The Theory of Computation

Bernard M.E. Moret

Addison-Wesley-Longman, 1998

Solution Manual
Chapter 7, Problem 7.19

The problem is in NP: we can detect whether or not a graph has a cycle in linear time by breadth-first search. We prove that it is NP-complete by transforming *3SAT* to it. Our graph will have a vertex for each clause element, plus two vertices for each variable (one for the uncomplemented literal, one for the complemented literal), plus two distinguished vertices to represent *true* and *false*. The true and the false vertices are connected in a 2-cycle; the three elements of each clause are connected in a 4-cycle with the false vertex; and any two vertices corresponding to a complemented and an uncomplemented copy of a variable are connected in a 2-cycle. This transformation is obviously polynomial-time.

Now, in order to have a solution to our problem, each cycle must be broken. For the 2-cycles, this means that the two vertices involved must end up on different sides of the partition; for the 4-cycle, this means that at least one of the four vertices must end up on the opposite side from the others. Thus the true and false vertices end up on different sides, thereby labelling one side as that of the true literals and the other as that of the false literals. The two vertices associated with each variables must end up on opposite sides, thereby giving us a truth assignment. At least one of the three vertices associated with each clause will end up on the true side (opposite the false vertex present in the 4-cycle) and thus ensure satisfaction of the clause. Finally, the 2-cycles connecting any two complementary literals guarantee the uniformity of the partition: all vertices representing the same literal must end up on the same side of the partition. Thus a solution to our problem yields a satisfying truth assignment for the *3SAT* problem. The converse is obvious.

A variation would omit the 2-cycles between complemented and uncomplemented copies of literals and represent each clause by a single vertex connected with the false vertex in a 2-cycle, but add 3 vertices for each clause, to serve as connection points; the clause vertex is connected to each of the 3 vertices by a 2-cycle, and each of the 3 vertices is in turn connected to the corresponding literal in the truth-setting component by a 2-cycle.

Each 2-cycle must be broken, thereby ensuring that all connecting vertices will end up in the false side of the partition; the rest follows easily.

Both versions can be simplified if one transforms *NAE3SAT*, since there is then no need for the true and false vertices. There are probably as many proofs for this problem as there are people to write proofs.