mathematics

MDPI

Article

Use the *K*-Neighborhood Subgraphs to Compute Canonical Labelings of Graphs

Jianqiang Hao [1,*], Yunzhan Gong [2], Jianzhi Sun [1] and Li Tan [1]

[1] Beijing Key Laboratory of Big Data Technology for Food Safety, Beijing Technology and Business University, No. 11, Fu Cheng Road, Beijing 100048, China

[2] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, No 10, Xitucheng Road, Haidian District, Beijing 100876, China

* Correspondence: Bshjq@vip.163.com; Tel.: +86-10-6898-5704

Received: 5 July 2019; Accepted: 27 July 2019; Published: 1 August 2019

Abstract: This paper puts forward an innovative theory and method to calculate the canonical labelings of graphs that are distinct to *Nauty*'s. It shows the correlation between the canonical labeling of a graph and the canonical labeling of its complement graph. It regularly examines the link between computing the canonical labeling of a graph and the canonical labeling of its *open k-neighborhood subgraph* . It defines *diffusion degree sequences* and *entire diffusion degree sequence* . For each node of a graph G, it designs a characteristic *m_NearestNode* to improve the precision for calculating canonical labeling. Two theorems established here display how to compute the first nodes of $MaxQ(G)$. Another theorem presents how to determine the second nodes of $MaxQ(G)$. When computing $C_{max}(G)$, if $MaxQ(G)$ already holds the first i nodes $u_1, u_2, \cdots, u_i$, Diffusion and Nearest Node theorems provide skill on how to pick the succeeding node of $MaxQ(G)$. Further, it also establishes two theorems to determine the $C_{max}(G)$ of disconnected graphs. Four algorithms implemented here demonstrate how to compute $MaxQ(G)$ of a graph. From the results of the software experiment, the accuracy of our algorithms is preliminarily confirmed. Our method can be employed to mine the frequent subgraph. We also conjecture that if there is a node $v \in S(G)$ meeting conditions $C_{max}(G-v) \leqslant C_{max}(G-w)$ for each $w \in S(G) \wedge w \neq v$, then $u_1 = v$ for $MaxQ(G)$.

Keywords: canonical labeling; open *k*-neighborhood subgraph; algorithm; adjacency matrix; diffusion degree sequence; entire diffusion degree sequences

1. Introduction

This paper is the close companion to Reference [1], in which a novel theory and method are presented for calculating the canonical labelings of digraphs. The center subgraph $Cen(G)$ [2] can also be used to determine the first vertex u_1 added into $MaxQ(G)$ of undirected graphs.

This article concentrates on the construction of a universal system and a way of calculating the *canonical labeling* $C_{max}(G)$ of undirected graphs [3–5], which is also called an *optimum code* [6], a *canonical code* [7] or a canonical form [8] and is the single string corresponding one-to-one to a graph such that two graphs are isomorphic if and only if both have the accurate same canonical labelings. Currently, the calculation of the *canonical labeling* as the graph isomorphism problem is an unsolved challenge in computational complexity theory such that no polynomial-time algorithm appears for calculating the *canonical labeling* of an undirected graph. The study of canonical labeling has contributed to the studies of problems in many fields, including quantum chemistry [9], biochemistry [10,11] and so on.

Many exponential algorithms have emerged to deal with the problem. However, different researchers like to define different canonical labeling. Given a graph of n vertices, Kuramochi and Karypis build

the canonical labeling by concatenating the columns of the upper-triangular part of its adjacency matrix [12,13]. Huan et al. concatenate the lower triangular elements (including the diagonal elements) of its adjacency matrix to create its canonical labeling [14]. Kashani et al. determine the canonical labeling by concatenating the rows of its adjacency matrix to form an n^2 binary number [15,16]. Each of these canonical labelings correspondings one-to-one to a lexicographically smallest graph whose adjacency matrix is seen as a linear string, which is lexicographically smallest. Nevertheless, the computation of the lexicographically least graph is NP-hard [8,17].

Throughout the paper, the *canonical labeling* $C_{max}(G)$ of a graph G is the lexicographically largest string constructed by concatenating the rows of the upper triangular portion of the adjacency matrix associated with G (see Definition 5). The computational complexity that determines the *canonical labeling* $C_{max}(G)$ of G is also NP-hard.

Babai and Luks used a general group-theoretic method to calculate canonical labeling [8]. Nevertheless, combinatorial approaches have operated well in numerous particular situations. For stochastic graphs, Babai et al. generate canonical labeling with high possibility [8,18]. Arvind et al. introduce two similar logspace algorithms for partial 2- and 3-Trees [17,19].

Currently, *Nauty* is the most prevalent and pragmatic means for determining the automorphism group and the *canonical labelings* of graphs [3,20–22]. It appears to have shifted the industry norm for determining the *canonical labeling* also the automorphism group. For calculating the *canonical labeling* and automorphism group, *Nauty* and Yan and Han [23] use the depth-first search to traverse the latent intermediary vertices in the search tree. The vertices of the search tree produced by *Nauty* are equitable ordered partitions of vertices in G. *Nauty* iteratively refines partitioning vertices until places the vertices that have the exact equivalent features into an automorphism orbit. As the partition refinement becomes finer and finer, it automatically makes the *canonical labeling*. Nonetheless, *Nauty* also needs exponential time to calculate the canonical labeling for a given Miyazaki graph [24]. Tener and Deo earned advances for processing the problem [25].

Besides *Nauty*, *Traces* [4], *Bliss* [5,26] and Conauto [27] are all state-of-the-art tools for graphs isomorphism testing. Based on the individualization of nodes, backtracking and partition refinement, *Bliss* [5,26] is powerful canonical labeling means for dealing with large and sparse graphs. Katebi et al. combine *Saucy* with *Bliss* and show that it is faster for computing the automorphism group of a graph with *Saucy* and then calculating its canonical labeling with *Bliss* than for alone calculating its canonical labeling with *Bliss* [28]. To fix the vulnerability of *Nauty*, *Traces* uses the policy of breadth-first search to decide the automorphism group and the canonical labeling [4]. *Conauto* also utilizes the fundamental individualization/refinement method and is quite quick for random graphs and several classes of hard graphs.

For the advancement of performance, current algorithms usually employ backtracking and orbit partitioning way to circumvent frequently visiting the same nodes and contrive to decrease the accessed nodes in the search tree. For the canonical labeling issue, McKay et al. present a full examination of the problem [22,23].

Nauty governed the area for several decades. Therefore, in-depth research for canonical labeling has been limited to the theoretical skeleton of *Nauty*. This implies that people are only like to support the study trajectory of *nauty* to extend and build further research.

Since there are several different definitions of canonical labeling, there is no uniform standard such that each researcher works on oneself standard. Besides the lack of a unified standard, the research on the connection between the distinct canonical labeling is also quite lacking. It is a hard task that one wants to confirm the accuracy of canonical labeling achieved by executing an algorithm. Up to now, the criterion by which one can decide which definition is better does not appear.

In this paper, the definition of *canonical labeling* is completely distinct that of *nauty*. Unless by chance, the canonical labeling produced by *Nauty* will be not a *canonical labeling* according to the definition presented by the paper. It is sometimes difficult to confirm the accuracy of the canonical labeling achieved by executing *Nauty* according to the criterion of *Nauty*. Since the insides of many

graphs contain a large number of automorphisms, the calculation of canonical labeling becomes extremely arduous in certain situations.

A graph invariant $I(G)$ is called complete if the equality of the invariants $I(G)$ and $I(H)$ implies the isomorphism of the graphs G and H. However, even polynomial-valued invariants such as the chromatic polynomial are not usually complete. A path graph is a graph consisting of exactly its maximal path. For example, the path graph with 4 vertices and the claw graph $K_{1,3}$ both have the same chromatic polynomial.

Many algorithms also use the identical definition of *canonical labeling* as adopted in the paper. However, their main goal is not to consider how to calculate *canonical labeling* but for other purposes such as mining the frequent subgraphs. Therefore, these algorithms can only run for some limited graph classes. Until now, based on current knowledge and Definition 5 present in this paper, a universal algorithm for calculating the *canonical labelings* of graphs does not appear.

Jianqiang Hao et al. also provide Propositions 5–6, Lemmas 1–3 and Theorems 10–13 in Reference [2] by which one can compute the proper vertices added into $MaxQ(G)$. However, they do not give any proof. In this article, we prove Theorems 3–6 that are one-to-one corresponding to Theorems 10–13 in Reference [2]. We also present the reasons for Lemmas 8–9, which are one-to-one corresponding to Lemmas 2–3 in Reference [2].

In the rest of this paper, Section 2 presents some fundamental vocabulary and preparatory knowledge. Section 3 represents the results followed by some analysis. Section 4 gives our algorithms for calculating the *canonical labeling*. Section 5 demonstrates the implementation of our algorithms and evaluates our way through many examples. Finally, Section 6 remarks on our conclusions and future work.

2. Preliminaries

This paper only handles limited undirected graphs with neither loops nor multiple edges. A graph consists of a set of nodes and a collection of edges. For a graph $G = (V(G), E(G))$, assume that $V(G)$ and $E(G)$ represent the set of vertices of G and the collection of edges of G. Any an edge $(u, v) \in E(G)$ joins two vertices $u \in G$ and $v \in G$. For this article to be self-contained, the relevant concepts and definitions are given below.

Definition 1. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. A vertex-induced subdigraph on $V_1 \subseteq V(G)$ of G is a subgraph with the vertices set V_1 together with any side whose endpoints are both in V_1, expressed by $G[V_1]$.*

Definition 2. *Given two undirected graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ of n vertices. If there is a bijection $f : V(G) \to V(H)$ such that $\forall (u, v) \in E(G)$ if and only if $(f(u), f(v)) \in E(H)$. We call f an isomorphic projection of $G \to H$. Furthermore, we state that the graph G and H are isomorphic, signified by $G \cong H$. An isomorphic map f of G onto itself is declared to be an automorphism of G.*

Let $X = (x_1, x_2, \cdots, x_i, \cdots x_m)$ in R^m and $Y = (y_1, y_2, \cdots, y_j, \cdots y_n)$ in R^n be two vectors, the issue emerges as to how to determine which one is larger. When comparing two vectors, the following rules must be satisfied.

Definition 3. *Given two vectors $X = (x_0, x_1, \cdots, x_i, \cdots x_m)$ and $Y = (y_0, y_1, \cdots, y_i, \cdots y_n)$ in N (the collection of natural numbers) satisfying $m \geq 0$ and $n \geq 0$. Then, the lexicographic order of the two vectors is defined as follows:*

1. *$X = Y$, if $m = n$ and $x_i = y_i$ for all $0 \leq i \leq m$.*
2. *$X < Y$ if and only if either of the following is true.*

   (a) $\exists k, 0 \leq k \leq \min(m, n), x_i = y_i$ *for* $i < k$, $x_k < y_k$.

(b) $x_i = y_i$ *for* $0 \leq i \leq m$ *and* $m < n$.

Definition 4. *Given two vectors* $Z_1 = (X_0, X_1, \cdots, X_i, \cdots X_m)$ *and* $Z_2 = (Y_0, Y_1, \cdots, Y_j, \cdots Y_n)$ *satisfying* $m \geq 0$ *and* $n \geq 0$, *where each* $X_i, Y_j, i = 0, 1, \cdots, m, j = 0, 1, \cdots, n$ *denotes a vector in N (the collection of natural numbers). Then, the lexicographic order of the two vectors is defined as follows:*

1. $Z_1 = Z_2$, *if* $m = n$ *and* $X_i = Y_i$ *for all* $0 \leq i \leq m$.
2. $Z_1 < Z_2$ *if and only if either of the following is true.*

(a) $\exists k, 0 \leq k \leq \min(m, n), X_i = Y_i$ *for* $i < k$, $x_k < y_k$.
(b) $X_i = Y_i$ *for* $0 \leq i \leq m$ *and* $m < n$.

Definition 5. *Suppose* $G = (V(G), E(G))$ *is an undirected graph of n vertices with adjacency matrix* $A(G) = (a_{i,j})_{n\times n}$. *To concatenate the rows of the upper triangular part of* $A(G)$ *in the following order* $a_{1,2}$, $a_{1,3}$, $\cdots$, $a_{1,n}$, $a_{2,3}$, $a_{2,4}$, $\cdots$, $a_{2,n}$, $\cdots$, $a_{i,i+1}$, $a_{i,i+2}$, $\cdots$, $a_{i,n}$, $\cdots$, $a_{n-1,n}$ *makes a corresponding binary number* $a_{1,2}a_{1,3}\cdots a_{1,n}\ a_{2,3}a_{2,4}\cdots a_{2,n}\cdots a_{i,i+1}a_{i,i+2}\cdots a_{i,n}\cdots a_{n-1,n}$, *which is a labeling of G, signified by* $C(G)$ (*see* (1)).

$$A(G) = \begin{pmatrix} 0 & \rightarrow a_{1,2} & \rightarrow a_{1,3} & \cdots & \cdots & \cdots & \cdots & \rightarrow a_{1,n} \\ a_{1,2} & 0 & \rightarrow a_{2,3} & \cdots & \cdots & \cdots & \cdots & \rightarrow a_{2,n} \\ \vdots & \cdots & \ddots & \cdots & \cdots & \cdots & \cdots & \downarrow\ \vdots \\ \vdots & \cdots & \cdots & \ddots & \cdots & \cdots & \cdots & \downarrow\ \vdots \\ a_{1,i} & a_{2,i} & \cdots & a_{i,i-1} & 0 & \rightarrow a_{i,i+1} & \cdots & \rightarrow a_{i,n} \\ \vdots & \cdots & \cdots & \cdots & \cdots & \ddots & \cdots & \downarrow\ \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \ddots & \rightarrow a_{n-1,n} \\ a_{1,n} & a_{2,n} & a_{3,n} & \cdots & \cdots & \cdots & a_{n-1,n} & 0 \end{pmatrix} \quad (1)$$

The first row of the upper triangular portion of $A(G)$ is the *labeling slice* 1 of $C(G)$, signified by $C^1(G)$. Likewise, the second row of the upper triangular portion is the *labeling slice* 2 of $C(G)$, signified by $C^2(G)$. $\cdots$. The $(n-1)$th row of the upper triangular portion is the *labeling slice* $n-1$ of $C(G)$, signified by $C^{n-1}(G)$. It is true that $C(G) = C^1(G)C^2(G)\cdots C^{n-1}(G)$.

A permutation π of the nodes of G is an order of the n nodes without repeating. The number of shifts of the nodes of G is $n!$. Further, each distinct permutation π of the n nodes of $V(G)$ determines a single adjacency matrix. Therefore, given a permutation π, one can get a *labeling* $C(G)$ corresponding to π by Definition 5. The collection of all *labeling* of G is represented by $L(G)$.

For every $C_1(G)$, $C_2(G) \in L(G)$, Suppose that $C_1(G) = i_1 i_2 \cdots i_m$, $C_2(G) = j_1 j_2 \cdots j_n$ with $i_1, i_2, \cdots, i_m$, $j_1, j_2, \cdots, j_n = 0$ or 1. Given $X = (i_1, i_2, \cdots, i_m)$ and $Y = (j_1, j_2, \cdots, \cdots j_n)$. By Definition 3, if $X > Y$, then we set $C_1(G) > C_2(G)$. Otherwise, if $X < Y$, then we set $C_1(G) < C_2(G)$. Otherwise, if $X = Y$, then we set $C_1(G) = C_2(G)$.

Definitely, $(L(G), \leq)$ is a well-ordered set, where $\leq$ signifies the less-than-or-equal-to binary relationship on the collection $L(G)$ stated as above. By the well-ordering theorem, it follows that $L(G)$ has a minimum and maximum element, signifies by $C_{min}(G)$ and $C_{max}(G)$ and called *minimum canonical labeling* of G and *maximum canonical labeling* of G respectively. We also call *maximum canonical labeling canonical labeling* of G.

The two shifts of the n nodes of G associated with $C_{min}(G)$ and $C_{max}(G)$ are the *minimum and maximum node sequence*, signifies by $MinQ(G)$ and $MaxQ(G)$. Furthermore, the two adjacency matrices of G associated with $C_{min}(G)$ and $C_{max}(G)$ are the *minimum and maximum canonical labeling matrix*, signifies by $A_{min}(G)$ and $A_{max}(G)$.

$C^1(G), C^2(G), \cdots, C^{n-1}(G)$ corresponding to $A_{min}(G)$ are *minimum canonical labeling slice* $1, 2, \cdots, n-1$ of canonical labeling $C(G)$, signified by $C^1_{min}(G)$, $C^2_{min}(G)$, $\cdots$, $C^{n-1}_{min}(G)$, respectively. Conversely, $C^1(G)$, $C^2(G)$, $\cdots$, $C^{n-1}(G)$ corresponding to $A_{max}(G)$ are *maximum canonical labeling slice* $1, 2, \cdots, n-1$ of canonical labeling $C(G)$, signified by $C^1_{max}(G)$, $C^2_{max}(G)$, $\cdots$, $C^{n-1}_{max}(G)$, respectively.

Based on the above definitions, the following equations are established.

$$C_{min}(G) = C^1_{min}(G)C^2_{min}(G)\cdots C^{n-1}_{min}(G) \quad (2)$$

$$C_{max}(G) = C^1_{max}(G)C^2_{max}(G)\cdots C^{n-1}_{max}(G) \quad (3)$$

Theorem 1. *Given two undirected graphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$ of n vertices with adjacency matrices $A(G)$ and $A(H)$ respectively. Then $G \cong H$ if and only if $C_{min}(G) = C_{min}(H)$ or $C_{max}(G) = C_{max}(H)$.*

Lemma 1. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices whose complement graph is $\overline{G} = (V(\overline{G}), E(\overline{G}))$. Then $C(G) = \overline{C(\overline{G})}$ and $C(\overline{G}) = \overline{C(G)}$ hold.*

Proof. The adjacency matrices of G and $\overline{G}$ meet the following condition.

$$A(G) + A(\overline{G}) = J = \begin{pmatrix} 0 & 1 & \cdots & \cdots & 1 \\ 1 & 0 & 1 & \cdots & \vdots \\ \vdots & 1 & 0 & 1 & \vdots \\ \vdots & \vdots & 1 & \ddots & 1 \\ 1 & \cdots & \cdots & 1 & 0 \end{pmatrix}.$$

where J is an $n \times n$ matrix of zeros and ones whose main diagonal entries are 0 and all other entries are 1. By the complement graph $\overline{G}$ and $A(G) = J - A(\overline{G})$, it can be seen that $C(G) = \overline{C(\overline{G})}$. Furthermore, by $A(\overline{G}) = J - A(G)$, it follows that $C(\overline{G}) = \overline{C(G)}$ for the complement graph $\overline{G}$ of G. □

Theorem 2. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices whose complement graph is $\overline{G} = (V(\overline{G}), E(\overline{G}))$. It follows that*

$$C_{min}(G) = \overline{C_{max}(\overline{G})} \quad (4)$$

$$C_{max}(G) = \overline{C_{min}(\overline{G})} \quad (5)$$

$$C_{min}(\overline{G}) = \overline{C_{max}(G)} \quad (6)$$

$$C_{max}(\overline{G}) = \overline{C_{min}(G)} \quad (7)$$

Proof. By Lemma 1, it holds that $C(G) = \overline{C(\overline{G})}$. Definitely, the k-bit of $C(\overline{G})$ is 0 if and only if the k-bit of $C(G)$ is 1. Therefore, one can make the *canonical labeling* $C_{max}(G)$ of G by executing a complement operation on $C_{min}(\overline{G})$. Likewise, by Lemma 1, the identity $C(\overline{G}) = \overline{C(G)}$ holds. Definitely, the k-bit of $C(G)$ is 0 if and only if the k-bit of $C(\overline{G})$ is 1. Hence, one can make the *canonical labeling* $C_{max}(\overline{G})$ of $\overline{G}$ by executing a complement operation on $C_{min}(G)$.

Since $C(K_n)$ is a constant binary number, one must maximize $C(\overline{G})$ to minimize $C(G)$. On the opposite, one must minimize $C(\overline{G})$ to maximize $C(G)$. Likewise, one must maximize $C(G)$ to minimize $C(\overline{G})$. Contrariwise, one must minimize $C(G)$ to maximize $C(\overline{G})$. From the above examination, the following results hold.

$$C_{min}(G) = \overline{C_{max}(\overline{G})}. \qquad C_{max}(G) = \overline{C_{min}(\overline{G})}.$$
$$C_{min}(\overline{G}) = \overline{C_{max}(G)}. \qquad C_{max}(\overline{G}) = \overline{C_{min}(G)}.$$

□

By Theorem 2, it can be seen that if one has computed the $C_{max}(\overline{G})$, one can simply make $C_{min}(G)$. Moreover, the computation means of $C_{max}(\overline{G})$ and $C_{max}(G)$ are precisely the same.

Theorem 2 shows the mutual relations between $C_{max}(G)$ and $C_{min}(G)$. Because of the existence of the relations, the paper only concentrates on the construction of effective ways to determine $C_{max}(G)$. The *TopMost graph* of G is a graph whose *labeling* $C(G)$ is lexicographically largest.

We signify by $d_G(u)$ the degree of a node u in G, by $d(G) = (d_G(u_1), d_G(u_2), \cdots, d_G(u_n))$ the *degree sequence* of G, by $d_G(V_1) = (d_G(v_1), d_G(v_2), \cdots, d_G(v_m))$ the degree series of a subset $V_1 \subseteq V(G)$ with $v_i \in V_1$, $i = 1, 2, \cdots, m$ and by $d_G(H) = (d_H(w_1), d_H(w_2), \cdots, d_H(w_s))$ the *degree sequence* of a subgraph $H \subseteq G$ with $w_j \in V(H)$, $j = 1, 2, \cdots, t$ and drop the symbol G when no vagueness can occur. We signify by $\delta(G)$ the minimum degree and by $\Delta(G)$ the maximum degree of all vertices of a graph G. Throughout this article, suppose $S(G) = \{u | u \in V(G) \wedge d(u) = \Delta(G)\}$. In the following text, when simultaneously involving two graphs, we always assume that their degree sequences are the same except specified.

For each $u \in V(G)$, the quantity of vertices with degree $d_G(u)$ is the *degree multiplicity* of u, signified by $dm_G(u)$. Unless otherwise specified, throughout this article, the degree sequence is decreasing. The distance $d(u, v)$ between any two nodes u and v is the number of edges on the shortest path from u to v.

For every $u \in V(G)$, an edge connecting two adjacent vertices of u is a *chord edge* of u and an edge connecting u and its an adjacent vertex is an *incident edge* of u. Given a set S of vertices in G, let $CE(S)$ signify the *set of chord edges* of all nodes in S and $IE(S)$ signify the *set of incident edges* of all vertices in S. Such as in the graph G shown in Figure 1, the edge $(3, 8)$ is a *chord edge* of the vertex 7 and each of the edges $(3, 7)$, $(7, 8)$ and $(10, 7)$ are its *incident edges*. $CE(\{7\}) = \{(3, 8)\}$ and $IE(\{7\}) = \{(3, 7), (7, 8), (10, 7)\}$.

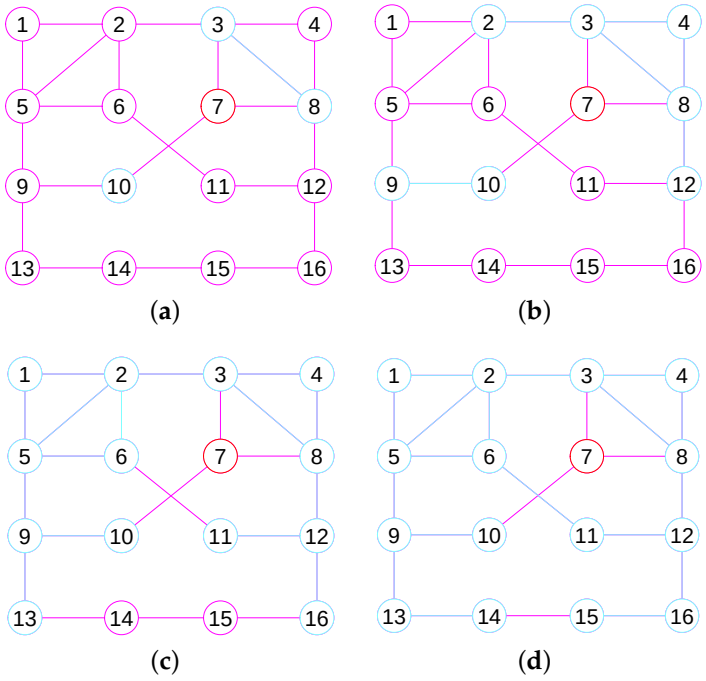

Figure 1. The 1, 2, 3, 4-neighborhood subgraph of vertex 7 in a graph G. (**a**) The 1-neighbor subgraph; (**b**) The 2-neighbor subgraph; (**c**) The 3-neighbor subgraph; (**d**) The 4-neighbor subgraph.

Definition 6. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. The open neighborhood subgraph of a node u in G is a subgraph of G, signified by $N(u) = (V(N(u)), E(N(u)))$ where $V(N(u))$ is the set of all nodes adjacent to u ($u \notin V(N(u))$) and $E(N(u))$ is the collection of all edges, each of which connects two vertices of $V(N(u))$.*

The open k-neighborhood subgraph of u with $k \geq 2$ is a subgraph, signified by $N_k(u) = (V(N_k(u)), E(N_k(u)))$ with $V(N_k(u)) = \{v \mid d(u,v) \leq k \wedge v \neq u \wedge v \in V(G)\}$, $E(N_k(u)) = \{(v, w) \mid v, w \in V(N_k(u)), (v,w) \in CE(V(N_{k-1}(u))) \vee IE(V(N_{k-1}(u)))\}$.

Definition 7. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. The open neighborhood subgraph of a vertices set $Q \subseteq V(G)$ is a subgraph of G, signified by $N(Q) = (V(N(Q)), E(N(Q)))$ where $V(N(Q))$ is the set of all nodes each of which is adjacent to at least one node in Q with $Q \cap V(N(Q)) = \emptyset$ and $E(N(Q))$ is the collection of all edges each of which joins two vertices of $V(N(Q))$.*

The open k-neighborhood subgraph of Q with $k \geq 2$ is a subgraph, signified by $N_k(Q) = (V(N_k(Q)), E(N_k(Q)))$ with $V(N_k(Q)) = \{v \mid v \notin Q \wedge v \in V(G) \wedge \exists u \in Q \wedge d(v,u) \leq k\}$, $E(N_k(Q)) = \{(v,w) \mid v, w \in V(N_k(Q)), (v,w) \in CE(V(N_{k-1}(Q))) \vee IE(V(N_{k-1}(Q)))\}$.

Remark 1. *For some graphs, there may be some edges whose two end vertices lie in $V(N_k(u))$ but do not belong to $N_k(u)$ by Definition 6. For example, consider the graph G given in Figure 1. Although the vertex 6 and 11 belong to $V(N_3(7))$, $(6, 11) \notin E(N_3(7))$. In addition, the vertex 14 and 15 belong to $V(N_4(7))$. However, $(14, 15) \notin E(N_4(7))$.*

Definition 8. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. For each $u \in V(G)$, the open kth neighborhood subgraph of u with $k \geq 0$ is a subgraph, signified by $T_k(u) = (V(T_k(u)), E(T_k(u)))$ with $V(T_k(u)) = \{v \mid d(u,v) = k \wedge v \neq u \wedge v \in V(G)\}$, $E(T_k(u)) = \{(v,w) \mid v, w \in V(T_k(u)) \wedge (v,w) \in CE(V(T_k(u)))\}$. For $k = 0$, let $V(T_0(u)) = \{u\}$ and $E(T_0(u)) = \emptyset$.*

Definition 9. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. Assume that $H \subseteq G$ and $u \in V(H)$. We signify by $N_H^k(u)$ the open k-neighborhood subgraph of u in H and by $T_H^k(u)$ the open kth neighborhood subgraph of u in H. For $k = 1$, we drop the superscript 1 for clarity and write $N_H(u) = N_H^1(u)$ and $T_H(u) = T_H^1(u)$, instead.*

Definition 10. *Suppose $G = (V(G), E(G))$ is an undirected connected graph of n vertices. For each $u \in V(G)$, there is a positive integer k meeting conditions $N_{k-1}(u) \subset G - u$ and $N_k(u) = G - u$. The value of k is defined as the diffusion radius of u, signified by $\rho_G(u)$ and we drop the symbol G when no vagueness can occur.*

By Definition 10, it is explicit that $N_{\rho(u)}(u) = G - u$ for each u in G. For notational convenience, we sometimes use $G - u$ as a shorthand for $N_{\rho(u)}(u)$.

Here, we discuss the connection between the *open k-neighborhood subgraph* $N_k(u)$ and $d(u)$ with $u \in V(N_k(u))$ (see Figure 2). We present some basic properties of the *open k-neighborhood subgraph* $N_k(u)$ by the following Lemma 2.

Lemma 2. *Let $N_1(u), N_2(u), \cdots, N_k(u), \cdots, N_{\rho(u)}(u)$ be the $1, 2, \cdots, k, \cdots, \rho(u)$-neighborhood subgraph of a vertex u in a graph G. If there exists a node $v \in V(N_1(u))$, then*

$$d_{N_2(u)}(v) = d_{N_3(u)}(v) \quad = \quad \cdots = d_{N_k(u)}(v) = \cdots = d_{N_{\rho(u)}(u)}(v) = d_G(v) - 1. \tag{8}$$

Likewise, if there exists a node $v \notin V(N_k(u)) \wedge v \in V(N_{k+1}(u))$ with $k \geq 1$, then

$$d_{N_{k+2}(u)}(v) = d_{N_{k+3}(u)}(v) \quad = \quad \cdots = d_{N_{\rho(u)}(u)}(v) = d_G(v). \tag{9}$$

Proof. (8) and (9) follow directly from Definition 10 (see Figure 2). □

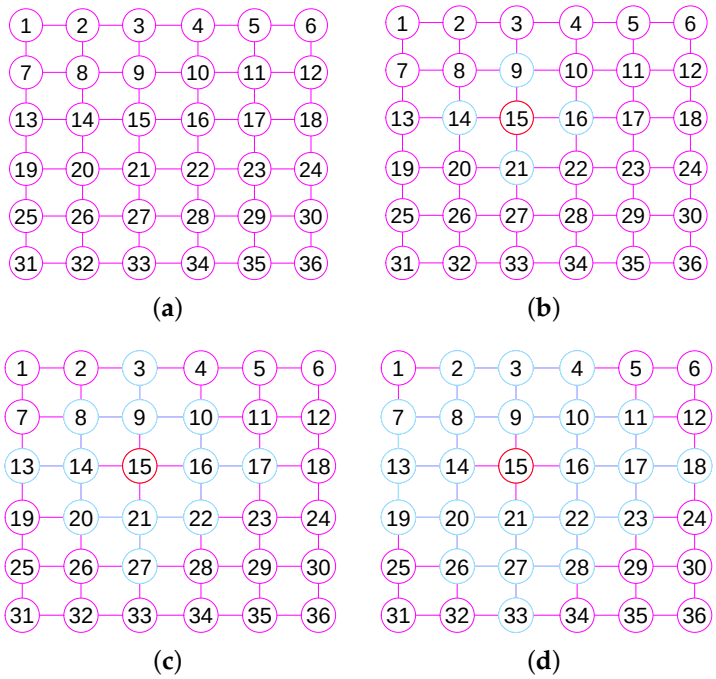

Figure 2. The 1, 2, 3-neighborhood subgraph of vertex 15 in the 6×6 grid graph $G_{6,6}$. (**a**) The 6×6 grid graph $G_{6,6}$; (**b**) The 1-neighbor subgraph of 15; (**c**) The 2-neighbor subgraph of 15; (**d**) The 3-neighbor subgraph of 15.

By Lemma 2, it is clear that for every $v \in V(N_{k+1}(u))$, satisfies $v \in V(N_k(u))$ or $v \notin V(N_k(u))$. According to whether v belongs to $V(N_k(u))$ or not, $V(N_{k+1}(u))$ can be partitioned into two disjoint sets $V^1_{k+1}(u) = \{v \mid v \in V(N_k(u)) \wedge v \in V(N_{k+1}(u))\}$ and $V^2_{k+1}(u) = \{w \mid w \notin V(N_k(u)) \wedge w \in V(N_{k+1}(u))\}$. Observe that $V^1_{k+1}(u) = V(N_k(u))$ and $V^2_{k+1}(u) = V(N(V^2_k(u)))$ hold by Definitions 6 and 7 (see Figure 1). Therefore $V(N_{k+1}(u)) = V^1_{k+1}(u) \cup V^2_{k+1}(u) = V(N_k(u)) \cup V(N(V^2_k(u)))$. $V^1_{k+1}(u)$ and $V^2_{k+1}(u)$ are the *degree stable vertices set* and *degree unstable vertices set* of $N_{k+1}(u)$ referred to as the *stable vertices set* and *unstable vertices set*. Correspondingly, a vertex $v \in V^1_{k+1}(u)$ is a *degree stable vertex* referred to as a *stable vertex* and a vertex $w \in V^2_{k+1}(u)$ is a *degree unstable vertex* referred to as an *unstable vertex*.

Further note that $E(N_{k+1}(u)) = E(N_k(u)) \cup E_{k+1}$ with $E_{k+1} = CE(N(V^2_k(u))) \cup IE(N(V^2_k(u)))$. Therefore, $E(N_{k+1}(u)) = E(N_k(u)) \cup CE(N(V^2_k(u))) \cup IE(N(V^2_k(u)))$. The following Lemma 3 sums up the above discussion.

Lemma 3. *Suppose $N_{k+1}(u) = (V(N_{k+1}(u)), E(N_{k+1}(u)))$ is the open $(k+1)$-neighborhood subgraph of vertex u in a graph G. Then there exist two disjoint sets, the stable vertices set $V^1_{k+1}(u)$ and the unstable vertices set $V^2_{k+1}(u)$, meeting conditions*

$$V(N_{k+1}(u)) = V^1_{k+1}(u) \cup V^2_{k+1}(u) = V(N_k(u)) \cup V(N(V^2_k(u))), \tag{10}$$

with $V^1_{k+1}(u) = V(N_k(u))$, $V^2_{k+1}(u) = V(N(V^2_k(u)))$. Further, it follows that

$$E(N_{k+1}(u)) = E(N_k(u)) \cup CE(N(V^2_k(u))) \cup IE(N(V^2_k(u))). \tag{11}$$

Let $V(N_0(u)) = \emptyset$ and $V^2_0(u) = u$, then $V^1_1(u) = \emptyset$ and $V^2_1(u) = V(N_1(u))$ hold for the neighborhood subgraph.

For the $\rho(u)$-neighborhood subgraph, $V^1_{\rho(u)+1}(u) = V(N_{\rho(u)+1}(u)) = V(N_\rho(u))$ holds with $V^2_{\rho(u)+1}(u) = \emptyset$.

By Lemma 3, it can be seen that the calculation of the *open* $(k+1)$*-neighborhood subgraph* can be simplified by means of the *open k-neighborhood subgraph* for every node v in G.

Definition 11. *Assume that* $G = (V(G), E(G))$ *is an undirected graph and* u *is a vertex in* G *whose open k-neighborhood subgraph is* $N_k(u)$ *with* $k = 1, 2, \cdots$. *A vertex in* $N(u)$ *is a one diffusion radius node of* u. *For* $k > 1$, *a node* v *in* $N_k(u)$ *meeting condition* $v \notin N_{k-1}(u)$ *is a k diffusion radius node of* u.

Each vertex v in G is attached a property *m_NearestNode* whose function is explained in Section 3.1.2.

Definition 12. *Assume that* $G = (V(G), E(G))$ *is an undirected graph and* u *is a node in* G *whose open k-neighborhood subgraph is* $N_k(u)$ *with* $k = 1, 2, \cdots$. *Suppose* H *is a connected component of* $N_k(u)$ *with* $k \geq 1$. *Suppose that* $V_m \subseteq V(H)$ *with* $m = 0, 1, 2, \cdots, t$, *where* V_0 *is in ascending order of attribute m_NearestNode with*

$v- > m\_NearestNode \geq 1$ *for every* $v \in V_0$ *and* $V_1, V_2, \cdots, V_t$ *contain the* $1, 2, \cdots, k$ *diffusion radius nodes of* u *respectively, meeting conditions* $V_0 \cup V_1 \cup \cdots \cup V_t = V(H)$ *and* $V_i \cap V_j = \emptyset$ *for* $i \neq j, i, j = 0, 1, \cdots, t$.

Let $d_\sigma(H) = (d_{N_k(u)}(V_0), \cdots, d_{N_k(u)}(V_i), \cdots, d_{N_k(u)}(V_t))$ *be defined as the diffusion degree sequence of* H *where* $d_{N_k(u)}(V_i)$ *with* $i = 0, 1, \cdots, t$ *are the degree sequences in non-increasing order induced by all nodes in* V_i *respectively.*

Definition 13. *Assume that* $G = (V(G), E(G))$ *is an undirected graph and* u *is a node in* G *whose open k-neighborhood subgraph k is* $N_k(u)$ *with* $k = 1, 2, \cdots$. *Assume that* $N_k(u)$ *has p connected components* H_1, $H_2, \cdots, H_p$ *with diffusion degree sequences* $d_\sigma(H_1), d_\sigma(H_2), \cdots, d_\sigma(H_p)$ *respectively, meeting conditions* $d_\sigma(H_1) \geq d_\sigma(H_2) \geq \cdots \geq d_\sigma(H_p)$.

We define $d^\sigma_G[N_k(u)] = (d_\sigma(H_1), d_\sigma(H_2), \cdots, d_\sigma(H_p))$ *to be the entire diffusion degree sequence of* $N_k(u)$ *about* u *in* G *and drop the symbol* G *when no vagueness can occur.*

We define $d^{max}_G[N_k(u)] = d_\sigma(H_1)$ *to be the maximum diffusion degree sequence of* $N_k(u)$ *about* u *in* G *and drop the symbol* G *when no vagueness can occur.*

Remark 2. *To define diffusion degree sequences* $d_\sigma(H) = (d_{N_k(u)}(V_0), d_{N_k(u)}(V_1), \cdots, d_{N_k(u)}(V_t))$ *and entire diffusion degree sequence* $d^\sigma_G[N_k(u)] = (d_\sigma(H_1), d_\sigma(H_2), \cdots, d_\sigma(H_p))$, *we pay a great deal of efforts into software testing and theoretical studies. We used more than 20 different kinds of degree sequences in the software experiments and compared the results of distinct degree sequences. Built on the preceding works, it is not difficult to find that performance of the two degree sequences specified by Definitions 12 and 13 is optimal. With the adoption of the two definitions, the accuracy of our algorithm significantly enhances.*

Given a graph G, the *entire diffusion degree sequence* $d^\sigma_G[N_k(u)] = (d_\sigma(H_1), d_\sigma(H_2), \cdots, d_\sigma(H_p))$ can be used in the literature of finance for the jump-diffusion models [29,30].

Definition 14. *Suppose* $G = (V(G), E(G))$ *is an undirected graph of n vertices. For each* $u, v \in V(G)$ *with* $u \neq v$ *and , let* $N_1(u)$, $N_2(u)$, $\cdots$, $N_{\rho(u)}(u)$ *be the* 1, 2, $\cdots$, $\rho(u)$ *neighborhood subgraph of* u *with entire diffusion degree sequences* $d^\sigma[N_1(u)]$, $d^\sigma[N_2(u)]$, $\cdots$, $d^\sigma[N_{\rho(u)}(u)]$, *respectively. Let* $N_1(v)$, $N_2(v)$, $\cdots$, $N_{\rho(v)}(v)$ *be the* 1, 2, $\cdots$, $\rho(v)$ *neighborhood subgraph of* v *with entire diffusion degree sequences* $d^\sigma[N_1(v)]$, $d^\sigma[N_2(v)]$, $\cdots$, $d^\sigma[N_{\rho(v)}(v)]$, *respectively. Let* $Z_1 = (d^\sigma[N_1(u)], d^\sigma[N_2(u)], \cdots, d^\sigma[N_{\rho(u)}(u)])$ *and* $Z_2 = (d^\sigma[N_1(v)], d^\sigma[N_2(v)], \cdots, d^\sigma[N_{\rho(v)}(v)])$. *If* $Z_1 > Z_2$, *we call* $u \succ v$ *concerning* G. *Otherwise, if* $Z_1 < Z_2$, *we call* $u \prec v$ *concerning* G. *Otherwise, if* $Z_1 = Z_2$, *we call* $u \asymp v$ *concerning* G *and ignore the sign* G *when no vagueness can occur. Signify* $u \succ v$ *or* $u \asymp v$ *by* $u \succeq v$ *and* $u \prec v$ *or* $u \asymp v$ *by* $u \preceq v$.

Observe that $\succ, \prec, \asymp, \succeq$ are all binary relations on the collection of vertices $V(G)$. By Definition 14, for each $u, v \in V(G)$ with $u \neq v$, one of the following assertions is true: (1) $u \succ v$. (2) $u \prec v$. (3) $u \asymp v$.

It can be noted that $(V(G), \succeq)$ is a well-ordered set, where $\succeq$ signifies the binary relation $u \succeq v$ on the set $V(G)$. By the well-ordering theorem, it holds that there is a maximum and minimum element in $V(G)$, signified by $G_{max}^{\succeq}$ and $G_{min}^{\succeq}$ respectively with $G_{max}^{\succeq} \in V(G)$ and $G_{min}^{\succeq} \in V(G)$. The superscript $\succeq$ can be ignored if no vagueness can occur. The following Lemmas 4–6 immediately hold from Definition 14.

Lemma 4. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. For each $u, v \in V(G)$ with $u \neq v$, if the sign $\succeq$ signifies the binary relation $u \succeq v$ on the collection $V(G)$, then, all of the vertices in G build a sole linkage L on G: $v_1 \succeq v_2 \succeq \cdots \succeq v_i \succeq \cdots \succeq v_n$ with $v_i \in V(G)$, $i = 1, 2, \cdots, n$.*

Lemma 5. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. Suppose $S(G) = \{u |\ u \in V(G) \wedge d(u) = \Delta(G) = s\}$. For each $u, v \in S(G)$ with $u \neq v$, if the sign $\succeq$ signifies the binary relation $u \succeq v$ on the set $S(G)$, then, all of the vertices in $S(G)$ build a sole linkage L on G: $v_1 \succeq v_2 \succeq \cdots \succeq v_i \succeq \cdots \succeq v_s$ with $v_i \in S(G), i = 1, 2, \cdots, s$.*

Lemma 6. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. For each $u \in V(G)$, Let $N_k(u)$ be the open k-neighborhood subgraph of u in G. Suppose $S(N_k(u)) = \{v |\ v \in V(N_k(u)) \wedge d(v) = \Delta(N_k(u)) = t\}$.*

For each $v, w \in S(N_k(u))$ with $v \neq w$, if the sign $\succeq$ signifies the binary relation $v \succeq w$ on the collection $S(N_k(u))$, then, all of the vertices in $S(N_k(u))$ build a single linkage L on $N_k(u)$: $v_1 \succeq v_2 \succeq \cdots \succeq v_i \succeq \cdots \succeq v_t$ with $v_i \in S(N_k(u)), i = 1, 2, \cdots, t$.

By Definitions 3, 4 and 14, the outcomes in the following Propositions 1 and 2 are uncomplicated.

Proposition 1. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. For each $u, v \in V(G)$ with $u \neq v$, let $N_1(u)$, $N_2(u)$, $\cdots$, $N_{\rho(u)}(u)$ be the 1, 2, $\cdots$, $\rho(u)$ neighborhood subgraph of u with entire diffusion degree sequences $d^{\sigma}[N_1(u)]$, $d^{\sigma}[N_2(u)]$, $\cdots$, $d^{\sigma}[N_{\rho(u)}(u)]$, respectively. Let $N_1(v)$, $N_2(v)$, $\cdots$, $N_{\rho(v)}(v)$ be the 1, 2, $\cdots$, $\rho(v)$ neighborhood subgraph of v with entire diffusion degree sequences $d^{\sigma}[N_1(v)]$, $d^{\sigma}[N_2(v)]$, $\cdots$, $d^{\sigma}[N_{\rho(v)}(v)]$, respectively. Let $Z_1 = (d^{\sigma}[N_1(u)], d^{\sigma}[N_2(u)], \cdots, d^{\sigma}[N_{\rho(u)}(u)]) = (X_1, Y_1)$ with $X_1 = (d^{\sigma}[N_1(u)], d^{\sigma}[N_2(u)], \cdots, d^{\sigma}[N_k(u)])$ and $Y_1 = (d^{\sigma}[N_{k+1}(u)], d^{\sigma}[N_{k+2}(u)], \cdots, d^{\sigma}[N_{\rho(u)}(u)])$. Let $Z_2 = (d^{\sigma}[N_1(v)], d^{\sigma}[N_2(v)], \cdots, d^{\sigma}[N_{\rho(v)}(v)]) = (X_2, Y_2)$ with $X_2 = (d^{\sigma}[N_1(v)], d^{\sigma}[N_2(v)], \cdots, d^{\sigma}[N_k(v)])$ and $Y_2 = (d^{\sigma}[N_{k+1}(v)], d^{\sigma}[N_{k+2}(v)], \cdots, d^{\sigma}[N_{\rho(v)}(v)])$. If $X_1 = X_2$, then $Y_1 > Y_2$ leads to $Z_1 > Z_2$. Otherwise, $Y_1 < Y_2$ leads to $Z_1 < Z_2$. Otherwise, $Y_1 = Y_2$ leads to $Z_1 = Z_2$. Accordingly, it follows that if $X_1 = X_2$, then $Y_1 > Y_2$ leads to $u \succ v$. Otherwise, $Y_1 < Y_2$ leads to $u \prec v$. Otherwise, $Y_1 = Y_2$ leads to $u \asymp v$.*

Proposition 2. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. For each $u, v \in V(G)$ with $u \neq v$, let $N_1(u)$, $N_2(u)$, $\cdots$, $N_{\rho(u)}(u)$ be the 1, 2, $\cdots$, $\rho(u)$ neighborhood subgraph of u with entire diffusion degree sequences $d^{\sigma}[N_1(u)]$, $d^{\sigma}[N_2(u)]$, $\cdots$, $d^{\sigma}[N_{\rho(u)}(u)]$, respectively. Let $N_1(v)$, $N_2(v)$, $\cdots$, $N_{\rho(v)}(v)$ be the 1, 2, $\cdots$, $\rho(v)$ neighborhood subgraph of v with entire diffusion degree sequences $d^{\sigma}[N_1(v)]$, $d^{\sigma}[N_2(v)]$, $\cdots$, $d^{\sigma}[N_{\rho(v)}(v)]$, respectively. If $d^{\sigma}[N_1(u)] = d^{\sigma}[N_1(v)]$, $d^{\sigma}[N_2(u)] = d^{\sigma}[N_2(v)]$, $\cdots$, $d^{\sigma}[N_{k-1}(u)] = d^{\sigma}[N_{k-1}(v)]$, $d^{\sigma}[N_k(u)] > d^{\sigma}[T_k(v)]$, then $u \succ v$ concerning G (see Definition 14).*

3. Results and Discussion

Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. In the section, we will examine how to determine the *canonical labeling* $C_{max}(G)$ of the graph G. The permutations of G associated with $C_{max}(G)$ are the $MaxQ(G)$. Without loss of generality, let $MaxQ(G) = (u_1, u_2, \cdots, u_i, \cdots, u_n)$. Throughout the article, all algorithms presented use an adjacency list to save the graph G.

3.1. Calculate $C_{max}(G)$ for a Connected Graph

In this subsection, we consider how to determine the *canonical labeling* $C_{max}(G)$ of a connected graph. What method should one take to calculate the *canonical labeling* $C_{max}(G)$? From the relationship between $C_{max}(G)$ and $A_{max}(G)$, a way for computing $C_{max}(G)$ must first get the permutation $MaxQ(G)$ associated with the adjacency matrix $A_{max}(G)$.

3.1.1. Calculate the First Node u_1 of $MaxQ(G)$

In this sub-subsection, we study how to calculate the first vertex u_1 of $MaxQ(G)$. Jianqiang Hao et al. employ the $Cen(G)$ to determine the first node u_1 of $MaxQ(G)$ for simple nonregular graphs [2]. This means that this method is invalid for regular graphs. Suppose that G is a connected graph with order $n > 1$. It can be seen that one must let $a_{1,2} = 1$ maximize $C(G)$ (see (1)). $a_{1,2} = 1$ can always be taken because G is connected with order $n > 1$. Besides, to get $C_{max}(G)$, one must choose u_1 from $S(G)$. Only by so doing, can there be more "1" s in the high bits of $C(G)$ such that guarantees maximum $C(G)$. Otherwise, $C(G)$ cannot attain the maximum value. From the previous analysis, the following Proposition 3 and Lemma 7 hold.

Proposition 3. *Assume that $G = (V(G), E(G))$ is an undirected graph of n vertices and $S(G) = \{u|u \in V(G) \wedge d(u) = \Delta(G)\}$. Then the choice of u_1 of $MaxQ(G)$ is from $S(G)$ for getting $C_{max}(G)$.*

Lemma 7. *Assume that $G = (V(G), E(G))$ be an undirected graph of n vertices and $S(G) = \{u|u \in V(G) \wedge d(u) = \Delta(G)\}$. If $|S(G)| = 1$ with $v \in S(G)$. Then $u_1 = v$ for $MaxQ(G)$.*

For $|S(G)| > 1$, the following Theorem 3 holds.

Theorem 3. *Assume that $G = (V(G), E(G))$ is an undirected connected graph of n vertices and $S(G) = \{u| u \in V(G) \wedge d(u) = \Delta(G) = s\}$ with $|S(G)| > 1$. Let $v \in S(G)$ with $v_1 \in V(N(v))$ and $d_{N(v)}(v_1) = \Delta(N(v))$. For every $w \in S(G) \wedge w \neq v$ with $w_1 \in V(N(w))$ and $d_{N(w)}(w_1) = \Delta(N(w))$, if condition $d_{N(v)}(v_1) > d_{N(w)}(w_1)$ holds, then $u_1 = v$ for $MaxQ(G)$.*

Proof. From (1), it follows that $C^1_{max}(G) = a_{1,2}a_{1,3} \cdots a_{1,n}$ and $C^2_{max}(G) = a_{2,3}a_{2,4} \cdots a_{2,n}$. Since $\triangle(G) = s$, it can be shown that $a_{1,2} = a_{1,3} = \cdots = a_{1,s+1} = 1$ for $C^1_{max}(G)$.

By the conditions of Theorem 3, it is clearly that $d_{N(v)}(v_1) = \triangle(N(v))$ and $d_{N(w)}(w_1) = \triangle(N(w))$. For clarity, let us suppose that $r = d_{N(v)}(v_1)$, $t = d_{N(w)}(w_1)$.

If conditions $r = d_{N(v)}(v_1) > t = d_{N(w)}(w_1)$ are satisfied for every $w \in S(G) \wedge w \neq v$ and make the $u_1 = w$, then at most $C^2_{max}(G)$=$a_{2,3}a_{2,4} \cdots a_{2,n}$ with $a_{2,3} = a_{2,4} = \cdots = a_{2,t+2} = 1$ and $a_{2,t+3} = a_{2,t+4} = \cdots = a_{2,s+1} = 0$ (see (1). Otherwise, if let the $u_1 = v$, then $C^2_{max}(G)$=$a_{2,3}a_{2,4} \cdots a_{2,n}$ with $a_{2,3} = a_{2,4} = \cdots = a_{2,r+2} = 1$ and $a_{2,r+3} = a_{2,r+4} = \cdots = a_{2,r+1} = 0$ (see (1)).

Because $r > t$, Theorem 3 follows by contrasting the above two outcomes of $C^2_{max}(G)$ got. □

Theorem 4. *Assume that $G = (V(G), E(G))$ is an undirected connected graph of n vertices and $S(G) = \{u| u \in V(G) \wedge d(u) = \Delta(G) = s\}$ with $|S(G)| > 1$. Let $v \in S(G)$ with $v_1 \in V(N(v))$ and $d_{N(v)}(v_1) = \Delta(N(v))$. For each $w \in S(G) \wedge w \neq v$ with $w_1 \in V(N(w))$ and $d_{N(w)}(w_1) = \Delta(N(w))$, if conditions $d_{N(v)}(v_1) = d_{N(w)}(w_1)$ and $d_G(v_1) > d_G(w_1)$ hold, then $u_1 = v$ for $MaxQ(G)$.*

Proof. From (1), it follows that $C^1_{max}(G) = a_{1,2}a_{1,3} \cdots a_{1,n}$ and $C^2_{max}(G) = a_{2,3}a_{2,4} \cdots a_{2,n}$. Since $\triangle(G) = s$, it can be shown that $a_{1,2} = a_{1,3} = \cdots = a_{1,s+1} = 1$ for $C^1_{max}(G)$.

By the conditions of Theorem 4, it is clear that $d_{N(v)}(v_1) = d_{N(w)}(w_1) = \triangle(N(v))$. For clarity, let us suppose that $t = d_{N(v)}(v_1) = d_{N(w)}(w_1)$ and $l = d_G(v_1) > m = d_G(w_1)$.

If conditions $d_G(v_1) > d_G(w_1)$ are satisfied for every $w \in S(G) \wedge w \neq v$ and make the $u_1 = w$, then at most $C^2_{max}(G)$=$a_{2,3}a_{2,4} \cdots a_{2,n}$ with $a_{2,3} = a_{2,4} = \cdots = a_{2,t+2} = 1$, $a_{2,t+3} = a_{2,t+4} = \cdots =$

$a_{2,s+1} = 0$, $a_{2,s+2} = a_{2,s+3} = \cdots = a_{2,s+m-t} = 1$, $a_{2,s+m-t+1} = a_{2,s+m-t+2} = \cdots = a_{2,n} = 0$ (see (1)). Otherwise, if let the $u_1 = v$, then $C^2_{max}(G) = a_{2,3}a_{2,4}\cdots a_{2,n}$ with $a_{2,3} = a_{2,4} = \cdots = a_{2,t+2} = 1$, $a_{2,t+3} = a_{2,t+4} = \cdots = a_{2,s+1} = 0$, $a_{2,s+2} = a_{2,s+3} = \cdots = a_{2,s+l-t} = 1$, $a_{2,s+l-t+1} = a_{2,s+l-t+2} = \cdots = a_{2,n} = 0$ (see (1)).

Because $l > m$, then the binary number $a_{2,s+2}a_{2,s+3}\cdots a_{2,s+l-t}$ > the binary number $a_{2,s+2}a_{2,s+3}\cdots a_{2,s+m-t}$. Hence, Theorem 4 is established. □

Conjecture 1. *Assume that $G = (V(G), E(G))$ is an undirected graph of n vertices and $S(G) = \{u|\ u \in V(G) \wedge d(u) = \Delta(G) = s\}$ with $|S(G)| > 1$. If there is a node $v \in S(G)$ meeting conditions $C_{max}(G - v) \leq C_{max}(G - w)$ for each $w \in S(G) \wedge w \neq v$, then $u_1 = v$ for $MaxQ(G)$.*

3.1.2. Calculate the Intermediate Nodes of $MaxQ(G)$

If our algorithm has calculated the first vertex u_1 of $MaxQ(G)$, how it determines the subsequent vertices for computing $C_{max}(G)$? Observe that a side of G corresponds to 1 bit of the upper triangular part of the adjacency matrix $A(G)$. To maximize $C(G)$ by maximizing $C^2(G)$, one must make u_2 belong to $N(u_1)$ such that makes $a_{1,2} = 1$ (see (1)). Otherwise, if $u_2 \notin N(u_1)$, then $a_{1,2} = 0$ (see (1)) and $C(G) \neq C_{max}(G)$. The subsequent Proposition 4 captures the essence of the previous discussion.

Proposition 4. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. Given the first vertex u_1 of $MaxQ(G)$, then the choice of u_2 is from $N(u_1)$ for getting $C_{max}(G)$.*

Lemma 8. *Suppose $G = (V(G), E(G))$ is an undirected graph of n vertices. Given the first vertex u_1 of $MaxQ(G)$, if there is a single vertex $v \in S(N(u_1)) = \{u|u \in V(N(u_1)) \wedge d(u) = \Delta(N(u_1))\}$, then $u_2 = v$ for $MaxQ(G)$.*

Proof. By Proposition 4, it can be seen that the choice of u_2 is from $N(u_1)$ for obtaining $C_{max}(G)$. By the condition of Lemma 8, we have that v is the only node of $S(N(u_1))$. Therefore Lemma 8 holds. □

Theorem 5. *Assume that $G = (V(G), E(G))$ be an undirected connected graph of n vertices and $S(G) = \{u|\ u \in V(G) \wedge d(u) = \Delta(G) = s\}$. For calculating $C_{max}(G)$, if $MaxQ(G)$ already includes the first vertex $u_1 = v$ with $V(N(v)) = \{v_1, v_2, \cdots, v_s\}$ meeting condition $d_{N(v)}(v_1) = \Delta(N(v))$ and one of the following conditions is satisfied, then $u_2 = v_1$ for $MaxQ(G)$.*

1. *$d_{N(v)}(v_1) > d_{N(v)}(v_i)$ for $i = 2, \cdots, s$.*
2. *$d_G(v_1) > d_G(v_i)$ hold for $d_{N(v)}(v_1) = d_{N(v)}(v_i)$ with $i \in \{2, \cdots, t\}$.*

Proof. (1). To maximize $C(G)$ by maximizing $C^2(G)$ (see (1)), it follows that u_2 must satisfy condition $d_{N(v)}(u_2) = \Delta(N(v))$. By $d_{N(v)}(v_1) = \Delta(N(v))$ and the condition (1) of Theorem 5, $d_{N(v)}(v_1) = \Delta(N(v)) > d_{N(v)}(v_i)$ hold for $i = 2, \cdots, s$. Assume that $d_{N(v)}(v_1) = t$. If $u_2 = v_1$, it can be seen that $a_{2,3} = 1, a_{2,4} = 1, \cdots, a_{2,t+2} = 1$ by properly arranging nodes $v_2, v_3, \cdots, v_s$ of $V(N(v))$ (see (1)). Otherwise, if $u_2 = v_i$ with $i \in \{2, \cdots, s\}$, regardless of how the nodes in $N(v)$ are arranged such that there exists at least one 0 among the t elements $a_{2,3}, a_{2,4}, \cdots, a_{2,t+2}$ since $d_{N(v)}(v_i) < d_{N(v)}(v_1) = t$ for $i = 2, \cdots, s$ (see (1)). Hence, the result (1) of Theorem 5 follows. (2). To maximize $C(G)$ by maximizing $C^2(G)$ (see (1)), it follows that u_2 must satisfy condition $d_{N(v)}(u_2) = \Delta(N(v))$. By $d_{N(v)}(v_1) = \Delta(N(v))$ and the condition (2) of Theorem 5, we have that if $u_2 = v_1$, then $C^2(G)$ is the largest canonical labeling *slice* 2. Otherwise, if $u_2 = v_i$ with $i \in \{2, \cdots, s\}$, regardless of how the nodes in $N(v)$ are arranged such that the corresponding $C^2(G)$ is not largest (see (1)). Hence, the result (2) of Theorem 5 follows. □

Our algorithm uses an adjacency list to store a graph G. To facilitate the calculation of the *open k-neighborhood subgraph* of a node v in G, it in advance saves all the adjacent nodes, *chord edges*and *incident edges* of v into an array, respectively. Moreover v contains three stand alone pointer to point to the start position of each array.

If our algorithm has calculated the first i nodes $u_1, u_2, \cdots, u_i$ of $MaxQ(G)$, how does it decide the following nodes $u_{i+1}, u_{i+2}, \cdots, u_n$ for computing $C_{max}(G)$? From the previous discussion for getting u_2, it can be noted that the choices of the subsequent nodes $u_{i+1}, u_{i+2}, \cdots, u_n$ of $MaxQ(G)$ are from $N(S)$ with $S = \{u_1, u_2, \cdots, u_i\}$.

Each vertex v of G is attached a characteristic *m_NearestNode*. Once the ith vertex u_i is added into $MaxQ(G)$, it records the index data i of u_i in the property field *m_NearestNode* of every vertex $v_j \in N(u_i) = \{v_1, v_2, \cdots, v_t\}$ with $j = 1, 2, \cdots, t$. If $v_j- > m\_NearestNode = +\infty$, then let $v_j- > m\_NearestNode = i$ for every $v_j \in N(u_i)$ with $j = 1, 2, \cdots, t$.

Lemma 9. *Assume that $G = (V(G), E(G))$ is an undirected connected graph of n vertices and $S(G) = \{u | u \in V(G) \wedge d(u) = \Delta(G) = s\}$ with $|S(G)| > 1$. If $u_1 = v \in S(G)$ with neighborhood subgraph $N(v)$, $V(N(v)) = \{v_1, v_2, \cdots, v_s\}$, then $u_2, u_3, \cdots, u_s, u_{s+1} \in \{v_1, v_2, \cdots, v_s\}$.*

Proof. From (1) and the condition $\triangle(G) = s$, it follows that $C^1_{max}(G) = a_{1,2}a_{1,3} \cdots a_{1,n}$ with $a_{1,2} = a_{1,3} = \cdots = a_{1,s+1} = 1$ and $a_{1,s+2} = a_{1,s+3} = \cdots = a_{1,n} = 0$ for obtaining $C_{max}(G)$. To ensure $a_{1,2} = a_{1,3} = \cdots = a_{1,s+1} = 1$ to maximize $C^1(G)$ (see (1)), it follows that $u_2, u_3, \cdots, u_s, u_{s+1} \in \{v_1, v_2, \cdots, v_s\}$. □

Definition 15. *Assume that $A = (a_{i,j})_{n\times n}$ and $B = (b_{i,j})_{n\times n}$ are two matrices with $a_{i,j}$, $b_{i,j} = 0, 1$ for $i, j = 1, 2, \cdots, n$. Then, the lexicographic order of the two matrix is defined as follows:*

1. *$A = B$, if $a_{i,j} = b_{i,j}$ for all $1 \le i, j \le n$.*
2. *$A < B$, if $\exists i, j, 1 \le i, j \le n$ meeting conditions $a_{k,l} = b_{k,l}$ for all $k \le i$, $l \le j$ and $a_{i,j+1} < b_{i,j+1}$ with $j < n$ or $a_{i+1,1} < b_{i+1,1}$ with $i < n, j = n$.*

Suppose X is a matrix. If there exists at least one positive entry and the remaining entries are 0, we say $X > 0$. Otherwise, if all entries of X are 0, we say $X = 0$.

Theorem 6 (Diffusion Theorem)**.** *Suppose $G = (V(G), E(G))$ is an undirected connected graph of n vertices. If $MaxQ(G)$ already includes the first m nodes $u_1, u_2, \cdots, u_m$ for calculating $C_{max}(G)$, then the following two results follow.*

1. *the selection of the $(m + 1)$th vertex for computing $C_{max}(G)$ is from the open neighborhood subgraph $N(Q)$ of the vertices set $Q = \{u_1, u_2, \cdots, u_m\}$.*
2. *the vertex-induced subgraph of the first m nodes is connected.*

Proof. (1). We prove by contradiction. If $u_{m+1} \notin V(N(Q))$, without loss of generality let us assume that $\pi_1 = \{u_1, u_2, \cdots, u_m, v_1, u_{m+2}, \cdots, v_2, \cdots, u_n\}$ is a permutation of $V(G)$, meeting conditions $v_1 = u_{m+1} \notin V(N(Q)), v_2 = u_i \in V(N(Q))$ with $m + 2 \le i \le n$.

Further, assume that if condition $u_{m+1} \notin V(N(Q))$ is satisfied, the $C(G)$ associated with π_1 is the largest. Assume that the vertex $v_2 = u_i \in V(N(Q))$ is the vertex whose index i in π_1 is the smallest index in π_1 than the indexes of any other vertices belonging to $V(N(Q))$ in π_1. This indicates that no vertex belonging to $V(N(Q))$ is between u_{m+2} and u_{i-1} of π_1 such that for each node $v \in \{u_{m+2}, u_{m+3}, \cdots, u_{i-1}\}$, $v \notin V(N(Q))$ is met.

Let $A_1(G)$ be the matrix associated with the arrangement π_1. Let W_1, W_2, W_3 and W_4 be the block submatrices of $A_1(G)$ including the first m rows and the $(m + 1)$th column, the $(m + 2)$th to $(i - 1)$th columns, the ith column and the $(i + 1)$th to nth columns, respectively.

Since $v_1 \notin V(N(Q))$, then $W_1 = 0$ is satisfied. Alike the above consequence acquired, for each vertex $v \in \{u_{m+2}, u_{m+3}, \cdots, u_{i-1}\}$, $v \notin V(N(Q))$ is true such that $W_2 = 0$. Clearly $W_3 > 0$ for $v_2 \in V(N(Q))$.

By merely swapping v_1 and v_2 of π_1, one can obtain another permutation $\pi_2 = \{u_1, u_2, \cdots, u_m, v_2, u_{m+2}, \cdots, v_1, \cdots, u_n\}$ with $v_1 \notin V(N(Q)), v_2 \in V(N(Q))$.

Alike $A_1(G)$, let $A_2(G)$ be the matrix associated with the permutation π_2. Let Y_1, Y_2, Y_3 and Y_4 be the block submatrices of $A_2(G)$ containing the first m rows and the $(m+1)$th column, the $(m+2)$th to $(i-1)$th columns, the ith column and the $(i+1)$th to nth columns, respectively.

Definitely, $Y_1 > 0$ follows for $v_2 \in V(N(Q))$. For each node $v \in \{u_{m+2}, u_{m+3}, \cdots, u_{i-1}\}$, as $v \notin V(N(Q))$ is true, then $Y_2 = 0$. Since $v_1 \notin V(N(Q))$, then $Y_3 = 0$ holds.

Observe that $W_4 = Y_4$ since W_4 and Y_4 are both the $m \times (n-i)$ block submatrices associated with the same vertices sequence $u_{i+1}, u_{i+2}, \cdots, u_n$.

It follows from the results discussed above that $W_1 = W_2 = 0$, $W_3 > 0$ and $Y_1 > 0$, $Y_2 = Y_3 = 0$.

Hence, the new $C(G)$ defined by $A_2(G)$ is larger than the $C(G)$ defined by $A_1(G)$ such that makes a contradiction with the former hypothesis that $u_{m+1} \notin N(Q)$. This contradiction proves that conclusion (1) is correct.

(2). Conclusion (1) immediately leads to the result. □

Theorem 7 (Nearest Node Theorem)**.** *Suppose $G = (V(G), E(G))$ is an undirected connected graph of n vertices. If $MaxQ(G)$ already includes the first m nodes $u_1, u_2, \cdots, u_m$ for calculating $C_{max}(G)$. Assume vertices set $Q = \{u_1, u_2, \cdots, u_m\}$.*

If there exists a vertex $v \in V(N(Q))$, for every $w \in V(N(Q)) \wedge w \neq v$, meeting conditions $v- > m\_NearestNode < w- > m\_NearestNode$, then the $(m+1)$th node in $MaxQ(G)$ is v.

Proof. By Diffusion Theorem 6, we know that the $(m+1)$th vertex in $MaxQ(G)$ is from $N(Q)$. We prove by contradiction. If the $(m+1)$th vertex in $MaxQ(G)$ is $w \in V(N(Q)) \wedge w \neq v$, satisfying condition $v- > m\_NearestNode < w- > m\_NearestNode$.

Let $V_1 = V(G) - Q - v$ and $V_2 = V(G) - Q - w$. Without loss of generality, let us assume that $\pi_1 = \{u_1, u_2, \cdots, u_m, v, u^1_{m+2}, \cdots, u^1_n\}$ is a permutation of $V(G)$ corresponding to v with $u^1_{m+2} \in V_1, \cdots, u^1_n \in V_1$. Let us assume that $\pi_2 = \{u_1, u_2, \cdots, u_m, w, u^2_{m+2}, \cdots, u^2_n\}$ is a permutation of $V(G)$ corresponding to w with $u^2_{m+2} \in V_2, \cdots, u^2_n \in V_2$.

This means that the $C(G)$ corresponding to π_1 is less than the $C(G)$ corresponding to π_2.

Let $i = v- > m\_NearestNode$, then $v \notin V(N(u_1)), v \notin V(N(u_2)), \cdots, v \notin V(N(u_{i-1}), v \in V(N(u_i))$.

By $v- > m\_NearestNode < w- > m\_NearestNode$, there are $w \notin V(N(u_1))$, $w \notin V(N(u_2))$, $\cdots$, $w \notin V(N(u_{i-1}))$, $w \notin V(N(u_i))$.

Let $A_1(G)$ be the matrix associated with the permutation π_1. Let W_1 and W_2 be the block submatrices of $A_1(G)$ defined by the first i row and the $(m+1)$th column and the $(m+2)$th to nth columns, respectively.

Let $A_2(G)$ be the matrix associated with the permutation π_2. Let Y_1 and Y_2 be the block submatrices of $A_2(G)$ defined by the first i row and the $(m+1)$th column and the $(m+2)$th to nth columns, respectively.

Since $v \in V(N(u_i))$, then $W_1 > 0$ holds. Clearly $Y_1 = 0$ holds due to $w \notin V(N(u_i))$. Therefore, $W_1 > Y_1$. Again let $T = \{u_1, u_2, \cdots, u_{i-1}\}$.

For π_1, $\forall x \in V_1 = V(G) - Q - v$, there must be $x \notin V(N(T))$. Then $v- > m\_NearestNode < x- > m\_NearestNode$. Therefore $W_2 = 0$.

For π_2, $\forall x \in V_2 = V(G) - Q - w$, there must be $x \notin V(N(T)) \vee x = v$.

If $x \notin V(N(T)))$, then $v- > m\_NearestNode < x- > m\_NearestNode$. The elements of the column vector corresponding to x in Y_2 are all 0.

Otherwise, if $x = v$, the elements of the column vector corresponding to x in Y_2 are not all 0.

Thus, for π_2, the matrix Y_2 is such a matrix with only one column vector whose elements are not all 0 and the remaining column vectors are zero.

Hence, no matter how $u^1_{m+2}, \cdots, u^1_n$; $u^2_{m+2}, \cdots, u^2_n$ is taken, it can be seen that $W_2 = 0, Y_2 \neq 0$.

Let $W = W_1 W_2$ be a new matrix by combining W_1 and W_2.

Let $Y = Y_1 Y_2$ be a new matrix by combining Y_1 and Y_2.

From the previous analysis, it follows that $W = W_1W_2 = W_1O$ where $O = 0$ is $i \times (n-m-1)$ block submatrix.

Further, it follows that $Y = Y_1Y_2 = OY_2$ where $O = 0$ is a column vector of i rows. The block submatrix Y_2 is such a matrix whose only one column vector, denoted by Z, is not 0 and whose remaining column vectors are all 0. It can be seen that $Z = W_1$.

By the comparison of the matrix W and Y, it holds that $W > Y$.

Hence, the $C(G)$ defined by $A_1(G)$ is larger than the $C(G)$ defined by $A_2(G)$. This contradicts the previous hypothesis that the $C(G)$ corresponding to π_1 is less than the $C(G)$ corresponding to π_2. Therefore, the conclusion of Theorem 7 holds. □

Suppose $G = (V(G), E(G))$ is an undirected connected graph of n vertices. For calculating $C_{max}(G)$, assume that $MaxQ(G)$ already includes the first m nodes $u_1, u_2, \cdots, u_m$. Let $Q = \{u_1, u_2, \cdots, u_m\}$ with open neighborhood subgraph $N(Q)$(see Definition 7). Let $R = V(G) - Q$ with vertex-induced subgraph $G[R]$.

3.2. Calculate $C_{max}(G)$ for a Disconnected Graph

Suppose $G = (V(G), E(G))$ is a disconnected undirected graph of n vertices with p connected components $G_1, G_2, \cdots, G_p$. In this subsection, we consider how to determine the *canonical labeling* $C_{max}(G)$ of G.

If $\Delta(G_1) > \Delta(G_2) > \cdots > \Delta(G_p)$, how does our algorithm proceed to determine the *canonical labeling* $C_{max}(G)$ of G? It can be seen that to get $C_{max}(G)$ one had to order all nodes of every connected component G_i with $i \in \{1, 2, \cdots, p\}$ together when building the adjacency matrix $A(G)$. The outcome also holds from the proving of Diffusion Theorem 6.

First, we examine the characteristics of the adjacency matrix $A(G)$. When building the adjacency matrix $A(G)$, we order all nodes of every connected component G_i with $i \in \{1, 2, \cdots, p\}$ together. Observe that the adjacency matrix $A(G)$ is a symmetric block matrix, every block of which is associated with a connected component. Next, we examine the link between $C(G)$ and $A(G)$. Besides, we present how to determine the $C_{max}(G_1)$ associated with the adjacency matrix $A(G)$.

Lemma 10. *Suppose $G = (V(G), E(G))$ is a disconnected undirected graph that have two disjoint connected components $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ with k and l vertices respectively. Assume that*

$$\begin{aligned} C_{max}(G_1) &= C^1_{max}(G_1)C^2_{max}(G_1)\cdots C^{k-1}_{max}(G_1), \\ C_{max}(G_2) &= C^1_{max}(G_2)C^2_{max}(G_2)\cdots C^{l-1}_{max}(G_2). \end{aligned}$$

If $\Delta(G_1) > \Delta(G_2)$, then $C_{max}(G)$ meets the following equation:

$$C_{max}(G) = C^1_{max}(G)C^2_{max}(G)\cdots C^{k-1}_{max}(G)C^k_{max}(G)C^{k+2}_{max}(G)\cdots C^{k+l-1}_{max}(G), \tag{12}$$

where

$$\begin{aligned} &C^1_{max}(G) = C^1_{max}(G_1)\overbrace{00\cdots0}^{l}, && C^2_{max}(G) = C^2_{max}(G_1)\overbrace{00\cdots0}^{l} && \cdots\cdots\cdots\cdots\cdots\cdots \\ &C^{k-1}_{max}(G) = C^{k-1}_{max}(G_1)\overbrace{00\cdots0}^{l}, && C^k_{max}(G) = \overbrace{00\cdots0}^{l}, && C^{k+1}_{max}(G) = C^1_{max}(G_2), \\ &C^{k+2}_{max}(G) = C^2_{max}(G_2), && \cdots\cdots\cdots\cdots\cdots\cdots, && C^{k+l-1}_{max}(G) = C^{l-1}_{max}(G_2). \end{aligned}$$

Proof. If $\Delta(G_1) > \Delta(G_2)$ is satisfied, then $\Delta(G) = \Delta(G_1)$. By Proposition 3, it can be seen that to get $C_{max}(G)$, one must pick the node with the maximum degree from G_1 as the first node u_1 of $MaxQ(G)$.

By Proposition 4, it follows that the second vertex u_2 of $MaxQ(G)$ must be from $N(u_1)$. By Diffusion Theorem 6, the following $k-2$ nodes of $MaxQ(G)$ must be picked from G_1. Furthermore,

by Diffusion Theorem 6, the next l nodes of $MaxQ(G)$ must be chosen from G_2. Thoughtfully considering (1), it is not hard to discover that (12) follows. □

Observe that to guarantee the maximization of $C_{max}(G)$, one has to add l 0 after $C^1_{max}(G_1)$, $C^2_{max}(G_1), \cdots, C^{k-1}_{max}(G_1)$ respectively, so that make $C^k_{max}(G)$ be equal l 0.

Theorem 8. *Suppose $G = (V(G), E(G))$ is a disconnected undirected graph of n vertices with p connected components $G_1, G_2, \cdots, G_p$ satisfying $|V(G_1)| = n_1$, $|V(G_2)| = n_2, \cdots, |V(G_p)| = n_p$. If $C_{max}(G_1) > C_{max}(G_2) > \cdots > C_{max}(G_p)$, then $C_{max}(G)$ meets the following equation:*

$$\begin{aligned} C_{max}(G) \quad = \quad & C^1_{max}(G_1)\overbrace{0\cdots0}^{n-n_1}\cdots C^{n_1-1}_{max}(G_1)\overbrace{0\cdots0}^{n-n_1}\overbrace{0\cdots0}^{n-n_1} \\ & C^1_{max}(G_2)\ \overbrace{0\cdots0}^{n-n_1-n_2}\ \cdots C^{n_2-1}_{max}(G_2)\ \overbrace{0\cdots0}^{n-n_1-n_2}\ \overbrace{0\cdots0}^{n-n_1-n_2} \\ & \qquad\qquad\qquad\vdots \\ & C^1_{max}(G_{p-1})\overbrace{0\cdots0}^{n_p}\cdots C^{n_{p-1}-1}_{max}(G_{p-1})\overbrace{0\cdots0}^{n_p}\overbrace{0\cdots0}^{n_p} \\ & C^1_{max}(G_p)\cdots C^{n_p-1}_{max}(G_p). \end{aligned} \tag{13}$$

Proof. We prove (13) by induction on the number p of branches. By the previous definition, it follows that $C_{max}(G) = C^1_{max}(G)C^2_{max}(G)\cdots C^{n-1}_{max}(G)$ for $p = 1$. Hence, (13) in Theorem 8 follows for $p = 1$. By Lemma 10, (13) is correct for $p = 2$.

By induction, assume that (13) is true for $p = k$. In the following, let us prove that Equation (13) is also correct for $p = k + 1$. We can now think of the front k branch graphs as graph H. Therefore, (13) is also true for graph H.

$$\begin{aligned} C_{max}(H) = & \\ & C^1_{max}(G_1)\ \overbrace{0\cdots0}^{n-n_{k+1}-n_1}\ \cdots C^{n_1-1}_{max}(G_1)\ \overbrace{0\cdots0}^{n-n_{k+1}-n_1}\ \overbrace{0\cdots0}^{n-n_{k+1}-n_1} \\ & C^1_{max}(G_2)\quad \overbrace{0\cdots0}^{n-n_{k+1}-n_1-n_2}\quad \cdots C^{n_2-1}_{max}(G_2)\quad \overbrace{0\cdots0}^{n-n_{k+1}-n_1-n_2}\quad \overbrace{0\cdots0}^{n-n_{k+1}-n_1-n_2} \\ & \qquad\qquad\qquad\vdots \\ & C^1_{max}(G_{k-1})\overbrace{0\cdots0}^{n_k}\cdots C^{n_{k-1}-1}_{max}(G_{k-1})\overbrace{0\cdots0}^{n_k}\overbrace{0\cdots0}^{n_k} \\ & C^1_{max}(G_k)C^2_{max}(G_k)\cdots C^{n_k-1}_{max}(G_k), \end{aligned} \tag{14}$$

where

$$\begin{aligned} C^1_{max}(H) &= C^1_{max}(G_1)\ \overbrace{0\cdots0}^{n-n_{k+1}-n_1}, \\ &\vdots \\ C^{n_1-1}_{max}(H) &= C^{n_1-1}_{max}(G_1)\ \overbrace{0\cdots0}^{n-n_{k+1}-n_1}, \\ C^{n_1}_{max}(H) &= \overbrace{0\cdots0}^{n-n_{k+1}-n_1}, \\ C^{n_1+1}_{max}(H) &= C^1_{max}(G_2)\quad \overbrace{0\cdots0}^{n-n_{k+1}-n_1-n_2}, \end{aligned}$$

$$
\begin{aligned}
&\vdots\\
&C_{max}^{n_1+n_2-1}(H) = C_{max}^{n_2-1}(G_2)\overbrace{0\cdots0}^{n-n_{k+1}-n_1-n_2},\\
&C_{max}^{n_1+n_2}(H) = \overbrace{0\cdots0}^{n-n_{k+1}-n_1-n_2},\\
&\vdots\\
&C_{max}^{n-n_{k+1}-n_k-n_{k-1}+1}(H) = C_{max}^{1}(G_{k-1})\overbrace{0\cdots0}^{n_k},\\
&\vdots\\
&C_{max}^{n-n_{k+1}-n_k-1}(H) = C_{max}^{n_{k-1}-1}(G_{k-1})\overbrace{0\cdots0}^{n_k},\\
&C_{max}^{n-n_{k+1}-n_k}(H) = \overbrace{0\cdots0}^{n_k},\\
&C_{max}^{n-n_{k+1}-n_k+1}(H) = C_{max}^{1}(G_k),\\
&C_{max}^{n-n_{k+1}-n_k+2}(H) = C_{max}^{2}(G_k),\\
&\vdots\\
&C_{max}^{n-n_{k+1}-1}(H) = C_{max}^{n_k-1}(G_k).
\end{aligned}
$$

By Lemma 10, we have

$$
\begin{aligned}
C_{max}(G) = &\\
&C_{max}^{1}(H)\overbrace{0\cdots0}^{n_{k+1}}C_{max}^{2}(H)\overbrace{0\cdots0}^{n_{k+1}}\cdots C_{max}^{n-n_{k+1}-1}(H)\overbrace{0\cdots0}^{n_{k+1}}\\
&\overbrace{0\cdots0}^{n_{k+1}}C_{max}^{1}(G_{k+1})C_{max}^{2}(G_{k+1})\cdots C_{max}^{n_{k+1}-1}(G_{k+1}).
\end{aligned}
\tag{15}
$$

By (14), substituting $C_{max}^{1}(H)$ to $C_{max}^{n-n_{k+1}-1}(H)$ into (15), we obtain

$$
\begin{aligned}
C_{max}(G) = &\\
&C_{max}^{1}(G_1)\overbrace{0\cdots0}^{n-n_{k+1}-n_1}\overbrace{0\cdots0}^{n_{k+1}}C_{max}^{2}(G_1)\overbrace{0\cdots0}^{n-n_{k+1}-n_1}\overbrace{0\cdots0}^{n_{k+1}}\cdots C_{max}^{n_k-1}(G_k)\\
&\overbrace{0\cdots0}^{n_{k+1}}\overbrace{0\cdots0}^{n_{k+1}}C_{max}^{1}(G_{k+1})C_{max}^{2}(G_{k+1})\cdots C_{max}^{n_{k+1}-1}(G_{k+1}).
\end{aligned}
\tag{16}
$$

Thus, we have

$$
\begin{aligned}
C_{max}(G) \quad = \quad & C_{max}^{1}(G_1)\overbrace{0\cdots0}^{n-n_1}\cdots C_{max}^{n_1-1}(G_1)\overbrace{0\cdots0}^{n-n_1}\overbrace{0\cdots0}^{n-n_1}\\
& C_{max}^{1}(G_2)\overbrace{0\cdots0}^{n-n_1-n_2}\cdots C_{max}^{n_2-1}(G_2)\overbrace{0\cdots0}^{n-n_1-n_2}\overbrace{0\cdots0}^{n-n_1-n_2}\\
& \qquad\qquad\vdots\\
& C_{max}^{1}(G_k)\overbrace{0\cdots0}^{n_{k+1}}\cdots C_{max}^{n_k-1}(G_k)\overbrace{0\cdots0}^{n_{k+1}}\overbrace{0\cdots0}^{n_{k+1}}\\
& C_{max}^{1}(G_{k+1})\cdots C_{max}^{n_{k+1}-1}(G_{k+1}).
\end{aligned}
\tag{17}
$$

Therefore, Equation (13) is correct for $p = k+1$. □

By Theorem 8, it can be observed that for getting $C_{max}(G)$, one has to first calculate $C_{max}(G_i)$ of every branch for $i = 1, 2, \cdots, p$, respectively. Besides, one must substitute $C_{max}(G_i)$ into (13) sequentially to obtain $C_{max}(G)$ of a disconnected undirected G.

If the above conditions are not met, how does one determine $C_{max}(G)$ of a disconnected undirected G? Based on an examination of the previous outcomes, the following Theorem 9 that is more useful than Lemma 10 is established.

Theorem 9. *Suppose $G = (V(G), E(G))$ is a disconnected undirected graph that have two disjoint connected components $G_1 = (V(G_1), E(G_1))$ and $G_2 = (V(G_2), E(G_2))$ with k and l vertices respectively. Let $S(G_1) = \{u | u \in V(G_1) \wedge d(u) = \Delta(G_1)\}$ and $S(G_2) = \{u | u \in V(G_2) \wedge d(u) = \Delta(G_2)\}$ meeting condition $\Delta(N(u)) > \Delta(N(v))$ for $\forall u \in S_1$ and $\forall v \in S_2$. Assume that*

$$C_{max}(G_1) = C^1_{max}(G_1)C^2_{max}(G_1)\cdots C^{k-1}_{max}(G_1), \quad (18)$$

$$C_{max}(G_2) = C^1_{max}(G_2)C^2_{max}(G_2)\cdots C^{l-1}_{max}(G_2), \quad (19)$$

If $\Delta(G_1) = \Delta(G_2)$, then $C_{max}(G)$ satisfies the following equation:

$$C_{max}(G) = C^1_{max}(G)C^2_{max}(G)\cdots C^{k-1}_{max}(G)C^k_{max}(G)C^{k+2}_{max}(G)\cdots C^{k+l-1}_{max}(G), \quad (20)$$

where

$$C^1_{max}(G) = C^1_{max}(G_1)\overbrace{00\cdots0}^{l}, \quad C^2_{max}(G) = C^2_{max}(G_1)\overbrace{00\cdots0}^{l} \quad \cdots\cdots\cdots\cdots\cdots\cdots$$

$$C^{k-1}_{max}(G) = C^{k-1}_{max}(G_1)\overbrace{00\cdots0}^{l}, \quad C^k_{max}(G) = \overbrace{00\cdots0}^{l}, \quad C^{k+1}_{max}(G) = C^1_{max}(G_2),$$

$$C^{k+2}_{max}(G) = C^2_{max}(G_2), \quad \cdots\cdots\cdots\cdots\cdots\cdots, \quad C^{k+l-1}_{max}(G) = C^{l-1}_{max}(G_2).$$

Proof. By the condition of Theorem 9, it follows that for $\forall u \in S(G_1)$ and $\forall v \in S(G_2)$ inequality $\Delta(N(u)) > \Delta(N(v))$ holds. By Theorem 3, one has to pick the first node u_1 of $MaxQ(G)$ from G_1 so that get $C_{max}(G)$.

By Diffusion Theorem 6, one must choose $u_2, u_3, \cdots, u_k$ into $MaxQ(G)$ from G_1 to get $C_{max}(G)$. Besides, by Diffusion Theorem 6, one has to pick the following l nodes into $MaxQ(G)$ from G_2. By (18) and (19), it can be seen that (21) is correct.

$$C_{max}(G) = C^1_{max}(G)C^2_{max}(G)\cdots C^{k-1}_{max}(G)C^k_{max}(G)C^{k+2}_{max}(G)\cdots C^{k+l-1}_{max}(G), \quad (21)$$

where

$$C^1_{max}(G) = C^1_{max}(G_1)\overbrace{00\cdots0}^{l}, \quad C^2_{max}(G) = C^2_{max}(G_1)\overbrace{00\cdots0}^{l}, \quad \cdots\cdots\cdots\cdots\cdots\cdots,$$

$$C^{k-1}_{max}(G) = C^{k-1}_{max}(G_1)\overbrace{00\cdots0}^{l}, \quad C^k_{max}(G) = \overbrace{00\cdots0}^{l}, \quad C^{k+1}_{max}(G) = C^1_{max}(G_2),$$

$$C^{k+2}_{max}(G) = C^2_{max}(G_2), \quad \cdots\cdots\cdots\cdots\cdots\cdots, \quad C^{k+l-1}_{max}(G) = C^{l-1}_{max}(G_2).$$

It can be observed that to guarantee the maximization of $C_{max}(G)$, one has to add l 0 after $C^1_{max}(G_1), C^2_{max}(G_1), \cdots, C^{k-1}_{max}(G_1)$ respectively and make $C^k_{max}(G)$ be equal l 0. □

4. Our Algorithms for Calculating the Canonical Labeling

In the Section, based on the outcomes of the previous sections, we offer our algorithms for calculating *canonical labelings* of graphs. We describe the main steps required for calculating the *canonical labeling* $C_{max}(G)$ of G. When our algorithm has computed the vertex u_1 of $MaxQ(G)$, then,

it builds the neighborhood subgraph $N(u_1)$ of the vertex u_1 (see Figures 3a and 4a), from which it chooses a few nodes into $MaxQ(G)$. For the convenience of description, we name this process *Procedure* 1. Then once more, it constructs the neighborhood subgraph $N(S_2)$ of the vertices set $S_2 = \{u_1, u_2\}$ (see Figures 3b and 4b), from which it chooses a few nodes into $MaxQ(G)$. We name this process *Procedure* 2. $\cdots$. Then once more, it constructs the neighborhood subgraph $N(S_r)$ of the vertices set $S_r = \{u_1, u_2, \cdots, u_r\}$ (see Figures 3c and 4c), from which it chooses a few nodes into $MaxQ(G)$. We name this process *Procedure r*. $\cdots$. This process lasts until it places all nodes in G into $MaxQ(G)$ (see Figures 3d and 4d).

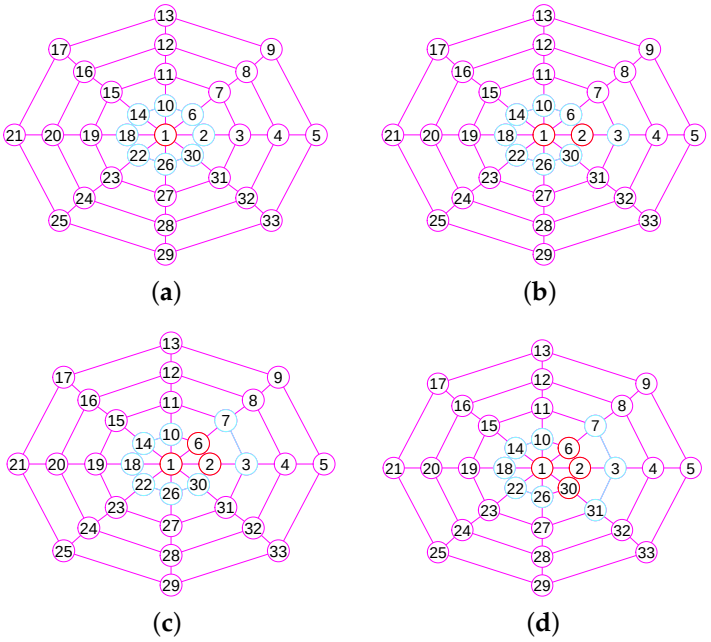

Figure 3. The 1-neighborhood subgraphs for the different nodes sets of a wheel graph G. (**a**) A wheel graph G and the 1-neighborhood subgraph $N(1)$ of the node 1; (**b**) The 1-neighborhood subgraph $N(S_1)$ of the nodes set $S_1 = \{1, 2\}$; (**c**) The 1-neighborhood subgraph $N(S_2)$ of the nodes set $S_2 = \{1, 2, 6\}$; (**d**) The 1-neighborhood subgraph $N(S_3)$ of the nodes set $S_3 = \{1, 2, 6, 30\}$.

For *Procedure* 1, after computing $N(S_1)$, by Lemma 4, our algorithm orders all vertices of $N(S_1)$ into a sole linkage L_1 (observe Algorithm 1). For clarify, make $V_1 = V(N(S_1))$. If there are two vertices $v_i, v_j \in L_1$ meeting condition $v_i \asymp v_j$ concerning $N(S_1)$, then it proceeds to decide whether $v_i \succeq v_j$ or $v_i \prec v_j$ concerning G. If $v_i \succeq v_j$, it repositions v_i in front of the v_j in L_1. Otherwise, it repositions v_i in back of the v_j in L_1.

For every *Procedure* r, $r = 2, 3, \cdots$, when computing $N(S_r)$, our algorithm successively calculates $V_2 = V(N(S_{r-1}) \cap N(u_r))$ (see Figure 5d), $V_3 = V(N(S_{r-1}) - N(u_r))$ (see Figure 5e), $V_4 = V(N(u_r) - N(S_{r-1}))$ (see Figure 5f) and the degree sequences $d_{N(S_r)}(V_1)$, $d_{N(S_r)}(V_3)$, $d_{N(S_r)}(V_4)$ in decreasing order respectively, where $S_{r-1} = \{u_1, u_2, \cdots, u_{r-1}\}$. It can be shown that $V_2 \cup V_3 \cup V_4 = V(N(S_r))$ and $V_i \cap V_j = \emptyset$ for $i \neq j, i, j = 2, 3, 4$. By Lemma 4, it orders all vertices of V_i into a sole linkage L_i (observe Algorithm 1) for the neighborhood subgraph $N(S_r)$ with $i = 2, 3, 4$, respectively.

Next, our algorithm successively executes the following processing paces for the vertices of L_i with $i = 2, 3, 4$:

1. Beginning from the front of L_2, it, in turn, decides whether every vertex $u \in L_2$ meets the *degree multiplicity* condition $dm_{N(S_r)}(u) = 1$. If the number of nodes meeting condition $dm_{N(S_r)}(u) = 1$ is less than 2 in L_2, it places u into $MaxQ(G)$. If there are two vertices $v_i, v_j \in L_2$ meeting condition $v_i \asymp v_j$ concerning $N(S_r)$, then it proceeds to decide whether $v_i \succeq v_j$ or $v_i \prec v_j$ concerning

G. If $v_i \succeq v_j$, it repositions the v_i in front of the v_j in L_2 (observe Algorithm 1). Otherwise, it repositions the v_i in back of the v_j in L_2 (view Algorithm 1).

2. Excluding the vertices added into $MaxQ(G)$, it utilizes a queue Q to save the middle vertices of $MaxQ(G)$. After executing Step 1, it consecutively decides whether or not each node $u \in L_2$ is in Q. If u is in Q and the number of vertices added into $MaxQ(G)$ is less than 2 in the previous process, it inserts u on the tail of $MaxQ(G)$ and concurrently deletes u from the head of Q. Otherwise, it places u on the back of Q.
3. For L_3, if the number of nodes of L_2, added into $MaxQ(G)$, is 0 and the number of nodes meeting condition $dm_{N(S_r)}(u)= 1$ with $u \in L_3$ is less than 2, it inserts u on the back of $MaxQ(G)$. The remaining procedure steps are the same as for L_2.
4. For L_4, if the number of nodes of L_2 and L_3, added into $MaxQ(G)$, is 0 and the number of nodes meeting the condition $dm_{N(S_r)}(u)= 1$ with $u \in L_4$ is less than 2, it places u on the back of $MaxQ(G)$. The remaining procedure steps are the same as for L_2.

Algorithm 1: Order all vertices of V_i into a sole linkage L_i for a neighborhood subgraph $N(S_r)$ with $i = 1, 2, 3, 4$, respectively where $V_1 = V(N(S_1))$.

Input :

1. An undirected connected graph G of n vertices and the neighborhood subgraph $N(S_r) = (V(N(S_r)), E(N(S_r)))$ of a vertices set $S_i \subseteq V(G)$.
2. A vertices set $V_i = \{v_1, v_2, \cdots, v_t\}$ deposited in an array *VertexArray*.

Output: A list L utilized to save and order all vertices of the linkage L_i.

```
1  l ← 1; j ← 1; sign ← 0 ;
2  Set the initial values of local parameters VertexMax, Vertex ;
3  Rank the degree sequence d_{N(S_r)}(V_i) = (d_{N(S_r)}(v_1), d_{N(S_r)}(v_2), ··· , d_{N(S_r)}(v_t)) in non-increasing
     order;
4  for (l ← 0 to t − 2){
5      VertexMax ← VertexArray[l];
6      for (j ← l + 1 to t − 1){
7          Vertex ← VertexArray[j];
8          sign ← 0;
9          Contrast_Two_Vertices(N(S_r), VertexMax, Vertex, sign); // observe Algorithm 2;
10         if (sign == 0)then // VertexMax ≻ Vertex regarding N(S_r)
11             move back to the start of the for-loop;
12         else if (sign == 1)then // VertexMax ≺ Vertex regarding N(S_r)
13             VertexMax ← Vertex;
14         else if (sign == 2)then // VertexMax ≍ Vertex regarding N(S_r)
15             sign ← 0;
16             Contrast_Two_Vertices(G, VertexMax, Vertex, sign); // observe Algorithm 2;
17             if (sign == 0 or 2)then // VertexMax ≽ Vertex regarding G
18                 move back to the start of the for-loop;
19             else if (sign == 1)then // VertexMax ≺ Vertex regarding G
20                 VertexMax ← Vertex;
21     Attach the VertexMax to the tail of the list L;
```

Algorithm 2: Contrast the *entire diffusion degree sequences* $d_H^\sigma[N_1(v)]$, $d_H^\sigma[N_2(v)]$, $\cdots$, $d_H^\sigma[N_{\rho(v)}(v)]$ and $d_H^\sigma[N_1(w)]$, $d_H^\sigma[N_2(w)]$, $\cdots$, $d_H^\sigma[N_{\rho(w)}(w)]$ of two vertices v and w in H.

```
1  void Contrast_Two_Vertices(Graph H, CNode v, CNode w, int &sign)
2  {
3      r ← 0; k ← 0; sign ← 0;
4      while ( r <= ρ(v) and r <= ρ(w)){ // r <= diffusion radius ρ(v) and ρ(w)
5          r ← r + 1;  k ← 1;
6          Determine the r neighborhood subgraph N_r(v) of v in H;
7          Determine the p connected components H_1, H_2, ···, H_p of N_r(v);
8          Work out the entire diffusion degree sequence d_H^σ[N_r(v)] = (d_σ(H_1), d_σ(H_2), ···,
             d_σ(H_p)), meeting conditions d_σ(H_1) ≥ d_σ(H_2) ≥ ··· ≥ d_σ(H_p);
9          Determine the r neighborhood subgraph N_r(w) of w in H;
10         Determine the l connected components J_1, J_2, ···, J_l of N_r(w);
11         Work out the entire diffusion degree sequence d_H^σ[N_r(w)] = (d_σ(J_1), d_σ(J_2), ···, d_σ(J_l)),
             meeting conditions d_σ(J_1) ≥ d_σ(J_2) ≥ ··· ≥ d_σ(J_l);
12         while (k <= p and k <= l){
13             sign ← 0;
14             // Contrast two diffusion degree sequences (observe Algorithm 3)
               Contrast_Two_Diffusion_Degree_Sequences(H, N_r(v), N_r(w), d_σ(H_k), d_σ(J_k),
               sign);
15             if (sign == 0 or sign == 1)then // v ≻ w or v ≺ w regarding H
16                 return;
17             k ← k + 1;
18         if ( k <= p and k > l)then
19             sign ← 0; return; // v ≻ w regarding H;
20         else if ( k > p and k <= l)then
21             sign ← 1; return; // v ≺ w regarding H;
22     if ( r <= ρ(v) and r > ρ(w))then
23         sign ← 0; return; // v ≻ w regarding H;
24     else if ( r > ρ(v) and r <= ρ(w))then
25         sign ← 1; return; // v ≺ w regarding H;
26     else
27         sign ← 2; return; // v ≍ w regarding H;
28 }
```

Algorithm 3: Contrast two *diffusion degree sequences* $d_\sigma(H_k)$ and $d_\sigma(J_k)$ of two vertices v and w in H.

```
1  void Contrast_Two_Diffusion_Degree_Sequences(Graph H, Graph N_r(v), Graph N_r(w),
     QueueArray d_σ(H_k), QueueArray d_σ(J_k), int &sign)
2  {
3      d_1 ← 0; d_2 ← 0; m ← 0; l ← 1;
4      Assume diffusion degree sequences d_σ(H_k) = (d_{N_r(v)}(V_0), d_{N_r(v)}(V_1), ··· , d_{N_r(v)}(V_s));
       // Meet V_0 ∨ V_1 ∨ ··· ∨ V_s = V(H_k) and V_i ∩ V_j = ∅ for i ≠ j, i, j = 0, 1, ··· , s;
5      Assume diffusion degree sequences d_σ(J_k) = (d_{N_r(w)}(U_0), d_{N_r(w)}(U_1), ··· , d_{N_r(w)}(U_t));
       // Meet U_0 ∨ U_1 ∨ ··· ∨ U_t = V(J_k) and U_i ∩ U_j = ∅ for i ≠ j, i, j = 0, 1, ··· , t;
6      while ( m <= s and m <= t){ // m <= s and m <= t
7          Assume the degree sequence d_{N_r(v)}(V_m) = (d_{N_r(v)}(a_1), d_{N_r(v)}(a_2), ··· , d_{N_r(v)}(a_j), ··· ,
             d_{N_r(v)}(a_λ)) in non-increasing order;// Meet V_m = {a_1, a_2, ··· , a_λ};
8          Assume the degree sequence d_{N_r(w)}(U_m) = (d_{N_r(w)}(b_1), d_{N_r(w)}(b_2), ··· , d_{N_r(w)}(b_j), ··· ,
             d_{N_r(w)}(b_μ)) in non-increasing order;// Meet U_m = {b_1, b_2, ··· , b_μ};
9          while (l <= λ and l <= μ){
10             d_1 ← d_{N_r(v)}(a_l);   d_2 ← d_{N_r(w)}(b_l);
11             // Contrast d_1 and d_2 (observe Algorithm 4);
12             Contrast_d_1_and_d_2(m, d_1, d_2, sign);
13             if ( sign == 0 or sign == 1)then
14                 return; // d_σ(H_k) ≠ d_σ(J_k) regarding H;
15             l ← l + 1;
16         if ( l <= λ and l > μ)then
17             sign ← 0; return; // d_σ(H_k) > d_σ(J_k) regarding H;
18         else if ( l > λ and l <= μ)then
19             sign ← 1; return; // d_σ(H_k) < d_σ(J_k) regarding H;
20         m ← m + 1;  l ← 1;
21     if ( m <= s and m > t)then
22         sign ← 0; return; // d_σ(H_k) > d_σ(J_k) regarding H;
23     else if ( m > s and m <= t)then
24         sign ← 1; return; // d_σ(H_k) < d_σ(J_k) regarding H;
25     else
26         sign ← 2; return; // d_σ(H_k) == d_σ(J_k) regarding H;
27 }
```

Algorithm 4: Contrast d_1 and d_2.

```
1  void Contrast_d1_and_d2(int m, int d1, int d2, int &sign)
2  {
3      sign ← 0;
4      if (m == 0)then
5          if (d1 < d2)then
6              sign ← 0; return;
7          else if ( d1 > d2)then
8              sign ← 1; return;
9      else
10         if (d1 > d2)then
11             sign ← 0; return;
12         else if ( d1 < d2)then
13             sign ← 1; return;
14     sign ← 2; return;
15 }
```

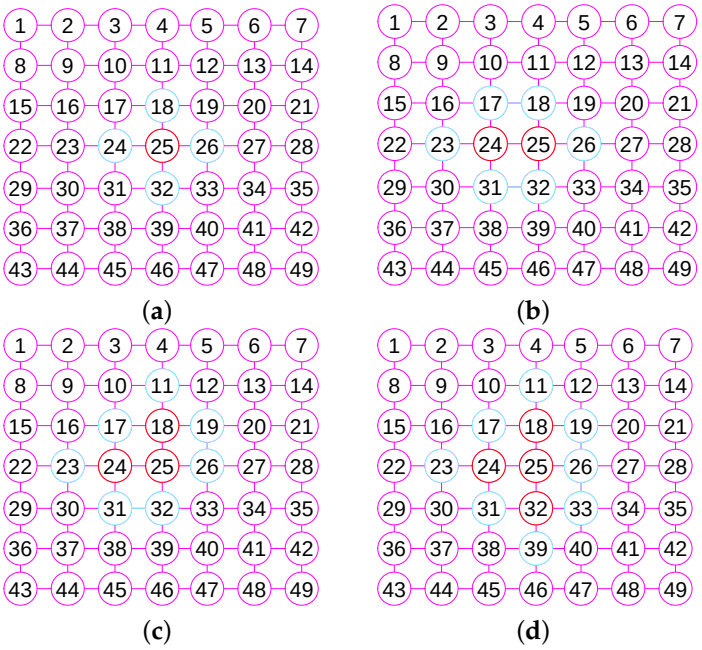

Figure 4. The 1-neighborhood subgraphs for the different nodes sets of the 7×7 grid graph $G_{7,7}$. (**a**) A graph G and the 1-neighborhood subgraph $N(25)$ of the node 25; (**b**) The 1-neighborhood subgraph of the nodes set $S_1 = \{24, 25\}$; (**c**) The 1-neighborhood subgraph of the nodes set $S_2 = \{24, 25, 18\}$; (**d**) The 1-neighborhood subgraph of the nodes set $S_3 = \{24, 25, 18, 32\}$.

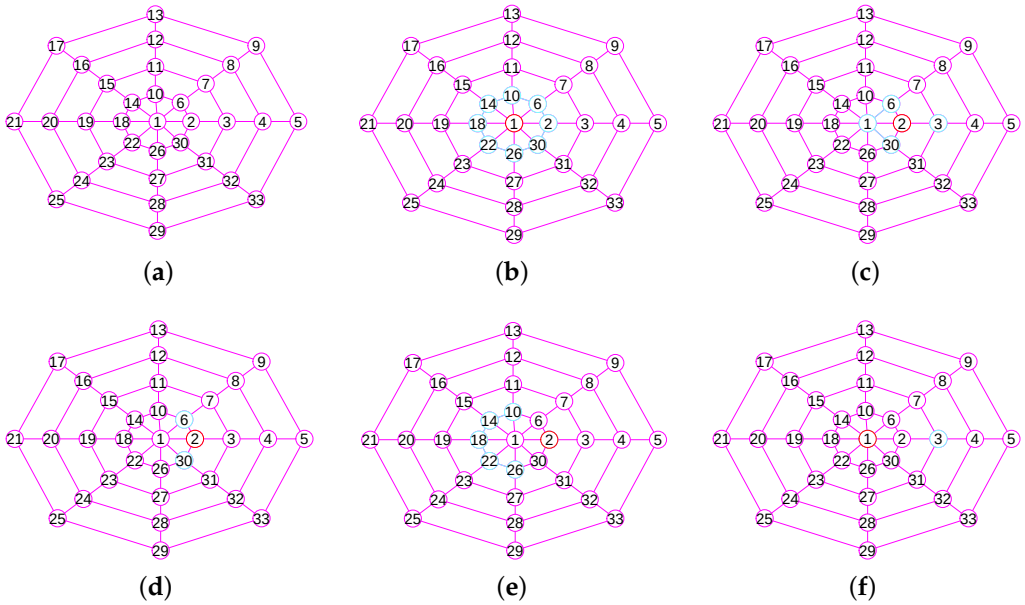

Figure 5. A wheel graph G, the 1-neighborhood subgraphs $N(1)$ and $N(2)$ and the three related vertices sets produced by the boolean operations of $N(1)$ and $N(2)$. (**a**) A wheel graph G; (**b**) The neighborhood subgraph $N(1)$; (**c**) The neighborhood subgraph $N(2)$; (**d**) $V(N(1)) \cap V(N(2)) = \{6, 30\}$; (**e**) $V(N(1)) - V(N(2)) = \{10, 14, 18, 22, 26\}$; (**f**) $V(N(2)) - V(N(1)) = \{3\}$.

Our algorithm utilizes an array $MaxQ$ to save the vertices of $MaxQ(G)$ and an array Q to store the vertices to be added to $MaxQ(G)$ briefly. Our algorithm has been optimized by Lemmas 2 and 3.

The results of experiments show that our method is a new approach by which one can precisely determine *TopMost* graphs (defined in Section 2) for many classes of graphs, including trees, grid graphs, wheel graphs, hypercube graphs, king graphs, triangular graphs and so on. Figures 6–23 given by our software display the accuracy of our software for computing *TopMost* graphs of these graph classes aforesaid.

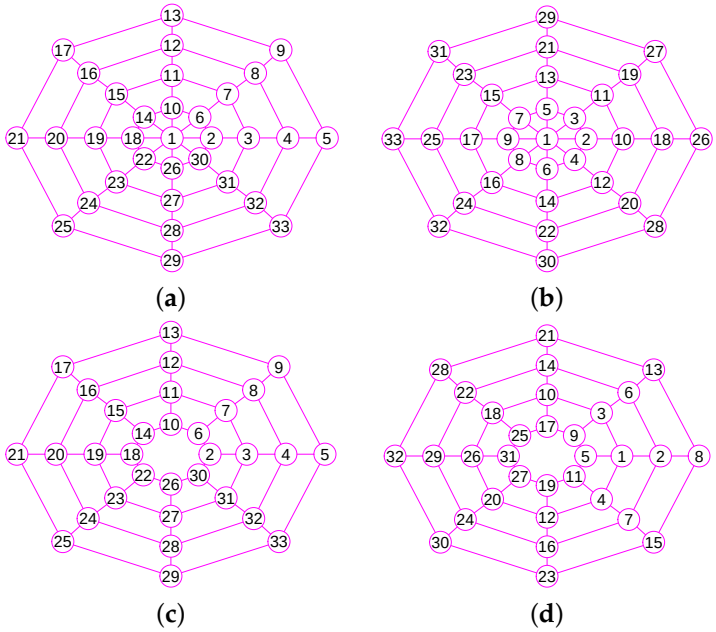

Figure 6. The *TopMost* graphs of two graphs, including a wheel graph G and the graph $G - u$ with $u = 1 \in V(G)$. (**a**) A wheel graph G with 33 nodes and 64 edges; (**b**) The *TopMost* graph of G; (**c**) The graph $G - u$ of 32 vertices and 56 edges; (**d**) The *TopMost* graph of $G - u$.

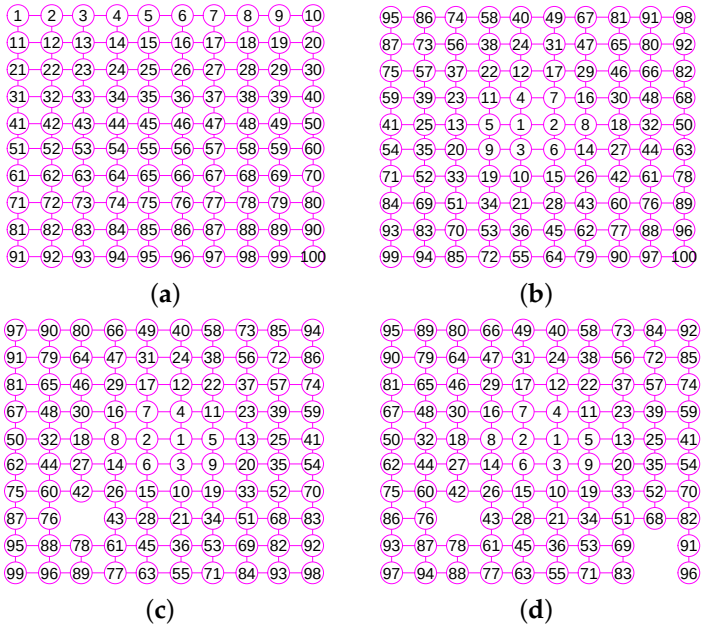

Figure 7. The *TopMost* graphs of three graphs, including the 10×10 grid graph $G_{10,10}$, $G_{10,10} - u$, $G_{10,10} - u - v - w$ with $u = 73$, $v = 88$, $w = 97 \in V(G_{10,10})$. (**a**) A 10×10 grid graph $G_{10,10}$ of 100 vertices and 180 edges; (**b**) The *TopMost* graph of $G_{10,10}$; (**c**) The *TopMost* graph of $G_{10,10} - u$; (**d**) The *TopMost* graph of $G_{10,10} - u - v - w$.

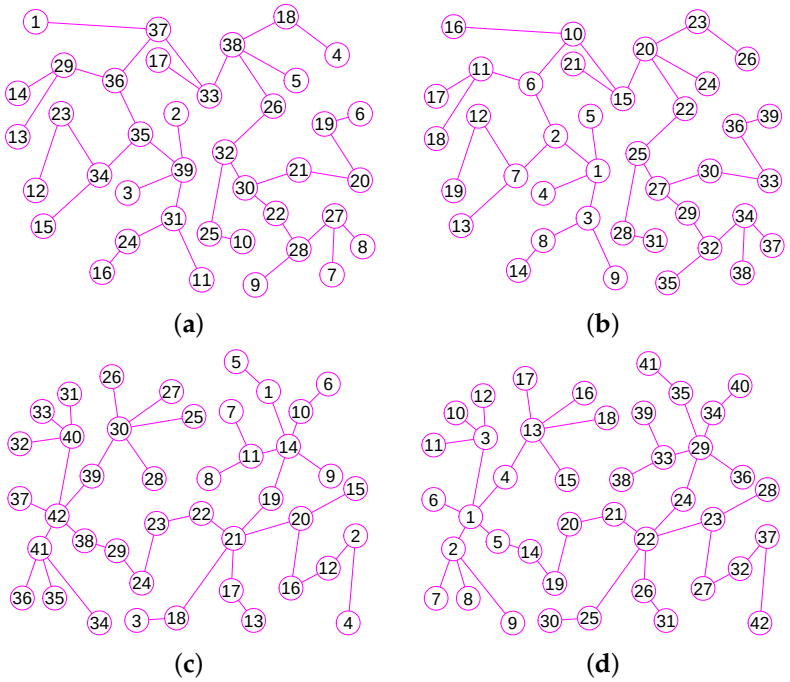

Figure 8. The *TopMost* graphs of two trees T_1 and T_2. (**a**) A tree T_1 of 39 vertices and 38 edges; (**b**) The *TopMost* graph of T_1; (**c**) A tree T_2 of 42 vertices and 41 edges; (**d**) The *TopMost* graph of T_2.

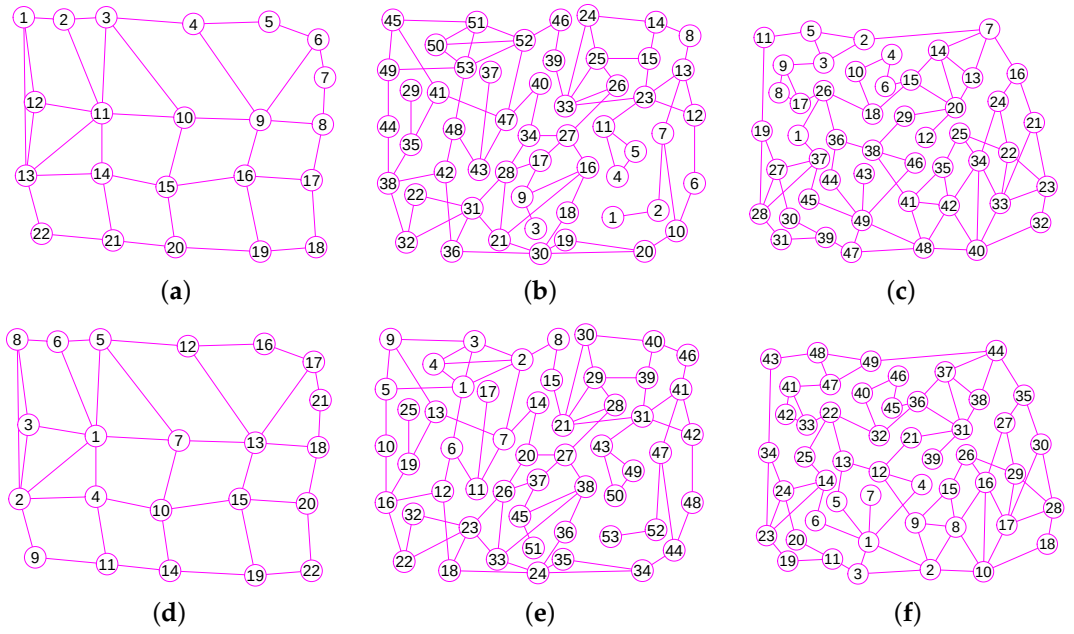

Figure 9. The *TopMost* graphs of three graphs G_1, G_2 and G_3. (**a**) A graph G_1 of 22 vertices and 37 edges; (**b**) A graph G_2 of 53 vertices and 80 edges; (**c**) A graph G_3 of 49 vertices and 78 edges; (**d**) The *TopMost* graph of G_1; (**e**) The *TopMost* graph of G_2; (**f**) The *TopMost* graph of G_3.

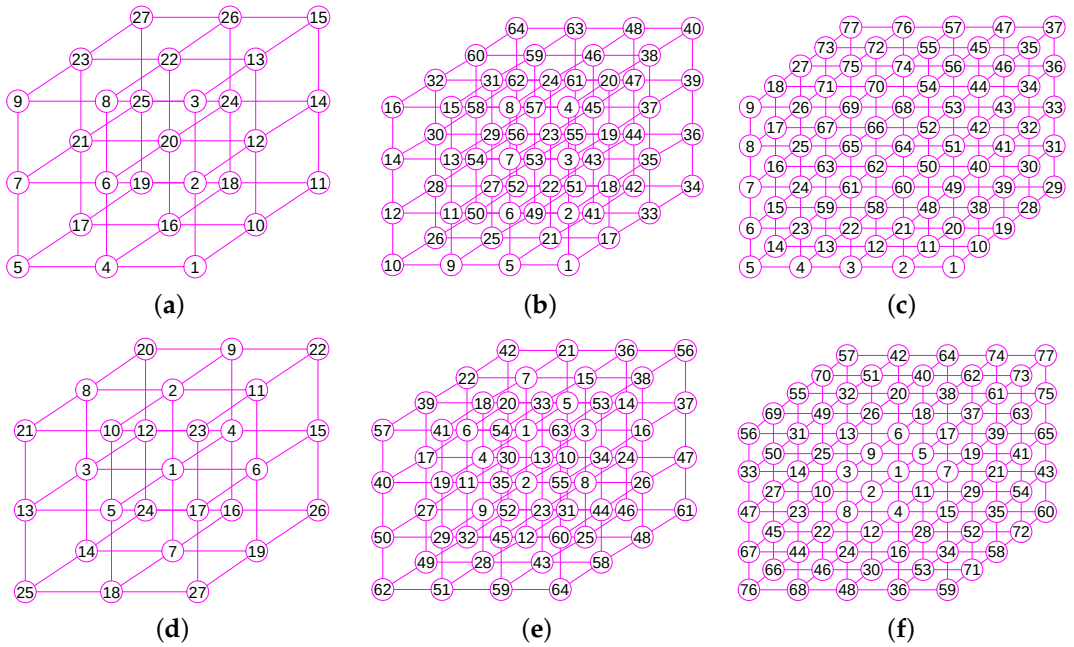

Figure 10. The *TopMost* graphs of three graphs $G_{3,3,3}$, $G_{4,4,4}$ and G_4. (**a**) The $3 \times 3 \times 3$ grid graph $G_{3,3,3}$ of 27 vertices and 54 edges; (**b**) The $4 \times 4 \times 4$ grid graph $G_{4,4,4}$ of 64 vertices and 144 edges; (**c**) A graph G_4 of 77 vertices and 196 edges; (**d**) The *TopMost* graph of $G_{3,3,3}$; (**e**) The *TopMost* graph of $G_{4,4,4}$; (**f**) The *TopMost* graph of G_4.

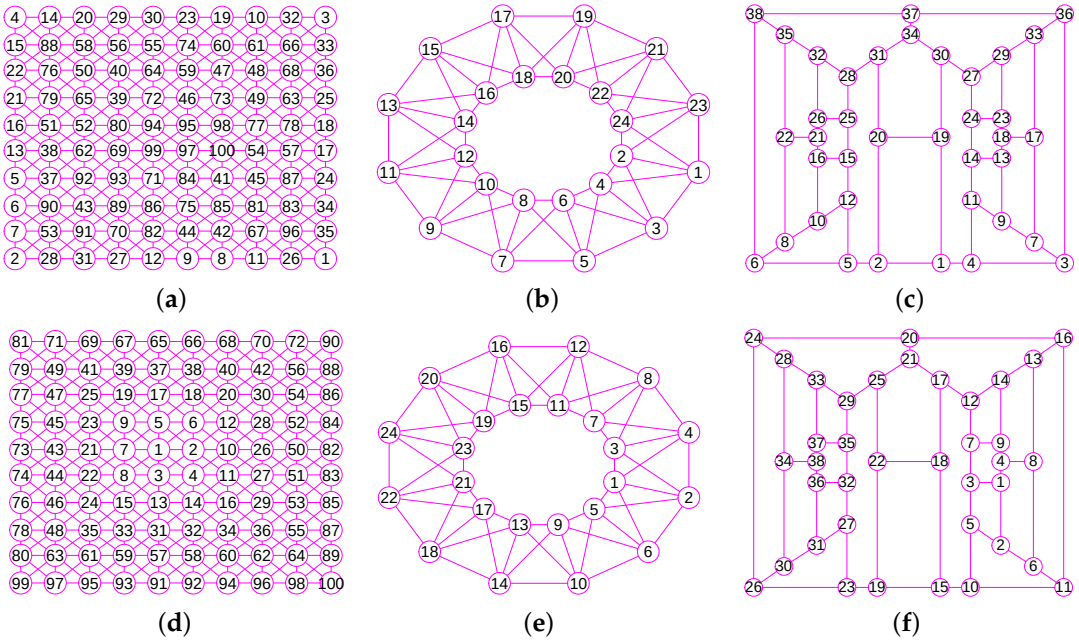

Figure 11. The *TopMost* graphs of three graphs G_5, G_6 and G_7. (**a**) The 10×10 king graph G_5 of 100 vertices and 342 edges; (**b**) The musical graph G_6 of 24 vertices and 60 edges; (**c**) The Barnette-Bosák-Lederberg graph G_7 of 38 vertices and 57 edges; (**d**) The *TopMost* graph of G_5; (**e**) The *TopMost* graph of G_6; (**f**) The *TopMost* graph of G_7.

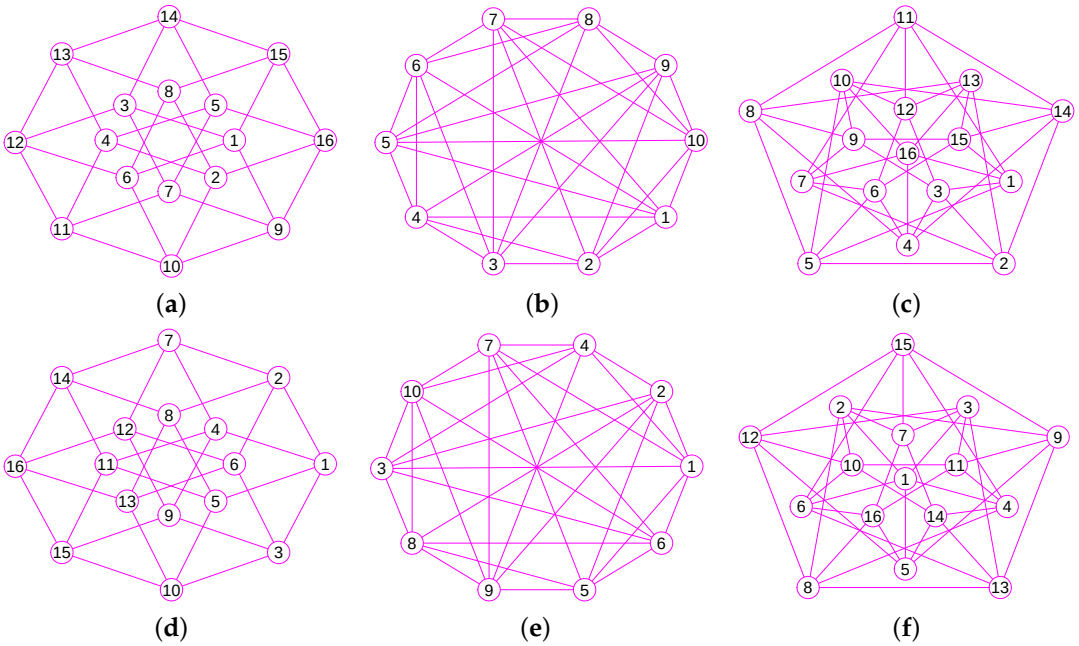

Figure 12. The *TopMost* graphs of three regular graphs G_8, G_9 and G_{10}. (**a**) The 4-hypercube graph G_8 of 16 vertices and 32 edges; (**b**) The triangular graph G_9 of 10 vertices and 30 edges; (**c**) The Clebsch graph G_{10} of 16 vertices and 40 edges; (**d**) The *TopMost* graph of G_8; (**e**) The *TopMost* graph of G_9; (**f**) The *TopMost* graph of G_{10}.

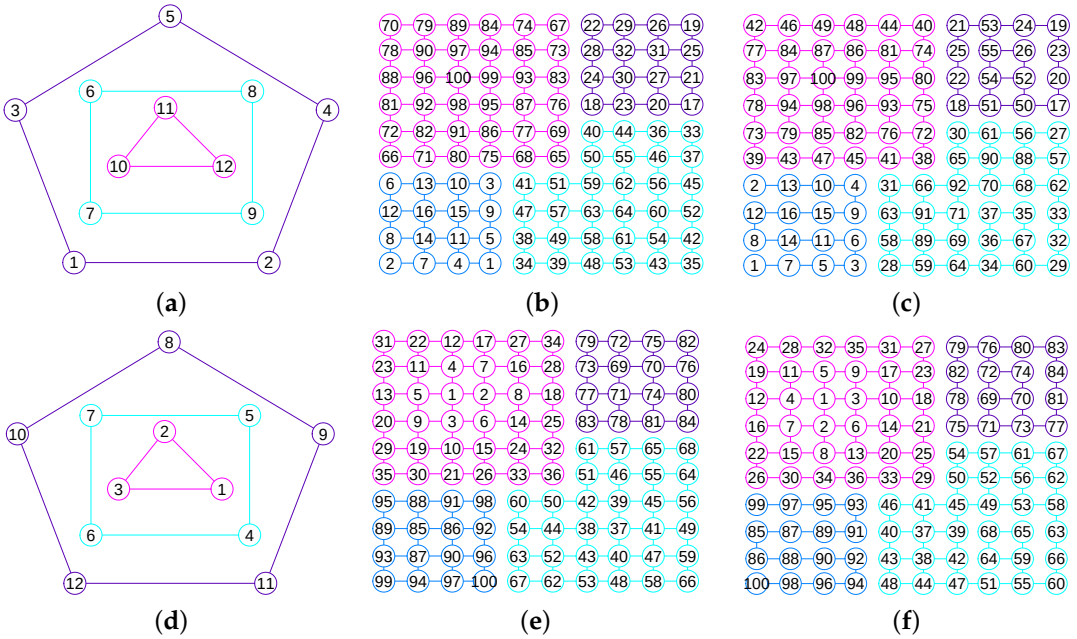

Figure 13. The *TopMost* graphs of three unconnected graphs G_{11}, G_{12} and G_{13}. (**a**) A disconnected graph G_{11} with 12 nodes and 12 edges; (**b**) A disconnected graph G_{12} that has four connected components and a total of 100 vertices and 160 edges; (**c**) A disconnected graph G_{13} that has four connected components and a total of 100 vertices and 131 edges; (**d**) The *TopMost* graph of G_{11}; (**e**) The *TopMost* graph of G_{12}; (**f**) The *TopMost* graph of G_{13}.

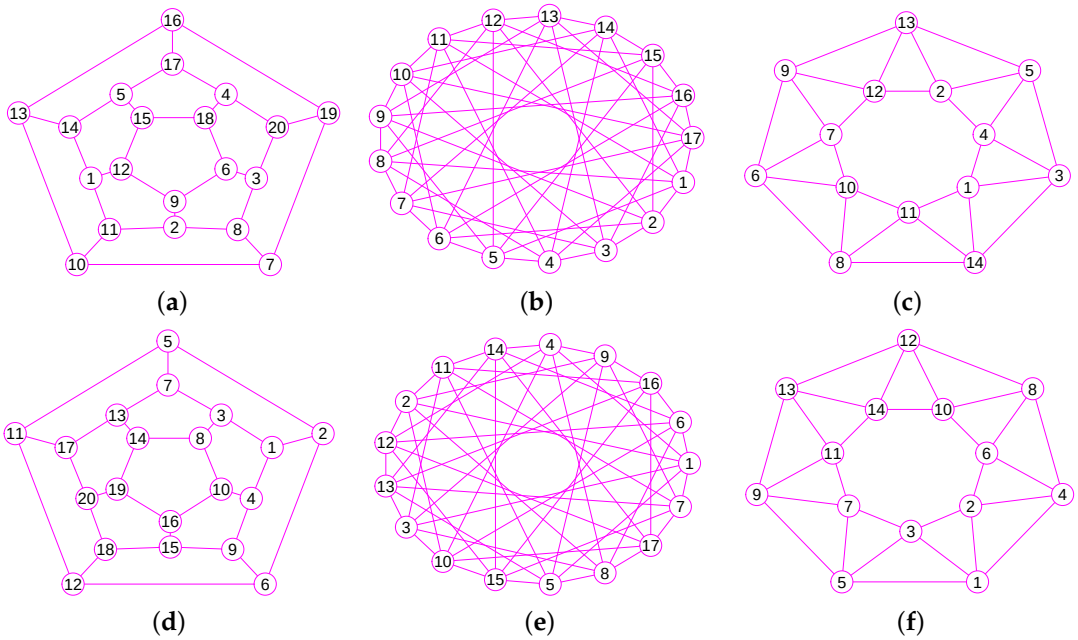

Figure 14. The *TopMost* graphs of three graphs G_{14}, G_{15} and G_{16}. (**a**) A Hamiltonian Graph G_{14} of 20 vertices and 30 edges; (**b**) The 6-Andrásfai graph G_{15} of 17 vertices and 51 edges; (**c**) The 7-antiprism graph G_{16} of 14 vertices and 28 edges; (**d**) The *TopMost* graph of G_{14}; (**e**) The *TopMost* graph of G_{15}; (**f**) The *TopMost* graph of G_{16}.

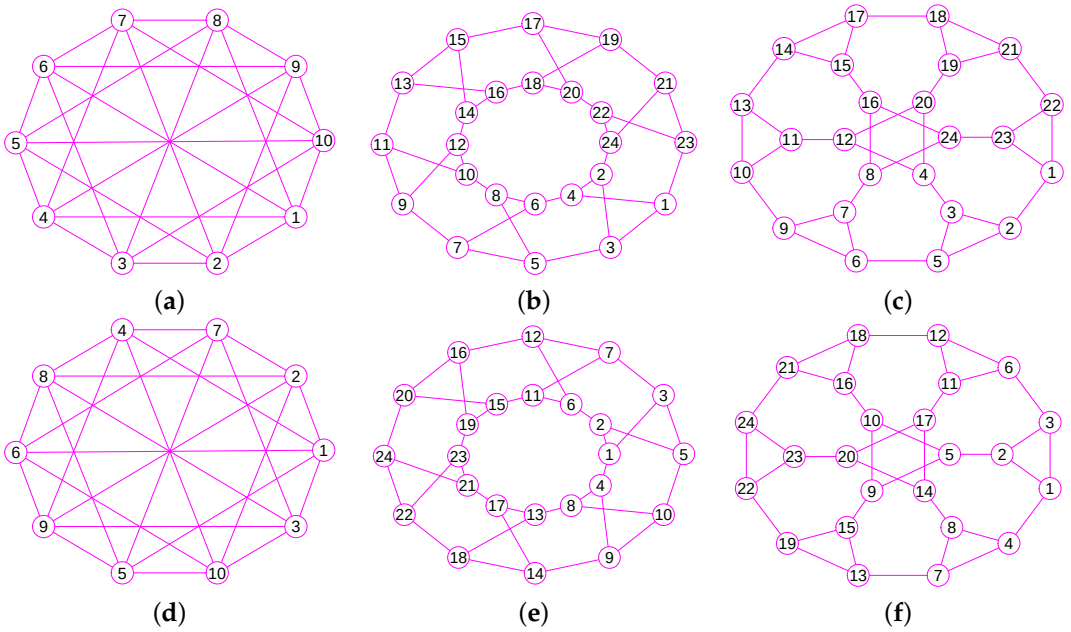

Figure 15. The *TopMost* graphs of three graphs $K_{(5,5)}$, G_{17} and G_{18}. (**a**) The complete bipartite graph $K_{(5,5)}$ of 10 vertices and 25 edges; (**b**) The 12-crossed prism graph G_{17} of 24 vertices and 36 edges; (**c**) The 6th order cube-connected cycle graph G_{18} of 24 nodes and 36 edges; (**d**) The *TopMost* graph of $K_{(5,5)}$; (**e**) The *TopMost* graph of G_{17}; (**f**) The *TopMost* graph of G_{18}.

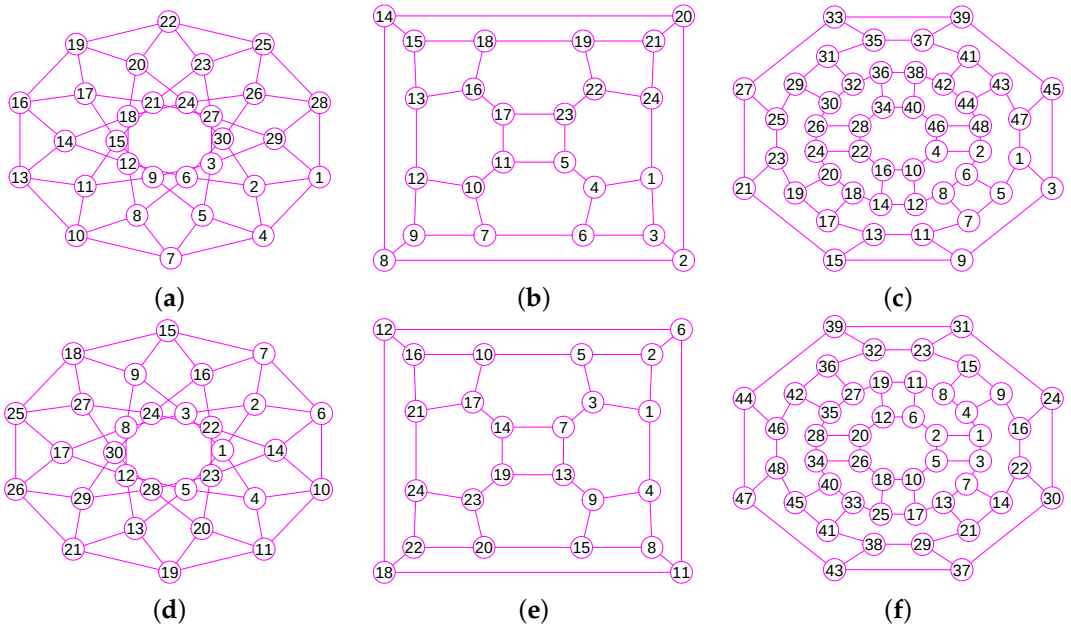

Figure 16. The *TopMost* graphs of three graphs G_{19}, G_{20} and G_{21}. (**a**) The Icosidodecahedral Graph G_{19} with 30 nodes and 60 edges; (**b**) The truncated octahedron graph G_{20} with 24 nodes and 36 edges; (**c**) The great rhombicuboctahedron graph G_{21} of 48 nodes and 72 edges; (**d**) The *TopMost* graph of G_{19}; (**e**) The *TopMost* graph of G_{20}; (**f**) The *TopMost* graph of G_{21}.

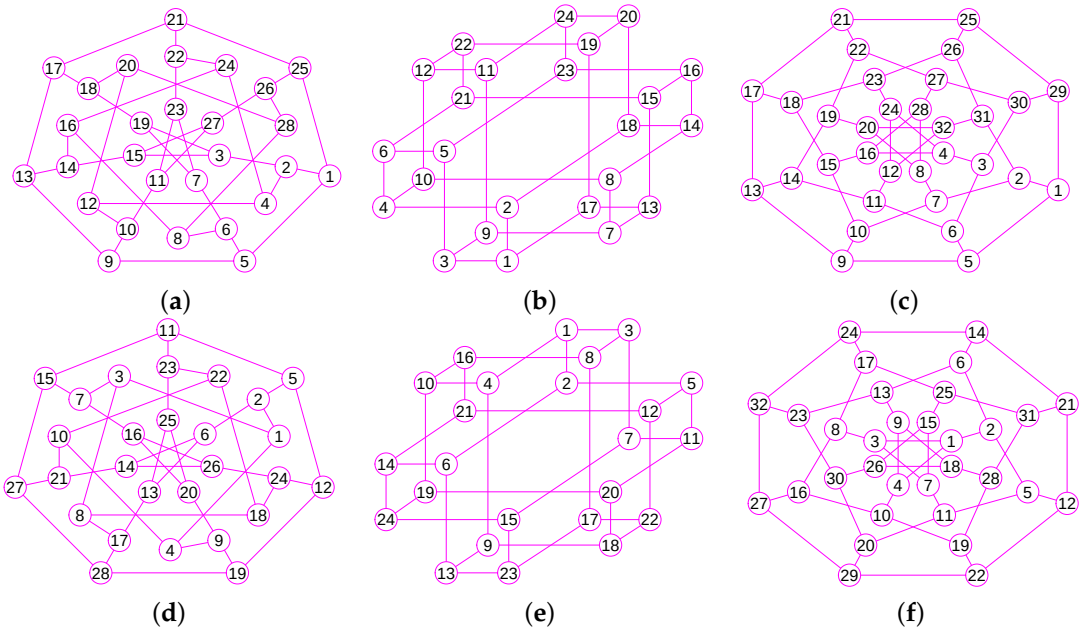

Figure 17. The *TopMost* graphs of three graphs G_{22}, G_{23} and G_{24}. (**a**) The coxeter graph G_{22} of 28 vertices and 42 edges; (**b**) The Nauru graph G_{23} with 24 nodes and 36 edges; (**c**) The Dyck graph G_{24} of 32 nodes and 48 edges; (**d**) The *TopMost* graph of G_{22}; (**e**) The *TopMost* graph of G_{23}; (**f**) The *TopMost* graph of G_{24}.

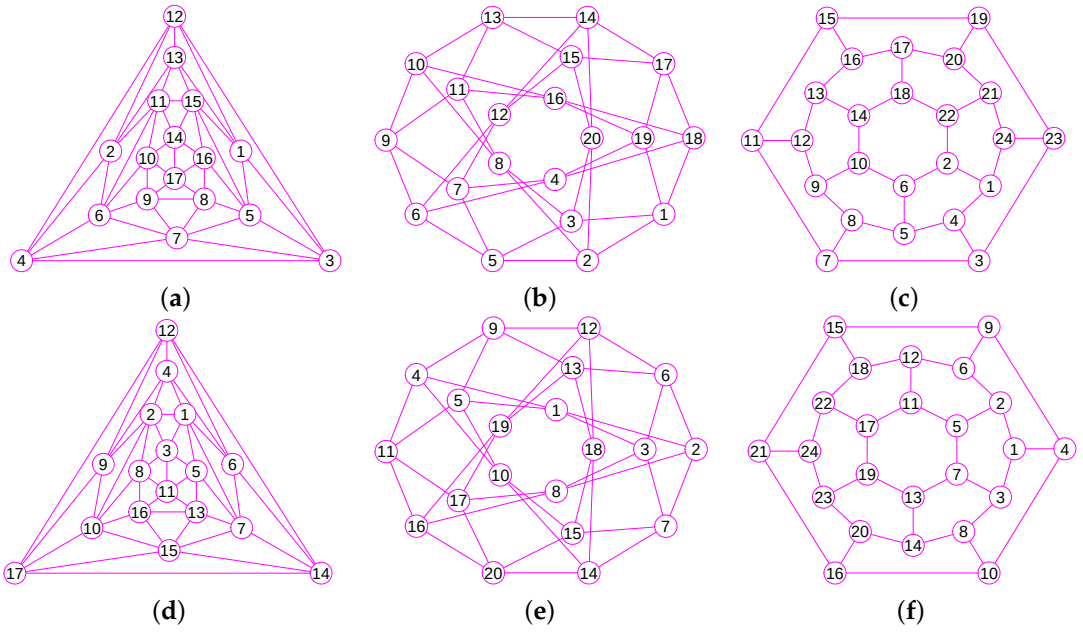

Figure 18. The *TopMost* graphs of three graphs G_{25}, G_{26} and G_{27}. (**a**) The Errera graph G_{25} of 17 vertices and 45 edges; (**b**) The Folkman graph G_{26} with 20 nodes and 40 edges; (**c**) A fullerene graph G_{27} of 24 nodes and 36 edges; (**d**) The *TopMost* graph of G_{25}; (**e**) The *TopMost* graph of G_{26}; (**f**) The *TopMost* graph of G_{27}.

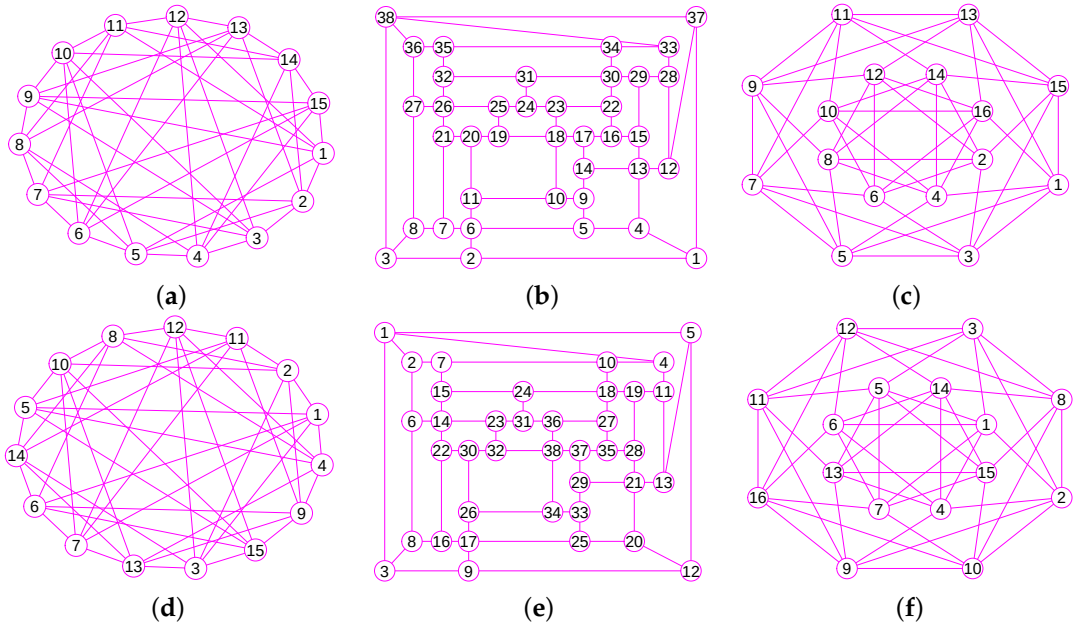

Figure 19. The *TopMost* graphs of three graphs G_{28}, G_{29} and G_{30}. (**a**) A generalized quadrangle graph G_{28} with 15 nodes and 45 edges; (**b**) A pentagonal icositetrahedral graph G_{29} with 38 nodes and 60 edges; (**c**) A Shrikhande graph G_{30} of 16 nodes and 48 edges; (**d**) The *TopMost* graph of G_{28}; (**e**) The *TopMost* graph of G_{29}; (**f**) The *TopMost* graph of G_{30}.

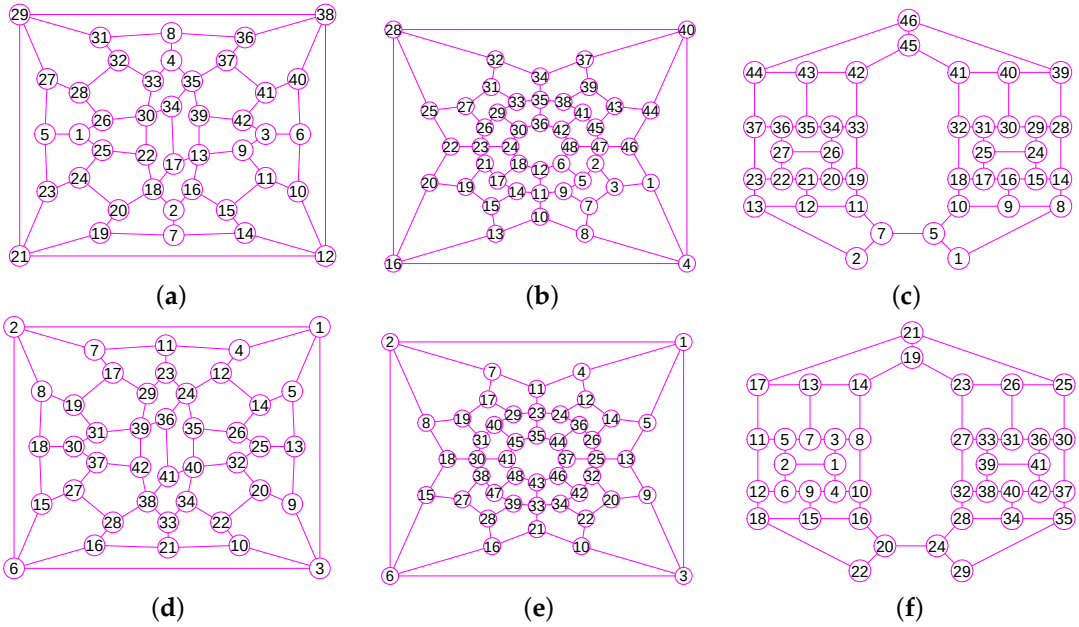

Figure 20. The *TopMost* graphs of three graphs G_{31}, G_{32} and G_{33}. (**a**) The Wiener-Araya graph G_{31} of 42 vertices and 67 edges; (**b**) The Zamfirescu graph G_{32} with 48 nodes and 76 edges; (**c**) The Faulkner-Younger graph G_{33} of 42 nodes and 62 edges; (**d**) The *TopMost* graph of G_{31}; (**e**) The *TopMost* graph of G_{32}; (**f**) The *TopMost* graph of G_{33}.

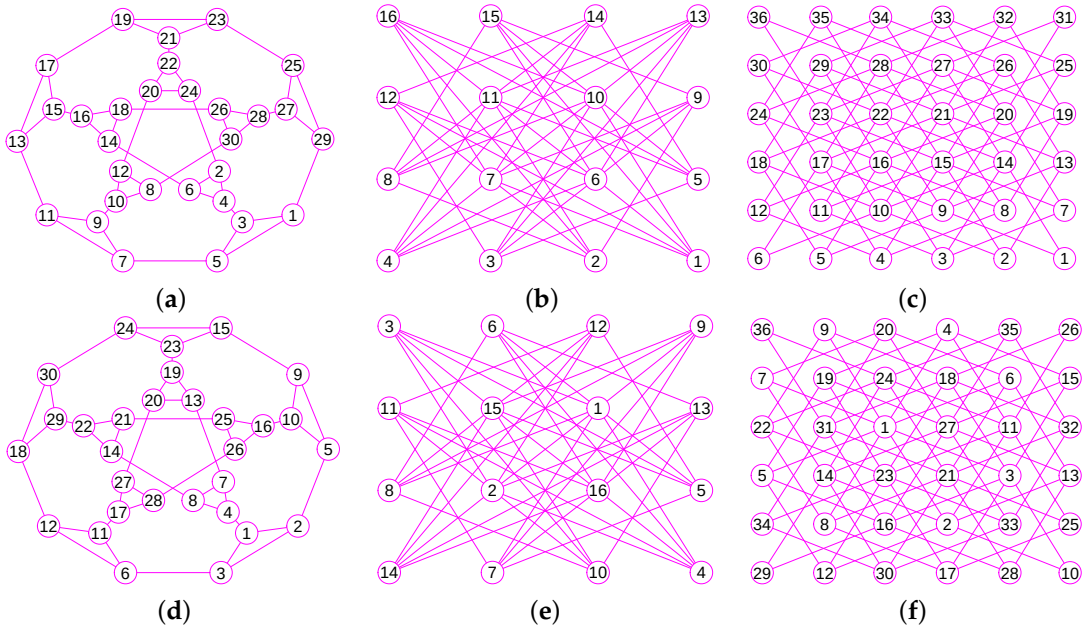

Figure 21. The *TopMost* graphs of three graphs G_{34}, G_{35} and G_{36}. (**a**) A triangle-replaced graph G_{34} with 30 nodes and 45 edges; (**b**) The 4-dimensional Keller graph G_{35} of 16 vertices and 46 edges; (**c**) The 6×6 knight graph G_{36} of 36 nodes and 80 edges; (**d**) The *TopMost* graph of G_{34}; (**e**) The *TopMost* graph of G_{35}; (**f**) The *TopMost* graph of G_{36}.

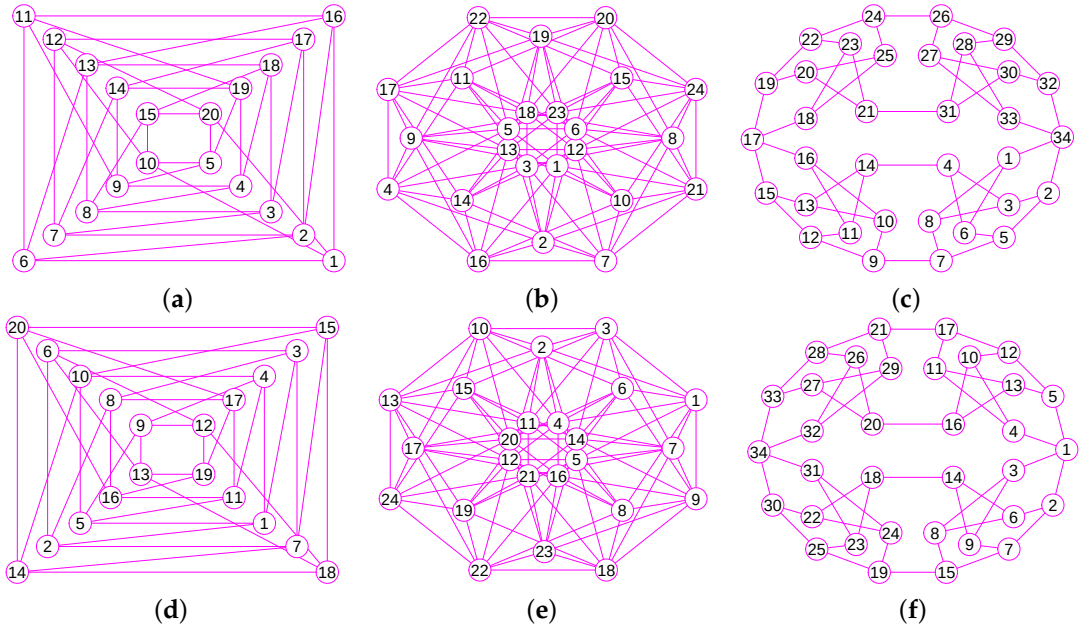

Figure 22. The *TopMost* graphs of three graphs G_{37}, G_{38} and G_{39}. (**a**) The Folkman graph G_{37} of 20 vertices and 40 edges; (**b**) The 24-cell graph G_{38} with 24 nodes and 96 edges; (**c**) The Thomassen graph G_{39} of 34 nodes and 52 edges; (**d**) The *TopMost* graph of G_{37}; (**e**) The *TopMost* graph of G_{38}; (**f**) The *TopMost* graph of G_{39}.

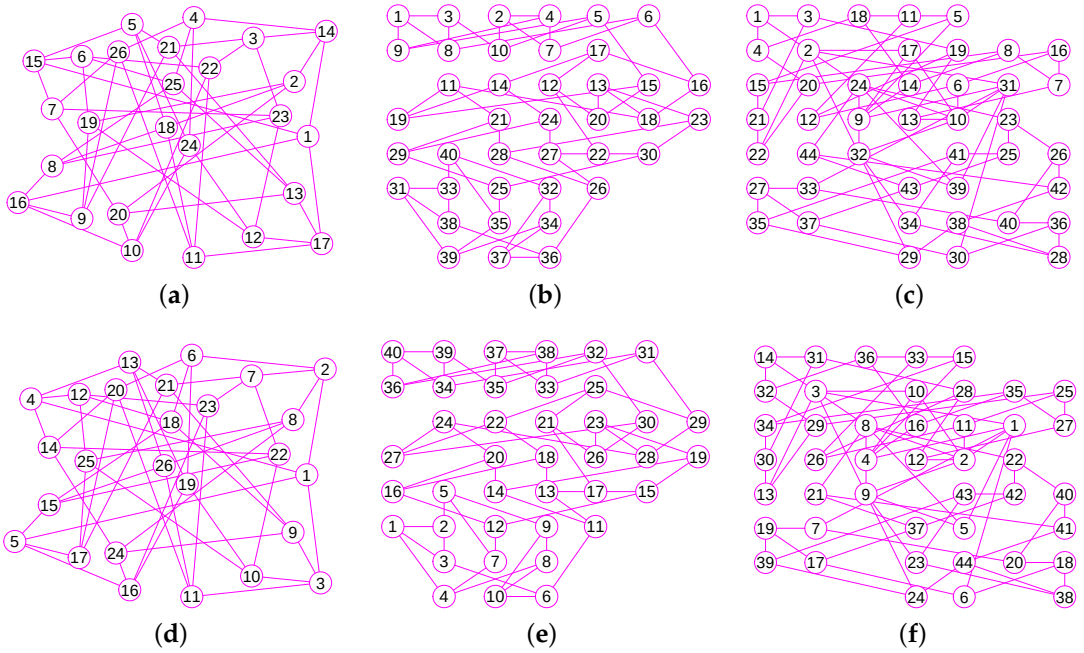

Figure 23. The *TopMost* graphs of three graphs G_{40}, G_{41} and G_{42}. (**a**) The projective plane graph G_{40} of 26 vertices and 52 edges; (**b**) The Miyazaki graph G_{41} with 40 nodes and 60 edges; (**c**) The Cubic Hypohamiltonian graph G_{42} of 44 vertices and 75 edges; (**d**) The *TopMost* graph of G_{40}; (**e**) The *TopMost* graph of G_{41}; (**f**) The *TopMost* graph of G_{42}.

5. Software Implementation

Utilizing the theory specified in the previous sections, we made a kit of software means called GraphLabel to calculate *canonical labelings* of graphs. Our experimental conditions included an Intel(R) Core(TM)2 Quad CPU Q6600 @2.40 GHz with 4.00 GB of RAM. The operating system was Microsoft Windows 8.1 Professional Edition. The graphics card was an NVIDIA GeForce 9800 GT. The display resolution was $1024 \times 768 \times 32$ bits (RGB). The internal hard drive was 500 GB. The programming environment was Microsoft Visual C++ 2012.

The software utilized object-oriented technique to construct many related classes, including *CNode*, *CNodeNeighbor*, *CEdge*, *CEdgeNeighbor*, *CGraph* and so on. A complete explanation of the software functions is beyond the range of this paper. We will fully describe it in the other articles. All figures displayed in this article were created by employing our software system.

We chose a graph collection to check the correctness of our algorithms. We used our own software program to generate a large number of graphs randomly as the test cases, including Figures 7c,d, 8, 9 and 10c. Besides, for enhancing the depth and breadth of experimentation, we also adopted many test cases from the online library [31] and library of benchmarks [32], including Figures 6, 7a, 10a,b and 11–23.

We applied our algorithms to as many classes of graphs as potential. These graphs presented here are only a small portion of the check graphs since the length of the paper is restricted. For comparing entirely, we offer both the initial and the resulting graph.

6. Summary and Future Work

In short, we get the following results: by Theorems 2–9, the paper has built a comparatively entire theoretical frame for computing the *canonical labelings* and *TopMost* graphs of graphs. Algorithms 1–4 are unique and can correctly determine *TopMost* graphs for many classes of graphs, including trees, grid graphs, wheel graphs, hypercube graphs, king graphs, triangular graphs and so on (see Figures 6–23). Algorithms 1–4 are also valid for detached undirected graphs. For every vertex

in a graph G, the definition of the property *m_NearestNode* enhances the accuracy of computing *canonical labeling* . By software evaluating, the accuracy of our algorithms is elementarily established. Our approach can be employed to excavate the frequent subgraph. Additionally, it proposes Conjecture 1.

Nevertheless, there are still many aspects we need to progress, including verifying the conjectures suggested by us, improving our software platform and employing more test cases to check our program. Specifically, we need to reinforce our algorithms so that they can determine the *canonical labelings* for more classes of graphs.

Currently, we are considering how to stretch our method to dig the frequent subgraphs and determine the *canonical labelings* of weighted graphs. We will present further research in other papers.

Author Contributions: Conceptualization, J.H., Y.G., J.S. and L.T.; Methodology, J.H.; Software, J.H.; Validation, J.H., Y.G., J.S. and L.T.; Writing—original draft, J.H.; and Writing—review and editing, J.H.

Funding: The work described in this paper was supported by the National Natural Science Foundation of China (grant numbers 61702020); Beijing Natural Science Foundation (grant numbers 4172013); and Beijing Natural Science Foundation-Haidian Primitive Innovation Joint Fund (grant numbers L182007).

Acknowledgments: We would also like to thank all anonymous reviewers for their inspiring and constructive comments which helped to improve the presentation of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hao, J.; Gong, Y.; Wang, Y.; Tan, L.; Sun, J. Using k-Mix-Neighborhood Subdigraphs to Compute Canonical Labelings of Digraphs. *Entropy* **2017**, *19*, 79. [CrossRef]
2. Hao, J.; Gong, Y.; Tan, L.; Duan, D. Apply Partition Tree to Compute Canonical Labelings of Graphs. *Int. J. Grid Distrib. Comput.* **2016**, *9*, 241–264.
3. McKay, B. *Computing Automorphisms and Canonical Labellings of Graphs Combinatorial Mathematics*; Lecture Notes in Mathematics; Springer: Berlin/Heidelberg, Germany, 1978; Volume 686, pp. 223–232.
4. Piperno, A. Search space contraction in canonical labeling of graphs. *arxiv* **2008**, arXiv:0804.4881.
5. Junttila, T.; Kaski, P. Engineering an Efficient Canonical Labeling Tool for Large and Sparse Graphs. In Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments and the Fourth Workshop on Analytic Algorithmics and Combinatorics, New Orleans, LA, USA, 6 January 2007; Siam: Philadelphia, PA, USA, 2007; pp. 135–149.
6. Shah, Y.J.; Davida, G.I.; McCarthy, M.K. Optimum Featurs and Graph Isomorphism. *IEEE Trans. Syst. Man Cybern.* **1974**, *SMC-4*, 313–319. [CrossRef]
7. Ivanciuc, O. Canonical Numbering and Constitutional Symmetry. In *Handbook of Chemoinformatics*; Wiley-VCH Verlag GmbH: Weinheim, Germany, 2008; pp. 139–160. [CrossRef]
8. Babai, L.; Luks, E.M. Canonical Labeling of Graphs. In Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing, Boston, MA, USA, 25–27 April 1983; ACM: New York, NY, USA, 1983; pp. 171–183. [CrossRef]
9. Jantschi, L.; Bolboaca, S.D. Conformational study of C-24 cyclic polyyne clusters. *Int. J. Quantum Chem.* **2018**, *118*, e25614. [CrossRef]
10. Joiţa, D.M.; Jäntschi, L. Extending the Characteristic Polynomial for Characterization of C20 Fullerene Congeners. *Mathematics* **2017**, *5*, 84. [CrossRef]
11. Bolboaca, S.; Jantschi, L. How good can the characteristic polynomial be for correlations? *Int. J. Mol. Sci.* **2007**, *8*, 335–345. [CrossRef]
12. Kuramochi, M.; Karypis, G. Finding Frequent Patterns in a Large Sparse Graph*. *Data Min. Knowl. Discov.* **2005**, *11*, 243–271. [CrossRef]
13. Kuramochi, M.; Karypis, G. An efficient algorithm for discovering frequent subgraphs. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1038–1051. [CrossRef]
14. Huan, J.; Wang, W.; Prins, J. Efficient mining of frequent subgraphs in the presence of isomorphism. In Proceedings of the Third IEEE International Conference on Data Mining, Melbourne, FL, USA, 22 November 2003; pp. 549–552.

15. Kashani, Z.; Ahrabian, H.; Elahi, E.; Nowzari-Dalini, A.; Ansari, E.; Asadi, S.; Mohammadi, S.; Schreiber, F.; Masoudi-Nejad, A. Kavosh: A new algorithm for finding network motifs. *BMC Bioinform.* **2009**, *10*, 318. [CrossRef] [PubMed]
16. He, P.R.; Zhang, W.J.; Li, Q. Some further development on the eigensystem approach for graph isomorphism detection. *J. Frankl. Inst.-Eng. Appl. Math.* **2005**, *342*, 657–673. [CrossRef]
17. Arvind, V.; Das, B.; Köbler, J. A Logspace Algorithm for Partial 2-Tree Canonization. In *Computer Science-Theory and Applications*; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5010, pp. 40–51. [CrossRef]
18. Babai, L.; Kucera, L. Canonical labelling of graphs in linear average time. In Proceedings of the 20th Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 29–31 October 1979; pp. 39–46. [CrossRef]
19. Arnborg, S.; Proskurowski, A. Canonical Representations of Partial 2- and 3-Trees. In Proceedings of the 2nd Scandinavian Workshop on Algorithm Theory, Bergen, Norway, 11–14 July 1990; Lecture Notes in Computer Science 477; Springer: Berlin, Germany, 1990; pp. 197–214.
20. McKay, B. *Practical Graph Isomorphism*; Department of Computer Science, Vanderbilt University: Nashville, TN, USA, 1981.
21. McKay, B.D. Isomorph-Free Exhaustive Generation. *J. Algorithms* **1998**, *26*, 306–324. [CrossRef]
22. Practical graph isomorphism, {II}. *J. Symb. Comput.* **2014**, *60*, 94–112. [CrossRef]
23. Yan, X.; Han, J. gSpan: Graph-based substructure pattern mining. In Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2003, Maebashi City, Japan, 9–12 December 2002; pp. 721–724. [CrossRef]
24. Miyazaki, T. *The Complexity of McKay's Canonical Labeling Algorithm*; Citeseer: University Park, PA, USA, 1997.
25. Tener, G.; Deo, N. Efficient isomorphism of miyazaki graphs. *Algorithms* **2008**, *5*, 7.
26. Junttila, T.; Kaski, P. *Conflict Propagation and Component Recursion for Canonical Labeling Theory and Practice of Algorithms in (Computer) Systems*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6595, pp. 151–162.
27. López-Presa, J.L.; Anta, A.F.; Chiroque, L.N. Conauto-2.0: Fast Isomorphism Testing and Automorphism Group Computation. *arXiv* **2011**, arXiv:1108.1060.
28. Katebi, H.; Sakallah, K.; Markov, I. Graph Symmetry Detection and Canonical Labeling: Differences and Synergies. *arXiv* **2012**, arXiv:1208.6271.
29. Habtemicael, S.; SenGupta, I. Pricing variance and volatility swaps for Barndorff-Nielsen and Shephard process driven financial markets. *Int. J. Financ. Eng.* **2016**, *3*, 1650027. [CrossRef]
30. Mariani, M.C.; SenGupta, I.; Bezdek, P. Numerical solutions for option pricing models including transaction costs and stochastic volatility. *Acta Appl. Math.* **2012**, *118*, 203–220. [CrossRef]
31. Weisstein, E.W. *Simple Graphs–from Wolfram MathWorld*; Wolfram Research: Champaign, IL, USA, 2015.
32. ALENEX 2007 Submission: Source Code, Benchmark Instances, and Summary Results. Available online: http://www.tcs.hut.fi/Software/benchmarks/ALENEX-2007/ (accessed on 30 June 2018).