Integer programming formulation of the stasfiability problem

A *boolean variable* is a variable x that can only take on the values true (1) or false (0). A *boolean formula* is a collection of boolean variables connected by the *logical connectives* "and", "or" or "not". If we represent "and" with "$\cdot$", "or" with "+", and "not" x with "$\overline{x}$", then follow are examples of a boolean formulas on the three boolean variables x_1, x_2 and x_3:

$$\overline{x}_3 \cdot (x_1 + \overline{x}_2 + x_3) \tag{1}$$

$$(x_1 + x_2 + x_3) \cdot (x_1 + \overline{x}_2) \cdot (x_2 + \overline{x}_3) \cdot (x_3 + \overline{x}_1) \cdot (\overline{x}_1 + \overline{x}_2 + \overline{x}_3) \tag{2}$$

The *clauses* of the two formulas are the boolean formulas contained in the parenthesis, for example the clauses of (2) are

$$x_1 + x_2 + x_3, x_1 + \overline{x}_2, x_2 + \overline{x}_3, x_3 + \overline{x}_1, \text{ and } \overline{x}_1 + \overline{x}_2 + \overline{x}_3.$$

We can think of the clauses are collections of *literals*, where a literal is a variable or its negation. For example, we can represent (2) as $\{C_1, \ldots, C_5\}$ where

$$C_1 = \{x_1, x_2, x_3\}, C_2 = \{x_1, \overline{x}_2\}, C_3 = \{x_2, \overline{x}_3\}, C_4 = \{x_3, \overline{x}_1\} \text{ and } C_5 = \{\overline{x}_1, \overline{x}_2, \overline{x}_3\}.$$

A *truth assignment* is an assignment of true or false (1 or 0) to the boolean variables in a formula. We say that a boolean formula is *satisfiable* if there exist a truth assignment that makes the entire formula evaluate to true. For example, (1) is satisfiable because if we let x_3 be false, x_1 be true and x_2 be true, the entire formula (1) is true (there are other true assignments that make (1) true). It is a good exercise to show that (2) is not satisfiable.

SATISFIABILITY is the problem is to determine if a given formula is satisfiable or not. (If you know what NP-complete means, then it should be noted that SATISFIABILITY is an NP-complete problem.) We can convert the problem of determining if a given formula is satisfiable, into the problem of determining if a specific integer program is feasible.

To see this, suppose we are given a boolean formula with clauses $C_1, \ldots, C_m$, then if we construct an integer program with the constraints

$$\left(\sum_{x_j \in C_i} x_j \right) + \left(\sum_{\overline{x}_j \in C_i} (1 - x_j) \right) \geq 1,$$

and we specify that each variable can only take on the values 0 or 1, then the integer program is feasible if and only if the boolean formula is satisfiable.

For example, if we are given the boolean formula (2), we would construct the integer program

$$\begin{array}{lll} \text{Maximize} & 0 & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq 1 \\ & x_1 + (1 - x_2) & \geq 1 \\ & x_2 + (1 - x_3) & \geq 1 \\ & x_3 + (1 - x_1) & \geq 1 \\ & (1 - x_1) + (1 - x_2) + (1 - x_3) & \geq 1 \\ & x_1, x_2, x_3 \in \{0, 1\} & \end{array}$$

that is not feasible, because (2) is not feasible.

So we can solve any satisfiability problem, if we can solve any integer programming problems. Therefore, integer programming is at least as hard as SATISFIABILITY.