Acta Universitatis Sapientiae

Informatica

Volume 10, Number 2, 2018

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Contents

Acta Univ. Sapientiae, Informatica **10,** 2 (2018) 137– 157

DOI: 10.2478/ausi-2018-0008

Estimating clique size by coloring the nodes of auxiliary graphs

Sándor SZABÓ
University of Pécs, Pécs, Hungary
email: sszabo7@hotmail.com

Abstract. It is a common practice to find upper bound for clique number via legal coloring of the nodes of the graph. We will point out that with a little extra work we may lower this bound. Applying this procedure to a suitably constructed auxiliary graph one may further improve the clique size estimate of the original graph.

1 Introduction

A graph is called a finite simple graph if it has finitely many nodes and edges and in addition it does not have any loop or double edge. Let $G = (V, E)$ be a finite simple graph. A subgraph Δ of G is called a clique if each two distinct nodes of Δ are adjacent. If the clique Δ has k nodes we call it a k-clique of G. For each finite simple graph G there is a well defined integer k such that G contains a k-clique but G does not contain any $(k+1)$-clique. This k is called the clique number of G and it is denoted by $\omega(G)$. Each k-clique in G is called a maximum clique of G. (For more background information and applications of the clique problem the reader should consult with [2], [4], [6], [12].)

We color the nodes of G such that each node has exactly one color and adjacent nodes cannot receive the same color. This type of coloring of the nodes of G is called legal coloring. For each finite simple graph G there is a

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C15, 05B45, 52C22
Key words and phrases: clique number, chromatic number, maximum clique legal coloring of the nodes, greedy coloring

well defined integer k such that the nodes of G can be legally colored using k colors but the nodes of G cannot be legally colored using $k-1$ colors. This k is called the chromatic number of G and it is denoted by $\chi(G)$.

It is well-known that the problems of determining $\omega(G)$ or $\chi(G)$ belong to the NP hard complexity class. (See [5].)

Many clique solver algorithms used in practice employ clique size upper estimates to curtail the size of the search space. (See [1], [7], [8], [11], [13], [15], [16].) Since $\omega(G) \leq \chi(G)$ holds it is a common practice to use a greedy coloring procedure to locate a legal coloring of the nodes of G and use the number of colors as an upper estimate for $\omega(G)$. We will point out that with a little more extra work one can reduce this upper bound.

Using the given graph G we construct an auxiliary graph Γ such that an upper estimate for $\omega(\Gamma)$ yields an upper estimate for $\omega(G)$. The new estimate is typically better but it comes for a computationally higher price. We will present two particular instances of such auxiliary graphs.

2 The basic procedure

In this section first we describe a procedure to estimate the clique size of a finite simple graph $G = (V, E)$. For the sake of easier reference we will call the proposed procedure as the method of profiles. As a starting point we legally color the nodes of G. We may use any coloring algorithm. (See [9], [3].) We do not assume that the number of colors we use is optimal. Let $C_1, \ldots, C_\gamma$ be the color classes of the nodes. Set

$$U = C_1 \cup \cdots \cup C_p \quad \text{and} \quad W = C_{p+1} \cup \cdots \cup C_\gamma, \tag{1}$$

where $p = \lfloor \gamma/2 \rfloor$. Let H and K be the subgraphs of G induced by the sets U and W, respectively.

To a node $u \in U$ we assign a quantity $\mathrm{cdeg}(u)$ called the clique degree of u. We form the subgraph L_u induced in G by the subset $N(u) \cap W$ of the nodes of G. Here $N(u)$ is the set of neighbors of u in G. We would prefer to set $\mathrm{cdeg}(u)$ to be $\omega(L_u)$. But computing $\omega(L_u)$ maybe overly time consuming. So we settle for an upper estimate of $\omega(L_u)$. We may use our favorite procedure to find an upper estimate for $\omega(L_u)$.

Analogously, to a node $w \in W$ we assign a clique degree $\mathrm{cdeg}(w)$. We consider the subgraph L_w induced by the set $N(w) \cap U$ and $\mathrm{cdeg}(w)$ is an upper estimate of $\omega(L_w)$.

For the remaining part of the description of the algorithm we assume that the clique degrees of the nodes of G are at our disposal. We define a profile

for the graph H which is a sequence of numbers $\alpha'_1, \ldots, \alpha'_p$. We set

$$\alpha_i = \max\{\mathrm{cdeg}(v) : \ v \in C_i\}, \quad 1 \le i \le p.$$

Then we arrange the numbers $\alpha_1, \ldots, \alpha_p$ into a non-increasing order to get the profile $\alpha'_1, \ldots, \alpha'_p$ of H. In a similar fashion we construct a profile $\beta'_1, \ldots, \beta'_q$ for the graph K, where $q = \gamma - p$. We set

$$\beta_i = \max\{\mathrm{cdeg}(v) : \ v \in C_i\}, \quad p+1 \le i \le \gamma.$$

Finally we list the numbers $\beta_{p+1}, \ldots, \beta_\gamma$ in a non-increasing order to get the profile $\beta'_1, \ldots, \beta'_q$ of K.

After this phase of the algorithm the profiles of the graphs H and K are available. We call an ordered pair

$$(r, s), \quad 0 \le r \le p, \quad 0 \le s \le q \tag{2}$$

qualifying if each of the following inequalities

$$\alpha'_1 \ge s, \ldots, \alpha'_r \ge s \tag{3}$$

$$\beta'_1 \ge r, \ldots, \beta'_s \ge r \tag{4}$$

holds. We do not exclude the $r = 0$ possibility. In the $r = 0$ case the inequalities (4) clearly hold and the condition (3) vacuously satisfied. Similarly, the $s = 0$ possibility is not excluded. When $s = 0$, the inequalities (4) obviously hold and the requirement (4) vacuously satisfied.

We inspect the $(p+1)(q+1)$ ordered pairs (r, s) in (2) in the order

$$(p-i, q), (p-i+1, q-1), \ldots, (p, q-i), \quad 0 \le i \le p+q$$

to find the quantity

$$t = \max\{r + s : \ (r, s) \text{ is qualifying}\}. \tag{5}$$

We claim that $\omega(G) \le t$. We state and prove this result more formally.

Lemma 1 *Let $G = (V, E)$ be a finite simple graph having at least one node. The quantity defined in (5) is an upper bound of the clique number of G.*

Proof. Set $k = \omega(G)$. Clearly G must contain a k-clique Δ. Let U' be the set of nodes of Δ that are in U and let W' be the set of nodes of Δ that are in

W. Here U and W are the subsets of V defined in (1). Obviously, $U' \cap W' = \emptyset$ and $|U'| + |W'| = k$. We distinguish four cases.

Case 1	$U' = \emptyset$	$W' = \emptyset$
Case 2	$U' = \emptyset$	$W' \neq \emptyset$
Case 3	$U' \neq \emptyset$	$W' = \emptyset$
Case 4	$U' \neq \emptyset$	$W' \neq \emptyset$

Since G has at least one node it must have a 1-clique. Thus $k \geq 1$ holds and so case 1 is not possible.

If $U' = \emptyset$, then Δ is a clique in the subgraph K of G induced by W. The nodes of K are legally colored with q colors and so $k \leq q$. Note that $p = 0$ and the ordered pair $(0, q)$ is a qualifying pair. It follows that $q \leq t$. Thus $k \leq q$ as required. This settles case 2. Case 3 can be sorted out in a similar way.

In case 4 the set of nodes of Δ is equal to $U' \cup W'$. This means that the unordered pair $\{u, w\}$ is an edge of G for each $u \in U'$, $w \in W'$. Let $r = |U'|$ and $s = |W'|$. The subgraph L_u of G induced by $N(u) \cap W$ must contain an s-clique. There are r choices for the node $u \in U'$. These choices show that the inequalities (3) hold. Similarly, the subgraph L_w of G induced by $N(w) \cap U$ must contain an r-clique. There are s choices for the node $w \in W'$. These choices show that the inequalities (4) hold. Therefore the ordered pair (r, s) is a qualifying pair. The inequality $k \leq r + s$ holds for each qualifying pair (r, s). Thus $k \leq t$, as required. □

3 A small size example

In this section we work out a small example in details to illustrate the method of profiles.

Example 2 *Let us consider the finite simple graph* $G = (V, E)$ *given by its adjacency matrix in Table 1. A geometric representation of* G *is depicted in Figure 1. The graph has* 16 *vertices and* 39 *edges.*

Using the simplest greedy sequential coloring procedure we colored the nodes of G legally. The procedure is presented in Table 2. The first column contains the nodes of the graph G. The last column holds the colors of the nodes. A column between the first and the last represents a partial coloring of the nodes of G. The "←" symbol points to the pivot node. The node to which we are assigning color at this phase. The "]" symbol after a color indicates that the

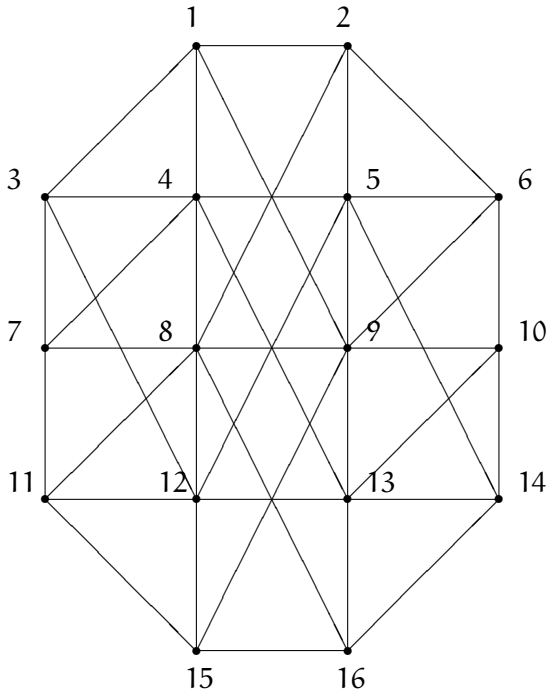

Figure 1: A geometric representation of the graph G in Example 2.

pivot node is adjacent to this node and the marked color cannot be assigned to the pivot node.

The color classes of the nodes are the following

$$C_1 = \{1, 5, 7, 10, 15\}, C_2 = \{2, 3, 9, 11, 14\}, C_3 = \{4, 6, 12, 16\}, C_4 = \{8, 13\}.$$

The coloring of the nodes gives that $\omega(G) \leq 4$. We try to reduce this upper estimate. We set

$$U = C_1 \cup C_2, \quad W = C_3 \cup C_4.$$

We computed the clique degrees of the nodes and the profiles of the graphs H, K. The results are summarized in the first three arrays of Table 3. An inspection of the qualifying pairs (r, s) reveals that $\omega(G) \leq 3$.

The inspection to decide if a given ordered pair (r, s) is qualifying or not is summarized in the last array of Table 3. We assume that there is a complete bipartite graph with independent sets A and B whose cardinalities are r and

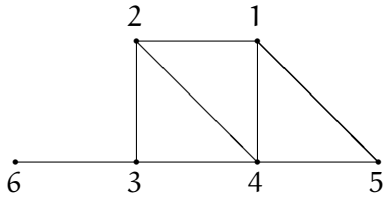

Figure 2: A graphical representation of the graph G in Examples 3 and 6.

s respectively and the graph of course has rs edges. Each of the r nodes of A needs to have a clique degree at least r and each of the s nodes of B needs to have clique degree at least r. These requirements are listed in a row labeled by the word "needed". The available clique degrees are listed in a row labeled by the word "found". Comparing these rows we can spot if the pair (r, s) is not qualifying. We used a "+" sign to indicate when the needed and the found clique degrees do not meet with the requirement.

We would like to emphasize that the method of profiles can produce an upper estimate for $\omega(G)$ which is below $\chi(G)$. (Such an estimate is termed as infra chromatic in the literature.) In order to exhibit such an example we note that $\chi(G) = 4$. Let us suppose on the contrary that $\chi(G) = 3$. Let us order the nodes of G as listed in the first column in the second array in Table 2. The nodes 1, 3, 4 are the nodes of a 3-clique in G. We may color these nodes by colors 1, 2, 3. After these choices the greedy coloring procedure will color the nodes up to node 10 uniquely. For node 13 we must use an additional color. The indirect assumption $\chi(G) = 3$ leads to a contradiction.

4 The first auxiliary graph

Let $G = (V, E)$ be a finite simple graph. Using G we construct a new graph $\Gamma_1 = (W, F)$. We call Γ_1 the first auxiliary graph associated with G. The nodes of Γ_1 are the ordered pairs (v, a), $v \in V$, $1 \leq a \leq 2$. If the unordered pair $\{v_1, v_2\}$ is an edge of G, then the four pair-wise distinct distinct nodes $(v_1, 1)$, $(v_1, 2)$, $(v_2, 1)$, $(v_2, 2)$ of Γ_1 are the nodes of a 4-clique in Γ_1. In other words if $\{v_1, v_2\} \in E$, then $\{w_1, w_2\} \in F$ for each distinct $w_1, w_2 \in \{(v_1, 1), (v_1, 2), (v_2, 1), (v_2, 2)\}$.

We illustrate the construction of the auxiliary graph in connection with a very small size toy example.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	×	●	●	●					●							
2	●	×			●	●		●								
3	●		×	●			●					●				
4	●		●	×	●		●	●					●			
5		●		●	×	●			●			●		●		
6		●			●	×			●	●						
7			●	●			×	●			●					
8		●		●			●	×	●		●	●				●
9	●				●	●		●	×	●			●		●	
10						●			●	×			●	●		
11							●	●			×	●			●	
12			●		●			●			●	×	●		●	
13				●					●	●		●	×	●		●
14					●					●			●	×		●
15									●		●	●			×	●
16								●					●	●	●	×

Table 1: The adjacency matrix of the graph G in Example 2.

Example 3 *Let us consider the finite simple graph* $G = (V, E)$ *given by its adjacency matrix in Table 4. A geometric representation of* G *is depicted in Figure 2. The graph has* 6 *vertices and* 8 *edges.*

A geometric representation of the auxiliary graph Γ_1 can be seen in Figure 3. The adjacency matrix of Γ_1 is given in Table 5. In fact two versions of the adjacency matrix are given. The nodes of Γ_1 are listed in different ways.

The clique numbers of the graphs G and Γ_1 are related. This is the content of the next lemma.

Lemma 4 *Let* G *be a finite simple graph and let* Γ_1 *be the associated auxiliary graph. Then* $2\omega(G) \leq \omega(\Gamma_1)$.

Proof. Set $k = \omega(G)$. The graph G contains a k-clique Δ. Let U be the set of nodes of Δ. Let $T = \{(u, a) : u \in U,\ 1 \leq a \leq 2\}$. Clearly $|T| = 2|U| = 2k$. Note that two distinct nodes (u_1, a_1), (u_2, a_2) in T are always adjacent nodes in Γ_1. □

Lemma 4 tells us that if t is an upper bound for $\omega(\Gamma_1)$, then $t/2$ is an upper bound for $\omega(G)$.

1	1]	1]	1]	1	1	1	1	1]	1	1	1	1	1	1	1	1
2	←	2	2	2]	2]	2	2]	2	2	2	2	2	2	2	2	2
3		←	2]	2	2	2]	2	2	2	2	2]	2	2	2	2	2
4			←	3]	3	3]	3]	3	3	3	3	3]	3	3	3	3
5				←	1]	1	1	1]	1	1	1]	1	1]	1	1	1
6					←	3	3	3]	3]	3	3	3	3	3	3	3
7						←	1]	1	1	1]	1	1	1	1	1	1
8							←	4]	4	4]	4]	4	4	4	4]	4
9								←	2]	2	2	2]	2	2]	2	2
10									←	1	1	1]	1]	1	1	1
11										←	2]	2	2	2]	2	2
12											←	3]	3	3]	3	3
13												←	4]	4	4]	4
14													←	2	2]	2
15														←	1]	1
16															←	3

1	1	1	1	1	1	1]	1]	1	1	1	1	1
3	2]	2	2	2]	2	2	2	2	2	2	2	2
4	3]	3]	3	3	3	3	3	3]	3	3	3]	3
7	←	1]	1]	1	1	1	1	1	1	1	1	1
8		←	2]	2]	2	2]	2]	2	2	2	2	2
11			←	3]	3]	3	3	3	3	3	3	3
12				←	1]	1	1	1]	1	1	1]	1
15					←	2	2]	2	2	2	2	2
2						←	3	3]	3]	3	3	3
9							←	3]	3]	3]	3]	3
5								←	2]	2	2	2
6									←	1]	1	1
10										←	2]	2
13											←	4
14												
16												

Table 2: The greedy coloring of the nodes in Example 2.

	C_1					C_2				
node	1	5	7	10	15	2	3	9	11	14
clique degree	1	1	1	1	1	1	1	1	2	2
maximum	1					2				

	C_3				C_4	
node	4	6	12	16	8	13
clique degree	2	2	2	1	2	2
maximum	2				2	

profile of H	2	1
profile of K	2	2

$r = 2, s = 2$				
needed	2	2	2	2
found	2	1	2	2
		+		
$r = 1, s = 2$				
needed	2		1	1
found	2		2	2

Table 3: The nodes with clique degrees and the profiles of H and K in Example 2.

	1	2	3	4	5	6
1	×	●		●	●	
2	●	×	●	●		
3		●	×	●		●
4	●	●	●	×	●	
5	●			●	×	
6			●			×

Table 4: The adjacency matrix of the graph G in Examples 3 and 6.

	1 1	2 1	3 1	4 1	5 1	6 1	1 2	2 2	3 2	4 2	5 2	6 2
1,1	×	•		•	•		•	•		•	•	
2,1	•	×	•	•			•	•	•	•		
3,1		•	×	•		•		•	•	•		•
4,1	•	•	•	×	•		•	•	•	•	•	
5,1	•			•	×		•			•	•	
6,1			•			×			•			•
1,2	•	•		•	•		×	•		•	•	
2,2	•	•	•	•			•	×	•	•		
3,2		•	•	•		•		•	×	•		•
4,2	•	•	•	•	•		•	•	•	×	•	
5,2	•			•	•		•			•	×	
6,2			•			•			•			×

	1 1	1 2	2 1	2 2	3 1	3 2	4 1	4 2	5 1	5 2	6 1	6 2
1,1	×	•	•	•			•	•	•	•		
1,2	•	×	•	•			•	•	•	•		
2,1	•	•	×	•	•	•	•	•				
2,2	•	•	•	×	•	•	•	•				
3,1			•	•	×	•	•	•			•	•
3,2			•	•	•	×	•	•			•	•
4,1	•	•	•	•	•	•	×	•	•	•		
4,2	•	•	•	•	•	•	•	×	•	•		
5,1	•	•					•	•	×	•		
5,2	•	•					•	•	•	×		
6,1					•	•					×	•
6,2					•	•					•	×

Table 5: The adjacency matrix of the auxiliary graph Γ_1 in Example 3.

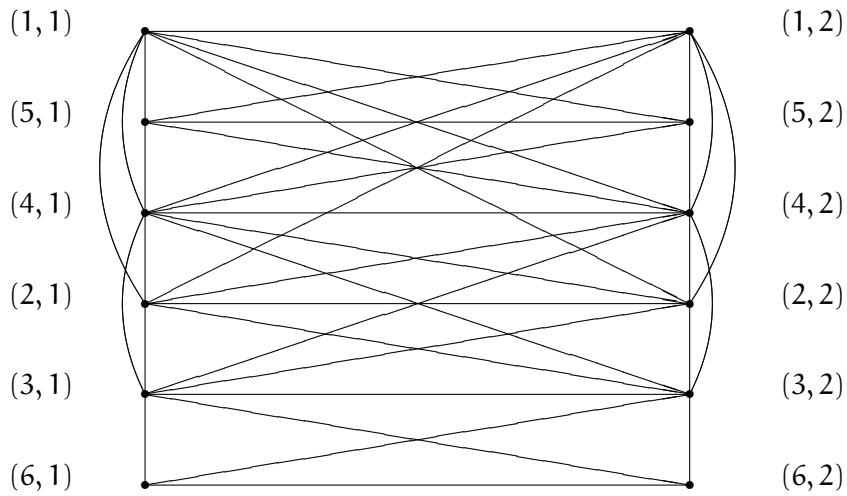

Figure 3: A graphical representation of the auxiliary graph Γ_1 in Example 3.

The chromatic numbers of the graphs G and Γ_1 are not independent of each other. This is the content of the next lemma.

Lemma 5 *Let G be a finite simple graph and let Γ_1 be the associated auxiliary graph. Then $\chi(\Gamma_1) \leq 2\chi(G)$.*

Proof. Set $k = \chi(G)$. The nodes of G have a legal coloring using k colors. The coloring of the nodes of G can be given by a function $f : V \to \{1, 2, \ldots, k\}$, where $f(v)$ is the color of node v of G. Let

$$D = \{(x, y) : 1 \leq x \leq k, 1 \leq y \leq 2\}.$$

Clearly D has $2k$ elements. Using f we construct a function $g : W \to D$ by setting $g((v, a)) = (f(v), a)$ for each $v \in V$, $a \in \{1, 2\}$. We would like to verify that g defines a legal coloring of the nodes of Γ_1.

Let $w_1 = (v_1, a_1)$ and $w_2 = (v_2, a_2)$ be two distinct nodes of Γ_1. Assume on the contrary that w_1, w_2 are adjacent nodes in Γ_1 and $g(w_1) = g(w_2)$. Note that $g(w_1) = g(w_2)$ implies $f(v_1) = f(v_2)$ and $a_1 = a_2$. As $a_1 = a_2$ it follows that v_1 and v_2 are adjacent nodes in G. In this situation $f(v_1) = f(v_2)$ cannot hold. This contradiction shows that g defines a legal coloring of the nodes of Γ_1. Thus $\chi(\Gamma_1) \leq 2k$, as required. □

Combining the results of Lemmas 4 and 5 gives that

$$2\omega(G) \leq \omega(\Gamma_1) \leq \chi(\Gamma_1) \leq 2\chi(G)$$

and so

$$\omega(G) \leq [\chi(\Gamma_1)]/2 \leq \chi(G).$$

Thus $[\chi(\Gamma_1)]/2$ gives a better estimate for the clique number of G than $\chi(G)$ does.

When G is a cycle of odd length, then $\chi(G) = 3$ and $\chi(\Gamma_1) = 5$. There are infinitely many cases with $\chi(\Gamma_1)/2 < \chi(G)$. In a typical application we do not compute chromatic numbers instead using a greedy coloring procedure we locate legal colorings for the nodes of G and Γ_1. The number of colors we find in this way are only upper estimates of the corresponding chromatic numbers. Lemma 5 says nothing about the relation of these upper bounds. On the other hand from the proof of Lemma 5 we can read off that if the nodes of G can be legally colored using k colors, then this coloring can be extended to a legal coloring of the nodes of Γ_1 using $2k$ colors.

When we use a computationally not demanding greedy coloring algorithm we may locate a legal coloring of the nodes of G and Γ_1. Then using the number of colors we establish two upper bounds for $\omega(G)$ and we can use the better one.

	12	14	15	23	24	34	36	45
1,2	×	•			•			
1,4	•	×	•		•			•
1,5		•	×					•
2,3				×	•	•		
2,4	•	•		•	×	•		
3,4				•	•	×		
3,6							×	
4,5		•	•					×

Table 6: The adjacency matrix of the auxiliary graph Γ_2 in Example 6.

5 The second auxiliary graph

Let $G = (V, E)$ be a finite simple graph. Using G we construct a new graph $\Gamma_2 = (W, F)$. We call this new graph the second auxiliary graph associated with G. The nodes of Γ_2 are the edges of G. Let $w_1 = \{u_1, v_1\}$, $w_2 = \{u_2, v_2\}$ be two distinct nodes of Γ_2. Set $U = \{u_1, v_1, u_2, v_2\}$. Note that as w_1 and w_2 are distinct edges in G, the cardinality of U is either 3 or 4. If the subgraph induced by the set U in G is a clique in G, then we connect the nodes w_1 and w_2 by an edge in Γ_2.

We work out the details of the construction of the auxiliary graph in connection with a small size graph.

Example 6 *Let us consider the finite simple graph* $G = (V, E)$ *given in Example 3.*

A geometric representation of the auxiliary graph Γ_2 is depicted in Figure 4 and Table 6 contains the adjacency matrix of Γ_2.

The clique numbers of the graphs G and Γ_2 are related. This is the content of the next lemma.

Lemma 7 *Let* G *be a finite simple graph and let* Γ_2 *be the associated auxiliary graph. Then* $[\omega(G)][\omega(G) - 1] \leq 2\omega(\Gamma_2)$.

Proof. Set $k = \omega(G)$. The graph G contains a k-clique Δ. Let U be the set of nodes of Δ. Let $T = \{\{u, v\} : u, v \in U, u \neq v\}$. Clearly $|T| = |U|(|U| - 1)/2 = k(k-1)/2$. Note that any two distinct nodes $\{u_1, v_1\}$, $\{u_2, v_2\}$ in T are always adjacent in Γ_2. □

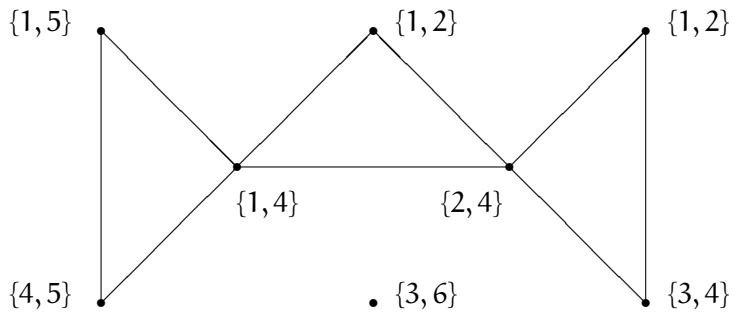

Figure 4: A graphical representation of the auxiliary graph Γ_2 in Example 6.

Using the method of profiles described in Section 2 we may establish that t is an upper bound of $\omega(\Gamma_2)$. By Lemma 7, $[\omega(G)][\omega(G)-1] \leq 2t$. Therefore if t' is the largest integer for which $t'(t'-1) \leq 2t$, then t' is an upper bound for $\omega(G)$.

The chromatic numbers of the graphs G and Γ_2 are not independent of each other. This is the content of the next lemma.

Lemma 8 *Let G be a finite simple graph and let Γ_2 be the associated auxiliary graph. Then $2\chi(\Gamma_2) \leq [\chi(G)][\chi(G)-1]$.*

Proof. Set $k = \chi(G)$. The nodes of G can be colored legally using k colors. The coloring can be given by a function $f : V \to \{1, 2, \ldots, k\}$, where $f(v)$ is the color of the node v of G. Set

$$D = \{\{x, y\} : 1 \leq x, y \leq k, x \neq y\}.$$

Obviously the cardinality of D is equal to $k(k-1)/2$. Using f we construct a function $g : W \to D$ defined by $g(\{u, v\}) = \{f(u), f(v)\}$.

We would like to show that g defines a legal coloring of the nodes of Γ_2. Let $w_1 = \{u_1, v_1\}$, $w_2 = \{u_2, v_2\}$ be two distinct adjacent nodes of Γ_2. Let $U = \{u_1, v_1, u_2, v_2\}$. Assume on the contrary that $g(w_1) = g(w_2)$.

Let us consider the case when the cardinality of the set U is four. In this case the nodes u_1, v_1, u_2, v_2 are pairwise distinct and they are nodes of a 4-clique in G. As $\{u_1, v_1\}$ is an edge of G and f is a legal coloring of the nodes of G it follows that $f(u_1) \neq f(v_1)$. Similarly $f(u_2) \neq f(v_2)$ must hold. From the

assumption

$$g(w_1) = \{f(u_1), f(v_1)\} = \{f(u_2), f(v_2)\} = g(w_2)$$

we get that either

$$f(u_1) = f(u_2), \quad f(v_1) = f(v_2)$$

or

$$f(u_1) = f(v_2), \quad f(v_1) = f(u_2).$$

The unordered pairs

$$\{u_1, u_2\}, \{u_1, v_2\}, \{v_1, u_2\}, \{v_1, v_2\}$$

are edges of G. This violates the fact that f is a legal coloring.

Let us turn to the case when U has three elements. In this case we may assume that $u_1 = u_2$ and $v_1 \neq v_2$ since this is only a matter of renaming the nodes. In this situation $\{v_1, v_2\}$ is an edge of G. The $g(w_1) = g(w_2)$ assumption reduces to $\{f(v_1)\} = \{f(v_2)\}$ and we get the contradiction that the end nodes of the edge $\{v_1, v_2\}$ are not legally colored. □

Let t be the largest integer for which $t(t-1) \leq 2\chi(\Gamma_2)$. Combining the results of Lemmas 7 and 8 we get

$$[\omega(G)][\omega(G)-1] \leq 2\omega(\Gamma_2) \leq 2\chi(\Gamma_2) \leq [\chi(G)][\chi(G)-1]$$

and so $\omega(G) \leq t \leq \chi(G)$. This means that using $\chi(\Gamma_2)$ one gets a better estimate for $\omega(G)$ than using $\chi(G)$.

J. Mycielski [10] proved the following result. For each positive integer n there is a graph M_n such that $\omega(M_n) = 2$ and $\chi(M_n) = n$. Let G be M_n. In this case the auxiliary graph Γ_2 consists of isolated nodes. Thus $\chi(\Gamma_2) = 1$. Now $\chi(\Gamma_2)$ and the inequality $[\omega(G)][\omega(G)-1] \leq 2\chi(\Gamma_2)$ provide $\omega(G) \leq 2$. In other words using $\chi(\Gamma_2)$ we get the upper bound 2 for $\omega(G)$ while using $\chi(G)$ we get n as an upper bound for $\omega(G)$.

6 Numerical experiments

In order to test the practical utility and feasibility of the method of profiles we have carried out numerical experiments. In this section we describe the results of these experiments.

The graphs we used are belonging to three families. However all test graphs are coming from coding theory. Monotonic matrices are related to certain one

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\omega}_1$	$\widehat{\omega}_1$	$\overline{\omega}_2$	$\widehat{\omega}_2$	$\overline{\omega}_4$	$\widehat{\omega}_4$
3	27	189	6	6	6	5	6	5
4	64	1 296	12	10	12	10	12	10
5	125	5 500	20	18	20	17	20	18
6	216	17 550	30	27	30	26	30	27
7	343	46 305	42	37	42	38	42	39
8	512	106 624	56	50	56	51	56	52
9	729	221 616	72	66	72	67	72	68
10	1 000	425 250	90	83	90	84	90	85
11	1 331	765 325	110	103	110	103	110	105
12	1 728	1 306 800	132	124	132	124	132	126
13	2 197	2 135 484	156	145	156	147	156	150

Table 7: Monotonic matrices, simple greedy coloring, first auxiliary graph.

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\omega}_1$	$\widehat{\omega}_1$	$\overline{\omega}_2$	$\widehat{\omega}_2$	$\overline{\omega}_4$	$\widehat{\omega}_4$
3	8	9	2	2	2	2	2	2
4	16	57	4	4	4	4	4	4
5	32	305	8	8	7	7	6	6
6	64	1 473	14	14	13	12	12	11
7	128	6 657	26	26	23	22	22	20
8	256	28 801	50	50	45	44	40	39
9	512	121 089	101	98	88	86	79	75
10	1 024	499 713	199	194	170	165	146	143
11	2 048	2 037 761	395	386	329	325	278	274

Table 8: Deletion error detecting codes, simple greedy coloring, first auxiliary graph.

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\omega}_1$	$\widehat{\omega}_1$	$\overline{\omega}_2$	$\widehat{\omega}_2$	$\overline{\omega}_4$	$\widehat{\omega}_4$
6	15	45	4	4	4	3	4	3
7	35	385	10	9	10	8	10	8
8	70	1 855	20	19	20	17	20	18
9	126	6 615	35	33	35	31	35	32
10	210	19 425	56	53	56	52	56	52
11	330	49 665	84	81	84	78	84	79
12	495	114 345	120	116	120	114	120	114
13	715	242 385	165	162	165	159	165	158
14	1 001	480 480	220	216	220	214	220	212
15	1 365	900 900	286	282	286	277	286	278
16	1 820	1 611 610	364	358	364	355	364	354

Table 9: Johnson codes, simple greedy coloring, first auxiliary graph.

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\omega}_1$	$\widehat{\omega}_1$	$\overline{\omega}_2$	$\widehat{\omega}_2$	$\overline{\omega}_4$	$\widehat{\omega}_4$
3	27	189	6	6	5	5	5	5
4	64	1 296	11	11	10	9	10	8
5	125	5 500	17	17	16	15	16	14
6	216	17 550	26	26	22	21	24	21
7	343	46 305	36	34	34	32	32	30
8	512	106 624	47	46	43	41	43	41
9	729	221 616	58	58	56	55	53	51
10	1 000	425 250	74	74	69	67	67	65
11	1 331	765 325	90	90	85	84	86	84

Table 10: Monotonic matrices, dsatur coloring, first auxiliary graph.

n	$\|V\|$	$\|E\|$	$\overline{\omega}_1$	$\widehat{\omega}_1$	$\overline{\omega}_2$	$\widehat{\omega}_2$	$\overline{\omega}_4$	$\widehat{\omega}_4$
3	8	9	2	2	2	2	2	2
4	16	57	4	4	4	4	4	4
5	32	305	7	7	6	6	6	6
6	64	1 473	13	13	12	12	11	11
7	128	6 657	23	23	22	21	22	20
8	256	28 801	43	43	40	40	43	40
9	512	121 089	79	79	80	77	79	77
10	1 024	499 713	156	156	154	154	153	151

Table 11: Deletion error detecting codes, dsatur coloring, first auxiliary graph.

n	$\|V\|$	$\|E\|$	$\overline{\omega}_1$	$\widehat{\omega}_1$	$\overline{\omega}_2$	$\widehat{\omega}_2$	$\overline{\omega}_4$	$\widehat{\omega}_4$
6	15	45	4	4	4	3	3	3
7	35	385	9	9	8	8	9	8
8	70	1 855	17	17	16	15	15	14
9	126	6 615	29	27	28	26	28	28
10	210	19 425	46	46	44	43	43	42
11	330	49 665	67	67	63	63	62	62
12	495	114 345	99	99	90	89	87	85
13	715	242 385	132	132	121	120	122	121
14	1 001	480 480	172	172	153	153	160	160
15	1 365	900 900	221	221	201	201	206	205

Table 12: Johnson codes, dsatur coloring, first auxiliary graph.

n	$\|V\|$	$\|E\|$	$\overline{\chi}$	$\overline{\omega}$	$\widehat{\chi}$	$\widehat{\omega}$
3	27	189	10	5	10	5
4	64	1 296	37	9	32	8
5	125	5 500	113	15	103	14
6	216	17 550	273	23	257	23
7	343	46 305	565	34	542	33

Table 13: Monotonic matrices, simple greedy coloring, second auxiliary graph.

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\chi}$	$\overline{\omega}$	$\widehat{\chi}$	$\widehat{\omega}$
3	8	9	1	2	1	2
4	16	57	6	4	6	4
5	32	305	17	6	16	6
6	64	1 473	60	11	53	10
7	128	6 657	221	21	207	20
8	256	28 801	875	42	846	41

Table 14: Deletion error detecting codes, simple greedy coloring, second auxiliary graph.

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\chi}$	$\overline{\omega}$	$\widehat{\chi}$	$\widehat{\omega}$
6	15	45	3	3	3	3
7	35	385	23	7	23	7
8	70	1 855	107	15	98	14
9	126	6 615	391	28	372	27
10	210	19 425	1 131	48	1 098	48
11	330	49 665	2 754	74	2 703	73

Table 15: Johnson codes, simple greedy coloring, second auxiliary graph.

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\chi}$	$\overline{\omega}$	$\widehat{\chi}$	$\widehat{\omega}$
3	27	189	10	5	10	5
4	64	1 296	31	8	31	8
5	125	5 500	83	13	75	12

Table 16: Monotonic matrices, dsatur coloring, second auxiliary graph.

n	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\chi}$	$\overline{\omega}$	$\widehat{\chi}$	$\widehat{\omega}$
3	8	9	1	2	1	2
4	16	57	6	4	6	4
5	32	305	15	6	15	6
6	64	1 473	50	9	50	9
7	128	6 657	196	20	183	19

Table 17: Deletion error detecting codes, dsatur coloring, second auxiliary graph.

$\mathfrak{n}$	$\lvert V\rvert$	$\lvert E\rvert$	$\overline{\chi}$	$\overline{\omega}$	$\widehat{\chi}$	$\widehat{\omega}$
6	15	45	3	3	3	3
7	35	385	23	7	22	7
8	70	1 855	111	15	101	14
9	126	6 615	340	25	323	24

Table 18: Johnson codes, dsatur coloring, second auxiliary graph.

error correcting codes. (The reader can find further details in [14].) A deletion error occurs when a fixed length code word is loosing one letter during transmission. The deletion error correcting codes are connected to this phenomenon. Binary codes with fixed length code words with a specified number of zeros are the Johnson codes. The graphs associated with these codes are commonly used for testing clique search algorithm. (See for instance [6].)

The method of profiles is flexible in the sense that we are free to choose any node coloring algorithm to construct a legal coloring of the nodes of the original graph or the auxiliary graphs. In the numerical experiments we carried out only two greedy coloring algorithms were employed. One of them is the most commonly used simple greedy sequential coloring. The other one is the dsatur coloring algorithm described in [3].

In Table 7 the first column contains the parameter $\mathfrak{n}$ of the graph. This parameter is related to the size of the alphabet over which the code is defined. The columns labeled by $|V|$ and $|E|$ hold the numbers of the nodes and the edges of the graph, respectively. The column headed by $\overline{\omega}_1$ holds the clique size estimate we get coloring the nodes of the original graph using the simple greedy coloring algorithm. Here the number of the colors is the upper estimate of the clique size. The column headed by $\widehat{\omega}_1$ holds the clique size estimate we get coloring the nodes of the original graph using the simple greedy coloring algorithm. This time the estimate of the clique size is the result of the method of profiles.

The column labeled by $\overline{\omega}_2$ refers to the clique size estimate we get coloring the nodes of the first auxiliary graph using the simple greedy coloring algorithm. Here half of the number of the colors is the upper estimate of the clique size. The column labeled by $\widehat{\omega}_2$ refers to the clique size estimate we get coloring the nodes of the first auxiliary graph using the simple greedy coloring algorithm. This time the estimate of the clique size is the result of the method of profiles.

The column labeled by $\overline{\omega}_4$ gives the clique size estimate we get coloring

the nodes of the first auxiliary graph of the first auxiliary using the simple greedy coloring algorithm. Here quarter of the number of the colors is the upper estimate of the clique size. The column labeled by $\widehat{\omega}_4$ gives the clique size estimate we get coloring the nodes of the first auxiliary graph of the first auxiliary graph using the simple greedy coloring algorithm. This time the estimate of the clique size is the result of the method of profiles.

Note that the number of the nodes of the first auxiliary graph is the double of the number of the nodes of the original graph. We adopt the terminology that the original graph is a 1-fold version of itself, the first auxiliary graph is a 2-fold version of the original graph, and the first auxiliary graph of the first auxiliary graph is a 4-fold version of the original graph. Using this terminology we may say that the column labeled by $\overline{\omega}_\mathfrak{b}$ contains the clique size estimate based on the $\mathfrak{b}$-fold version of the original graph not using the proposed procedure. Further the column labeled by $\widehat{\omega}_\mathfrak{b}$ contain the clique size estimate based on the $\mathfrak{b}$-fold version of the original graph using the method of profiles.

Tables 8 and 9 exhibit analogous information as Table 7. In these tables the graphs associated with monotonic matrices are replaced by graphs associated with deletion error correcting and Johnson codes, respectively.

Tables 10, 11, 12 summarize similar results that are in Tables 7, 8, 9. The only difference is that at these occasions the simple greedy coloring algorithm is replaced by the dsatur coloring algorithm.

The first three columns in Table 13 are labeled by $\mathfrak{n}$, $|\mathsf{V}|$, $|\mathsf{E}|$ record the parameter, the number of the nodes, the number of the edges of the graph associated with a monotonic matrix. The columns labeled by $\overline{\chi}$ and $\overline{\omega}$ contain the number of colors produced by simple greedy coloring procedure applied to the second auxiliary graph and the clique size estimate derived from this number of colors, respectively. The last two columns labeled by $\widehat{\chi}$ and $\widehat{\omega}$ show the reduced number of colors the method of profiles gives and the derived clique size estimate, respectively.

Tables 14 and 15 present similar results as Table 13 the only thing which has changed is that the graph associated with monotonic matrices are replaced by graphs associated with deletion error correcting and Johnson codes.

Finally Tables 16, 17, 18 exhibit similar results as Tables 13, 14, 15 but this time the simple greedy coloring procedure is replaced by the dsatur coloring algorithm.

References

[1] E. Balas, J. Xue, Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring, *Algorithmica* **15** (1996), 397–412. ⇒138

[2] I. M. Bomze, M. Budinich, P. M. Pardalos, M. Pelillo, *The Maximum Clique Problem, Handbook of Combinatorial Optimization* Vol. 4, Kluwer Academic Publisher, 1999. ⇒137

[3] D. Brélaz, New methods to color the vertices of a graph, *Communications of the ACM* **22** (1979), 251–256. ⇒138, 155

[4] R. Carraghan, P. M. Pardalos, An exact algorithm for the maximum clique problem, *Operation Research Letters* **9** (1990), 375–382. ⇒137

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, New York, 2003. ⇒138

[6] J. Hasselberg, P. M. Pardalos, and G. Vairaktarakis, Test case generators and computational results for the maximum clique problem, *Journal of Global Optimization* **3** (1993), 463–482. ⇒137, 155

[7] J. Konc and D. Janežič, An improved branch and bound algorithm for the maximum clique problem, *MATCH Communications in Mathematical and Computer Chemistry* **58** (2007), 569–590. ⇒138

[8] D. Kumlander, *Some Practical Algorithms to Solve the Maximal Clique Problem*, PhD. Thesis, Tallin University of Technology, 2005. ⇒138

[9] F. T. Leighton, A graph coloring algorithm for large scheduling problems, *Journal of Research of National Bureau of Standards* **84** (1979), 489–506. ⇒138

[10] J. Mycielski, Sur le coloriage des graphes, *Colloq. Math.* **3** (1955), 161–162. ⇒150

[11] P. R. J. Östergård, A fast algorithm for the maximum clique problem, *Discrete Applied Mathematics* **120** (2002), 197–207. ⇒138

[12] P. Prosser, Exact algorithms for maximum clique: A computational study, *Algorithms* **5** (2012), 545–587. ⇒137

[13] P. San Segundo, C. Tapia, Relaxed approximate coloring in exact maximum clique search, *Computers and Operations Research.* **44** (2014), 185–192. ⇒138

[14] S. Szabó, Monotonic matrices and clique search in graphs, *Annales Univ. Sci. Budapest., Sect. Computatorica* **41** (2013), 307–322. ⇒155

[15] E. Tomita and T. Seki, An efficient branch-and-bound algorithm for finding a maximum clique, *Lecture Notes in Computer Science* **2631** (2003), 278–289. ⇒138

[16] D. R. Wood, An algorithm for finding a maximum clique in a graph, *Operations Research Letters* **21** (1997), 211–217. ⇒138

Received: February 23, 2018 • Revised: September 10, 2018

Acta Univ. Sapientiae, Informatica **10,** 2 (2018) 158–182

DOI: 10.2478/ausi-2018-0009

Connecting the Last.fm Dataset to LyricWiki and MusicBrainz. Lyrics-based experiments in genre classification

Zalán BODÓ
Babeş–Bolyai University, Faculty of Mathematics and Computer Science
Cluj-Napoca, Romania
email: zbodo@cs.ubbcluj.ro

Eszter SZILÁGYI
Cluj-Napoca, Romania
email: bordieszter@gmail.com

Abstract. Music information retrieval has lately become an important field of information retrieval, because by profound analysis of music pieces important information can be collected: genre labels, mood prediction, artist identification, just to name a few. The lack of large-scale music datasets containing audio features and metadata has lead to the construction and publication of the Million Song Dataset (MSD) and its satellite datasets. Nonetheless, mainly because of licensing limitations, no freely available lyrics datasets have been published for research.

In this paper we describe the construction of an English lyrics dataset based on the Last.fm Dataset, connected to LyricWiki's database and MusicBrainz's encyclopedia. To avoid copyright issues, only the URLs to the lyrics are stored in the database. In order to demonstrate the eligibility of the compiled dataset, in the second part of the paper we present genre classification experiments with lyrics-based features, including bag-of-n-grams, as well as higher-level features such as rhyme-based and statistical text features. We obtained results similar to the experimental outcomes presented in other works, showing that more sophisticated textual features can improve genre classification performance, and indicating the superiority of the binary weighting scheme compared to tf–idf.

Computing Classification System 1998: H.3.3, H.5.5, I.2.6, I.2.7
Mathematics Subject Classification 2010: 68T05, 68T50, 68U15
Key words and phrases: music information retrieval, lyrics dataset, music genre classification

1 Introduction

A central problem of music information retrieval (MIR) is the similarity search of music tracks. Since in the last two decades online music streaming services and music stores have become exceedingly popular, to facilitate the search for similar music, recommender systems became vitally important too. Automatic genre classification is considered an equally important problem, since classification arises in simply browsing music by genre information as well as in music recommendation.

The lack of large-scale music datasets containing audio features and metadata has lead to the construction and publication of the Million Song Dataset[1] (MSD) [3] and its satellite datasets. However, as pointed out in [37] or [17], no freely available large-scale lyrics dataset has yet been published for research, mainly due to copyright problems. Although the musiXmatch dataset[2] offers hundreds of thousands of music tracks with lyrics given as bag-of-words vectors [50], this representation narrows down its applicability.

In this paper we describe the compilation of an English lyrics dataset based on the Last.fm Dataset, connected to LyricWiki's database and MusicBrainz's encyclopedia. Because of the copyright issues mentioned earlier, the dataset does not explicitly contain song lyrics, but LyricWiki page URLs pointing to the lyrics. Beside the URL we also included the MusicBrainz ID of the track, if found, together with album and release year information. In order to demonstrate the eligibility of the compiled dataset we conducted genre classification experiments with lyrics-based features, including bag-of-n-grams as well as higher-level features such as rhyme-based and statistical text features. We obtained results similar to the experimental outcomes presented in other works, showing that sophisticated textual features can improve genre classification performance, and indicating the superiority of the binary weighting scheme compared to tf–idf (term frequency $\times$ inverse document frequency).

The remainder of this paper is structured as follows. In Section 2—without striving for completeness—we review the works related to our research: MIR datasets, lyrics collections and classification experiments performed using these sets. Section 3 describes the process underlying the construction of the dataset: the databases involved in the compilation procedure, the scheme of the dataset, as well as some statistics. In Section 4 we present genre classification experiments based on the lyrics of the music tracks, using bag-of-words, n-grams, rhyme-based and statistical text features. Section 5 presents the con-

[1] `http://labrosa.ee.columbia.edu/millionsong/`

[2] `http://labrosa.ee.columbia.edu/millionsong/musixmatch`

crete experimental settings and results, while Section 6 discusses the results and concludes the paper.

2 Related work

Although the need for large-scale music information databases is of increasing concern, only a few such resources are accessible for research or commercial applications. One of the largest collection made available for MIR is the Million Song Dataset [3] and its numerous complementary datasets. The work [20] surveys the state-of-the-art problems in music analysis, and thus, it is a thorough collection of related bibliographical references and datasets. Another recent and comprehensive work on MIR is [25], likewise containing a large bibliography and references to associated datasets.

The authors of [4] have demonstrated by EEG experiments that the lyrics and tunes of a song are processed independently in the brain, therefore one can deduce that using textual features from the lyrics may improve the performance of a genre classification system. Song lyrics evidently contain valuable information—even the absence of the lyrics is an important clue when guessing music genre. All the information we obtain from the lyrics as textual data are inherently present in the audio signal. However, extracting lyrics directly from the audio data is still a very difficult task [15]. Therefore, we rely on the different versions that can be found in specific databases or on the Internet, the results of independent voluntary transcription procedures undertaken by different persons, in most cases. Thus, it is not uncommon to find a few differences because of different spellings, marking of chorus or verses, annotation of background voices, abbreviations, censored words, etc. In [26] the alignment of song lyrics is accomplished by multiple sequence alignment in order to eliminate typographical errors. These and related problems, however, can be overcome by community maintenance [48]. Hence, using a community-maintained lyrics database such as LyricWiki might prove to be more accurate.

It is important to mention the seminal work of [59] on genre recognition based on audio features. The paper introduces the famous GTZAN dataset[3], which despite of its inaccuracies [55] it is widely accepted and used. Other prevalent collections are the ISMIR 2004[4] and the CAL500[5] datasets. We also mention here some of the recent works on audio feature based music genre

[3] http://marsyasweb.appspot.com/download/data_sets/

[4] http://ismir2004.ismir.net/genre_contest

[5] http://labrosa.ee.columbia.edu/millionsong/pages/additional-datasets

recognition using convolutional neural networks [30, 12, 14, 47, 8], deep neural networks [53], deep belief networks [21] and multiscale approaches [13]. For an excellent presentation of these and similar approaches we direct the reader to [24].

In [38] the problems of music genre classifications are studied and analyzed: ambiguities and subjectivity inherent to genre, album rather than individual recording labeling, relatively frequent emergence of new genres, etc. It is also emphasized the importance of assigning multiple genre labels to music tracks, as this would result in a more realistic evaluation of classification systems. Bag-of-words features are combined with rhyme, part-of-speech (POS) and statistical text features in [36] for genre classification. The experiments are performed on a collection of 397 randomly sampled songs distributed among ten genres. The source of the lyrics data was not revealed in the paper. In [31] genre classification in the MSD is performed based on various feature types: audio (timbre, loudness and tempo), textual (bag-of-words, emotional valence) and combined features. Learning is accomplished via regularized multiclass logistic regression. The work [17] presents genre and best vs. worst music classification and release date prediction experiments using n-gram features extended with other higher-level features, including POS tags, rhymes, echoisms, semantic fields, etc. The experiments are carried out on a dataset built by the authors specifically for the targeted classification tasks, in which lyrics, genre information, album ratings and release dates were obtained from different online databases. The F_1 scores obtained in our experiments are very similar to their results.

The differences between poetry, song lyrics and other articles are studied in [54] using the adjectives extracted from the text. The presented method is also able to differentiate between poetic lyricists and non-poetic ones. The source for the lyrics data is not specified in the article. The authors of [10] perform lyrics-based mood prediction in the MSD using various term weighting schemes and find no statistically significant differences in the accuracy results. In [9] music subject classification based on lyrics and user interpretations are compared. The data was obtained from songmeanings.com and songfacts.com. Mood classification is studied in [33] using the lyrics of music tracks. The authors also study the relation between features and emotions to identify the most discriminative features for each quadrants. The lyrics data used in the experiments was collected using lyrics.com, ChartLyrics and MaxiLyrics, the tracks being annotated manually. The work [56] discusses evaluation approaches in music genre recognition, but also contains a useful list of existing datasets.

As the number of recent publications show, the lyrics collection provided by LyricFind[6]—through a signed research agreement—is becoming more and more popular. This is usually used together with the iTunes Search API[7] to obtain genre and other meta-information about the songs. It also has a bag-of-words version similar to musiXmatch, containing the bag-of-words representation of 275 905 lyrics.[8] In [16] lexical novelty of song lyrics is studied, and the authors find the already suspected fact that top-100 music is less lexically innovative than less popular music. A lyrics-based network is built to analyze musical relationships over time in [2]. It is observed that self-reference correlates highly with influence, the most central genres being jazz, pop and rock. LyricFind's collection is used in [58] as well, and a hierarchical attention network is applied to classify genre based on song lyrics in two scenarios, using 117 and 20 classes, respectively. The learning model allows to inspect the importance of words, lines and segments in lyrics.

The recent work [60] presents the construction of the ALF-200k dataset including 176 audio and lyrics features of more than 200 000 music tracks, together with their occurrence statistics in user playlists. Using the different sets of features the authors perform playlist membership prediction by adding random tracks originally not belonging to the playlist. Connecting with other databases it would be intriguing to perform genre classification experiments using these features too.

As related work shows, since no standard lyrics dataset can be found to work with, almost every study uses its own data, comparison between different methods being utterly complicated. This was the main reason behind building the collection connecting the Last.fm Dataset to LyricWiki and MusicBrainz. LyricWiki was chosen over other similar databases because of the advantages of community maintenance. To avoid copyright issues, instead of the actual verses only the LyricWiki URLs of the lyrics were included. We also publish unigram, bigram and trigram versions of this dataset, i.e. containing the n-gram representation of the lyrics. In order to validate the usage of the compiled dataset, genre classification experiments are presented in the second part of the present paper.

[6] http://lyricfind.com/

[7] http://apple.co/1qHOryr

[8] https://www.smcnus.org/lyrics/

3 Construction of the lyrics dataset

3.1 The Million Song and the Last.fm Dataset

The Million Song Dataset is a free collection of audio features and metadata for one million contemporary music tracks. It was released for research purposes in 2011 by the Laboratory for the Recognition and Organization of Speech and Audio (LabROSA) department of the Columbia University[9] in collaboration with The Echo Nest[10]. MSD is more than a single dataset, it is also a cluster of many spin-off datasets[11]: SecondHandSongs (cover songs), musiXmatch (lyrics), Last.fm (song-level tags and similarity), Taste Profile (user data), thisismyjam-to-MSD mapping (user data), tagtraum genre annotations (genre labels), Top MAGD dataset (genre labels).

The Last.fm Dataset[12] is a complementary set of MSD, containing song tags and similarity information, built in collaboration with Last.fm[13], an online music database and music recommendation system. Last.fm also provides an API for metadata retrieval[14], also used by us to connect and extend the Last.fm Dataset.

Last.fm data (i.e. tags, musical samples, etc.) has been used in numerous experiments. An interesting work we mention is [35], in which the authors analyze the evolution of popular music and musical revolutions identifiable in the collected data, using data mining techniques such as latent Dirichlet allocation and novelty detection.

3.2 LyricWiki

LyricWiki[15] is a community-maintained lyrics database, offering music metadata services, released in 2006.

In March 2013 it was the seventh largest MediaWiki installation[16], and as of August 2018 contains over two million pages. LyricWiki also provided a web API for searching songs and lyrics, however, due to licensing restrictions, in

[9] http://labrosa.ee.columbia.edu/
[10] http://the.echonest.com/
[11] http://labrosa.ee.columbia.edu/millionsong/pages/additional-datasets
[12] http://labrosa.ee.columbia.edu/millionsong/lastfm
[13] http://www.last.fm
[14] http://www.last.fm/api
[15] http://lyrics.wikia.com/wiki/LyricWik
[16] https://en.wikipedia.org/wiki/LyricWiki

2016 the API has been discontinued.[17] Interestingly, as of December 2018, the API[18] is again functional.

3.3 MusicBrainz

MusicBrainz[19] is an open online music encyclopedia of music metadata launched in 2000 [57]. As of 2018, the database, more precisely its *recording* index, contains over 19 million entries, being one of the largest such databases. MusicBrainz provides a web service[20] for metadata retrieval too.

The web API was used by us to obtain additional release information about a song.

3.4 Building the dataset

The Last.fm Dataset contains 839 122 training and 104 212 test records.[21] We succeeded in using 224 762 (199 217 training and 25 545 test) data, i.e. we managed to find the lyrics of that many songs in LyricWiki's database.

The dataset consists of tracks, where every track is identified by a unique Echo Nest ID. Beside the ID, every track has the following fields: *artist*, *title*, *timestamp*, *similars*, *tags*. The timestamp stores the date of creation. Similar tracks are enumerated as a list of tuples, containing the ID of the proximal track along with a similarity, a scalar value between 0 and 1. Similarly, the assigned tags are given as a list of tuples, consisting of a tag name, e.g. "rock", and a relevance value, an integer between 0 and 100.[22]

The *timestamp* and *similars* fields were removed from the tracks, however, if needed, one can easily retrieve this information by connecting our dataset to the Last.fm Dataset using the track ID. Only tags having a relevance value greater than or equal to 50 have been kept. Such a step is motivated by the fact that tracks have been tagged quite freely by the Last.fm users, therefore, one can also find some strange ones, as shown in Table 1. Thus, we considered a tag relevant only if at least 50% of the time it was assigned to the track.

[17] In 2015, using the API, we managed to connect the Last.fm Dataset to LyricWiki pages, using the artist's name and the title of the song.

[18] `http://lyrics.wikia.com/api.php`

[19] `http://musicbrainz.org/`

[20] `http://musicbrainz.org/doc/Development/XML_Web_Service/Version_2`

[21] The dataset had been downloaded on 12.05.2016 and had contained a total of 943 334 files, 13 tracks less than the value published on the official site of the dataset.

[22] A value of r means that in r% of the cases the respective tag was assigned to the track by the users.

what I want to hear at my funeral	super happy feel good
vagany	one of the best solos
amaaazzzinnnggg	songs to fall asleep to in a good way
banging the head on the wall	betterfriend
soooooo beautiful I died again	holy riffs

Table 1: Some random tags from the Last.fm Dataset with frequency 1.

Artist queried:	Queen & David Bowie
Song queried:	Under Pressure (Rah Mix) (Radio Edit) (1999 Digital Remaster)
Artist returned:	Queen
Song returned:	Under Pressure

Table 2: Answer returned by LyricWiki for track `TRTTPMY128F4258EAC`.

We encountered only one case where the relevance value did not exceed this threshold and it happened for track `TRQXIYJ128F930A292` from the training set—we left this record in the dataset with its maximum-valued tag.[23]

3.4.1 The lyrics

The *url* field contains the LyricWiki link of the lyrics. This LyricWiki page URL was obtained by using the LyricWiki API. LyricWiki returns the artist name, the song title, a short snippet of the lyrics and the link to the page containing the full lyrics and song information. There were three cases when the respective track was omitted from the dataset: not found, instrumental or not English. The language of the lyrics and the release information can be deduced from the page of the full lyrics.

Because lyrics of musical tracks are proprietary work, in most of the cases its publication are forbidden, therefore, we only offer the LyricWiki page where the lyrics can be extracted from, and n-gram datasets (up to trigrams) from which the lyrics cannot be reconstructed.

The artist name and song title returned by LyricWiki API can differ from the queried data, probably because of the preprocessing steps built into the search engine. Though we have not found any documentation regarding the

[23]The track in question is Bobby Brown's song, 'Pretty Little Girl', and is assigned only two tags, namely 'killer shredding' with a relevance score 2 and 'mod psych' along with a value of 0.

indexing/search operations, we have found evidence of text normalization.[24] An example is shown in Table 2. Because of these differences we decided to also store the returned data in the *artist_new* and *title_new* fields, respectively.

3.4.2 Release information and MusicBrainz IDs

We decided to extend the dataset by including release information (album and release year) for each song, and also to store the MusicBrainz IDs[25] (MBIDs) of the tracks.

The MSD comes with additional databases including a metadata SQLite database[26], containing metadata information such as song title, release, year, etc. Our *album* and *year* fields correspond to the *release* and *year* fields of this database. Sometimes release information occurred on LyricWiki pages, suggesting also that the respective track appeared on multiple releases. If found, using the first release mentioned on the page, this forms the content of the *album_new* field. In case this information was not to be found on the LyricWiki page, we made additional efforts to obtain it from the MusicBrainz encyclopedia. In order to connect MusicBrainz, we first performed a search with the Last.fm API using the artist name and song title from the Last.fm dataset, and then another search using the retrieved song and artist information by LyricWiki. In this way we obtained two MusicBrainz identifiers for each track, and stored it in the *mbid* and *mbid_new* fields, respectively. Knowing the MBID of a music track it is simple to query its releases, from which we stored the title of the first one in the *album_new* field.

For getting the release year of the track we acted similarly: if the year was found on the LyricWiki page, that one was stored in the *year_new* field, otherwise it was queried from MusicBrainz.

In some cases differences in artist names, song and album titles are due to slight spelling discrepancies. However, of course, incomplete information—on either side—can also cause it. The differences may also arise from multiple releases of the same song: original song/original album, live edition/concert album, remixed version of the song, compilation album, etc. Errors, mismatches can also appear in such databases. The used databases, however, were not

[24] For example, searching for 'Déjà Vu' by 'The Tear Garden', 'Deja Vu' is found and returned—of which, surprisingly, the returned form without diacritical marks is the correct song title (`https://www.discogs.com/Tear-Garden-Tired-Eyes-Slowly-Burning/master/7843`).

[25] `https://musicbrainz.org/doc/MusicBrainz_Identifier`

[26] `http://labrosa.ee.columbia.edu/millionsong/pages/getting-dataset`

```
"TRFDMMO128F424D545": {
 "mbid":       "ed3dccc3-e47b-4c81-90be-fdd7e820647a",
 "mbid_new":   "ed3dccc3-e47b-4c81-90be-fdd7e820647a",
 "title":      "6:00",
 "title_new":  "6:00",
 "artist":     "Dream Theater",
 "artist_new": "Dream Theater",
 "album":      "Awake",
 "album_new":  "Awake",
 "year":       "1994",
 "year_new":   "1994",
 "url":        "http://lyrics.wikia.com/Dream_Theater:6:00",
 "tags":       [["Progressive metal", "100"]]
}
```


Figure 1: Sample data from our dataset.

checked against a ground-truth dataset, therefore, no such information can be reported by us.

Summing it up, the Echo Nest ID being the key, the fields of a record in the dataset are the following:

- *mbid* – MusicBrainz ID returned by the Last.fm API for the artist and track name as given in the Last.fm Dataset (or MSD)
- *mbid_new* – MusicBrainz ID from the Last.fm API for the artist and track name as returned by LyricWiki
- *title* – title of the song according to MSD
- *title_new* – title of the song returned by LyricWiki
- *artist* – artist according to MSD
- *artist_new* – artist returned by LyricWiki
- *album* – album/release name according to MSD
- *album_new* – album/release name extracted from LyricWiki/using the Last.fm and MusicBrainz API
- *year* – release year according to MSD
- *year_new* – release year extracted from LyricWiki/using the Last.fm and MusicBrainz API
- *url* – LyricWiki URL of the song's lyrics
- *tags* – list of the tags assigned to the track filtered by the relevance value (≥ 50)

A sample data is shown in Figure 1.

	Training data	Test data
album	0	0
album_new	6858	1088
year	31 452	4323
year_new	34 363	4787
mbid	27 254	3446
mbid_new	14 070	2137

Table 3: Counts of missing fields in the dataset.

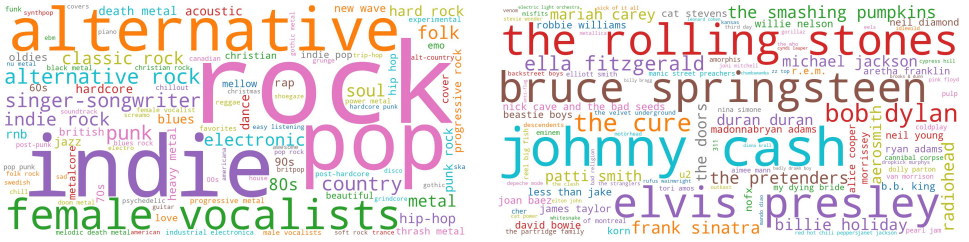

Figure 2: Word clouds of the (a) 100 most frequent tags and (b) 100 most popular artists.

3.5 Removal of duplicates and some statistics

As mentioned at the beginning of Section 3.4, linking the Last.fm dataset with LyricWiki pages we obtained 224 762 track records. According to millionsong's blog entry *"The 921,810 song dataset – duplicates"*[27], duplicate songs have been found in the MSD, which also affects the Last.fm Dataset. Using the official duplicate list of the MSD[28], we removed 27 529 records from the training and 3164 records from the test dataset. Therefore, our final dataset contains 171 688 and 22 381 (a total of 194 069) records.[29]

Table 3 shows the statistics of missing fields in the dataset.

In Figure 2 the most frequent tags and artists are shown, while the histograms of Figure 3 show the distribution of lyrics over years—as expected, a

[27] `http://labrosa.ee.columbia.edu/millionsong/blog/11-3-15-921810-song-dataset-duplicates`

[28] `http://labrosa.ee.columbia.edu/millionsong/sites/default/files/AdditionalFiles/msd_duplicates.txt`

[29] The dataset, under research-only, non-commercial license, is available for download at `http://www.cs.ubbcluj.ro/~zbodo/lastfm.html`.

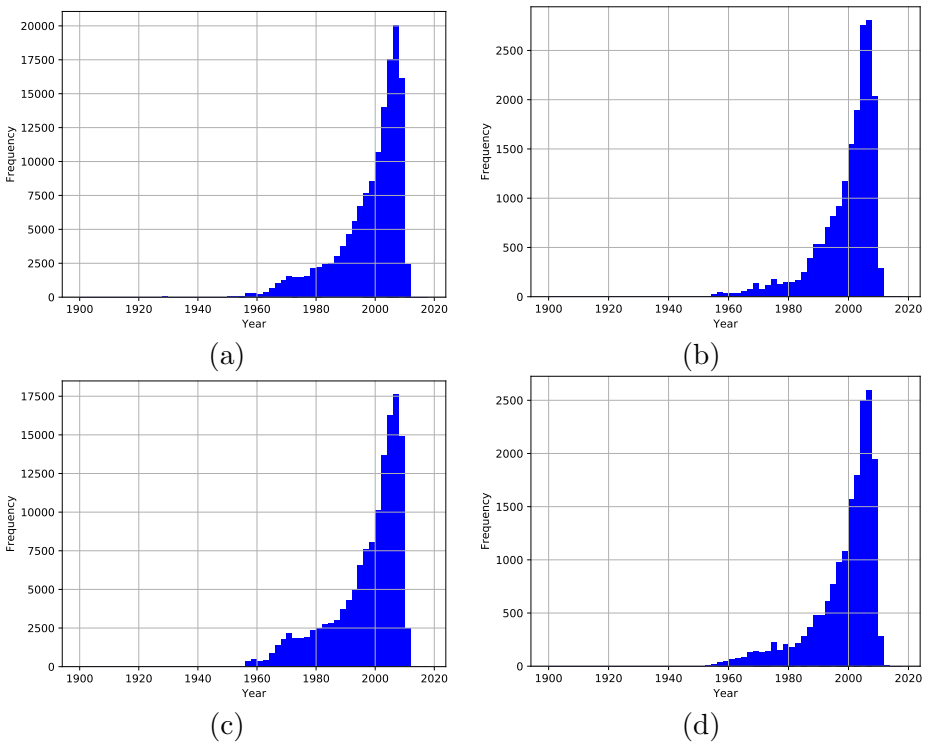

Figure 3: Distribution of lyrics over years in: (a) training data using *year*, (b) test data using *year*, (c) training data using *year_new*, (d) test data using *year_new*.

steep increase can be observed in the amount of available lyrics over time.

4 Lyrics-based genre classification

Determining the genre of a music track is considered to be an important task in MIR, which can be viewed as a special case of the music similarity problem. To assign genre labels to a song is a difficult task even for human annotators, because there are no clear and precise definitions of genres, thus often yielding subjective classifications. However, we mention that genre information is usually assigned to an artist or an album, rather than to a musical piece, which would be preferable. As pointed out in [38], musical genre classification should be based on numerous, complex features, including low-level, e.g. timbre-based features, but also high-level, e.g. cultural features. In [4] the au-

thors conducted experiments to demonstrate whether the lyrics and tunes of a song are processed independently in the brain. The analyzed electroencephalogram recordings showed that semantic and musical incongruities indeed do not affect each other. Hence, assuming that the lyrics can contain genre-related information, using lyrics-based textual features may have beneficial effects on classification.

For a general and also detailed discussion of music genre classification see [38].

In this section we describe a lyrics-based genre classification approach similar to [36, 31, 17]. The main goal of this experiment is to demonstrate the utility of the compiled dataset.

4.1 Choosing the genres

In order to use the dataset in some experiments the problem of musical genre classification, or more generally thematic categorization of lyrics was chosen [32, 38, 36, 17]. As it was already shown in Tables 1 and 2(a), the dataset contains a large variety of tags, ranging from genre related to other diverse descriptive labels. More precisely, the 194 069 records are assigned a number of 76 746 different tags, using the reduced tag lists. To perform a supervised learning task, we decided to choose only a subset of these, possibly denoting musical genres.

Determining a good taxonomy of musical genres is itself a difficult problem, and almost all online music stores and retailers use a different genre hierarchy. Thus, we were not able to find a suitable taxonomy consisting of a smaller number of meta genres, and decided to randomly select some of the most popular tags from our dataset (see Figure 2(a)). On how to derive better, objective genre taxonomies see the works [43] and [51].

Because of the almost limitless freedom given for the users in tagging, not all tags convey genre information about a song in the Last.fm Dataset. But, as described in Section 3.4, we used a threshold of 50 when deciding whether to keep a tag for a track. Thus, we expect that most of the time the remaining tags are valid, consisting of genre annotations and not deliberately misleading labels as described in [6].

The chosen tags—and the corresponding data counts—are the following:[30]

- *rap*: 2972 training, 451 test data (28th most popular tag)

[30] In order for the experiments to be reproducible, we mention that for each tag we required an exact match with case insensitivity.

- *reggae*: 1608 training, 178 test data (56th most popular tag)
- *jazz*: 3300 training, 339 test data (26th most popular tag)
- *punk*: 6760 training, 551 test data (8th most popular tag)
- *country*: 6360 training, 866 test data (9th most popular tag)
- *folk*: 6074 training, 831 test data (11th most popular tag)
- *pop*: 14267 training, 1855 test data (3rd most popular tag)
- *classic rock*: 6413 training, 482 test data (12th most popular tag)
- *electronic*: 5851 training, 851 test data (13th most popular tag)

The genres were chosen quite randomly, but some extra care was taken not to produce an extremely skewed distribution among the classes. This is the reason, for example, behind choosing *classic rock* instead of the *rock* label.

Thus, we are given a total of 60 009, i.e. 53 605 training and 6404 test data distributed unevenly among the 9 classes. The only non-overlapping class pairs are *reggae* and *country*, whilst the largest overlap of 643 (training and test) records happens between classes *pop* and *electronic*.

4.2 Choosing the features

4.2.1 N-grams

The bag-of-words model is a successful representation in information retrieval, which, despite its simplicity, yields surprisingly good results in categorizing text documents [50, 1]. The main drawback of the model is the assumption of independence between the words, but its effectiveness indicates that most of the time one can determine the category based on specific keywords, or more precisely by the distribution of these keywords. A somewhat better model that takes into account the word order is the n-gram representation [18]. In the experiments we successively extended the bag-of-words representation—i.e. the unigram model—with bigrams and trigrams.

4.2.2 Rhyme features

Can rhyme schemes be used to discriminate between musical genres? This is the question we wanted to answer by including rhyme features into the lyrics representation.

The Merriam-Webster dictionary gives us the definition of rhyme as the "correspondence in terminal sounds of units of composition or utterance (as two or

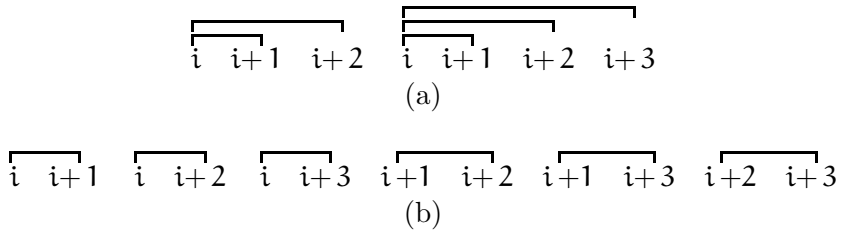

Figure 4: Rhyme features: (a) n-gram rhymes ($n_s = 3, n_e = 4$), (b) pairwise rhymes ($n_m = 4$).

more words or lines of verse)"[31], while the authors of [49] define rhyming as follows: "two words rhyme if their final stressed vowels and all following phonemes are identical". In our more primitive interpretation, two words rhyme if their last syllables are similar, and we will use this definition in building the rhyme features. The last two definitions, however, are incomplete: to find the rhyme scheme of a stanza one should also consider that a rhyme may span more words or syllables in line [22, 49]. Nonetheless, because of the relatively rare occurrences we decided not to consider these cases. Internal rhymes can also appear in song lyrics [22, 17], however, these do not influence the rhyme scheme. The authors of [17] used these kind of rhymes too to build lyrics-based features, calling them echoisms.

To find rhyming words in lyrics using our definition from above three components and a threshold are needed:

(a) hyphenator,
(b) phonetic algorithm,
(b) string similarity,

and a threshold for considering two syllables sufficiently similar. For determining the syllables the hyphenation method of OpenOffice and LibreOffice was used [42], representing the pronunciation of the last syllable was realized using the phonetic algorithm of Soundex [27], and for comparing these pronunciations we selected the Levenshtein distance [29, 19].

Two rhyme feature sets were used: an n-gram rhyme set, parametrized by n_s and n_e, and a pairwise rhyme set parametrized by n_m.[32] The n-gram rhyme set starts with rhyme schemes of length n_s and continues to build features

[31] Full definition of rhyme (2a), `http://www.merriam-webster.com/dictionary/rhyme`.

[32] The denomination *n-gram* is not the best here, since it does not fully reflect the nature of this feature, but hopefully the example given will clarify the vagueness.

	number of verses	number of rows	average no. of words in rows
rap	(9.5918,	72.7481,	8.7017)
classic rock	(7.1029,	33.0973,	6.6492)

Table 4: Example of statistical text feature vectors.

until length n_e, as shown in Figure 4(a). A feature is thus described by k binary values, $k = n_s - 1, n_s, \ldots, n_e - 1$, each value indicating the presence or absence of a rhyme. For example, from the alternating rhyme scheme ABAB—assuming it was correctly recognized—we obtain the following features with $n_s = 3$, $n_e = 4$: $(0, 1)$ (this will appear twice, because of ABA and BAB show the same pattern), $(0, 1, 0)$. The other rhyme feature set checks for pairwise rhymes between the i-th and $(i + k)$-th row of the lyrics independently, $k = 1, 2, \ldots, n_m - 1$, for all i. In contrast to the previous rhyme feature set, these features are encoded by pairs describing the distance between the row indices and the binary value showing whether or not a rhyme was found. Thus, using the same example as before, we get the following pairwise rhyme features using $n_m = 4$: $(1, 0)$ (three times, for indices $(0, 1)$, $(1, 2)$, $(2, 3)$), $(2, 1)$ (twice, for indices $(0, 2)$ and $(1, 3)$), and $(3, 0)$ (for the pair $(0, 3)$).

4.2.3 Statistical text features

Statistical text features—number of verses, average number of words in a row, etc.—are useful characteristics when classifying lyrics [36]. One could expect for example that the lyrics of a rap song is longer in average than the lyrics of a rock or pop song, which indeed turns out to be true. Comparing the averages of number of verses, number of rows and average number of words in rows between tracks of *rap* and *classic rock* we get the results shown in Table 4.

In our experiments we used the following 14 statistical text features: number of verses, number of rows, average number of words in rows, average word length, number of special characters (`!`, `.`, `?`, `:`, `;`, `-`, `,`, `'`, `"`), average frequency of numbers in the rows of the lyrics.

5 Experimental results

By performing the experiments we wanted to show the following: (i) n-gram features alone can yield good results, (ii) using rhyme and statistical text

Features	Micro F_1	Macro F_1
Unigrams, TF–IDF	48.47%	46.47%
Unigrams, frequency	50.02%	46.83%
Unigrams, binary	52.24%	50.30%
Unigrams (1000/class), binary	52.11%	49.44%
Uni + bigrams, TF–IDF	49.32%	48.03%
Uni + bigrams, frequency	50.26%	48.41%
Uni + bigrams, binary	54.50%	52.77%
Uni + bi + trigrams, TF–IDF	50.69%	48.56%
Uni + bi + trigrams, frequency	51.77%	49.42%
Uni + bi + trigrams, binary	**56.61**%	**54.53**%

Table 5: Micro and macro F_1 scores obtained using logistic regression.

Features	Micro F_1	Macro F_1
Rhyme + statistical text features	37.19%	22.44%
+ uni+bi+trigrams (binary)	**57.59**%	**54.99**%

Table 6: Micro and macro F_1 scores obtained using logistic regression with the new feature set alone and by augmenting the unigram, bigram and trigram feature set with it.

features can improve on the performance of the classifier. The goal was to approximately reproduce the experiments described in [36] and [17], and show the usefulness of the newly compiled dataset.

We applied a single-label classifier for learning [52], namely logistic regression [11, 5], using the *scikit-learn* Python library[33]. As mentioned in the previous section, we had slightly overlapping categories, which means that better results could have probably been achieved by using a ranking classifier and finding good thresholds for the categorization status values [52]. As for proving the above-mentioned two claims we needed no multilabel classifier. In our experiments, for every track the most frequent tag was used as its label. Thus, we are given 50 622 and 6113 test data.

To evaluate the models, micro- and macro-averaged F_1 scores were calculated [34].

Table 5 shows the results obtained using only n-gram features. We experimented with three weighting schemes: term frequency, tf–idf and binary weights [52, 34]. With unigrams we obtained 171 207 features, with unigrams

[33] `http://scikit-learn.org/`

	Genre	Uni+bi+trigram	Uni+bi+trigram+rhyme and stat. text features
1	rap	87.83%	88.36%
2	reggae	45.45%	42.67%
3	jazz	53.47%	54.07%
4	punk	48.33%	49.60%
5	country	65.26%	65.41%
6	folk	47.74%	47.61%
7	pop	61.79%	63.22%
8	classic rock	39.41%	40.63%
9	electronic	41.52%	43.34%
	Micro-averaged	56.61%	57.59%
	Macro-averaged	54.53%	54.99%

Table 7: F_1 scores for each genre for the unigram, bigram and trigram model and the same representation augmented with the rhyme and statistical text features.

and bigrams 3 845 117, while using unigrams, bigrams and trigrams together a total of 16 433 472 features were obtained.[34]

Table 6 shows the results obtained first by using the rhyme and statistical text features only, and in its second row the scores achieved by augmenting the n-gram document vectors with the rhyme and statistical text features. Table 7 lists the F_1 results for each genre separately.

The parameters of generating the features described in Section 4.2.2 were $n_s = 4$, $n_e = 5$, $n_m = 7$. Together with the statistical text features the new feature set has a cardinality of $36 + 14$. Finding the rhymes was performed using the Soundex algorithm.[35] The outputs of this algorithm, the phonetic representations of the input words—more precisely, of the last syllable of the input words—have to be used as inputs of a similarity or distance function with a predetermined threshold. For this we used normalized Levenshtein distance with threshold 0.7.[36] We mention that none of the parameters used in the experiments were selected using cross-validation or a similar procedure, therefore it is highly probable that by tuning the parameters better performance can be achieved.

[34]The unigrams training data has a sparsity of 5.68×10^{-4}%, the uni+bigrams 6.62×10^{-5}%, while using uni+bi+trigrams a sparsity of 2.59×10^{-5}% is observed.

[35]Fuzzy, `https://pypi.python.org/pypi/Fuzzy`.

[36]py_stringmatching, `https://pypi.org/project/py_stringmatching/`.

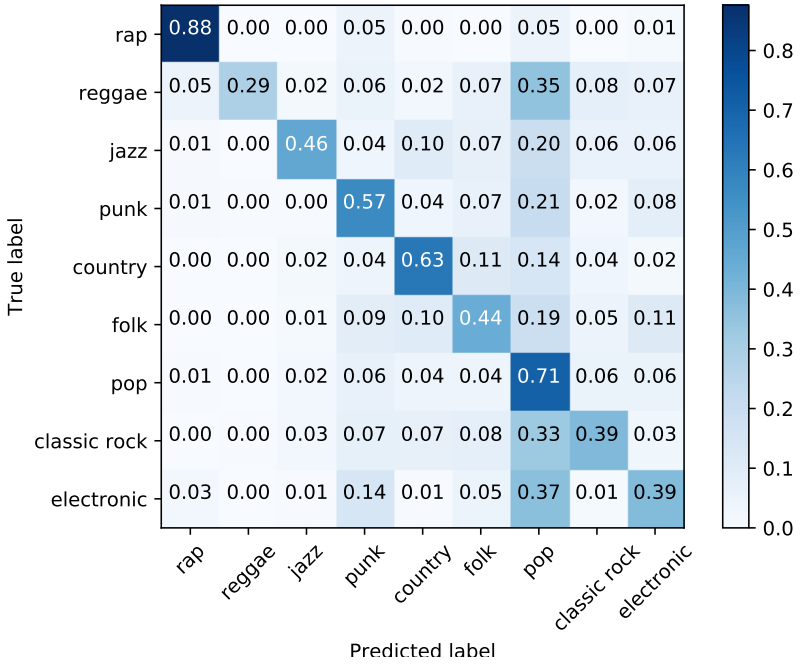

Figure 5: Confusion matrix for the uni+bi+trigram model augmented with rhyme and statistical text features. For the categories see Table 7.

6 Discussion and conclusions

We presented the compilation of a lyrics dataset linking the Last.fm Dataset, LyricWiki and MusicBrainz. The dataset contains the lyrics, i.e. LyricWiki URLs of English songs of the Last.fm Dataset (or MSD) found in LyricWiki's database, extended with additional release information. Knowing the MBID of a music track the dataset can be further extended with ease. After linking the Last.fm Dataset with LyricWiki and removing the duplicates—as described in Section 3.5—the final set contains 171 688 training and 22 381 test records. The tags of the music tracks were copied from the Last.fm Dataset without any text normalization, but the lists were thresholded at a relevance score ≥ 50. For the complete description of the database fields see Section 3.4.2.

In the second part of the paper we described the genre classification experiments conducted using the new dataset and considering some of the most frequent tags as genres. From previous research we already knew that using higher-level lyrics-based features improves on the performance of the genre

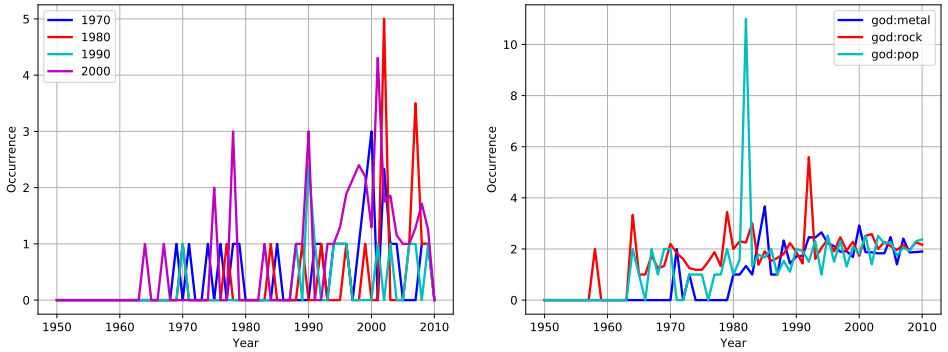

Figure 6: (a) Occurrences of *1970*, *1980*, *1990* and *2000* and (b) the occurrence of *god* in lyrics from 1950 to 2010. The counts are normalized by the number of tracks per year.

prediction system, but we found a rather interesting fact regarding word features. Namely, that the best representation happens to be the binary weighting (see Table 5). These results suggest that the presence or absence of a term or n-gram is most likely a better indicator of the genre than the importance weighted distribution. This is similar to sentiment analysis, where binary word counts usually induce a better performance [23].

From the confusion matrix shown in Figure 5, we can see which genres are problematic to predict: reggae and pop, rock and pop, and electronic and pop are the most easily confusable in our system, while falsely predicting a track as being of pop genre is moderately high for every genre (see column 7 of the confusion matrix). One possible explanation of this phenomenon could be that indeed, analyzing the lyrics of these music genres, no significant differences can be found between them. Another explanation of the above is the proximity of the genres in question, for example in case of rock and pop. The Wikipedia article about rock music[37] says the following: "Like pop music, lyrics often stress romantic love but also address a wide variety of other themes that are frequently social or political." Also, in the article of pop music[38] we can find the following: "'Pop' and 'rock' were roughly synonymous terms until the late 1960s, when they became increasingly differentiated from each other." This might imply joining together some of the above categories and studying the labels of the misclassified tracks.

[37] `https://en.wikipedia.org/wiki/Rock_music`
[38] `https://en.wikipedia.org/wiki/Pop_music`

The constructed dataset can also be used in *culturomics* [39]. Figure 6 compares the occurrences of *1970*, *'80*, *'90* and *2000*, as well as the occurrence of *god* in lyrics from 1950 to 2010.

The compiled dataset was made publicly available to stay at the disposal of possible future MIR, psychological, linguistics, etc. research.[39] We publish the following datasets: (a) the dataset as described in Section 3.4.2 and (b) bag-of-n-grams representations of the lyrics, $n \in \{1, 2, 3\}$.

Though the primary goal of this paper was the description of the newly compiled LyricWiki-based dataset, lyrics-based genre classification can also be further studied in detail. A good starting point would be the more precise determination of rhymes, studying other phonetic algorithms like Metaphone and Double Metaphone [45, 46], or applying automatic rhyme detection methods as in [22]. Another direction would be the application of word and document embedding methods for lyrics representation [40, 41, 28, 44]. Since the parameters of our system were selected arbitrarily, a compulsory next step would be tuning these using cross-validation. Applying large-scale semi-supervised methods for learning [7] is also a possible future direction one can investigate. Finally, but not less important, we mention the assessment of the importance of different rhyme and statistical text features in predictions.

References

[1] C. Apté, F. Damerau, and S. M. Weiss. Toward language independent automated learning of text categorization models. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 23–30, Dublin, Ireland, 1994. Springer-Verlag. ⇒171

[2] J. Atherton and B. Kaneshiro. I said it first: Topological analysis of lyrical influence networks. In *ISMIR*, pages 654–660, 2016. ⇒162

[3] T. Bertin-Mahieux, D. P. W. Ellis, B. Whitman, and P. Lamere. The million song dataset. In A. Klapuri and C. Leider, editors, *ISMIR*, pages 591–596. University of Miami, 2011. ⇒159, 160

[4] M. Besson, F. Faita, I. Peretz, A.-M. Bonnel, and J. Requin. Singing in the brain: Independence of lyrics and tunes. *Psychological Science*, 9(6):494–498, 1998. ⇒160, 169

[5] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006. ⇒174

[6] M. J. T. Carneiro. Towards the discovery of temporal patterns in music listening using Last.fm profiles. Master's thesis, Faculdade de Engenharia da Universidade do Porto, 2011. ⇒170

[39] See Section 3.5.

[7] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 2006. ⇒178

[8] K. Choi, Gy. Fazekas, M. Sandler, and K. Cho. Convolutional recurrent neural networks for music classification. In *ICASSP*, pages 2392–2396. IEEE, 2017. ⇒ 161

[9] K. Choi, J. H. Lee, X. Hu, and J. S. Downie. Music subject classification based on lyrics and user interpretations. In *Proceedings of the 79th ASIS&T Annual Meeting: Creating Knowledge, Enhancing Lives through Information & Technology*. American Society for Information Science, 2016. ⇒161

[10] H. Corona and M. P. O'Mahony. An exploration of mood classification in the million songs dataset. In *12th Sound and Music Computing Conference*, Ireland, 2015. Music Technology Research Group, Department of Computer Science, Maynooth University. ⇒161

[11] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 2(2):215–242, 1958. ⇒174

[12] S. Dieleman, P. Brakel, and B. Schrauwen. Audio-based music classification with a pretrained convolutional network. In *ISMIR*, pages 669–674, 2011. ⇒161

[13] S. Dieleman and B. Schrauwen. Multiscale approaches to music audio feature learning. In *ISMIR*, pages 116–121, 2013. ⇒161

[14] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *ICASSP*, pages 6964–6968. IEEE, 2014. ⇒161

[15] D. P. W. Ellis. Extracting information from music audio. *Communications of the ACM*, 49(8):32–37, 2006. ⇒160

[16] R. J. Ellis, Z. Xing, J. Fang, and Y. Wang. Quantifying lexical novelty in song lyrics. In *ISMIR*, pages 694–700, 2015. ⇒162

[17] M. Fell and C. Sporleder. Lyrics-based analysis and classification of music. In J. Hajic and J. Tsujii, editors, *COLING*, pages 620–631. ACL, 2014. ⇒ 159, 161, 170, 172, 174

[18] J. Fürnkranz. A study using n-gram features for text categorization, 1998. ⇒ 171

[19] W. H. Gomaa and A. A. Fahmy. A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13):13–18, April 2013. ⇒172

[20] S. Gupta. Music data analysis: A state-of-the-art survey. *arXiv preprint arXiv:1411.5014*, 2014. ⇒160

[21] P. Hamel and D. Eck. Learning features from music audio with deep belief networks. In *ISMIR*, volume 10, pages 339–344, 2010. ⇒161

[22] H. Hirjee and D. G. Brown. Using automated rhyme detection to characterize rhyming style in rap music. *Empirical Musicology Review*, 5(4), 2010. ⇒ 172, 178

[23] D. Jurafsky and J. H. Martin. *Speech and language processing*. 2017. 3rd edition draft. ⇒177

[24] A. Kiss. Classification of hungarian folk music from Transylvania with convolutional neural networks. Master's thesis, Faculty of Mathematics and Computer Science, Babeş–Bolyai University, Romania, 2018. ⇒161

[25] P. Knees and M. Schedl. *Music Similarity and Retrieval*. Springer, Berlin–Heidelberg, 2016. ⇒160

[26] P. Knees, M. Schedl, and G. Widmer. Multiple lyrics alignment: Automatic retrieval of song lyrics. In *ISMIR*, pages 564–569, 2005. ⇒160

[27] D. E. Knuth. *The Art of Computer Programming, Vol. 3: Sorting and Searching*. Addison-Wesley, Reading, MA, 1973. ⇒172

[28] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1188–1196, 2014. ⇒178

[29] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710, 1966. ⇒172

[30] T. L. H. Li, A. B. Chan, and A. Chun. Automatic musical pattern feature extraction using convolutional neural network. In *Proc. Int. Conf. Data Mining and Applications*, 2010. ⇒161

[31] D. Liang, H. Gu, and B. O'Connor. Music genre classification with the million song dataset. Technical report, Machine Learning Department, CMU, 2011. ⇒161, 170

[32] J. P. G. Mahedero, A. Martinez, P. Cano, M. Koppenberger, and F. Gouyon. Natural language processing of lyrics. In *ACM Multimedia*, pages 475–478. ACM, 2005. ⇒170

[33] R. Malheiro, R. Panda, P. Gomes, and R. Paiva. Classification and regression of music lyrics: Emotionally-significant features. In *8th International Conference on Knowledge Discovery and Information Retrieval*, Porto, Portugal, 2016. ⇒161

[34] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. ⇒174

[35] M. Mauch, R. M. MacCallum, M. Levy, and A. M. Leroi. The evolution of popular music: USA 1960–2010. *Royal Society Open Science*, 2(5), 2015. ⇒163

[36] R. Mayer, R. Neumayer, and A. Rauber. Rhyme and style features for musical genre classification by song lyrics. In J. P. Bello, E. Chew, and D. Turnbull, editors, *ISMIR*, pages 337–342, 2008. ⇒161, 170, 173, 174

[37] R. Mayer and A. Rauber. Music genre classification by ensembles of audio and lyrics features. In A. Klapuri and C. Leider, editors, *ISMIR*, pages 675–680. University of Miami, 2011. ⇒159

[38] C. McKay and I. Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved? In *ISMIR*, pages 101–106, 2006. ⇒161, 169, 170

[39] J.-B. Michel, Y. K. Shen, A. P. Aiden, A. Veres, M. K. Gray, J. P. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. A. Nowak, and E. Lieberman Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 331:176–182, 2011. ⇒178

[40] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. ⇒178

[41] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013. ⇒178

[42] L. Németh. Automatic non-standard hyphenation in OpenOffice.org. *TUGboat*, 27(1):32–37, 2006. ⇒172

[43] F. Pachet and D. Cazaly. A taxonomy of musical genres. In J.-J. Mariani and D. Harman, editors, *RIAO*, pages 1238–1245. CID, 2000. ⇒170

[44] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014. ⇒178

[45] L. Philips. Hanging on the metaphone. *Computer Language Magazine*, 7(12):38, December 1990. ⇒178

[46] L. Philips. The double metaphone search algorithm. *C/C++ Users Journal*, 18(6), June 2000. ⇒178

[47] J. Pons, T. Lidy, and X. Serra. Experimenting with musically motivated convolutional neural networks. In *CBMI*, pages 1–6. IEEE, 2016. ⇒161

[48] R. Priedhorsky, J. Chen, S. T. K. Lam, K. Panciera, L. Terveen, and J. Riedl. Creating, destroying, and restoring value in Wikipedia. In *Proceedings of the 2007 international ACM conference on Supporting group work*, pages 259–268. ACM, 2007. ⇒160

[49] S. Reddy and K. Knight. Unsupervised discovery of rhyme schemes. In *ACL (Short Papers)*, pages 77–82. The Association for Computer Linguistics, 2011. ⇒172

[50] G. Salton, A. Wong, and A. C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18:229–237, 1975. ⇒159, 171

[51] H. Schreiber. Improving genre annotations for the million song dataset. In M. Müller and F. Wiering, editors, *ISMIR*, pages 241–247, 2015. ⇒170

[52] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. ⇒174

[53] S. Sigtia and S. Dixon. Improved music feature learning with deep neural networks. In *ICASSP*, pages 6959–6963. IEEE, 2014. ⇒161

[54] A. Singhi and D. G. Brown. Are poetry and lyrics all that different? In H.-M. Wang, Y.-H. Yang, and J. H. Lee, editors, *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27–31, 2014*, pages 471–476, 2014. ⇒161

[55] B. L. Sturm. An analysis of the gtzan music genre dataset. In *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, pages 7–12. ACM, 2012. ⇒160

[56] B. L. Sturm. A survey of evaluation in music genre recognition. In *International Workshop on Adaptive Multimedia Retrieval*, pages 29–66. Springer, 2012. ⇒161

[57] A. Swartz. MusicBrainz: a semantic Web service. *IEEE Intelligent Systems*, 17(1):76–77, 2002. ⇒164

[58] A. Tsaptsinos. Lyrics-based music genre classification using a hierarchical attention network. In *ISMIR*, pages 694–701, 2017. ⇒162

[59] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002. ⇒160

[60] E. Zangerle, M. Tschuggnall, S. Wurzinger, and G. Specht. Alf-200k: Towards extensive multimodal analyses of music tracks and playlists. In *European Conference on Information Retrieval*, pages 584–590. Springer, 2018. ⇒162

Received: September 7, 2018 • Revised: December 5, 2018

Acta Univ. Sapientiae, Informatica **10,** 2 (2018) 183–217

DOI: 10.2478/ausi-2018-0010

Sampling k-partite graphs with a given degree sequence

Koko K. Kayibi
Department of Mathematics, University of Bristol, United Kingdom
email: kokokayibi@yahoo.co.uk

U. Samee
Department of Mathematics, Islamia College for Science and Commerce, Srinagar, India
email: drumatulsamee@gmail.com

Shariefuddin Pirzada
University of Kashmir, Srinagar, India
email: pirzadasd@kashmiruniversity.ac.in

Muhammad Ali Khan
Department of Mathematics and Computer Science, University of Lethbridge, Canada
email: ma.khan@uleth.ca

Abstract. The authors in the paper [15] presented an algorithm that generates uniformly all the bipartite realizations and the other algorithm that generates uniformly all the simple bipartite realizations whenever A is a bipartite degree sequence of a simple graph. The running time of both algorithms is $\mathcal{O}(m)$, where $m = \frac{1}{2}\sum_{i=1}^{n} a_i$. Let $A = (A_1 : A_2 : \ldots : A_k)$ be a k-partite degree sequence of a simple graph, where A_i has n_i entries such that $\sum n_i = n$. In the present article, we give a generalized algorithm that generates uniformly all the k-partite realizations of A and another algorithm that generates uniformly all the simple k-partite realizations of A. The running time of both algorithms is $\mathcal{O}(m)$, where $m = \frac{1}{2}\sum_{i=1}^{n} a_i$.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C07, 65C05
Key words and phrases: Degree sequence, contraction of a degree sequence, degree sequence bipartition, contraction of a graph, deletion of a graph, ecological occurrence matrix

1 Introduction

A k-*partite* graph G is a graph whose vertex set, denoted by $V(G)$, can be partitioned into k parts, $V_1(G), V_2(G), ..., V_k(G)$, such that two vertices in the same part are not adjacent. That is, if $E(G)$ denotes the edge set of G and $e = (v_i, v_j) \in E(G)$, then $v_i \in V_s(G)$ and $v_j \in V_r(G)$ such that $s \neq r$. An edge $e \in E(G)$ is said to be a *multiple* edge if there is another edge f incident to the same vertices. Following Matroid Theory terminology, we say that e and f are *parallel*. A *simple* k-partite graph is a k-partite graph with no multiple edges and loops. The *degree* of a vertex v_i, denoted by a_i, is defined as the number of edges incident to v_i with a loop contributing twice to the degree of v_i. The degree sequence of a graph G is formed by listing the degrees of vertices of G. If $A = (a_1, a_2, ..., a_n)$ is a sequence of integers and G is a k-partite graph that has A as its degree sequence, we say that G is a *realization* of A, and such a sequence of integers is called a k-*partite degree sequence*. Thus entries of A can be partitioned as $A_1, A_2, ..., A_k$, where A_i denotes the degree sequence of the part $V_i(G)$. In the sequel, we denote a k-partite degree sequence A as $(A_1 : A_2 : ... : A_k)$ and the sequence $(A_1 : A_2 : ... : A_k)$ is called a k-*partition* of A.

Observation 1 *An easy observation used in the sequel is that, if* $A = (A_1 : A_2 : ... : A_k)$ *is a* k*-partite degree sequence having* n *entries, and* A_i *has* n_i *entries, then the following is true.*

1. $n_1 + n_2 + ... + n_k = n$

2. For every i *such that* $1 \leq i \leq k$*, the maximal entry of* A_i *is less or equal to* $\sum_{j \neq i} n_j$.

The *Degree Sequence Problem* is to find some or all graphs with a given degree sequence [20]. More detailed analysis of the Degree Sequence Problem and its relevance can be found in [18]. Several algorithms are known to construct random realizations of degree sequences and each one of them has its strengths and limitations. Most of these algorithms can be fitted in two categories: MonteCarlo Markov chains methods based on edge-swappings [5, 8, 9, 10, 11, 13, 14, 16, 17] and random matching methods [1, 2, 3, 4, 21]. In particular, algorithms proposed in [1, 3, 7] are based on inserting edges sequentially according to some probability scheme. The basic ideas of the algorithm presented in the present paper have already been used successfully to sample uniformly all the simple realizations of a bipartite degree sequence in [15]. Those basic ideas may be seen as implementing a "dual sequential

method", as it inserts sequentially vertices instead of edges. For other undefined notations and terminology in graph theory, the readers are referred to [19].

Indeed, in the theory of the Tutte polynomial, there are two operations, deletion and contraction, that are dual of each other, see [6] for more details on this topic. Let G be a graph having n vertices and m edges. The operation of deleting the edge $e = (v_i, v_j)$ from G consists of removing the edge e and leaving anything else unchanged. The graph thus obtained, denoted by $G\backslash e$, is a graph on n vertices and $m-1$ edges where h the degrees of both the vertices v_i and v_j go down by 1. The operation of contracting the graph G by $e = (v_i, v_j)$ consists of deleting the edge e and identifying the vertex v_i and v_j. The graph thus obtained, denoted by G/e, is a graph on $n-1$ vertices and $m-1$ edges, where the new vertex obtained by identifying v_i and v_j has degree $a_i + a_j - 2$. Deletion is said to be the dual of contraction as the incidence matrix of $G\backslash e$ is orthogonal to the incidence matrix of G^*/e, where G^* is the dual of G if G is planar.

If A is a degree sequence having n entries, it can be easily shown that random matching methods used in [1, 2, 3, 4, 21] are equivalent to starting from a known realization G of A, delete all the edges one by one, and keeping track of the degrees of vertices after each deletion, until one reaches the empty graph having n vertices. Then, reconstructing a random realization of A consists of taking the reverse of the deletion. That is, starting from the empty graph on n vertices, re-insert edges one by one by choosing which edge to insert according to the degrees of the vertices and some probability scheme depending of the stage where the algorithm is at, and subject to not getting double edges if one would like to get simple graphs or not linking two vertices on the same part if one wants to get bipartite graphs. The algorithm presented in this paper is based on the dual operation of contraction that is slightly modified to suit our purpose. It is equivalent to starting from a known realization G of A, contract all the edges one by one, and keeping track of the vertices after each contraction, until one reaches the graph on one vertex and $\frac{1}{2}\sum_1^n a_i$ loops. Then, reconstructing a random realization of A consists of reversing the process of contraction. That is, starting from the graph on one vertex and $\frac{1}{2}\sum_1^n a_i$ loops, the algorithm re-inserts vertices one by one by choosing which vertex to connect to which according to degrees of the vertices and some probability that depends on the stage of the algorithm.

While algorithms that are based on Markov chains [14, 17] or on reversing the deletion operation [1, 3] are easy to implement, our algorithm seems more complex as one has to satisfy not only the degrees of the vertices, but also

some added graphical structures imposed by the contraction. But this is more of a bonus than an inconvenience, as apart from the fact the the running time is even better, the extra structure allows an easier analysis of the algorithm. Moreover, the internal structure imposed by the contraction operation allows the algorithm to avoid most of the shortcomings of the previous algorithms. Indeed, not only the algorithm never restarts, but the algorithm also allows to sample all bipartite realizations with equal probability, making their approximate counting much easier than by the importance sampling used in [1, 3]. Better still, this technique can be extended, as we do it in the present paper, to construct k-partite realizations of a k-partite degree sequence A, for $k \geq 3$, where a k-partite degree sequence is defined in a natural way by extending the definition of a bipartite degree sequence.

This paper is organized as follows. First we define a recursion chain of a degree sequence, then we present routines for constructing all k-partite realizations. These basic routines are then modified to get a uniform distribution on the set of all k-partite realizations. Then comes the section that presents criteria to generate simple realizations graphs only. We modify our routines to new routines that generate all simple k-partite realizations uniformly at random.

2 Construction of all k-partite realizations of given degrees

2.1 Recursion chain of degree sequences

Let G be a graph with n vertices and m edges. Throughout we assume that the vertices of G are labelled $v_1, v_2, \ldots, v_n$. Let $A = (a_1, \ldots, a_n)$ be the degree sequence of G, where a_i denotes the degree of the vertex v_i. Define an arithmetic operation on A, called *contraction* as follows. For an ordered pair (a_i, a_j) of entries a_i and a_j of A with $i \neq j$, the operation of *contraction* by (a_i, a_j) means changing a_i to $a_i + a_j$ and deleting the entry a_j from A. We write $A/(i, j)$ to denote the new sequence thus obtained. The sequence $A/(i, j)$ is called the (i, j)-*minor* or simply a *minor* of A. The following example illustrates the definition given above for a tripartite degree sequence.

Example 2 *Let* $A = (5, 5 : 4, 2 : 3, 3, 2)$ *where* $a_1 = 5$, $a_2 = 5$, $a_3 = 4$ *and* $a_4 = 2, a_5 = 3, a_6 = 3, a_7 = 2$. *We have* $A/(1, 2) = (10, 4, 2, 3, 3, 2)$ *and* $A/(4, 2) = (5, 4, 7, 3, 3, 2)$.

Let A be sequence of integers. Then A is said to be *graphic* if there is a graph G, not necessarily simple nor k-partite, such that G has A as its degree sequence. Moreover, it is trivial to observe that a sequence of integers is graphic if and only if the sum of its entries is even. Further, we have the following observation.

Theorem 3 *A sequence* A *is graphic if and only if all its minors are graphic.*

Proof. Obviously, if A is graphic, then $A/(a_i, a_j)$ is graphic as, by definition of contraction, the sum of its entries is even. Now suppose that $A/(a_i, a_j)$ is graphic and G'' is a realization of $A/(a_i, a_j)$. To prove that A is also graphic, we present an algorithm, much used in the sequel, that constructs a realization of A, denoted by G, from G''.

Algorithm AddVertex()
Step 1. To G'' add an isolated vertex labelled v_j (as in Figure 1).
Step 2 If the degree of v_j is a_j, stop, output G. Else
Step 3. Amongst the a'_i edges incident to v_i, counting loops twice, choose one edge
$e = (v_i, v_k)$ with probability $\pi(e)$ and connect e to v_j so that e becomes (v_j, v_k). Go to Step 2.

Now, in G the degree of v_j is a_j, by Step 2 of algorithm AddVertex(). Moreover, by the definition of contraction, the degree of v_i is equal to $a_i + a_j$ in G''. Since AddVertex() takes a_j edges away from v_i, the degree of v_i is a_i in G. Moreover all the other vertices are left unchanged by AddVertex(). Thus G is a realization of A. □

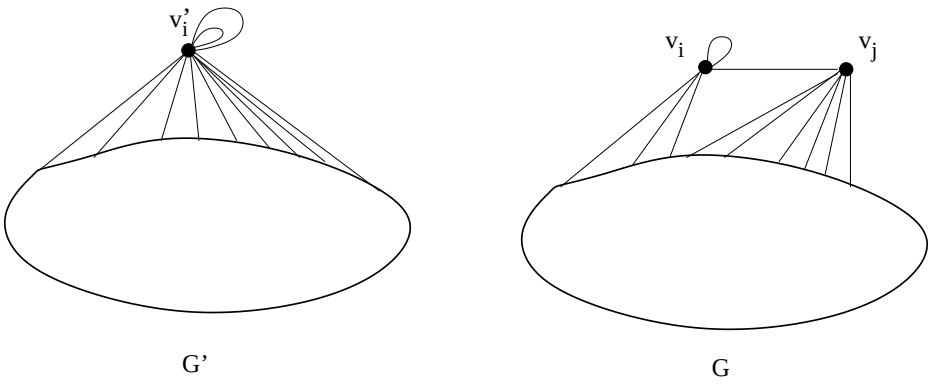

Figure 1: Construction of a graph G from its minor G''

If AddVertex() chooses the edge $e = (v_i, v_k)$ and connects e to v_j so that e becomes (v_j, v_k), we say that AddVertex() (or v_i, or v_k) concedes e to v_j. To help intuition, observe that if G'' is a realization of $A/(a_i, a_j)$ and G is a realization of A constructed by AddVertex(), then G'' is obtained from G by contraction of the edge (v_i, v_j). Now, mimicking the process of recursive contraction of matroid as used in the theory of the Tutte polynomial, we define a process of recursive contraction for a degree sequence. A *recursion chain* of a degree sequence A is a unary tree rooted at A, where nodes are integer sequences and every node, except for the root, is a minor of the preceding one. The recursive procedure of contraction is carried on from the root A until a node with a single entry is reached. As for the Tutte polynomial, the amazing fact, which is then used to construct all the realizations of A, is that the order of contraction is immaterial. Despite this basic fact, we still impose a particular order to ease many proofs in the sequel.

Notes on notations: For the sake of convenience, we refer to a node of a recursion chain of a degree sequence A by $A^{(i)}$, where i is the number of entries in the node. Thus we denote the root A by $A^{(n)}$, the next node by $A^{(n-1)}$, and so on until the last node $A^{(1)}$. Similarly, we denote by $G^{(i)}$ the realization of $A^{(i)}$. The n entries of A are labelled from 1 to n. To keep tract of the vertices, we preserve the labelling of entries of A into its minors so that when a contraction by the pair (a_i, a_j) is performed, the new vertex is labelled a_i, the label a_j is deleted, and all the other entries keep the labelling they have before the contraction.

In this paper, we consider the recursion chain, called the k-*partiteaccumulating recursion chain*, constructed as follows. Let $A = (A_1 : A_2 : ... : A_k)$ be a k-partite degree sequence. We label each entry as $a_{s,r}$, where s ranges from 1 to n and r ranges from 1 to k, so that the entry $a_{s,r}$ belongs to A_r.

Example 4 *Let* $A = (5,5 : 4,2 : 3,3,2)$. *We order entries of* A *as* $a_{1,1} = 5$, $a_{2,1} = 5$, $a_{3,2} = 4$, $a_{4,3} = 3$, $a_{5,3} = 3$, $a_{6,2} = 2$, $a_{7,3} = 2$. *Thus* $\vec{A} = (5,5,4,3,3,2,2)$.

Note. The vertex having degree $a_{s,r}$ is denoted by $v_{s,r}$. But, to avoid clustering the notation, we sometimes just right a_s or v_s, when we deem not necessary to specify the part A_r corresponding to the degree $a_{s,r}$.

Algorithm ConstructKpartiteRecursionChain()
Given an ordered k-partite degree $\vec{A}$. Let $i = n$.
Step 1 If $i = 1$, stop, return $\{A^{(1)}, A^{(2)}, ..., A^{(n)}\}$. Else
Step 2 Let $A^{(i-1)} = A^{(i)}/(1, i)$. That is, get the $(i-1)^{th}$ recursive minor of A by contracting the $(i)^{th}$ recursive minor by its first entry and the last entry.
Step 3 Decrement i by 1 and go back to Step 1.

We denote the accumulation recursion chain of $\vec{A}$ by $W = (A^{(1)}, A^{(2)}, ..., A^{(n)})$.

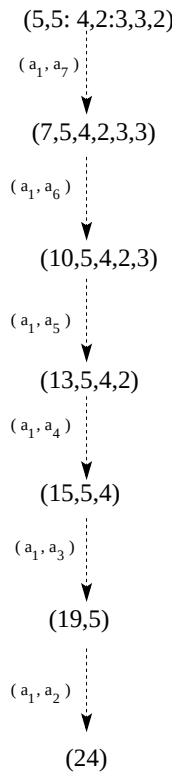

Figure 2: The accumulating recursion chain of the tripartition $[5, 5 : 4, 2 : 3, 3, 2]$.

The following is an algorithm for constructing a k-partite realization if A is a k-partite degree sequence. The graph constructed below is not necessarily simple. Loosely speaking, this algorithm consists of reversing the recursive process of contraction as implemented by ConstructKpartiteRecursionChain(). The algorithm starts from $G^{(1)}$ the sole realization of $A^{(1)}$, and by calling

AddVertex() recursively, it constructs $G^{(2)}$, then $G^{(3)}$, and so on until $G^{(n)}$, that is a realization of $A^{(n)} = A$. The only condition imposed on the choice to built edges is that if the vertex to insert has degree $a_{s,r}$, then we label the vertex as $v_{s,r}$. Then AddVertex links a vertex $v_{x,y}$ to $v_{s,r}$ only if $j \neq r$, unless $x = 1$. Recall that $\delta(x, y) = 1$ if $x = y$ and $\delta(x, y) = 0$ otherwise.

Algorithm ConstructKpartiteRealization()

Given $W = (A^{(1)}, A^{(2)}, ..., A^{(n)})$, the Kpartite accumulating recursion chain of A, we do the following.

Step 1. Let $i = 1$ and build the realization of the node $A^{(1)}$, denoted by $G^{(1)}$, which is the graph consisting of one vertex and m loops, where $m = \frac{1}{2}\sum_{i=1}^{n} a_i$.

Step 2. Let $G = G^{(i)}$. If G has n vertices, stop, return G. Else,

Step 3. Using $G^{(i)}$ and $A^{(i+1)}$ as input, Call Algorithm AddVertex() to construct $G^{(i+1)}$ as a realization of $A^{(i+1)}$. If $v_{s,r}$ is the vertex being inserted, then AddVertex only constructs edges $(v_{x,y}, v_{s,r})$ where $\delta(y, r) = 0$ and $\delta(x, s) = 0$, unless $x = 1$. Increment i by 1, go back to Step 2.

See Figure 3 for an illustration of Algorithm ConstructBipartiteRealization().

The following definitions are needed in the sequel. In the process of contraction implemented by the accumulating recursion chain, we observe that the degrees are accumulating on $a_{1,1}$. This is equivalent to say that edges are accumulating on $v_{1,1}$ as $v_{1,1}$ seems to 'swallow' the other vertices one by one. Hence, when reversing the contraction operation in ConstructKpartiteRealization(), vertex $v_{1,1}$ plays the role of the 'mother that spawns' all the other vertices one by one and concedes some edges to them according to their degrees. Thus, AddVertex() can attach an edge e to a new vertex $v_{s,r}$ only if e is incident to $v_{1,1}$. This observation prompts the following formal definitions. Let A be a k-partite degree sequence where A_i has n_i entries such that $\sum_i n_i = n$. The $(s, t)^{th}$ stage of ConstructKpartiteRealization() is the iteration where the algorithm inserts the t^{th} edge of the vertex $v_{s,r}$. At the $(s, t)^{th}$ stage an edge is *available* if it is a loop incident to $v_{1,1}$ or $e = (v_{1,1}, v_{x,y})$ where $y \neq r$ and $x < s$. An edge e is *lost* otherwise. Let E_{av} denote the set of all available edges and E_{v_j} the set of edges (v_1, v_j). We recall that an edge $e = (v_{1,1}, v_{x,y})$ is *conceded* if AddVertex() disconnects it from $v_{1,1}$ so that e becomes $e = (v_{x,y}, v_{s,r})$ for some vertex $v_{s,r} \neq v_{1,1}$. We then say that $v_{1,1}$ (or sometimes E_{v_j} or just v_j) concedes the edge e. A vertex v_s having degree a_s is *fully inserted* if a_s edges are conceded to it. A graph G is said to be *(re)constructed* if it is an output of

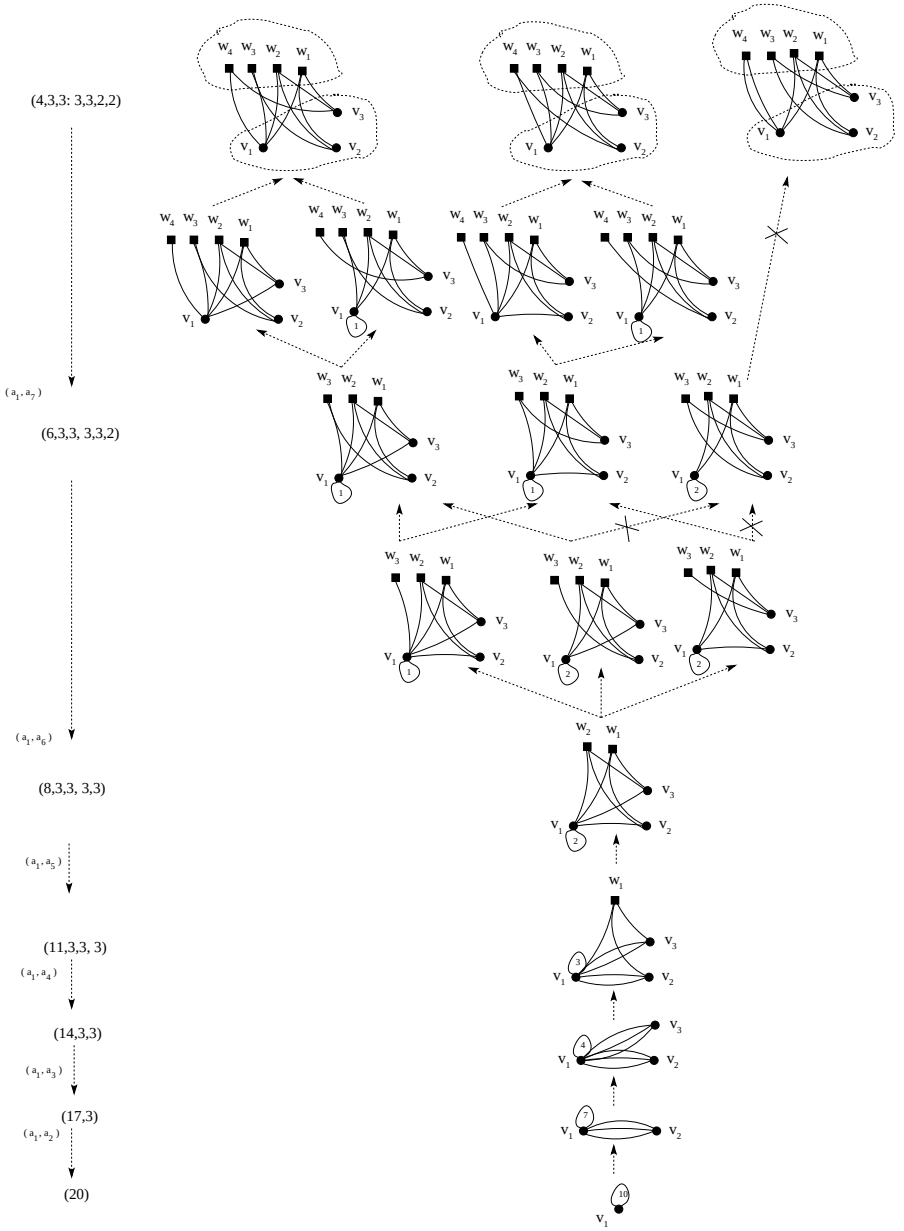

Figure 3: Random reconstruction tree of $(4,3,3:3,3,2,2)$. The level of $\mathcal{T}$ on the same height as the degree sequence $A^{(i)}$ corresponds to all the graphs having $A^{(i)}$ as their degree sequence.

ConstructKpartiteRealization(). Now we state and prove a paramount result of this paper.

Theorem 5 *Let* $A = (a_1, a_2, ..., a_n) = (A_1 : A_2 : ... : A_k)$ *be a* k*-partite degree sequence having* n *entries where* A_i *has* n_i *entries, such that* $\sum n_i = n$ *and* $m = \frac{a_i + a_2 + ... + a_n}{2}$ *. Let* W *be the* k*-partite recursion chain of* A*. Then Algorithm ConstructKpartiteRealization() constructs in time linear on* m *a* k*-partite graph* G *having* n *vertices and* m *edges such that* G *is a realization of* A*. Moreover, every* k*-partite realization of* A *can be constructed in this way.*

Proof. By Algorithm AddVertex(), the graph $G^{(n)}$ output by Algorithm ConstructKpartiteRealization() is assured to be a realization of A. We need only to prove that $G^{(n)}$ is k-partite. Now, the routine AddVertex() constructs edges $(v_{s,x)}, v_{r,y}$ only if $\delta(s, r) = \delta(x, y) = 0$, unless $s = x = 1$. Thus vertices corresponding to degrees in different parts A_x and A_y are never adjacent. Thus, we only have to show that in G^n, $v_{1,1}$ is not adjacent to any vertex $v_{j,1}$. We need the following fact.

Observation 6 *Suppose that* A *is a* k*-partite degree sequence. From the* v_1^{th} *iteration of ConstructKpartiteRealization(), the number of available edges is equal to the number of edges left to be inserted until ConstructKpartiteRealization() terminates.*

This is because the number of available edges at the end of $(v_1)^{th}$ is equal to half the sum of degrees $a_i \in A_1$. By the definition of the bipartite degree sequence, this number is equal to half the sum of degrees $a_j \in A_2$

Now, to prove that $G = G^n$ is k-partite, in the proof of Theorem 5, suppose for a contradiction, G contains an edge $e = (v_{1,1}, v_{j,1})$. Consider the graph H obtained from G by contracting all the edges not incident to any vertex $v_{j,1}$. It is easy to see that the degree sequence of H, denoted by B, is the degree sequence obtained from A by contracting the entries corresponding to vertices contracted in G. Thus B is a k-partite degree sequence. Now, suppose that ConstructKpartiteRealization() outputs a realization K of B with en edge $(v_{1,1}, v_{j,1})$. Then by Observation 6 one vertex of K is not fully inserted which is a contradiction. Thus ConstructKpartiteRealization() reconstructs H has no edge $(v_{1,1}, v_{j,1})$. This is the final contradiction we are looking for.

It remains to prove that any k-partite realization of A can be constructed this way. We recall that v_i (a_i) denotes the i^{th} vertex (degree) in the ordering, regardless of the second index. So, let G be a realization of A and let $e = (v_i, v_j)$ be any edge of G such that vertex v_i has degree a_i and vertex v_j has degree

a_j. Also, suppose that the vertcies v_i and v_j were inserted at the i^{th} and j^{th} iteration of ConstructKpartiteRealization() respectively. We need to show that at the j^{th} iteration, there is a positive probability to have an edge e which is incident to v_i and e is available. If not, that is, at the j^{th} iteration all the edges incident to v_i are lost. Now all the edges incident to v_i are lost before the j^{th} iteration only if, at some stage of the running of Algorithm ConstructKpartiteRealization(), there are only the edges that are available and they are exhausted before reaching the j^{th} iteration. Thus, at the j^{th} iteration there are no more available edges. Especially, there are no loops incident to v_1. But this means that $a_1 + a_2 + ... + a_j \geq 2m$, contradicting the definition of accumulating recursion chain.

As for the running time, Algorithm ConstructKpartiteRealization() calls Algorithm AddVertex() once for every new vertex v_k to insert. If v_k has degree a_k, then Algorithm AddVertex() has to go through a_k iterations to insert the a_k edges of v_k. Hence the total number of iterations to terminate ConstructKpartiteRealization() is $a_1 + a_2 + ... + a_n = 2m$. □

2.2 Sampling all k-partite realizations uniformly

Although Theorem 5 shows that the routine ConstructKpartiteRealization() can construct a realization of A in linear time, we need the next result to show that it can construct any k-partite realization of A with equal probability, provided we define the probability $\pi(e)$ with which AddVertex() has to insert the edge e.

We recall that if at its s^{th} iteration ConstructKpartiteRealization() is to insert the vertex v_s that has degree a_s, (regardless of the second index of v_s), then ConstructKpartiteRealization() has to *call* AddVertex() that has to go through a_s iterations. Let the $(s,t)^{th}$ stage of ConstructKpartiteRealization() be the iteration, where AddVertex() inserts the t^{th} edge of the s^{th} vertex and let $G^{(s,t)}$ denote the graph obtained at that $(s,t)^{th}$ stage. With this notation, let $G^{(s)}$ be the graph $G^{(s,a_s)}$.

The *random reconstruction tree*, denoted by $\mathcal{T}$, is a directed rooted tree, where the root is the sole realization of the degree sequence $A^{(1)}$, and the $(s,t)^{th}$ level contains all those possible graphs obtainable after inserting the t^{th} edge of the s^{th} vertex, and there is an arc from a graph H at level i to the graph G at level $i+1$ if it is possible to move from H to G by the concession of a single available edge. Realizations of A are thus the leaves of the tree $\mathcal{T}$. With this formalism, sampling a random k-partite realization of the degree sequence A is equivalent to performing a random walk from the root until a

leaf is reached, and every step of the random walk consists of walking along a random arc of $\mathcal{T}$. See Figure 3 for an illustration.

Observations about uniform sampling

(1) Let G^1 denote the root of $\mathcal{T}$ and G^1 be the graph on a single vertex and m loops. So the stage $(2,1)$ of ConstructKpartiteRealization consists of inserting the first edge of second vertex v_2. Suppose at some stage (s,t) of ConstructKpartiteRealization(), the random walk along $\mathcal{T}$ is at the graph $G^{(s,t)}$ with probability $\pi(G^{(s,t)})$ and that AddVertex() is to concede an edge e to v_s. Suppose also that the vertex v_j of $G^{(s,t)}$ has $|E_{v_j}|$ available edges. That is, v_j is connected to v_1 by $|E_{v_j}|$ parallel edges. Thus, the next level of $\mathcal{T}$ would contain $|E_{v_j}|$ identical graphs whose edge sets will contain the edge (v_j, v_s). Hence, if Algorithm AddVertex() chooses every available edge uniformly at random, the random walk will reach such a graph with probability $\frac{|E_{v_j}|}{|E_{av}|}$. Thus, if this graph is a leaf of $\mathcal{T}$, ConstructBipartiteRealization() will be biased toward it with a probability proportional to $\frac{|E_{v_j}|}{|E_{av}|}$. So, if the random walk has to reach each different child of $G^{(s,t)}$ with the same probability, we have to move from $G^{(s,t)}$ to its child obtained by adding the edge $e \in E_{v_j}$ with probability $\frac{|E_{v_j}|}{|E_{av}|}\frac{1}{|E_{v_j}|} = \frac{1}{|E_{av}|}$. Equivalently, let $|V(G^{(s)})|$ be the number of vertices already inserted up to the s^{th} iteration of ConstructKpartiteRealization() and let $|V'(G^{(s)})|$ be the number of vertices in $V(G^{(s)})$ which are adjacent to v_1. Obviously $v_1 \in V'(G^{(s)})$, if there is a loop incident to v_1. We may choose any vertex $v_j \in V'(G^{(s)})$ uniformly at random. If $e \in E_{v_j}$, then we concede e with probability $\frac{|V'(G^{(s)})|}{|V(G^{(s)})|}$.

(2) Suppose that $G^{(s,t+1)}$ is obtained from $G^{(s,t)}$ by the concession of edge e to vertex v_s. If in $G^{(s,t)}$ the vertex v_s is adjacent to b_s (with $b_s \leq a_s$) different vertices, then it is obvious that the random walk on $\mathcal{T}$ reaches $G^{(s,t)}$ in b_s different paths. Thus there is a bias proportional to b_s towards $G^{(s,t)}$. To remove this bias, the random walk would rather move away from $G^{(s,t)}$ with a reducing factor of $\frac{1}{b_s}$. Similarly $G^{(s+1,1)}$ is obtained from $G^{(s,a_s)}$ by the concession of edge e to vertex v_{s+1}. These two observations prompt the following extension of the routines Addvertex() and ConstructKpartiteRealization() to sample uniformly.

Algorithm BiasedAddvertex()

(A modification of Algorithm Addvertex() to get a uniform distribution, dividing by the number of vertices inserted up to the s^{th} iteration).
Step 1 To the graph G^s, add an isolated vertex called v_{s+1}. Let a_{s+1} be the number of edges to concede to v_{s+1} and let $j = 0$.
Step 2 If v_{s+1} is incident to a_{s+1} edges, return G^{s+1}. Else,
Step 3 Let b^i_{s+1} be the number of different vertices v_i which are incident to v_{s+1} after the insertion of its j^{th} edge, with $j \geq 1$. If $j = 0$, let b^j_{s+1} be the number of different vertices adjacent to v_s.
Step 4 Choose vertex v_q uniformly at random amongst all the vertices adjacent to v_1.
Step 5 If $e \in E_{v_q}$, then concede e to v_{s+1} with probability $\frac{|V'(G^{(s)})|}{|V(G^{(s)})|.b^j_{s+1}}$, where $V(G^{(s)})$ and $V'(G^{(s)})$ are respectively the set of vertices inserted up to the s^{th} iteration and the set of vertices in $V(G^{(s)})$ that are adjacent to v_1. Increment j by 1 and go back to Step 2.

Accordingly, we modify the routine ConstructBipartiteRealization() as follows.

Algorithm BiasedConstructKpartiteRealization()

Given the k-partite recursion chain of $A = (A_1 : A_2 : ... : A_k)$, where A_i has n_i entries such that $n_1 + n_2 + ... + n_k = n$, do the following.
Step 1. Let $s = 1$ and build the realization of the node $A^{(1)}$, denoted by $G^{(1)}$, that is the graph consisting of one vertex and m loops.
Step 2. Let $G = G^{(s)}$. If G has n vertices, stop, return G. Else,
Step 3. Using $G^{(s)}$ and $A^{(s+1)}$ as input, call Algorithm BiasedAddVertex() to construct $G^{(s+1)}$ as a realization of $A^{(s+1)}$. If $v_s = v_{s,y}$ BiasedAddvertex only concedes loops or edges $(v_{1,1}, v_{r,x})$ such that $\delta(x, y) = 0$. Increment s by 1, go back to Step 2.

Theorem 7 *(1) For the degree sequence* $A = (A_1 : A_2 : ... : A_k)$*, where* A_i *has respectively* n_i *vertices such that* $n_1 + n_2 + n_k = n$*, BiasedConstructKpartiteRealization() reaches every leaf of* $\mathcal{T}$ *uniformly at random with probability* $\frac{1}{\prod_{s=1}^{n} s^{a_{s+1}}}$.

(2) The set of leaves of $\mathcal{T}$ *is the set of realizations of* A

Proof. (1) Suppose that BiasedConstructKpartiteRealization() calls BiasedAd-

dVertex() to insert vertex v_s that has degree a_s, where $s > n_i$, and that BiasedAddvertex() is conceding the j^{th} edge of v_s. We know there are $|V(G^{(s)})|$ vertices inserted up to the s^{th} iteration. Now, if $e \in E_{v_t}$, then the routine BiasedAddVertex() will choose e with probability $\frac{1}{|V'(G^{(s)})|}$ and would concede it with probability $\frac{|V'(G^{(s)})|}{|V(G^{(s)})|.b_s^{j-1}}$. But the graph $G^{(s,j-1)}$ is reached through b_s^{j-1} different paths. Hence the graph $G^{(s,j)}$ obtained by conceding e to v_s will be sampled with probability given as

$$\frac{b_s^{j-1}}{|V'(G^{(s)})|}.\frac{|V'(G^{(s)})|}{|V(G^{(s)})|.b_s^{j-1}} = \frac{1}{|V(G^{(s)})|} = \frac{1}{s-1}.$$

Finally, we know that vertex v_s needs a_s iterations of BiasedAddVertex to be fully inserted. Hence, it takes altogether $\sum_{i=1}^{n} a_i = 2m$ iterations before BiasedConstructBipartiteRealization() terminates. Thus multiplying all the probabilities to move from one level to the next from the root to a leaf yields probability $\frac{1}{1^{a_1}2^{a_2}\ldots n^{a_n}}$.

(2) Obviously, by construction, every leaf of $\mathcal{T}$ is a realization of A. The proof that every realization of A is a leaf of $\mathcal{T}$ is given in the proof of Theorem 5. □

3 Construction of simple k-partite graphs

Up to now, BiasedConstructKpartiteRealization() generates all the k-partite realizations of the k-partite degree sequence A. But, it is easy to modify BiasedAddVertex() so that the output of BiasedConstructKpartiteRealization() is always a simple graph. One obvious condition is stated as follows.

(a) If the Algorithm is inserting the j^{th} edge of vertex v_s with $j > 1$ and $s > n_i$ with $1 \le i \le k$ and v_l is already adjacent to v_s, then no more available edge incident to v_l should be chosen. This will prevent BiasedConstructKpartiteRealization() from outputting graphs with multiple edges (v_s, v_l). Thus this condition is necessary but is not sufficient. Indeed, it is easy to see that the following must also apply.

(b) While inserting vertex v_s and avoiding choosing edges incident to v_l so as not to construct multiple edges (v_s, v_l), BiasedConstructKpartiteRealization() may fall into a stage where there are more edges incident to v_l than there are vertices left to be inserted, and G, the graph output by BiasedConstructBipartiteRealization() will then have a multiple edge (v_1, v_l).

Although (a) and (b) seem to contradict each other, this section defines all these conditions in a formal settings and proves that they can be satisfied simultaneously. Although the analysis seems long, this set of conditions are just inequalities involving the number of edges and vertices already inserted and the number of edges and vertices left to insert at each stage of the Algorithm. Moreover, checking these conditions at each iteration of BiasedAddVertex() requires checking $\mathcal{O}(n^2)$ inequalities altogether. Thus it does not add crucially to the running time.

Let $A = (A_1 : A_2 : ... : A_k)$ be a k-partite degree sequence of a simple graph, where A_i has n_i vertices such that $\sum n_i = n$. Suppose that BiasedConstructKpartiteRealization() is at the iteration of inserting vertex v_s. We recall that E_{av} represents the set of available edges at the $(s,t)^{th}$ stage. That is, edges that are incident to v_1 and vertices inserted before the iteration inserting the t^{th} edge of the s^{th} vertex. We also recall that E_{v_j} are the edges (v_1, v_j) at that stage. In particular, E_{v_1} is the set of loops incident to v_1. The set of *excess edges* of v_j, denoted by Ee_{v_j}, is the same as the set E_{v_j} if $v_j = v_{j,1}$. In particular, a loop is an excess edge incident on v_1. If $v_j = v_{j,r}$ for $r \neq 1$, the set of *excess edges* of v_j is the set E_{v_j} except one edge. That is $|Ee_{v_j}| = |E_{v_j}| - 1$.

The aim of this section is to show that it is possible to choose edges so that the algorithm never stalls after choosing a 'wrong' edge. If at its s^{th} iteration, Algorithm BiasedConstructKpartiteRealization() is inserting the vertex v_s that has degree a_s, then BiasedConstructKpartiteRealization() has to *call* the routine BiasedAddVertex() which has to go through a_s iterations. We recall that the $(s,t)^{th}$ stage of BiasedConstructKpartiteRealization() is the iteration, where BiasedAddVertex() inserts the t^{th} edge of the s^{th} vertex. Let $X_{s,t}$ and $|X|_{s,t}$ respectively denote a set and its cardinality at the $(s,t)^{th}$ stage of BiasedConstructKpartiteRealization().

To help the reader, we first introduce the motivation behind every definition. Obviously, if at some stage, the number of excess edges is greater than the number of edges left to be inserted, then BiasedConstructKpartiteRealization() can never produce a simple graph as the left-over of excess edges would result in a multiple edge or a loop in the final graph. Thus the choice of edges by BiasedAddvertex() must be such that this contingency never happens. This prompts the following definitions.

The $(s,t)^{th}$ stage of Algorithm BiasedConstructKpartiteRealization() is *critical* if

$$|Ee|_{st} = a_s - (t-1) + a_{s+1} + a_{s+2} + ... + a_n. \tag{1}$$

That is, the number of excess edges equals the number of edges left to insert until the end of BiasedConstructRealization(). The $(s,t)^{th}$ stage is *spoilt* if

$$|Ee|_{st} > a_s - (t-1) + a_{s+1} + a_{s+2} + ... + a_n. \tag{2}$$

That is, there are too many excess edges and whatever the future choices might be, a simple graph can never be output. A stage is *normal* if it is neither critical nor spoilt.

Now, at each stage of constructing a simple k-partite graph, every vertex $v_{r,x}$ must be connected to any other $v_{s,y}$, with $\delta(x,y) = 0$ unless $r = x = 1$, by at most one common edge. So, if some vertex $v_{r,x}$, with $r \neq 1$, has more excess edges than the number of vertices $v_{s,y}$, with $\delta(x,y) = 0$, left to be inserted, BiasedConstructKpartiteRealization() would never be able to get rid of all these multiple edges, which will then appear in the final graph. This prompts the following definition. Let $N(\bar{x})_{st}$ be the the set of vertices $v_{q,y}$ with $q < s$ and $y \neq x$. At the $(s,t)^{th}$ stage, the vertex $v_{r,x}$ with $r \leq s$ and $r \neq 1$ is *due* if

$$|Ee_{v_{r,x}}|_{st} = |N(\bar{x})|_{st}, \tag{3}$$

that is, $Ee_{v_{r,x}}$ has got as many excess edges as there are vertices left to be inserted to which it can concede an edge. The vertex $v_{r,x}$ is *overdue* if

$$|Ee_{v_{r,x}}|_{st} > |N(\bar{x})|_{st}, \tag{4}$$

that is, there are too many excess edges incident to $v_{r,x}$ and whatever the future choices might be, the Algorithm will never output a simple graph. The vertex $v_{r,x}$ is *undue* if it is neither due or overdue.

Now, although v_1 may concede many edges to a vertex v_s, there is also a limit, other than a_s, to the number of edges it can concede to v_s if the end result is to be a simple graph. For example, BiasedAddvertex() can construct at most 1 edge (v_1, v_n), by conceding a loop incident to v_1 to the vertex v_n. Similarly BiasedAddvertex() can construct at most 2 edges (v_1, v_{n-1}) . Otherwise, these vertices will be overdue. More generally, BiasedAddvertex() can construct at most q edges (v_1, v_{n-q+1}). This requirement prompts the following definitions.

The vertex v_1 is *due* if

$$|Ee_{v_1}|_{st} = 1 + 2 + ... + (n-s) + (a_s - t),$$

that is, Ee_{v_1} has got just enough loops to make each of the remaining vertices due.

The vertex v_1 is *overdue* if

$$|Ee_{v_1}|_{st} > 1 + 2 + ... + (n - s) + (a_s - t).$$

Similarly to other vertices, v_1 is *undue* if it is neither due nor overdue.

Moreover, if the degree sequence A has no entry $a_i = 1$, then the vertex v_1 is *ripe* if

$$|Ee_{v_1}|_{s,t} = \sum_{j \neq 1} |Ee_{v_j}|_{s,t}. \tag{5}$$

It is *overipe* if

$$|Ee_{v_1}|_{s,t} > \sum_{j \neq 1} |Ee_{v_j}|_{s,t}. \tag{6}$$

If the degree sequence A has an entry $a_i = 1$, then the vertex v_1 is *ripe* if

$$|Ee_{v_1}|_{s,t} = \sum_{j \neq 1} |Ee_{v_j}|_{s,t} - 1. \tag{7}$$

It is *overipe* if

$$|Ee_{v_1}|_{s,t} > \sum_{j \neq 1} |Ee_{v_j}|_{s,t} - 1. \tag{8}$$

In both cases, this means that there are more loops than all the other excess edges put together, and so, if the stage is also critical, whatever the future choices might be, the Algorithm will never produce a simple graph, as there will be at least one loop left incident to v_1. The vertex v_1 is *unripe* if it is neither ripe nor overipe. We also say that a stage is *due* (*overdue*, *undue*, *ripe*, *overipe*, *unripe*) if it contains a vertex that is due (overdue, undue, ripe, overipe, unripe). It should be understood that saying that E_{v_i} is *due* (*overdue*, *undue*, *ripe*, *overipe*, *unripe*) only means that v_i is *due* (*overdue*, *undue*, *ripe*, *overipe*, *unripe*).

The next lemma only means that once BiasedConstructKpartiteRealization() has taken a 'wrong path', it is impossible to mend the situation.

Lemma 8 *Suppose that BiasedConstructKpartiteRealization() is inserting the vertex $v_{s,y}$ and suppose that BiasedAddvertex() satisfies the following condition.*

Condition (1) For each vertex $v_{r,x}$ with $\delta(x, y) = 0$ and $r \neq 1$, BiasedAddvertex() must choose at most one edge from E_{v_r} so that there is never a double edge (v_r, v_s).

Then the following hold.

(a) If the $(s,t)^{th}$ *stage is critical, then the next stage is critical or some future stage is spoilt.*

(b) If the $(s,t)^{th}$ *stage is spoilt, then any future stage is spoilt.*

(c) If the vertex $v_{r,x}$ *is due, it is due or overdue at the next stage. If it is overdue, it is overdue at any future stage.*

(d) If the $(s,t)^{th}$ *is critical and* v_1 *is overripe, then the last stage is spoilt.*

(e) If the $(s,t)^{th}$ *stage is spoilt, then the previous stage (the stage inserting the previous edge) is either spoilt or critical.*

(f) If the $(s,t)^{th}$ *stage is overipe, then the previous stage (the stage inserting the previous edge) is either overipe or ripe.*

(g) If the $(s,t)^{th}$ *stage is overdue, then the previous stage (the stage inserting the previous edge) is either due or overdue.*

Proof. At the $(s,t)^{th}$ stage, where the t^{th} edge of the vertex $v_{s,y}$ the following types of edges are available.

(1) A loop incident to v_1 and, if conceded, the resulting edge (v_1, v_s) is single,

(2) a loop incident to v_1 and, if conceded, the resulting edge (v_1, v_s) is a multiple edge,

(3) a single edge $(v_1, v_{r,x})$ for some $r < s$ and $\delta(x,y) = 0$ or

(4) a multiple edge $(v_1, v_{r,x})$ for $r < s$ and $\delta(x,y) = 0$.

(a) In case of choice (1), both the right side and the left side of Equation 1 go down by one. Thus the next stage is still critical. In case of choice (2), the resulting edge (v_1, v_s) forms a multiple edge with a previously inserted edge. Thus, the next stage is spoilt as the left hand side stays the same while the right hand side goes down by 1. If Algorithm BiasedAddvertex() chose an edge of type (3) then the left side of Equation 1 will stay the same while the right side goes down by 1, hence the next stage would be spoilt. If BiasedAddvertex() chose an edge of type (4), then both the right side and the left side of Equation 1 will go down by one. Thus the next stage will still be critical. Thus, whatever the choice, the next stage is either critical or spoilt.

(b) Using the same arithmetic arguments as above, it is easy to see that if a stage is spoilt, the next stage is spoilt.

(c) Suppose that $j \neq 1$, the vertex v_j is due and $v_j = v_{j,x}$. If $v_s \notin N(\bar{x})_{st}$, then the next step is due. So, suppose that $v_s \in N(\bar{x})_{st}$. If BiasedAddvertex() makes the choice (3) or (4) from E_{v_r} with $r \neq j$, then since no edge of E_{v_j} is chosen, the left side of Equation 3 stays the same while the right hand side either goes down by one if BiasedConstructKpartiteRealization() moves to a new vertex

v_{s+1} or stays the same if the algorithm moves to another edge $t+1$ of the same vertex v_s. Hence the next stage is due or overdue. If BiasedAddvertex() makes the choice (4) by choosing an edge from E_{v_j}, then both sides go down by 1 and the next stage is due. Obviously, if BiasedAddvertex() makes the choice (1) or (2), v_j stays due or becomes overdue since in any case the left side of Equation 3 stays the same while the right side either goes down by one if the algorithm moves to a new vertex v_{s+1} or stays the same if the algorithm moves to a new edge of the same vertex v_s.

If v_j is overdue, it stays overdue since, for any choice, Condition (1) makes the right side of Equation 4 to go down by 1 while the left side may go down by 1 or stays the same.

A similar argument, replacing loop by edge of type (3) or (4), and vice versa, holds for the case where v_1 is due.

Suppose the vertex v_s to be inserted, is due. If BiasedAddvertex() chose a loop, the left hand goes up by 1 while the right hand side of Equation 3 stays the same. Thus v_s is overdue at the next stage. If an edge of type (3) or (4) is chosen, then both the left and the right hand sides of Equation 3 stay the same. Thus v_s is due at the next stage.

(d) By (a), the future stages will be either critical or spoilt. Suppose that there are more loops than other excess edges. If two loops are conceded to the same vertex, then the next stage is spoilt. So suppose v_1 concedes at most one loop to each of the remaining vertices. BiasedAddvertex() is then forced to pick edges from other vertices. If it picks an edge of type (3), the next stage is spoilt. So it must pick edges of type (4) only. But then edges of type (4) will be exhausted before the loops. Hence there will be at least one vertex that must conceded two loops. Thus the last stage will be spoilt.

(e) Suppose that the $(s,t)^{th}$ stage is spoilt but the previous stage (the stage inserting the previous edge) is normal. Then at the previous stage we have

$$|Ee|_{st} < a_s - (t-1) + a_{s+1} + a_{s+2} + ... + a_n. \tag{9}$$

The insertion of one edge always lowers the right hand side of Equation 9 by 1 while the left hand side is the same if BiasedAddvertex() chooses an edge of type (2) or (3) or is lowered by 1 if an edge of type (1) or (4) is chosen. Hence the $(s,t)^{th}$ stage is either critical or normal. This is a contradiction.

(f) Suppose that A has no entry $a_i = 1$ and suppose that v_1 is overipe at the $(s,t)^{th}$ stage but is unripe at the stage inserting the previous edge. That is, at that previous stage we have

$$|E_{v_1}| < \sum_j |Ee_{v_j}|. \tag{10}$$

Now, the last edge inserted is of type (1), (2), (3) or (4). If the chosen edge is of type (1) or (2) or (3), then Equation 10 is unchanged. Hence v_1 is unripe at the $(s,t)^{th}$ stage. This is a contradiction. If the chosen edge is of type (4), then the right hand side of Equation 10 goes down by 1 while the left hand side is unchanged. Hence v_1 is ripe at the $(s,t)^{th}$ stage and this is also a contradiction. The argument is similar if A has an entry $a_i = 1$

(g) Suppose $v_{j,x}$ is overdue at the $(s,t)^{th}$ stage but is undue at the stage inserting the previous edge. Then at the previous stage, we have

$$|Ee_{v_{j,x}}| < |N_{\bar{x}}|. \tag{11}$$

Now, the last edge inserted is of type (1) or (2) or (3) or (4). Moreover, in either case, BiasedConstructKpartiteRealization() moves to a new vertex or not. If it stays on the same vertex and the chosen edge of type (1) or (2) or (3), then the right and the left hand sides of Equation 11 are both unchanged. Hence v_j is undue at the $(s,t)^{th}$ stage. This is a contradiction. If the chosen edge is of type (4) from Ee_{v_j}, then the left hand side of Equation 11 goes down by 1 while the right hand side is unchanged. Hence v_j is also undue at the $(s,t)^{th}$ stage and this is also a contradiction. If the chosen edge is of type (4) from Ee_{v_i} with $i \neq j$, then both the left hand side and the right hand side of Equation 11 are unchanged. Hence v_j is also undue at the $(s,t)^{th}$ stage and this is also a contradiction.

Suppose that the algorithm moves to a new vertex. Now either $v_{s-1} \in N(\bar{x})_{s-1,t}$ or not. If $v_{s-1} \notin N(\bar{x})_{s-1,t}$, then the previous stage was overdue. This is a contradiction. So suppose that $v_{s-1} \in N(\bar{x})_{s-1,t}$. If the chosen edge is of type (1) or (2) or (3), then the right hand side of Equation 10 goes down by 1 while the right hand side is unchanged. Hence v_j is due at the $(s,t)^{th}$ stage and this is a contradiction. If the chosen edge is of type (4) from Ee_{v_j}, then both left hand and right hand sides of Equation 10 goes down by 1. Hence v_j is normal at the $(s,t)^{th}$ stage and this is a contradiction. If the chosen edge is of type (4) from Ee_{v_i} with $i \neq j$, then left hand stays the same and right hand side of Equation 10 goes down by 1. Hence v_j is normal at the $(s,t)^{th}$ stage and this is a contradiction. The same argument holds if v_1 is overdue. □

Edges of types (1) and (4) are *safe* edges while edges of types (2) and (3) are *risky* edges. Thus, the basic intuition is that our Algorithm aims at con-

structing a simple k-partite graph has to avoid risky edges as much as possible. Now, let $\hat{N}(\bar{x})_s$ be the set of vertices $v_{q,\bar{x}}$ where $q < s$ and whose second index is not x. Observe that if the degree of the $v_{s,x}$, the vertex being inserted, is 2 units greater than $|\hat{N}(\bar{x})_s| - 2$, then BiasedConstructKpartiteRealization() must take preventive measures so that the stage inserting the first edge of v_s is not critical. Otherwise, it will be impossible to insert all the edges of v_s without conceding too many risky edges of type (2). This intuition prompts the following definitions.

A vertex v_r is the *red* vertex of BiasedConstructKpartiteRealization() if v_r is the first vertex during the insertion of which BiasedConstructKpartiteRealization() can reach a critical stage. That is, if BiasedAddvertex() can choose as many risky edges of type (2) as possible, without making previously inserted vertex overdue, then the first critical stage occurs during the insertion of vertex v_r. A vertex v_f is a *fat* vertex if $a_f \geq \hat{N}(\bar{x})_f + 2$. That is, the degree of v_f is greater by at least 2 than the number of vertices v_j with $j < f$ and that can concede an edge to v_f. If this is the case, then since by Condition (1) v_f can be conceded only $\hat{N}(\bar{x})_f$ edges of types (3) or (4), BiasedAddvertex() will be forced to conceded $a_f - \hat{N}(\bar{x})_f \geq 2$ loops. Thus, the insertion of v_f is likely to lead to a spoilt stage if a critical stage is reached before inserting all the edges of v_f. The sequence of vertices $(v_r, v_{r+1}, \ldots, v_z)$ is the *red-fat sequence* of BiasedConstructKpartiteRealization() if v_r is red and v_z is fat. Now, given a red-fat sequence, BiasedConstructKpartiteRealization() has to take preventive measures so that a critical stage is not reached before the last fat vertex is inserted.

Another observation is that if BiasedConstructKpartiteRealization() is inserting vertex v_s that is fat, then as above, Condition (1) imposes that the vertex $v_{s,x}$ can be connected to at most $\hat{N}(\bar{x})_s$ vertices $v_{j,x}$ with $j < s$ and $j \neq 1$. Thus, it is easy to see that the maximal number edges of type (4), denoted by $|E4|_{st}$, which BiasedAddvertex() may concede, from the insertion of the t^{th} edge of vertex v_s until the insertion of the last edge of the last fat vertex v_z is

$$|E4|_{st} = \hat{N}(\bar{x})_s + \hat{N}(\bar{x})_{s+1} + \ldots + \hat{N}(\bar{x})_z.$$

These observations prompt the following definitions. Suppose there is a red-fat sequence $RF = (v_r, v_{r+1}, \ldots, v_z)$, $v_s \in RF$ and $E_{v_s} = a_s - \hat{N}(\bar{x})_s$. Then v_s is *fat-critical* if

$$|Ee|_{st} - |E4|_{st} - (z - s) = a_{z+1} + \ldots + a_n. \tag{12}$$

The $(s,t)^{th}$ stage is *fat-spoilt* if

$$|Ee|_{st} - |E4|_{st} - (z - s) > a_{z+1} + ... + a_n. \quad (13)$$

To make sense of Equation 12, observe that its left hand side is equal to $|Ee|_{z,a_z}$ if BiasedAddvertex() chose the maximal possible number of edges of type (4) (and, conversely, the minimal number of loops). Indeed, an edge $e \in Ee_{z,a_z}$ only if $e \in Ee_{s,t}$. Now, starting from $e \in Ee_{s,t}$, to get the number of edges that are still in the set of excess edges at the $(z, a_z)^{th}$ stage, one has to remove the edges of type (4) which are conceded and thus become unavailable, and there are at most $|E4|_{st}$ of them. Moreover, since $a_s \geq a_{s+1} \geq \cdots \geq a_z$, all these vertices are fat. Thus to each such vertex $v_{j,x}$, v_1 must concede $a_j - \hat{N}(\bar{x})_j$ loops. But, if p loops are conceded to v_j, then $p - 1$ excess-edges (v_1, v_j) are constructed. Hence for every vertex v_j inserted between v_s and v_z, there is one excess edge lost. Hence the term $z - s$ has to be subtracted. Hence the left hand side of Equation 12 is indeed equal to $|Ee|_{z,a_z}$ if the maximal possible number of edges of type (4) is conceded. Using this fact, it is easy to observe that Equation 12 means that the $(s,t)^{th}$ stage is not critical, but if Algorithm BiasedAddvertex() chooses the maximal number of edges of type (4), then the first critical stage would occur after inserting the last edge of the last fat vertex v_z.

Lemma 9 *(i) If the* $(s,t)^{th}$ *stage is fat-critical, then the next stage is fat-critical or fat-spoilt.*

(ii) If the $(s,t)^{th}$ *stage is fat-spoilt, then some future stage is spoilt.*

Proof. (i) First, observe that whatever the choice of edge, the right hand side of Equation 12 stays the same. If BiasedAddvertex() chose an edge of type (4), then on the left hand side, $|Ee|_{st}$ will go down by 1 and $|E4|_{st}$ will also go down by 1, while $z - s$ will stay the same. Hence the next stage is fat-critical. Suppose a loop is chosen, then $|Ee|_{st}$ will stay the same as one loop is lost but an edge of type (4) is created, $|E4|_{st}$ will go down by 1 as v_s can not be connected to the maximal number of edges of type (4) while $z - s$ will stay the same. Hence the next stage is also fat-spoilt.

(ii) The left hand side of Equation 13 is equal to $|Ee|_{z,a_z}$. Hence, if the $(st)^{th}$ stage is fat-spoilt, the $(z, a_z)^{th}$ stage is spoilt. □

While Lemmas 8 and 9 say that once the random walk on $\mathcal{T}$ takes a wrong path, it is impossible to mend it. The next routine gives the preventive measure to avoid getting into that wrong path in the first place. If the algorithm is inserting the t^{th} edge of vertex $v_{s,y}$, then an edge e is available when e is a loop (incident on v_1) or $e \in E_{v_{r,x}}$ with $r < s$ and $\delta(x, y) = 0$.

Routine ChooseCorrectEdge()[Routine choosing edges that lead to a simple graph]

Suppose that BiasedConstructKpartiteRealization() is at its $(s,t)^{th}$ stage. That is, it is inserting the t^{th} edge of vertex $v_{s,y}$. Then

(1) For each vertex $v_{r,x}$ with $r < s$ and $\delta(x,y) = 0$, choose at most one edge from E_{v_r}.

(2) If the stage is normal but not due, choose any available edge uniformly at random.

(3) If the stage is critical but not due nor ripe, choose any available edge of type (1) or (4) uniformly at random.

(4) For $j > s$, if the vertex v_j is due, pick an edge from E_{v_j}. If many such vertices are due, pick an edge uniformly at random from the vertices that are due.

(5) If the stage is critical and ripe, pick a loop.

(6) If v_s, the vertex being inserted, is due, then pick any available edge of type (3) or (4) uniformly at random.

(7) If the stage is fat-critical, then pick any available edge of type (4) uniformly at random.

We illustrate the Routine ChooseCorrectEdge() in Figure 4. Before proving that this algorithm is necessary and sufficient to sample a simple k-partite graph at random, we observe that it runs in $\mathcal{O}(n^2)$ steps, where n is the number of vertices in any realizations of A. Indeed, BiasedConstructKpartiteRealization() calls BiasedAddVertex() n times and BiasedAddVertex() calls ChooseCorrectEdge() a_i times to insert all the edges of vertex v_i. At the i^{th} iteration of BiasedConstructKpartiteRealization(), ChooseCorrectEdge() has to check Equations 1, 7 and 12 once each. Moreover, it has to check Equation 3 for at most $i-1$ vertices. Hence throughout the running of BiasedConstructKpartiteRealization(), ChooseCorrectEdge() has to perform at most $3n + \frac{(n-1)(n-2)}{2}$ checks.

Theorem 10 *Algorithm BiasedConstructKpartiteRealization() reconstructs a simple* k*-partite graph if and only if BiasedAddVertex() calls the routine ChooseCorrectEdge(). In other words, BiasedConstructKpartiteRealization() outputs a simple* k*-partite graph if and only if the choice of edges satisfies conditions (1)–(7).*

The following lemma is required in the proof of Theorem 10.

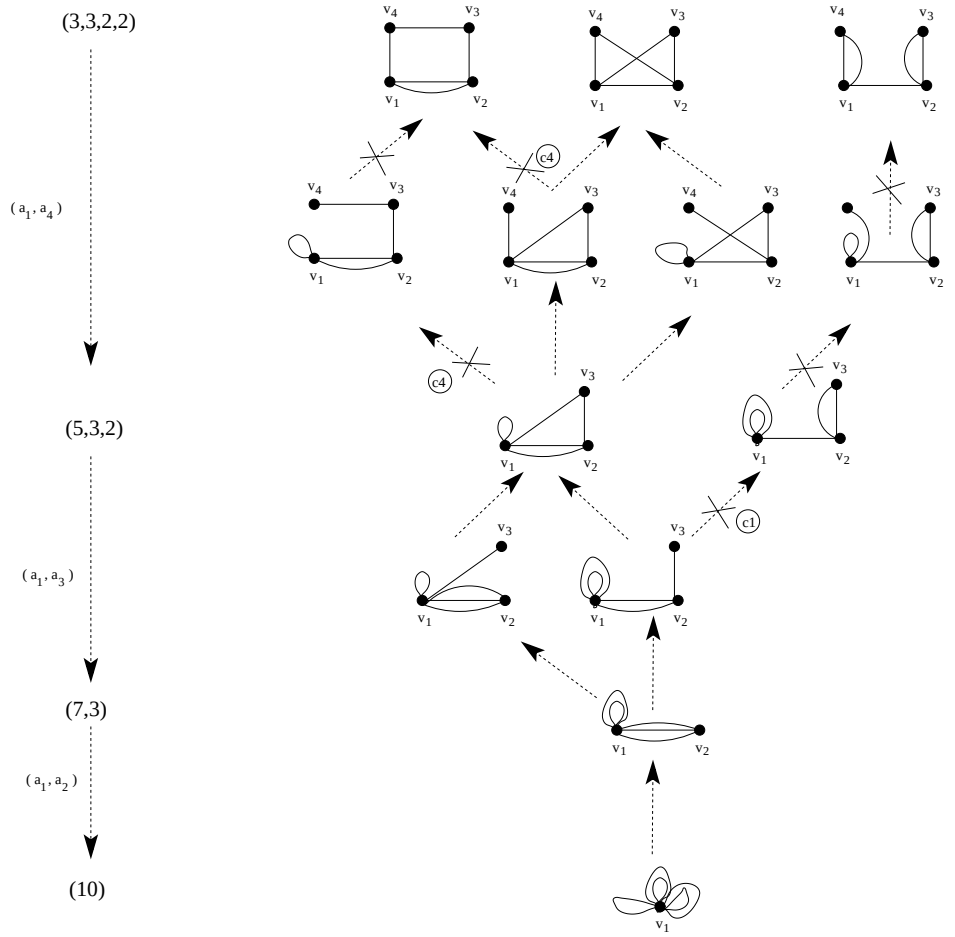

Figure 4: Random reconstruction tree of $(3,3,2,2)$ for simple graph. It is similar to that in Figure 3, but some choice of edges are forbidden. Forbidden moves are marked by a cross and the condition that they fail to satisfy. For example, c1 means condition 1 of the Routine ChooseCorrectEdge()

Lemma 11 *If* A *is a degree sequence of a simple* k*-partite graph having* n *vertices, then the* $(2,1)^{the}$ *stage of Algorithm BiasedConstructKpartiteRealization() is neither critical nor spoilt nor overdue nor overipe.*

Proof. Before the insertion of vertex v_2, there are m loops incident to v_1 where $m = \frac{\sum_{i=1}^{n} a_i}{2}$. That is,

$$|Ee|_{2,1} = \frac{\sum_{i=1}^{n} a_i}{2}. \tag{14}$$

To prove the result, we only need to show that $|Ee|_{2,1} \leq \sum_{i=2}^{n} a_i$. That is, the number of excess edges is not greater than the number of edges left to insert.

But by Erdos-Gallai criterion, we have

$$a_1 \leq \sum_{i=2}^{n} a_i. \tag{15}$$

Rearranging Equation 14 and plugging Equation 15 into it, we get that

$$2|Ee|_{2,1} = \sum_{i=1}^{n} a_i \leq 2(\sum_{i=2}^{n} a_i). \tag{16}$$

Hence the first stage is not spoilt and vacuously it is not overdue. □

Proof.[Proof of Theorem 10] Suppose for a contradiction that conditions (1)–(7) hold at all the stages but BiasedConstructKpartiteRealization() outputs a k-partite graph G with multiple edges or loops. By condition (1) there can not be a multiple edge connecting two vertices $v_{r,x}$ and $v_{s,y}$ with $r < s$. Moreover, the algorithm would prevent any edge $v_{r,x}$ and $v_{s,x}$ with $r < s$ unless $r = 1$. Hence if G fails to be a simple graph, it must have either a loop or a multiple edge incident to v_1.

So, suppose that in G the vertex v_1 is incident to either a loop e or a multiple edge. Thus, the stage inserting the last edge of the last vertex v_n is either spoilt, or critical and overipe or overdue.

If the last stage is spoilt (overdue), then by Lemma 8 (e, f, g), the previous stage was either spoilt (overdue) or critical (due). If it were spoilt (overdue), then the one prior to it was spoilt (overdue) or critical (due), and so on. Thus by induction, the first stage of BiasedConstructKpartiteRealization() was either spoilt (overdue) or critical (due). This contradicts Lemma 11. Hence some stage later than the $(3,1)^{th}$ must have been critical (due). Thus BiasedConstructKpartiteRealization() must have gone through a series of normal (undue) stages, then a series of critical (due), then a (possible) series of spoilt (overdue) stages prior to the stage inserting the last edge.

So, let the first spoilt (overdue) stage be the $(q,p)^{th}$ stage. So the stage preceding it was critical (due). But, by condition (4), (condition (5)) Algorithm BiasedAddvertex() must have chosen a safe edge so that the $(q,p)^{th}$ stage should be critical (due) by Lemma 8. This is a contradiction.

Suppose that the last stage is critical and overipe. Then using an argument similar to the case where the last stage is spoilt, we also get a contradiction.

Conversely, suppose that some condition (1)–(7) is not satisfied at the $(s,t)^{th}$ stage and let G be the realization output at the end of BiasedConstructKpartiteRealization(). If condition (1) is not satisfied at the $(s,t)^{th}$ stage, then this would create a double edge (v_j, v_s) with $j, s \neq 1$. In that case, the edge stops to be available, and since Algorithm BiasedAddvertex() can not concede it anymore, the double edge would appear in G. Hence G would not be simple.

If condition (3) is not satisfied at the $(s,t)^{th}$, then by Lemma 8(a) and (b) any future stage is spoilt. Hence G is not a simple graph. Suppose that condition (4) is not satisfied. That is, there is a vertex v_j that is due at the $(s,t)^{th}$ stage but Algorithm BiasedAddvertex() does not pick any of the elements of Ee_{v_j} for all the remaining edges conceded to v_s. Then v_j is overdue at the insertion of vertex v_{s+1}, and by Lemma 8(c) it remains overdue until the end of the Algorihm. Hence G is not simple.

Suppose that condition (5) is not satisfied at the $(s,t)^{th}$ stage that is critical. That is, Ee_{v_1}, the set of loops incident to v_1 is ripe but Algorithm BiasedAddvertex() does not pick a loop. Then, by Lemma 8(d), any future stage is spoilt. Hence G is not simple.

Suppose that condition (6) is not satisfied at the $(s,t)^{th}$ stage, that is, the vertex v_s is due and Algorithm BiasedAddvertex() picks a loop. Then v_s will become overdue and G will exhibit a multiple edge (v_1, v_s). If condition (7) is not satisfied, then by Lemma 9 (2) Algorithm ConstructRealization() reaches a spoilt stage. □

Let a *correct edge* be an edge chosen by Algorithm ChooseCorrectEdge. So, if BiasedConstructKpartiteRealization() terminates, we have shown that it always outputs a simple graph. It remains to show that it always terminates by showing that there is always a correct edge so that conditions (4)–(7) can be satisfied.

Theorem 12 *Algorithm ChooseCorrectEdge() always terminates. That is, conditions (4)–(7) can always be satisfied.*

The proof that is quite involved is left to the end of the paper. Still, we have to show that every simple realization can be reached. The next result is instrumental in showing that every simple realization of A can be reconstructed by BiasedConstructKpartiteRealization() if conditions (1)–(7) are satisfied.

Lemma 13 *Let* $G = G_{n_1,n_2,\ldots,n_k}$ *be the* $n_1, n_2, \ldots, n_k$*- complete* k*-partite graph. That is, the* k*-partite graph where the* i^{th} *part contains* n_i *vertices each having degree* $\sum_{j\neq i} n_j$*. Then BiasedConstructKpartiteRealization() satisfying conditions (1)–(7) can reconstruct* G *as a realization of* $A = (A_1 : A_2 : \ldots : A_k)$*, where* A_i *has* n_i *entries equal to* $\sum_{j\neq i} n_j$*.*

Proof. At the second iteration, the algorithm inserts the vertex $v_{2,x}$ by conceding $\sum_{j\neq i} n_j$. It is routine to check that $v_{2,x}$ is due. Now suppose that $v_3 = v_{3,y}$. If $\delta(x, y) = 1$, then v_1 will concede $\sum_{j\neq i} n_j$ loops to v_3. If $\delta(x, y) = 0$, then by Condition (4), v_2 will concede one edge to v_3 and v_1 will have to concede $\sum_{j\neq i} n_j - 1$ loops to v_3. In both cases v_3 is due and by Lemma 8 v_2 is also due. Now, for an induction, suppose that the algorithm is inserting the vertex $v_{s,z}$ and all the preceding vertices are due. Recall that $\hat{N}(\bar{z})_i$ is the set of vertices inserted before $v_{i,z}$ and whose second index in not z. Then v_i will be conceded $|\hat{N}(\bar{z})_i|$ edges from vertices in $\hat{N}(\bar{z})_i$ and $\sum_{j\neq z} n_j - |\hat{N}(\bar{z})_i|$ loops from v_1. Hence v_i will also be due and all the vertices preceding it will be due by Lemma 8. Now it is routine to check that at the n^{th} every vertex is incident to a single available edge. Thus, each of them will concede it and Algorithm BiasedConstructBipartiteRealization() outputs the graph G. □

Let G be a graph, a *delete-minor* of $G' = G\backslash e$ is the graph obtained from G by deleting the edge e. If $A = (A_1 : A_2 : \ldots : A_k)$ is a k-partite degree sequence, let A' be the degree sequence obtained from A by subtracting 1 from two of its entries a_r and a_s, where $a_r \in A_i$ and $a_r \in A_j$ with $i \neq j$. Thus, if A is the degree sequence of a k-partite graph G, then A' is the degree sequence of some delete-minor of G.

Lemma 14 *If BiasedConstructKpartiteRealization() satisfying conditions (1)–(7) can reconstruct* G *as a realization of* A*, then it can reconstruct all the delete-minors of* G *that are realizations of* A'*.*

Proof. Let G be a k-partite graph output by Algorithm BiasedConstructKpartiteRealization() and let $G\backslash e$ be a delete-minor of G. Suppose in the graph G, the edge e is incident to vertices $v_{r,x}$ and $v_{s,y}$ having respectively degrees a_r and a_s, and where $r < s$ and $\delta(x, y) = 0$. Thus in $G\backslash e$, vertices v_r and v_s have degrees $a_r - 1$ and $a_s - 1$. Let f be any edge of $G\backslash e$. Now, since G is output by BiasedConstructKpartiteRealization(), then there is a series of choices of correct edges such that f can be inserted. Now, in that series of choices either e is inserted before f or after. If e is inserted after f, then the same series of choices would insert f in $G\backslash e$. If e is inserted before f, then the same series of

choices, minus the insertion of e, will also lead to the insertion of f in $G\backslash e$, since Algorithm BiasedConstructKpartiteRealization() does not need to insert any edge incident to v_r and v_s as their degrees are down by 1. □

Corollary 15 *Let* G *be a simple* k*-partite realization of a degree sequence* $A = (A_1 : A_2 : ... : A_k)$*, where* A_i *has* n_i *entries. Then there is a positive probability that* G *is output by Algorithm BiasedConstructKpartiteRealization() if conditions (1)–(2) are satisfied.*

Proof. Every simple k-partite graph whose j^{th} part contains n_j vertices can be obtained from $G_{n_1,n_2,...,n_k}$ by a series of deletions. □

3.1 Sampling simple realizations uniformly

The calling of BiasedAddVertex() by BiasedConstructKpartiteRealization() allows to sample all k-partite realizations of A with equal probability. But to construct simple k-partite realizations only, the choice of edges is dictated by the routine ChooseCorrectEdge(). We recall that correct edges are those edges chosen by the routine ChooseCorrectEdge(). It is easy to check that the number of correct edges is not constant across all the graphs on the same level of $\mathcal{T}$. This remark prompts to modify the routine BiasedAddVertex() as follows. A vertex $v_{r,x}$, where $r < s$ and $\delta(x, y) = 0$, is said to be *correctly-adjacent* to v_1 at the $(s, t)^{th}$ stage if v_r is connected to v_1 by a correct edge at that stage.

SimpleBiasedAddvertex() (A modification of Algorithm Addvertex() to get a uniform distribution on the set of simple k-partite realizations, dividing by the number of vertices inserted up to the s^{th} iteration)
Step 1 To the graph G^l add an isolated vertex called v_{l+1}. Let a_{l+1} be the number of edges to concede to v_{l+1} and let $t = 0$.
Step 2 If v_{l+1} is incident to a_{l+1} edges, return G^{l+1}. Else,
Step 3 Let b^t_{l+1} be the number of different vertices incident to v_{l+1} after the insertion of its t^{th} edge, with $t \geq 1$. If $t = 0$, let b^t_{l+1} be the number of different vertices adjacent to v_l.
Step 4 Choose vertex v_r uniformly at random amongst all the vertices that are correctly-adjacent to v_1.
Step 5 If $e \in E_{v_r}$, then concede e to v_{l+1} with probability $\frac{|V'(G^s)|}{|V(G^s)|.b^t_{l+1}}$, where $V(G^s)$ and $V'(G^s)$ are respectively the sets of vertices inserted up to the s^{th}

iteration and the set of vertices in $V(G^s)$ that are correctly-adjacent to v_1. Increment t by 1 and go back to Step 2.

Theorem 16 *For the degree sequence* $A = (A_1 : A_2 : ... : A_k)$*, where* A_i *has* n_i *entries such that* $n_1 + n_2 + ... + n_k = n$*, BiasedConstructKpartiteRealization() reaches every simple* k*-partite realization of* A *uniformly at random with probability.*

Proof. Corrolary 15 shows that all simple realizations can be reached by the BiasedConstructKpartiteRealization() if the choice of edges is dictated by the routine ChooseCorrectEdge(). The proof for uniformity is similar to that of Theorem 7.

□

We now give the overall Algorithm and its running time.

Algorithm UniformGenerateSimpleKpartiteRealization()
Input: k-partite degree sequence $A = (A_1 : A_2 : ... : A_k)$ where A_i has n_i entries.
Output: A random k-partite realization of A.
Step 1 Put A in non increasing order.
Step 2 Construct the recursion chain of A by calling the routine ConstructKpartiteRecursionChain().
Step 3 Construct a random k-partite realization of A by calling BiasedConstructSimpleKpartiteRealization().

Now, it is known that Step 1 takes $\log(n)$ iterations and as shown earlier Step 2 takes n iterations. In Step 3, ChooseCorrectEdge() does $\frac{n(n-1)}{2}$ checks altogether. Finally, BiasedAddVertex() needs $2m$ iterations to insert all the vertices. Thus, the overall running time is at most

$$\log(n) + \frac{n(n-1)}{2} + 2m \leq n^2 + 2m.$$

3.2 Proof that ChooseCorrectEdge() always terminates successfully

Recall that an edge is *correct* at the $(s,t)^{th}$ stage if ChooseCorrectEdge() may choose it at the $(s,t)^{th}$ stage. We have shown that if BiasedConstructKpartiteRealization() terminates, it always outputs a simple graph. It remains to

show that it always terminates by showing that there is always a correct edge so that conditions (4)–(7) can be satisfied.

Proof.[Proof of Theorem 12]

The proof is by induction on the stage where BiasedConstructKpartiteRealization() is and consists of many lemmas, each dealing with one of the conditions. Obviously all the conditions are satisfied at the $(2,1)^{th}$ stage. Suppose they hold up to the $(s,t-1)^{th}$ stage and let BiasedConstructKpartiteRealization() be at its $(s,t)^{th}$ stage, where the vertex $v_{s,y}$ is being inserted. The next lemma shows that condition (4) is always met. Recall that a safe edge is an edge of type (1) or type (4). □

Lemma 17 *Let the Algorithm BiasedConstructKpartiteRealization() satisfy conditions (1)–(7) and the* $(s,t)^{th}$ *stage is unripe, undue and critical. Then it is always possible to concede a safe edge.*

Proof. Suppose the $(s,t)^{th}$ is critical, but there is no edge of type (1) or type (4). That is, $v_{s,y}$ is already adjacent to v_1 and all vertices $v_{r,x}$, such that $r < s$ and $\delta(x,y) = 0$, have no correct edge.

(a) Suppose v_1 and all the vertices $v_{r,x}$ such that $r < s$ and $\delta(x,y) = 0$ are connected to v_s. Then there is no safe edge if $a_s > \hat{N}(\bar{x})_s$. Thus the vertex v_s is fat. But by condition (7), BiasedConstructKpartiteRealization() can not reach a critical stage before inserting all the edges of v_s. This is a contradiction.

(b) Now suppose there is one vertex, $v_{q,x}$ with $q < s$ and $\delta(x,y) = 0$, that is not connected to $v_{s,y}$. If $|E_{v_q}| > 1$, then there is a correct edge in E_{v_q}. This is a contradiction. So let $|E_{v_q}| \leq 1$. We need the following fact.

Fact 18 *Let* $A = (a_1, a_2, \ldots, a_s, \ldots a_n)$ *be a degree sequence, let* G *be a simple realization of* A *and let* $\hat{G}$ *be the simple graph obtained from* G *by deleting the vertex* v_q *and all edges incident to* v_q*. Then* $\hat{G}$ *is a simple realization of the degree sequence* $\hat{A}$ *obtained from* A *by removing the entry* a_q *and subtracting 1 to all entries* a_i *such that* v_i *is adjacent to* v_q *in* G*. Hence* $\hat{A}$ *is the degree sequence of a simple graph. We call* $\hat{A}$ *a q-reduction of* A*.*

Now, for the degree sequence A, let $P_A = (e_1, e_2, \ldots, e_r)$ be the sequence of correct edge choices leading the critical stage (s,t), where some preceding vertices are not connected to v_s and let v_q be such a vertex. Then $P_{\hat{A}}$ obtained from P_A by removing all the edges incident to v_q is obviously a sequence of correct edge choices for $\hat{A}$ and leading to a critical stage. They are correct, since if e_i is of type (4) or (1) in P_A, it is still of type (4) or (1) in $P_{\hat{A}}$ as

removing edges incident to v_q will leave at least one other edge parallel to it. They lead to a critical stage, since removing edges incident to v_q does not affect the number of excess edges, as v_q is connected to v_1 by 1 edge at most. But according to part (a) the sequence $P_{\hat{A}}$ leads to a critical stage satisfying condition (4). That is, there is a edge e that is correct. Hence the same edge e is correct for A. Thus, we have proved that condition (4) is satisfied in case (b).

□

The next lemma shows that condition (5) is always met.

Lemma 19 *Suppose that BiasedConstructKpartiteRealization() satisfies conditions (1)–(7) and at the* $(s,t)^{th}$ *stage, the vertex* $v_{r,x}$ *is due. Then it is always possible to concede an edge from* E_{v_r}.

Proof. Let a_r denote the degree of the vertex v_r. The proof uses induction on s, the number of vertices already inserted. For a contradiction, we suppose that v_r is due but SimpleBiasedAddvertex() can not pick an edge from E_{v_r}. This is possible only if there are too many vertices that are due. For $s = 1$, this is vacuously not possible. Now suppose for all the stages up to the $(s-1)^{th}$ stage, the result holds, so that none of these stages is spoilt or overdue. In particular, at the insertion of vertex $v_{q,y}$, where q is the greatest index less than s whose second index is y, all the vertices that were due conceded one edge. Suppose at the insertion of vertex v_s, the number of sets that are due is greater than a_s.

(i) Let $\nu_s = |\hat{N}(\bar{y})|$ be the number of vertices inserted before $v_{s,y}$ and whose second index is not y. Suppose v_1 is not due and that there are $\nu_s - p$ (for $p \geq 2$) vertices v_j, with $j \neq 1$, that are due at the insertion of vertex v_q. By hypothesis, all the $\nu_s - p$ concede an edge to vertex v_q and by Lemma 8(c) all these $\nu_s - p$ vertices are due at the insertion of vertex v_s. Hence, by hypothesis, $a_s < \nu_s - p$.

If the $(s,t)^{th}$ stage is normal or critical then we have

$$|Ee|_{st} \leq a_s - t + a_{s+1} + \ldots + a_n. \tag{17}$$

But, at $t = 1$, that is, at the insertion of the first edge of vertex v_s, there are at least $\nu_s - p$ vertices v_j that are due. Thus, on one hand, we have

$$a_s - t + a_{s+1} + \ldots + a_n < (\nu_s - p)(n - s + 1). \tag{18}$$

Equation 18 follows from the fact $a_s < \nu_s - p$, as we assume that there are too many due vertices, and $a_u \leq a_s$ for $u > s$, and there are $n - s + 1$ vertices left to insert.

On the other hand, recalling that E_{ν_1} denotes the set of loops incident to ν_1, we have

$$\begin{aligned} |Ee|_{st} &= |E_{\nu_1}| + |Ee_{\nu_2}| + \cdots + |Ee_{\nu_{s-1}}| \\ &= |E_{\nu_1}| + |Ee|_{due} + |Ee|_{undue} \\ &= |E_{\nu_1}| + |Ee|_{undue} + (\nu_s - p)(n - s + 1), \end{aligned}$$

where $|Ee|_{due}$ and $|Ee|_{undue}$ denote the set of excess-edges from vertices that are due and from vertices that are not due respectively, and $|Ee|_{due} = (\nu_s - p)(n - s + 1)$, since there are at least $\nu_s - p$ vertices that are due and each has $n - s + 1$ excess-edges as there are $n - s + 1$ vertices waiting to be inserted.

Hence, if the stage is normal or critical, we have

$$|E_{\nu_1}| + |Ee|_{undue} + (\nu_s - p)(n - s + 1) \leq a_s - t + a_{s+1} + \ldots + a_n < (\nu_s - p)(n - s + 1). \tag{19}$$

This is impossible. If the stage is spoilt, then SimpleBiasedAddvertex() chose an edge of type (2) or (3) at the $(s-1)^{th}$ stage. This contradicts the inductive hypothesis.

(ii) Suppose ν_1 is due at the $(s, t)^{th}$ stage, so that there are too many vertices that are due and one of them is ν_1. Thus, we have

$$|E_{\nu_1}| + |Ee|_{notdue} + (\nu_s - p)(n - s + 1) \leq a_s - t + a_{s+1} + \ldots + a_n \leq (\nu_s - p)(n - s + 1). \tag{20}$$

Here we have $\leq$ as $a_s = (\nu_s - p)$, where $\nu_s - p$ is the number of due vertices other than ν_1. But this is possible only if $|E_{\nu_1}| = 0$ and this is a contradiction. □

The next lemma shows that Condition (6) is always met.

Lemma 20 *Suppose that Algorithm BiasedConstructRealization() satisfies conditions (1)–(7) and the* $(s, t)^{th}$ *stage is critical and ripe. Then it is always possible to concede a loop from* E_{ν_1}.

Proof. The proof uses induction on i, the number of vertices already inserted. The lemma holds vacuously for $i = 1$. Suppose now it holds for all $i < s$. At the insertion of the s^{th} vertex, it may not hold only if there are also at least $\nu_s - p > a_s$ other vertices that are due. Suppose the stage is normal. In that case, as in the proof of Lemma 19, we have

$$|E_{\nu_1}| + |Ee|_{notdue} + (\nu_s - p)(n-k+1) \leq a_s - t + a_{s+1} + \ldots + a_n \leq (\nu_s - p)(n-s+1). \tag{21}$$

But this is possible only if $|E_{\nu_1}| = 0$ and this is a contradiction. Similarly, there is a contradiction if the stage is critical as $|E_{\nu_1}|$ must also be null. Moreover the inductive hypothesis is contradicted if the stage is spoilt.

□

The next lemma shows that condition (7) is always met.

Lemma 21 *Suppose that Algorithm BiasedConstructRealization() satisfies conditions (1)–(7) and at its* $(s,t)^{th}$ *stage* ν_s*, the vertex being inserted, is due. Then there is an available edge that is not a loop.*

Proof. Suppose after the insertion of the $(\tau_1 + \tau_2)^{th}$ edge of vertex ν_s, (where τ_2 edges are of type (3) or (4)) the vertex ν_s is due, but $\tau_1 + \tau_2 < a_s$. That is, $|Ee_{\nu_s}| = |N(\bar{x})|_s$ but ν_s is not completely inserted. If for some $r \neq 1$ and $r < s$ the vertex $\nu_{r,x}$, with $\delta(x,y) = 0$, is not adjacent to ν_s and $E_{\nu_r} \neq \phi$, then SimpleBiasedAddvertex() picks an edge from E_{ν_r}.

If not, suppose all the vertices $\nu_{r,x}$, with $r < s$ and $\delta(x,y) = 0$ and that are not adjacent to ν_s, we have $E_{\nu_j} = \phi$. Now either (i) the stage is critical or (ii), it is not.

If (i) and all the available edges are loops, then the stage is critical and overipe and this contradicts the inductive hypothesis. Thus, some are loops and some are multiple edges incident to $\nu_{r,x}$ with $r \not\equiv 1$, $\delta(x,y) = 0$, and ν_r is adjacent to ν_s. But, then there must be a vertex ν_j not incident to ν_s with $a_j < a_s$. Let there be p vertices adjacent to ν_s and q vertices ν_j not adjacent to ν_s such that $E_{\nu_j} = \phi$. Then, as assumed above, we have $t + p < a_s$ as ν_s is not fully inserted. But ν_j must be adjacent to some of the vertices ν_i that are adjacent to ν_s. Thus $a_j \leq p < a_s$. This contradicts the fact that $a_1 \geq a_2 \geq \ldots \geq a_n$.

Now, let all previously inserted vertices ν_j be connected to ν_s. It is easy to see that there must be $s - 2$ such vertices which are not ν_1. Thus $a_s = n - s + 1 + (s - 2) = n - 1$. But, by Erdös-Gallai criterion, we also have

$a_s \leq n-1$. Hence the vertex v_s is already completely inserted . This is a contradiction. □

The next lemma shows that condition (7) is always met.

Lemma 22 *Suppose that Algorithm BiasedConstructRealization() satisfies conditions (1)–(7) and that the* $(s,t)^{th}$ *is fat-critical. Then there is an available edge of type (4).*

Proof. Suppose, for a contradiction, the $(s,t)^{th}$ is fat-critical, but there is no edge of type (4). That is, all available edges are loops or type (3). That is, for all vertices $v_{r,x}$ such that $r < s$ and $\delta(x,y) = 0$ we have that $a_{r,x} < |\widehat{N}(\bar{x})|_r + |N(\bar{x})|_r$. But we know, for all r with $r < s$, $a_s \leq a_r$ and $a_{s,y} > |\widehat{N}(\bar{y})|_s$ as v_s is fat. This is a contradiction. □

Thus, we have proved that conditions (4)–(7) are always satisfied at all stages of the running of Algorithm BiasedConstructRealization(). Hence it always terminates reaching a leaf of $\mathcal{T}$ that is a simple graph.

References

[1] M. Bayati, J. H. Kim and A. Saberi, A sequential algorithm for generating random graphs, *Algorithmica* **58** (2010) 860–910. ⇒184, 185, 186

[2] E. A. Bender and E. R. Canfield, The assymptotic number of labelled graphs with given degree sequence, *J. Combin. Theory, Ser A.* **24**, 3 (1978) 296–307. ⇒184, 185

[3] J. Blitzstein and P. Diaconis, A sequential importance sampling algorithm for generating random graphs with prescribed degree sequence, *Internet Math.* **6**, 4 (2011) 489–522. ⇒184, 185, 186

[4] B. Bollobas, A probalistic proof of an assymptotic formula for the number of labelled regular graphs, *European J. Combin.* **1**, 4 (1980) 311–316. ⇒184, 185

[5] R. A. Brualdi, Matrices of zeroes and ones with fixed row and column sum vectors, *Linear Algebra Appl.* **33** (1980) 159–231. ⇒184

[6] T. Brylawsky and J. Oxley, *The Tutte polynomial and its applications*, in N. White, ed., Matroid Applications, Encyclopedia of Mathematics and its Applications, Cambridge University Press, (1992) 123–225. ⇒185

[7] Y. Chen, P. Diaconis, S. Holmes and J. S. Liu, Sequential Monte Carlo methods for statistical analysis of tables, *J. Am. Stat. Assoc.* **100** (2005) 109–120. ⇒184

[8] G. W. Cobb and Y. Chen, An application of Markov Chains Monte Carlo to community ecology, *Amer. Math. Month.* **110** (2003) 265–288. ⇒184

[9] C. Cooper, M. Dyer and C. Greenhill, Sampling regular graphs and Peer-to-Peer network, *Combinatorics, Probability and Computing* **16** (2007) 557–594. ⇒184

[10] P. Diaconis and A. Gangolli, Rectangular arrays with fixed margins, In Discrete Probability and Algorithms (Minneapolis, MN, 1993), *IMA Vol. Math. Appl.* **72** 15–41 , New York Springer, 1995. ⇒184

[11] P. Diaconnis and B. Sturmfels, Algebraic algorithms for sampling from conditional distributions, *Ann Statist.* **26**, 11 (1998) 363–397. ⇒184

[12] P. Erdös and T. G. Gallai, Graphs with prescribed degrees of vertices (Hungarian), *Mat. Lapok* **11** (1960) 264–274. ⇒184, 185

[13] M. Jerrum and A. Sinclair, Approximating the permanent, *SIAM J. Comput.* **18**, 6 (1989) 1149–1178. ⇒184

[14] R. Kannan, P. Tetali and S. Vempala, Simple Markov-chain algorithms for generating bipartite graphs and tournaments, *Random Struct. Algorithms* **14**, 4 (1999) 293–308. ⇒184, 185

[15] K. K. Kayibi, M. A. Khan and S. Pirzada, Rejection sampling of bipartite graphs with given degree sequence, *Acta Univ. Sapientia, Mathematica* **10,** 2 (2018) 249–275. ⇒183, 184

[16] M. Luby, D. Randall and A. Sinclair, Markov chain algorithms for planar lattice structures, *SIAM J. Comput.* **31**, 1 (2001) 167–192. ⇒184

[17] I. Miklós, P. L. Erdös and L. Soukup, Towards random uniform sampling of bipartite graphs with given degree sequence, Preprint. ⇒184, 185

[18] M. E. J. Newman, A. L. Barabasi and D. J. Watts, *The structure and dynamics of networks* (Princeton Studies in Complexity, Princeton UP) (2006) pp 624. ⇒ 184

[19] S. Pirzada, *An Introduction to Graph Theory*, Universities Press, Hyderabad, India, 2012. ⇒185

[20] V. V. Vazirani, *Approximation algorithms*, Springer-Verlag, Berlin, Heildelberg, New York, 2003. ⇒184

[21] N. Wormald, Models of random regular graphs, In Surveys in Combonatorics (Canterbury), Cambridge University Press, *London, Math. Soc. Lecture Note Ser.* **267** (1999) 239–298. ⇒184, 185

Received: October 22, 2018 • Revised: November 21, 2018

Acta Univ. Sapientiae, Informatica **10,** 2 (2018) 218–240

DOI: 10.2478/ausi-2018-0011

Minimum covering reciprocal distance signless Laplacian energy of graphs

Abdollah Alhevaz
Faculty of Mathematical Sciences, Shahrood University of Technology, P.O. Box: 316-3619995161, Shahrood, Iran
email: a.alhevaz@gmail.com

Maryam Baghipur
Faculty of Mathematical Sciences, Shahrood University of Technology, P.O. Box: 316-3619995161, Shahrood, Iran
email: maryamb8989@gmail.com

Ebrahim Hashemi
Faculty of Mathematical Sciences, Shahrood University of Technology, P.O. Box: 316-3619995161, Shahrood, Iran
email: eb_hashemi@shahroodut.ac.ir

Yaser Alizadeh
Department of Mathematics, Hakim Sabzevari University, Sabzevar, Iran
email: y.alizadeh@hsu.ac.ir

Abstract. Let G be a simple connected graph. The reciprocal transmission $\mathrm{Tr}'_G(v)$ of a vertex v is defined as

$$\mathrm{Tr}'_G(v) = \sum_{u \in V(G)} \frac{1}{d_G(u,v)}, \quad u \neq v.$$

The reciprocal distance signless Laplacian (briefly RDSL) matrix of a connected graph G is defined as $RQ(G) = \mathrm{diag}(\mathrm{Tr}'(G)) + RD(G)$, where $RD(G)$ is the Harary matrix (reciprocal distance matrix) of G and $\mathrm{diag}(\mathrm{Tr}'(G))$ is the diagonal matrix of the vertex reciprocal transmissions in G. In this paper, we investigate the RDSL spectrum of some classes of graphs that are arisen from graph operations such as cartesian product, extended double cover product and InduBala product. We

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C12, 05C05, 05C35.
Key words and phrases: Reciprocal distance signless Laplacian matrix; reciprocal distance signless Laplacian energy; minimum covering reciprocal distance signless Laplacian energy.

introduce minimum covering reciprocal distance signless Laplacian matrix (or briefly MCRDSL matrix) of G as the square matrix of order n, $RQ_C(G) := (q_{i,j})$,

$$q_{i,j} = \begin{cases} 1 + Tr'(v_i) & \text{if } i = j \text{ and } v_i \in C \\ Tr'(v_i) & \text{if } i = j \text{ and } v_i \notin C \\ \dfrac{1}{d(v_i, v_j)} & \text{otherwise,} \end{cases}$$

where C is a minimum vertex cover set of G. MCRDSL energy of a graph G is defined as sum of eigenvalues of RQ_C. Extremal graphs with respect to MCRDSL energy of graph are characterized. We also obtain some bounds on MCRDSL energy of a graph and MCRDSL spectral radius of G, which is the largest eigenvalue of the matrix $RQ_C(G)$ of graphs.

1 Introduction

Throughout the paper, we consider G as a simple connected graph with vertex set $V(G)$ and edge set $E(G)$. A graph G of order n and size m is called an (n, m) graph. Distance between two vertices u and v is denoted by $d(u, v)$. The diameter of G is the maximum distance between any pair of vertices and is denoted by $\mathrm{diam}(G)$.

For a vertex v, $\deg(v)$ denotes the degree of v. Energy of a graph introduced by Ivan Gutman [12] as the sum of the absolute values of the eigenvalues of adjacency matrix of G. The concept of energy of graph have been extensively studied; for more information we refer to surveys [13, 23, 24]. Various kinds of graph energy such as Laplacian energy [14], minimum covering energy [1], minimum covering distance energy [21], and minimum covering Harary energy [22] of a graph were proposed and some mathematical aspects of them were investigated. A subset C of $V(G)$ is called a vertex covering set of G if every edge of G is incident to at least one vertex of C. A vertex covering set with minimum cardinality is called minimum vertex covering set. The cardinality of a minimum vertex covering set in a graph G is known as the vertex covering number of G, denoted by $\tau(G)$. A set of vertices that no pair of which are adjacent is called vertex independent set.

A vertex independent set with maximum cardinality is called maximum vertex independent set.

The cardinality of maximum independent set in G is called independence number of G, denoted by $\alpha(G)$. Clearly if C is a vertex covering set of G, the $V(G) - C$ make an independent set for G. This follows the well known relation $\tau(G) + \alpha(G) = n$, where n is order of G. Two distinct edges in a graph G

are independent if they do not share a common vertex in G. A matching of a graph G is a set of pairwise independent edges in G. The matching number of G, $\beta(G)$ is the number of edges in the largest matching of G. The investigation of matrices related to various graph structures is a very large and growing area of research. In what follows, some of such matrices are introduced.

Let C be a minimum covering set of a graph G. The minimum covering matrix [1] of G is defined as $A_C(G) = (a_{ij})$, where

$$a_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in E(G) \\ 1 & \text{if } i = j \text{ and } v_i \in C \\ 0 & \text{otherwise.} \end{cases}$$

The Harary matrix of a graph G, $RD(G)$ was introduced by Ivanciuc et al [17] and successfully used in computer generation of acyclic graphs based on local vertex invariants and topological indices. The Harary matrix $RD(G) = (RD_{ij})$ is a square matrix of order n, where

$$RD_{ij} = \begin{cases} 0 & \text{if } i = j \\ \dfrac{1}{d(v_i, v_j)} & \text{otherwise.} \end{cases}$$

The Harary matrix can be used to derive a variant of the Balaban index, Harary index and topological indices based on reciprocal distance in graphs. The minimum covering energy of G is defined to be absolute values of the eigenvalues of $A_C(G)$. The minimum covering Harary matrix [22] of G, is a square matrix $n \times n$ defined as $RDC(G) = (RDC_{ij})$, where

$$RDC_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } v_i \in C \\ 0 & \text{if } i = j \text{ and } v_i \notin C \\ \dfrac{1}{d(v_i, v_j)} & \text{otherwise.} \end{cases}$$

Analogously, minimum covering Harary energy of G is defined as $HE_C(G) = \sum_{i=1}^{n} |\lambda_i|$, where $\lambda_1, \lambda_2, \ldots, \lambda_n$ are eigenvalues of $H_C(G)$. The mathematical aspects of the minimum covering Harary energy was reported in [22].

The reciprocal transmission $Tr'_G(v)$ of a vertex v is defined as

$$Tr'_G(v) = \sum_{u \in V(G)} \frac{1}{d_G(u, v)}, \quad u \neq v$$

and $Tr'(G)$ is the diagonal matrix whose main entries are the vertex reciprocal transmissions in G. For $1 \leq i \leq n$, one can easily see that $Tr'_G(v_i)$ is just the i-th row sum of $RD(G)$. The Harary index of a graph G, denoted by $H(G)$, has been introduced independently by Plavšić et al. [19] and by Ivanciuc et al. [17] in 1993. It has been named in honor of Professor Frank Harary on the occasion of his 70th birthday. The Harary index is defined as: $H(G) = \sum_{\{u,v\}\subseteq V(G)} \frac{1}{d(u,v)}$. Let α be a real number, we use notations $H_\alpha(G)$ $\sigma_\alpha(G)$ for $\sum_{\{u,v\}\subseteq V(G)} \frac{1}{d(u,v)^\alpha}$ and $\sum_{v\in V(G)} Tr'(v)^\alpha$, respectively. Note that if $\alpha \neq 1$ then $H_\alpha(G) = H(G)$ if and only if G is a complete graph. The first and the second Zagreb indices of a graph G, denoted by $M_1(G)$ and $M_2(G)$ are defined as:

$$M_1(G) = \sum_{uv\in E(G)} deg(u) + deg(v),$$

$$M_2(G) = \sum_{uv\in E(G)} deg(u)deg(v).$$

The *reciprocal distance signless Laplacian matrix* (or briefly RDSL matrix) is defined as $RQ(G) = Tr'(G) + RD(G)$. Since the matrix $RQ(G)$ is irreducible, non-negative, symmetric and positive semi-definite, all its eigenvalues are non-negative [2]. The set of eigenvalue of $RQ(G)$ is called RDSL spectrum of G.

Motivated by the concept of minimum covering distance matrix, we define the *minimum covering reciprocal distance signless Laplacian matrix* (or briefly MCRDSL matrix) of G as the square matrix of order n, $RQ_C(G) := (q_{i,j})$, where

$$q_{i,j} = \begin{cases} 1 + Tr'(v_i) & \text{if} \quad i = j \quad \text{and} \quad v_i \in C \\ Tr'(v_i) & \text{if} \quad i = j \quad \text{and} \quad v_i \notin C \\ \dfrac{1}{d(v_i, v_j)} & \text{otherwise.} \end{cases}$$

Let $\rho_1 \geq \rho_2 \geq \cdots \geq \rho_n$ be the eigenvalues of the RDSL matrix $RQ(G)$. The largest eigenvalue $\rho_1 = \rho(G)$ of $RQ(G)$ is called the RDSL *spectral radius* of G. By the Perron-Frobenius theorem, there is a unique normalized positive eigenvector of $RQ(G)$ corresponding to ρ_1, which is called the (RDSL) principal eigenvector of G. Since the matrices $RQ_C(G)$ is irreducible, non-negative, symmetric and positive semi-definite, all their eigenvalues are non-negative.

For MCRDSL matrix, *auxiliary energy* (briefly MCRDSL energy) is defined as sum of its eigenvalues and denoted by $E_{RQ_C}(G)$.

This paper is organized as follows. In the next section, RDSL spectrum of some classes of graphs that are constructed by graph operations, is determined. In section 3, MCRDSL spectrum of some standard graphs such as complete graph, complete bipartite graph and cocktail party graph are computed. Extremal graphs with respect to MCRDSL energy of graph is obtained in section 4. Finally, in section 5, more bounds are given for MCRDSL energy of graph and RDSL spectral radius in terms of the eigenvalues of RDSL matrix, Zagreb indices and Harary index.

2 RDSL spectrum of some classes of graphs

It is a well known fact that almost all graphs are of diameter 2. Therefore in this section, we get the RDSL spectrum of some classes of graphs of diameter 2 or 3 that are arisen from graph operations such as cartesian product, InduBala product, extended double cover graph and complement of a graph.

The following lemma will be helpful in the sequel.

Lemma 1 [11] *Let*

$$A = \begin{pmatrix} A_0 & A_1 \\ A_1 & A_0 \end{pmatrix}$$

be a symmetric 2×2 *block matrix. Then, the spectrum of* A *is the union of the spectra of* $A_0 + A_1$ *and* $A_0 - A_1$.

We begin first with cartesian product of K_2 and a graph of diameter at most 2. The cartesian product of two graphs G and H, $G \times H$ is the graph with vertex set $V(G) \times V(H)$ and two vertices (u_1, u_2) and (v_1, v_2) are adjacent if and only if $u_1 = v_1$ and $u_2v_2 \in E(H)$ or $u_2 = v_2$ and $u_1v_1 \in E(G)$.

Theorem 2 *Let* G *be an* r*-regular graph of diameter at most 2 with an adjacency matrix* A *and* $\mathrm{Spec}(G) = \begin{pmatrix} r & \lambda_i \\ 1 & n_i \end{pmatrix}, i = 2, 3, \ldots, k$. *Then, the RDSL spectrum of* $H = G \times K_2$ *is as follows,* $\mathrm{Spec}(RQ(G)) =$

$$\begin{pmatrix} n + r - \frac{1}{6} & \frac{5n+4r+1}{3} & \frac{4\lambda_i+5n+4r+2}{6} & \frac{2\lambda_i+5n+4r-1}{6} \\ 1 & 1 & n_i & n_i \end{pmatrix}$$

for $i = 2, 3, \ldots, k$.

Proof. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$, $V(K_2) = \{w_1, w_2\}$. Let A and $\bar{A}$ be the adjacency matrix of G and $\bar{G}$ respectively and J denotes the $n \times n$ square matrix whose all entries are 1. From the fact $d_H((v_i, w_j), (v_s, w_t)) = d_G(v_i, v_s) + d_{K_2}(w_j, w_t) = d_G(v_i, v_s) + 1$, one can see that all vertices of H have a same reciprocal transmission and $Tr'_H(v_i, w_j) = \frac{1}{6}(5n + 4r + 1)$. Then $Tr'(G) = \frac{1}{6}(5n + 4r + 1)I$. Since G is a graph of diameter 1 or 2, diameter of H is 2 or 3 and H is $r + 1$ regular. Thus the $RD(H)$ is of the form

$$RD(H) = \begin{pmatrix} A + \frac{1}{2}\bar{A} & J - \frac{1}{2}A - \frac{2}{3}\bar{A} \\ J - \frac{1}{2}A - \frac{2}{3}\bar{A} & A + \frac{1}{2}\bar{A} \end{pmatrix}$$

and consequently the RDSL matrix of H is of the form

$$RQ(H) = \begin{pmatrix} A + \frac{1}{2}\bar{A} + (\frac{5}{6}n + \frac{2}{3}r + \frac{1}{6})I & J - \frac{1}{2}A - \frac{2}{3}\bar{A} \\ J - \frac{1}{2}A - \frac{2}{3}\bar{A} & A + \frac{1}{2}\bar{A} + (\frac{5}{6}n + \frac{2}{3}r + \frac{1}{6})I \end{pmatrix}.$$

Now, by Lemma 1 and the fact $\bar{A} = J - I - A$, the spectrum of $RQ(H)$ is the union of the spectra

$$\frac{1}{6}(4A + 5J + (5n + 4r + 2)I)$$

and

$$\frac{1}{6}(2A + J + (5n + 4r - 1)I).$$

□

The next considered graph operation is extended double cover graph of a graph that is introduced by N. Alon [3] to studying networks. Spectra of extended double cover graphs was investigated in [7]. Let G be a graph on the vertex set $\{v_1, \ldots, v_n\}$. The extended double cover graph of G, denoted by G^*, is the bipartite graph with partitions X and Y where $X = \{x_1, x_2, \ldots x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$, in which x_i and y_j are adjacent if and only if $i = j$ or v_i and v_j are adjacent in G. Now, we obtain the RDSL spectrum of the G^* of a regular graph G with diameter 2.

Theorem 3 *Let* G *be an* r*-regular graph on* n *vertices with diameter 2 and let* $\mathrm{Spec}(G) = \begin{pmatrix} r & \lambda_i \\ 1 & n_i \end{pmatrix}, i = 2, 3, \ldots, k$. *Then, the RDSL spectrum of* G^* *is* $\mathrm{Spec}(RQ(G^*)) =$

$$\begin{pmatrix} \frac{5n+4r+1}{3} & n-1 & \frac{4\lambda_i+5n+4r+2}{6} & \frac{-4\lambda_i+5n+4r-6}{6} \\ 1 & 1 & n_i & n_i \end{pmatrix},$$

$$i = 2, 3, \ldots, k.$$

Proof. First note that G^* is $r+1$ regular graph with diameter 3 and any vertex $v \in V(G^*)$ has reciprocal transmission $\frac{1}{6}(5n+4r+1)$. It is not difficult to see that $RD(G^*)$ has the form

$$RD(G^*) = \begin{pmatrix} \frac{1}{2}(J-I) & A + \frac{1}{3}\bar{A} + I \\ A + \frac{1}{3}\bar{A} + I & \frac{1}{2}(J-I) \end{pmatrix},$$

and then we have

$$RQ(G^*) = \begin{pmatrix} \frac{1}{6}(3J + (5n+4r-2)I) & A + \frac{1}{3}\bar{A} + I \\ A + \frac{1}{3}\bar{A} + I & \frac{1}{6}(3J + (5n+4r-2)I) \end{pmatrix}.$$

Then, by Lemma 1, the spectrum of $RQ(G)$ is the union of the spectra

$$\frac{1}{6}(4A + 5J + (5n+4r+2)I)$$

and

$$\frac{1}{6}(-4A + J + (5n+4r-6)I).$$

□

Next graph operation is InduBala product. InduBala product of graphs introduced in [16], where the distance spectrum of InduBala product of graphs is determined. Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs on disjoint sets of n_1 and n_2 vertices, respectively, then their *union* is the graph $G_1 \cup G_2 =$

$(V_1 \cup V_2, E_1 \cup E_2)$. Their *join* is denoted by $G_1 \nabla G_2$ and consists of $G_1 \cup G_2$ and all lines joining V_1 and V_2. The InduBala product of graphs is defined as follows. Let $V(G_1) = \{u_1, u_2, \ldots, u_{n_1}\}$ and $V(G_2) = \{v_1, v_2, \ldots, v_{n_2}\}$. Take a disjoint copy $G_1' \nabla G_2'$ of $G_1 \nabla G_2$ with vertex sets $V(G_1') = \{u_1', u_2', \ldots, u_{n_1}'\}$ and $V(G_2') = \{v_1', v_2', \ldots, v_{n_2}'\}$. Now make v_i adjacent with v_i' for each $i = 1, 2, \ldots, n_2$. Structure of InduBala product of two graphs P_4 and K_3 is illustrated in Figure 1. Occasionally, it so happens that for certain families of

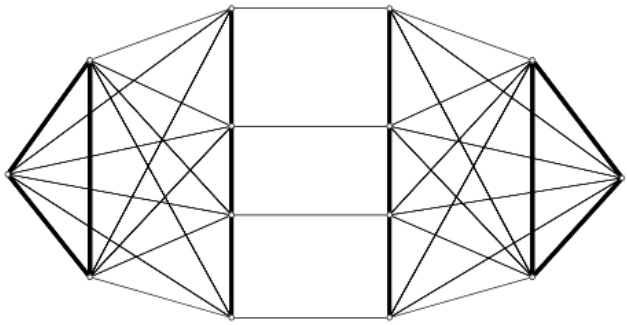

Figure 1: *The graph* $K_3 \blacktriangledown P_4$.

graphs it is possible to identify a graph by looking at the spectrum. Now, we describe the RDSL spectrum of the join of a regular graph with the union of two regular graphs of distinct vertex degrees.

Theorem 4 *For* $i = 0, 1, 2$, *let* G_i *be an* r_i*-regular graph of order* n_i *and eigenvalues* $\lambda_{i,1} = r_i \geq \lambda_{i,2} \geq \ldots \geq \lambda_{i,n_i}$ *of the adjacency matrix* $A(G_i)$. *Then the RDSL spectrum of* $G_0 \nabla (G_1 \cup G_2)$ *consists of eigenvalues*

$$\frac{1}{2}(2m - n_0 + \lambda_{0,j} + r_0 - 2), \quad j = 2, \ldots, n_0,$$

and

$$\frac{1}{2}(m + n_0 + \lambda_{i,j} + r_i - 2), \quad i = 1, 2 \quad \textit{and} \quad j = 2, 3, \ldots, n_i,$$

where $m = \sum_{i=0}^{2} n_i$, *and three more eigenvalues which are the eigenvalues of the following matrix*

$$\begin{pmatrix} m + r_0 - 1 & n_1 & n_2 \\ n_0 & m - \frac{1}{2}n_2 + r_1 - 1 & \frac{1}{2}n_2 \\ n_0 & \frac{1}{2}n_1 & m - \frac{1}{2}n_1 + r_2 - 1 \end{pmatrix}. \tag{1}$$

Proof. The reciprocal distance signless Laplacian matrix $F = G_0 \nabla (G_1 \cup G_2)$ has the form

$$\begin{pmatrix} S_0 & J & J \\ J & S_1 & \frac{1}{2}J \\ J & \frac{1}{2}J & S_2 \end{pmatrix},$$

where

$$S_0 = \frac{1}{2}\left((2m - n_0 + r_0 - 2)I + J + A(G_0)\right)$$

and for $i = 1, 2$

$$S_i = \frac{1}{2}\left((m + n_0 + r_i - 2)I + J + A(G_i)\right).$$

As a regular graph, G_0 has the all-one vector $\mathbf{1}$ as an eigenvector corresponding to the eigenvalue r_0, while all the other eigenvectors are orthogonal to $\mathbf{1}$. Let λ be an arbitrary eigenvalue of the adjacency matrix of G_0 with corresponding eigenvector X, such that $\mathbf{1}^T X = 0$, then $[X^T \quad 0 \quad 0]^T$ is an eigenvector of $RQ(F)$ corresponding to the eigenvalue $\frac{1}{2}(2m - n_0 + r_0 - 2 + \lambda)$. Now, let μ, ξ be arbitrary eigenvalues of the adjacency matrix of G_1 and G_2 with corresponding eigenvector Y and Z, respectively. In a similar way the vectors $[0 \quad X^T \quad 0]^T$ and $[0 \quad 0 \quad X^T]^T$ are eigenvectors of $RQ(F)$ with corresponding eigenvalues $\frac{1}{2}(m + n_0 + r_1 - 2 + \mu)$ and $\frac{1}{2}(m + n_0 + r_2 - 2 + \xi)$, respectively.

In this way we obtain eigenvectors of the form $[X^T \quad 0 \quad 0]^T$, $[0 \quad X^T \quad 0]^T$ and $[0 \quad 0 \quad X^T]^T$ and these account for a total of $m - 3$ eigenvectors. All these eigenvectors are orthogonal to $[\mathbf{1}^T \quad 0 \quad 0]^T$, $[0 \quad \mathbf{1}^T \quad 0]^T$ and $[0 \quad 0 \quad \mathbf{1}^T]^T$. Thus the remaining three eigenvectors of $RQ(F)$ are of the form $[\alpha\mathbf{1} \quad \beta\mathbf{1} \quad \gamma\mathbf{1}]^T$ for some $(\alpha, \beta, \gamma) \neq (0, 0, 0)$.

If ν is an eigenvalue of $RQ(F)$ with an corresponding eigenvector $(\alpha\mathbf{1}, \beta\mathbf{1}, \gamma\mathbf{1})^T$, then from $RQ(F)(\alpha\mathbf{1}, \beta\mathbf{1}, \gamma\mathbf{1})^T = \nu(\alpha\mathbf{1}, \beta\mathbf{1}, \gamma\mathbf{1})^T$, and $A(G_i)\mathbf{1} = r_i\mathbf{1}$ for $i = 0, 1, 2$, we get the system of equations:

$$\begin{aligned} (m + r_0 - 1)\alpha + n_1\beta + n_2\gamma &= \nu\alpha, \\ n_0\alpha + (m - \frac{1}{2}n_2 + r_1 - 1)\beta + \frac{1}{2}n_2\gamma &= \nu\beta, \\ n_0\alpha + \frac{1}{2}n_1\beta + (m - \frac{1}{2}n_1 + r_2 - 1)\gamma &= \nu\gamma, \end{aligned}$$

which have a nontrivial solution if and only if ν is an eigenvalue of (1). Further, it is obvious from above that any nontrivial solution of above system forms

an eigenvector of $RQ(F)$ corresponding to eigenvalue ν. Since all 3 remaining eigenvectors of $RQ(F)$ must be formed in this way, we conclude that each eigenvalue of (1) is an eigenvalue of $RQ(F)$ as well. □

Theorem 5 *For $i = 1, 2$, let G_i be an r_i-regular graph of order n_i and let $\lambda_{i,1} = r_i \geq \lambda_{i,2} \geq \ldots \geq \lambda_{i,n_i}$ be the eigenvalues of the adjacency matrix $A(G_i)$. Then the RDSL spectrum of $G_1 \blacktriangledown G_2$ is the set consisting of eigenvalues*

$$\frac{1}{2}\left(\frac{5}{3}n_1 + 3n_2 + \lambda_{1,j} + r_1 - 2\right), \quad j = 2, 3, \ldots, n_1 \quad \textit{each with multiplicity 2},$$

and

$$\frac{1}{2}\left(3n_1 + \frac{5}{3}n_2 + \frac{4}{3}\lambda_{2,j} + \frac{4}{3}r_2 + \frac{2}{3}\right) \quad j = 2, 3, \ldots, n_2$$

and

$$\frac{1}{2}\left(3n_1 + \frac{5}{3}n_2 + \frac{2}{3}\lambda_{2,j} + \frac{4}{3}r_2 - 2\right), \quad j = 2, 3, \ldots, n_2,$$

also four more eigenvalues which are the eigenvalues of the matrix

$$\begin{pmatrix} m_1 & n_2 & \frac{1}{2}n_2 & \frac{1}{3}n_1 \\ n_1 & m_2 & \frac{1}{3}(n_2 + \frac{1}{6}r_2 + \frac{2}{3}) & \frac{1}{2}n_1 \\ \frac{1}{2}n_1 & \frac{1}{3}(n_2 + \frac{1}{6}r_2 + \frac{2}{3}) & m_2 & n_1 \\ \frac{1}{3}n_1 & \frac{1}{2}n_2 & n_2 & m_1 \end{pmatrix}, \tag{2}$$

where $m_1 = \frac{1}{2}\left(\frac{8}{3}n_1 + 3n_2 + 2r_1 - 2\right)$ *and* $m_2 = \frac{1}{2}\left(\frac{8}{3}n_2 + 3n_1 + \frac{7}{3}r_2 - \frac{2}{3}\right)$.

Proof. The RDSL matrix $H = G_1 \blacktriangledown G_2$ has the form

$$RQ(H) = \begin{pmatrix} S_1 & J & \frac{1}{2}J & \frac{1}{3}J \\ J & S_2 & \frac{1}{3}J + \frac{2}{3}I + \frac{1}{6}A(G_2) & \frac{1}{2}J \\ \frac{1}{2}J & \frac{1}{3}J + \frac{2}{3}I + \frac{1}{6}A(G_2) & S_3 & J \\ \frac{1}{3}J & \frac{1}{2}J & J & S_4 \end{pmatrix},$$

where

$$S_i = \frac{1}{2}\left(J + A(G_1) + (\frac{5}{3}n_1 + 3n_2 + r_1 - 2)I\right), \quad i = 1, 4$$

and

$$S_i = \frac{1}{2}\left(J + A(G_2) + (3n_1 + \frac{5}{3}n_2 + \frac{4}{3}r_2 - \frac{2}{3})I\right), \quad i = 2, 3.$$

By analogy to the proof of Theorem 4, let λ be an arbitrary eigenvalue of the adjacency matrix of G_1 with corresponding eigenvector X, such that $\mathbf{1}^T X = 0$. Then $[X^T \quad 0 \quad 0 \quad 0]^T$ is an eigenvector of $RQ(H)$ corresponding to the eigenvalue $\frac{1}{2}(\frac{5}{3}n_1 + 3n_2 + \lambda + r_1 - 2)$. In a similar way the vector $[0 \quad 0 \quad 0 \quad X^T]^T$ is an eigenvector of $RQ(H)$ corresponding to the eigenvalue $\frac{1}{2}(\frac{5}{3}n_1 + 3n_2 + \lambda + r_1 - 2)$. Now let μ be an arbitrary eigenvalue of the adjacency matrix of G_2 with corresponding eigenvector Y, such that $\mathbf{1}^T Y = 0$. Then by a similar argument we see that the vectors $[0 \quad Y^T \quad Y^T \quad 0]^T$ and $[0 \quad -Y^T \quad Y^T \quad 0]^T$ are eigenvectors of $RQ(H)$ with corresponding eigenvalues $\frac{1}{2}\left(3n_1 + \frac{4}{3}\mu + \frac{5}{3}n_2 + \frac{4}{3}r_2 + \frac{2}{3}\right)$ and $\frac{1}{2}\left(3n_1 + \frac{2}{3}\mu + \frac{5}{3}n_2 + \frac{4}{3}r_2 - 2\right)$ respectively. In this way we obtain eigenvectors of the form $[X^T \quad 0 \quad 0 \quad 0]^T$, $[0 \quad 0 \quad 0 \quad X^T]^T$, $[0 \quad Y^T \quad Y^T \quad 0]^T$ and $[0 \quad -Y^T \quad Y^T \quad 0]^T$ and these account for a total of $2(n_1 + n_2) - 4$ eigenvectors. All these eigenvectors are orthogonal to $[\mathbf{1}^T \quad 0 \quad 0 \quad 0]^T$, $[0 \quad \mathbf{1}^T \quad 0 \quad 0]^T$, $[0 \quad 0 \quad \mathbf{1}^T \quad 0]^T$ and $[0 \quad 0 \quad 0 \quad \mathbf{1}^T]^T$. This means that these four vectors span the space spanned by the remaining four eigenvectors of $RQ(H)$. Thus the remaining four eigenvectors of $RQ(H)$ are of the form $[\alpha\mathbf{1} \quad \beta\mathbf{1} \quad \gamma\mathbf{1} \quad \delta\mathbf{1}]^T$ for some $(\alpha, \beta, \gamma, \delta) \neq (0, 0, 0, 0)$. If ν is an eigenvalue of $RQ(H)$ with an eigenvector $(\alpha\mathbf{1} \quad \beta\mathbf{1} \quad \gamma\mathbf{1} \quad \delta\mathbf{1})^T$, from

$RQ(H)(\alpha\mathbf{1} \quad \beta\mathbf{1} \quad \gamma\mathbf{1} \quad \delta\mathbf{1})^T = \nu(\alpha\mathbf{1} \quad \beta\mathbf{1} \quad \gamma\mathbf{1} \quad \delta\mathbf{1})^T$, and $A(G_i)\mathbf{1} = r_i\mathbf{1}$ for $i = 1, 2$, we get the system of equations:

$$\begin{aligned}
&\frac{1}{2}\left(\frac{8}{3}n_1 + 3n_2 + 2r_1 - 2\right)\alpha + n_2\beta + \frac{1}{2}n_2\gamma + \frac{1}{3}n_1\delta = \nu\alpha,\\
&n_1\alpha + \frac{1}{2}\left(\frac{8}{3}n_2 + 3n_1 + \frac{7}{3}r_2 - \frac{2}{3}\right)\beta + \frac{1}{3}(n_2 + \frac{1}{6}r_2 + \frac{2}{3})\gamma + \frac{1}{2}n_1\delta = \nu\beta,\\
&\frac{1}{2}n_1\alpha + \frac{1}{3}(n_2 + \frac{1}{6}r_2 + \frac{2}{3})\beta + \frac{1}{2}\left(\frac{8}{3}n_2 + 3n_1 + \frac{7}{3}r_2 - \frac{2}{3}\right)\gamma + n_1\delta = \nu\gamma\\
&\frac{1}{3}n_1\alpha + \frac{1}{2}n_2\beta + n_2\gamma + \frac{1}{2}\left(\frac{8}{3}n_1 + 3n_2 + 2r_1 - 2\right)\delta = \nu\delta,
\end{aligned}$$

which have a nontrivial solution if and only if ν is an eigenvalue of (2). Further, it is obvious from above that any nontrivial solution of above system forms an eigenvector of $RQ(H)$ corresponding to eigenvalue ν. Since all four remaining eigenvectors of $RQ(H)$ must be formed in this way, we conclude that each eigenvalue of (2) is an eigenvalue of $RQ(H)$ as well. □

3 MCRDSL energy of some standard graphs

In this section, E_{RQ_C} is computed for some standard graphs such as complete graph, complete bipartite graph and cocktail party graph.

Example 6 Complete graph K_n.
For $n \geq 2$, the eigenvalues of the minimum covering Harary matrix of complete graph K_n was determined in [1, 22] *as* $Spec(RD_C(K_n)) =$

$$\begin{pmatrix} 0 & \frac{n-1+\sqrt{(n+3)(n-1)}}{2} & \frac{n-1-\sqrt{(n+3)(n-1)}}{2} \\ n-2 & 1 & 1 \end{pmatrix}.$$

Easily one can see that for complete graph K_n with vertex set $V(K_n) = \{v_1, v_2, \ldots, v_n\}$ and minimum covering set $C = \{v_1, v_2, \ldots, v_{n-1}\}$,

$$RQ_C(K_n) = (n-1)I + A_C(K_n).$$

Therefore, the eigenvalues of the matrix $RQ_C(K_n)$ *are as* $Spec(RQ_C(K_n)) =$

$$\begin{pmatrix} n-1 & \frac{3(n-1)+\sqrt{(n+3)(n-1)}}{2} & \frac{3(n-1)-\sqrt{(n+3)(n-1)}}{2} \\ n-2 & 1 & 1 \end{pmatrix}$$

and consequently $E_{RQ_C}(K_n) = n^2 - 1$. Another simpler way to compute the $E_{RQ_C}(K_n)$ is as follows

$$E_{RQ_C}(K_n) = trace(RQ_C(K_n)) = |C| + \sum_{i=1}^{n} Tr'(v_i) = n - 1 + n(n-1) = n^2 - 1.$$

Complete bipartite graph $K_{m,n}$**.**
Let $V(K_{m,n}) = \{v_1, v_2, \ldots, v_m\} \bigcup \{w_1, w_2, \ldots, w_n\}$ and $C = \{v_1, v_2, \ldots, v_m\}$ be the minimum covering set of $K_{m,n}$, $(m \leq n)$. Then,

$$E_{RQ_C}(K_{m,n}) = |C| + \sum_{i=1}^{m} Tr'(v_i) + \sum_{i=1}^{n} Tr'(w_i) = 2mn + \frac{1}{2}(m^2 + n^2 + m - n).$$

Cocktail party graph.
The Cocktail party graph of order n, $K_{n\times 2}$ is formed from the complete graph K_{2n} by removing n disjoint edges. Note that all vertices of $K_{n\times 2}$ have a same reciprocal transmission $\frac{3}{2}(n-1)$ and a minimum covering set is of order $2n-2$. Therefore,

$$E_{RQ_C}(K_{n\times 2}) = 2n - 2 + 2n\left(\frac{3}{2}(n-1)\right) = 3n^2 - n - 2.$$

4 Extremal graphs with respect to ERQ_C

In this section, we are concerned with the extremal graphs with respect to the minimum covering reciprocal distance signless Laplacian energy.

Theorem 7 *Let* G *be a simple graph with* n *vertices and* m *edges. If* C *is the minimum covering set of* G*, then*

$$ERQ_C(G) = \tau(G) + 2H(G).$$

Proof. Let $\rho_1, \rho_2, \ldots, \rho_n$ be the eigenvalues of the matrix $RQ_C(G)$. The result follows from the well known fact that $\sum_{i=1}^{n} \rho_i = Trace(RQ_C(G))$ and the eigenvalues of $RQ_C(G)$ are non-negative. □

Corollary 8 *Let* G *be a graph of order* n*, then* $E_{RQ_C}(G) \leq n^2-1$*. The equality holds if and only if* $G \cong K_n$*.*

Proof. It is a well known fact that adding any edge to graph G, increase the Harary index of G and do not decrease the vertex covering number of G. Consequently, K_n has the maximum Harary index among all graphs of order n. Clearly, $\tau(G) \leq n-1$. Therefore

$$E_{RQ_C}(G) = \tau(G) + 2H(G) \leq n-1+n(n-1) = n^2-1,$$

and the equality holds if and only if $G \cong K_n$. □

Corollary 9 *Let* $G \neq K_n$ *be a graph of order* n*. Then*

$$ERQ_C(G) \leq n^2-3,$$

with equality holds if and only if $G \cong K_n - e$*, where* e *is an edge of* K_n*.*

Let $G(n,\beta)$ denotes the constructed graph by join of K_β and $\overline{K_{n-\beta}}$. Note that $\tau(G(n,\beta)) = \beta$. Let $T(n,\beta)$ be a tree obtained from $K_{1,n-\beta}$ by attaching a pendant vertex to its $\beta-1$ pendant vertices. In [15] and [9] lower and upper bounds on Harary index were obtained in terms of independence number and matching number. It was proved that graphs $G(n,\beta)$ and $T(n,\beta)$ have the maximum value of Harary index among all graphs and trees of a same order n and same independence number $n-\beta$, respectively. Consequently, $G(n,\beta)$ and $T(n,\beta)$ get the maximum value of ERQ_C among graphs of order n and vertex covering number β as well. Hence we conclude that:

Theorem 10 *Let* G *be a graph of order* n *and vertex cover number* β*. Then,*

$$ERQ_C(G) \leq 2\binom{\beta}{2} + \binom{n-\beta}{2} + \beta(2n-2\beta+1).$$

The equality holds if and only if $G \cong G(n,\beta)$*.*

In [10], it is proved that of all trees of order n, star graph S_n is the unique graph of maximum value of Harary index. But it is not true for ERQ_C. For example, see the figure 2, two graphs S_5 and $T(5,2)$ where $ERQ_C(T(5,2)) > ERQ_C(S_5)$.

In the following, an upper bound is given for trees of order n and vertex cover β.

Figure 2: *Graphs* S_5 *and* $T(5,2)$, $ERQ_C(T(5,2)) > ERQ_C(S_5)$

Theorem 11 *Let* T *be a tree of order* n *and vertex cover number* β. *Then,*

$$ERQ_C(T) \leq \frac{1}{12}\left(6n^2 + (10 - 4\beta)n + \beta^2 + 21\beta - 22\right).$$

The equality holds if and only if $T \cong T(n, \beta)$.

It is a well known fact that for any bipartite graph G of order n, $\alpha(G) + \beta(G) = n$, (see [5]). Therefore, the following corollary is immediate.

Corollary 12 *Let* T *be a tree of order* n. *If* T *has perfect matching, then*

$$ERQ_C(T) \leq \frac{1}{48}(15n^2 + 82n - 88),$$

with equality holding if and only if $T \cong T(n, \frac{n}{2})$.

Lemma 13 *Let* T *be a tree of order* n *and diameter* d. *Then* $\lceil \frac{d+1}{2} \rceil \leq \alpha(T) \leq \lceil n - \frac{d}{2} \rceil$.

Proof. Notice that T has P_{d+1} as subgraph. The proof follows from the fact that $\alpha(T) \geq \alpha(P_{d+1}) = \lceil \frac{d+1}{2} \rceil$ and $\tau(T) \geq \tau(P_{d+1}) = \lfloor \frac{d}{2} \rfloor$. □

A lower bound for Harary index among trees of diameter d and order n is obtained by Xu et al. [9] as follow.

Lemma 14 *Let* T *be a tree of order* n *and diameter* d. *Then*

$$H(T) \leq \frac{1}{24}(M_1(G) + 2M_2(G) + 3n^2 + 11n - 24),$$

with equality holds if and only if T *is of diameter at most* 4.

Now, an upper bound for ERQ_C of trees is obtained by using Lemmas 13 and 14 as:

Corollary 15 *Let T be a tree of order n and diameter d. Then*

$$ERQ_C(T) \leq \frac{1}{12}(M_1 + 2M_2 + 3n^2 + 23n - 24) - \lfloor \frac{d+1}{2} \rfloor,$$

with equality holds if and only if $T = P_n$ where $2 \leq n \leq 5$ or T is a graph constructed by P_5 and attaching a vertex to the central vertex of P_5.

Proof. From Lemma 13, we get $\tau(T) \leq n - \lfloor \frac{d+1}{2} \rfloor$. Among trees of diameter $d \leq 4$, trees P_n, $2 \leq n \leq 5$ and a graph constructed by P_5 and attaching a vertex to the central vertex of P_5, have vertex cover $\tau(T) = n - \lfloor \frac{d+1}{2} \rfloor$. □

Let $\Gamma(n, d)$ be the set of all graphs of order n and diameter d, obtained from a path P_{d+1} and a complete graph K_{n-d-1} that each vertex of K_{n-d-1} is connected to a central vertex in P_{d+1} and its two neighbors. In [10], some upper and lower bounds were obtained for graphs of given diameter and number of edges. In the following, we show that graphs of $\Gamma(n, d)$ get the maximum value of Harary index and ERQ_C among graphs of given order n and diameter d. Let $H_n = \sum_{k=1}^{n} \frac{1}{k}$ denotes the n-th harmonic number. It is easy to see that $H(P_n) = nH_{n-1} - n + 1$.

Theorem 16 *Let G be a graph on n vertices and diameter d. Then*

$$H(G) \leq (d+1)H_d - d + \binom{n-d-1}{2} + (n-d-1)(H_{\lfloor \frac{d}{2} \rfloor} + H_{\lfloor \frac{d+1}{2} \rfloor} + 1),$$

with equality holds if and only if $G \in \Gamma(n, d)$.

Proof. Let P_{d+1} be a path connecting two vertices of distance d. Let $W_1 = V(P_{d+1})$ and $W_2 = V(G) - V(P_{d+1})$. Note that each vertex of W_2 is connected to at most 3 vertices of W_1. It is not difficult to see that in a path P_m, a central vertex x has maximum reciprocal transmission $Tr'(x) = H_{\lfloor \frac{m-1}{2} \rfloor} + H_{\lfloor \frac{m}{2} \rfloor}$.

Let x be a central vertex of P_{d+1}. Then

$$\begin{aligned} H(G) &= H(P_{d+1}) + \sum_{\{u,v\} \subseteq W_2} \frac{1}{d(u,v)} + \sum_{u \in W_1} \sum_{v \in W_2} \frac{1}{d(u,v)} \\ &\leq H(P_{d+1}) + \binom{n-d-1}{2} + (n-d-1)(Tr'_{P_{d+1}}(x) + 1). \end{aligned}$$

The equality holds if and only if all vertices in W_2 are adjacent and for each vertex $v \in W_2$, the equality $\sum_{u \in W_1} \frac{1}{d(u,v)} = Tr'_{P_{d+1}}(x) + 1$ holds if and only if v is adjacent to x and its two neighbors in P_{d+1}. Thus $G \in \Gamma(n, d)$. □

5 More bounds on ERQ_C and largest eigenvalue of RQ_C matrix

The following lemmas refer to the real non-negative numbers, and will be helpful in the sequel.

Lemma 17 [18] *If a_i and $b_i, 1 \le i \le n$, are non-negative real numbers, then*

$$\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2 - \Big(\sum_{i=1}^{n} a_i b_i\Big)^2 \le \frac{n^2}{4}(M_1 M_2 - m_1 m_2)^2,$$

where $M_1 = \max_{1\le i\le n} a_i$, $M_2 = \max_{1\le i\le n} b_i$, $m_1 = \min_{1\le i\le n} a_i$ and $m_2 = \min_{1\le i\le n} b_i$.

Lemma 18 [20] *If a_i and $b_i, 1 \le i \le n$, are positive real numbers, then*

$$\sum_{i=1}^{n} a_i^2 \sum_{i=1}^{n} b_i^2 \le \frac{1}{4}\left(\sqrt{\frac{M_1 M_2}{m_1 m_2}} + \sqrt{\frac{m_1 m_2}{M_1 M_2}}\right)^2 \left(\sum_{i=1}^{n} a_i b_i\right)^2,$$

where $M_1 = \max_{1\le i\le n} a_i$, $M_2 = \max_{1\le i\le n} b_i$, $m_1 = \min_{1\le i\le n} a_i$ and $m_2 = \min_{1\le i\le n} b_i$.

Lemma 19 [8] *If a_i and $b_i, 1 \le i \le n$, are non-negative real numbers for which there exist real numbers r and R, so that $r \le \frac{b_i}{a_i} \le R, a_i \ne 0$, for each $i = 1, 2, \ldots, n$. Then*

$$\sum_{i=1}^{n} b_i^2 + rR \sum_{i=1}^{n} a_i^2 \le (r + R) \sum_{i=1}^{n} a_i b_i.$$

Equality holds if and only if $b_i = a_i r$ or $b_i = a_i R$ for at least one i, where $1 \le i \le n$.

Lemma 20 *Let G be a graph of order n and C be a minimum vertex covering set. If $\rho_1, \rho_2, \ldots, \rho_n$ are the eigenvalues of $RQ_C(G)$, then*

$$\sum_{i=1}^{n} \rho_i^2 = \sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v \in C} Tr'(v).$$

Proof. We have

$$\begin{aligned}\sum_{i=1}^{n}\rho_i^2 &= \sum_{j=1}^{n}\sum_{i=1}^{n} q_{ij}q_{ji} = \sum_{i=1}^{n}(q_{ii})^2 + 2\sum_{1\le i<j\le n}(q_{ij})^2\\ &= \sum_{v\notin C}(\mathrm{Tr}'(v))^2 + \sum_{v\in C}(1+\mathrm{Tr}'(v))^2 + 2\sum_{1\le i<j\le n}(q_{ij})^2\\ &= \sigma_2(G) + \tau(G) + 2H_2(G) + 2\sum_{v\in C}\mathrm{Tr}'(v).\end{aligned}$$

□

Corollary 21 *Let* G *be an* (n,m) *graph with diameter at most 2 and* C *be a minimum covering set. If* $\rho_1,\rho_2,\ldots,\rho_n$ *are the eigenvalues of* $RQ_C(G)$*, then*

$$\sum_{i=1}^{n}\rho_i^2 = \frac{1}{2}(n+1)\binom{n}{2} + \frac{1}{4}M_1(G) + nm + n\tau(G) + \sum_{v\in C}\deg(v),$$

where $M_1(G) = \sum_{i=1}^{n}\deg(v_i)^2$ *is known as the first Zagreb index.*

Proof. Let $V(G) = \{v_1, v_2, \ldots, v_n\}$. Since $\mathrm{diam}(G) \le 2$, hence we get $\mathrm{Tr}'(v) = \frac{1}{2}(n+\deg(v)-1)$. Let $RQ_C(G) = (q_{ij})$. From the fact $\sum_{i=1}^{n}\rho_i^2 = \mathtt{trace}(RQ_C(G))^2$, we get

$$\begin{aligned}\sum_{i=1}^{n}\rho_i^2 &= \sigma_2(G) + \tau(G) + 2H_2(G) + 2\sum_{v\in C}\mathrm{Tr}'(v)\\ &= \sum_{v\in V(G)}(\mathrm{Tr}'(v))^2 + \tau(G) + 2\sum_{v\in C}\mathrm{Tr}'(v) + m + \binom{n}{2}\\ &= \frac{1}{4}\Big(n(n-1)^2 + M_1(G) + 4(n-1)m\Big) + n\tau(G)\\ &\quad + \sum_{v\in C}\deg(v) + m + \binom{n}{2}\\ &= \frac{1}{2}(n+1)\binom{n}{2} + \frac{1}{4}M_1(G) + nm + n\tau(G) + \sum_{v\in C}\deg(v),\end{aligned}$$

and the proof is complete. □

In the following, some bounds are presented for $E_{RQ_C}(G)$.

Theorem 22 *Let* G *be a simple graph of order* n. *If* C *is the minimum vertex covering set and* $\Delta = \det(RQ_C(G))$, *then*

$$\begin{aligned}&\sqrt{\sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v) + n(n-1)\Delta^{\frac{2}{n}}}\\ &\leq E_{RQ_C}(G)\\ &\leq \sqrt{n\Big(\sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v)\Big)}.\end{aligned}$$

Proof. Let $\rho_1, \rho_2, \ldots, \rho_n$ be the eigenvalues of $RQ_C(G)$. First, we show the right-hand side inequality. Setting $a_i = 1$ and $b_i = \rho_i$ in the Cauchy Schwarz inequality, $(\sum_{i=1}^n a_i b_i)^2 \leq (\sum_{i=1}^n a_i^2)(\sum_{i=1}^n b_i^2)$ and using Lemma 20, we get

$$\begin{aligned}\Big(\sum_{i=1}^n \rho_i\Big)^2 &\leq \Big(\sum_{i=1}^n 1\Big)\Big(\sum_{i=1}^n \rho_i^2\Big)\\ \Big(E_{RQ_C}(G)\Big)^2 &\leq n\Big(\sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v)\Big).\end{aligned}$$

For the left inequality, consider the AM-GM inequality (which says that arithmetic mean of a set of non-negative real number is greater than or equal to geometric mean of them), on the set of $\{\rho_i\rho_j | 1 \leq i < j \leq n\}$, then

$$\begin{aligned}\frac{1}{\binom{n}{2}}\sum_{1\leq i<j\leq j} \rho_i\rho_j &\geq \Big(\prod_{1\leq i<j\leq n} \rho_i\rho_j\Big)^{\frac{1}{\binom{n}{2}}}\\ &= \Big(\prod_{i=1}^n \rho_i\Big)^{\frac{n-1}{\binom{n}{2}}} = \Big(\prod_{i=1}^n \rho_i\Big)^{\frac{2}{n}}\\ &= \Delta^{\frac{2}{n}}.\end{aligned}$$

Now, we get

$$\begin{aligned}E^2_{RQ_C}(G) &= \Big(\sum_{i=1}^n \rho_i\Big)^2 = \sum_{i=1}^n \rho_i^2 + 2\sum_{1\leq i<j\leq n} \rho_i\rho_j\\ &\geq \sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v) + n(n-1)\Delta^{\frac{2}{n}}.\end{aligned}$$

Theorem 23 *If ρ_1 is the largest eigenvalue of $RQ_C(G)$, then $\rho_1 \geq \frac{4H(G)+\tau(G)}{n}$.*

Proof. Let $X = \underbrace{(1,1,\ldots,1)}_{n}^{T}$ be the all one vector. Then, by the Rayleigh Principle (see [4]),

$$\begin{aligned}\rho_1 &\geq \frac{X^T RQ_C(G)X}{X^T X} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}}{n} \\ &= \frac{2\sum_{1\leq i<j\leq n}\frac{1}{d(v_i,v_j)} + \sum\limits_{i=1,v_i\notin C}^{n} Tr'_{v_i} + \sum\limits_{i=1,v_i\in C}^{n}(1+Tr'_i)}{n} \\ &= \frac{4H(G)+\tau(G)}{n}.\end{aligned}$$

□

Using Lemmas 17, 18 and setting $a_i = 1$ and $b_i = \rho_i$, we get the following two lower bounds for E_{RQ_C} of a graph G.

Theorem 24 *Let G be a connected graph which $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_n$ are the eigenvalues of MCRDSL matrix of G. Then*

$$E_{RQ_C}(G) \geq \sqrt{n\left(\sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v)\right) - \frac{n^2}{4}(\rho_1-\rho_n)^2}, \quad (3)$$

and

$$E_{RQ_C}(G) \geq \frac{2\sqrt{\rho_1\rho_n}}{\rho_1+\rho_n}\sqrt{n\left(\sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v)\right)}. \quad (4)$$

Lemma 25 [6] *If a_i and $b_i, 1 \leq i \leq n$, are non-negative real numbers for which there exist real numbers a, b, A and B, so that for each $i = 1,\ldots,n$, we have $a \leq a_i \leq A$ and $b \leq b_i \leq B$. Then*

$$\left| n\sum_{i=1}^{n} a_i b_i - \sum_{i=1}^{n} a_i \sum_{i=1}^{n} b_i \right| \leq \alpha(n)(A-a)(B-b),$$

where $\alpha(n) = n[\frac{n}{2}](1-\frac{1}{n}[\frac{n}{2}])$, while $[x]$ denotes integer part of a real number x. Equality holds if and only if $a_1 = a_2 = \cdots = a_n$ and $b_1 = b_2 = \cdots = b_n$.

Another lower bound is obtained for E_{RQ_C} of a graph by applying Lemma 25 and setting $a_i = b_i = \rho_i$, $a = b = \rho_n$ and $A = B = \rho_1$.

Theorem 26 *Let G be a connected graph and $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_n$ be the eigenvalues of $RQ_C(G)$. Then*

$$E_{RQ_C}(G) \geq \sqrt{n\left(\sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v)\right) - \alpha(n)(\rho_1 - \rho_n)^2}. \quad (5)$$

Corollary 27 *Since $\alpha(n) = n[\frac{n}{2}](1 - \frac{1}{n}[\frac{n}{2}]) \leq \frac{n^2}{4}$, then according to (5), we have that*

$$\begin{aligned} E_{RQ_C}(G) &\geq \sqrt{n\left(\sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v)\right) - \alpha(n)(\rho_1 - \rho_n)^2} \\ &\geq \sqrt{n\left(\sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v)\right) - \frac{n^2}{4}(\rho_1 - \rho_n)^2}. \end{aligned}$$

This means that inequality (5) is stronger than inequality (3).

Theorem 28 *Let G be a connected graph and $\rho_1 \geq \rho_2 \geq \ldots \geq \rho_n$ be the eigenvalues of $RQ_C(G)$. Then*

$$E_{RQ_C}(G) \geq \frac{n\rho_1\rho_n + \sigma_2(G) + 2H_2(G) + \tau(G) + 2\sum_{v\in C} Tr'(v)}{\rho_1 + \rho_n}. \quad (6)$$

Proof. The result follows by setting $a_i = 1$, $b_i = \rho_i$ $R = \rho_1$ and $r = \rho_n$ in the Lemma 19. □

Acknowledgements. The authors would like to thank the editor and the referee for the helpful suggestions. The research of A. Alhevaz and E. Hashemi was in part supported by a grant from Shahrood University of Technology.

References

[1] C. Adiga, A. Bayad, I. Gutman, S. A. Srinivas, The minimum covering energy of a graph, *Kragujevac J. Sci.*, **34** (2012), 39–56. ⇒219, 220, 229

[2] A. Alhevaz, M. Baghipur, H. S. Ramane, Computing the reciprocal distance signless Laplacian eigenvalues and energy of graphs, *Le Matematiche*, (2018), in press. ⇒221

[3] N. Alon, Eigenvalues and expanders, *Combinatorica*, **6** (1986), 83–96. ⇒223

[4] G. E. Backus, J. F. Gilbert, Numerical applications of a formalism for geophysical inverse problems, *Geophys. J. R. Ustr. Soc.*, **13** (1967), 247–276. ⇒237

[5] J. A. Bondy, U. S. R. Murty, *Graph Theory with Applications*, Springer, Macmillan, New York, 1976. 1976. ⇒232

[6] M. Biernacki, H. Pidek, C. Ryll-Nardzewski, Sur une inégalité entre des intégrales définies. (French) *Ann. Univ. Mariae Curie-Sklodowska. Sect. A.* **4** (1950), 1–4. ⇒237

[7] Z. Chen, Spectra of extended double cover graphs, *Czechoslovak J. Math.*, **54** (2004), 1077–1082. ⇒223

[8] J. B. Diaz, F. T. Metcalf, Stronger forms of a class of inequalities of G. Pólya–G. Szegö, and L.V. Kantorovich, *Bull. Amer. Math. Soc.*, **69** (1963), 415–418. ⇒234

[9] K. Ch. Das, K. Xu, I. Gutman, On Zagreb and Harary Indices, *MATCH Commun. Math. Comput. Chem.*, **70** (2013), 301–314 ⇒231, 232

[10] K. Ch. Das, B. Zhou, N. Trinajstić, Bounds on Harary index, *J. Math. Chem.*, **46** (2009), 1377–1393. ⇒231, 233

[11] P. J. Davis, *Circulant Matrices*, John Wiley and Sons, New York, 1979. ⇒222

[12] I. Gutman, The energy of a graph, *Ber. Math-Statist. Sekt. Forschungsz. Graz* **103** (1978), 1–22. ⇒219

[13] I. Gutman, *The energy of a graph: old and new results*, Algebraic Combinatorics and Applications, A. Betten, A. Kohnert, R. Laue and A. Wassermann, eds., Springer, Berlin, (2001), pp. 196–211. ⇒219

[14] I. Gutman, B. Zhou, Laplacian energy of a graph, *Linear Algebra Appl.*, **414** (2006), 29–37. ⇒219

[15] A. Ilić, G. Yu, L. Feng, The Harary index of trees, *Utilitas Math.*, **87** (2012), 21–32. ⇒231

[16] G. Indulal, R. Balakrishnan, Distance spectrum of Indu-Bala product of graphs, *AKCE Int. J. Graphs Combin.*, **13** (2016), 230–234. ⇒224

[17] O. Ivanciuć, T.S. Balaban, A.T. Balaban, Reciprocal distance matrix, related local vertex invariants and topological indices, *J. Math. Chem.* **12** (1993), 309–318. ⇒220, 221

[18] N. Ozeki, On the estimation of inequalities by maximum and minimum values, *J. College Arts Sci. Chiba Univ.*, **5**(2) (1968), 199–203. ⇒234

[19] D. Plavšić, S. Nikolić, N. Trinajstić, Z. Mihalić, On the Harary index for the characterization of chemical graphs, *J. Math. Chem.*, **12** (1993), 235–250. ⇒221

[20] G. Pólya, G. Szegö, *Problems and Theorems in Analysis, Series, Integral Calculus, Theory of Functions* (*Classics in Mathematics*), Springer, Berlin, 1972. ⇒ 234

[21] M. R. Rajesh Kanna, B. N. Dharmendra, G. Sridhara, Minimum covering distance energy of a graph, *Appl. Math. Sci.*, **7**(11) (2013), 5525–5536. ⇒219

[22] M. R. Rajesh Kanna, B. N. Dharmendra, R. Pradeep Kumar, Minimum covering Harary energy of a graph, *Int. J. Pure. Appl. Math.*, **90**(3) (2014), 371–385. ⇒ 219, 220, 229

[23] H. Wang, H. Hua, Note on Laplacian energy of graphs, *MATCH Commun. Math. Comput. Chem.*, **59** (2008), 373–380. ⇒219

[24] B. Zhou, I. Gutman, On Laplacian energy of graphs, *MATCH Commun. Math. Comput. Chem.*, **57** (2007), 211–220. ⇒219

Received: August 8, 2018 • Revised: November 11, 2018

Acta Univ. Sapientiae, Informatica **10,** 2 (2018) 241–256

DOI: 10.2478/ausi-2018-0012

Statistical complexity of the quasiperiodical damped systems

Ágnes FÜLÖP
Loránd Eötvös University
Faculty of Informatics
Budapest, Hungary
email: fulop@caesar.elte.hu

Abstract. We consider the concept of statistical complexity to write the quasiperiodical damped systems applying the snapshot attractors. This allows us to understand the behaviour of these dynamical systems by the probability distribution of the time series making a difference between the regular, random and structural complexity on finite measurements. We interpreted the statistical complexity on snapshot attractor and determined it on the quasiperiodical forced pendulum.

1 Introduction

There is no universal definition of complexity in natural sciences. In the last two decades several complexity measures have been introduced, which contain various aspects of complex systems. We mention some ones as algorithmic complexity (Kolmogorov) [17], amount of information about the optimal predict the future corresponding to the expected past (Crutchfield, Young) [8], (Boffetta, Cencini, Falconi, Vulpiani) [6], complexity of finite sequence (Lempel, Ziv) [22].

Computing Classification System 1998: F.1.3
Mathematics Subject Classification 2010: 68U20
Key words and phrases: statistical complexity, Shannon entropy, chaos, snapshot attractor, on-off intermittency

An important contribution to this issue due to P. Grassberger [13], he studied the pattern-generation by the dynamics of a system and introduced the effective entropy considering the mixture of order and disorder, regularity and randomness, because the most complex situation is neither the one with highest Shannon information $\mathcal{S}$ (random structure) nor the one with lowest $\mathcal{S}$ (ordered structures).

We negotiate the statistical complexity in this article, which allows a description of a finite measured series to consider more complicated dynamical structures which was published in the article [23] (1995). It was widely used in the chaotic regim [9], biology [32], symbolic sequences [1], pseudorandom bit generator [12], earthquake [24], the number system [11].

The entropy appoints the direction of flow. The complexity determines the inner structure of the dynamical system. The statistical approaches are easier to implement than solving equations of motion and they offer the only way of dealing with otherwise intractable problems.

We study the complexity of the quasiperiodic driven dynamical systems considering the snapshot attractor [30] on the set of points, which are determined by the Poencaré section. This object corresponds to a given time moment, which contains the points of the trajectories ensemble. These orbits were initialized in the past and the time dependent behaviour was determined by the same equation of motion.

We determined the numerical approximation of the aperiodic driven pendulum, which reflects the behaviour of complexity on the snapshot attractor. This system shows on-off intermittency [20] near to the axis (L=0), due to the fluctuation of the maximal Lyapunov exponent. The maximal Lyapunov exponent linearly intersects the axis, therefore it can be seen, that the system has scaling behaviour [19].

The structure of the article contains the next parts: In the Section 2 we introduce the idea of complexity accordingly the measure of entropy and disequilibrium. We discuss the statistical complexity is extended to generalized complexity considering the Tsallis, Wooters, Rényi entropy and the Kullbac-Shannon, Kullbac-Tsallis, Kulback-Rényi divergency. We explained the quasi periodical motion by snapshot attractor, which disposes the on-off intermittency between chaotic and nonchaotic condition and this system provides the scaling behaviour in the Section 3. A numerical approximation of the aperiodic forced system is illustrated by the quasiperiodic driven pendulum comparing the complexity and the dissipation rate, which due to the on-off intermittency (Section 4).

2 Complexity

We investigate the statistical complexity, which is based on the effective entropy by P. Grassberger [13] and the main idea is published by R. López-Ruiz, H.L. Manchini, X. Calbet (LMC) [23, 25, 3, 7].

The expanded definition of the statistical complexity so called generalized statistical complexity measures were introduced by M.T. Martin, A. Plastino, O.A. Rosso (2006) [26], which apply various kinds of entropy and disequilibrium measure [18].

We use the notation of a measured sequence by the article [10]. The time series of the ensemble is denoted by $y_{1,j}, \dots, y_{n,j}$, where $y_{i,j}$ means the measurement of the quantity y at time $t_i = t_0 + i\Delta t$, the time interval $\Delta t > 0 \in \mathbb{R}$ and j ($1 \le j \le m$) assigned a number of the trajectory in the ensemble. The unit of the time interval Δt equals to a constant in this description. The $\underline{x}^{(n)}$ indicates the trajectory of length n in $\mathbb{R}^d$, which means a time series of the measurement. The kth point of the orbit of length n in the manifold is written by $x_{k,j}^{(n)}$, where ($k = 1, \dots, n$), and j means the number of trajectory ($j = 1, \dots, m$) in the manifold. We will research the sequent of $x_{1,j}^{(n)}, x_{2,j}^{(n)}, \dots, x_{n,j}^{(n)}$ as a time series over the jth trajectory. Let us choose this time development quantity of the ensemble at a given moment $i = t'$, then we determine the probability distribution of the $x_{t',j}^{(n)} (j = 1, \dots, m)$ over the points of the trajectories of the manifold.

We rephrase this notation by the symbolic dynamics, because the concept of complexity is much more universal idea than this question. We distinguish M different value of the measurements. The points of the trajectories of length n $x_{i,j}^{(n)}$ ($1 \le i \le n$) ($1 \le j \le m$) are noted by the symbol $o_{i,j}$, which is chosen from the set $\{1, \dots, M\}$. Then the jth path of length n in the ensemble corresponds to $O_j^{(n)} = (o_{1,j}, o_{2,j}, \dots, o_{n,j})$. A series $O_j^{(n)}$ ($1 \le j \le m$) occurs with probability $P(O_j^{(n)})$ along a sequent of length N ($n \le N$).

2.1 Statistical complexity

The statistical complexity is well applicable concept characterizing finite measurement sequences with its probability distribution. This allows a statistical approximation of the measured quantities. We apply the basic article [23] to introduce this idea.

We suppose that there are N various symbol series of length n ($o_{1,j}, \dots, o_{N,j}$) ($j = 1, \dots, m$) in the ensemble, then these series dispose the set of discrete

probability distribution $P \equiv \{p_{1,j}, \ldots, p_{N,j}\}$, where $p_{i,j} := P(o_{i,j})$ ($\sum_{i=1}^{N} p_{i,j} = 1$) ($1 \leq j \leq m$) and $p_{i,j} > 0$ for all i.

The statistical complexity measures contains two compositions: (i) entropy $\mathcal{H}$ and (ii) disequilibrium $\mathcal{D}$ i.e. distances in probability-space. It is introduced by the information theory, where the Shannon entropy assigns the gain of the information storage and the disequilibrium features the distance from uniform distribution. So the LMC measure gives a simultaneous quantification of randomness and correlation structures in the systems.

2.1.1 Measure of entropy and disequilibrium

Information measure Information measure $\mathcal{I}[P]$ i.e. the uncertainty can be described by the probability distribution $P = \{p_j, j = 1, \ldots, N\}$, with N the number of possible states of the system ($\sum_{j=1}^{N} p_j = 1$). In the information theory we define the quantity of disorder $\mathcal{H}$ for a given probability distribution P corresponding to the information measure $\mathcal{I}[P]$ in the next term:

$$\mathcal{S}[P] = \mathcal{I}[P]/\mathcal{I}_{max}[P_e], \tag{1}$$

where $0 \leq \mathcal{H} \leq 1$, $\mathcal{I}_{max}$ means the maximal value of $\mathcal{I}$, and P_e is the uniform probability distribution. Let us consider the Shannon-Kinchin paradigm then $\mathcal{I}$ can be defined as a term of entropies. The statistical complexity was introduced by the Shannon entropy [33], therefore we will investigate this form in a finite system:

$$\mathcal{S} = -\sum_{i=1}^{N} p_i \log p_i. \tag{2}$$

This quantity approximately equals to zero $\mathcal{S} \sim 0$, if the symbol sequence $O_{j_c}^{(n)}$ has a high probability ($p_c \sim 1$) and other $O_j^{(n)}$ has very small probability ($p_c \sim 0$). In the range of entropy the maximal values $\mathcal{S}_{max}$ corresponds to the uniform probability distribution $p_e = \{1/N, 1/N, \ldots, 1/N\}$, which means the the equal probability symbol sequence $O_{j_e}^{(n)}$, which leads to the maximum value of information. The normalized quantity $\mathcal{H}$ derives from $\mathcal{H} = \mathcal{S}/\mathcal{S}_{max}$, than $0 \leq \mathcal{H} \leq 1$, where $\mathcal{S}_{max} = \log N$.

Disequilibrium measure We introduce the function of disequilibrium $\mathcal{D}$ on the probability distribution $\{p_j : j = 1 \ldots, N\}$. The LMC uses some distance $\mathcal{D}$ of a given P compared to the uniform distribution P_e in the states of the

system [23]. Therefore we investigate the disequilibrium by a distance-form:

$$\mathcal{Q}[P] = \mathcal{Q}_0 \cdot \mathcal{D}[P, P_e], \tag{3}$$

where $\mathcal{Q}_0$ is a normalization constant ($0 \leq \mathcal{Q}_0 \leq 1$), which equals to the inverse of the maximum possible value of the distance $\mathcal{D}[P, P_e]$. The maximum distance is obtained, when one of the components of P i.e. p_l equals to one and the remaining elements equal to zero. The disequilibrium $\mathcal{Q}$ differs from zero, if there exist preferred states among the accessible ones, i.e. this quantity features the systems' architecture.

In the original definition of the statistical complexity the Euclidean norm ($\mathbb{R}^N$) have been used i.e. the quadratic distances from the probability distribution of each symbol sequences $P(O_j^{(n)})$ ($1 \leq j \leq m$) to the equal probability $P(O_{j_e}^{(n)})$. This choice for the distance $\mathcal{D}$ is written:

$$\mathcal{D}[P, P_e] = \sum_{i=1}^{N} (p_i - p_e)^2, \quad \text{where} \quad p_e = \frac{1}{N}. \tag{4}$$

In the range of this quantity becomes maximum, when the disequilibrium achieves prevalent symbol sequences $O_{j_c}^{(n)}$ with $p_c \sim 1$ and $\mathcal{D}_c \to 1$ for N is growing. Otherwise the disequilibrium equals to zero approximately $\mathcal{D} \sim 0$ for $p_i \sim 1/N$. The value of the $\mathcal{D}$ changes between these extrema corresponding to advanced probability distribution. The normalized factor equals to $\mathcal{Q}_0 = \frac{N}{N-1}$.

2.1.2 Measure of statistical complexity

The whole complexity concept includes the functional product of disorder $\mathcal{H}$ and disequilibrium $\mathcal{D}$, which based on the various probability distribution corresponding to sequent of the advanced quantity. This shows the transition between the information stored in the system and its disequilibrium. The statistical complexity $\mathcal{C}$ is introduced by the published article of LMC [23]:

$$\mathcal{C} = \mathcal{H} \cdot \mathcal{D} = -\left(\sum_{i=1}^{N} p_i \log p_i\right)\left(\sum_{i=1}^{N}\left(p_i - \frac{1}{N}\right)^2\right). \tag{5}$$

The value $\mathcal{C} \in \mathbb{R}^+$. The normalized $\mathcal{C}$ can be described as follows $\overline{\mathcal{C}} = \overline{\mathcal{H}} \cdot \overline{\mathcal{D}} = (\mathcal{H}/\log N)(\mathcal{D} \cdot (N/(N-1)))$. The range of complexity measure is finite and limiting between $\mathcal{C}_{min}$ and $\mathcal{C}_{max}$, but $\mathcal{H}$ is not necessarily a unique function.

We applied finite system, therefore the statistical complexity disposes the scaling behaviour. A new set of symbol sequences $O_j^{(n)}$ turns out at each scale of measurement, which provides an advanced probability distribution $P(O_j^{(n)})$ so the value of complexity becomes different.

Three basic cases are distinguished of the statistical complexity: (a) this is monotonous increasing in the function of entropy (b) it corresponds to a convex function, which contains a maximal $\mathcal{C}_{max}$ at the probability distribution p_e and the minimum $\mathcal{C}_{min}$ appears, where the $\mathcal{H} = 0$ i.e. total order and $\mathcal{H} = 1$ and third kind is (c) the monotonous decreasing with increasing entropy [26].

There are two extremist situations of complexity $\mathcal{C}$ depending on entropy $\mathcal{H}$. On the one hand each set of series assigned to each set of symbol sequence $O_j^{(n)}$, which has the same probability distribution. All of them contribute to the information stored in equal measure as the ideal gas [21]. On the other hand, if we study an object, which features some symmetries properties and distance, then this system can be written by minimal information as mineral or symmetry in quantum mechanics.

2.1.3 Generalized statistical complexity

Generalized entropy We expend the concept of classical statistical complexity to different measures of the entropy and disequilibrium. Tsallis introduced a generalisation of the Shannon-Boltzmann-Gibbs entropic measure [35]:

$$\mathcal{S}_q^{(T)}[P] = \frac{1}{(q-1)} \sum_{j=1}^{N} \left[p_j - (p_j)^q\right], \tag{6}$$

where q real number. Rényi suggested a definition of entropy for discrete probability distribution in 1950s years [28]:

$$\mathcal{S}_q^{(R)}[P] = \frac{1}{(1-q)} \ln \left\{ \sum_{j=1}^{N} (p_j)^q \right\}. \tag{7}$$

Then the generalized entropy $\mathcal{S}_q^{(\kappa)}$ denotes $\kappa = S, T, R$ Shannon,Tsallis and Rényi entropy.

$$\mathcal{H}_q^{(\kappa)}[P] = \mathcal{S}_q^{(\kappa)}[P]/\mathcal{S}_{max}^{(\kappa)}, \tag{8}$$

where $\mathcal{S}_{max}^{(\kappa)}$ means the maximum value of the information measure, which corresponds the uniform probability distribution. The maximal value of Shannon

and Rényi entropy correspond to $\mathcal{S}^{(S)}_{max} = \mathcal{S}^{(R)}_{max} = \ln N$ and Tsallis entropy involves $\mathcal{S}^{(T)}_{max} = \frac{1-N^{1-q}}{q-1}$ for $q \in (0,1) \cup (1,\infty)$.

Generalized disequilibrium The Euclidean distances was investigated in the LMC. We introduce the generalized disequilibrium $\mathcal{D}$. The Wootters distance was applied for two probability distributions [36], which can be used in the quantum mechanic:

$$\mathcal{D}_W[P_1, p_2] = \cos^{-1}\left\{\sum_{j=1}^{N}(p_j^{(1)})^{1/2}(p_j^{(2)})^{1/2}\right\}. \tag{9}$$

Consider two divergence-classes, which were published by Basseville [4]. The first class contains the divergence which is defined by the relative entropies. The second class consists of the divergence, which was introduced as the difference of the entropies. The Kullbac-Shannon expression following

$$\mathcal{D}_{\mathcal{K}^S}[P, P_e] = \mathcal{K}^{(\mathcal{S})}[P|P_e] = \mathcal{S}_1^{(\mathcal{S})}[P_e] - \mathcal{S}_1^{(\mathcal{S})}[P]. \tag{10}$$

The Kullback-Tsallis entropy is introduced:

$$\mathcal{D}_{\mathcal{K}_q^T}[P, P_e] = \mathcal{K}_q^{(T)}[P|P_e] = N^{q-1}(\mathcal{S}_q^{(T)}[P_e] - \mathcal{S}_q^{(T)}[P]). \tag{11}$$

The Kulback-Rényi etropy following

$$\mathcal{D}_{\mathcal{K}_q^R}[P, P_e] = \mathcal{K}_q^{(R)}[P|P_e] = (\mathcal{S}_q^{(R)}[P_e] - \mathcal{S}_q^{(R)}[P]). \tag{12}$$

Then the generalized disequilibrium denoted by this form:

$$\mathcal{Q}_q^{(\nu)}[P] = \mathcal{Q}_0^{(\nu)}\mathcal{D}_\nu[P, P_e], \tag{13}$$

where $\nu = E, W, K, K_q$ and $\mathcal{Q}_0^{(\nu)}$ normalization constant ($0 \le \mathcal{Q}_q^{(\nu)} \le 1$), and these are:

$$\mathcal{Q}_0^{(E)} = \tfrac{N}{N-1}, \quad \mathcal{Q}_0^{(W)} = 1/\cos^{-1}\left\{\left(\tfrac{1}{N}\right)^{1/2}\right\},$$
$$\mathcal{Q}_0^{\mathcal{K}_q^R} = \tfrac{1}{\ln N} \quad \mathcal{Q}_0^{\mathcal{K}_q^T} = \tfrac{q-1}{N^{(q-1)}-1}.$$

Generalized statistical complexity This quantity is defined following

$$\mathcal{C}_{\nu,q}^{(K)}[P] = \mathcal{Q}_q^{(\nu)}[P] \cdot \mathcal{H}_q^{(K)}[P], \tag{14}$$

where $K = S, R, T$ for fixed q and the index $\nu = E, W, K_q$ means the disequilibrium with appropriated distance measures. This term (14) is a family of the statistical complexity corresponding to functional product form $\mathcal{C} = \mathcal{H} \cdot \mathcal{Q}$. We notice that the entropic difference $\mathcal{S}[P_1] - \mathcal{S}[P_2]$ does not specify the information gain or divergence, because this quantity is not necessary positive definit. This lead to the Jensen divergence.

Shinner, Davidson and Landsberg (SDL) published a term for the statistical complexity [34], this term is expressed for $\nu = K, K_q$:

$$\mathcal{C}^{(\kappa)}_{K_q}[P] = (1 - \mathcal{H}^{(\kappa)}_q[P]) \cdot \mathcal{H}^{(\kappa)}_q[P]. \tag{15}$$

We get the LMC statistical complexity at $\kappa = S$, $q = 1$ so $\mathcal{C}_{LMC} = \mathcal{C}^{(S)}_{K,1}$.

2.2 Time evolution

In statistical physics we study the isolated systems, which are characterized by initial, arbitrary and discrete probability distribution. The uniform distribution P_e develops during the evolution towards equilibrium. Then we can research the time evolution of the LMC i.e. $\mathcal{C}$ versus time t graph. Thanks to the second law of thermodynamics the entropy grows monotonically with time ($d\mathcal{H}/dt \geq 0$) in isolated system. Therefore $\mathcal{H}$ behaves as an arrow of time, so we can study the figure of $\mathcal{C}$ versus $\mathcal{H}$ as the time evolution of the LMC, thus the normalised entropy-axis can be substituted by the time-axis. This picture $\mathcal{H} \times \mathcal{C}$ can be utilized to research the changes in the dynamics of a system, which derives from the modulated parameters.

3 Driven system

In both experimental and theoretical physics, periodically excited nonlinear systems play important role. A typical case of these systems is described by this equation:

$$\frac{d^2\Theta}{dt^2} + \nu\frac{d\Theta}{dt} + \sin\Theta = f(t), \tag{16}$$

where the damped forcing f(t) is periodic in time, for example:

$$f(t) = K + V\cos(\omega t). \tag{17}$$

Such equations are used in many cases of physical research, as forced damped pendulum, the Stewart-McCumber model of the current-driven Josephson

junction [2] and simple phenomenological model of sliding charge-density waves [5]. The nonlinear dynamical model can be characterized by strange attractor, period doubling cascades, crises, intermittency, fractal basin boundaries etc. These equations are intensively used in low dimensional chaotic dynamics research.

Periodic excitation is extended to examine aperiodic cases, when $f(t)$ is quasiperiodic for example:

$$f(t) = K + V[\cos(\omega_1 t) + \cos(\omega_2 t)], \tag{18}$$

where ω_1 and ω_2 are the incommensurate frequencies. It appears in the transition from quasiperiodic to chaos inside an electronic Josphson-junction simulator driven by two independent sources [5]. It is applied on the experiments inside an electron-hole plasma in germanium excited by two frequencies quasiperiodic external perturbations they observed stable three frequencies mode locking, and chaos [14, 15].

These are presented with quasiperiodic systems by two incommensurate frequencies. We consider the following quasiperiodically forced damped pendulum [29]:

$$\frac{d^2\Theta}{dt^2} + \nu\frac{d\Theta}{dt} + \sin\Theta = K + V[\cos(\omega_1 t) + \cos(\omega_2 t)], \tag{19}$$

where Θ is an angle of pendulum with the vertical axis, ν is the dissipation rate, K is a constant, V is the forcing amplitude and ω_1 and ω_2 are the incommensurate frequencies. Let us investigate new variables, $t \to \nu t$ and $\phi = \Theta + \frac{\pi}{2}$. Then the equation (19) becomes:

$$\frac{1}{p}\frac{d^2\phi}{dt^2} + \frac{d\phi}{dt} - \cos\phi = K + V[\cos(\omega_1 t) + \cos(\omega_2 t)], \tag{20}$$

where $p = \nu^2$ is a new parameter, and ω_1 and ω_2 are rescaled as follows: $\omega_1 \to \omega_1\nu$ and $\omega_2 \to \omega_2\nu$. In the expressions of the dynamical variables ϕ,$v \equiv \frac{d\phi}{dt}$ and $z \equiv \omega_2 t$, then we have

$$\left.\begin{array}{rcl} \frac{d\phi}{dt} & = & v, \\ \frac{dv}{dt} & = & p\left\{K + V\left[\cos\left(\frac{\omega_1}{\omega_2}z\right) + \cos z\right] + \cos\phi - \nu\right\}, \\ \frac{dz}{dt} & = & \omega_2. \end{array}\right\} \tag{21}$$

The equation (21) contains rich dynamical behaviour [31]. This system shows a special behaviour in some range of parameter space. Therefore we apply the

snapshot attractor, because this structure reflects the properties of a dynamical systems at a given moment.

The dynamic of the quasiperiodic damped pendulum is characterised by the sign of maximal Lyapunov exponent which changes near to the axis (L=0), because this system has finite fluctuation. The Lyapunov exponent becomes to negative, then the system contracts on the nonchaotic side($L \leq 0$). Otherside the Lypunov exponent turns into positive, then the expansion characterizes the dynamical behaviour on the chaotic side ($L \geq 0$). Therefore the collective properties of the orbits can be studied near to the transition. This is a special behaviour of this model which is called on-off intermittency of the snapshot attractor. The trajectories spend stretches of time expanding (leading to nonzero-size snapshot attractor), yet there are also long time during the trajectories experience contraction, resulting extremely small-size of snapshot attractor. Then the time dependent size of snapshot attractor can be written by the dispersion rate [16]:

$$S(t) = \left(\frac{1}{N} \sum_{i=1}^{N} \left\{ [\phi_i(t) - < \phi(t) >]^2 + [\nu_i(t) - < \nu(t) >]^2 \right\} \right)^{1/2}, \qquad (22)$$

where N is the number of points on the snapshot attractor, $[\phi(t), \nu(t)]$ defines the geometric center of these points at a given time: $< \phi(t) >= \frac{1}{N} \sum_{i=1}^{N} \phi_i(t)$ and $< \nu(t) >= \frac{1}{N} \sum_{i=1}^{N} \nu_i(t)$.

The time averaged size of the snapshot attractor on the chaotic side near to the transition is defined as $< S(t) >= \lim_{T \to \infty} \int_0^T S(t) dt$ which obeys the following scaling relation:

$$< S(t) > \sim L \sim |p - p_c|, \qquad (23)$$

where p_c means the $p \geq p_c$ ($L \leq 0$) and $p \leq p_c$ ($L \geq 0$). This scaling behaviour of the transition to chaos was published in quasiperiodically driven dynamical systems [19] i.e. a route of chaos was investigated, where the largest Lyapunov exponent passes through zero linearly near the transition to chaos. Because the orbits burst out to separate from each others during the expansion time intervals [27], and the trajectories merge all together during the contraction time interval therefore the size of the chaotic snapshot attractor changes widely in time near to the transition in an intermittent fashion. The average size of the snapshot attractor scales linearly with a parameter similarly as the average interval between the bursts also scales linearly with parameter during transition (23).

4 Numerical approximation

In this Section we consider the statistical complexity of the quasiperiodic driven systems, which is represented by the aperiodic forced pendulum (19). We calculated the Poencaré section ($z = 0$) of the time dependent model (21), which contains a snapshot attractor (Figure 1) at the parameter values $V = 0.55, p = 0.6, K = 0.8$ $\omega_1 = (\sqrt{5}-1)/2$, $\omega_2 = 1.0$. The initial values of the orbits are chosen by uniform distribution in a small volume $<< 10^{-6}$ in the phase space. We show the predictability of the intermittency between chaotic and nonchaotic regions.

Statistical complexity We determined the statistical complexity of this system which was introduced by the quantity of information theory, where the entropy and the disequilibrium depend on the probability distribution (Section 2.1.2). The snapshot attractor is written by the ensemble of trajectories instead of the long orbit N'. Therefore we redefine the probability distribution of the manifold at a time instant t'.

The ensemble of the snapshot attractor contains the $x^{(n)}_{1,j}, x^{(n)}_{2,j}, \ldots, x^{(n)}_{n,j}$ points of the trajectories of length n ($1 \leq j \leq m$). Therefore the $x^{(n)}_{t',1}, x^{(n)}_{t',2}, \ldots, x^{(n)}_{t',m}$ series corresponds to $\{p_1, p_2 \ldots, p_m\}$ probability distribution, where $p_j := P(x^{(n)}_{t',j})$ ($j = 1, \ldots, m$).

The entropy $\mathcal{H}$, disequilibrium $\mathcal{D}$ and statistical complexity $\mathcal{C}$ can be determined by appropriate measures using the term of LMC complexity (5). The structure of the complexity is plotted in the $\mathcal{C} \times \mathcal{H} \times \mathcal{D}$ space (Figure 4), where parameter p changes between 0 and 1. The behaviour of complexity $\mathcal{C}$ monotone decreasing, so this corresponds to class (c) over the parameter range [0,1]. In a small interval at $p \simeq 0.8$ the shape of complexity formed convex curve (class (b)).

On-off intermittency The quasiperiodic damped pendulum provides on-off intermittency in a special values of parameter, where Lyapunov exponent has a finite fluctuation around axis of $L = 0$ as it was detailed in the Section (3). The volume of the snapshot attractor is widely changing near to axis ($L = 0$), therefore we applied the dispersion rate $S(t)$ (22), which scales by the maximal Lyapunov exponent (23). The structure of the complexity $\mathcal{C}(p)$ shows local maximums (Figure 3) similarly as the average of the dispersion $< S(t) >$ over the parameter space p (Figure 2). The complexity reflects the behaviour of the pendulum i.e. the on-off intermittency. Then the probability

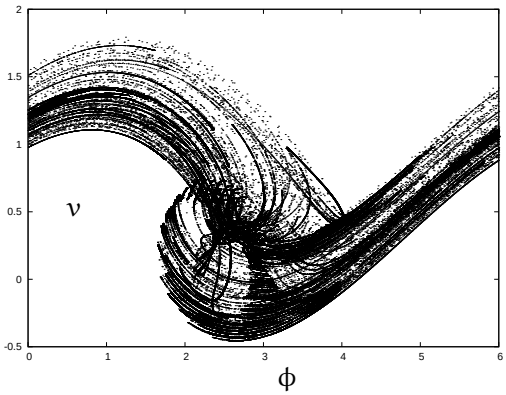

Figure 1: The snapshot attractor of the quasiperiodic driven pendulum at parameters $K = 0.8$, $V = 0.55$, $p = 0.6$.

distribution of the snapshot attractor i.e. dispersion of the trajectories' points in the manifold at a given time instant t' shows similar behaviour as the affect of the on-off intermittency.

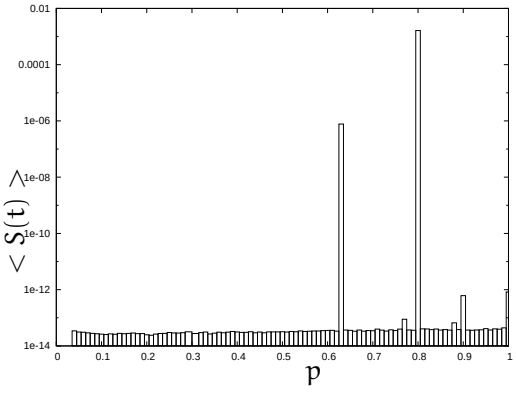

Figure 2: The $< S(t) >$ depends on parameter p in logarithm scale.

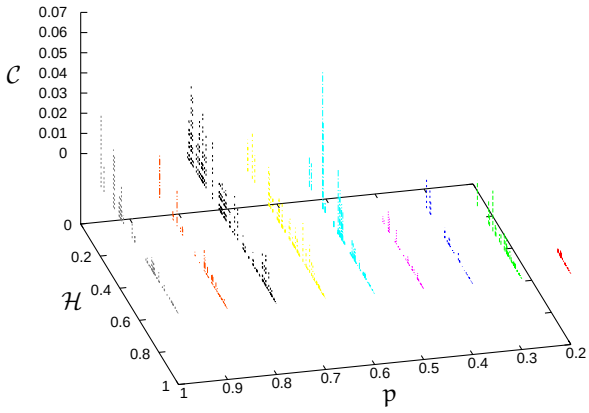

Figure 3: The complexity $\mathcal{C}$ depends on the entropy $\mathcal{S}$ and parameter $\mathfrak{p}$.

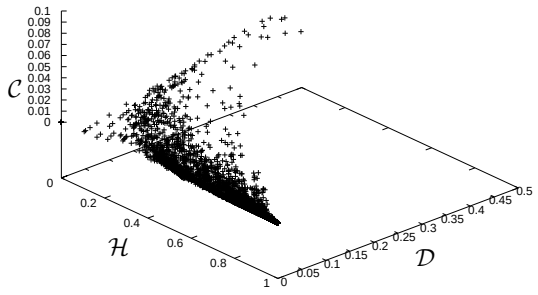

Figure 4: The complexity $\mathcal{C}$ depends on the entropy $\mathcal{H}$ and disequilibrium $\mathcal{D}$ for different parameter $\mathfrak{p}$ ($0 \leq \mathfrak{p} \leq 1$).

5 Conclusion

The inner structure of statistical complexity is determined in a quasiperiodical driven system at a given scale. The effect of the on-off intermittency appears in complexity of the aperiodic damped system, which allows the predictability by the by the distribution probability of the snapshot attractor.

References

[1] C. Adami, N.T. Cerf, Physical complexity of symbolic sequences, *Physica D* **137** (2000) 62–69. ⇒242

[2] A. Aiello, A. Barone, G. A. Ovsyannikov, Influence of nonlinear conductance and cos φ term on the onset of chaos in Josephson junctions, *Phys. Rev. B* **30** (1984) 456. ⇒249

[3] C. Anteneodo, A.R. Plastino, Some features of the Lpez-Ruiz-Manchini-Calbet (LMC) statistical measure of complexity, *Physics Letters A* **223** (1996) 348–354. ⇒243

[4] M. Basseville, Information: Entropies, Divergences et Mayennes, (IRISA) Publication Interne **1020** (1996) (Campus Universitaire de Beaulieu, 35042 Rennes Cedex, France). ⇒247

[5] T. Bhor, P. Bak, M. H. Jensen, Transition to chaos by interaction of resonances in dissipative systems. II. Josephson junctions, charge-density waves, and standard maps, *Phys. Rev. A* **30** (1984) 1970. ⇒249

[6] G. Boffetta, M. Cencini, M. Falcioni, A. Vulpiani, Predictability: a way to characterize complexity, *Phys. Reports* **356**(2002) 367–474. ⇒241

[7] X. Calbet, R. López-Ruiz, Tendency towards maximum complexity in a nonequlibrium isolated system, *Phys. Rev. E* **63** 066116. ⇒243

[8] J. P. Crutchfield, K. Young, Inferring statistical complexity, *Phys. Rev. Lett.* **63** (1989) 105. ⇒241

[9] G.L. Ferri, F. Pennini, A. Plastino, LMC-complexity and various chaotic regime, *Physics Letters A* **373** (2009) 2210–2214. ⇒242

[10] Á. Fülöp, Estimation of the Kolmogorov entropy in the generalized number system, *Annales Univ. Sci. Budap est Sect. Comp.* **40** (2013) 245–256. ⇒243

[11] Á. Fülöp, Statistical complexity and generalized number system, *Acta Univ. Sapientiae, Informatica* **6** (2) (2014) 230–251, ⇒242

[12] C.M. Gonzalez, H.A Larrondo, O.A. Rosso,Statistical complexity measure of pseudorandom bit generators, *Physica A* **354** (2005) 281. ⇒242

[13] P. Grassberger, Toward a Quantitative Theory of Self-Generated Complexity, *Int. Journ. Theor. Phys.* **25** (1988) 907–938. ⇒242, 243

[14] D-R. He, W. J. Yeh, Y. H. Kao,Transition from quasiperiodicity to chaos in a Josephson-junction analog, Phys. Rev. B **30** (1984) 172. ⇒249

[15] G. A. Held, C. Jeffries, Quasiperiodic Transitions to Chaos of Instabilities in an Electron-Hole Plasma Excited by ac Perturbations at One and at Two Frequencies, *Phys. Rev. Lett.* **56** (1986) 1183. ⇒249

[16] I-A. Khovanov, N-A. Khovanova, P-V-E. McClintock,V-S. Anishchenko, The effect of noise on the strange nonchaotic attractors, *Phys. Lett. A* **268** (2000) 315–322. ⇒250

[17] A.N. Kolmogorov, Entropy per unit time as a metric invariant of automorphism, *Doklady of Russian Academy of Sciences*, **124** (1959) 754–755. ⇒241

[18] A-M. Kowalski, M-T. Martin, A. Plastino, O-A. Rosso, M. Casas, Distances in Probability space and the statistical complexity setup, *Entropy* **13** (2011) 1055–1075. ⇒243

[19] Y-C. Lai, U. Feudel, C. Grebogi, Scaling behaviour of transition to chaos in quasiperiodically driven dynamical systems *Phys. Rev. E*, **54** (6) (1996) 6070–6073. ⇒242, 250

[20] Y-C. Lai, C. Grebogi, Intermingled basins and two-state on-off intermittency, *Phys. Rev. E* **52** (4) (1995) R3313–R3316. ⇒242

[21] P.T. Landsberg, J.S. Shiner, Disorder and complexity in an ideal non-equilibrium Fermi gas, *Phys. Lett. A***245** (1998) 228. ⇒246

[22] A. Lempel, J. Ziv On the complexity of finite sequences, *IEEE Trans. Inform Theory* **22** (1976) 75–81. ⇒241

[23] R. López-Ruiz, H.L. Mancini, X. Calbet, A statistical measure of complexity, *Phys. Letters A* **209** (1995) 321–326. ⇒242, 243, 245

[24] M. Lovallo, V. Lapenna, L. Telesca, Transitionmatrix analysis of earthquake magnitude sequences *Chaos,Soliton and Fractals* **24** (1) (2005) 33–43. ⇒242

[25] M.T. Martin, A. Plastino, O.A. Rosso, Statistical complexity and disequilibrium, *Physics Letters A* **311** (2003) 126–132. ⇒243

[26] M-T. Martin, A. Plastino, O.A. Rosso, Generalized statistical complexity measures: Geometrical and analytical properties, *Physica A* **369** (2006) 439–462. ⇒243, 246

[27] N. Platt, E-A. Spiegel, C. Tresser, On-off intermittency: a mechanism for bursting, *Phys. Rev. Lett.***70** (3) (1993) 279–282. ⇒250

[28] A. Rényi, Probability Theory *(Akadémia Kiadó, Budapest 1970).* ⇒246

[29] F. J. Romeiras, A. Bondenson, E. Ott, T. M. Antonsen, C. Grebogi, Quasi-Periodically forced dynamic-systems with strange nonchaotic attractors *Physica D* **26** (1987) 277. ⇒249

[30] F.J. Romeiras, C. Grebogi, E. Ott, Multifractal properties of snapshot attractors of random maps, *Phys. Rev A* **41** (2) (1990) 784–799. ⇒242

[31] F-J. Romeiras. E. Ott, Strange nonchaotic attractors of the damped pendulum with quasiperiodic forcing, *Phys. Rev. A* **35** (10) (1987)4404–4413. ⇒249

[32] P.T. Saunders, and M.W. Ho, On the increase in complexity in Evolution II. The relativity of complexity and the principle of minimum increase, *Journ. of Theor. Biol.* **90** (1981) 515. ⇒242

[33] C.E. Shannon, The Mathematical Theory of Communication, *Bell System Technical Journal*, **27** (1948) 379-423, 623–656. ⇒244

[34] J-S. Shiner, M. Davison, P-T. Landsberg, Simple measure for complexity, *Phys. Rev. E* **59**(2)(1999)1459–1464. ⇒248
[35] C. Tsallis, Possible generalization of Boltzmann-Gibbs statistics, *J. Stat. Phys.* **52** (1988) 479. ⇒246
[36] W.K. Wooters, Statistical distance and Hilbert space, *Phys. Rev. D* **23** (1981) 357. ⇒247

Received: November 2, 2018 • Revised: December 4, 2018

Acta Univ. Sapientiae, Informatica **10,** 2 (2018) 257–267

DOI: 10.2478/ausi-2018-0013

On scores in tournaments

T. A. Naikoo
Department of Mathematics, Islamia College of
Science and Commerce,
Srinagar, Kashmir, India
email: tariqnaikoo@rediffmail.com

Abstract. A tournament is an orientation of a complete simple graph. The score of a vertex in a tournament is the outdegree of the vertex. In this paper, we obtain various results on the scores in tournaments.

1 Introduction

A tournament is an orientation of a complete simple graph. Let T be a tournament with vertex set $\{v_1, v_2, \ldots, v_n\}$. The score of a vertex v_i is defined as the outdegree of v_i and is denoted by s_{v_i}(or simply by s_i). Clearly $0 \leq s_i \leq n-1$ for all i, $1 \leq i \leq n$. The sequence $[s_1, s_2, \ldots, s_n]$ in non-decreasing order is called the score sequence of the tournament T. Several results on tournament scores can be seen in [21, 23]. The concept of scores in tournaments was extended to oriented graphs by Avery [1] and many results on oriented graph scores can be found in [19, 21, 22]. Pirzada et al. generalized score structure to other classes of digraphs and details can be seen in [17, 18]. Further score structure has been extended to hypertournaments, a generalization of tournaments [4, 5, 8, 9, 10, 11, 12, 13, 14, 15, 24].

The following result [6] gives necessary and sufficient conditions for a sequence of non-negative integers to be the score sequence of some tournament and this result is also known as Landau's theorem.

Computing Classification System 1998: G.2.2
Mathematics Subject Classification 2010: 05C20
Key words and phrases: Tournament, score, score sequence, Landau's theorem.

Theorem 1 (Landau [6]) *A sequence* $[s_1, s_2, \ldots, s_n]$ *of non-negative integers in non-decreasing order is a score sequence of some tournament if and only if*

$$\sum_{i=1}^{k} s_i \geq \frac{k(k-1)}{2}, \tag{1}$$

for $1 \leq k \leq n$ *with equality when* $k = n$.

More work for scores in tournaments can be found in [2, 3, 7, 16].

For any two distinct vertices u and v of a tournament T, we have one of the following possibilities.
(i) An arc directed from u to v, denoted by $u(1-0)v$.
(ii) An arc directed form v to u, denoted by $u(0-1)v$.

2 Main Results

Now, we obtain the following results.

Theorem 2 *Let* $[s_1, s_2, \ldots, s_n]$ *be the score sequence of a tournament. Then the lowest score of the tournament is zero if* $\sum_{i=1}^{n} s_i^2$ *is maximum.*

Proof. Let v_1 be the vertex of the tournament with lowest score s_1. We shall show that $s_1 = 0$.
Suppose on contrary $s_1 > 0$. Then there exists a vertex v_p with score s_p such that $v_1(1-0)v_p$. Since $s_p \geq s_1$, therefore there exists another vertex v_q with score s_q such that $v_p(1-0)v_q$.
Now, by changing the arcs $v_1(1-0)v_p$ and $v_p(1-0)v_q$ to $v_1(0-1)v_p$ and $v_p(0-1)v_q$ respectively we get a new score sequence $[t_1, t_2, \ldots, t_n]$ where $t_1 = s_1 - 1, t_q = s_q + 1, t_r = s_r$ for all $r,\ 2 \leq r \leq n$ with $r \neq q$. Then

$$\begin{aligned}
\sum_{i=1}^{n} t_i^2 &= \sum_{i=2, i\neq q}^{n} t_i^2 + t_1^2 + t_q^2 = \sum_{i=2, i\neq q}^{n} s_i^2 + (s_1 - 1)^2 + (s_q + 1)^2 \\
&= \sum_{i=2, i\neq q}^{n} s_i^2 + s_1^2 + 1 - 2s_1 + s_q^2 + 1 + 2s_q = \sum_{i=1}^{n} s_i^2 + 2(s_q - s_1 + 1) \\
&> \sum_{i=1}^{n} s_i^2,
\end{aligned}$$

since $s_q \geq s_1$. This is a contradiction as $\sum_{i=1}^{n} s_i^2$ was assumed to be maximum. Hence the result follows. □

Theorem 3 *Let* $[s_1, s_2, \ldots, s_n]$ *be the score sequence of a tournament. Then the highest score of the tournament is* $n-1$ *if* $\sum_{i=1}^{n} s_i^2$ *is maximum.*

Proof. Let v_n be the vertex of the tournament with highest score s_n. We shall show that $s_n = n-1$. Suppose on contrary $s_n < n-1$. Then there exits a vertex v_p with score s_p such that $v_p(1-0)v_n$. Since $s_n \geq s_p$, therefore there exists another vertex v_q with score s_q such that $v_q(1-0)v_p$ and $v_q(0-1)v_n$.

Now, by changing the arcs $v_p(1-0)v_n$ and $v_q(1-0)v_p$ to $v_p(0-1)v_n$ and $v_q(0-1)v_n$ respectively we get a new score sequence $[t_1, t_2, \ldots, t_n]$ where $t_q = s_q - 1, t_n = s_n + 1, t_r = s_r$ for all $r,\ 1 \leq r \leq n-1$ with $r \neq q$. Then

$$\begin{aligned}\sum_{i=1}^{n} t_i^2 &= \sum_{i=1, i\neq q}^{n-1} t_i^2 + t_q^2 + t_n^2 = \sum_{i=1, i\neq q}^{n-1} s_i^2 + (s_q-1)^2 + (s_n+1)^2 \\ &= \sum_{i=1, i\neq q}^{n-1} s_i^2 + s_q^2 + 1 - 2s_q + s_n^2 + 1 + 2s_n = \sum_{i=1}^{n} s_i^2 + 2(s_n - s_q + 1) \\ &> \sum_{i=1}^{n} s_i^2 \ \text{ since } \ s_n \geq s_q,\end{aligned}$$

which is a contradiction, since $\sum_{i=1}^{n} s_i^2$ was assumed to be maximum. Hence the result follows. □

Theorem 4 *Let* $[s_1, s_2, \cdots, s_n]$ *be the score sequence of a tournament with vertex set* V *and let* m_i *be the average of the scores of the vertices* v_j *such that* $v_i(1-0)v_j$. *Then*

$$\max\{s_j + m_j : v_j \in V\} \leq \frac{3n-4}{2}, \tag{2}$$

with equality if and only if $s_i = n-1$ *where* $i = n$.

Proof. Let v_i be the vertex of a tournament where $s_i + m_i$ is maximum and let S be the sum of the scores of the vertices v_j such that $v_i(1-0)v_j$. Then

$$\max\{s_j + m_j : v_j \in V\} = s_i + m_i = s_i + \frac{S}{s_i}.$$

Again, let g_i be the average of the scores of the vertices v_k such that $v_k(1-0)v_i$. Then

$$\frac{n(n-1)}{2} = s_i + S + (n - s_i - 1)g_i, \quad \text{(by (1))}$$

$$\text{or} \quad \frac{n}{2} + n - 2 = \frac{s_i + S + (n - s_i - 1)g_i}{n-1} + n - 2,$$

$$\text{or} \quad \frac{3n-4}{2} = \frac{s_i + S + (n - s_i - 1)g_i}{n-1} + n - 2.$$

So, we have to prove that

$$s_i + \frac{S}{s_i} \leq \frac{s_i + S + (n - s_i - 1)g_i}{n-1} + n - 2,$$

$$\text{or} \quad (n-1)\left(s_i + \frac{S}{s_i}\right) \leq s_i + S + (n - s_i - 1)g_i + (n-1)(n-2),$$

$$\text{or} \quad (n-1)\left(n - 2 - s_i - \frac{S}{s_i}\right) + s_i + S + (n - s_i - 1)g_i \geq 0,$$

$$\text{or} \quad (n-1)\left(n - 2 - \frac{S}{s_i}\right) - (n-1)s_i + s_i + S + (n - s_i - 1)g_i \geq 0,$$

$$\text{or} \quad (n-1)\left(n - 2 - \frac{S}{s_i}\right) - s_i\left(n - 2 - \frac{S}{s_i}\right) + (n - s_i - 1)g_i \geq 0,$$

$$\text{or} \quad (n - 1 - s_i)\left(n - 2 - \frac{S}{s_i}\right) + (n - s_i - 1)g_i \geq 0,$$

$$\text{or} \quad (n - s_i - 1)\left(n - 2 + g_i - \frac{S}{s_i}\right) \geq 0. \qquad (3)$$

If $s_i = n - 1$, then (3) holds. Now, if $s_i \leq n - 2$, then there is at least one vertex v_k such that $v_k(1-0)v_i$, so that $g_i \geq 1$. Also $\frac{S}{s_i} \leq n - 1$. Therefore (3) holds.

This completes the proof of first part.

Now assume that equality holds in (2). Then from (3), we have

$$(n - s_i - 1)\left(n - 2 + g_i - \frac{S}{s_i}\right) = 0,$$

which gives (a) $s_i = n - 1$ or (b) $\frac{S}{s_i} - g_i = n - 2$.

Case (a). $s_i = n-1$. This is possible only when $i = n$, that is, when $s_n = n-1$.

Case (b). $\dfrac{S}{s_i} - g_i = n - 2$. Since $s_n \geq \dfrac{S}{s_i}$, therefore

$$s_n \geq n - 2 + g_i. \qquad (4)$$

Also $g_i \geq 0$ and $s_n \leq n-1$. Then from (4), we have $0 \leq g_i \leq 1$. If $g_i = 0$, then $s_n = n-1$. Again if $0 < g_i \leq 1$, then there is at least one vertex v_k such that $v_k(1-0)v_i$. Therefore $g_i \geq 1$. Hence $g_i = 1$. Thus from (4), we have $s_n \geq n-1$. Since $s_n \leq n-1$, therefore $s_n = n-1$.

Conversely, let $s_n = n-1$. Then $s_k \leq n-2$ for all k, $1 \leq k < n$. Now

$$\begin{aligned} s_k + m_k &= s_k + \frac{1}{s_k}\sum_{j=1}^{n}\{s_j : v_k(1-0)v_j\} \\ &\leq s_k + \frac{1}{s_k}\left\{\frac{s_k(s_k-1)}{2} + s_k(n-2-s_k)\right\} \\ &= s_k + \frac{s_k-1}{2} + n - 2 - s_k \\ &\leq \frac{n-2-1}{2} + n - 2 = \frac{3n-7}{2} \end{aligned}$$

and

$$\begin{aligned} s_n + m_n &= s_n + \frac{1}{s_n}\sum_{j=1}^{n}\{s_j : v_n(1-0)v_j\} \\ &= n-1 + \frac{1}{n-1}\sum_{i=1}^{n-1} s_i \\ &= n-1 + \frac{1}{n-1}\left\{\sum_{i=1}^{n} s_i - s_n\right\} \\ &= n-1 + \frac{1}{n-1}\left\{\frac{n(n-1)}{2} - (n-1)\right\} \quad \text{(by (1))} \\ &= \frac{3n-4}{2}. \end{aligned}$$

Hence, $\max\{s_j + m_j : v_j \in V\} = \dfrac{3n-4}{2}$, completing the proof. □

Theorem 5 *Let* $[s_1, s_2, \ldots, s_n]$ *be the score sequence of a tournament and let* m_i *be the average of the scores of the vertices* v_j *such that* $v_i(1-0)v_j$. *Then*

$$s_i + m_i \leq \frac{n}{2} + \frac{n-2}{n-1}s_i + (s_n - s_1)\left(1 - \frac{s_i}{n-1}\right), \tag{5}$$

holds for each i. *Further, the equality holds if and only of* $s_i = n-1$ *where* $i = n$ *or the vertex* v_i *is such that* $v_i(1-0)v_j$ *for the* s_n *score vertices* v_j *and* $v_i(0-1)v_k$ *for the* s_1 *score vertices* v_k.

Proof. Let v_i be the vertex of score s_i in the tournament T. We consider two cases: (a) $s_i = n-1$ (b) $s_i < n-1$.
Case (a). $s_i = n-1$. Then $i = n$, so that $s_n = n-1$. Therefore

$$\begin{aligned} s_n + m_n &= n-1+\frac{1}{s_n}\sum_{j=1}^{n}\{s_j : v_n(1-0)v_j\} = n-1+\frac{1}{n-1}\sum_{j=1}^{n-1}s_j \\ &= n-1+\frac{1}{n-1}\left\{\sum_{j=1}^{n}s_j - s_n\right\} \\ &= n-1+\frac{1}{n-1}\left\{\frac{n(n-1)}{2}-(n-1)\right\} \quad \text{(by (1))} \\ &= \frac{3n-4}{2}. \end{aligned}$$

Hence (5) holds.
Case (b). $s_i < n-1$. Change the orientation of the arcs $v_k(1-0)v_i$, if any, to $v_i(1-0)v_k$. Suppose this new tournament is T_1 and let $max\{s_j + m_j : v_j \in V\}$ occurs at the vertex v_i and let it be $s'_i + m'_i$.
Now for T_1, we have

$$\begin{aligned} s'_i + m'_i &= n-1+\frac{1}{s'_i}\left\{\sum_{j=1}^{n}s'_j : v_i(1-0)v_j\right\} \\ &= n-1+\frac{1}{n-1}\left\{\sum_{j=1}^{n}s'_j - s'_i\right\} \\ &= n-1+\frac{1}{n-1}\left\{\frac{n(n-1)}{2}-(n-1)\right\} \quad \text{(by (1))} \\ &= \frac{3n-4}{2}. \end{aligned} \tag{6}$$

Let S be the sum of the scores of the vertices v_j such that $v_i(1-0)v_j$ in the tournament T. Then $s_i + m_i = s_i + \dfrac{S}{s_i}$. Now,

$$\begin{aligned}(s_i' + m_i') - (s_i + m_i) &= n - 1 + \frac{1}{s_i'}\sum_{j=1}^{n}\{s_j' : v_i(1-0)v_j\} - \left(s_i + \frac{S}{s_i}\right)\\ &= n - s_i - 1 + \frac{1}{n-1}\{S + (n - s_i - 1)g_i - (n - s_i - 1)\} - \frac{S}{s_i}\\ &= n - s_i - 1 + \frac{1}{n-1}\{S + (n - s_i - 1)(g_i - 1)\} - \frac{S}{s_i},\end{aligned}$$

(where g_i is the average score of the vertices v_k such that $v_k(1-0)v_i$ in T), that is,

$$\begin{aligned}s_i + m_i &= s_i' + m_i' - (n - s_i - 1) - \frac{1}{n-1}\{S + (n - s_i - 1)(g_i - 1)\} + \frac{S}{s_i}\\ &= \frac{3n-4}{2} - (n - s_i - 1) - \frac{1}{n-1}\{S + (n - s_i - 1)(g_i - 1)\} + \frac{S}{s_i} \quad \text{(by (6))}\\ &= \frac{3n-4}{2} - (n - s_i - 1) - \frac{S}{n-1} - \frac{1}{n-1}\{(n - s_i - 1)(g_i - 1)\} + \frac{S}{s_i}\\ &= \frac{3n-4}{2} - \frac{(n - s_i - 1)(n - 1 + g_i - 1)}{n-1} + \frac{S}{s_i} - \frac{S}{n-1}\\ &= \frac{3n-4}{2} - \left(1 - \frac{s_i}{n-1}\right)(n - 2 + g_i) + \frac{S}{s_i}\left(1 - \frac{s_i}{n-1}\right)\\ &= \frac{3n-4}{2} - \left(1 - \frac{s_i}{n-1}\right)\left(n - 2 + g_i - \frac{S}{s_i}\right)\\ &= \frac{3n-4}{2} - \left(1 - \frac{s_i}{n-1}\right)(n-2) - \left(1 - \frac{s_i}{n-1}\right)\left(g_i - \frac{S}{s_i}\right)\\ &= \frac{3n-4}{2} - \left(n - 2 - \frac{(n-2)s_i}{n-1}\right) - \left(1 - \frac{s_i}{n-1}\right)\left(g_i - \frac{S}{s_i}\right)\\ &= \frac{n}{2} + \frac{n-2}{n-1}s_i - \left(1 - \frac{s_i}{n-1}\right)\left(g_i - \frac{S}{s_i}\right). \qquad (7)\end{aligned}$$

Clearly $\dfrac{S}{s_i} \le s_n$, that is, $\dfrac{S}{s_i} - s_n \le 0$ and $g_i \ge s_1$, that is $g_i - s_1 \ge 0$. Therefore $g_i - s_1 \ge \dfrac{S}{s_i} - s_n$, that is, $g_i - \dfrac{S}{s_i} \ge s_1 - s_n$. Using this in (7), we have

$$s_i + m_i \ge \frac{n}{2} + \frac{n-2}{n-1}s_i - \left(1 - \frac{s_i}{n-1}\right)(s_1 - s_n),$$

that is, $s_i + m_i \geq \frac{n}{2} + \frac{n-2}{n-1}s_i + (s_n - s_1)\left(1 - \frac{s_i}{n-1}\right)$. This completes the proof of first part.

Now assume that equality holds in (5). Then $s_i = n-1$ or $-\left(g_i - \frac{S}{s_i}\right) = s_n - s_1$, that is, $s_i = n-1$ where $i = n$ or $-g_i s_i + S = s_n s_i - s_1 s_i$. From $-g_i s_i + S = s_n s_i - s_1 s_i$, we have $\frac{-P}{n-s_i-1}s_i + s_1 s_i = s_n s_i - S$, (where P is the sum of the scores of the vertices v_k such that $v_k(1-0)v_i$ in T) or $s_1 - \frac{P}{n-s_i-1} = \frac{s_n s_i - S}{s_i}$, or $\frac{s_n s_i - S}{s_i} = \frac{(n-s_i-1)s_1 - P}{n-s_i-1}$ or $s_1 - \frac{P}{n-s_i-1} = \frac{s_n s_i - S}{s_i}$, or $\frac{(n-s_i-1)s_1 - P}{n-s_i-1} = \frac{s_n s_i - S}{s_i} \geq 0$, since $\frac{S}{s_i} \leq s_n$, that is, $(n-s_i-1)s_1 - P \geq 0$, or $P \leq (n-s_i-1)s_1$. But $P \geq (n-s_i-1)s_1$. Therefore $P = (n-s_i-1)s_1$. This means that all those vertices v_k with $v_k(1-0)v_i$ are of score s_1. Using this fact in

$$\frac{(n-s_i-1)s_1 - P}{n-s_i-1} = \frac{s_n s_i - S}{s_i},$$

we have

$$\frac{(n-s_i-1)s_1 - (n-s_i-1)s_1}{n-s_i-1} = \frac{s_n s_i - S}{s_i},$$

or $\frac{s_n s_i - S}{s_i} = 0$ or $S = s_n s_i$ or $\frac{S}{s_i} = s_n$. This means that all those vertices v_j with $v_i(1-0)v_j$ are of score s_n.

Conversely, let $s_i = n-1$, where $i = n$ or $v_i(1-0)v_j$ for the s_n score vertices v_j and $v_i(0-1)v_k$ for the s_1 score vertices v_k. For $s_i = n-1$, where $i = n$, the equality holds in (5) by using case (a). Now, if $v_i(1-0)v_j$ for the s_n score vertices v_j and $v_i(0-1)v_k$ for the s_1 score vertices v_k, then

$$s_i + m_i = s_i + \frac{s_n s_i}{s_i} = s_i + s_n$$

and

$$\begin{aligned}
&\frac{n}{2}+\frac{n-2}{n-1}s_i+(s_n-s_1)\left(1-\frac{s_i}{n-1}\right)\\
&\qquad=\frac{n(n-1)}{2}\frac{1}{n-1}+\frac{n-2}{n-1}s_i+\frac{(s_n-s_1)(n-1-s_i)}{n-1}\\
&\qquad=\frac{1}{n-1}\left\{\sum_{i=1}^{n}s_i+(n-2)s_i+(s_n-s_1)(n-1-s_i)\right\}\text{ by (1)}\\
&\qquad=\frac{1}{n-1}\big\{s_i+s_ns_i+s_1(n-s_i-1)\\
&\quad+(n-2)s_i+s_n(n-1-s_i)-s_1(n-1-s_i)\big\}\\
&\qquad=\frac{1}{n-1}\big\{s_i+s_ns_i+ns_i-2s_i+ns_n-s_n-s_ns_i\big\}\\
&\qquad=\frac{1}{n-1}\big\{(n-1)s_i+(n-1)s_n\big\}=s_i+s_n.
\end{aligned}$$

Therefore, the equality holds in (5). □

Corollary 6 *Let* $[s_1,s_2,\dots,s_n]$ *be the score sequence of a tournament and let* m_i *be the average of the scores of the vertices* v_j *such that* $v_i(1-0)v_j$. *Then*

$$s_i+m_i\le\frac{n}{2}+\frac{n-2}{n-1}s_n+(s_n-s_1)\left(1-\frac{s_n}{n-1}\right),\tag{8}$$

holds for each i. *Further, the equality holds if and only if* $s_i=n-1$ *where* $i=n$ *or the vertex* v_i*(score is* s_n*) is such that* $v_i(1-0)v_j$ *for the* s_n *score vertices* v_j *and* $v_i(0-1)v_k$ *for the* s_1 *score vertices* v_k.

Proof. Since (5) is true for each i and since $s_i\le s_n$, therefore we get (8). Using Theorem 5, we conclude that the equality holds if and only if $s_i=n-1$ where $i=n$ or the vertex v_i(score is s_n) is such that $v_i(1-0)v_j$ for the s_n score vertices v_j and $v_i(0-1)v_k$ for the s_1 score vertices v_k. □

References

[1] P. Avery, Score sequences of oriented graphs, *J. Graph Theory*, **15, 3** (1991) 251–257. ⇒257

[2] R. A. Brualdi and J. Shen, Landau's inequalities for tournament scores and a short proof of a Theorem on transitive sub-tournaments, *J. Graph Theory*, **38** (2001) 244–254. ⇒258

[3] J. R. Griggs and K. B. Reid, Landau's theorem revisited, *Australasian J. Combinatorics*, **20** (1999) 19–24. ⇒258
[4] K. K. Kayibi, M. A. Khan and S. Pirzada, Uniform sampling of k-hypertournaments, *Linear and Multilinear Algebra*, **61, 1** (2012) 123–138. ⇒257
[5] M. A. Khan, S. Pirzada and K. K. Kayibi, Scores, inequalities and regular hypertournaments, *J. Math. Ineq. Appl.*, **15, 2** (2012) 343–351. ⇒257
[6] H. G. Landau, On dominance relations and the structure of animal societies, III, the condition for a score structure, *Bull. Math. Biophys.*, **15** (1953) 143–148. ⇒257, 258
[7] J. W. Moon, *Topics on Tournaments*, Holt, Rinehart and Winston, New York (1968). ⇒258
[8] S. Pirzada and G. Zhou, Score sequences in oriented k-hypergraphs, *European J. Pure and Applied Mathematics*, **1, 3** (2008) 10–20. ⇒257
[9] S. Pirzada, T. A. Chishti and T. A. Naikoo, Score lists in [h-k]-bipartite hypertournaments, *Discrete Mathematics and Applications*, **19, 3** (2009) 321-328. ⇒257
[10] S. Pirzada and G. Zhou, New proof on k-hypertournament scores, *Acta Univ. Sapientiae, Informatica*, **2, 1** (2010) 5–9. ⇒257
[11] S. Pirzada, On scores in multipartite hypertournaments, *Eurasian Math. J.* **2, 1** (2011) 112–119. ⇒257
[12] S. Pirzada, On scores in bipartite hypertournaments, *Math. Vesnik*, **64,4** (2012) 286-296. ⇒257
[13] S. Pirzada, G. Zhou and A. Ivanyi, Score lists of multipartite hypertournaments, *Acta Univ. Sapien. Informatica*, **2, 2** (2010) 184–193. ⇒257
[14] S. Pirzada, An Introduction to Graph Theory, Universities Press, Hyderabad, India, 2012. ⇒257
[15] S. Pirzada, T. A. Naikoo and Zhou Guofei, Score lists in tripartite hypertournaments, *Graphs and Combinatorics*, **23, 4** (2007) 445-454. ⇒257
[16] K. B. Reid, Tournament, scores, kings, generalizations and special topics, *Congressus Numerantium*, **115** (1996) 171–211. ⇒258
[17] S. Pirzada and T. A. Naikoo, Inequalities for marks in digraphs, *J. Math. Inequalities Appl.*, **9, 2** (2006) 189–198. ⇒257
[18] S. Pirzada and U. Samee, Mark sequences in digraphs, *Seminare Lotharingien de Combinatoire*, **55** (2006) Art.B55c. ⇒257
[19] S. Pirzada and T. A. Naikoo, Score sets for oriented graphs, *Applicable Analysis and Discrete Mathematics*, **2, 1** (2008) 107–113. ⇒257
[20] S. Pirzada and T. A. Naikoo, Score sets in k-partite tournaments, *J. Applied Mathematics and Computing*, **22, 1-2** (2006) 237–245. ⇒257
[21] S. Pirzada, T. A. Naikoo and T. A. Chishti, Score sets in oriented bipartite graphs, *Novi Sad J. Mathematics*, **36, 1** (2006) 35–45. ⇒257
[22] S. Pirzada, T. A. Naikoo and N. A. Shah, Score sequences in oriented graphs, *J. Applied Mathematics and Computing*, **23, 1-2** (2007) 257–268. ⇒257

[23] S. Pirzada and T. A. Naikoo, Score sets in tournaments, *Vietnam J. Math.*, **34, 2** (2006) 157–161. ⇒257

[24] Zhou Guofei and S. Pirzada, Degree Sequences in oriented k-hypergraphs, *J. Applied Math. and Comput.*, **27** (2008) 149–158. ⇒257

Received: November 18, 2018 • Revised: December 20, 2018

Acta Universitatis Sapientiae, Informatica is covered by the following services:

ACM Digital Library
Baidu Scholar
Cabell's Directory
Celdes
CNKI Scholar (China National Knowledge Infrastructure)
CNPIEC
Dimensions
DOAJ (Directory of Open Access Journals)
EBSCO (relevant databases)
EBSCO Discovery Service
Engineering Village
Genamics JournalSeek
Google Scholar
Inspec
io-port.net
J-Gate
Japan Science and Technology Agency (JST)
JournalTOCs
KESLI-NDSL (Korean National Discovery for Science Leaders)
Micosoft Academic
Naviga (Softweco)
Primo Central (ExLibris)
ReadCube
Sherpa/RoMEO
Summon (Serials Solutions/ProQuest)
TDNe)
Ulrich's Periodicals Directory/ulrichsweb
WanFang Data
Web of Science – Emerging Sources Citation Index
WorldCat (OCLC)
Zentralblatt für Mathematik

Acta Universitatis Sapientiae

The scientific journal of Sapientia Hungarian University of Transylvania publishes original papers and surveys in several areas of sciences written in English.
Information about each series can be found at
http://www.acta.sapientia.ro.

Acta Universitatis Sapientiae, Informatica

Each volume contains two issues.


Sapientia University


Sciendo by De Gruyter


Scientia Publishing House

ISSN 1844-6086
http://www.acta.sapientia.ro

Information for authors

Acta Universitatis Sapientiae, Informatica publishes original papers and surveys in various fields of Computer Science. All papers are peer-reviewed.

Papers published in current and previous volumes can be found in Portable Document Format (pdf) form at the address: http://www.acta.sapientia.ro.

The submitted papers should not be considered for publication by other journals. The corresponding author is responsible for obtaining the permission of coauthors and of the authorities of institutes, if needed, for publication, the Editorial Board is disclaiming any responsibility.

Submission must be made by email (acta-inf@acta.sapientia.ro) only, using the LaTeX style and sample file at the address http://www.acta.sapientia.ro. Beside the LaTeX source a pdf format of the paper is necessary too.

Prepare your paper carefully, including keywords, ACM Computing Classification System codes (http://www.acm.org/about/class/1998) and AMS Mathematics Subject Classification codes (http://www.ams.org/msc/).

References should be listed alphabetically based on the Intructions for Authors given at the address http://www.acta.sapientia.ro.

Illustrations should be given in Encapsulated Postscript (eps) format.

One issue is offered each author free of charge. No reprints will be available.

Contact address and subscription:
Acta Universitatis Sapientiae, Informatica
RO 400112 Cluj-Napoca
Str. Matei Corvin nr. 4.
Email: acta-inf@acta.sapientia.ro

Printed by Idea Printing House
Director: Péter Nagy

ISSN 1844-6086
http://www.acta.sapientia.ro