The Theory of Computation
Bernard M.E. Moret
Addison-Wesley-Longman, 1998

Solution Manual
Chapter 7, Additional Problem 15

Membership in NP is clear (a solution can be verified edge by edge). We transform *NAE3SAT* to our problem, relying as usual on triangles, as follows. Each clause will be represented by a triangle. Since at most one edge per triangle can be part of the matching, each triangle must have at least one node labelled *null*; if we can also ensure that at least one edge from each triangle must be present in the matching, then the labelling of each triangle is fixed, with each vertex having a different label. Each variable will be represented by a simple path; if variable x_i occurs n_i times in all, its component includes $4n_i$ vertices, call them x_{ij}, $\overline{x}_{ij}$, a_{ij}, and b_{ij}, for $1 \leq j \leq n_i$, and $4n_i - 1$ edges, connecting x_{ij} to $\overline{x}_{ij}$, $\overline{x}_{ij}$ to a_{ij}, and a_{ij} to b_{ij}, as well as b_{ij} to $x_{i,\,j+1}$. The intent is to have the x_{ij} and b_{ij} vertices labelled by either *source* or *sink* ("true" and "false") and the $\overline{x}_{ij}$ and a_{ij} vertices labelled the other way, with edges $\{x_{ij}, \overline{x}_{ij}\}$ and $\{a_{ij}, b_{ij}\}$ in the matching. (We could also simply do without the a_{ij} and b_{ij} by connecting $\overline{x}_{ij}$ to $\overline{x}_{i,\,j+1}$, and flip every time.) Finally, we connect the triangle vertices to the matching connecting points; an uncomplemented literal gets connected to the appropriate x_{ij}; a complemented literal to the appropriate $\overline{x}_{ij}$. The matching size is set to $7m$, where m is the number of clauses. The construction produces $15m$ vertices and $17m$ edges and is easily carried out in polynomial time.

Given a solution to *NAE3SAT*, a solution to the transformed instance is easily constructed. The components corresponding to the variables account for $6m$ edges in the matching with a labelling as described above. In each triangle, one true literal is labelled "source," one false literal is labelled "sink," and the remaining literal is labelled "null." This contributes one matching edge for each triangle, for a total of $7m$. That the labeling conditions are respected is easily verified.

Conversely, note that the required number of edges in the matching forces $14m$ of the $15m$ vertices to be involved in the matching; in other words, $7m$ vertices must be labelled *sink*, $7m$ others must be labelled *source*, and the remaining m must be labelled *null*. A triangle and its connecting edges can only yield from one to three matching edges; the one matching edge can be a triangle edge, but in order to obtain two or more matching edges, the triangle edges cannot be included. Moreover, in order to obtain three matching edges, all triangle vertices must be labelled with the same label (either all are labelled *source* or all are labelled *sink*)—and thus the other endpoints of the matching edges are also labelled with the same label. Now, note that using a connecting edge in the matching reduces by one the number of matching edges that a variable's component can provide. Thus, using one connecting edge per triangle gains nothing (over using just one triangle edge) and loses one; using two connecting edges gains one and loses two; using three connecting edges gains two and loses three. Since we have provided separate attachment points for these edges, the losses, as well as the gains, are independent of each other. In sum, using *any* connecting edge always results in a loss of one; since the required matching size is just met by using triangle edges, it cannot be met if any connecting edge is used. Hence any solution must be composed of one edge per triangle, no connecting edge, and $6m$ edges from the variables' components. The proof follows easily.

A variation on this theme keeps the triangles for the clauses, but uses only one edge per variable, connecting the endpoints (uncomplemented and complemented literals) to the triangle vertices with a suitable graph component. The component ensures that the two vertices it connects cannot be labelled one *source* and the other *sink*. A suitable component connects the two vertices, say u and v, with an edge and adds 8 more vertices, say u_{11}, u_{12}, u_{21}, and u_{22}, with similar four for v, and 10 more edges, $\{u, u_{11}\}$, $\{u, u_{21}\}$, $\{u_{11}, u12\}$,$\{u_{21}, u_{22}\}$, with similar versions for v, and $\{u_{11}, v_{11}\}$ and $\{u_{21}, v_{21}\}$. With u and v given identical labels, or with one of them labelled *null*, four edges of the components can be matched, by labelling the x_{-1} vertices *source* and the x_{-2} vertices *sink* (or vice-versa, depending on the non-*null* label given to u or v). With the u, v vertices such that $\{u, v\}$ is in the matching, at most three edges of the component can be matched. The bound is then set to $13m + n$ ($12m$ for the $3m$ components, m for the clause triangles, and n for the variable segments). The proof follows immediately. A simpler component just uses two extra vertices, say e_1 and e_2, with connections $\{u, v\}$, $\{u, e_1\}$, $\{e_1, v\}$, and $\{e_1, e_2\}$; this last edge can be matched if the vertices u and v are not matched.

A third style of proof keeps the clause triangles and the simple connections between the clauses and the variables, but replaces the variables single edge by a simple lozenge of four edges, with the two extreme points representing the two literals. Since using connecting edges at most yields $2n$ matches (for n variables) and since we now require $2n + m$ matches, compensation cannot occur through connecting edges. The proof follows immediately.

Finally, starting from *NAE Max2SAT*, we can devise a transformation where no *null* label is ever needed. For each variable use a lozenge as above; for each clause use a single edge; connect literal vertex in clause to literal vertex in lozenge through two copies of a lozenge with one tail (of one edge) at each end; set $K_{TR} \leftarrow K_{2SAT} + 2n + 8m$. The proof follows the lines above.