"GRAPH ALGORITHMS"
CHRISTOFIDES

Chapter 3

Independent and Dominating Sets —The Set Covering Problem

1. Introduction

Given a graph $G = (X, \Gamma)$, one is often interested in finding subsets of the set X of vertices of G which possess some predefined property. For example, what is the maximum cardinality of a subset $S \subseteq X$ so that the subgraph $\langle S \rangle$ is complete? or what is the maximum cardinality of a subset S so that $\langle S \rangle$ is totally disconnected? The answer to the first question is known as the *clique number* of G and the answer to the second question as the *independence number*. Another problem is to find the minimum cardinality of a subset S so that every vertex of $X-S$ can be reached from a vertex of S by a single arc. The answer to this problem is known as the *dominance number* of G.

These numbers and the subsets of vertices from which they are derived, describe important structural properties of the graph, and have a variety of direct applications in project scheduling [3], cluster analysis and numerical taxonomy [36, 2], parallel processing on a computer [17] and in facilities location and placement of electrical components [66, 56].

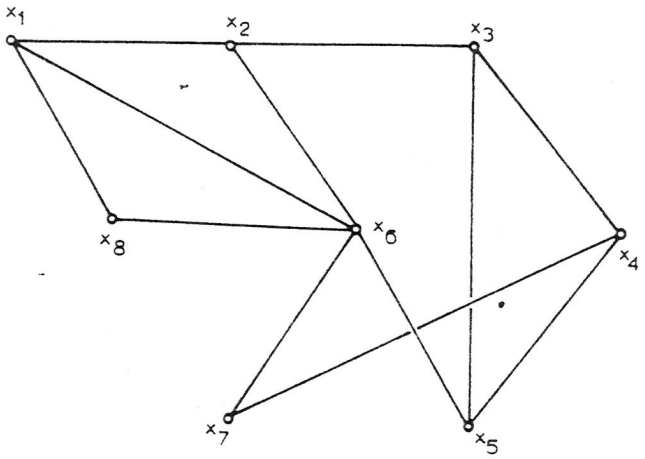

FIG. 3.1

In the present Chapter algorithms are given for the determination of the above numbers and several of the application areas are discussed. The *Set Covering Problem*—which is a generalization of the problem of determining the dominance number of graph—is introduced and a method for its solution is described. This last problem is very important not only because of the large number of its direct applications, but also in that it appears often as a subproblem in other areas of graph theory covered by this book. In particular it plays an important role in the determination of the "chromatic number" (Chapter 4), graph "centres" (Chapter 5) and "matchings" (Chapter 12).

2. Independent Sets

Consider a nondirected graph $G = (X, \Gamma)$. An *independent vertex set* (also known as an *internally stable set* [11]) is a set of vertices of G so that no two vertices of the set are adjacent; i.e. no two vertices are joined by a link. Hence, any set $S \subset X$ which satisfies the relation:

$$S \cap \Gamma(S) = \phi \tag{3.1}$$

is an independent vertex set. For the graph of Fig. 3.1 for example, the sets of vertices: $\{x_7, x_8, x_2\}, \{x_3, x_1\}, \{x_7, x_8, x_2, x_5\}$ etc. are all independent vertex sets. When no confusion arises the word "vertex" will be dropped and these will simply be referred to as independent sets.

An independent set is called maximal when there is no other independent set that contains it. Thus, a set S is a *maximal independent set* if it satisfies (3.1) together with the relation

$$H \cap \Gamma(H) \neq \phi, \qquad \forall H \supset S \tag{3.2}$$

hence for the graph of Fig. 3.1 the set $\{x_7, x_8, x_2\}$ is not maximal but the set $\{x_7, x_8, x_2, x_5\}$ is. The sets $\{x_1, x_3, x_7\}$ or $\{x_4, x_6\}$ are also maximal independent sets of the graph in Fig. 3.1 and, therefore, in general there is more than one maximal independent set to a given graph. One should also note that the number of elements (vertices) in the various maximal independent sets is not the same for all the sets as can be seen from the above example.

If Q is the family of internally stable sets then the number

$$\alpha[G] = \max_{S \in Q} |S| \tag{3.3}$$

is called the *independence number* of a graph G, and the set S^* from which it is derived is called a *maximum independent* set.

In the graph of Fig. 3.1 the family of maximal independent sets is composed of the sets:

$$\{x_8, x_7, x_2, x_5\}, \{x_1, x_3, x_7\}, \{x_2, x_4, x_8\}, \{x_6, x_4\}, \{x_6, x_3\}$$
$$\{x_7, x_5, x_1\}, \{x_1, x_4\}, \{x_3, x_7, x_8\}.$$

The largest of these sets has 4 elements and hence $\alpha[G] = 4$. The set $\{x_8, x_7, x_2, x_5\}$ is a maximum independent set.

2.1 Example: project selection

Consider n projects which must be executed, and suppose that each project x_i requires some subset $R_i \subseteq \{1, \ldots, p\}$ of p available resources for its execution. Further assume that each project (given its resource requirements), can be executed in a single time period. Now form a graph G each vertex of which corresponds to a project and with links (x_i, x_j) added whenever x_i and x_j have some resource requirement in common, i.e. whenever $R_i \cap R_j \neq \phi$. A maximal independent set of G then represents a maximal set of projects that can be executed simultaneously during the single time period.

In a dynamic system new projects become available for execution after each time period, so that the family of maximal independent sets of G has to be found repeatedly in order to choose which maximal set of projects to execute during the current period. In a practical system, it is not sufficient simply to execute the set of projects corresponding to the maximum independent set at every period, since some projects may then be delayed indefinitely. A better way would be to give a penalty p_i to every project (vertex) x_i which increases as the delay in the execution of x_i increases, and then at every point choose from the family of maximal independent sets that set which maximizes some function of the penalties on the vertices that it contains.

2.2 Maximal complete subgraphs (cliques)

A concept which is the opposite of that of the maximal independent set is that of a maximal complete subgraph. Thus, a *maximal complete subgraph* (*clique*) of G is a subgraph based on the set S of vertices which is complete and which is maximal in the sense that any other subgraph of G based on a set $H \supset S$ of vertices is not complete. Hence, in contrast to a maximal independent set for which no two vertices are adjacent, the set of vertices of a clique are all adjacent to each other. It is quite obvious, therefore, that the maximal independent set of a graph G corresponds to a clique of the graph $\bar{G}$ and *vice versa*, where $\bar{G}$ is the graph complementary to G.

It is also quite apparent, that for a nondirected graph, a clique is a concept which is similar, but more stringent, than that of a strong component of a

graph (see Chapter 2). A clique may in fact also be considered as a strong component where reachabilities are restricted to single arcs as mentioned in Section 5 of Chapter 2.

In a way analogous to the definition of the independence number by equation (3.3) we can define the *clique number* (also known as the *density*) as being the maximum number of vertices in a clique. Then, roughly speaking, for a "dense" graph the clique number is likely to be high and the independence number low, whereas for a "sparse" graph the opposite is likely to be the case.

2.3 The computation of all maximal independent sets

Because of the relationship between cliques and maximal independent sets mentioned above, the methods of computation presented in this section and which are described in terms of the maximal independent sets can also be used directly for the computation of cliques. One may, at first sight, suppose that the computation of all the maximal independent sets of a graph is a very simple task, which could be done by systematically enumerating independent sets, testing if they are the largest possible (by attempting to add any other vertex to the set whilst preserving independence) and storing the maximal ones. The supposition is certainly true for small graphs of, say, up to 20 vertices, but as the number of vertices increases this method of generation becomes computationally unwieldy. This is so, not so much because the number of maximal independent sets becomes excessive, but owing to the fact that during the process a very large number of independent sets are formed and subsequently rejected because they are found to be contained in other previously generated sets and are, therefore, not maximal in themselves.

In the current section we will describe a systematic enumerative method due to Bron and Kerbosch [14], which substantially overcomes this difficulty so that independent sets—once generated—need not be checked for maximality against the previously generated sets.

2.3.1 THE BASIS OF THE ALGORITHM. The algorithm is essentially a simple enumerative tree search during which—at some stage k—an independent set of vertices S_k is augmented by the addition of another suitably chosen vertex to produce an independent set S_{k+1}, at stage $k+1$, until no further augmentation is possible and the set becomes a maximal independent set. At stage k let Q_k be the largest set of vertices for which $S_k \cap Q_k = \phi$, i.e. any vertex from Q_k added to S_k produces a set S_{k+1} which is independent.

At some point during the algorithm, Q_k consists of two types of vertices: vertices in Q_k^- (say) which have already been used earlier in the search to augment S_k and vertices in Q_k^+ (say) which have not yet been used. A forward branching during the tree search then involves choosing a vertex $x_{i_k} \in Q_k^+$,

appending it to S_k to produce

$$S_{k+1} = S_k \cup \{x_{i_k}\} \tag{3.4}$$

and creating new sets Q_{k+1} as:

$$Q_{k+1}^- = Q_k^- - \Gamma(x_{i_k}) \tag{3.5}$$

and

$$Q_{k+1}^+ = Q_k^+ - \Gamma(x_{i_k}) - \{x_{i_k}\} \tag{3.6}$$

A backtracking step of the algorithm involves the removal of x_{i_k} from S_{k+1} to revert back to S_k, and the removal of x_{i_k} from the old set Q_k^+ and its addition to the old set Q_k^- to form two new sets Q_k^+ and Q_k^-.

The following observations are quite obvious. A set S_k is a maximal independent set only if it cannot be augmented further, i.e. only if $Q_k^+ = \phi$. If $Q_k^- \neq \phi$, it immediately follows that the current S_k was at some previous stage augmented by some vertex in Q_k^- and is therefore not maximal. If $Q_k^- = \phi$, the current S_k has not been previously augmented, and since sets are generated without duplication, S_k is a maximal independent set. Thus, the necessary and sufficient conditions for S_k to be a maximal independent set are:

$$Q_k^+ = Q_k^- = \phi \tag{3.7}$$

It is now quite apparent that if a stage is reached when some vertex $x \in Q_k^-$ exists for which $\Gamma(x) \cap Q_k^+ = \phi$, then regardless of which vertex from Q_k^+ is used to augment S_k in any number of forward branchings, vertex x can never be removed from Q_p^- at any future step $p > k$. Thus, the condition:

$$\exists x \in Q_k^- \quad \text{so that} \quad \Gamma(x) \cap Q_k^+ = \phi \tag{3.8}$$

is sufficient for a backtracking step to be taken since no maximal independent set can result from any forward branching from S_k.

As in all tree search methods, it is obviously beneficial to aim for early backtracking steps since these tend to limit the unnecessary part of the tree search. It is therefore worth while to aim at forcing condition (3.8) as early as possible by a suitable choice of the vertices used in augmenting the set S_k. During a forward step one can choose any vertex $x_{i_k} \in Q_k^+$ with which to augment set S_k, and on backtracking x_{i_k} will be removed from Q_k^+ and inserted into Q_k^-. Now if x_{i_k} is chosen to be a vertex $\in \Gamma(x)$ for some $x \in Q_k^-$, then at the corresponding backtracking step the number:

$$\Delta(x) = |\Gamma(x) \cap Q_k^+| \tag{3.9}$$

will be decreased by one (from its value prior to the completion of the forward and backward steps) so that condition (3.8) is now more nearly true.

Thus, one possible way of choosing the vertex x_{i_k} with which to augment S_k, is to first determine the vertex $x^* \in Q_k^-$ with the smallest possible value of $\Delta(x^*)$ and then choose x_{i_k} from the set $\Gamma(x^*) \cap Q_k^+$. Such a choice of x_{i_k} at every step will cause $\Delta(x^*)$ to decrease by one every time a backtracking step at level k is made, until eventually vertex x^* satisfies condition (3.8), at which time backtracking can occur.

It should be noted that since at a backtracking step x_{i_k} enters Q_k^-, it may be that this new entry has a smaller value of Δ than the previously fixed vertex x^* so that this new vertex should now become the target for the attempt to force condition (3.8). This is particularly important in the first forward branchings when $Q_k^- = \phi$.

2.3.2 DESCRIPTION OF THE ALGORITHM

Initialisation

Step 1. Set $S_0 = Q_0^- = \phi$, $Q_0^+ = X$, $k = 0$.

Forward step

Step 2. Choose a vertex $x_{i_k} \in Q_k^+$ as mentioned earlier and form S_{k+1}, Q_{k+1}^- and Q_{k+1}^+ keeping Q_k^- and Q_k^+ intact. Set $k = k + 1$.

Test

Step 3. If condition (3.8) is satisfied ~~to~~ go to step 5, else go to step 4.

Step 4. If $Q_k^+ = Q_k^- = \phi$ print out the maximal independent set S_k and go to step 5. If $Q_k^+ = \phi$ but $Q_k^- \neq \phi$ go to step 5. Otherwise go to step 2.

Backtrack

Step 5. Set $k = k - 1$. Remove x_{i_k} from S_{k+1} to produce S_k. Retrieve Q_k^- and Q_k^+, remove x_{i_k} from Q_k^+ and add it to Q_k^-. If $k = 0$ and $Q_0^+ = \phi$, Stop. (All maximal independent sets have been printed out). Otherwise goto step 3.

Comments on the implementation of the above algorithm are given in [14] together with an Algol program listing. This program has been tested on a number of graphs (including the Moon–Moser graphs—see problem 10), and the computation times per maximal independent set generated were found to be almost constant and independent of the size of the graph itself, which makes the algorithm close to the best possible.

3. Dominating Sets

For a graph $G = (X, \Gamma)$ a *dominating vertex set* (also known as an *externally stable set* [11]), is a set of vertices $S \subseteq X$ chosen so that for every vertex x_j not in S, there is an arc from a vertex in S to x_j.