Theory and Applications of Graphs

Volume 8 | Issue 1 Article 3

March 2021

An Efficient Algorithm to Test Potential Bipartiteness of Graphical Degree Sequences

Kai Wang
Georgia Southern University, kwang@georgiasouthern.edu

Recommended Citation
Wang, Kai (2021) "An Efficient Algorithm to Test Potential Bipartiteness of Graphical Degree Sequences," *Theory and Applications of Graphs*: Vol. 8 : Iss. 1 , Article 3.
DOI: 10.20429/tag.2021.080103
Available at: https://digitalcommons.georgiasouthern.edu/tag/vol8/iss1/3

An Efficient Algorithm to Test Potential Bipartiteness of Graphical Degree Sequences

Cover Page Footnote

This research has been supported by a research seed grant of Georgia Southern University. The computational experiments have been supported by the Talon cluster of Georgia Southern University. We thank the referees for their valuable suggestions for improving the quality of the manuscript.

Abstract

As a partial answer to a question of Rao, a deterministic and customizable efficient algorithm is presented to test whether an arbitrary graphical degree sequence has a bipartite realization. The algorithm can be configured to run in polynomial time, at the expense of possibly producing an erroneous output on some "yes" instances but with very low error rate.

Keywords— graphical degree sequence, bipartite realization

1 Introduction

Given an arbitrary graphical degree sequence $\mathbf{d} = (d_1 \geq d_2 \geq \cdots \geq d_n)$, let $\mathscr{R}(\mathbf{d})$ denote the set of all of its non-isomorphic realizations. As usual, let $\chi(G)$ and $\omega(G)$ denote the chromatic number and clique number of a finite simple undirected graph G respectively. It is known from Punnim [12] that for any given $\mathbf{d}$ the set $\{\chi(G) : G \in \mathscr{R}(\mathbf{d})\}$ is exactly a set of integers in some interval. Define $X(\mathbf{d})$ to be $\max\{\chi(G) : G \in \mathscr{R}(\mathbf{d})\}$ and $\chi(\mathbf{d})$ to be $\min\{\chi(G) : G \in \mathscr{R}(\mathbf{d})\}$. These two quantities can be interesting for the structural properties of all the graphs in $\mathscr{R}(\mathbf{d})$.

Good lower and upper bounds on $X(\mathbf{d})$ are known from Dvořák and Mohar [3] in terms of $\Omega(\mathbf{d}) = \max\{\omega(G) : G \in \mathscr{R}(\mathbf{d})\}$, which can be easily computed for any given $\mathbf{d}$ using the algorithm from Yin [18]. For example, $X(\mathbf{d}) \geq \Omega(\mathbf{d})$, $X(\mathbf{d}) \leq \frac{4}{5}\Omega(\mathbf{d}) + \frac{1}{5}\max(\mathbf{d}) + 1$ and $X(\mathbf{d}) \leq \frac{6}{5}\Omega(\mathbf{d}) + \frac{3}{5}$.

It appears computationally intractable to compute $\chi(\mathbf{d})$ for any given zero-free $\mathbf{d}$. In this paper we are concerned with the related, somewhat easier, decision problem of whether $\chi(\mathbf{d}) = 2$. Clearly, this is equivalent to decide whether $\mathbf{d}$ has a bipartite realization, which is actually the first listed unsolved problem in Rao [13] to characterize potentially bipartite graphical degree sequences and which remains unsolved to our knowledge. We will try to design an efficient algorithm to decide whether the answer is "yes" or "no". In case one is interested in an actual bipartite realization when the answer is "yes", then one can construct it easily from what our designed algorithm computes together with other known methods. We will briefly explain this later.

Note that the input $\mathbf{d}$ is a single sequence of vertex degrees. A related problem is to decide, given two sequences of positive integers $(a_1, a_2, \cdots, a_m; b_1, b_2, \cdots, b_n)$, where $a_1 \geq a_2 \geq \cdots \geq a_m$ and $b_1 \geq b_2 \geq \cdots \geq b_n$ and $\sum_{i=1}^m a_i = \sum_{i=1}^n b_i$, whether there is a bipartite graph whose two partite sets have $\mathbf{a} = (a_1, a_2, \cdots, a_m)$ and $\mathbf{b} = (b_1, b_2, \cdots, b_n)$ as their respective degree sequences. This problem can be easily solved by applying the Gale-Ryser theorem [6, 15], which states that the answer is "yes" if and only if for each $k = 1, \ldots, m$,

$$\sum_{i=1}^{k} a_i \leq \sum_{i=1}^{n} \min\{k, b_i\},$$

which is also equivalent to for each $k = 1, \ldots, n$,

$$\sum_{i=1}^{k} b_i \leq \sum_{i=1}^{m} \min\{k, a_i\}.$$

Using the standard concept of conjugate integer partition in number theory and the dominance relation between a pair of partitions of the same integer, one can easily see that the Gale-Ryser inequalities stated above can be represented in an equivalent and concise way. The conjugate partition $\mathbf{p}^* = (p_1^*, p_2^*, \ldots, p_{p_1}^*)$ of a partition $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ with $p_1 \geq p_2 \geq \ldots \geq p_n$ can be defined as $p_i^* = |\{j | 1 \leq j \leq n, p_j \geq i\}|$ for $1 \leq i \leq p_1$, which can be verified through the Ferrers diagram of the partition $\mathbf{p}$. Here we use the common definition of dominance (sometimes also called majorization) ordering between two partitions of the same integer: a partition $\mathbf{p} = (p_1, p_2, \cdots)$ dominates a partition $\mathbf{q} = (q_1, q_2, \cdots)$, denoted $\mathbf{p} \trianglerighteq \mathbf{q}$ or $\mathbf{q} \trianglelefteq \mathbf{p}$, if $\sum_{i=1}^{\infty} p_i = \sum_{i=1}^{\infty} q_i$, $\sum_{i=1}^{j} p_i \geq \sum_{i=1}^{j} q_i$ for each $j = 1, 2, \cdots$. By convention, $p_j = 0$ for $j > \ell(\mathbf{p})$, where $\ell(\mathbf{p})$ denotes the number of parts in the partition $\mathbf{p}$. The reader may refer to [1, 16] for more details about these concepts. With these notations, the Gale-Ryser condition above can be represented as $\mathbf{a} \trianglelefteq \mathbf{b}^*$ or $\mathbf{b} \trianglelefteq \mathbf{a}^*$. We also use $|\mathbf{p}|$ to denote the weight of the partition $\mathbf{p}$, that is, the sum of all the parts of $\mathbf{p}$. The multiplicity of a part p_i in a partition $\mathbf{p}$ is denoted $\mu(p_i)$.

Krause [9] gives an alternative proof of the Gale-Ryser theorem, which actually provides a procedure to construct a bipartite realization of any bigraphic pair $(\mathbf{a}, \mathbf{b})$ (i.e. a pair $(\mathbf{a}, \mathbf{b})$ that satisfies the Gale-Ryser condition). If one is interested in a bipartite realization of the input $\mathbf{d}$ that is potentially bipartite, this procedure can be used since our designed algorithm will compute a bigraphic pair $(\mathbf{a}, \mathbf{b})$ from the input $\mathbf{d}$ in case the algorithm decides the answer is "yes".

The rest of the paper is organized as follows. Section 2 describes the algorithm to decide whether a given $\mathbf{d}$ has a bipartite realization. Section 3 gives a time complexity analysis of the algorithm. Section 4 presents some experimental results. Section 5 discusses alternative designs of the algorithm and comments on the complexity of the decision problem. Section 6 concludes with further research directions.

2 Description of the Algorithm

Clearly, to decide whether any zero-free graphical degree sequence $\mathbf{d} = (d_1 \geq d_2 \geq \cdots \geq d_n)$ with weight $|\mathbf{d}| = \sum_{i=1}^{n} d_i$ has a bipartite realization, we first need to determine whether it has a bipartition into $\mathbf{a}$ and $\mathbf{b}$ of equal weights $|\mathbf{d}|/2$ (for convenience, we call such bipartitions of $\mathbf{d}$ *candidate* bipartitions). Whether there exists a candidate bipartition can be decided with a standard dynamic programming algorithm for the subset sum problem, which runs in pseudo-polynomial time $O(nT)$ where n is the number of given integers and T is the target sum to be found. Since every term in $\mathbf{d}$ of length n is less than n, our target $T = |\mathbf{d}|/2$ is $O(n^2)$ so this decision can made in $O(n^3)$ time. In fact, many inputs admit a large number of candidate bipartitions. Now we can see that the decision problem boils down to checking whether $\mathbf{d}$ has at least one candidate bipartition $(\mathbf{a}, \mathbf{b})$ that satisfies the Gale-Ryser condition (i.e. $\mathbf{a} \trianglelefteq \mathbf{b}^*$ or $\mathbf{b} \trianglelefteq \mathbf{a}^*$).

A naive algorithm can simply enumerate all candidate bipartitions of $\mathbf{d}$ and check each of them against the Gale-Ryser condition. Such an algorithm necessarily runs in exponential time in the worst case. Our algorithm is more sophisticated than that. It has two phases. The first phase utilizes up to seven rules that can all be easily checked to reach a conclusion for many inputs so that exhaustive enumeration can be avoided. As a matter of fact, in

Section 4 we will show that most of the inputs can be resolved by this phase alone. The second phase is the enumeration phase, in which we perform "brute-force" search in a clever way.

In describing and justifying the seven rules in the first phase, we seek a candidate bipartition of $\mathbf{d}$ into the left side $\mathbf{a}$ and the right side $\mathbf{b}$ in such a way that at least half of the largest terms in $\mathbf{d}$ appear in $\mathbf{a}$, without loss of generality. For example, for any input $\mathbf{d}$ of length 50 with the largest term 34 whose multiplicity is 5 (i.e. there are exactly 5 copies of 34 in $\mathbf{d}$), we will seek a candidate bipartition such that the left side $\mathbf{a}$ contains at least 3 copies of 34.

Rule 1. *If* $\mathbf{d}$ *does not have a candidate bipartition, then it is not potentially bipartite.*

Proof. This rule is obvious. As mentioned above, this rule can be easily implemented through dynamic programming for the PARTITION problem. □

Rule 2. *If* $|\mathbf{d}| > \frac{n^2}{2}$*, then* $\mathbf{d}$ *is not potentially bipartite.*

Proof. Based on Mantel's theorem [10], any simple undirected bipartite graph on n vertices has at most $\frac{n^2}{4}$ edges. So the degree sum cannot exceed $\frac{n^2}{2}$ for any $\mathbf{d}$ that is potentially bipartite. □

Rule 3. *If* $d_1 + d_{n+1-d_1} > n$*, then* $\mathbf{d}$ *is not potentially bipartite.*

Proof. Suppose $\mathbf{d}$ is potentially bipartite. The left partite set contains a vertex v_1 of degree d_1 so the right partite set contains at least d_1 vertices ($v_1's$ neighbors), each of which has a degree at most $n - d_1$ since the left partite set has at most $n - d_1$ vertices. Consequently, $\mathbf{d}$ must contain at least d_1 degrees that are $\leq n - d_1$. Therefore, d_{n+1-d_1} must be $\leq n - d_1$ for $\mathbf{d}$ to be potentially bipartite. □

Rule 4. *If* $\sum_{i=1}^{n-d_1} d_i < \frac{|\mathbf{d}|}{2}$*, then* $\mathbf{d}$ *is not potentially bipartite.*

Proof. As mentioned in the proof of Rule 3, the left partite set has at most $n - d_1$ vertices. Clearly, the degree sum $|\mathbf{a}|$ of the left side $\mathbf{a}$ is impossible to exceed $\sum_{i=1}^{n-d_1} d_i$. Therefore, $\sum_{i=1}^{n-d_1} d_i$ must be at least $\frac{|\mathbf{d}|}{2}$ for $\mathbf{d}$ to be potentially bipartite. □

Rule 5. *If* $\sum_{d_i > n-d_1} d_i > \frac{|\mathbf{d}|}{2}$*, then* $\mathbf{d}$ *is not potentially bipartite.*

Proof. As shown in the proof of Rule 3, each of the right side degrees in $\mathbf{b}$ is at most $n - d_1$. Therefore, every degree larger than $n - d_1$ must be in the left side $\mathbf{a}$ and the sum of such degrees should not exceed $\frac{|\mathbf{d}|}{2}$ for any $\mathbf{d}$ that is potentially bipartite. □

For the following rule, we will need the concept of *residue* of a finite simple undirected graph G or a graphical degree sequence $\mathbf{d}$ introduced in Favaron et al. [5] and we use $R(G)$ and $R(\mathbf{d})$ to indicate this function. We also use $\overline{\mathbf{d}}$ to denote the complementary graphical degree sequence of $\mathbf{d}$: $(n - 1 - d_n \geq n - 1 - d_{n-1} \geq \cdots \geq n - 1 - d_1)$, which is the degree sequence of the complementary graph of any realization of $\mathbf{d}$. The residue $R(\mathbf{d})$ of a graphical degree sequence $\mathbf{d} = (d_1 \geq d_2 \geq \cdots \geq d_n)$ is the number of zeros obtained in the last step by the iterative procedure consisting of deleting the largest term d_1 from $\mathbf{d}$, subtracting 1 from the d_1 following terms (i.e. $d_2, d_3, \dots, d_{d_1+1}$) and sorting the new sequence in weakly decreasing order. This iterative procedure is actually Havel and Hakimi's method [8, 7] to decide whether an arbitrary weakly decreasing sequence of non-negative integers is a graphical degree sequence, which says the answer is "yes" if and only if a sequence consisting entirely of zeros is obtained in the end. The reader can easily verify the residue of the graphical degree sequence $(4,4,3,3,2,2)$ is 2.

Rule 6. *If $R(\overline{\mathbf{d}}) \geq 3$, then $\mathbf{d}$ is not potentially bipartite.*

Proof. As proved in [5], the residue $R(\mathbf{d})$ of a graphical degree sequence $\mathbf{d}$ is a lower bound on the independence number of any realization of $\mathbf{d}$. Then clearly $R(\overline{\mathbf{d}})$ is a lower bound on the clique number of any realization of $\mathbf{d}$. The result follows because any graph with a clique of size at least 3 is not bipartite. □

The following is a similar rule that uses the concept of Murphy's bound introduced in Murphy [11], denoted $\beta(G)$ or $\beta(\mathbf{d})$ here, which is also a lower bound on the independence number of any realization G of $\mathbf{d}$. Murphy's bound $\beta(\mathbf{d})$ of a graphical degree sequence $\mathbf{d} = (d_1 \geq d_2 \geq \cdots \geq d_n)$ can be computed as follows. Let $d'_i = d_{n+1-i}$ for $1 \leq i \leq n$. Define an auxiliary iterative function $f\colon \mathbb{N} \to \{d'_1, d'_2, \dots, d'_n, \infty\}$ as: Set $f(1) = d'_1$ and if $f(j) = d'_k$ for $1 \leq j \leq n$, then

$$f(j+1) = \begin{cases} d'_{k+f(j)+1}, & \text{if } k + f(j) + 1 \leq n; \\ \infty, & \text{otherwise.} \end{cases}$$

If $f(j) = \infty$ then $f(j+1) = \infty$. Murphy's bound $\beta(\mathbf{d})$ is then defined as $\max\{j \in \mathbb{N} : f(j) \neq \infty\}$. The reader can verify that Murphy's bound of the graphical degree sequence $(6,6,5,5,3,2,2,2,1)$ is 3. Note that both $R(\mathbf{d})$ and $\beta(\mathbf{d})$ can be calculated easily based on their definitions.

Rule 7. *If $\beta(\overline{\mathbf{d}}) \geq 3$, then $\mathbf{d}$ is not potentially bipartite.*

If the input $\mathbf{d}$ passes the tests of all of the above seven rules and cannot be resolved as a "no" instance, then our algorithm will enter the second phase called the enumeration phase. From Rule 5 we know that by now we must have $\frac{|\mathbf{d}|}{2} \geq \sum_{d_i > n-d_1} d_i$. In the special case that equality holds, which means the left side $\mathbf{a}$ must contain exactly those degrees that are larger than $n - d_1$ should $\mathbf{d}$ be potentially bipartite, our algorithm can immediately stop based on the result of the Gale-Ryser conditional test on this candidate bipartition of $\mathbf{d}$. Otherwise, our algorithm continues with $S = \frac{|\mathbf{d}|}{2} - \sum_{d_i > n-d_1} d_i > 0$, which is the sum of the additional degrees that need to be in the left side $\mathbf{a}$ besides those that are larger than $n - d_1$. For convenience, we use $\mathbf{a_f}$ to denote the subsequence of $\mathbf{d}$ consisting of those degrees that are larger than $n - d_1$. Note that $\mathbf{a_f}$ is an empty sequence when $n \geq 2d_1$.

The second phase will then enumerate candidate bipartitions of $\mathbf{d}$ into $(\mathbf{a}, \mathbf{b})$ by specifying which degrees will be in the left side $\mathbf{a}$, which also automatically specifies $\mathbf{b} = \mathbf{d} - \mathbf{a}$. As we already know, we need to choose a subsequence of $\mathbf{d} - \mathbf{a_f}$ (i.e. from those degrees in $\mathbf{d}$ that are at most $n - d_1$) with sum S and concatenate $\mathbf{a_f}$ with this subsequence of degrees to form $\mathbf{a}$ based on the above discussion. Several restrictions regarding $\ell(\mathbf{a})$ (recall we use $\ell(\mathbf{a})$ to denote the number of terms in $\mathbf{a}$) can be put on the left side $\mathbf{a}$ for the candidate bipartitions $(\mathbf{a}, \mathbf{b})$ to possibly satisfy the Gale-Ryser conditional test so that our algorithm will enumerate as few candidate bipartitions as possible.

Restriction 1. *The number of degrees $\ell(\mathbf{a})$ in the left side $\mathbf{a}$ cannot exceed $n - d_1$. This is because the right side $\mathbf{b}$ contains at least d_1 degrees.*

Restriction 2. *Let l_1 be the maximum number of degrees in $\mathbf{d}$ with sum at most $\frac{|\mathbf{d}|}{2}$. Then the number of degrees $\ell(\mathbf{a})$ in the left side $\mathbf{a}$ cannot exceed l_1. This is because the degrees in the left side $\mathbf{a}$ must have sum $\frac{|\mathbf{d}|}{2}$.*

Restriction 3. *Let d_m be the minimum largest degree in any subsequence of $\mathbf{d}$ with sum at least $\frac{|\mathbf{d}|}{2}$. Then the number of degrees $\ell(\mathbf{a})$ in the left side $\mathbf{a}$ must be at least d_m. This is because the largest degree in the right side $\mathbf{b}$ must be at least d_m and the conjugate of $\mathbf{a}$ should dominate $\mathbf{b}$.*

Restriction 4. *Let l_2 be the minimum number of degrees in $\mathbf{d}$ with sum at least $\frac{|\mathbf{d}|}{2}$. Then the number of degrees $\ell(\mathbf{a})$ in the left side $\mathbf{a}$ must be at least l_2. The reason is similar to that for Restriction 2.*

It's not hard to see that d_m, l_1 and l_2 can all be easily calculated with greedy algorithms. The above discussion shows we can enumerate all subsequences $\mathbf{a}$ of $\mathbf{d}$ that satisfies the following three requirements:

1. it includes all degrees in $\mathbf{a_f}$ (i.e. those degrees in $\mathbf{d}$ that are greater than $n - d_1$).

2. it has sum $|\mathbf{a}| = \frac{|\mathbf{d}|}{2}$.

3. its number of degrees $\ell(\mathbf{a})$ should satisfy $\max\{d_m, l_2\} \leq \ell(\mathbf{a}) \leq \min\{n - d_1, l_1\}$.

In order to find a successful (i.e. satisfying the Gale-Ryser condition) candidate bipartition $(\mathbf{a}, \mathbf{b})$ of $\mathbf{d}$, our intuition is to include a suitable number of large degrees from $\mathbf{d} - \mathbf{a_f}$ and as many small degrees of $\mathbf{d} - \mathbf{a_f}$ as possible into $\mathbf{a}$ without violating the inequalities in requirement 3 mentioned above. In this way $\mathbf{b} = \mathbf{d} - \mathbf{a}$ will not include many of the largest degrees in $\mathbf{d}$ while $\mathbf{a}$ will still include enough number of degrees, which makes it more likely for the conjugate of $\mathbf{a}$ to dominate $\mathbf{b}$.

Following this intuition we calculate a maximum index $x_0 \in \{1, 2, \cdots, n\}$ such that $\mathbf{a}$ cannot include all $\{d_1, d_2, \cdots, d_{x_0}, d_{x_0+1}\}$ in order for its conjugate to dominate $\mathbf{b}$. This index x_0 can be easily calculated as follows. Starting from $x = 1$, if for some x, when we include all $\{d_1, d_2, \cdots, d_x\}$ in $\mathbf{a}$ and include from $\mathbf{d} - \{d_1, d_2, \cdots, d_x\}$ as many smallest degrees as possible into $\mathbf{a}$ while still maintaining the correct sum $|\mathbf{a}| = \frac{|\mathbf{d}|}{2}$, and when the number of degrees $\ell(\mathbf{a})$ in $\mathbf{a}$ starts to fall below $\max\{d_m, l_2\}$, then x_0 can be chosen to be $x - 1$.

After x_0 has been calculated, we will try to find out if we can include a subsequence $\mathbf{d_S}$ of $\{d_1, d_2, \cdots, d_{x_0}\}$ into $\mathbf{a}$ together with some degrees in $\mathbf{d} - \mathbf{d_S}$ such that the conjugate of $\mathbf{a}$ dominates $\mathbf{b}$. Without loss of generality, this subsequence $\mathbf{d_S}$ can be chosen to be some largest terms of $\{d_1, d_2, \cdots, d_{x_0}\}$. Or, equivalently, we can remove some smallest terms from $\{d_1, d_2, \cdots, d_{x_0}\}$ one at a time to obtain these subsequences. For each such subsequence $\mathbf{d_S} = \{d_1, d_2, \cdots, d_x\}$, where $\max\{\frac{\mu(d_1)}{2}, \ell(\mathbf{a_f})\} \leq x \leq x_0$ since $\mathbf{d_S}$ has to include all degrees in $\mathbf{a_f}$ and we insist that $\mathbf{a}$ include at least half of the largest terms of $\mathbf{d}$ according to the above discussion, we perform the following two enumerative steps to fully construct $\mathbf{a}$:

1. starting from the largest possible, choose some degree d_y from $\mathbf{d}-\mathbf{d_{S'}}$ and include some copies of d_y into $\mathbf{a}$. We also stipulate that no degree larger than d_y from $\mathbf{d}-\mathbf{d_{S'}}$ will be included into $\mathbf{a}$. Here $\mathbf{d_{S'}}$ is defined as follows. If $\mathbf{d_S}$ includes all $\mu(d_x)$ copies of d_x from $\mathbf{d}$, then $\mathbf{d_{S'}}$ includes $\mathbf{d_S}$ together with all copies of the degree from $\mathbf{d}$ which is immediately smaller than d_x. If $\mathbf{d_S}$ does not include all $\mu(d_x)$ copies of d_x from $\mathbf{d}$, then $\mathbf{d_{S'}}$ includes $\mathbf{d_S}$ together with all the remaining copies of d_x from $\mathbf{d}$. The motivation for such a definition is that we don't want d_y to equal a degree we have just excluded from a previous consideration of $\mathbf{d_S}$ when x is being reduced starting from x_0.

2. include some small terms that are all less than d_y from $\mathbf{d}-\mathbf{d_{S'}}-\mathbf{d_y}$ into $\mathbf{a}$, where $\mathbf{d_y}$ is the subsequence of $\mathbf{d}$ consisting of all $\mu(d_y)$ copies of d_y. We can generate a number of possible combinations of small terms with each combination summing to a suitable value based on the choice of $\mathbf{d_S}$ and the choice in the enumerative step (1) and having a suitable number of terms so that $\ell(\mathbf{a})$ satisfies the inequalities in the above requirement 3. An appropriate procedure can be designed for this purpose such that those combinations with more smaller terms are generated first and each combination can be generated in $O(n)$ time.

Note that both of these steps are enumerative steps. Step (1) must be exhaustive by trying each possible distinct d_y from $\mathbf{d}-\mathbf{d_{S'}}$ and each of the possible number of copies up to its multiplicity $\mu(d_y)$ in $\mathbf{d}$. Step (2) can be non-exhaustive, which means we can impose a limit l_c on the number of possible combinations of small terms to be included into $\mathbf{a}$. This parameter l_c is the place where our algorithm is customizable and in reality we can choose l_c to be a constant or a low degree polynomial of n. This non-exhaustive enumeration step does open the possibility of our algorithm making an error on some "yes" input instances if the specified limit l_c will cause our algorithm to skip some of the possible combinations. However, this step will not introduce any error on "no" input instances. We also note that some of the choices in these two steps can be pruned during the enumerative process to speed up the enumeration phase when they will cause $\ell(\mathbf{a})$ to fail to satisfy the inequalities in the above requirement 3. In fact, the lower bound on $\ell(\mathbf{a})$ can be dynamically increased during the process as x is being reduced starting from x_0 so that the largest degree in $\mathbf{b}$ dynamically increases.

The reader may have noticed that these enumerative steps are more sophisticated and complicated than the simple naive scheme of enumerating all possible subsequences of $\mathbf{d}-\mathbf{a_f}$ with sum S. We will discuss several alternative enumeration schemes later in Section 5. The presented enumeration scheme above is the fastest we found through experiments.

During the enumeration phase, the algorithm will stop and output "yes" if a successful candidate bipartition $(\mathbf{a},\mathbf{b})$ is found. Otherwise, it will stop enumeration and output "no" when the index x falls below $\max\{\frac{\mu(d_1)}{2},\ell(\mathbf{a_f})\}$.

We note that the enumeration phase can be easily parallelized with respect to the different choices of $\mathbf{d_S}$. However, it may not be worth it in practice given the good run time performance of the serial version unless the input is long and hard (say when $n=\ell(\mathbf{d})>500$). See the following sections for run time complexity analysis and experimental evaluations.

3 Analysis of Run Time Complexity

The seven rules in the first phase can all be checked in polynomial time. It can be easily verified that the total run time of these rules is $O(n^3)$.

In the second phase, the three quantities d_m, l_1 and l_2 can all be computed in $O(n)$ time. The maximum index x_0 can be calculated in $O(n^2)$ time. The number of choices for $\mathbf{d_S}$ is $O(n)$. For each choice of $\mathbf{d_S}$, the number of choices for d_y and its number of copies to be included in $\mathbf{a}$ in the enumerative step (1) is $O(n)$. The maximum number l_c of combinations of the remaining small terms to be included in $\mathbf{a}$ in the enumerative step (2) can be chosen to be $O(1)$, $O(n)$, etc. Each combination can be generated in $O(n)$ time. Whenever a full left side $\mathbf{a}$ has been constructed, the Gale-Ryser conditional test on the candidate bipartition $(\mathbf{a}, \mathbf{b})$ can be performed in $O(n)$ time. Overall, we can see that the second phase runs in $O(n^5)$ time when l_c is $O(n)$ or $O(n^4)$ time when l_c is $O(1)$. Note this run time complexity is achieved at the expense of the algorithm possibly producing an erroneous output on some "yes" instances due to the fact that when l_c is limited the enumerative step (2) may skip some candidate bipartitions $(\mathbf{a}, \mathbf{b})$. However, the observed error rate is so low that we consider the limit on l_c worthwhile for the sake of efficiency. On the other hand, if no limit is placed on l_c, then our algorithm will always produce a correct output, at the expense of possibly running in exponential time in the worst case.

In summary, our algorithm can be customized to run in polynomial time with satisfactory low error rates (see Section 4 for some evidence of error rates). Also note that it is a deterministic instead of a randomized algorithm.

4 Experiments

We mainly tested our C++ implementation of the decision algorithm with the parameter l_c customized as $l_c \le n = \ell(\mathbf{d})$. In certain situations when n is small, unlimited l_c is chosen to estimate approximately how large l_c should be for the algorithm to make no errors. The implementation is compiled with a reasonably recent g++ compiler with -O3 optimization and tested under typical Linux workstations with at least 16GB memory. We first show the low error rates of the algorithm and then show the good run time performance.

4.1 Error Rates

We first demonstrate the somewhat surprising power of the seven rules in the first phase. In Table 1 we show the number $r(n)$ of all zero-free graphical degree sequences of length n that can be resolved by one of these rules and their proportion among all $D(n)$ zero-free graphical degree sequences of length n. Based on the description of the rules, these $r(n)$ instances that can be resolved in phase one are all "no" instances. The function values $r(n)$ are obtained through a program that incorporates our decision algorithm into the algorithm to enumerate all degree sequences of a certain length from Ruskey et al. [14]. Let $B(n)$ be the number of zero-free potentially bipartite graphical degree sequences of length n. Clearly $B(n) \le D(n) - r(n)$ since some of the "no" instances are resolved in the second phase. It looks safe to conclude from this table that $\frac{r(n)}{D(n)}$ tends to 1 as n grows towards infinity and so $\frac{B(n)}{D(n)}$ tends to 0. Note that these are just empirical observations. Rigorous proofs of the asymptotic orders of these functions or their relative orders might require advanced techniques [17] and are out of our reach at this moment.

In fact, those instances that can be resolved by one of the seven rules are not the only ones that can avoid the enumeration phase of our algorithm. For example, those instances that have $S = \frac{|\mathbf{d}|}{2} - |\mathbf{a_f}| = 0$ can also be resolved immediately following the tests of the seven rules according to our description in Section 2.

Table 1: The number $D(n)$ of zero-free graphical degree sequences of length n, and the number $r(n)$ of them that can be resolved by one of the seven rules.

n	$D(n)$	$r(n)$	$\frac{r(n)}{D(n)}$
6	71	53	0.746479
7	240	203	0.845833
8	871	770	0.884041
9	3148	2902	0.921855
10	11655	10995	0.943372
11	43332	41603	0.960099
12	162769	158074	0.971155
13	614198	601556	0.979417
14	2330537	2295935	0.985153
15	8875768	8780992	0.989322
16	33924859	33663505	0.992296
17	130038230	129315300	0.994441
18	499753855	497745844	0.995982
19	1924912894	1919319963	0.997094
20	7429160296	7413535855	0.997897
21	28723877732	28680124185	0.998477
22	111236423288	111113621955	0.998896
23	431403470222	431058118392	0.999199

Next we demonstrate the low error rates of our algorithm. In Table 2 we show the number $B_w(n)$ of all zero-free potentially bipartite graphical degree sequences of length n that will be incorrectly reported as a "no" instance if we set $l_c = 1$ (the smallest possible chosen limit on the number of combinations) and their proportion among all $B(n)$ zero-free potentially bipartite graphical degree sequences of length n. Even with this smallest possible l_c, our algorithm makes very few errors on the "yes" instances. In fact, if we set $l_c = n$, then our algorithm makes no error on all zero-free graphical degree sequences of length $n \leq 23$. However, the observed trend suggests that the limit l_c need to grow with n for our algorithm to always produce correct outputs. For example, l_c must be at least 19 for all instances with $n \leq 23$ to be correctly decided and must be at least 91 for $n \leq 28$. We do not have more data like this for $n \geq 29$ since exhaustive enumeration when $n \geq 29$ will become too time-consuming and we are unable to prove whether there is any polynomial of n to bound l_c such that our algorithm can always give correct outputs or the error rate is always below some constant. If l_c grows faster than every polynomial of n or is unlimited, then our algorithm could run more than polynomial time in the worst case.

Table 2: The number $B(n)$ of zero-free potentially bipartite graphical degree sequences of length n, and the number $B_w(n)$ of them that will be misidentified as "no" instances if l_c is set to 1.

n	$B(n)$	$B_w(n)$	$\frac{B_w(n)}{B(n)}$ (with $l_c = 1$)
6	18	0	0
7	37	0	0
8	100	0	0
9	241	0	0
10	640	0	0
11	1639	0	0
12	4378	0	0
13	11601	2	0.000172399
14	31318	8	0.000255444
15	84642	32	0.000378063
16	230789	117	0.000506957
17	631159	482	0.000763674
18	1736329	1667	0.000960072
19	4790928	6107	0.00127470
20	13272233	20826	0.00156914
21	36869887	72879	0.00197665
22	102727688	244266	0.00237780
23	286893582	821331	0.00286284

We note that the error rates reported in Table 2 are with respect to the $B(n)$ "yes" instances. The error rate will be much lower if they are computed with respect to all $D(n)$ zero-free instances of length n because, as we know from Table 1, by far the majority of the "no" instances have already been correctly detected by the seven rules. For example, $\frac{B_w(23)}{D(23)} \approx 1.9 \times 10^{-6}$ at the setting of $l_c = 1$, which is a lot lower than $\frac{B_w(23)}{B(23)}$. Plus, increasing l_c from $O(1)$ to $O(n)$ also further reduces the error rate. For example, $\frac{B_w(23)}{D(23)} = 0$ at the setting of $l_c = 23$.

Regarding error rates of our algorithm for larger n, say $n \geq 50$ or $n \geq 100$, we do not have direct numeric estimates. However, we have reason to believe it will still be quite low. See the next subsection.

4.2 Run Time Performance

In the last subsection we demonstrated the low error rates of our algorithm mainly through exhaustive testing on inputs with small n. Run time performance test for small n is not informative since each individual input can always be decided almost instantly when $n \leq 50$. For larger n we resort to testing randomly generated instances. Basically, what we observe is that varied inputs might have quite different run times. The run time of our algorithm reported in this subsection are all obtained under the setting of $l_c = n$.

We have already shown in Section 3 that our algorithm runs in polynomial time if l_c is bounded by a polynomial of n. We want to generate random graphical degree sequences of specified length n, largest term d_1 and smallest term d_n. The control of the largest term d_1 and smallest term d_n allows us to be able to mainly focus on hard instances since, according to the seven rules used in phase one, random instances with d_1 and d_n in certain ranges are more likely to be resolved in phase one, which will be uninteresting since phase one can always be finished almost instantly for $n \leq 500$.

Tested random graphical degree sequences are generated by first sampling uniform random integer partitions of specified length n, largest term d_1 and smallest term d_n with an adaptation of a sampling algorithm in appendix A of Burns [2]. Such random partitions are then chosen as inputs to our algorithm if they pass the Erdős-Gallai conditional test [4], which is a criterion to decide whether an integer partition is a graphical degree sequence. We do not believe these random inputs to our algorithm will continue to follow the uniform distribution like those before they pass the Erdős-Gallai conditional test. However, they are still varied enough for us to observe a diverse range of run time behavior.

We use an automatic tester program and tested a total of 20000+ random instances. Each single run of the automatic tester program allows us to specify n, d_1, d_n and the number of random instances to test. For a wide range of $n \leq 500$, we found that the hardest instances for our algorithm are approximately in the range of $0.5n \leq d_1 \leq 0.6n$ and $1 \leq d_n \leq 0.1n$. The random instances in these ranges are the most likely to cause our algorithm to enter the enumeration phase. However, even the hardest instances we found in our random tests for $n \leq 500$ can be finished within about a couple of minutes, which are necessarily those inputs reported as "no" instances that also undergo the entire enumeration phase without any successful candidate bipartition being found. All the tested "no" instances that are decided in the first phase can be finished almost instantly, which agrees with the run time complexity analysis in Section 3. All of the tested "yes" instances detected in the enumeration phase can be decided in at most tens of seconds due to the empirical evidence that most of these "yes" instances have a successful candidate bipartition that can be found even if l_c were set to 1 (because our implementation additionally reports the number C of combinations actually used in enumerative step (2) when a successful candidate bipartition is found for a "yes" instance). This further strengths our belief that the algorithm has a low error rate for large n under the setting of $l_c = n$ because, if a "yes" instance is to be erroneously reported as a "no" instance, the actual number of combinations used in enumerative step (2) must exceed $l_c = n$. However, no reported "yes" instance in our tested random inputs with $n \geq 100$ was ever observed to use an actual number C of combinations in enumerative step (2) with $1 < C \leq n$, which suggests "yes" instances that require $l_c > n$ to be detected by our algorithm are probably quite rare. We do believe that such rare "yes" instances exist. They are only hard to be found through our random generation process.

In summary, our random tests of long inputs not only demonstrate good run time performance of our algorithm, but also give some heuristic evidence that error rates will continue to be low when n becomes large.

5 Discussions

We mentioned in Section 2 that our algorithm is customizable through the limit l_c in the enumerative step (2). In this section we describe several alternative designs of the enumeration phase that we have considered and their consequences. We also comment on the complexity of the decision problem itself.

5.1 Alternative Designs of the Algorithm

We note that the alternative designs we presented here are experimentally verified to be less efficient than the version in Section 2. The reader may skip this part if not interested.

In the enumerative step (1) we have chosen d_y from the largest possible value to smallest. Instead, we can choose d_y from the smallest possible value to largest. On average, we found that the former has better run time performance.

In the enumerative step (2) we prefer to enumerate the combinations of smallest terms first. Instead, we can choose to enumerate those of largest terms first. On average, we still found that the former has better run time performance.

The enumerative steps (1) and (2) can even be combined into one step to make the enumeration phase simpler. That is, we can exhaustively enumerate all possible combinations of terms from $\mathbf{d} - \mathbf{d}_{\mathbf{S}'}$ with an appropriate sum subject to the restrictions in the requirement 3 about the number of terms $\ell(\mathbf{a})$ in $\mathbf{a}$. (Or, to make it more naive, we could exhaustively enumerate all possible combinations of terms from $\mathbf{d} - \mathbf{a}_{\mathbf{f}}$ with the sum S.) With these schemes we still face the choice of enumerating largest terms first or smallest terms first. On average, the choice of "smallest terms first" still enjoys better run time performance. However, in order to achieve similar low error rates in these alternative schemes with this choice of "smallest terms first," the limit on the number of combinations to be generated will usually have to be much larger than the chosen limit l_c in our design in Section 2, causing these alternatives to have much worse run time performance on those instances that require the second phase to decide. If no limit is placed on the number of combinations to be generated, these alternatives will all produce correct outputs always. Nevertheless, the run time performance could become terrible. For example, for some hard instances with length n from 100 to 300, it could take days to detect a successful candidate bipartition for "yes" instances and tens of days to decide for "no" instances when no limit on l_c is imposed, a clear evidence of exponential run time behavior. For longer hard instances in the range $300 \leq n \leq 500$, these more naive enumeration phases with unlimited l_c might take years or longer time to finish.

As mentioned before, our algorithm always produces a correct conclusion for "no" instances. But it could give an incorrect output for some "yes" instances depending on the limit l_c set in the enumeration phase. This kind of behavior can be contrasted with some randomized decision algorithms. The error our algorithm might produce is *fixed* and it comes from the fact that not all potentially bipartite graphical degree sequences exhibit the kind of pattern that can be captured by the particular "limited" enumeration process of our algorithm. Simply put, our algorithm is deterministic. If it makes an error on an "yes" input under a particular setting of l_c, it always outputs an erroneous answer on that input under the same setting. If a randomized decision algorithm makes an error on an input, then it could produce a correct output the next time it runs on the same input. And usually the probability of correct output of a randomized decision algorithm is greater than one half and can be analyzed or estimated.

5.2 Regarding Problem Complexity

Now we comment on the complexity of the decision problem of potential bipartiteness of graphical degree sequences. It is obviously in NP since a successful candidate bipartition $(\mathbf{a}, \mathbf{b})$ can be verified in polynomial time with the Gale-Ryser condition. We don't know whether it is in co-NP or in P, nor do we know whether it is NP-complete since a reduction to prove its NP-hardness is not found.

In this paper we dealt with the decision problem of whether $\chi(\mathbf{d}) = 2$. In the case that $\mathbf{d}$ is not potentially bipartite (i.e. $\chi(\mathbf{d}) > 2$) and it is desired to compute $\chi(\mathbf{d})$, we can decide, for each successive fixed $k \geq 3$, whether there is a k-colorable realization of $\mathbf{d}$, until the answer becomes "yes." Each of these decision problems for fixed $k \geq 3$ is clearly also in NP and we conjecture them to be NP-complete.

6 Summary and directions for future research

We presented a fast algorithm to test whether a graphical degree sequence is potentially bipartite. The algorithm works very well in practice. It remains open whether the decision problem can be solved in polynomial time. The complexity of the decision problem of whether $\chi(\mathbf{d}) \leq k$ for fixed $k \geq 3$ also remains to be resolved.

References

[1] G. E. Andrews. *The Theory of Partitions.* Cambridge University Press, 1984.

[2] J. M. Burns. *The Number of Degree Sequences of Graphs.* PhD thesis, Massachusetts Institute of Technology. Dept. of Mathematics, 2007.

[3] Z. Dvořák and B. Mohar. Chromatic number and complete graph substructures for degree sequences. *Combinatorica*, 33(5):513–529, 2013.

[4] P. Erdős and T. Gallai. Graphs with given degree of vertices. *Mat. Lapok*, 11:264–274, 1960.

[5] O. Favaron, M. Mahéo, and J.-F. Saclé. On the residue of a graph. *Journal of Graph Theory*, 15(1):39–64, 1991.

[6] D. Gale. A theorem on flows in networks. *Pacific J. Math*, 7(2):1073–1082, 1957.

[7] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. I. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962.

[8] V. Havel. A remark on the existence of finite graphs. *Časopis pro pěstování matematiky*, 80(4):477–480, 1955.

[9] M. Krause. A simple proof of the Gale-Ryser theorem. *The American Mathematical Monthly*, 103(4):335–337, 1996.

[10] W. Mantel. Problem 28 (solution by H. Gouwentak, W. Mantel, J. Teixeira de Mattes, F. Schuh and W. A. Wythoff). *Wiskundige Opgaven*, 10:60–61, 1907.

[11] O. Murphy. Lower bounds on the stability number of graphs computed in terms of degrees. *Discrete Mathematics*, 90(2):207–211, 1991.

[12] N. Punnim. Degree sequences and chromatic numbers of graphs. *Graphs and Combinatorics*, 18(3):597–603, 2002.

[13] S. B. Rao. A survey of the theory of potentially P-graphic and forcibly P-graphic degree sequences. In Siddani Bhaskara Rao, editor, *Combinatorics and Graph Theory: Lecture Notes in Mathematics, vol 885*, pages 417–440. Springer Berlin Heidelberg, 1981.

[14] F. Ruskey, R. Cohen, P. Eades, and A. Scott. Alley CATs in search of good homes. In *25th S.E. Conference on Combinatorics, Graph Theory, and Computing*, volume 102, pages 97–110. Congressus Numerantium, 1994.

[15] H. J. Ryser. Combinatorial properties of matrices of zeros and ones. *Canadian Journal of Mathematics*, 9:371–377, 1957.

[16] R. P. Stanley. *Enumerative Combinatorics, Volume 2.* Cambridge University Press, 1999.

[17] K. Wang. Efficient counting of degree sequences. *Discrete Mathematics*, 342(3):888–897, 2019.

[18] J. H. Yin. A short constructive proof of A.R. Rao's characterization of potentially k_{r+1}-graphic sequences. *Discrete Applied Mathematics*, 160(3):352–354, 2012.