ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
CONSIGLIO NAZIONALE DELLE RICERCHE

F. Malucelli, S. Nicoloso

OPTIMAL PARTITION OF A BIPARTITE GRAPH INTO NON–CROSSING MATCHINGS

R. 564 Febbraio 2002

Federico Malucelli- Dipartimento di Elettronica e Informazione del Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy. E-mail: malucell@elet.polimi.it

Sara Nicoloso - Istituto di Analisi dei Sistemi ed Informatica del CNR, viale Manzoni 30 - 00185 Roma, Italy. E-mail: nicoloso@iasi.rm.cnr.it.

Istituto di Analisi dei Sistemi ed Informatica, CNR
viale Manzoni 30
00185 ROMA, Italy

tel. ++39-06-77161
fax ++39-06-7716461
email: **iasi@iasi.rm.cnr.it**
URL: **http://www.iasi.rm.cnr.it**

Abstract

Given a bipartite graph and a layout of it, we address the problem of partitioning the edge set of the graph into the minimum number of non–crossing matchings, that is subsets of edges no two of which share a common vertex or cross each other in the plane. We discuss some lower and upper bounds on the minimum number of classes of such a partition into non–crossing matchings, and devise an exact almost linear algorithm.

KEYWORDS: *colouring, complexity, matchings, bipartite graph*s.

1. Introduction

In this paper we shall deal with a problem which depends on both the topology of a bipartite graph and on the geometry defined by a given layout of it. The topology of the bipartite graph is fully represented by $G=(L\cup R,E)$, where $L=\{l_1,l_2\ldots,l_{n_L}\}$ and $R=\{r_1,r_2\ldots,r_{n_R}\}$ are two independent sets of nodes, and E is the set of the m edges, each of which is an ordered pair (l_i,r_j) of nodes. As for the geometry of the problem, we shall assume that a *layout* λ is given, and precisely, that the two node subsets of G are arranged in two distinct parallel columns, following top–down the total order given by their indices in their own set. In particular, nodes in L are on the left column (*left nodes)*, and nodes in R are on the right column (*right nodes*) (see Fig.1). Throughout the rest of the paper the pair (G, λ) will denote a bipartite graph G and a layout λ of it.

A subset of edges no two of which *intersect*, that is, no two of which share a common node or cross each other in the plane is called *non–crossing matching with respect to* λ (the two conditions are mutually exclusive, by definition). Notice that the edges of a non–crossing matching can be totally ordered by increasing values of the indices of left (or right) nodes of the edges in it: in fact, two distinct edges (l_i,r_j), $(l_h,r_k)\in E$ share a common node if and only if either $i=h$ or $j=k$, and cross each other in the plane if and only if either $i<h$ and $j>k$ or $i>h$ and $j<k$; thus, for any pair of non–intersecting edges (l_i,r_j), (l_h,r_k) one has that $i<h$ implies $j<k$. In figure 1, two different layouts λ and λ' for a graph G are drawn, and a subset M of edges is shown, which is a non–crossing matching w.r.t. λ, but not w.r.t. λ'.

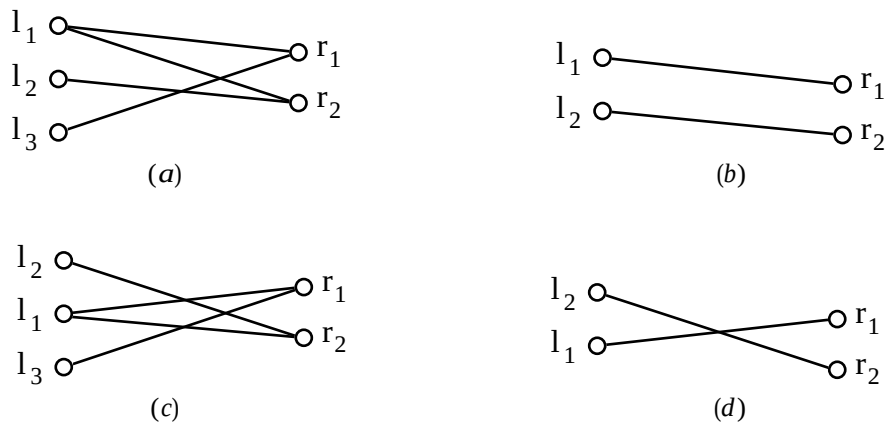

Figure 1
(*a*) a pair (G,λ) and (*b*) a subset M of edges which is a non–crossing matching w.r.t. λ;
(*c*) a pair (G,λ') and (*d*) subset M of edges which is not a non–crossing matching w.r.t. λ'.

The problem we deal with is stated as follows:

P: PARTITION INTO NON–CROSSING MATCHINGS:

Given: a pair (G,λ), where $G=(L\cup R,E)$ is a bipartite graph and λ is a *layout* of it,

4.

Find: a partition $<E_1,E_2, \ldots, E_{x(G,\lambda)}>$ of the edge set E into the minimum number $x(G,\lambda)$ of non–crossing matchings.

Let $\Delta(G)$ denote the maximum degree of a node in the given graph G, that is, $\Delta(G) = \max \{v \in V, deg(v)\}$and $\chi'(G)$ denote the chromatic index of G, that is, the minimum number of colours needed to color the edges of the graph in such a way that adjacent edges receive different colours. The problem definition implies immediately that $x(G,\lambda) \geq \chi'(G) \geq \Delta(G)$ on arbitrary graphs. On bipartite graphs, the relation can be enforced to $x(G,\lambda) \geq \chi'(G) = \Delta(G)$, applying the König's edge colouring theorem. Finally note that a colouring of the edges of a bipartite graph which makes use of $\chi'(G)$ colours can be found in polynomial time by means of a matching algorithm.

The following lower bound also hold for $x(G,\lambda)$: $x(G,\lambda) \geq \lceil |E| / \alpha_2(G,\lambda) \rceil \geq \lceil |E| / \alpha_1(G) \rceil$ where $\alpha_1(G)$ denotes the cardinality of a maximum (classical) matching of G (which does not depend on λ), and $\alpha_2(G,\lambda)$ denotes the cardinality of a maximum non–crossing matching of G (which does depend on λ). In figure 2, an optimum solution of the problem for the pair (G,λ) of figure 1*a* is shown.

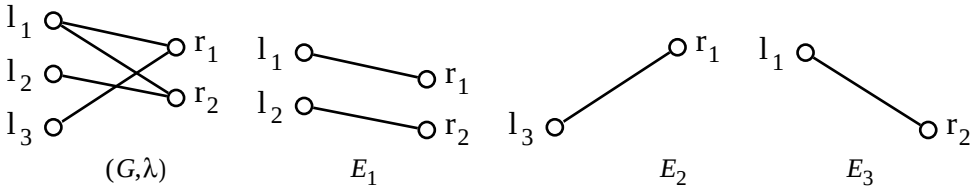

Figure 2
Left to right, the pair (G,λ) of fig. 1*a* and the sets E_1,E_2,E_3 which form a partition of E into the minimum number of non–crossing matchings.

Problem P can also be interpreted as a particular instance of edge colouring, where the edges of G are to be coloured using a minimum number of colours so that edges which intersect receive different colours. It is immediate to see that any subset E_i in a solution $<E_1,E_2,E_3,\ldots>$ for the problem represents those edges which can be assigned the same colour. In the sequel we shall often use the colouring terminology, where, for example, *colour subset* denotes a subset of the partition. Very well-known results in graph-colouring allows for stating that $x(G,\lambda) \geq \chi'(G) \geq \Delta(G)$, where $\chi'(G)$ represents the chromatic index of G, that is, the minimum number of colours needed to colour the edges of G so that adjacent edges receive different colours. Besides its generale validity, in our case this lower bound happens to coincide with the previous one as, for bipartite graphs, $\chi'(G) = \Delta(G)$ (see for example (Berge, 1973)).

The (classical) maximum cardinality matching problem on bipartite graphs is a very well-known problem, which can be solved in $O(n^{1/2}m)$ (for details, see the annotated bibliography by Dell'Amico and Martello, 1997). The maximum cardinality non–crossing matching on bipartite graphs defined on n nodes is studied in (Malucelli et al., 1993), where an $O(n \log \log n)$ algorithm is proposed for its solution.

Actually, the decision version of problem P can be reduced to DEGREE–1 BOOK EMBEDDING (Wood, 2001), which is stated as follows: given a positive integer k, an arbitrary graph G' and a linear ordering of its vertices along the spine of a book, find an assignment of the edges of G' to k pages of the book so that edges assigned to the same page can be drawn on the page without crossings, and the

number of edges incident to a vertex is at most 1 in each page. To reduce the decision version of P to this last problem it suffices to derive, from the given layout λ of the bipartite graph G in an instance of P in decision form, the following linear order of the vertices $l_1,l_2\ldots,l_{n_L},r_{n_R}\ldots,r_2,r_1$. In (Wood, 2001) an $O(m \log\log n)$ algorithm is described for instances of DEGREE–1 BOOK EMBEDDING on bipartite graphs.

The remaining of the paper is organized as follows: in Section 2 some notations and properties are introduced, in Section 3 an efficient exact algorithm is discussed, and implementation issues are addressed to yield a complexity of $O(m \log\log \min\{n_L,n_R\})$; Section 4 concludes the paper.

2. Notations and properties

The *incidence matrix* $M(G,\lambda) = [m_{ij}]$ of the bipartite graph G is a (0,1)–matrix with n_L rows and n_R columns, which are in one–to–one correspondence with the left and right nodes of G, respectively, and follow (from top to bottom and from left to right, respectively) the same order imposed on the nodes. As for its elements, we have that $m_{ij}=1$ if and only if $(l_i,r_j)\in E$, and $m_{ij}=0$ otherwise.

The structure of $M(G,\lambda)$ reflects the intersections between edges in a very immediate way. Any two distinct edges (l_i,r_j), $(l_h,r_k)\in E$ which are incident on a same node $l_i=l_h$ ($r_j=r_k$, respectively) correspond to the 1–entries on a same row $i=h$ (resp., column $h=k$). Any two distinct edges (l_i,r_j), $(l_h,r_k)\in E$ which intersect each other in the plane correspond to two distinct 1–entries m_{ij} and m_{hk} such that $i<h$ and $j>k$ or $i>h$ and $j<k$. Like we did for pairs of edges, we shall say that two distinct 1–entries m_{ij} and m_{hk} *intersect* if either $i\le h$ and $j\ge k$ or $i\ge h$ and $j\le k$. These relations suggest to introduce the following

Definition 1: *Be (i,j) a pair of row and column indices of $M(G,\lambda)$, respectively. Then $NE(i,j)$ denotes the set of all pairs $(h,k)\neq(i,j)$ of (row and column) indices verifying $h\le i$ and $k\ge j$, and, symmetrically, $SW(i,j)$ denotes the set of all pairs $(h,k)\neq(i,j)$ verifying $h\ge i$ and $k\le j$.*

Clearly, for any two distinct pairs (i,j), (h,k), it is the case that $(i,j)\in \mathrm{NE}(h,k)$ if and only if $(h,k)\in \mathrm{SW}(i,j)$.

From the discussion above, we immediately derive that if $m_{hk}=1$, the 1–entries corresponding to pairs in $\mathrm{NE}(h,k)\cup\mathrm{SW}(h,k)$ identify all the edges which intersect edge $(l_h,r_k)\in E$ (in a node or in the plane). Thus, any non–crossing matching of G corresponds to a set of 1–entries of $M(G,\lambda)$ none of which belongs to the set $\mathrm{NE}(\cdot,\cdot)$ of some other element in the set. The viceversa is also true, and allows for establishing a lower bound for $x(G,\lambda)$. Consider a set A of pairs of row and column indices of 1–entries of $M(G,\lambda)$ such that any two distinct $(i,j),(h,k)\in A$ verify $(i,j)\in \mathrm{NE}(h,k)\cup\mathrm{SW}(h,k)$. Since, by definition, all edges corresponding to the elements of A are mutually intersecting and no two of them can be assigned to the same colour subset, we may write $x(G,\lambda)\ge |A|$.

The elements of such a set A do always admit a total order, in the sense that it is possible to order them in a sequence by increasing values of the row index (note that the same sequence can be obtained by ordering the elements of A by decreasing values of the column index, and that this fact does not hold for arbitrary sets of pairs of row and column indices of 1–entries of $M(G,\lambda)$). This suggests the following definition.

Definition 2: *A NE–sequence $S = \langle(i_1,j_1), (i_2,j_2), \ldots, (i_{|S|},j_{|S|})\rangle$ is a totally ordered set of $|S|$ pairs of row and column indices corresponding to 1–entries of $M(G,\lambda)$ such that (i_x,j_x) precedes (i_y,j_y) in the sequence if and ony if $(i_y,j_y)\in NE(i_x,j_x)$.*

That is, $i_1 \geq i_2 \geq i_3 \geq \ldots$ (as well as $j_1 \leq j_2 \leq j_3 \leq \ldots$) and $NE(i_1,j_1) \supset NE(i_2,j_2) \supset NE(i_3,j_3) \supset \ldots$.

As a consequence, the proposed lower bound for $x(G,\lambda)$ may be written $x(G,\lambda) \geq \max\{|S|$, for any NE–sequence $S\}$. The size of a (longest) NE–sequence of $M(G,\lambda)$ can be computed by making use of a directed graph D whose vertices are in one to one correspondence with the 1-entries of M (that is, with the edges in E), and an arc connects the vertex corresponding to m_{hk} to the vertex corresponding to m_{ij} if and only if $(h,k) \in NE(i,j)$. Since any NE–sequence corresponds to a directed path in D (and viceversa), the searched value is exactly the size of a directed path with a maximum number of arcs in D. Since D is acyclic, this task can be accomplished in $O(|E_D|)$, where $|E_D| \leq m^2$ is the number of edges of D. This process yields a lower bound to $x(G,\lambda)$ but does not help in finding a solution to problem P. In the next Section we propose an algorithm which provides an optimum solution to the problem.

It is worth noticing that the pairs (i,j) of row and column indices of the 1–entries m_{ij} which lay on a same anti-diagonal of $M(G,\lambda)$, that is those which verify $i+j=k$, for k=2 to n_L+n_R, can be ordered by decreasing value of row index (or by increasing value of column index) so as to form the NE–sequence $S(k) = <(i_1,k-i_1), (i_2,k-i_2), \ldots,(i_S,k-i_S)>$. Thus, the quantity $\max\{|C(k)|$, k=2 to $n_L+n_R\}$ represents a lower bound for $x(G,\lambda)$.

As far as upper bounds on the number $x(G,\lambda)$ are concerned, a trivial one is given by the number of edges in the graph, that is, $x(G,\lambda) \leq |E| \leq n_L * n_R$. Slightly better ones can be derived by analyzing the structure of $M(G,\lambda)$. Consider the set $D(k)$ of the pairs (i,j) of row and column indices of the 1–entries m_{ij} which verify $i-j=k$, for $k=1-n_R$ to n_L-1. All the entries corresponding to pairs in $D(k)$ lay on a same diagonal, and the corresponding edges form a non–crossing matching of G, as they do not intersect. Therefore it is possible to assign them to the same colour set. Since the number of diagonals $D(k)$ is one less than the number of nodes, we get that $x(G,\lambda) \leq n_L+n_R-1$. Since, clearly, there is no need to assign a colour to an empty set, this upper bound can be further tightened on the specific instance, yielding $x(G,\lambda) \leq n_L+n_R-1-d_0$, where d_0 is the number of sets $D(k)$ which are empty, that is $d_0 = |\{D(k)=\emptyset, k=n_L-1,\ldots,1-n_R\}|$. By assigning a different color to the 1-entries corresponding to pairs in each non-empty subset $D(k)$ we clearly get a feasible but not necessarily optimal partition of E into non–crossing matchings. All upper bounds are immediately computed in $O(m)$ time while reading the graph.

3. The algorithm

In this Section we describe an exact almost linear algorithm for problem P. We shall call *candidate edge of node* $r_j \in R$ the uncoloured edge connecting r_j with the highest indexed node $l_i \in L$, if any.

ALGORITHM

At the beginning all edges are uncoloured, and all colour sets E_k are empty. Be k=1. The colour set E_k is built this way. We examine the candidate edge of nodes r_1 to r_{n_R} as follows: if the candidate edge of node r_i does not intersect (nor in a node nor in the plane) any edge already inserted E_k, we insert it into (colour) set E_k, and mark it as coloured, otherwise we leave it for further insertion. When the candidate edge of node r_{n_R} has been examined, we start constructing the next colour set (i.e. update k to k+1), until all edges are coloured.

Testing whether the current candidate edge intersects the current E_k is easily conducted recalling that any two distinct edges (l_i,r_j), (r_h,l_k) do not intersect if and only if both $i<h$ and $j<k$. Define I_k, to be the index the left node of the last edge inserted into E_k: the current candidate edge (l_y,r_i) of node

r_i does not intersect any edge already inserted into the current colour set E_k, if $y > I_k$. If this is the case, (l_y, r_i) is inserted into E_k, and I_k is updated to y, otherwise we proceed examining the candidate edge of r_{i+1}, if any.

Theorem 3: *The subsets $E_1, E_2, E_3, \ldots$ output by ALGORITHM form a feasible solution for Problem P.*

Proof: The subsets $E_1, E_2, E_3, \ldots$ are a feasible solution for Problem P if each of them is a non–crossing matching and if, alltogether, they form a partition of E, that is if $E_1 \cup E_2 \cup E_3 \cup \ldots = E$ and $E_i \cap E_j = \emptyset$ for any $i \neq j$. We shall prove both claims.

The first claim follows by construction: for each node r_j, ALGORITHM inserts into E_k at most one edge, the candidate one, if and only if it does not intersect (nor in a node, nor in the plane) any other edge already inserted into E_k. Thus, each colour subset E_k is a non–crossing matching for the given bipartite graph G.

The second claim also follows by construction. Whenever ALGORITHM inserts an edge into a colour set it never removes it. Since ALGORITHM terminates only when all edges are coloured, we get $E_1 \cup E_2 \cup E_3 \cup \ldots = E$. On the other hand, as soon as a candidate edge is inserted into a colour set it is marked as coloured, and it will never be a candidate edge again. Since ALGORITHM only inserts candidate edges into a colour set, we get $E_i \cap E_j = \emptyset$ for any $i \neq j$, and the theorem follows. o

Theorem 4: *The solution $<E_1, E_2, \ldots, E_q>$ output by ALGORITHM is a partition into the minimum number $x(G,\lambda)$ of colour sets, that is $q = x(G,\lambda)$.*

Proof: For the sake of simplicity we shall refer to the matrix representation of the graph, where edge $(l_i, r_j) \in E$ corresponds to the pair of row and column indices (i,j). We claim that for any $(h,k) \in E_\alpha$ there exists at least one pair $(i,j) \in E_{\alpha-1}$, denoted $Pred(h,k)$, such that $(h,k) \in \mathrm{NE}(Pred(h,k))$, for $\alpha = 2, \ldots, q$. If this was not the case, in fact, ALGORITHM would have inserted element (h,k) into colour subset $E_{\alpha-1}$. Consider any node $(i,j) \in E_q$, and consider Pred(i,j), Pred(Pred(i,j)), etc. until Pred(Pred(…(Pred(i,j))…)) $\in E_1$. Then, by construction, <Pred(Pred(…(Pred(i,j))…)), …, Pred(Pred(i,j))), Pred(i,j), (i,j)> is a NE–sequence. Since, by definition, $(h,k) \in E_\alpha$ implies Pred$(h,k) \in E_{\alpha-1}$, in this NE–sequence there is exactly one element from each colour subset, that is, its length is exactly q. Since max $\{|S|$, for any NE–sequence $S\}$, is a lower bound for $x(G,I)$, as discussed in Section 2, the claimed thesis follows. o

3.1. Implementation aspects

A thorough implementation of the algorithm requires $O(m\, x(G,\lambda))$, since, in the worst case, we test for insertion each edge in all colour sets. However the algorithm can be implemented in a more efficient way.

Instead of constructing each colour set through the examination of the candidate edge of all the right nodes r_1 to r_{n_R}, we process all the edges of each right node and assign them to the correct colour set. To do this efficiently, we make use of an array $Last(\cdot)$ whose i^{th} component represents the index corresponding to the left node of the last edge assigned to colour set i. Note that the number of components of $Last(\cdot)$ is bounded from above by $n_L + n_R - 1 - d_0 \ldots$, which is in fact an upper bound for the minimum number $x(G,\lambda)$ of classes of a partition of E into non-crossing matchings, as discussed in Section 2).

At the beginning, all the components of $Last(\cdot)$ are set to zero. We start by considering the left nodes of all edges incident on node r_1, in decreasing order $l_{i_1}, l_{i_2}, l_{i_3}, \ldots$ (that is $i_1 > i_2 > i_3 > \ldots$). We

assign edge (l_{i_1},r_1) to colour set E_1 and update $Last(1):=i_1$; then we assign edge (l_{i_2},r_1) to colour set E_2 and update $Last(2):=i_2$; then we assign edge (l_{i_3},r_1) to colour set E_3 and update $Last(3):=i_3$; we proceed this way until all the edges incident on r_1 are coloured. We then consider the left nodes of all edges incident on node r_2, in decreasing order l_{i_1}, l_{i_2}, l_{i_3},... (that is $i_1>i_2>i_3>\ldots$): we search for lowest index y_1 such that $Last(y_1) < i_1$, we assign edge (l_{i_1},r_2) to colour set E_{y_1} and update $Last(y_1):=i_1$; then we search for lowest index y_2 such that $Last(y_2) < i_2$, we assign edge (l_{i_2},r_1) to colour set E_{y_2}, and update $Last(y_2):=i_2$; then we search for lowest index y_3 such that $Last(y_3) < i_3$, we assign edge (l_{i_3},r_1) to colour set E_{y_3}, and update $Last(y_3):=i_3$; and so on until all the edges incident on r_2 are coloured. We then process (the edges incident on) nodes r_3 to r_{n_R} like we did for r_2.

The complexity of our algorithm depends on the data structure we adopt to search and maintain the array $Last(\cdot)$. In particular, if the test for searching the correct index of the colour set into which inserting an edge is performed as a binary search on $Last(\cdot)$, or if the components of $Last(\cdot)$ are organised in a balanced tree, the computational complexity amounts to $O(m \log x(G,\lambda))$.

Better computational complexity are obtained by making use of more sophisticated data structures, like, for example, the Bounded Dictionary data structure (Mehlhorn and Näher, 1990) or the priority queue data structure (van Emde Boas, 1977). These data structures allow to perform set manipulation operations and queries on a subset S of the integers in the range $[1,N]$ in $O(\log\log N)$ time and $O(N)$ space. Since in our case the elements of $Last(\cdot)$ belong to the range $[1,n_R]$ and for each edge we perform a search on the elements of $Last(\cdot)$, the computational complexity of the proposed algorithm in this implementation amounts to $O(m \log\log n_R)$. The algorithm can be applied exchanging the role of rows and columns, giving a final complexity of $O(m \log\log \min\{n_L,n_R\})$.

Another way for solving our optimization problem P consists in considering its decision version and applying a dichotomic search on the values of the objective function, as lower and upper bounds are known for it. As observed in the Introduction, the decision version of P reduces to DEGREE–1 BOOK EMBEDDING, thus we can apply the $O(m \log\log n)$ algorithm described in (Wood, 2001), obtaining an overall complexity of $O(m \log n \log\log n)$, which is dominated by the complexity of our algorithm.

Example: Consider the bipartite graph which has the following as incidence matrix $M(G,\lambda)$

	r_1	r_2	r_3	r_4	r_5
l_1	1	1	0	0	0
l_2	1	1	1	0	0
l_3	1	0	0	0	0
l_4	0	1	1	1	0
l_5	0	0	0	0	1
l_6	1	0	1	0	1

The last upper bound discussed in Section 2 ensures that $n_L+n_R-1-d_0$ colors will suffice. This result allows for fixing the size of the array $Last(\cdot)$ to 6 (in fact, $n_L=6$, $n_R=5$, and $d_0=4$, as $D(k)=\emptyset$, for $k=4,-2,-3,-4$).

The algorithm starts initializing all colour sets E_k to empty, and all the components of $Last(\cdot)$ to zero. Then it assigns edge (l_6,r_1) to colour set E_1 and updates $Last(1):=6$; then edge (l_3,r_1) to colour set E_2 and updates $Last(2):=3$; then edge (l_2,r_1) to colour set E_3 and updates $Last(3):=2$; finally, it assigns

edge (l_1,r_1) to colour set E_4 and updates $Last(4):=1$. In this way, one gets $Last=(6,3,2,1,0,0)$. The algorithm then processes the edges incident on node r_2: as for edge (l_4,r_2), it searches for the lowest indexed component with value smaller than 4, which is $Last(2)$, it assigns edge (l_4,r_2) to colour set E_2, and updates $Last(2):=4$; as for edge (l_2,r_2), the algorithm searches for the lowest indexed component with value smaller than 2, which is $Last(4)$, it assigns edge (l_2,r_2) to colour set E_4, and updates $Last(4):=2$; finally, as for edge (l_1,r_2) it searches for the lowest indexed component with value smaller than 1, which is $Last(5)$, it assigns edge (l_1,r_2) to colour set E_5, and updates $Last(5):=1$. By doing so, one gets $Last=(6,4,2,2,1,0)$. The algorithm continues iterating, so that after having processed the edges incident on r_3 one has $Last=(6,6,4,2,2,0)$, after having processed the edges incident on r_4 one has $Last=(6,6,4,4,2,0)$, and finally, after having processed the edges incident on r_5 one has $Last=(6,6,6,5,2,0)$.

The optimal solution is $<E_1,E_2,E_3,E_4,E_5> = <\{(l_6,r_1)\}, \{(l_3,r_1),(l_4,r_2),(l_6,r_3)\}, \{(l_2,r_1),(l_4,r_3),(l_6,r_5)\}, \{(l_1,r_1),(l_2,r_2),(l_4,r_4),(l_5,r_5)\}, \{(l_1,r_2),(l_2,r_3)\}>$, and makes use of $x(G,\lambda)=5$ colours.

Note that the lower bounds proposed in the Introduction would have given a value of 4 (in fact $|E|=13$, $\chi'(G)=\Delta(G)=4$, $\alpha_1(G)=5$ and $\alpha_2(G,\lambda)=4$), the lower bound computed by means of the sets $C(k)$ is 2 (achieved for $k=3,4,7$), and that the tigther upper bound has value 6, as observed above.

4. Conclusions and open problems

In this paper we proposed an exact $O(m \log\log \min\{n_L,n_R\})$ algorithm for partitioning the edge set of a bipartite graph into the minimum number $x(G,\lambda)$ non–crossing matchings. The problem corresponds to the optimization version of DEGREE–1 BOOK EMBEDDING (Wood, 2001), which is solved (in decision version) in $O(m \log\log n)$ time on graphs with m edges and n vertices.

Extensions of the studied problem lead to still open problems or to well–studied ones.

One is the following: given a pair (G,λ), where $G=(L\cup R,E)$ is a bipartite graph and λ is a *layout* of it, and a positive integer $b(v)$ for each $v\in L\cup R$, find a partition $<E_1,E_2, \ldots, E_{x(G,\lambda)}>$ of the edge set E into the minimum number $x_b(G,\lambda)$ of subsets such that, for each E_k, no two edges belonging to E_k cross in the plane, and no more than $b(v)$ edges in E_k are incident on v, for each $v\in L\cup R$. This problem is studied in (Malucelli and Nicoloso, 2001), where an efficient exact algorithm is proposed.

Another extension leads to the so–called *pagenumber* $t(G)$ of a graph G, which is stated as follows: given a graph G, find a linear order of the nodes and a partition $<E_1,E_2, \ldots, E_{t(G)}>$ of the edge set E into the minimum number $t(G)$ of subsets such that the edges in a subset do not (pairwise) cross in the plane. Notice that we are allowed to assign to the same subset more than one edge from each vertex. This problem is well–studied: see for example (Shahrokhi and Shi, 1998) or, if G is bipartite, (Muder et al., 1988).

As already noticed, $x(G,\lambda)$ depends on the bipartite graph and on the given layout. In particular, a change in the layout changes, generally speaking, the cardinality $x(G,\lambda)$ of the partition. In this sense, a possible extension consists in the following: given a family F of distinct layouts of a bipartite graph $G=(L\cup R,E)$, find one, say λ^*, which verifies $x(G,\lambda^*) = \min\{x(G,\lambda), \lambda\in F\}$. Observe that given the total order of the nodes in one set of a bipartite graph, finding a total order of the nodes in the other set so that the number of edges which intersect each other in the plane is minimum is an NP–complete problem (Demetrescu and Finocchi, 2000 and 2001).

If we keep the layout fixed, we may also want to find, among the matchings of maximum cardinality, one with the minimum number of edge crossings (Blasum et al., 1999).

Other variations of the problem discussed in the paper consist of bounding the cardinality of the colour sets, or considering an arbitrary graph and a given layout of it.

10.

Acknoledgement

The authors wish to thank Prof. David R. Wood for the helpful discussions.

References

Berge, C., *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973.

Blasum, U., M.R. Bussieck, W. Hochstättler, H.-H. Scheel, and T. Winter, "Scheduling Trams in the Morning", Mathematical Methods of Operations Research 49,1 (1999), 137–148.

Dell'Amico, M., and S. Martello, "Linear Assignment", in Annotated Bibliographies in Combinatorial Optimization, Dell'Amico, M., F. Maffioli, and S. Martello, editors, Wiley 1997.

Demetrescu, C. and I. Finocchi, "Removing Cycles for Minimizing Crossings", Technical Report ALCOMFT–TR-01-148, ALCOM–FT EU Programme IST–1999-14186, May 2001, available at the URL: http://www.brics.dk/cgi–alcomft/db.

Demetrescu, C. and I. Finocchi, "Break the 'Right' Cycles and Get the 'Best' Drawing", Proceedings of the 2nd Int. Conf. on Algorithms and Experimentations (ALENEX'00), San Francisco, CA, January 2000, 171–182.

De Werra, D., "Decomposition of Bipartite Multigraphs into Matchings", Zeit. Oper. Res. 16 (1972), 85-90.

Malucelli, F., and S. Nicoloso , "Optimal Partition of a Bipartite Graph into Non–Crossing *b*–Matchings", manuscript, 2001.

Malucelli, F., T. Ottmann, and D. Pretolani, "Efficient Labelling Algorithm for the Maximum Non Crossing Matching Problem", Discrete Applied Mathematics 47 (1993), 175–179.

Mehlhorn, K., and S. Näher, "Bounded Ordered Dictionaries in $O(\log \log N)$ Time and $O(n)$ space", Information Processing Letters 35 (1990), 183–189.

Muder, D.J., M. L. Weaver, and D. B. West, "Pagenumber of Complete Bipartite Graphs", Journal of Graph Theory 12,4 (1988), 469–489.

Shahrokhi, F., and W. Shi, "On Crossing Sets, Disjoint Sets and the Pagenumber", DIMACS Technical Report 98-46, October 1998.

van Emde Boas, P., "Preserving Order in a Forest in Less than Logarithmic Time and Linear Space", Information Processing Letters 6,3 (1977), 80–82.

Wood, D.R., "Bounded Degree Book Embeddings and Three–Dimensinal Orthogonal Graph Drawing", Proceedings of the Graph Drawing Conf. GD2001, Wien, September 2001, in Lecture Notes in Computer Science, Springer Verlag, to be printed.