too complex from the point of view of applications and computations.

The interval algorithms are simple and computationally effective, because the traditional comparison of two intervals based on the concept of infimum-like intervals and distances is replaced by comparing two real values. We may conclude that interval shortest route algorithms with lower complexity are needed to deal with uncertainty.

References

[1] E. W. Dijkstra, A Note on Two Problems in Connexion with Graphs, Journal of Numerische Mathematik vol. 1, pp. 269-271, 1959.
[2] S. E. Dreyfus, An appraisal of some shortest-path algoS rithms, Journal of Operations Research, vol. 17, pp. 395-412, 1969.
[3] Gatev, G., Interval analysis approach and algorithms for solving network and dynamic programming problems under parametric uncertainty, Proceedings of the Technical University - Sofia, vol. 48, pp. 345-350, 1995.
[4] G. Gatev, A. Hossain, Interval Shortest Route Algorithms, Proceedings of the International Conference, Automation Information, vol. 2, pp. 68-71, Sofia, Bulgaria, 24-26 Oct., 2000.
[5] G. Gatev, A. Hossain, Interval Algorithms for Solving Minimal Spanning Tree and Shortest-Route Models, Journal of Information Technologies and Control, vol. 4, pp. 37-46, 2007.
[6] F. Glover, D. Klingman, New Sharpness Properties, Algorithms and Complexity bound for Partitioning Shortest Path Procedures, Journal of Operations Research, vol. 37, pp. 542-546, 1989.
[7] F. Glover, D. Klingman, N. V. Phillips, R. F. Schneider, New Polynomial Shortest Path Algorithms and their Computational Attributes, Journal of Management Science, vol. 31, pp. 1106-1128, 1985.
[8] S. F. Hillier, G. J. Lieberman, Introduction to Operations Research, Fifth edition. McGraw-Hill Publishing Company, New York, 1990.
[9] B. Golden, Shortest-Path Algorithm: A Comparison, Journal of Operations Research, vol. 24, pp. 1164-1168, 1976.
[10] A. Hossain, Interval Algorithms for solving Network and Dynamic programming under parametric uncertainty, Masters Thesis, Technical University of Sofia, 1999.
[11] A. Hossain, Network Models under Parametric Uncertainty, PhD Thesis, Technical University of Sofia, 2009.
[12] Don T. Phillips, A. Garcia-Diaz, Fundamentals of Network Analysis, Prentice-Hall Inc., Englewood Cliffs, N.J., 1981.
[13] M. Pollack, W. Wiebenson, Solution of the Shortest-Route Problem - A Review, Journal of Operations Research, vol. 8, pp. 224-230, 1960.
[14] H. A. Taha, Operations Research, An Introduction, Eighth edition, Pearson Education Inc., Delhi, 2007.

Zonohedra, Zone Graphs, and Their Linear Area Straight Line Grid Drawing

Muhammad A. Adnan, Ifat Afrin, Masud Hasan, Tahmina Khanam, and Mir M. Al-Mahmud
Department of CSE, BUET, Dhaka-1000, Bangladesh
Email: adnan@cse.buet.ac.bd, emi.buet@gmail.com,
masudhasan@cse.buet.ac.bd, mily_ampere@yahoo.com,sajib086_cse@yahoo.com

***Abstract*—*Zonohedra* are an important and infinite subclass of convex polyhedra having the property that the faces of a zonohedron are parallelograms. A *straight line grid drawing* of a planar graph G is a plane drawing of G such that the vertices of G are drawn on grid points and each edge is drawn by a straight line segment. The area of such a drawing is the area of the subscribing axis-aligned rectangle. It is a well known open problem to find subclasses of planar graphs that admit such drawings in subquadratic area. In quest of such classes of planar graphs, in this paper we investigate a subclass of 3-connected planar graphs, called *zone graphs*, which contains the naturally induced graphs of zonohedra. We then give a linear time algorithm for a linear area straight line grid drawing of a subclass of zone graphs, called *degree-constraint zone graphs*.**

I. Introduction

A (3D) *convex polyhedron* P is the bounded intersection of a finite number of half-spaces. One of the simplest subclasses of convex polyhedra are *generalized zonohedra*, where in every face the edges are in parallel pairs [4] (see Figure 1(a,b)). Coxeter [1] considered two other definitions of zonohedra to mean more special cases: (i) all faces are parallelograms and (ii) all faces are rhombi. Coxeter called these two types as *zonohedra* and *equilateral zonohedra* respectively. In this paper, by *zonohedra* we mean both zonohedra and equilateral zonohedra defined by Coxeter.

There is a very nice and remarkable relationship between a convex polyhedron and its graph. In 1916, in a famous theorem Steinitz stated that a graph is the naturally induced graph of a convex polyhedron if and only if it is 3-connected planar [2], [5].

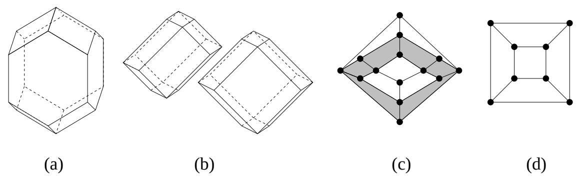

Fig. 1. (a) A generalized zonohedron. (b) Two zonohedra. (c) Graph of the zonohedra in (b); also a degree-constraint zone graph. A zone is shown as shaded. (d) Smallest zone graph (also degree-constraint) obtained by deleting a zone from the zone graph of (c); this is the graph of a parallelepiped.

a) Our results.: We investigate an infinite class of 3-connected planar graphs, called zone graphs, roughly whose definition is that the faces are quadrilaterals and grouped into "zones" and the zones intersect in a certain fashion. This class contains the naturally induced graphs of zonohedra, and therefore, is an infinite class. We study some properties of this class. In particular, we show that (1) a zone graph G can be partitioned into two equal halves by a "mid-cycle", (2) deleting a zone from G results into another zone graph, and (3) both the number of zones and the number of faces in each zone of G are $O(\sqrt{n})$.

We then investigate a subclass of zone graphs, called degree-constraint zone graphs, which follows some additional criteria on the degree of its vertices and that all the zones "touch" two specific vertices. For this class too, we show that deleting a zone from a degree-constraint zone graph results into another degree-constraint zone graph. Although, we can not find straight line grid drawing of zone graphs in subquadratic area, we give a linear time algorithm for drawing degree-constraint zone graphs in linear area. This result was motivated by the result in (3) as mentioned above, in the way that we shall draw a zone of G in $O(\sqrt{n})$ area "on average", and then drawing $O(\sqrt{n})$-many zones of G will take $O(n)$ area.

II. Zonohedra and zone graphs

Lemma 1. *Let P be a zonohedron and let G be the naturally induced graph of P. Then, G has the following three properties: (C1) G is 3-connected planar, (C2) the faces of G are quadrilaterals and grouped into at least three zones and each face belong to exactly two zones, and (C3) any two zones of G intersect into two faces that partition each zone into two non-empty equal chains of faces (excluding those two faces).*

Definition 1. *A graph G is called a* zone graph *if G has the three properties C1-C3 as mentioned in Lemma 1.*

See Fig. 1(c,d).

Theorem 1. *Naturally induced graphs of zonohedra are zone graphs.*

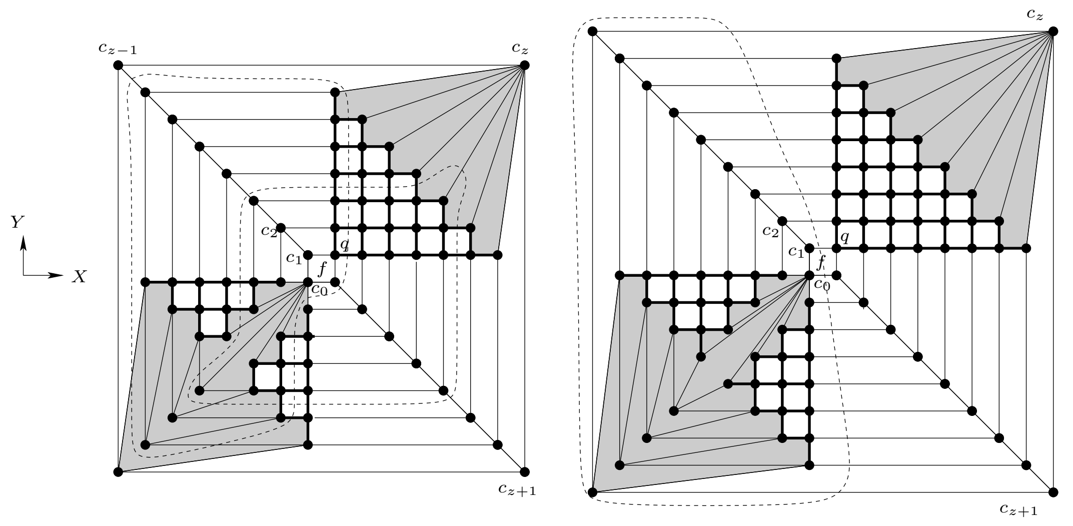

Fig. 2. Linear area grid drawing of G with an odd number of zones ($z = 9$, left figure) and an even number of zones ($z = 10$, right figure). $\mathcal{S}$, $\mathcal{H}_{top}$ and $\mathcal{H}_{right}$ are shown in bold and $\mathcal{U}$ and $\mathcal{T}$ are shown in shaded.

III. Properties of zone graphs

We define a *mid-cycle* C of G, which is a cycle of $2z$ edges, from a zone $\mathcal{Z}$ as follows: fix two reciprocal edges e and e' of $\mathcal{Z}$, take e, then starting from a right (or left) boundary edge of $\mathcal{Z}$ that is adjacent to e, continue taking the right (left) boundary edges of $\mathcal{Z}$ until we reach e', then take e', and then continue taking left (right) boundary edges in the same direction until we reach e.

Theorem 2. *Let G be a zone graph and let C be a mid-cycle of G. Then, (1) C partitions every zone of G into two equal chains of faces at a pair of reciprocal edges, and therefore, (2) C partitions G into two equal halves.*

Theorem 3. *Let G be a zone graph with $z \geq 4$ zones and let $\mathcal{Z}$ be a zone of G. Let G' be the resulting graph after contracting $\mathcal{Z}$ in G. Then G' is a zone graph of $z - 1$ zones.*

Theorem 4. *Let G be a zone graph with n vertices and z zones. Then, $z = O(\sqrt{n})$.*

IV. Degree-constraint zone graphs

Let C be a mid-cycle of G. Remember that C has $2z$ vertices. Each pair of vertices that partition C into two equal chains are called *reciprocal* vertices of G.

Definition 2. *A zone graph G is called a* degree constraint zone graph *if G follows the following two additional criteria: (C4) there exist two reciprocal vertices p and p' in G, called* poles *of G, such that every zone has these two poles in their boundary, one in the left boundary and the other one in the right boundary. This is called* touching *the two poles by a zone. (C5) the two poles have degree z, all neighbors of the poles have degree three, and all other vertices (if any) have degree four.*

For example, see Fig. 1(c,d).

Theorem 5. *Let G be a degree-constraint zone graph with $z \geq 4$ zones. Let G' be the graph that we achieve after contracting a zone $\mathcal{Z}$ from G. Then G' is a degree-constraint zone graph of $z - 1$ zones.*

We now present our most important result.

Theorem 6. *Given a degree-constraint zone graph G, its linear area straight line grid drawing can be found in $O(n)$ time, where n is the number of vertices in G.*

Since the above drawing generalizes for G with any number of zones, the following corollary is implied.

Corollary 1. *Degree-constraint zone graphs are an infinite class.*

V. Conclusion

We leave open drawing zone graphs in subquadratic area. It would also be interesting to see results similar to those in this paper, in particular an $o(n^2)$-area drawing, for the graphs of generalized zonohedra. Since there are algorithms for generating zonohedra [3], it may not be difficult to generate degree-constraint zonohedra.

References

[1] H. S. M. Coxeter. *Regular Polytopes*. Macmillan, New York, 1973.
[2] G. Das and M. T. Goodrich. On the complexity of optimization problems for 3-dimensional convex polyhedra and decision trees. *Computational Geometry*, 8(3):123–137, 1997.
[3] D. Eppstein. Zonohedra and zonotopes. *Mathematica in Education and Research*, 5(4):15–21, 1996.
[4] J. Taylor. Zonohedra and generalized zonohedra. *Amer. Math. Month.*, 99(2):108–111, 1992.
[5] G. M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, Berlin, 1995.

Algorithms for Bar 1-Visibility Drawings of Graphs

Shaheena Sultana, Md. Saidur Rahman, Arpita Roy, Suraiya Tairin
Graph Drawing and Information Visualization Laboratory,
Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology
shaheenaasbd@yahoo.com, saidurrahman@cse.buet.ac.bd,
{arpita116, suraiya_pakhi}@yahoo.com

***Abstract*—A bar 1-visibility drawing of a graph G is a drawing of G where each vertex is drawn as a horizontal line segment called a bar, each edge is drawn as a vertical line segment where the vertical line segment representing an edge must connect the horizontal line segments representing the end vertices and a vertical line segment corresponding to an edge intersects at most one bar which is not an end point of the edge. A graph G is bar 1-visible if G has a bar 1-visibility drawing. A graph G is 1-planar if G has a drawing in a 2-dimensional plane such that an edge crosses at most one other edge. In this paper we give linear-time algorithms to find bar 1-visibility drawings of diagonal grid graphs and maximal outer 1-planar graphs. We also show that recursive quadrangle 1-planar graphs and pseudo double wheel 1-planar graphs are bar 1-visibile graphs.**

I. Introduction

A 1-*planar drawing* of a graph G is a drawing of G on a two dimensional plane where an edge can be crossed by at most another edge. A graph G is 1-*planar* if G has a 1-planar drawing. A *straight-line drawing* of a graph G is a drawing of G such that every edge of G is drawn as a straight-line segment. A *right angle crossing drawing* or *RAC drawing* is a straight-line drawing where any two crossing edges form right angles at their intersection point. A *RAC graph* is a graph that has a RAC drawing. A *bar* 1-*visibility drawing* of a graph G is a drawing of G where each vertex is drawn as a horizontal line segment called a bar, each edge is drawn as a vertical line segment where the vertical line segment representing an edge must connect the horizontal line segments representing the end vertices and a vertical line segment corresponding to an edge intersects at most one bar which is not an end point of the edge. A graph G is a *bar* 1-*visible* if G has a bar 1-visibility drawing. A bar 1-visible graph and a bar 1-visibility drawing of the same graph is shown in Figures 1 (a), and (b), respectively.

In this paper we give a linear-time algorithm to find a bar 1-visibility drawing of a diagonal grid graph. We show that a diagonal grid graph is a RAC graph, a 1-planar graph and a bar 1-visible graph. We also develop a linear-time algorithm for finding a bar 1-visibility drawing of a maximal outer 1-plane graph which is RAC drawable [1], 1-planar and bar 1-visible. We also show that recursive quadrangle 1-planar graphs and pseudo double wheel 1-planar graphs, which are not RAC graphs, are bar 1-visible graphs.

The rest of the paper is organized as follows. Sections II deals with linear algorithms for finding bar 1-visibility drawings of diagonal grid graphs and maximal outer 1-plane graphs and and Section III deals with algorithms for finding bar 1-visibility drawings of recursive quadrangle 1-planar graphs and pseudo double wheel 1-planar graphs. Finally, Section IV concludes our paper with a list of open problems.

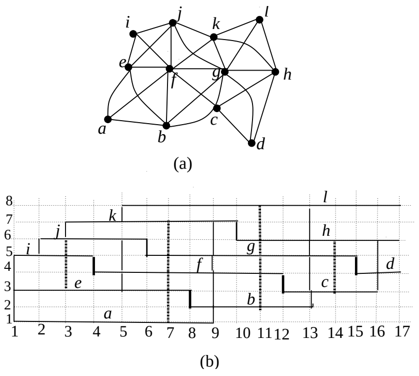

Fig. 1. (a) A bar 1-visible graph, and (b) a bar 1-visibility drawing of the same graph.

II. Linear Algorithms for Bar 1-Visibility Drawings of Graphs

Some interesting labeling properties of diagonal grid graphs have been studied recently by Selvaraju and Pricilla [2]. Recently Dehkordi showed that outer-1-plane graphs are RAC graphs [1]. In this section we find diagonal grid graphs and maximal outer 1-planar graphs as two classes of graphs which are RAC drawable, 1-planar and bar 1-visible. In this section, we develop algorithms for finding a bar 1-visibility drawing of a diagonal grid graph and for finding a bar 1-visibility drawing of a maximal outer 1-plane graph.

A $p \times q$-*grid graph* is a graph whose vertices correspond to the grid points of a $p \times q$-grid in the plane and edges correspond to the grid lines between two consecutive grid points. *A diagonal grid graph* $G_{p,q}$ is a $p \times q$-grid graph with diagonal edges are introduced in each cell. In this Section we prove the following theorem.

Theorem 1. *A bar 1-visibility drawing of a diagonal grid graph can be drawn in linear-time.*

A 1-planar graph G is *optimal* if no edges can be added to G without losing 1-planarity. That is, an optimal 1-planar graph of n vertices has the highest number of edges among all 1-planar graphs of n vertices. An *outer 1-plane graph* is a topological embedding of a graph such that all vertices lie on the outer face and there is at most one crossing on each edge. An outer 1-plane graph $G = (V, E)$ is a *maximal outer 1-plane graph* if for each pair u, v of vertices where (u, v) is not an edge, adding the edge (u, v) to G makes it not outer 1-plane; that is, $G' = (V, E \cup (u, v))$ is not outer 1-plane for every drawing of the edge (u, v). In this Section we give an algorithm for obtaining a bar 1-visibility drawing of a maximal outer 1-plane graph. This problem has an interesting correlation with a constrained visibility drawing of a planar st-graph [3]. We can introduce diagonal labeling on the maximal outer 1-plane graph. We have the following lemma.

Lemma 1. *Every maximal outer 1-plane graph admits diagonal labeling.*

Using this diagonal labeling, we can construct a bar 1-visibility drawing of a maximal outer 1-plane graph as mentioned in the following theorem.

Theorem 2. *A bar 1-visibility drawing of a maximal outer 1-plane graph can be drawn in linear-time.*

III. Bar 1-Visible 1-Planar Graphs

In the previous section, we showed that diagonal grid graphs and maximal outer 1-planar graphs are two classes of 1-planar graphs. Recently Eades and Liotta showed that every maximally dense RAC graph is 1-planar [4]; on the other hand, they introduced a class of 1-planar graphs which is not RAC drawable. In this section we call the class recursive quadrangle 1-planar graphs. Suzuki studied the combinatorial properties of the optimal 1-planar graphs having $4n-8$ edges [5] which we will define as “pseudo double wheel 1-planar graphs”. Since a RAC graph with $n > 3$ vertices has at most $4n-10$ edges [6], pseudo double wheel 1-planar graphs are not RAC graphs. In this section we show that recursive quadrangle 1-planar graphs and pseudo double wheel 1-planar graphs are bar 1-visible graphs.

The class recursive quadrangle 1-planar graph is defined recursively, as follows. G_0 is a 1-planar graph of eight vertices. G_0 has an 1-planar drawing where the cycle $abcd$ is drawn as the outer rectangle and none of the four edges on the outer rectangle has a crossing. Let $abcd$ be the outer rectangle of G_i, $i \geq 0$. Graph G_{i+1} is obtained from G_i by adding a new outer rectangle $a'b'c'd'$ and 16 new edges where the four edges on the the outer face do not have any crossing. Let G be a 1-planar graph and x be a vertex of G and (x, y) be an edge of G. We denote by $\Gamma(G)$, $\Gamma(x)$, $\Gamma(x, y)$ a bar 1-visibility drawing of G, the drawing of a vertex x as a horizontal bar in $\Gamma(G)$, and the drawing of an edge (x, y) as a vertical line segment in $\Gamma(G)$, respectively. We now have the following theorem.

Theorem 3. *Every recursive quadrangle 1-planar graph $G_i, i \geq 0$ is a bar 1-visible graph.*

Let C be a cycle $v_1u_1v_2u_2...v_nu_n$ of even number of vertices embedded on a plane. Let x and y be two vertices outside and inside of C, respectively. We add x with u_i and y with v_i for $i = 1...n$. Let H be the resulting plane graph. We add a pair of crossing edges to each face of H. The resulting graph is an optimal 1-planar graph as introduced by Suzuki [5]. We call this optimal 1-planar graph *even pseudo double wheel 1-planar graph*. Q_v *splitting* is an expansion operation at v_1 defined as follows : (i) Identify v_2 and v_3 such that there are v_1v_2 and v_1v_3 edges but no v_2v_3 edge. (ii) Split $v_1v_2v_3$ path (iii) Rename one copy of v_1 as v_4. (iv) Join v_1 , v_4 and v_2 , v_3. An optimal 1-planar graph obtained from even pseudo double wheel 1-planar graph by one splitting operation is called *odd pseudo double wheel 1-planar graph*. We now have the following theorem.

Theorem 4. *Every pseudo double wheel 1-planar graph is a bar 1-visible graph.*

IV. Conclusion

In this paper we give linear-time algorithms to find bar 1-visibility drawings of diagonal grid graphs and maximal outer 1-planar graphs. We recognized that diagonal grid graphs and maximal outer 1-planar graphs are RAC drawable, 1-planar and bar 1-visible but recursive quadrangle 1-planar graphs and pseudo double wheel 1-planar graphs are not RAC drawable but bar 1-visible graphs. We also developed algorithms for finding bar 1-visibility drawings of these classes of graphs. We were able to construct bar 1-visibility drawing of every 1-planar graph that we considered as an example, but yet to find a formal proof. We thus conjecture as follows.

Conjecture 1. *Every 1-planar graph is a bar 1-visible graph.*

Several interesting open problems have come out from this works.

1) Recognition of both RAC graphs and 1-planar graphs are NP-complete problems. It is interesting to know the complexity of recognizing a bar 1-visible graph. Finding a complete characterization of bar 1-visible graph is also an interesting open problem.
2) How to find a 1-planar embedding of a 1-planar graph?
3) Can we find a complete characterization of bar k-visibility drawing?

References

[1] H. R. Dehkordi, *On Algorithmic Right Angle Crossing Graph Drawing*, ser. Thesis, MPhil, The University of Sydney, August, 2012.

[2] P. Selvaraju and B. Pricilla, “On Cordial Labeling: The Grid, Diagonal Grid, Structured Web Graphs,” *International Journal of Algorithms, Computing and Mathematics*, vol. 2, no. 1, pp. 5–14, 2009.

[3] D. Battista, R. Tamassia, and I. G. Tollis, “Constrained visibility representation of graphs,” in *Inform. Process. Letters*, vol. 41, 1992, pp. 1–7.

[4] P. Eades and G. Liotta, “Right angle crossing graphs and 1-planarity,” in *The Proceedings of Graph Drawing’11*, ser. Lecture Notes in Computer Science, vol. 7034. Springer-Verlag, 2012, p. 206217.

[5] Y. Suzuki, “Optimal 1-planar graphs which triangulate other surfaces,” *Discrete Mathematics*, vol. 310, no. 1, pp. 6–11, 2010.

[6] W. Didimo, P. Eades, and G. Liotta, “Drawing graphs with right angle crossings,” in *The Proceedings of WADS’09*, ser. Lecture Notes in Computer Science, vol. 5664. Springer-Verlag, 2009, p. 206217.

A New Clustering Technique using (k, w)-Core Decomposition for Restructuring Software Functions

Aftab Hussain and Md. Saidur Rahman
Graph Drawing and Information Visualization Laboratory
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology
Email: aftab.hussain46@gmail.com, saidurrahman@cse.buet.ac.bd

***Abstract*—In this paper, we introduce a hierarchical agglomerative clustering technique (HAC) based on a new graph theoretic algorithm, (k, w)-Core Decomposition. Previous HACs generate clustering trees or dendrograms with a large number of cut-points and bad clusters. The new technique generates lower readings for these two parameters, while giving results of competitive quality, efficiently. To establish this, we implemented our HAC and previous HACs for restructuring software functions, and made a comparative analysis of the results.**

I. Introduction

Hierarchical agglomerative clustering (HAC) algorithms have found wide utility in different applications primarily because of the efficiency with which they could be implemented. One such application is the restructuring of low-cohesive software modules[1]. HACs return a hierarchy of clusters, which is visualized as a dendrogram (Fig. 1): a 2D diagram in which a scale of similarity from 1 to 0 is represented in the vertical axis and entities are indicated in the horizontal axis. Each horizontal line in the dendrogram indicates a cluster, the height of which indicates the level of similarity of the cluster. A *cut-point* in the dendrogram is the level of similarity at which a dendrogram is cut to obtain a partition of the entities. Each cluster corresponds to a cut-point. Each cut-point yields a partition of clusters, which give advice on how to restructure the module. Previous HACs, Single Linkage Algorithm (SLINK), Complete Linkage Algorithm (CLINK), Weighted Pair Group Method of Arithmetic Averages (WPGMA), and Adaptive K-Nearest Neighbour Algorithm (A-KNN) ([2], [3], [4], [5]), return dendrograms with a large number of cut-points, of which only a few yield clusters that lead to a meaningful restructuring. We present an efficient HAC based on (k, w)-Core Decomposition, (k, w)-CC, that generates clustering trees which contain lower numbers of cut-points and bad clusters. The technique is implemented to restructure software functions and shown to give competitive results with respect to previous HACs.

II. Preliminaries

We shall first present definitions and lemmas related to two key elements of (k, w)-CC: the *k-core*, first introduced by Seidman [6], and the (k, w)*-core*, introduced in this work.

[1]The cohesion of a module is the degree to which the module performs a unique task [1].

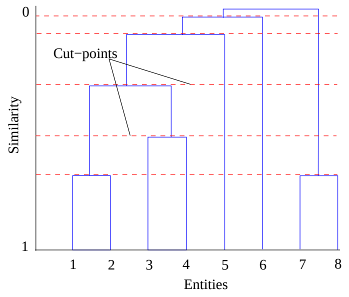

Fig. 1. A dendrogram with cut-points.

Definition 1: Let $G = (V, E)$, be a graph, where V is the set of vertices and E is the set of edges. A subgraph H_k of G induced by a vertex set $V \subseteq V$ is a k*-core* of G if every vertex in V has degree at least k in H_k, and H_k is the maximum subgraph with this property [7].

Lemma 1: If H_{k_1}, H_{k_2} are the k_1- and k_2-cores, respectively, of a graph G, where $k_2 > k_1$, then H_{k_2} is a subgraph of H_{k_1}.

Definition 2: Let W be the set of different edge weights of graph G, where $w \in W$. Then a (k, w)*-core* of G is a subgraph of G where the degree of each vertex of the subgraph is at least k and the weight of each edge of the subgraph is at least w, and this subgraph is the maximum subgraph with this property.

Lemma 2: A (k, w)-core of a weighted graph G is a subgraph of a k-core of G.

III. (k, w)-Core clustering ((k, w)-CC

Fig. 2 illustrates the overall approach of our novel clustering technique. The clustering technique operates on a weighted graph, where vertices represent entities and edges represent the presence of some similarity between the vertices (entities) joined by the edges[2]. Each edge carries a weight equal to the similarity value between the vertices joined by the edge. Firstly, (Fig. 2(a)), all possible (k, w)-cores are generated from the weighted graph. Then, (Fig. 2(b)), cores are systematically selected to form clusters, which together form a cluster hierarchy. We now discuss these steps in detail.

[2]In the context of software restructuring at the function level, entities denote statements of the function and similarity values denote the relationship between entities based on a resemblance coefficient. (See [5].)

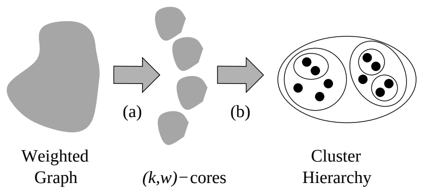

Fig. 2. Clustering approach using (k, w)-Core Clustering.

(k, w)-Core Decomposition. This decomposition algorithm is based on the k-core decomposition algorithm [7]. The algorithm begins by scanning every vertex v of input graph G to find the k-core for each $k \epsilon D$, where D is the ordered set of distinct degrees of the graph. If the degree of a vertex v is found to be less than k, v is deleted and the degrees of v's neighbours are decremented. In case the degrees of any of v's neighbours, fall below k, those vertices are also deleted recursively. In this manner, a k-core is generated. For generating k-cores for successive k values in D, it is sufficient to scan the k-core with the preceding k value rather than the entire graph (see Lemma 1).

By Lemma 2, for a particular value of k, say k_n, all (k_n, w)-cores of G can be obtained from the k_n-core of G. In order to do this the algorithm scans all edges of the k_n-core. For each weight $w \epsilon W$, where W is the set of distinct weights of G, edges with weights less than w are deleted. We thus get a set of intermediate graphs for each value of w. Next, the degrees of the vertices of each intermediate graph is checked and deleted if their degrees fall below k. We thus obtain all the (k_n, w)-cores of G. Carrying the same steps on all k-cores gives us all the (k, w)-cores of G.

Core Selection and Clustering Tree Generation. The basic idea of this phase is to select the (k, w)-cores as clusters in order to form a hierarchy. Cores are ranked using a new metric that relies on the k and w value of the core[3]. The selection process terminates until all entities have been selected.

IV. Experimental results

In our experiments we restructured 11 functions (ranging from 9 to 41 LOCs) extracted from published papers ([5], [2], [8]) and a real-life software, Sweet Home 3D ([9]), using SLINK, CLINK, WPGMA, A-KNN, and (k, w)-CC. We recorded the **number of cut-points** and the **number of bad clusters** that were observed in their respective dendrogram outputs. We also measured the **maximum cohesion improvement** that was attainable through each technique and the **execution time** taken by each technique to generate the dendrograms. We present the results as follows,

Number of cut-points and bad clusters. On average, (k, w)-CC gave 29.23%, 39.47%, 52.58%, 31.34% fewer number of cut-points than did SLINK, CLINK, WPGMA, and A-KNN, respectively. In addition, (k, w)-CC gave 57.41%, 62.90%, 68.49%, 54.00% fewer number of bad clusters than did SLINK, CLINK, WPGMA, and A-KNN, respectively.

[3]A detailed explanation of this metric has been omitted in this short version.

Maximum Cohesion Improvement. Overall, the maximum cohesion improvement level that was possible to achieve with SLINK, CLINK, WPGMA, A-KNN, (k, w)-CC was 99.88%, 99.92% 99.92%, 99.88%, 99.88%. Thus, the quality of (k, w)-CC's results competes well against those of the other techniques.

Execution Time. We measured the time taken, in milliseconds (ms), by each clustering technique to generate dendrograms for each of the functions that were analyzed [4]. Overall, (k, w)-CC performed 59.72% percent faster than SLINK, CLINK, and WPGMA. However, A-KNN performed the fastest among all the techniques. (It was 94.77% faster SLINK, CLINK, and WPGMA).

V. Conclusion

In this work, we developed a new hierarchical clustering technique based on (k, w)-core decomposition, (k, w)-Core Clustering ((k, w)-CC), for restructuring functions. We compared the performance of (k, w)-CC with four previous HACs (SLINK, CLINK, WPGMA, and A-KNN). The techniques were implemented on functions extracted from published papers and real-life software. Our technique gave competitive restructuring solutions through dendrograms that contained a smaller number of cut-points and a smaller number of bad clusters. As a result, (k, w)-CC's dendrograms were easier to analyze, from which meaningful suggestions were more readily retrievable. Performance-wise, although (k, w)-CC was slower than A-KNN, it was considerably faster than SLINK, CLINK, and WPGMA.

Acknowledgments

We would like to thank Mohammad Tanvir Parvez, Abdulaziz Alkhalid, Mohammad Alshayeb, and Sabri Mahmoud for providing the implementation details of the A-KNN algorithm.

References

[1] R. Pressman, *Software Engineering: A Practitioner's Approach*, 6th ed. McGraw-Hill, 2004.
[2] A. Alkhalid, M. Alshayeb, and S. Mahmoud, "Software refactoring at the function level using new adaptive k-nearest neighbor algorithm," *Journal of Advances in Engineering Software*, vol. 41, no. 10-11, pp. 1160–1178, 2010.
[3] ——, "Software refactoring at the package level using clustering techniques," *IET Software*, vol. 5, no. 3, pp. 276–284, 2011.
[4] N. Anquetil and T. C. Lethbridge, "Experiments with clustering as a software remodularization method," in *Proceedings of 6th Working Conference on Reverse Engineering*, 1999, pp. 235–255.
[5] C. H. Lung, X. Xu, M. Zaman, and A. Srinivasan, "Program restructuring using clustering techniques," *Journal of Systems and Software*, vol. 79, no. 9, pp. 1261–1279, 2006.
[6] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, no. 3, pp. 269–287, 1983.
[7] V. Batagelj and M. Zaversnik, "An o(m) algorithm for cores decomposition of networks," in *CoRR (Computing Research Repository), cs.DS/0310049*, 2003.
[8] J. M. Bieman and B.-K. Kang, "Measuring design-level cohesion," *IEEE Transactions on Software Engineering*, vol. 24, no. 2, pp. 111–124, 1998.
[9] "Sweethome3d," http://sourceforge.net/projects/sweethome3d/?source=directory, July 2012.

[4]All executions were carried out in a system with a 2.4 GHz processor and a 4096 Mb RAM.

Irregular Total Labelling of Möbius Ladder

Kh. Md. Mominul Haque
Department of Computer Science and Engineering
Sylhet International University
Email: momin66@gmail.com

***Abstract*—The total edge irregularity strength $tes(G)$ and total vertex irregularity strength $tvs(G)$ are invariants analogous to irregular strength $s(G)$ of a graph G for total labellings. Bača et al [Discrete Mathematics, 307:1378–1388, (2007)] determined the bounds and precise values for some families of graphs concerning these parameters. In this paper, we show the exact values of the total edge irregularity strength is $tes(M_n) = n + 1$ and total vertex irregularity strength is $tvs(M_n) = \lceil \frac{n}{2} \rceil + 1$ for the Möbius Ladder (M_n).**

I. Introduction

We consider only finite undirected graphs without loops or multiple edges. Let $G = (V, E)$ be a graph with vertex set V and edge set E.

An edge irregular total k-labelling of a graph G is a labelling of the vertices and edges with labels 1, ..., k that for every two different edges their weights are distinct where the weight of an edge is the sum of its label and the labels of its two endvertices. A vertex irregular total k-labelling of a graph G is a labelling of the vertices and edges with labels 1, ..., k that for every two different vertices their weights are distinct where the weight of a vertex is the sum of its label and the labels of its incident edges. The minimum k for which the graph G has an edge irregular total k-labelling is called the total edge irregularity strength of the graph G, $tes(G)$. Analogously, the minimum k for which there exists a vertex irregular total k-labelling is called the total vertex irregularity strength of G, $tvs(G)$.

The notions of the total edge irregularity strength and total vertex irregularity strength were first introduced by Bača et al. [1]. They may be taken as an extension of the irregularity strength of a graph [3], [2], [5], [7], [8], [10], [11]. In [1], the authors put forward the lower bounds of $tes(G)$ and $tvs(G)$ in terms of the maximum degree Δ, minimum degree δ, $|E(G)|$ and $|V(G)|$, which may be stated as the Theorem 1.1 and 1.2:

Theorem 1.1. $tes(G) \geq max\{\lceil \frac{\Delta+1}{2} \rceil, \lceil \frac{|E(G)|+2}{3} \rceil\}$.

Theorem 1.2. $tvs(G) \geq \lceil \frac{|V(G)|+\delta}{\Delta+1} \rceil$.

Bača et al. [1] then determined the exact values of the total edge irregularity strength for Path P_n, Cycle C_n, Star S_n, Wheel W_n and friendship graph F_n, and obtained the exact values of the total vertex irregularity strength for Tree T with n pendant vertices and no vertex of degree 2, Star S_n, complete graphs K_n, cycle C_n and prism D_n. The author proved that irregular total labellings of Generalized Petersen graphs $P(n, k)$ in [9]. We refer the readers for some recent result [3], [5], [8].

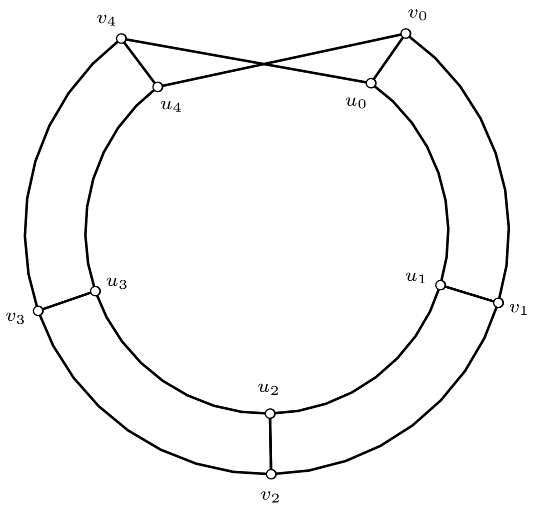

Figure 1.1. Möbius Ladder M_5

In this paper, we deal with the Möbius Ladder M_n.

The Möbius ladder M_n is defined to be a graph on $2n(n \geq 3)$ vertices with $V(M_n) = \{v_i, u_i : 0 \leq i \leq n-1\}$ and $E(M_n) = \{v_iu_i : 0 \leq i \leq n-1\} \cup \{v_iv_{i+1}, u_iu_{i+1} : 0 \leq i \leq n-2\} \cup \{u_{n-1}v_0, v_{n-1}u_0\}$.

In Figure 1.1, we show the Möbius Ladder M_5.

II. Irregular total labelling of M_n

Theorem 2.1. $tes(M_n) = n + 1$.

Proof.

We construct the function f as follows:

$$
\begin{array}{rcll}
f(u_i) & = & 1, & 0 \leq i \leq n-1, \\
f(v_i) & = & \begin{cases} n, & i = 0, \\ n+1, & 1 \leq i \leq n-1, \end{cases} \\
f(u_iu_{i+1}) & = & i+1, & 0 \leq i \leq n-2, \\
f(u_{n-1}v_0) & = & 1, \\
f(v_{n-1}u_0) & = & 1, \\
f(u_iv_i) & = & \begin{cases} i+3, & i = 0, \\ i+2, & 1 \leq i \leq n-1, \end{cases} \\
f(v_iv_{i+1}) & = & \begin{cases} i+3, & i = 0, \\ i+2, & 1 \leq i \leq n-2. \end{cases}
\end{array}
$$

Observe that

$$
\begin{array}{l}
wt(u_iu_{i+1}) = 1 + (i+1) + 1 \\
, \qquad = \{3, 4, \ldots, n+1\}, \qquad 0 \leq i \leq n-2, \\
wt(u_{n-1}v_0) = 1 + 1 + n = n+2, \\
wt(v_{n-1}u_0) = (n+1) + 1 + 1 = n+3, \\
wt(u_iv_i) = \begin{cases} 1 + (i+3) + n = n+4, & i = 0, \\ 1 + (i+2) + (n+1) & \\ = \{n+5, n+6, \ldots, 2n+3\}, & 1 \leq i \leq n-1, \end{cases} \\
wt(v_iv_{i+1}) = \begin{cases} n + (i+3) + (n+1) = 2n+4, & i = 0, \\ (n+1) + (i+2) + (n+1) & \\ = \{2n+5, 2n+6, \ldots, 3n+2\}, & 1 \leq i \leq n-2. \end{cases}
\end{array}
$$

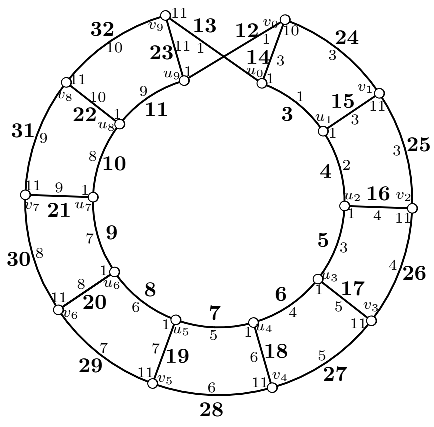
Figure 2.1. Edge irregular total labellings for M_{10}

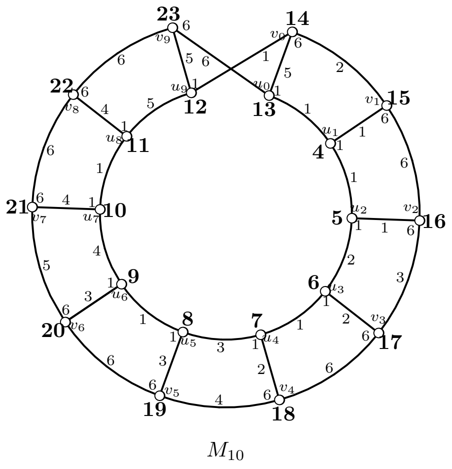
Figure 2.3. Vertex irregular total labellings for M_{10}

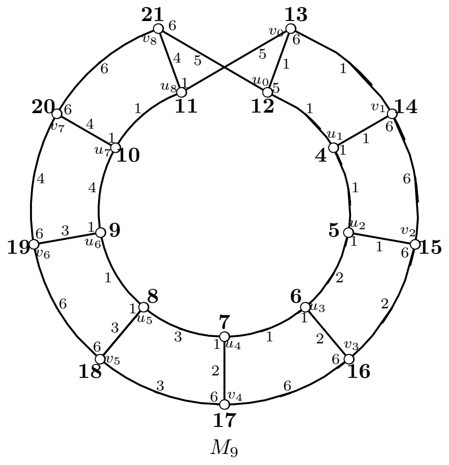
Figure 2.2. Vertex irregular total labellings for M_9

So the weights of edges of M_n under the labelling f constitute the set $\{3, 4, \ldots, 3n+2\}$ and the function f is a map from $M_n \bigcup E(M_n)$ into $\{1, 2, \ldots, n+1\}$.

It is clearly that the total labellings f have the required properties of an edge irregular total labelling. Hence $tes(M_n) \leq n+1$. By Theorem 1.1, $tes(M_n) \geq \lceil \frac{|E(G)|+2}{3} \rceil = \lceil \frac{3n+2}{3} \rceil = n+1$. This concludes the proof.

In Figure 2.1, we show the edge irregular total labellings for M_{10} .

Theorem 2.2. $tvs(M_n) = \lceil \frac{n}{2} \rceil + 1$.

In Figure 2.2. and 2.3., we show the vertex irregular total labeling for M_9 andM_{10}.

References

[1] M. Bača, S. Jendrol, M. Miller, J. Ryan, On irregular total labellings, *Discrete Mathematics*, 307(2007), 1378–1388.

[2] J. L. Baril, H. Kheddouci and O. Togni, The irregularity strength of circulant graphs, *Discrete Mathematics,* 304(2005), 1–10.

[3] T. Bohman and D. Kravitz, On the irregularity strength of trees, *Journal of Graph Theory,* 45(4)(2004), 241–254.

[4] J. Ivančo and S. Jendrol, The total edge irregularity strength of trees, *Discussiones Mathematicae. Graph Theory,* to appear.

[5] S. Jendrol, M. Tkáč, The irregularity strength of tKp, *Discrete Mathematics,* 145(1995), 301–305.

[6] S. Jendrol, J. Miškuf and R. Soták Total edge irregularity strength of complete graphs and complete bipartite graphs, *Electronic Notes in Discrete Mathematics,* 28(2007), 281–285.

[7] S. Jendrol and V. Žoldák, The irregularity strength of generalized Petersen graphs, *Mathematica Slovaca,* 45(2)(1995), 107–113.

[8] A. Meissner, M. A. Niwińska and K. T. Zwierzynski, Computing an irregularity strength of selected graphs, *Electronic Notes in Discrete Mathematics,* 24(2006), 137–144.

[9] Khandoker Mohammed Mominul Haque, Irregular Total Labellings of Generalized Petersen graphs, *Theory of Computing Systems,* 50(2012), 537–544.

[10] O. Togni, Irregularity strength of the toroidal grid, *Graphs and combinatorics,* 165/l66(1997), 609–620.

[11] O. Togni, Irregularity strength and compound graphs, *Discrete Mathematics,* 218(2000), 235–243.

Invited Speakers

Subhas Chandra Nandy	ISI Kolkata, India
Md. Rezaul Karim	Dept. of CSE, DU
M. Kaykobad	Dept. of CSE, BUET
Masud Hasan	Dept. of CSE, BUET

GDGA 2013 Committees

Advisory Committee

Md. Mostofa Akbar	Dept. of CSE, BUET
Muhammad Masroor Ali	Dept. of CSE, BUET
Masud Hasan	Dept. of CSE, BUET
Abu Sayed Md. Latiful Hoque	Dept. of CSE, BUET
Mohammad Mahfuzul Islam	Dept. of CSE, BUET
Md. Monirul Islam	Dept. of CSE, BUET
M. Kaykobad	Dept. of CSE, BUET
Md. Abul Kashem Mia	Dept. of CSE, BUET
Md. Saidur Rahman	Dept. of CSE, BUET

Programme Committee

Masud Hasan	Dept. of CSE, BUET
Md. Rezaul Karim	Dept. of CSE, DU
M. Kaykobad	Dept. of CSE, BUET Chair
Md. Saidur Rahman	Dept. of CSE, BUET
M. Sohel Rahman	Dept. of CSE, BUET

Organizing Committee

Md. Iqbal Hossain	Dept. of CSE, BUET
A. B. M. Alim Al Islam	Dept. of CSE, BUET Secretary
A.S.M. Sohidull Islam	Dept. of CSE, BUET
Md. Abu Sayeed Mondol	Dept. of CSE, BUET
Md. Mustafizur Rahman	Dept. of CSE, BUET
Md. Saidur Rahman	Dept. of CSE, BUET Chair
Sajjadur Rahman	Dept. of CSE, BUET
Shubhra Kanti Karmaker Santu	Dept of CSE, BUET
Shaheena Sultana	Dept. of CSE, BUET

This is a publication of the Department of Computer Science and Engineering, BUET under the HEQEP Subproject CP # 2082. The graphs used in the cover page of the proceedings (clockwise: Paley graph, Clebsch graph, Foster graph, Errera Graph) were collected from $http : //en.wikipedia.org/$.