Assignment 1

due Monday, September 19, 2022

The homework is due at 6pm on Monday, September 19, 2021. Please submit your homework via Gradescope. Note that you can submit pages, code, or both. Pages and code are different assignments within Gradescope. The questions below are mostly encoding questions. No external tools are not allowed to help answering Question 1.

We prefer answers that consist of a generator that produces the requested DIMCAS file in a common programming language, such as Python or C(++). Encoding tools, such as PySAT, are allowed for Questions 2 and 3. Alternatively, you can submit the encoding answers as a LaTeXdocument. However, Questions 1(d), 2 (b), 2(c), 3(b), and 3(c) can only be solved using a generated DIMACS file.

The maximum number of regular points for this assignment is 50: the 30 points of Question 1 + either 20 points of Question 2 or 20 points of Question 3 (a) and (b). Additionally, 10 bonus points can be earned in Question 3 (c).

Question 1 (no encoding tools allowed)

(a) [10 points] Given the Boolean variables $x_1, \ldots, x_5$, construct two different encodings in conjunctive normal form that express that at most two of them can be true: $x_1 + \ldots + x_5 \leq 2$. The first encoding can only use the variables $x_1, \ldots, x_5$, while the second encoding must also use auxiliary variables.

(b) [10 points] Let us refer to the above encodings as AtMostTwoA (w/o auxiliary variables) and AtMostTwoB (with auxiliary variables). Encode the following to formulas into CNF: $y_1 \leftrightarrow \text{AtMostTwoA}(x_1, \ldots, x_5)$ and $y_2 \leftrightarrow \text{AtMostTwoB}(x_1, \ldots, x_5)$ using the Tseitin transformation.

(c) [5 points] Encode whether there exists an assignment to $x_1, \ldots, x_5$ that falsifies y_1 and satisfies y_2 by combining $y_1 \leftrightarrow \text{AtMostTwoA}(x_1, \ldots, x_5)$ and $y_2 \leftrightarrow \text{AtMostTwoB}(x_1, \ldots, x_5)$.

(d) [5 points] Solve the resulting formula using a SAT solver and show the output of the solver. (Hint: the formula should be unsatisfiable, so no local search solver can be used.)

Question 2 (answer this question or Question 3)

(a) [10 points] Consider a $n \times m$ grid of squares and all possible rectangles within the grid whose length and width are at least 2. Encode whether there exists a coloring of the grid using three colors so that no such rectangle has the same color for its four corners. (Hint: The encoding requires two types of constraints. First, each square needs to have at least one color. Second, if four squares form the corners of a rectangle, then they cannot have the same color.)

(b) [5 points] Solve the encoding for a 10×10 grid using a SAT solver and decode the solution into a valid coloring. Show the output of the SAT solver and a valid 3-coloring similar to the one above of the 9×9 grid.

(c) [5 points] Solve the encoding for a 9×12 grid using a SAT solver and decode the solution into a valid coloring. Show the output of the SAT solver and a valid 3-coloring similar to the one above of the 9×9 grid.

```
0 0 1 1 2 2 0 1 2
2 0 0 1 1 2 2 0 1
1 2 0 0 1 1 2 2 0
0 1 2 0 0 1 1 2 2
2 0 1 2 0 0 1 1 2
2 2 0 1 2 0 0 1 1
1 2 2 0 1 2 0 0 1
1 1 2 2 0 1 2 0 0
0 1 1 2 2 0 1 2 0
```

Question 3 (answer this question or Question 2)

An *almost square* is a $n \times (n+1)$ rectangle. One can cover the almost square 4×5 using the smallest three almost squares: 1×2, 2×3, and 3×4. A solution is shown below.

```
1 1 3 3 3
2 2 3 3 3
2 2 3 3 3
2 2 3 3 3
```

(a) [10 points] Encode whether the smallest k almost squares can cover an almost square. A satisfying assignment of the encoding should represent a covering. In case the smallest k almost squares don't add up to an almost square, the encoding should simply print a formula with only the empty clause.

(b) [10 points] Solve the encoding for the smallest 8 almost squares, which can cover the almost square 15×16, and decode the solution into a valid cover. Show the output of the SAT solver and valid cover similar to the one above of the 4×5 grid.

(c) [Bonus: 10 points] Construct a compact encoding for the smallest 20 almost squares, which can cover the almost square 55×56. Auxiliary variables are useful to reduce the size of the encoded formula. Bonus points are awarded for reasonably small encodings: 2 points for less than 3 million clauses; 4 points for less than 2 million clauses; 6 points for less than a million clauses; and 8 points for less than half a million clauses. All 10 points are awarded for any encoding for which you can show that a SAT solver can find a satisfying assignment. Warning: this problem is challenging.