Palestine Journal of Mathematics

Vol 13(4)(2024) , 1444–1458 © Palestine Polytechnic University-PPU 2024

A New Characterisation of Total Graphs

Mahipal Jadeja, Rahul Muthu and Ravi Goyal

Communicated by Ayman Badawi

MSC 2010 Classifications: Primary 05C76; Secondary 05C85.

Keywords and phrases: Auxiliary Graphs, Total Graphs, Total Graph of Complete Graphs

The authors would like to thank the reviewers and editor for their constructive comments and valuable suggestions that improved the quality of our paper.

.

Corresponding Author: Mahipal Jadeja

Abstract Graphs constructed to translate some graph problem into another graph problem are usually called auxiliary graphs. Specifically, total graphs of simple graphs are used to translate the total colouring problem of the original graph into a vertex colouring problem of the transformed graph. In this paper, we obtain a new characterisation of total graphs of simple graphs. We also design algorithms to compute the inverse total graph when the input graph is a total graph. These results improve upon the work of Behzad (in terms of complexity and simplicity of the algorithm), by using novel observations on the properties of the local structure in the neighbourhood of each vertex. The earlier algorithm was based on BFS and distances. Our theorems result in an iterative partitioning of the vertex set of the total graph into the original graph (inverse total graph) and its line graph. We obtain direct constructive results for total graph of complete graphs.

1 Introduction

A total colouring of a simple graph is a simultaneous assignment of labels to its vertices and edges such that adjacent vertices get distinct colours, adjacent edges get distinct colours and the colour of each edge is distinct from the colours of its endpoint vertices (or equivalently the colour of each vertex is distinct from the colours of its incident edges). It is thus a combination of a proper vertex colouring, a proper edge colouring and a further restriction on the interplay between these colourings. The notion of total colouring was introduced by Behzad [1] and Vizing [2] and those papers also conjectured that $\chi_T(G) \leq \Delta(G) + 2$ Here, $\chi_T(G)$ denotes total chromatic number (the fewest colors needed in any total coloring of the graph) of G and $\Delta(G)$ denotes the maximum degree of the graph G. It is immediate that $\chi_T(G) \geq \Delta(G) + 1$, since a vertex of maximum degree and its incident edges must all get distinct colours. A lot of work has been done on total colouring [3, 4], based on frugal colouring [5], the list colouring conjecture [6] etc.

Definition 1. The total graph $T(G)$ of a graph $G = (V, E)$ has one vertex for each edge as well as each vertex in G as a vertex set. Two vertices in $T(G)$ are adjacent precisely when the elements (vertex or edge) of G they represent are adjacent/incident to each other in G.

Vertex colouring is a way of colouring the vertices of a graph such that no two adjacent vertices are of the same colour. Whereas, in the total colouring, no adjacent edges, no adjacent vertices and no edge and either end vertex are assigned the same colour. From the definition of the total graph, it is clear that total colouring of G becomes a vertex colouring of $T(G)$.

In this work, our aim is not to make headway in the total colouring problem but rather to get a complete characterisation of the class of total graphs of simple graphs. Work on characterising naturally defined classes of graphs has attracted the attention of researchers in the field of graph theory, and thus our work is an important contribution. Examples of work on characterising graph classes include planar graphs [7], line graphs [8], interval graphs [9], bipartite graphs [10], graph factoring under the cartesian product operation [11].

The vertex set of the total graph of a simple graph can be partitioned into two sets, one corresponding to the vertex set of the original graph (inverse total graph) and the other the line graph of the original graph, with crossing edges between these two vertex sets. A result characterising total graphs was obtained by Behzad [12] in 1970, and earlier works obtained interesting properties on this class of graphs [13]. The fact that any total graph has a unique preimage under the inverse total graph operator was proved in [14].

We obtain a new characterisation of total graphs based on the induced subgraphs on the neighbourhood of maximum degree vertices. These characterisations allow us to distinguish vertex vertices from edge vertices among the vertices of maximum degree. We also rely on the preponderance of maximal triangles (maximal cliques on exactly three vertices) between the vertex graph and the line graph consisting of two vertices from the vertex part and one vertex from the line graph part. Using this characterisation, we develop an efficient algorithm which iteratively creates the partition of the vertex set of the candidate total graph into its inverse total graph and line graph.

The paper is organised as follows. The detailed definitions of structures as well as notation is presented in Section 2. In the same section, basic properties and known results on total graphs are also presented along with our results for complete graphs. In Section 3, we present our results characterising the two classes of vertices and an algorithm for reconstructing the inverse total graph of a given total graph. Detailed comparison with the existing approach is presented in Section 4. We summarise our work and indicate possible future directions for research in Section 5.

2 Preliminaries

In this section we present some of the basic definitions and notation we use, throughout the paper. We also present known results related to total graphs. All graphs we consider are finite, simple, undirected and connected. Disconnected graphs do not add any new dimension to the problem.

Our main focus in this work is on total graphs. The vertex set of the total graph of a graph can be partitioned (uniquely upto isomorphism) into two parts such that the subgraph induced on one part is the original graph (inverse total graph) and the subgraph induced on the other is the line graph of the original graph. The bipartite subgraph induced by this partition joins a vertex in the original graph to a vertex (representing an edge) in the line graph if and only if the vertex is an endpoint of the edge. Our interest in line graphs is limited to their role in total graphs, and their role in the partitioning procedure to obtain the inverse total graph.

Definition 2. The **line graph** $L(G)$ of a graph $G = (V, E)$ is defined as the graph with vertex set having one vertex corresponding to each edge in G and an edge between two vertices of $L(G)$ precisely when the edges of G that those vertices correspond to, have a common endpoint.

Not all graphs are line graphs of a simple graph (or for that matter even of a multigraph). Similarly not all graphs are total graphs. Viewing the total graph concept as a function from the class of graphs to the class of graphs analogous to line graphs, the function is non-surjective. A natural problem, therefore, is to determine the range of this function. In addition, it is also interesting to design an algorithm that either reports that an input graph is not a total graph of any simple graph or returns the inverse total graph of the given total graph. It has been established that the total graphs viewed as a function from the class of graphs to the class of graphs is injective [14].

With reference to a total graph $T(G)$ we have a partition of its vertex set into two parts, inducing G and $L(G)$ as explained in the previous paragraph. For a total graph such a partition is unique upto isomorphism and is called a **valid partition**. The individual vertices belonging to these two parts in a valid partition are introduced in the following definition.

Definition 3. The vertex set of the total graph of a graph can be partitioned into:

(i) The vertices of the original graph (we call such a vertex a **vertex vertex**)

(ii) The vertices of the line graph (we call such a vertex an **edge vertex**).

Definition 4. A **mixed clique** in a total graph is a clique which has at least one vertex from the set of vertex vertices and at least one vertex from the set of edge vertices in a valid partition of the total graph.

Definition 5. A **pure clique** in a total graph is a clique consisting exclusively of vertex vertices or exclusively of edge vertices.

We show intuitive step by step process of construction of the total graph of the given graph. To visualise this, view any graph as a union of stars centred at each of its vertices. These stars translate into complete graphs with number of vertices equal to the degree of the central vertex of the corresponding star in the line graph portion (see Figure 1).Since we used a decomposition rather than a partition, some pairs of vertices from these cliques in the line graph will have non-empty intersection with respect to their neighbours in the original graph. Collapse them into identical vertices (See Figure 2). Connect the central vertex of each star to each vertex of its corresponding clique in the line graph. The resultant graph is the total graph of the given graph (See Figure 3).

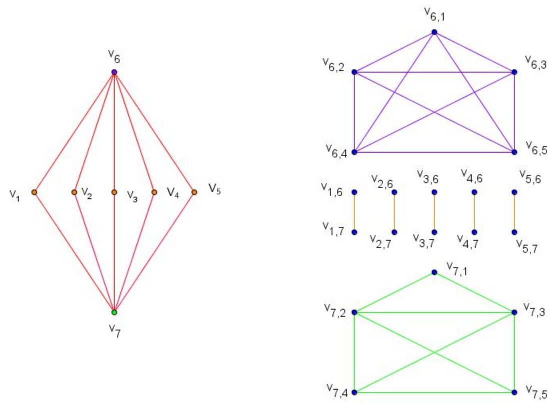

Figure 1: Line graph construction from the given graph

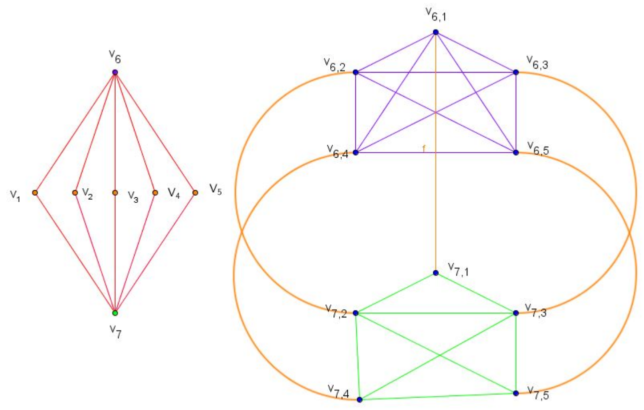

Figure 2: Line graph

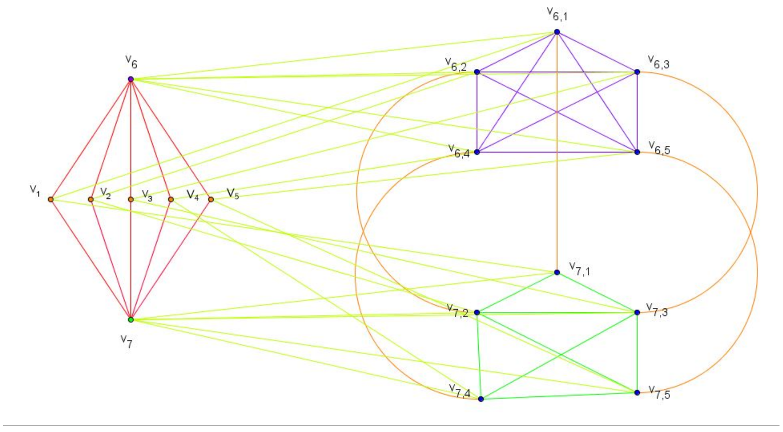

Figure 3: Total graph

We now present results expressing the degrees of vertices of a total graph in terms of the degrees of vertices in its inverse total graph. We denote the degree of a vertex u in a graph G by $d_G(u)$. The following two results are derived in [15]. Figures 4 and 5 illustrate these statements.

Lemma 2.1. *The degree of a vertex vertex in a total graph is 2 times the degree of the original vertex in the inverse total graph.*

Lemma 2.2. *The degree of an edge vertex in a total graph is equal to the sum of the degrees of the endpoint vertices of the original edge in the inverse total graph.*

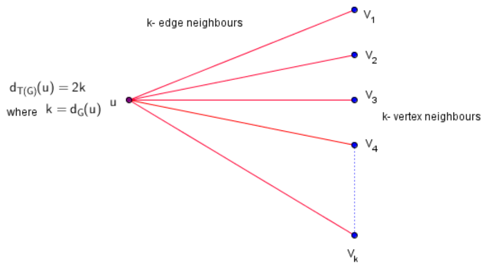

Figure 4: Degree characteristics of vertex-vertex

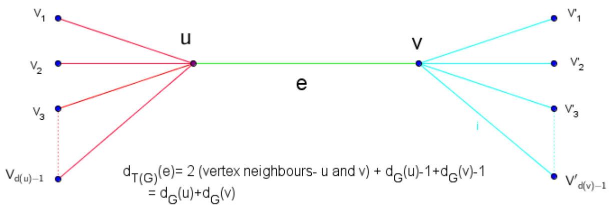
V1
V2
V3
Vd(u)-1
u
v
e
V'1
V'2
V'3
V'd(v)-1
$d_{T(G)}(e)$= 2 (vertex neighbours- u and v) + d_G(u)-1+d_G(v)-1
= d_G(u)+d_G(v)

Figure 5: Degree characteristics of edge-vertex

The following Lemma is an immediate consequence of Lemmas 2.1 and 2.2 and taken from [13].

Lemma 2.3. *The total graph of a graph is regular if and only if the original graph is regular.*

We list down results expressing the number of vertices and edges of a total graph in terms of the corresponding parameters of the inverse total graph [16].

Lemma 2.4. *Let $T(G) = (V', E')$ for the given $G = (V, E)$.*

(i) $|V'| = |V| + |E|$

(ii) $|E'| \leq |E|(|V| + 1)$ *(after simplifying the original experssion $|E'| = 2|E| + \frac{1}{2}\sum d_i^2$ where d_i represents vertex degrees of G)*

Results on Complete Graphs

The following lemmas give the structure and other parameters of the total graphs of complete graphs. Lemma 2.5 is taken from [17].

Lemma 2.5. *Let $T(G) = (V', E')$ for the given $K_{|V|}$.*

(i) $|V'| = \frac{|V|(|V|+1)}{2}$

(ii) $|E'| = \frac{|V|(|V|-1)(|V|+1)}{2}$

(iii) $\forall v \in T(G), d_{T(G)}(v) = 2(|V| - 1)$.

Isomorphism between the line graph of a complete graph and the total graph of a complete graph with one fewer vertex is well established [18, 19].

Lemma 2.6. $L(K_n) = T(K_{n-1})$ *where $L(K_n)$ denotes the line graph of K_n.*

Our proposed characterisation assumes that the given input graph is not the total graph of cycle graph or complete graph. The same assumption was also made by Behzad [12]. Hence, we present an efficient algorithm to identify whether the input graph is the total graph of a complete graph or not. This algorithm helps us in eliminating the special case of complete graph while characterising. **Algorithm 1: Is a given graph the total graph of K_n**

(i) From Lemma 6, $T(K_n) = L(K_{n+1})$. For the given input graph, obtain its inverse line graph in $O(E')$ [20].

(ii) If the obtained inverse line graph is K_{n+1} then the G' is $T(K_n)$ (complexity of this step $O(E')$).

Since the total graph of a complete graph is considered as a special case, it is interesting to explore its properties. Also, any total graph is a subset of the total graph of complete graph (for some value p, $T(G) \subseteq T(K_p)$). Hence, it is possible to construct a total graph of any graph (with n vertices) by constructing the total graph of K_n first and then, removing zero or more number of edges from it. With this motivation, we present an elegant direct construction method for the total graphs of complete graphs.

(i) Consider $n + 1$ disjoint groups each consisting of exactly n vertices. (These correspond to the $n + 1$ cliques of size n of the total graph of the K_n).

(ii) The vertices in i^{th} group G_i are labelled $\{1, \ldots, n + 1\} \setminus \{i\}$. Therefore, $|G_i| = n$.

(iii) For each G_i, construct K_n by connecting all its n vertices pairwise.

(iv) Combine the j^{th} vertex of group i and the i^{th} vertex of group j into a single vertex. The neighbourhood of the new vertex is the union of the individual neighbourhoods. The degree of each new vertex is exactly $2(n - 1)$ because degree of each original vertex is $n - 1$.

(v) The resultant graph is the total graph of the complete graph K_n.

The following observations explain why the proposed method is consistent with Lemma 2.5 (point 2).

- No edge is destroyed during the entire procedure.
- Each clique has $\binom{n}{2}$ edges and initially $n + 1$ distinct cliques are considered.
- Thus, $\binom{n}{2} \times (n + 1)$ edges are present in the graph which remains constant through the course of this construction.

Construction of the total graph of K_3 using this direct method is shown in Figures 6 and 7. In the figure the group number is written as subscript for each vertex and the vertex number within the group is written in normal font. Note: For the rest of the paper, we assume that the input graph is not the total graph of complete or cycles graphs. The same assumption was also made by Behzad [12] also. These cases are considered as special cases and they are also characterised separately [13]. Other significant contributions related to total graphs are documented in [21, 22].

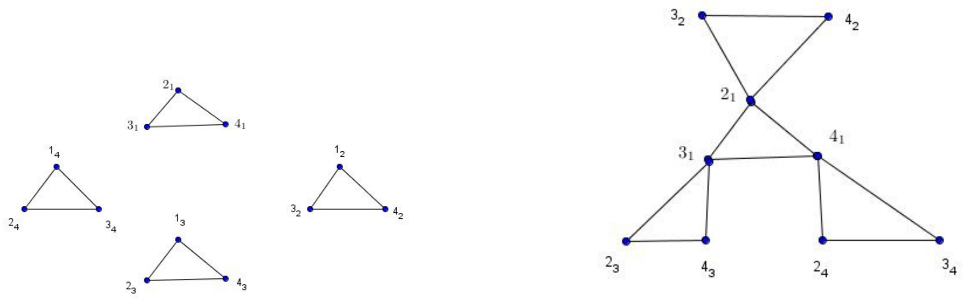

Figure 6: Initial steps of direct construction of the total graph of K_3

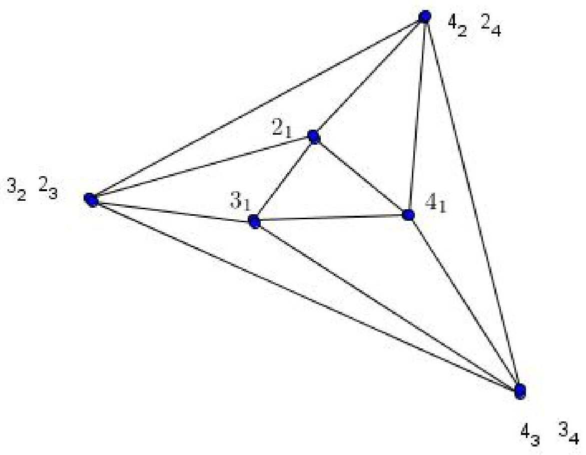

Figure 7: Resultant Total Graph of K_3 after step 4

3 Main Results

As explained earlier, an algorithm for computing the inverse total graph of a given graph (if it is a total graph) is based on finding a valid partition of the vertex set. The following propositions together form the building blocks of this partition algorithm.

Proposition 1. The largest mixed clique consisting of at least two vertex vertices in a total graph is of size 3.

Proof. Consider a clique consisting of three vertex vertices. Clearly there is no edge in any graph incident to three distinct vertices. Hence this clique cannot be augmented to include any edge vertices, and thus is not the subset of any mixed clique. It follows that a mixed clique can consist of at most two adjacent vertex vertices. In this case this can be augmented by only one vertex, the edge vertex representing the link between these two adjacent vertex vertices. □

As a corollary, we have the following result:

Corollary 3.1. *A maximal mixed clique is either:*

(i) A vertex vertex and all its adjacent edge vertices; or

(ii) Two adjacent vertex vertices and the connecting edge vertex.

In summary, a maximal mixed clique is either of size 3 or of size $k+1$ where $2k$ is the degree of its only vertex vertex in the total graph. In fact, every edge of the original graph (inverse total graph) gives rise to a unique maximal mixed clique of size 3 involving one edge vertex and two vertex vertices. This is together with its two end points in the original graph. Such a triangle has two of its edges in the bipartite subgraph of the total graph induced by a valid partition and the third edge is between the two vertices in the vertex part. These are the maximal triangles we referred to in Section 1.

The next result follows from Definition 4 and Proposition 1.

Proposition 2. Consider a maximal mixed clique of size 3 in a total graph with two vertex vertices. Let the two vertex vertices be of degrees a and b, with $a \geq b$. Then the degree of the edge vertex of this clique is $c = \frac{a+b}{2}$. Clearly $a \geq c \geq b$ with equalities holding if and only if $a = b$.

One can also immediately infer the following proposition.

Proposition 3. In the inverse total graph there are two adjacent vertices of maximum degree if and only if there is a maximal mixed clique of exactly three maximum degree vertices two from the vertex part and one from the line graph part in the total graph.

We will use this property extensively in our algorithm. If there is no triangle of vertices of maximum degree, a maximum degree vertex along with two of its neighbours with degree in arithmetic progression constitutes a mixed triangle with the highest and lowest degree vertices among them being from the vertex part and the mean value degree vertex from the edge part. This fact significantly reduces the work involved in finding a partition of the vertex set of the total graph into the vertex part and the line graph part iteratively.

Proposition 4. Given a vertex vertex v_a of degree $2k$ in a total graph, its neighbours can be divided into k pairs representing:

- Its distinct incident edges $(v_1, \ldots, v_k)$ in the inverse total graph and
- the other endpoints of those edges $(v'_1, \ldots, v'_k)$ respectively.

The k pairs are $\{(v_1, v'_1), \ldots, (v_k, v'_k)\}$.

Proof. See Figure 8. The degrees of each pair is related to the degree of the selected vertex vertex according to Proposition 2.

□

Now we prove two propositions which give conditions for a maximum degree vertex in a candidate total graph to be a vertex vertex or an edge vertex. These propositions are used to iteratively find a maximum degree vertex vertex (one is guaranteed to exist by Proposition 2) and partition its neighbours into vertex vertices and edge vertices. At each round a maximum degree vertex vertex is selected as a part of the inverse total graph and it along with its edge vertex neighbours are eliminated to get a smaller graph to recurse on. A partition of the vertex set of the given graph into the inverse total graph and the line graph of the inverse total graph is created iteratively, if one exists; or we infer that no such partition exists if there is a violation of the combinatorial conditions at any iteration. Our propositions in this section work only for total graphs of non-complete graphs. Thus the algorithm developed in this section also uses the algorithm of Section 2, when appropriate, to handle the case of complete graphs.

Proposition 5. An arbitrary maximum degree vertex v (of H) of degree $2k$, is a vertex vertex, if and only if the following properties hold in the subgraph induced on its open neighbourhood.

(i) Its neighbours can be divided into two disjoint and exhaustive groups of k vertices each, one corresponding to its vertex neighbours $N^{\mathcal{V}}(v)$ in the inverse total graph and the other corresponding to its incident edges $N^{\mathcal{E}}(v)$ in the inverse total graph.

(ii) The maximum degree of $G[N^{\mathcal{V}}(v) \cup N^{\mathcal{E}}(v)]$ is k. This number is achieved by each vertex in $N^{\mathcal{E}}(v)$ and at least one vertex of $N^{\mathcal{V}}(v)$ falls short of this degree.

Proof. The first statement follows from Proposition 4.
Vertices present in $N^{\mathcal{E}}(v)$ are corresponding to k edges incident on v and hence all the k edge vertices will be pairwise adjacent to one another (from the definition of total graph).
Vertices present in $N^{\mathcal{V}}(v)$ are corresponding to k vertex neighbours of v and if, all these vertices achieve degree k, then the graph in the vertex part becomes complete graph. This is because if $G(N^{\mathcal{V}}(v))$ is k-regular then all the vertices are saturated in terms of degree (if a vertex v' is present with unsaturated degree p, then $d_G(v') = k + p$ and hence $d_{T(G)}(v') = 2(k + p)$ which contradicts the assumption of $2k$ maximum degree) and hence, no more vertices can be present in the vertex part. We assume that the given graph is not complete and hence, at least one vertex from $N^{\mathcal{V}}(v)$ falls short of this k degree.

□

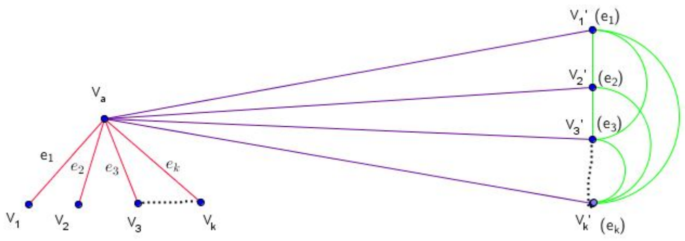

Figure 8: Characteristics of Vertex-Vertex

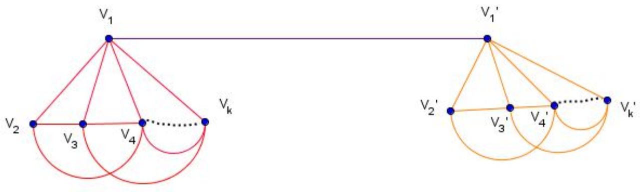

Figure 9: Characteristics of Edge-Vertex

Proposition 6. An arbitrary maximum degree vertex v (of H), of degree $2k$, is an edge vertex, if and only if the following property holds in the subgraph induced on its open neighbourhood.

(i) Its neighbours can be partitioned into 2 maximal cliques of exactly k vertices each. Each of these cliques consist of one vertex vertex and $k - 1$ edge vertices.

Proof. The statement follows from an inspection of Figure 9.

□

In 4-regular candidate total graphs (total graphs of cycles), each vertex satisfies the conditions of both the above propositions. That is because the two parts of any valid partition are isomorphic to each other.

Theorem 3.2. *Given any input (candidate total) graph H, if it is indeed a total graph then the graph H' is a total graph where H' is obtained from H by eliminating a vertex with maximum degree (satisfying the criteria of Proposition 5) and all its edge neighbours (edge vertices).*

Proof. Let v be the vertex satisfying Proposition 5. If the given input graph H is the total graph of some graph (say G), then removing v, along with its edge neighbours will effectively remove the vertex vertex v and edge vertices incident on v. This effectively eliminates the vertex and the incident edges from the inverse total graph G, converting it into G'. Clearly, $H' = T(G')$.

□

Note: Even if the input graph is not total, it can satisfy the criteria of Proposition 5 (and/or Proposition 6) in some iteration(s), but eventually, the condition(s) will be violated (see Algorithm 4 and its correctness proof for more details).
Algorithm 2: **Algorithmic version of Proposition 5**
Input: Consider a maximum degree vertex vertex x of a candidate total graph H
Output: Classification of neighbours of x into vertex neighbours (vertex vertices) and edge neighbours (edge vertices)

(i) x has a neighbour with degree less than $d_H(x)$ (since H is not a total graph of a complete graph/cycle graph).

(ii) Consider a neighbour of x, y such that $d_H(y) < d_H(x)$ but there is no vertex z in H such that $d_H(y) < d_H(z) < d_H(x)$. In other words, consider a neighbour of x having second highest degree in H. Classify such a y as an edge vertex (from Proposition 2).

(iii) Identify a common neighbour of x and y (say z) where $d_H(y) = \frac{d_H(z)+d_H(x)}{2}$. Classify z as a vertex vertex (from Proposition 2).

(iv) Find all common neighbours of x and y (except z) and classify them as edge vertices (this is because y is a part of a clique containing k vertices and x is adjacent to all the vertices of this clique).

(v) Classify remaining vertices of the neighbourhood of x as vertex vertices.

Theorem 3.3. *If G is not a complete graph and $H = T(G)$ then Algorithm 3 correctly identifies a vertex vertex of maximum degree.*

Proof. **Algorithm 3: Find Max Vertex Vertex**.
Input: The input graph H
Output: $v \in V(H)$ (where v is maximum degree vertex vertex)
The input graph has a partition of its vertex set into vertex vertices and edge vertices.

(i) We find a maximal triangle of maximum degree vertices if one exists (Proposition 3). From this triangle, we identify one vertex which satisfies Proposition 5 and return such vertex as maximum degree vertex vertex v.

(ii) If there is no triangle of maximum degree vertices, then it is guaranteed that any vertex of maximum degree, say v, is a vertex vertex by Proposition 2 and Corollary 3.1. Verify if it is indeed a vertex vertex using the Proposition 5 and return v.

As discussed earlier, for the maximum degree vertex vertex v only two cases are possible: 1) v is a part of a maximum degree triangle (where all 3 vertices are having maximum degrees- Proposition 3) or 2) v is a part of a triangle (where degrees in arithmetic progression- Proposition 2). Both cases are considered in Algorithm 3 and the identified vertex v is further verified with respect to structural properties with the help of Proposition 5.

□

In H, for the maximum degree vertex vertex (satisfying the conditions of Proposition 5), partition its neighbours into vertex vertices and edge vertices. We are left with the (candidate) total graph of the graph with one vertex deleted i.e. H'. Hence by recursing on the smaller graph, we can obtain the partition or conclude that one does not exist if at some iteration there is a maximum degree vertex violating both Proposition 5 and Proposition 6. We thus have an algorithm which starts with a candidate total graph of a non-complete graph and decides whether it is indeed a total graph by recursing on the smaller graph or recurse to the case of complete graph.

Algorithm 4: Inverse Total Graph.
Input: The input graph H
Output: The inverse total graph G (where $T(G) = H$) if the input graph H is total. Else conclude that the input graph H is not a total graph.

(i) **Check if** the given graph is the **total graph of a complete graph** using Algorithm 1. If so augument the vertices of that complete subgraph to the vertices obtained in earlier iterations and return.

(ii) Else consider maximal triangles containing three vertices of maximum degree using Proposition 3. If a such a triangle exists then find a maximal clique containing all these three vertices. Some vertex in this clique must be a maximum degree vertex vertex. Identify such a vertex x using Proposition 5. In the process of examining these vertices, if any of them violates both Propositions 5 and 6 or no such x exists, conclude that the input graph is not a total graph.

(iii) If no triangle of maximum degree vertices were found in step 2 then find a vertex x of maximum degree involved in a triangle satisfying Proposition 2. Check that x satisfies Proposition 5 and if not, conclude that the input graph is not a total graph.

(iv) If such an x was found in step 2 or 3, then partition its neighbours into vertex vertices and edge vertices by the algorithmic version of Proposition 5.

(v) Add x to the vertex set of the inverse total graph and repeat the steps with the graph obtained by deleting x and its edge neighbours.

(vi) Return the set of vertices (and the induced subgraph on them) accumulated in Step 5 over all the iterations.

Correctness Proof:
Part 1: The algorithm 4 correctly identifies the non-total input graph.
Proof: In the case of non-total graph, lets say, in some iteration i, a vertex x is the maximum degree vertex.

(i) If $d_H(x)$ is odd then the input graph is not a total graph. This case is handled by step 2 and 3 of the algorithm (since both the steps are using Proposition 5 and Proposition 6 where whether the maximum degree is even or not is checked first).

(ii) If $d_H(x)$ is even then, there are two cases possible:a) The candidate graph may mimic as a total graph for the iteration i and in this case, after obtaining valid partition, the algorithm will be executed for the next j iterations. b) After $i + j$ iterations (where $j \geq 0$), lets say the (new) maximum degree vertex is y ($d_H(y)$ is even and $y = x$ if $j = 0$). Since the input graph is not total, it is guaranteed that valid partition with respect to vertex y is not possible and hence structural conditions of Proposition 5 and/or Proposition 6 will be violated and the algorithm correctly handles this case too (in steps 2 and 3).

Part 2: The algorithm 4 correctly identifies the total (input) graph.
The algorithm's invariant is based on Proposition 3.2 (step 5). All the steps are correct because of correctness of Algorithm 1 (step 1), algorithmic version of Proposition 5- Algorithm 2 (step 4), Proposition 3 (step 2), Proposition 2(step 3), Proposition 5 (steps $2 - 3$), and Proposition 6 (step 2).

An example of the application of the algorithm

Consider a candidate input graph as shown in figure 10. We will classify vertices into vertex vertices (to be highlighted by red colour) and edge vertices (to be highlighted by green colour). In the first step, the maximum degree vertex B is identified as vertex vertex. Neighbours of B are classified as per the algorithmic version of Proposition 5. E is one of the neighbour of B having second highest degree in H and hence, we classify it as an edge-vertex. A satisfies the criteria of Proposition 2, so we classify it as a vertex vertex. F and I are common neighbours (except A) of B and E. Hence, we classify them as edge vertices (see Figure 11. Remaining neighbours of B i.e. C and D are classified as vertex vertices. B is removed along with its edge neighbours (neighbouring edge vertices: E, F and I). The modified graph H' is shown in Figure 12. C has the highest degree in H' and it is already classified as vertex vertex. H is one of the neighbour of C having second highest degree in H' and hence, we classify it as an edge-vertex. G is common neighbour H and C (except A) and hence, we classify it as edge vertex. All the vertices of H are correctly partitioned and hence, we stop and conclude that the candidate input graph is a total graph.

Runtime analysis of Algorithm 3:

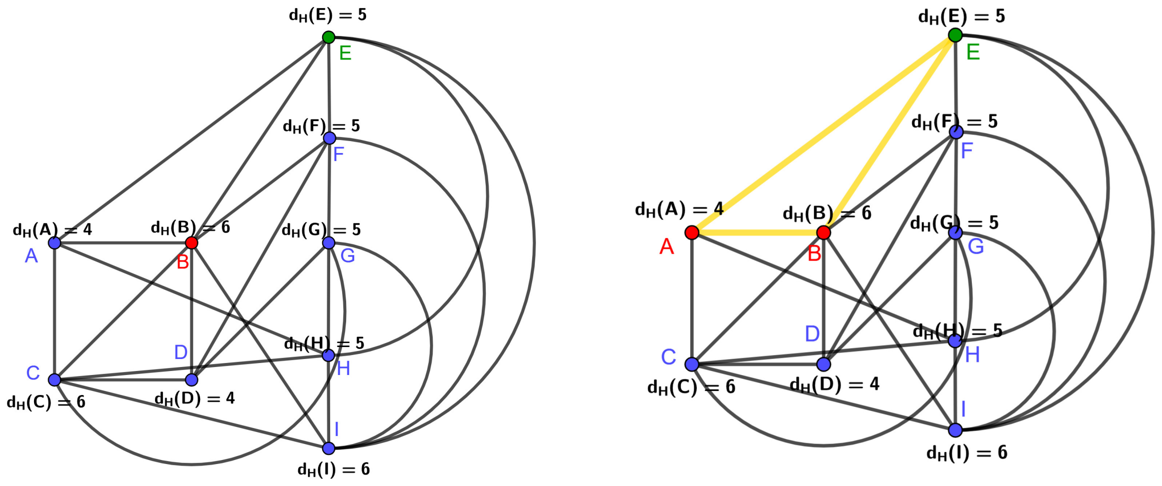

Figure 10: Input Graph and Classification of vertices A, B and E

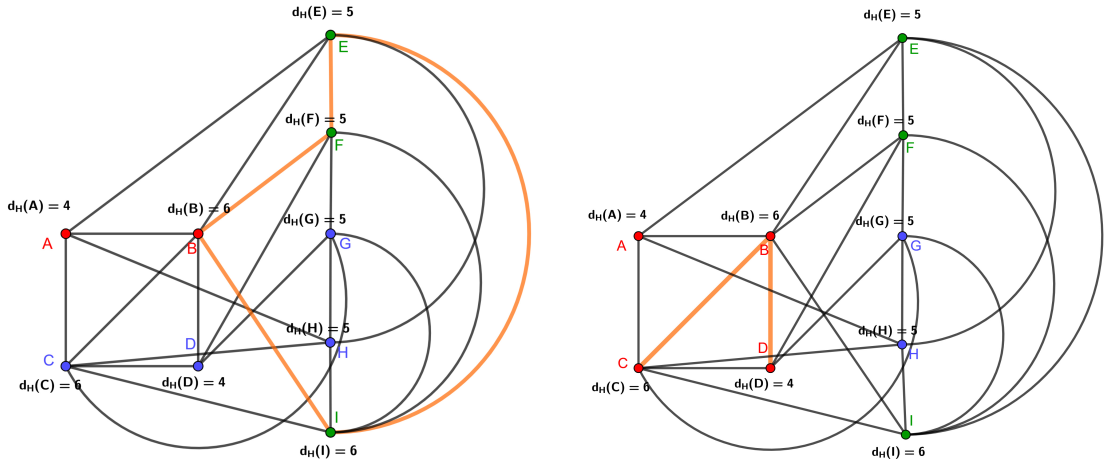

Figure 11: Classification of the neighbourhood of B into edge vertices and vertex vertices

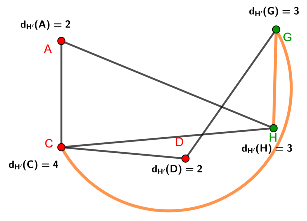

Figure 12: Classification of the neighbourhood of C into edge vertices and vertex vertices

Assume input graph is $G'(V', E')$ and the representation of G' is adjacency list representation.
Pre-processing:
Find the degrees of all the vertices of the graph in $O(|E'| + |V'|)$ and sort all the vertices in non-increasing order of their degrees in $O(|V'|log|V'|)$.

Note: In each iteration, we identify exactly one vertex vertex and hence after $|V|$ iterations, all the vertex vertex (and hence the inverse total graph) will be found.

(i) Step 1: Equivalent to complexity of Algorithm 1: $O(|E'|)$

(ii) Step 2: Check for Proposition 4, $O(1)$. Proposition 6, $O(2k)$ for identification of neighbours and $O(|k^2|)$ for checking condition 2 of Proposition 6. Same amount of time is required for checking conditions of Theorem 3.2. At most p times check for Propositions 6 and Theorem 3.2 (where p is the size of the clique containing maximum degree vertices).

  Total complexity of this step : $O(|V| * |E|) + O(|E'| - |E|)$

  Note: After each iteration, a maximum degree vertex vertex is removed along with its edge vertex neighbours and hence $|E'| - |E|$ edges will be considered exactly once over all iterations in the worst case.

(iii) Step 3 and 4: We can use algorithmic version of Proposition 6 for identification of all vertex vertex and edge vertex neighbours of v and also for partitioning, which will take $O(|V| * |E|) + O(|E'| - |E|)$ time in total (similar to the previous step) Verify Proposition 3 in $O(1)$ after getting the required partition.

(iv) Step 5: Repeat steps $2, 3, 4$ and 5 for $O(|V|)$ times. Already considered.

(v) Step 6: $O(|V|)$ time.

After each iteration, for each deleted vertex w, the degrees of neighbours of w will be decremented by 1. After each decrement in the degree value, in order to maintain the sorted list of degrees, the vertex degree which is decremented will be compared with the immediate next degree (in the sorted list) and if required a swap will be done. In total, $|E'|$ edges will be removed and hence, $O(|E'|)$ operations are required for re-computation of degrees and maintaining the sorted list of degrees.

The complexity of Algorithm 3 is: $O(|V| * |E|)$ since $O(|V| * |E|)$ is the dominant term.

4 Comparing with the existing characterisation

Behzad's characterisation uses a similar type of terminology with respect to vertices. In his work, vertex vertices are called special vertices and edge vertices are called non-special vertices.

The first step in Behzad's approach is consideration of a non special vertex v and classify all the vertices of the graph in $\{i\}$ where $i = 0, 1, 2, ...n$ denote the distance. i.e. vertex u will be in class i if and only if distance between v and u is i.

The core idea behind Behzad's algorithm is to identify special vertices for each class $\{i\}$ where $i = 0, 1, 2, \dots n$ based on the following key observations [12]:

- With respect to a non special vertex v, each nonspecial vertex of graph in $\{i\}$ is adjacent with exactly two special vertices of graph both of which are in $\{i\}$ or one in $\{i\}$ and the other in $\{i + 1\}$.
- Every $\{i\}$ (where $1 \leq i \leq (n - 1)$) contains both special and non-special vertices of graph while the class $\{n\}$ can not solely contain some non special vertices.
- Each nonspecial element in i for $i \geq 1$ is adjacent with at least one nonspecial element in $i - 1$ and each special element in i for $i \geq 2$ is adjacent to at least one special element in $i - 1$.

Assume that we already have all the classes $\{i\}$ with respect to a non special vertex v. Let S denotes the complete set of special vertices and N denotes the complete set of non-special vertices. The objective is to compute S_o, S_1, $\dots S_n$ where S_i denotes a set of special vertices corresponding to class $\{i\}$.

Clearly, $S_0 = \phi$ and $N_0 = \{v\}$. From the structure of the total graph, we can say that v is adjacent to exactly two special vertices, say u_1 and v_1 and hence $S_1 = \{u_1, v_1\}$ and $N_1 = \{1\} - S_1$. Since each non-special element of $\{2\}$ is adjacent to at least one non-special element of $\{1\}$, we can conclude that $N_1 \neq \phi$.

Let $w_1 \in N_1$. w_1 is adjacent with one of u_1 and v_1, say u_1 and a special element say u_2 which is in $\{2\}$. Since there is a unique special vertex u_2 which is adjacent with both w_1 and u_1, u_2 can be uniquely determined. By repeating the same argument for all the elements of N_1, we can obtain a set S_2.

Using above mentioned procedure we can compute S_3, $S_4 \ldots S_n$ (Separation of special and non-special elements of $\{i\}$). $S = S_0 \cup S_1 \cup \ldots S_n$. The induced subgraph on S is the original graph of the given total graph.

The complexity of Behzad's Algorithm depends upon the initial guess about v(a non special vertex). If the initial guess is correct then one can apply BFS in order to efficiently compute classes $\{i\}$ and then based on distance related observations, one can compute S.The time complexity of BFS is O($|V'| + |E'|$). In at most $|V'| - |V|$ iterations, all the special vertices will be discovered. Notice that, $|V'| - |V| = |E|$ where $|E|$ denotes total number of edges in the inverse total graph.

In each iteration, O($|V'|$) comparisons are required in order to identify a special vertex based on distances and common neighbour criteria (assuming adjacency matrix representation of the graph). So total time complexity is O($|V'|+|E'|$)+ O($|V'|\times|E|$) and hence the overall complexity is O($|V'| \times |E|$).

As discussed earlier, the complexity of our proposed algorithm is is O($|V| \times |E|$) which is better than Behzad's approach.

Further, as compared to the Behzad's characterisation, our characterisation is simpler due to following reasons:

- Behzad's approach only provides guidelines for the selection of the very first vertex v as a non-special vertex and u_1 and v_1 as special vertices. The algorithm vaguely mentions about suitable non-special vertex v and special vertices (u_1 and v_1). In the case of wrong choice of v, multiple calls of the Behzad's algorithm are required in order to correctly identify the inverse total graph or to declare that the graph is not a total graph of any simple graph. The paper [12] itself mentions that the procedure for determining whether the given graph is total or not is long. The approach also does not discuss ways to eliminate possibilities for u_1 and v_1. This is mainly due to missing characterisation for a special as well as non-special vertex.

- We have provided characterisations for a maximum degree vertex vertex (special vertices) and a maximum degree edge vertex (non special vertices), using which it is clear what is to be done in each step precisely. There is no ambiguity in our proposed algorithm.

- Behzad's algorithm does not consider cases where the given graph is a total graph of a cycle graph or a complete graph whereas our approach considers those cases too.

5 Conclusions & Future Directions

We have proved properties of the vertex degrees of total graphs. We have developed a precise characterisation of the structure of the neighbourhoods of maximum degree vertices of the total graph of any graph. Combining these results we have designed an efficient iterative algorithm to compute the inverse total graph of a candidate total graph, or report that the graph is not a total graph. We also present a direct construction for the total graphs of complete graphs. One interesting direction of future research is to see if a given n and m pair admits a connected unique total graph if any. One can also look at minimum number of dynamic graph operations (adding/deleting vertices and edges or moving edges around the graph) to transform a non-total graph into a total graph.

References

[1] M. Behzad, *The total chromatic number of a graph: A survey*, Combinatorial Mathematics and its Applications, 1–8 (1971).

[2] V. G. Vizing, *Some unsolved problems in graph theory*, Russian Mathematical Surveys, **23(6)**, 125–141 (1968).

[3] H. Hind, M. Molloy, and B. Reed, *Total colouring with* $\Delta +$ *poly (log* Δ*) colours*, (2000).

[4] M. Behzad, *Graphs and their chromatic numbers*, Ph.D. dissertation, Michigan State University, (1965).

[5] H. Hind, M. Molloy, and B. Reed, *Colouring a graph frugally*, Combinatorica, **17(4)**, 469–482 (1997).

[6] B. Reed, *The list colouring constants*, J. Graph Theory, **31(2)**, 149–153 (1999).

[7] S. Mac Lane et al., *A structural characterization of planar combinatorial graphs*, Duke Math. J., **3(3)**, 460–472 (1937).

[8] J. Krausz, *Démonstration nouvelle d'une théorème de Whitney sur les réseaux*, Mat. Fiz. Lapok, **50**, 75–85 (1943).

[9] F. Gardi, *The Roberts characterization of proper and unit interval graphs*, Discrete Math., **307(22)**, 2906–2908 (2007).

[10] A. S. Asratian, T. M. Denley, and R. Häggkvist, *Bipartite graphs and their applications*, Cambridge University Press, **131**, (1998).

[11] W. Imrich and I. Peterin, *Recognizing Cartesian products in linear time*, Discrete Math., **307(3)**, 472–483 (2007).

[12] M. Behzad, *A characterization of total graphs*, Proc. Am. Math. Soc., **26(3)**, 383–389 (1970).

[13] M. Behzad and H. Radjavi, *Structure of regular total graphs*, J. Lond. Math. Soc., **1(1)**, 433–436 (1969).

[14] M. Behzad and H. Radjavi, *The total group of a graph*, Proc. Am. Math. Soc., 158–163 (1968).

[15] M. Behzad and G. Chartrand, *Total graphs and traversability*, Proc. Edinb. Math. Soc. (Series 2), **15(2)**, 117–120 (1966).

[16] F. Harary, *Graph Theory*, Addison-Wesley Publishing Company, (1969).

[17] A. K. Joseph, *Few results on total graph of complete graph*, Int. Adv. Res. J. Sci. Eng. Technol., **6**, 72–78 (2019).

[18] M. Behzad, G. Chartrand, and E. Nordhaus, *Triangles in line-graphs and total graphs*, Indian J. Math., **10**, 109–120 (1968).

[19] D. M. Cvetković, *Spectrum of the total graph of a graph*, Publ. Inst. Math. (Beograd), **16(30)**, 49–52 (1973).

[20] P. G. H. Lehot, *An optimal algorithm to detect a line graph and output its root graph*, J. ACM, **21(4)**, 569–575 (1974). [Online]. Available: http://doi.acm.org/10.1145/321850.321853

[21] K. Nazzal, *Total Graphs Associated to a Commutative Ring*, Palestine J. Math., (2016).

[22] T. T. Chelvam and M. Balamurugan, *On the generalized total graph of fields and its complement*, Palestine J. Math., **7(2)**, (2018).

Author information

Mahipal Jadeja, Malaviya National Institute of Technology Jaipur, Jaipur, India.
E-mail: `mahipaljadeja.cse@mnit.ac.in`

Rahul Muthu, Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India.
E-mail: `rahul_muthu@daiict.ac.in`

Ravi Goyal, Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India.
E-mail: `ravi_goyal@daiict.ac.in`

Received: April 27, 2023.
Accepted: June 19, 2024.