17 Combinatorics

17.5 Matchings (6 units)

In this project you will need to be able to generate random $n \times n$ bipartite graphs, that is, bipartite graphs whose each class has n vertices. The edges appear independently and at random with probability p. Sometimes it will be necessary to generate a random bipartite graph process; starting with an edgeless $n \times n$ bipartite graph, the edges are added one by one until the complete bipartite graph is obtained, the edge to be added at any stage being chosen at random from those currently absent.

A *matching* in a graph is a set of *independent* edges (edges with no common vertex). A *1-factor* in a graph is a 1-regular spanning subgraph, or, in other words, a matching meeting every vertex. A 1-factor in a $n \times n$ bipartite graph is a matching of size n.

Let G be a bipartite graph with vertex classes X and Y, where $|X| = |Y| = n$. If A is a subset of X its *neighbourhood* $\Gamma(A)$ is the set of vertices in Y joined to at least one member of A. A set $A \subseteq X$ is called a *blocking* set if $|\Gamma(A)| < |A|$. Hall's theorem tells us that G has a 1-factor if and only if G has no blocking set. The main task in this project is to develop an algorithm which, given input G, will output either a 1-factor or a blocking set.

Question 1 One algorithm to check whether G has a 1-factor is to check each subset A to see if it is a blocking set. Why is this a poor method?

Our algorithm to find a 1-factor will consist of finding successively larger matchings, starting with the empty matching. Let M be a matching meeting the sets of vertices $A \subseteq X$ and $B \subseteq Y$. Let $u \in X \setminus A$. An *alternating path* is a path from u to a vertex $v \in Y$, such that every second edge is in M. We say that v is *reachable* from u by an alternating path; let V be the subset of Y consisting of vertices reachable from u.

Question 2 How can a matching larger than M be found if $V \not\subseteq B$?

Question 3 How can a blocking set be found if $V \subseteq B$?

The set V can be computed by initially setting $V = \emptyset$. Add to V the neighbours of u, then find the vertices of X matched to V, then add to V the neighbours of those vertices, and so on. For each vertex added, the preceding vertex should be recorded which led to the addition.

Question 4 Write down clearly an algorithm to find a 1-factor based on these ideas. You should give enough detail to make it clear exactly how the computation is done, but do not give implementation details.

Question 5 Implement your algorithm. Your program should do its computation using the adjacency matrix of the input graph, and the output should show (at least) whether a 1-factor exists or, if not, the size of a blocking set. For each of the fourteen values of p varying from 0.06 to 0.42 by steps of 0.06 and from $0.1 \ln n/n$ to $1.9 \ln n/n$ by steps of $0.3 \ln n/n$, run the program on twenty random $n \times n$ bipartite graphs with $n = 40$ Tabulate your results, and comment on them briefly; give some attention in your comments to the sizes of the minimal blocking sets that your algorithm finds.

Question 6 Run your program on ten random bipartite graph processes, with $n = 20$. Tabulate your results. What simple properties of a graph are necessary for a 1-factor to exist? What do you notice about your results? Why do you think the above values of p were chosen?

An alternative way to store the graph in the computer is by an *adjacency list*. This is where, for each vertex in X and for each vertex in Y, a list of its neighbours is stored. Thus if x_4 is joined only to y_1, y_9 and y_{14}, the list for x_4 is just (y_1, y_9, y_{14}).

Question 7 What is the (order of magnitude) complexity of your algorithm when the adjacency matrix is used? What would be the effect of using the adjacency list instead? (Do not implement a version using the adjacency list.)

Question 8 Is the alternating path method for finding a 1-factor effective for all graphs, and not just bipartite ones? Justify your answer.

If a bipartite graph fails to have a 1-factor, it is nevertheless of interest to find a matching of the maximum possible size.

Question 9 In what ways can your algorithm be modified to find a maximum matching in a bipartite graph?

References

[1] Bollobas, B., Modern Graph Theory, Springer 1998.

