Univ. u Novom Sadu
Zb. Rad. Prirod.–Mat. Fak.
Ser. Mat. 21, 2 (1991), 161-173

Review of Research
Faculty of Science
Mathematics Series

A CONSTRUCTION OF NON-ISOMORPHIC "SMALL" BIPARTITE GRAPHS [1]

Dragan M. Acketa, Zoran Budimac, Ratko Tošić
Institute of Mathematics, University of Novi Sad
Trg Dositeja Obradovića 4, 21000 Novi Sad, Yugoslavia

Abstract

All the non-isomorphic bipartite graphs with at most 11 vertices are constructed by using two independent algorithms based on two different representations of these graphs.

AMS Mathematics Subject Classification (1980): 05C25,68C05
Key words and phrases: (non)-isomorphic, bipartite graphs, algorithms, catalogues.

1. Preliminaries

We assume familiarity with very basic notions of graph theory. Non-defined notions can be found in [2]. By a "graph" we shall always mean a simple non-oriented graph.

A bipartite (m,n)–graph is a graph G with the property that the vertex-set of G can be partitioned into two (basic) vertex-sets M and N of cardinality m and n respectively, such that each edge of G has one vertex in M and the other one in N. The graph G can be considered at the same time as the reversed bipartite (n,m)–graph, which is obtained by interchanging the roles of the sets M and N in the representation of G.

[1] This research was supported by Science Fund of Serbia

We restrict our construction to connected bipartite graphs. Non-isomorphic disconnected bipartite graphs can be easily enumerated by applying the formulae given in our last section.

We proceed with two algorithms (both sketched in pseudoPascal), which are based on two different representations of bipartite graphs. In both cases we primarily define the canonicity conditions for the representations, provided that $m \neq n$ (a canonical representation is the unique representative of a class of mutually isomorphic bipartite graphs). The case $m = n$ requires an additional test, since the reversed bipartite (n,m)–graph should be also taken into account when determining canonicity (exactly one of the two possibly different mutually reversed representations should be kept).

Sets are denoted without brackets and commas.

2. The first algorithm

2.1. Representation

Given a bipartite (m,n)–graph G ($m \leq n$), let the vertices of the smaller basic vertex-set M be denoted by $1,...,m$ respectively. The graph G is represented by the family $F = \{S[1],...,S[n]\}$ of subsets of the set $\{1,...,m\}$, such that the subset $S[j]$, $1 \leq j \leq n$, consists of those vertices of M, which are adjacent to the j–th vertex of the (possibly) larger basic vertex-set N.

Such a representation of a bipartite graph is called *standard* if it is "triply sorted", i.e., if the following three conditions hold:

a) Each subset $S[j]$ (= a combination without repetitions) is an increasing sequence of integers.

b) Let $F(i)$, $1 \leq i \leq m$, denote the subfamily of F consisting of subsets $S[j]$ of cardinality i. The families $F(i)$ are listed one after another in decreasing order of i.

c) The sets within each $F(i)$ are listed in the usual lexicographic order.

A standard representation F is called *minimal* if there does not exist a permutation p of $\{1,...,m\}$, such that $p(F)$ after standardization (i.e.,

after the sorting in accordance with the rules a), b), c)) is lexicographically LESS than F.

The representation of the same bipartite graph G, considered as the reversed bipartite (n,m)–graph, is the family $F' = \{S'[1],...,S'[n]\}$ of subsets of the set $\{1,...,n\}$, where the i–th vertex of M belongs to the set $S[j]$ if and only if the j–th vertex of N (denoted by j) belongs to the set $S'[i]$, for $1 \leq i \leq m$ and for $1 \leq j \leq n$.

Given $m < n$, the representation of a bipartite graph is *canonical* if and only if it is a minimal standard. If $m = n$, then special attention should be paid to the bipartite graphs with self-reversable representations. A family F is a self-reversable representation if it remains invariant after application of the operations "reverse the representation" and "find the corresponding minimal standard representation" to F. All the other bipartite (n,n)–graphs have two different minimal standard representations. Thus the exact number of non-isomorphic bipartite (n,n)–graphs is equal to

0.5 · (the number of generated minimal standard representations
+ the number of self-reversable among these representations).

2.2. Algorithm

```
(* main *)
  READ  (m, n); (m ≤ n)
  Generate  Group  of all the permutations of the set {1,...,m};
  FOR  each combination  Comb  with repetitions, of length  n,
            composed of elements  1,...,m     DO BEGIN
       Let  Frequency  (i),  1 ≤ i ≤ m, denote
       the number of appearances of element  i  in the combination  Comb;
       Construct_1 ( m, n, Group )
  END;
```

(* The elements of the combination `Comb` are the cardinalities of the sets $S[i]$ in the representations of bipartite graphs. The above call of the recursive procedure `Construct_1` generates all non-isomorphic bipartite (m,n)–graphs with the same combination `Comb`. The procedure `Construct_1` handles the parameters `Comb` and the therefrom derived `Frequency` by using the side effect *)

```
PROCEDURE Construct_1 ( Maxcard, Remains, Temporary_group );
  (* Let Family denote a combination WITH repetitions of length
    Frequency(Card) of combinations WITHOUT repetitions of
    length Card of elements 1,...,m *)
  BEGIN (* Construct_1 *)
    IF Remains > 0 THEN BEGIN
      FOR each Card (*inality*)
        from Maxcard downto 1 DO
        IF Frequency (Card) > 0 THEN
          FOR each Family DO
            IF Family is lexicographically the first with
                respect to Group THEN BEGIN
              Append Family to Great_family;
              Cut Temporary_group on the basis of Family
                (the result is Cut_group);
              New_remains := Remains - Frequency(Card);
              Construct_1 ( Card, New_remains, Cut_group )
                (* the recursive call *)
            END (* IF Family ... *)
    END
    ELSE (* if Remains = 0 *)
      IF the bipartite (m,n)-graph represented by
          Great_family is connected THEN
        IF m < n THEN
          print( Great_family )
        ELSE (* if m = n *) BEGIN
          Generate Reversed_comb ;
            (* Reversed_comb stands instead of Comb
              after the roles of the basic vertex-sets
              M and N are interchanged *)
          IF Comb is lexicographically LESS than
              Reversed_comb THEN
            print( Great_family )
          ELSE
            IF Comb = Reversed_comb THEN BEGIN
              Generate Reversed_great_family;
                (* the minimal standard representation
                  of the reversed variant of the same
```

```
                    bipartite (n,n)- graph *)
                IF    Reversed_great_family  coincides
                  with  Great_family
                  (* the self-reversable case *)  OR
                  OR   Great_family  is lexicographically
                  LESS than  Reversed_great_family
                THEN
                  print( Great_family  )
              END  (*  Comb = Reversed_comb  *)
          END  (*  m = n  *)
END; (*  Construct_1  *)
```

2.3. Remarks and hints

`Great_family` is initialized separately for each combination of set cardinalities. The recursive calls of the procedure `Construct_1` on deeper levels change only the "tail" (= the last several sets) of `Great_family`.

"Cutting a group on the basis of a family" means the extraction of merely those permutations from the group, which fix the family (i.e., the automorphisms of the family). The subfamilies consisting of equicardinal sets may be treated independently during cutting the automorphism group. The automorphism of `Great_family` is the intersection of the automorphism groups of such subfamilies.

We proceed with some hints which have reduced the necessary computing time:

The main saving of the computer time is gained by choosing the SMALLER of the two basic vertex-sets of a bipartite graph to be the ground-set of the representing family. As a consequence, the bipartite graphs on at most 11 vertices can be constructed by using the permutation groups of maximal size not greater than $5! = 120$.

The lexicographical test within the procedure `Construct_1` is considerably shortened by applying the cutting of the automorphism group. Thus we should apply very few permutations in the last stages of recursion.

The lexicographical test should be completely avoided when `maxcard` $= m$. The family of sets of cardinality m is necessarily lexicographically the first.

The connectivity test can be completely avoided when the representing family contains a set of cardinality m. The affirmative answer is guaranteed in that case.

Most of the superfluous bipartite (n,n)–graphs are eliminated at the "combination level". The same bipartite graphs considered as reverse bipartite graphs are effectively constructed merely in the cases when some two mutually reversed combinations coincide.

2.4. Example

Let $m = n = 4$ and `Comb` $= 1123$, which implies that `Frequency` $= 2110$ (there should be two sets in `Great_family` of cardinality 1 and one set of cardinalities 2 and 3 respectively). Suppose that the first two sets chosen to `Great_family` are 123 and 14 respectively. Note that `Group` is reduced from 24 permutations to 6 permutations after addition of the set 123 to the empty `Great_family` and to only 2 permutations (the identical permutation and the transposition (23)) after addition of the set 24. Now there are 7 lexicographically minimal (w.r.t. the last Group) possibilities for `Family` consisting of two one-element sets: $\{1,1\}$, $\{1,2\}$, $\{1,4\}$, $\{2,2\}$, $\{2,3\}$, $\{2,4\}$ and $\{4,4\}$. If m were different from n, then we would test the corresponding `Great_family` only w.r.t. connectivity. However, in this case we should also consider the reversed representation (obtained by interchanging the sets M and N).

We give a list of the corresponding 7 representations of bipartite graphs, each of which is followed by its `Comb` and `Reversed_comb` (these combinations can be viewed as the degree sequences of the vertices in M and N respectively). All these representations have already passed the connectivity test.

num.	representation	Comb	Reversed_comb
1)	123 – 14 – 1 – 1	1123	1114
2)	123 – 14 – 1 – 2	1123	1123
3)	123 – 14 – 1 – 4	1123	1123
4)	123 – 14 – 2 – 2	1123	1123
5)	123 – 14 – 2 – 3	1123	1222
6)	123 – 14 – 2 – 4	1123	1222
7)	123 – 14 – 4 – 4	1123	1123

The representation 1) is rejected at "combination level", since

`Reversed_comb` is lexicographically less than `Comb`. Due to the opposite reason, our algorithm decides that representations 5) and 6) should be kept in the final catalogue of non–isomorphic bipartite graphs. The remaining four representations, which satisfy that `Comb = Reversed_comb`, should be reversed and lexicographically minimized w.r.t. the rules a),b),c). If we adopt that the vertices of the set N are numerated by 1,2,3,4 in an order which is in accordance with the order of sets in `Great_family`, then we have:

	initial family	reversed family	minimal standard representation after reversing
2)	123 – 14 – 1 – 2	123 – 14 – 1 – 2	123 – 14 – 1 – 2
3)	123 – 14 – 1 – 4	123 – 14 – 1 – 24	123 – 14 – 2 – 2
4)	123 – 14 – 2 – 2	12 – 134 – 1 – 2	123 – 14 – 1 – 4
7)	123 – 14 – 4 – 4	12 – 1 – 1 – 234	123 – 14 – 4 – 4

It follows that the representations 2) and 7) are self- reversable (thus both of them should be kept), while the representations 3) and 4) correspond to isomoprhic bipartite graphs (we keep 3, since it is lexicographically less). As a conclusion, there are five non-isomorphic bipartite graphs with `Comb` = 1123,
`S[1]` = 123, `S[2]` = 14. Their representations are numerated above by 2), 3), 5), 6), 7) respectively.

3. The second algorithm

3.1. Representation

A bipartite (m,n)–graph G ($m \leq n$) is represented by the $m \times n$ $0-1$ matrix `G[i,j]` satisfying

`G[i,j]` = 1 if and only if the i–th vertex of the basic vertex-set M is adjacent to the j–th vertex of the basic vertex-set N.

The representation of G considered as the reversed bipartite (n,m)–graph is obtained by transposing the matrix `G[i,j]` .

The matrix code associated to the matrix `G[i,j]` is the natural number with the binary expansion `G[1,1],...,G[1,n],G[2,1],...,G[2,n],...`
`...,G[m,1],...,G[m,n]` . The matrix code C is called *maximal* if

there does not exist any permutation of rows and columns of the matrix such that the code of permuted matrix is GREATER than F. Given $m \neq n$, the representation of a bipartite (m,n)–graph is *canonical* if and only if its code is maximal. If $m = n$, then we should keep that one of the two mutually transposed matrices, which has the larger code (or arbitrary one if the codes are equal).

Construction of bipartite graphs takes place for m and n (cardinal numbers of M and N, respectively) and arrangement of edges adjacent to vertices of N. Arrangement is of the form $r_1, r_2, ..., r_m$, where $r_1 \geq r_2 \geq ... \geq r_m$. Arrangement shows that the k–th vertex of M is adjacent to the r_k vertices of N, where $k = 1, ..., m$.

3.2. Algorithm

```
READ (m,n); ( m ≤ n )
FOR  e := m+n-1 to m·n  DO
    (* e denotes the number of edges *)
  FOR each (non-ordered) partition Part of the number e
      into m summands DO
    Construct_2 ( m, n, Part, 1 );
      (* observe that the partitions Part used here coincide
        with the combinations Comb used with the first algorithm *)
PROCEDURE Construct_2 (m, n, Part, k);
    (* the natural number k is the serial number of the matrix
    row to which the current call of Construct_2 is applied *)
  BEGIN (* Construct_2 *)
    FOR each arrangement of Part[k] 1's and n- Part[k]
        0's in the k-th matrix row DO BEGIN
      Let Temporary_matrix denote the partial representation
          of a bipartite graph, consisting of the first k rows;
        IF k > 2 THEN
          IF NOT Canonical( k, Temporary_matrix ) THEN
            EXIT;
        IF k < m THEN
            (* the representation is not yet constructed *)
          IF Temporary_matrix cannot be completed with
            m - k rows so that a representation of a connected
            bipartite (m,n)-graph is produced THEN EXIT
```

```
            ELSE  (* the representation is completed *)
              IF  the bipartite graph represented
                  by  Temporary_matrix  is connected  THEN
                print( Temporary_matrix )
    END  (* FOR *)
  END;  (*  Construct_2  *)
```

(* The canonicity test for the constructed partial and complete representations is performed by calling the function Canonical, which takes into account the transposed representing matrix as well *)

```
FUNCTION  Canonical ( k, Matrix ): boolean;
  FUNCTION  Can ( k, Matrix ): boolean;
    BEGIN  (*  Can  *)
      Can := false;
      FOR  each permutation of columns of  Matrix  DO
        FOR  each permutation of rows of  Matrix, which
          preserves the sequence  Part[1],...,Part[m]  non-
          increasing  DO
            IF  Code_of_Permuted_matrix  >
              Initial_code  THEN  EXIT;
            (* the canonicity test ceases immediately after
              the first permutation with larger code is found *)
      Can := true
    END; (*  Can  *)
  BEGIN  (*  Canonical  *)
    Canonical  := true;
    Initial_code  :=  Code_of_Matrix;
    IF NOT  Can (k, Matrix)  THEN BEGIN
      Canonical  := false;
      EXIT
    END;
    IF  ( n = m )  AND  ( k = m )  THEN  BEGIN
      (* this is the only case with canonicity test when
        the transposed matrix is necessary *)
      Transpose (Matrix) ;
      Canonical := Can(k, Transposed_matrix)
      (* now the canonicity depends on whether there exists a
        permutation with larger code with  Transposed_matrix  *)
```

```
    END
END;  (* Canonical *)
```

3.3. Some hints

We shall proceed with some hints which have reduced (or will reduce) the computing time of the algorithm:

- When arranging the first row of the matrix ($k = 1$), there is always only one canonical arrangement: the first r_1 positions of the row are filled with r_1 1's – the other positions are filled with 0's.
- In order to reduce computing time, every partially constructed matrix (for $k \geq 3$) is tested. If it cannot produce a connected matrix (graph) in the following steps, it is rejected immediately.
- Function Canonical is the most time-consuming one, because of its nested permutations. However, it is possible to cut short every loop from m! executions to no more than $m_1! \cdot m_2! \cdot m_3! \cdot m_4!$ executions, where $m_1+m_2+m_3+m_4 = m$. In every partial constructed matrix with more than two rows ($k > 2$) is possible to determine at most four successive classes of columns which can be permuted for themselves and combined with permutations in other classes. This feature radically reduces the computing time of the function, especially for $m > 6$.

4. Example

Let $m = 3$, $n = 4$ and e (the number of edges) = 8. The only partitions of ($e =$) 8 into ($m =$) 3 summands are: 4–3–1, 4–2–2 and 3–3–2.

Every partition $i-j-k$ (of the above three), represents all the canonical matrices with i 1's and $4-i$ 0's in the first row, j 1's and $4-j$ 0's in the second row and k 1's and $4-k$ 0's in the third row.

The matrix will be represented as its code combined with '–' showing the ends of the rows.

For the sake of simplicity we shall assume that the first and the second row of every matrix are constructed so that they are already canonical; and

that only the third row is the subject of every possible arrangement and canonicity test.

4–3–1: The part of the code which is "fixed" (i.e. constructed) is 1111–1110– and one 1 and three 0's have to be arranged in the third row.

Arrangement 0001 is the first one generated and it gives the maximal code because no other permutation of columns will produce a greater code than 1111–1110–0001.

Arrangements 0010 and 0100 will not produce the maximal code because there is such a permutation of columns which "preserves" the constructed part of the matrix and gives a greater code. For instance, 1111–1110–1000 is such a code for both cases.

Arrangement 1000 of the third row gives the maximal code 1111–1110–1000.

4–2–2: The part of the code which is constructed is 1111–1100– and two 1's and two 0's have to be arranged in the third row.

All the possible arrangements can be displayed as follows:
1111–1100–0011 maximal, accepted
1111–1100–0101 rejected, 1111–1100–0110 is greater
1111–1100–1001 rejected, 1111–1100–1010 is greater
1111–1100–0110 rejected, 1111–1100–1010 is greater
1111–1100–1010 maximal, accepted
1111–1100–1100 maximal, accepted

3–3–2: The first part of the code which we shall assume constructed is 1110–1101– and two 1's and two 0's have to be arranged in the third row.

All the possible arrangements can be displayed as follows:

1110–1101–0011 maximal, accepted
1110–1101–0101 rejected, 1111–1101–0110 is greater
1110–1101–1001 rejected, 1111–1100–1010 is greater
1111–1100–0110 rejected, 1111–1100–1010 is greater
1110–1101–1010 maximal, accepted
1110–1101–1100 maximal, accepted

The part of the code which we shall assume constructed is 1110–1110– and two 1's and two 0's have to be arranged in the third row.

All the possible arrangements can be displayed as follows:

1110–1110–0011 rejected, 1110–1110–0110 is greater
1110–1110–0101 rejected, 1110–1110–1001 is greater
1110–1110–1001 maximal, accepted
1110–1110–0110 rejected, 1110–1110–1010 is greater
1110–1110–1010 rejected, 1110–1110–1100 is greater
1110–1110–1100 rejected, it is disconnected

5. Results

We primarily give the table of the number of non-isomorphic connected bipartite (m,n)–graphs, for $m \leq n$, $m+n \leq 11$:

m/n	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1
2	2	4	6	9	12	16	20	25	
3		10	34	76	155	290	510		
4			93	558	1824	5375			
5				1897	19687				

Thus the total number $c(k)$ of non-isomorphic connected bipartite graphs on k vertices is for k between 1 and 11 given by:

k	1	2	3	4	5	6	7	8	9	10	11
$c(k)$	0	1	1	3	5	17	44	182	730	4032	25598

The number $t(k)$ of general (both connected and disconnected) non-isomorphic bipartite graphs on ≤ 11 vertices, which have no isolated vertices, can be easily calculated by applying the following well-known formula ([1]):

$$t(k) = \sum_{(d(1),\ldots,d(k))} \prod_{i=1}^{k} \binom{c(i)+d(i)-1}{d(i)} ,$$

where $(d(1),\ldots,d(k))$ is a k–tuple, which uniquely determines a partition of the natural number k by

$$d(1) + 2 \cdot d(2) + \ldots + k \cdot d(k) = k .$$

(such a formula may be applied to any graph property, which is hereditary to its connected components; the property "to be bipartite" is such a property).

Thus we obtain the following table of the number $t(k)$:

k	1	2	3	4	5	6	7	8	9	10	11
$t(k)$	0	1	1	4	6	22	53	215	816	4360	26824

Finally, if the isolated vertices are also allowed, then the total number $u(k)$ of general bipartite graphs on k vertices can be easily calculated as :

$$u(k) = t(k) + t(k-1) + ... + t(2) + 1$$

This gives the following table:

k	1	2	3	4	5	6	7	8	9	10	11
$u(k)$	1	2	3	7	13	35	88	303	1119	5479	32303

References

[1] Acketa,D.: Graphic representations of graphic matroids on 9 elements, Proceedings of 8th Yugoslav Seminar in Graph Theory, (Novi Sad, 1987), Institute of Mathematics, Novi Sad, 1989., 1-30.

[2] Harary, F.: Graph Theory, Addison-Wesley, 1969.

REZIME

JEDNA KONSTRUKCIJA NEIZOMORFNIH "MALIH" BIPARTITNIH GRAFOVA

Konstruisani su svi neizomorfni bipartitni grafovi sa najviše 11 čvorova. Pri konstrukciji su korišćena dva nezavisna algoritma zasnovana na dve različite reprezentacije ovih grafova.

Received by the editors January 19, 1990