Chapter 6

Optimizing Various Invariants of Bird Graph

Overview

In Chapter 5, a new class of perfect graph is introduced. In the said chapter, the specific graph class is defined formally and then characterized following its structural aspects. Based on different characterizations, corresponding recognition algorithm is derived that is proved to successfully recognize a graph instance whether it is bird graph (or not) in polynomial time. The related chapter terminates with a proof that the graph class belongs to the field of perfect graph. In this respect, the operation complete multipartite cutset is utilized [63]. In this chapter, algorithms are developed to compute some of the invariants in polynomial time. All the algorithms are followed by the required theorems regarding the time and space consumed by respective algorithms and whether the algorithms perform the computation successfully.

6.1 Introduction

In Chapter 5, a new class of perfect graph, named bird graph, is introduced formally followed by all the essential theorems and lemma (In Characterization, regarding hereditary property). However, the prime characterization of bird graph is the existence of at least one highest prioritized chest vertex, deletion of which along with its incident edges produces another bird graph. The operation as stated here if applied in succession ultimately results a forest at the end. Utilizing this characterization, algorithms to compute the invariants (and corresponding theorems and lemmas) related to definition of perfect graph, i.e., clique number, chromatic number, independence number, and clique cover number are included in this chapter.

6.2 Computation of Clique Number of a Bird Graph

According to the recognition algorithm of bird graph, a highest prioritized chest vertex is deleted in each of the iterations. After successive deletion of chest vertices with subsequent priorities (in the residual induced subgraphs), in due course the graph turns into a forest.
Now, while computing the clique number (or the size of a maximum clique of a given bird graph), we merely do the reverse process. That means we start from the forest we finally achieve. Here, for each edge in the forest, the clique size is two. We also consider the sequence of deleting (most prioritized) chest vertices in reverse order. As we include the last chest vertex we deleted, 3-cycles are introduced, and we identify the clique(s). Then, the next chest vertex is added into it. Thus, after each successive addition of chest vertices in the said order (as stated earlier), we compute maximal cliques involving each of the (newly added) chest vertices (along with their associated edges). As the graph gradually turns into a given complete bird graph (after inclusion of the most prioritized chest vertex that had been deleted first), the largest clique coupled with any of the chest vertices is the requisite clique, and thus, the (maximal) remaining cliques are discarded.

Below, in the next section, we have provided the algorithm *Computing_Clique_Number* to compute the clique number of a given bird graph to view it at a glance.

**6.2.1 Algorithm for Computing Clique Number in Stepwise Form**

*Computing_Clique_Number*

**Input:** A bird graph.

**Output:** The cardinality of a clique of maximum size.

1. Call the algorithm for recognizing bird graph and delete all the vertices marked as chest vertices in the order followed by the said algorithm.

2. As the graph turns into a forest, compute all maximal cliques comprising each edge in separation; this forest does not contain any chest vertex further.

3. Now starting from the partially constructed graph, around each maximal clique computed so far, a possible maximal clique of highest cardinality (consisting of a maximum number of vertices) is traced out after introducing each of the chest vertices (and its induced edges) in the reverse order of deletion following the bird graph recognition algorithm.

4. Repeat Step 3 until all the chest vertices (along with their induced edges) are added in reverse order of their deletion (following the bird graph recognition algorithm).
5. Output the size (i.e., the number of vertices) of the clique with the maximum cardinality.

The following section is equipped with essential theorem and lemma, proving that the algorithm for computation of the clique number of a bird graph successfully produces desired result in polynomial time.

### 6.2.2 Theorems Related to Computing Clique Number of a Bird Graph

Below, in Lemma 6.1, we have proved that the clique number computation is bounded by some polynomial function of number of vertices of the given graph in time and space. Whereas, Theorem 6.1 assures that the algorithm developed in this section correctly produces the desired result.

**Lemma 6.1:** The time and space complexities required for computing the clique number of a given bird graph $G = (V, E)$ are $O(n^4)$ and $O(n + e)$, respectively, where $|V| = n$ and $|E| = e$.

**Proof:** Here in the worst case $O(n)$ number of chest vertices are deleted by the application of bird graph recognition algorithm. At this step, after deletion of all chest vertices, the number of maximal cliques belonging to the resultant forest is bounded by the number of edges in it.

Now, according to the order opposite of their deletion, each of the chest vertices is added to the partially constructed (bird) graph (starting from the forest). Thereafter, maximal cliques are computed around the newly introduced chest vertex. As each of the chest vertices (along with its induced edges) is added to the partially constructed bird graph (towards achieving the given bird graph, $G$), $O(n)$ comparisons are required for adjacency checking in the worst case with a partially constructed clique (that may be an edge of the graph being turned into a forest) and $O(n)$ comparisons for the remaining cliques constructed in the graph (to check whether the newly added chest vertex is adjacent to a vertex being member of the partially constructed bird graph or to one or more maximal cliques computed up to now. Therefore, in the worst case $O(n^3)$ time is sufficient to compute the clique number.

As a result, following the logic of this algorithm, the computation of clique number of a bird graph essentially needs the help of bird graph recognition algorithm. Thus, the overall complexity of computing clique number of a bird graph is subjugated by the complexity of recognizing a bird graph, which is $O(n^4)$ for such a
graph $G = (V, E)$, where $|V| = n$. Moreover, as the linked linear lists representation of $G$ takes $O(n+e)$ space and the queue implementation of BFS also takes $O(n+e)$ space, where $|E| = e$, hence the space complexity of the clique number computation algorithm acquires $O(n+e)$ space. Therefore, the computational complexities of tracing out the clique number of a bird graph depend on the complexities necessary for recognizing a bird graph in terms of time and space both.

**Theorem 6.1:** The algorithm for calculating the clique number correctly computes the clique of maximum size.

**Proof:** The clique computation algorithm of a bird graph commences execution from the forest that the bird graph recognition algorithm produced after successive deletion of chest vertices (along with all adjacent edges). Thus, every edge of the forest results a maximal clique, say $\text{cliq}$, of size two each. Let the chest vertices being deleted from $G$ in order to attain to a forest be represented by a sequence $S$. Now, for all $v \in S$, where $S \subset V$, for a bird graph $G = (V, E)$, $v$’s are added in the reverse order of their deletion in $S$ by the recognition algorithm into the partially constructed graph (starting from the resulting forest of $G$).

As a $CV_x$, for some $x$ that varies from 1 through 4, is taken in the partially constructed graph that grows from the forest, cardinality of some clique, $\text{cliq}_a$ may be incremented if there exist $\{CV_x, v\} \in E$, all $v \in \text{cliq}_a$. This scenario may happen for more than one $\text{cliq}$ being members of the set of all cliques constructed so far. On the contrary, $\{CV_x, v\} \in E$, but $\{CV_x, v'\} \notin E$, whereas, $\{v, v'\} \in E$, and also $\{v, v'\} \in \text{cliq}_a$. Hence, the algorithm to trace out the clique number produces a new clique, $\text{cliq}_b$, where all $v_b \in \text{cliq}_b$, $\{CV_x, v_b\} \in E$.

In this way, as all the $CV_x$’s (as they are added in succession) are compared for inclusion within the clique(s) computed so far, that eventually becomes an even larger clique after inclusion of each $CV_x$, if one exists, which is computed. Therefore, we conclude that the algorithm for computing the clique size of a given bird graph $G$ correctly computes the clique number of $G$. ♦

Below, the bird graph that we have utilized to elucidate the execution of algorithm for recognizing the graph class is again used to illustrate the claim of Theorem 6.1.
Following the algorithm of computing the clique and the clique number of a graph, first of all the bird graph depicted in Figure 5.5 is turned into a forest (following the bird graph recognition algorithm) as shown in Figure 6.1(a).

Now, the least prioritized chest vertex (or the last chest vertex) $d$ (that was deleted during the recognition algorithm) is now added first to the graph (or the forest) to yield a graph as depicted in Figure 6.1(b). Before addition of $d$ to the forest, the clique number of the graph in Figure 6.1(a) is two. Now, after vertex $d$ being added (along with its adjacent edges) as obtain in Figure 6.1(b), each of the maximum cliques computed (i.e., $\langle d, e, o \rangle$, $\langle d, m, n \rangle$ and $\langle d, n, p \rangle$) is of size three. The next chest vertex included in the partially constructed bird graph is $k$; hence, the maximum cliques involving vertex $k$ are $\langle k, c, i \rangle$, $\langle k, f, d \rangle$, and $\langle k, c, f \rangle$, though the clique number is still three (see Figure 6.1(c)). The next chest vertex added to the graph in Figure 6.1(c) is vertex $b$. Hence, the maximum cliques acquired (or added to the graph) are $\langle b, g, h \rangle$ and $\langle b, h, j \rangle$, keeping the clique number same as three as the fact is depicted in Figure 6.1(d).

Now the next vertex going to be added in the partially constructed graph is vertex $r$, generating two maximal cliques $\langle r, g, b \rangle$ and $\langle r, e, o \rangle$, thereby constraining the size of the maximum clique to be of three following Figure 6.1(e). Lastly, the most prioritized chest vertex $a$ being added to the graph in Figure 6.1(e) yielding the original bird graph in Figure 6.1(f).

Here, due to addition of vertex $a$, two maximal cliques are produced that are $\langle a, b, c \rangle$ and $\langle a, e, d \rangle$. So, ultimately it may be concluded that the clique number of the graph of Figure 5.5(a) is three. For ease of reader, each chest vertex being successively added in the reverse order is distinguished by black hole from non-chest vertices, which are depicted by clear holes.

However, it is possible to be the cardinality of clique number of a bird graph more than three. Such a case occurs if a non-basic structure (say, a clique of size four or more) is allowed to blend the basic structures. Although, basic structures must share vertices or edges of the said non-basic structure in such a way that successive deletion of chest vertices of $BS$, following the imposed order of priority, belonging to the bird graph under concern yields a forest.
Figure 6.1: Computation of maximum clique of a bird graph depicted in Figure 5.5. 
(a) The forest that was generated after successive deletion of chest vertices \{a, r, b, k, d\}, following the application of bird graph recognition algorithm. (b) The graph after inclusion of the last deleted vertex \(d\) (along with its edges) yielding three maximal cliques of size three each involving vertex \(d\). (c) In the second step, the graph we obtain after addition of the last but one deleted vertex, i.e., \(k\) (along with its edges); the newly produced maximal cliques still do not exceed the maximum size three. (d) The next (chest) vertex inserted is \(b\) (along with its edges); hence, the graph contains each of the maximum cliques of size three only. (e) Similarly, the graph we obtain after inclusion of the second highest prioritized chest vertex \(r\) (along with its edges) without producing a larger clique than size three. (f) The given bird graph of Figure 5.5(a) is obtained after addition of the highest prioritized chest vertex (i.e., \(a\)) in reverse order that also contains several maximum cliques of size three each.
In a similar way, the algorithms for computing the chromatic number, the independence number, and the clique cover number, with associated lemmas and theorems provided in Section 6.3, Section 6.4, and Section 6.5, respectively.

6.3 Computation of Chromatic Number of a Bird Graph

Following the characterization of bird graph, there is at least one most prioritized chest vertex. Deletion of the chest vertex produces another bird graph having at least one chest vertex with highest priority. In this way, successive deletion of chest vertices ultimately produces a forest.

Now while computing the chromatic number (similar to the process of computing the maximum clique), we do start from the forest. In the first step, we assign colours to the vertices of the forest. It is quite clear that at most two colours are needed to assign to the vertices of the forest.

From then onwards, within the forest the lowest prioritized (or one of the lowest prioritized) chest vertex (vertices) is added in succession in an order opposite to the order of deletion of chest vertices. The newly added chest vertex is assigned a colour that is distinct to the colours assigned to the vertices of its neighbourhood (within the graph formed so far). In this way, after all vertices are added to the graph, all the vertices receive a colour that is different from its neighbour vertices.

Now, successive addition and assignment of colours to the vertices of the graph actually produces the minimum colouring to the vertices of the graph under construction. Therefore, the number of colours assigned in this way to the vertices of a bird graph actually provides us the chromatic number.

Below, we have provided the algorithm Computing_Chromatic_Number as detailed in text above in stepwise form, which follows the essential lemmas related to time and space complexities and a proof that the algorithm in no doubt figure out the accurate chromatic number from a given bird graph.

6.3.1 A Snapshot View of the Algorithm Computing_Chromatic_Number

<table>
<thead>
<tr>
<th>Computing_Chromatic_Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A bird graph.</td>
</tr>
<tr>
<td><strong>Output:</strong> Chromatic number of the given graph.</td>
</tr>
</tbody>
</table>
**Step 1:** Call the algorithm for recognizing bird graph and then delete chest vertices in succession following its logic.

**Step 2:** Assign a colour to each of the vertices of the forest.

**Step 3:** Add chest vertices that were deleted in Step 1, one at a time successively and assign colour to the newly added vertices depending on the already assigned colours of its neighbourhood. Keep a count of number of distinguished colours used so far.

**Step 4:** Repeat Step 3 until all the chest vertices are added in non-descending order of priority.

**Step 5:** Terminate the algorithm until all the vertices (chest) have been added and assigned colours.

**Step 6:** Output the count of colours used as the chromatic number of the graph given as input.

In the following section, we have proved (by a lemma) that the algorithm developed here is bounded by polynomial time. Within the same section we have offered a theorem proving that whenever a bird graph is given as input, the algorithm to compute the chromatic number successfully executes and produces correct result.

### 6.3.2 Theorems Related to Computation of Chromatic Number

**Lemma 6.2:** The time and space complexities required for computing the chromatic number of a bird graph $G = (V, E)$ is $O(n^4)$ and $O(n + e)$, respectively, where $|V| = n$ and $|E| = e$.

**Proof:** Here in worst case, $O(n)$ number of chest vertices has been deleted in the process of recognizing a bird graph. Now, at first we assign colours to the vertices of the forest. This step requires scanning at most $O(n)$ vertices (present as member vertices of the forest). Now assigning a colour to an existing vertex (within the forest) requires at most $O(n)$ comparisons (or checking) to the neighbour vertices of the vertex (i.e., the degree of the vertex). According to the algorithm for calculating the chromatic number, as each of the chest vertices is added within the partially constructed graph, again $O(n)$ comparisons are required for adjacency checking in worst case with a partially constructed bird graph. Now this process may continue for $O(n)$ comparisons for remaining chest vertices (in worst case) that are yet to be added.
and compared with $O(n)$ adjacent vertices of the graph. Therefore, in worst case $O(n^3)$ time is sufficient to compute the chromatic number.

However, as stated in Lemma 6.1, the algorithm for deducing the chromatic number of a bird graph depends on the recognition algorithm of bird graph; as the algorithm described in this section initiates execution after a bird graph is transformed in to a forest. Therefore, ultimately the time and space complexities of computing chromatic number of a bird graph is bounded by $O(n^4)$ and $O(n + e)$ for a graph $G = (V, E)$, where $|V| = n$ and $|E| = e$. ♦

**Theorem 6.2:** The chromatic number computation algorithm correctly computes the chromatic number of a bird graph.

**Proof:** As stated in the algorithm, it depends on the recognition algorithm of bird graph. Following the characterization of the graph, after deletion of chest vertices based on priorities assigned to the vertices, a forest is produced. Here colours are encoded by natural number (greater than zero) or by the first letter of the English alphabet representing the colour. Now, the chromatic number of a forest cannot exceed two. This is because the same colour is assigned to the alternating vertices of the forest. As the chest vertices are added following the priorities (in reverse order of which they get deleted in the process of recognizing the graph), the minimum number (or a distinct colour) is assigned to the vertex, which is not assigned to any of its neighbour vertices.

Now, in this way all the vertices are coupled in succession and assigned the minimum number denoting a colour based on the colours already assigned to its adjacent vertices included in the partially constructed graph. Therefore, we may conclude that our algorithm correctly assigns the minimum number of colours to the vertices, as the graph turns into its entire form. So, the chromatic number of a bird graph is successfully computed. ♦

By Figure 6.2, we have given an agreeable attempt to make it visualize how chromatic number of a bird graph is traced out. As stated above, the execution of the algorithm commences from a forest (that had been obtained from a bird graph), where all the chest vertices of a bird graph in Figure 5.5(a) get deleted. Therefore, as usual, in Figure 6.2(a), the vertices of the forest are coloured by two colours, as shown by black and white (those are symbolized by ‘B’ and ‘W’ after the nomenclature assigned
to the vertices). In the next step, chest vertex $d$ is introduced into the graph in (b). Now, vertex $d$ is adjacent to vertices $e$, $m$, $n$, $p$, and $f$. Here, some of these adjacent vertices of vertex $d$ are coloured white (as vertices $p$ and $m$) whereas the remaining is coloured black. Hence, to colour vertex $d$, the algorithm assigns a new colour to vertex $d$, here it is red (represented by ‘$R$’).

In a similar way, in the next step, chest vertex $k$ is added into the partially constructed structure. However, similar case occurs for vertex $k$, as some of its neighbour vertices are assigned black whereas some of the vertices are remained white. Again vertex $k$ and vertex $d$ are independent to each other.

**Figure 6.2:** Computation of chromatic number of a given bird graph as depicted in Figure 5.5(a). (a) The forest we obtain after deletion of chest vertices from the given bird graph shown in Figure 5.5. (b) The last most prioritized chest vertex $d$ is added to the forest in (a) yielding a partially constructed bird graph. Hence vertex $d$ gets a different colour. (c) The chest vertex $k$ is introduced into the graph in (b). As vertex $k$ is independent to vertex $d$, hence, the colour assigned to vertex $d$ is also assigned to vertex $k$.

So, there is no problem in allocating a same colour to these two vertices as depicted in Figure 6.2(c). In this way, the algorithm for computing the chromatic number of a bird graph progresses.
6.4 Computation of the Largest Independent Set of a Bird Graph

To compute the independence number of a bird graph, again we have to start from the forest that is reached by deleting all the chest vertices by non-increasing order of priority (by application of recognition algorithm of bird graph). Initially, to find out the independent vertices, alternating vertices of the forest are picked up. Now, we add the chest vertices starting from the vertex deleted last based on priority.

After addition of a chest vertex within the partially computed independent set, we check whether inclusion of a very vertex updates the size of an independent set computed so far. It may be possible that the inclusion of the very chest vertex in the residual graph structure does not increment the size of the independent set computed so far.

However, irrespective of whether size of the largest independent set computed so far gets modified or not, each of the chest vertices added into the partially constructed bird graph is being encompassed by at least one independent set. In addition to that, an independent set is preserved comprising only non-chest vertices. Therefore, a new chest vertex as it is being included in the partially constructed graph structure is also included to those independent sets that preserve its property of being independent. Moreover, accumulation of the chest vertex should increment the sizes of the corresponding independent sets as well. The algorithm continues until all the vertices of the graph are included one after another and the given bird graph is obtained. After all the chest vertices are added in the reverse order (in which the said vertices get deleted by recognition algorithm of bird graph), the algorithm, Computing_Largest_Independence_Number, merely outputs the set with maximum cardinality.

6.4.1 The Algorithm at a Glance for Computing_Largest_Independence_Number of a Bird Graph

<table>
<thead>
<tr>
<th>Computing_Largest_Independence_Number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> A bird graph.</td>
</tr>
<tr>
<td><strong>Output:</strong> A largest independence number of the given graph.</td>
</tr>
</tbody>
</table>

**Step 1:** Call the algorithm for recognizing bird graph, and then sequentially delete chest vertices.

**Step 2:** Compute the largest independence set from the forest.
Step 3: Add a chest vertex that was deleted in reverse order in Step 1 in the residual graph.

Step 4: Generate one new independent set comprising the chest vertex newly added into the partially constructed bird graph.

Step 5: Preserve all the independent sets computed so far.

Step 6: Repeat Steps 3-5 until all the chest vertices are added in reverse order of their deletion by the chest vertices (along with its edges) to the graph constructed so far.

Step 7: Terminate the algorithm until all the (chest) vertices (along with their edges) in succession have been added and output the size of the independent set largest among those computed and maintained.

Following the logic of computing the largest independence number, the said algorithm commences working on the forest that is reached from the bird graph of Figure 5.5(a), after successive deletion of chest vertices. Now, largest independent set from the forest is traced out, which is eight (comprising vertices e, q, n, l, f, i, g and j). Now in the next step, the lowest prioritized chest vertex d gets added into the forest. After inclusion of vertex d, the size of the largest independent set is unaltered. Still including d, a new independent set is computed, comprising vertices g, j, i, d, q, and l. However, encompassing chest vertex d does not get better result in terms of the size of the independent set computed so far. Still we preserve the independent set encompassing vertex d for possible increment of size by addition of some future chest vertex (or vertices).

In the next step, the last to second chest vertex k is added into the partially constructed bird graph as depicted in Figure 6.3(c). However, similar to the earlier step, addition of chest vertex k does not change the size of the largest independent set computed so far. However, another independent set is preserved including chest vertex k, i.e., g, j, k, m, p, o. In this way, the algorithm progresses until all the chest vertices are added in succession.

To note that, the size of the largest independent set stays unaltered, nevertheless, we include vertex k in a separate independent set with vertices g, j, k, m, p and o. This is the way, the algorithm progresses as the chest vertices are added one after another.

128
Ultimately after all the chest vertices are added sequentially following the reverse order of their deletion by Step 1 of the algorithm, the independent set largest in cardinality is the required output.

The following section is equipped with essential theorem and lemma, proving that the algorithm for computation of the largest independence number of a bird graph successfully produces desired result and terminates in polynomial time.

6.4.2 Theorems Essential to Establish the Algorithm

**Lemma 6.3:** The algorithm for computing the largest independence number of a bird graph $G = (V, E)$ traces out the same in both polynomial time and space (when $|V| = n$ and $|E| = e$).

**Proof:** According to the logic of computing the largest independence number, at first the recognition algorithm is applied on the bird graph. The recognition algorithm yields a forest after successive deletion of chest vertices (as the graph under concern is a bird graph). Now, from the forest, it is of complexity $O(n)$ to identify the independent vertices, or more formally, the largest independent set. At each of the steps, as the chest vertices are added starting from the forest (along with their adjacent edges connecting the existing vertices in the forest or the last graph obtained), one at a time, we check the adjacent vertices of the vertex (newly added), that are within the forest or more formally within the largest independent set(s) from the partially computed graph.

If inclusion of the vertex in the partially constructed bird graph, under concern does not increase the size of the largest independent set computed so far, then the vertices included in the largest independent set remain same. However, we preserve the newly added chest vertex in a possibly new independent set with a probability that addition of some chest vertex (or vertices) in future iterations may lead the comparatively smaller independent set to the largest in cardinality.

Therefore, at most $O(n)$ vertices are added in succession to obtain the given bird graph. Due to inclusion of these vertices $O(n)$ comparisons are needed to trace out whether the size of the largest independent set computed so far is modified, and whether they are suitable to be included within the largest independent set, requiring a maximum degree time comparison, i.e., $O(n)$. The whole task is performed at most $O(n)$ times for each of the chest vertices already included or added in the partially
constructed bird graph. Hence, we may conclude that the time consumed by the algorithm to figure out the largest independent set is $O(n^3)$. However, the time and space consumed by the algorithm is overshadowed by the time and space complexities of recognition algorithm of bird graph. So, we may conclude that time essential for tracing out the largest independent set is $O(n^4)$, whereas the space is bounded by $O(n + e)$ for a bird graph of order $n$ and size $e$.

![Figure 6.3](image)

**Figure 6.3:** Execution of the algorithm to compute a largest independent set from a bird graph shown in Figure 5.5(a). (a) The forest that is obtained at the end of the bird graph recognition algorithm when all chest vertices are deleted in succession from the given bird graph. Here eight vertices are independent to each other. (b) The graph after inclusion of the least prioritized chest vertex $d$ along with its adjacent edges connecting the existing vertices in the forest in (a). However, inclusion of vertex $d$ as a member of the independent set decreases its size. However, due to inclusion of vertex $d$ along with its incident edges eventually renders the size of the independent set same. Hence, we exclude vertex $d$ from being a member of the partially computed largest independent set. (c) Similar case occurs as next chest vertex $k$ gets added into the partially constructed bird graph. This is the way, the algorithm progresses as the chest vertices are added one after another.
**Theorem 6.3:** The algorithm for computing a largest independence number of a bird graph correctly computes the same.

**Proof:** To compute the largest independence number of a bird graph, we have initially formed an independent set from the vertices that are picked up from the forest. It is already declared that at first we have applied recognition algorithm to find out the forest by successive deletion of chest vertices. It is obvious that there are more vertices in the eventual residual graph (i.e., the forest as referred to), with respect to the number of chest vertices being deleted one after another by the recognition algorithm. Now, iteratively we pick up chest vertices (in the order that is opposite of deletion of them) and add them in the partially constructed graph. At the same time we check whether the recently added chest vertex, if included within the partially constructed independent set decreases its size (or not).

If instead of decreasing the size of the independent set (that aims to be of largest size at the end), its size increases by accumulation of the very chest vertex, then it is safe to include the very chest vertex. Otherwise, maintain an independent set (may not be the largest in size at this moment but for future processing), as chest vertices are going to be included successively in reverse order.

In this way, initiating from forest, the algorithm for computing the largest independent set of vertices, makes sure that no chest vertex is left for inclusion within some of the independent sets. So, we may state that at the end of the algorithm, a largest independent set is successfully computed. ♦

**6.5 Computation of the Clique Cover Number of a Bird Graph**

In order to figure out the maximal cliques covering vertices of a given bird graph, we have to work on the largest stable (or independent) set. Hence, using the algorithm to compute the stable set (or the largest independent set), we trace out the independent vertices belonging to the set. Now, as usual, computation of clique cover number is based on the forest that is reached by application of the recognition algorithm of bird graph. In the next step, vertices (not being member of the largest independent set) are added based on their non-decreasing degree as in the original graph. In this purpose, we have maintained an adjacency list, whose each of the header nodes is an independent vertex and the vertices forming cliques with this vertex are linked together through a linked list.
Therefore, whenever a vertex is added to the partially constructed bird graph, according to its adjacency with any of the vertices belonging to the largest independent sets, and its adjacent vertices forming a clique (as listed by the linked list from the adjacency header node structure), the vertex is added to a minimum number of the lists maintained through link fields of the independent vertices.

It is quite natural that a new vertex being added in the partially constructed bird graph may be adjacent to the vertices of more than one clique. However, as clique cover encompasses all vertices by maximal cliques, the vertex under concern is added to only those cliques that are not yet turned into a maximal one by its adjacent vertices already included in the partially constructed bird graph. After addition of all the chest vertices (along with their associated edges) we obtain the given input bird graph back.

Now, the linked lists linked to the header nodes having one to one correspondence with the vertices being members of the largest independent set of the graph, is certain to produce a set of maximal cliques covering all vertices of the given graph. During addition of a chest vertex, in reverse way, it is added to only those sets not reaching to maximal clique. Hence, the number of maximal cliques produced in this way is sure to be the minimum one. If with a vertex of the (largest) stable set no chest vertices are being added till end of the algorithm Computing_Clique_Cover, then the very vertex may form a maximal clique of cardinality one.

6.5.1 A Stepwise Algorithm for Computing the Clique Cover Number

*Computing_Clique_Cover*

**Input:** A bird graph.

**Output:** The clique cover number of the given graph.

**Step 1:** Call the algorithm for recognizing bird graph, and delete all the vertices marked as chest vertices.

**Step 2:** Apply the algorithm to find out the largest independent set.

**Step 3:** Compute the maximal cliques each comprising at least one of the vertices of the largest independent set.

**Step 4:** Chest vertices are added successively, that are not included within the largest independent set based on lowest degree.

**Step 5:** Compare the adjacencies of these vertices and include them one at a time in a minimum number of the maximal cliques.
**Step 6:** Continue Steps 4-5, until each of the remaining vertices is included in a single clique that is maximal with respect to the graph computed so far.

**Step 7:** Terminate the algorithm and as the clique cover computation is over, output number of corresponding maximal cliques that covers all vertices of the graph.

The following section holds a theorem and a lemma. By the theorem, we have proved that the algorithm devised in this chapter for computation of clique cover number successfully executes, when a bird graph is given as input. In addition to that a lemma is associated with the theorem, proving that the algorithm yields the desired result in polynomial time.

### 6.5.2 Theorems Necessary to compute the Clique Cover Number of a Bird Graph

**Lemma 6.4:** The time complexity required for computing clique cover number of a bird graph $G = (V, E)$ is $O(n^4)$, while the space consumed by the algorithm is bounded by $O(n + e)$, where $|V| = n$ and $|E| = e$.

**Proof:** The algorithm for computing the clique cover number depends on the algorithm for tracing out the largest independence number. According to the logic of the algorithm for computing the clique cover number, we start from each of the vertices of the largest independent set. Here each of the vertices of the largest independent set is present in one of the cliques (computed so far). From then onwards, we add the remaining chest vertices (along with their associated edges) to a minimum number of the partially computed cliques (considering the adjacency of the vertex to be included within a clique). This step consumes $O(n)$ comparisons for at most $O(n)$ number of vertices.

Therefore, we may conclude that the time complexity of computing clique cover number is bounded by the time consumed by the algorithm for computing the largest independence number. However, all of these optimization algorithms first of all apply the recognition algorithm to reach to the forest starting from the bird graph shown in Figure 5.5(a). Therefore, the actual complexities in terms of both time and space are bounded by the complexities of recognition algorithm of the bird graph. Hence, the time complexity is bounded by $O(n^4)$ and space complexity is bounded by $O(n + e)$ for a bird graph $G = (V, E)$, where $|V| = n$ and $|E| = e$. ♦
Theorem 6.4: The algorithm for computing the clique cover number of a bird graph successfully computes the same.

Proof: The algorithm developed in this chapter at first traces out vertices of the largest independent set by application of the algorithm in computing the largest independence number. It is evident that the vertices of the independent set belong to distinct maximal cliques. Therefore, starting from the vertices of a partially computed clique (initially from single vertices belonging to the stable set under concern), as the new vertices are added in non-decreasing degree, following the logic of the algorithm one vertex is added to only one clique (or a minimum number of cliques to make these cliques maximal). As we have started from the vertices of a largest independent set, the remaining vertices may be adjacent to more than one vertex of the member vertices of the largest independent set. However, we have included one vertex not belonging to the largest independent set into a single set (or a minimum number of set of vertices) forming a maximal clique of the bird graph constructed so far.

Here, vertices are added in succession following consequent degrees in non-decreasing way. Now, the algorithm initiates its execution from the largest independent set computed by the algorithm in Section 6.4. Each of the vertices (not being member of the largest independent set) is added into the minimum number of partially computed cliques based on corresponding degrees in non-descending order. Hence, ultimately all the vertices are covered by some maximal clique and as the computation commences from the largest independent set, we claim that the vertices of the given graph is covered by a minimum number of maximal cliques. So, we may conclude that the algorithm performs perfectly and produces the result in desired form.

6.6 Summary

In the previous chapter, we have discovered a class of perfect graph, such that instances of the graph class are constructed by sharing vertices and/or edges of basic structures or basic building blocks following some logic. However, the blended basic structures have some chest vertices which if deleted following a particular sequence eventually yield a forest. We have characterized the graph class that has led to the recognition algorithm of the bird graph and a proof that the graph class belongs to the domain of perfect graph. The characterization that we have derived in Section 5.2,
yielded algorithms of polynomial functions for computing various optimization problems of the said graph class.

The optimization problems covered in this chapter encompass computation of clique number, chromatic number, clique cover number, and stable (or independent) number. Again, based on the characterization of bird graph, it is possible to devise logic of implementing bird graph of larger sizes from the basic building blocks. Although we have not shed light on the application domain of bird graph, we strongly guess that there exist thoughtful applications of modeling problems by bird graph belonging to varieties of fields like molecular biology, social network, connecting web pages of World Wide Web, and so on and so forth.