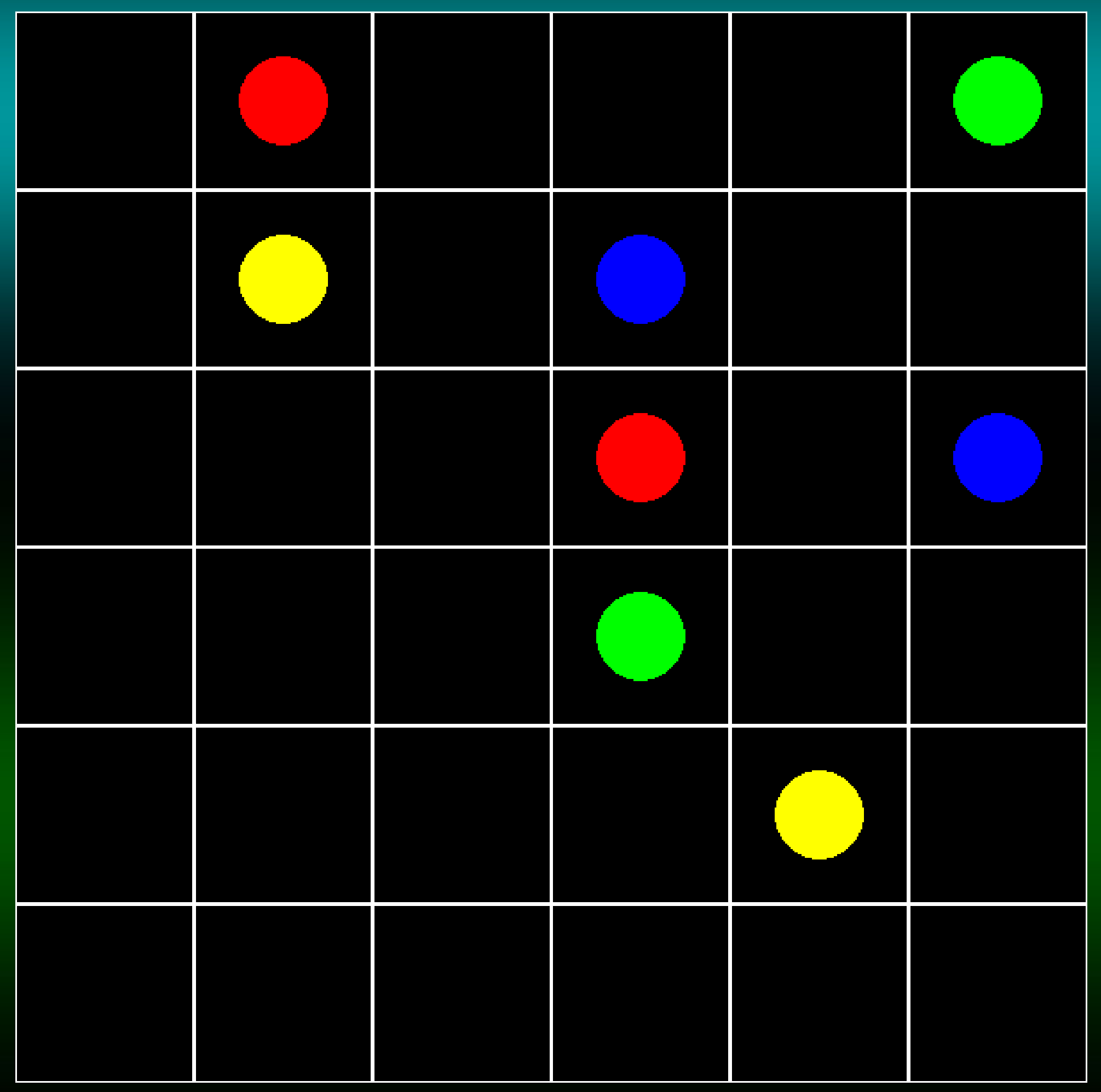
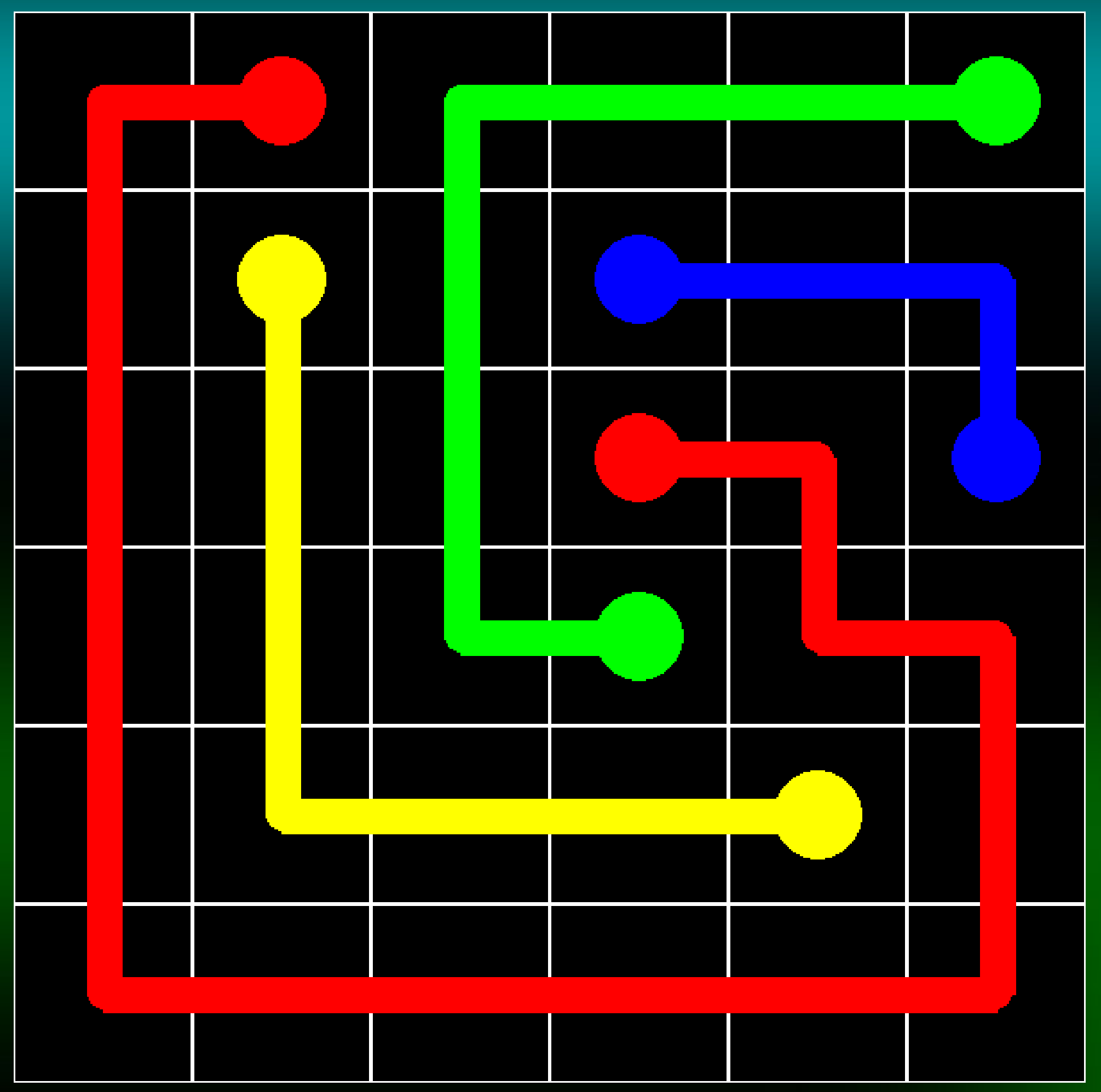
Flow Free Solver Using CNF and SAT

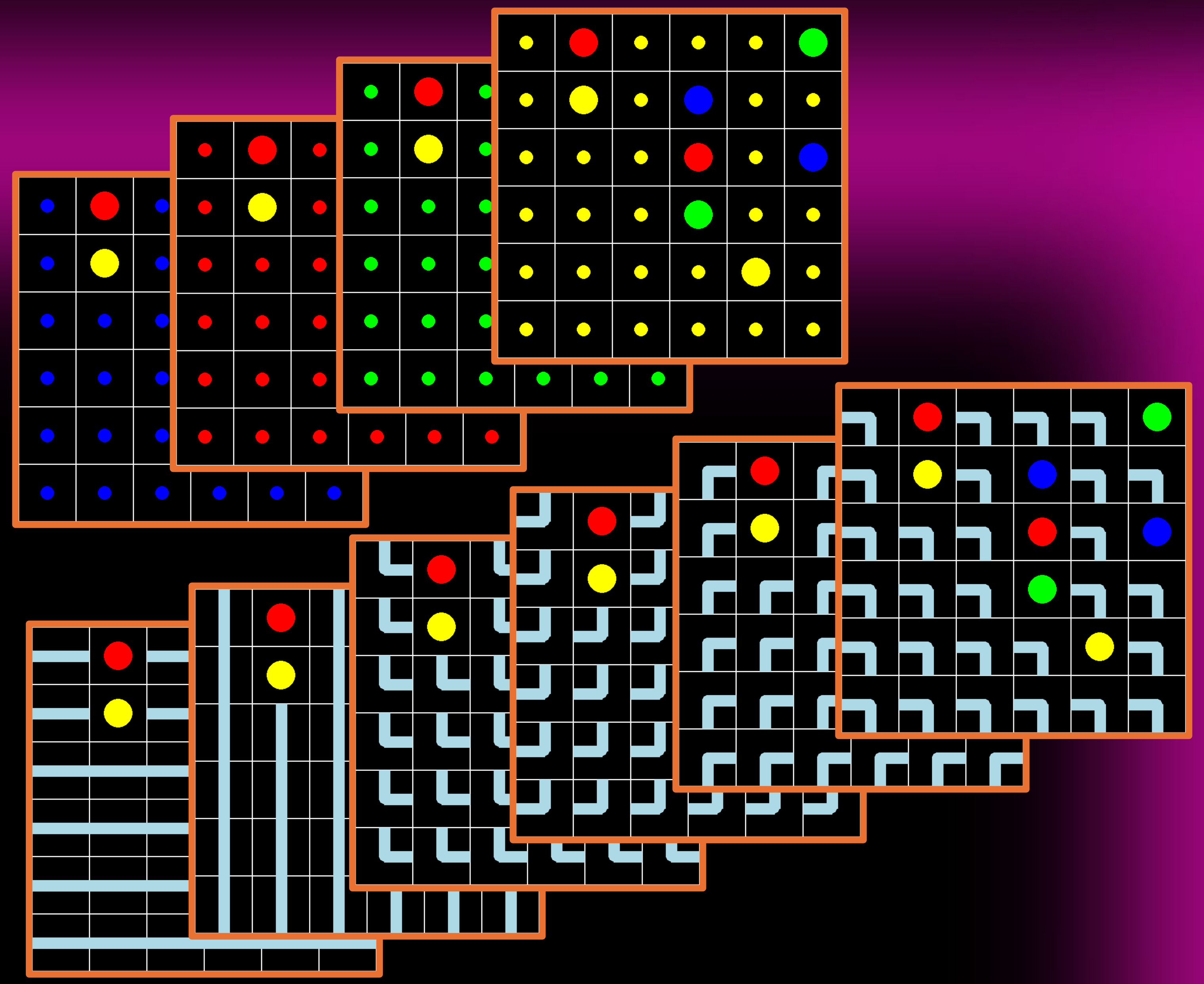




Andrew Berman, Harrison Lampert

CNF Variables:

For every cell, there is a CNF variable for every color and direction.
That means that there are:
(rows * cols * num_colors) + (rows * cols * 6) = Number of CNF Variables



Why CNF w/ SAT and not A*?

- PycoSAT, one of Python's native SAT solvers, can find a solution for 5,000 clauses in 0.001 seconds
- The idea is that if we can do most of the work up front by converting the constraints of the game into CNF, we can avoid the time and memory it would take to build and traverse the search space using A*
  - In other words, if we do more mental work ourselves, we can ease the work the computer has to do, which will improve performance.

Constraints:

- For every cell, there can only be EXACTLY ONE color and direction
- For every dot, EXACTLY ONE neighbor shares it's color
- For every direction, the connecting 2 neighbors must be the same color

Given a list of CNF variables, you can constrain them such that EXACTLY ONE is true by combining the two constraints:

- AT LEAST ONE is true:
  - $(x_1, x_2, \dots, x_n)$
- AT MOST ONE is true:
  - $(\neg x_1, \neg x_2)$ $(\neg x_1, \neg x_3)$ $(\neg x_1, \neg x_4)$ …
    $(\neg x_2, \neg x_3)$ $(\neg x_2, \neg x_4)$ …
    …
    $(\neg x_{n-1}, \neg x_n)$

	Clause Creation (ms)	Our Model (ms)	PyCoSAT (ms)	PySAT (ms)
6x6	10.867	33.213	1.718	2.220
7x7	20.849	89.550	3.196	2.963
8x8	27.273	166.891	6.348	5.544
9x9	41.320	336.799	5.564	8.413
10x10	57.983	594.686	11.762	15.043
12x12	121.185	1167.734	20.290	24.422

SAT Solver Algorithm (CDCL):

1. **Propagate unit clauses**
   - A unit clause is a clause where all but one variable is FALSE, while the remaining is unassigned
2. **Check if all clauses are satisfied** (at least one var is TRUE)
3. **Pick and assign an unassigned var and record the decision level**
4. **Propagate unit clauses again**
   - If a conflict occurred
     - Analyze the conflict and come up with a clause that will keep the conflicting assignment from being made again
     - Backtrack to the decision level that the conflicting assignment was made
     - Assign the var the opposite Boolean
   - If no conflict occurred, then repeat 2-4
5. **Conflict was at level 0** (backtracking took you back to level 0 and another conflict immediately occurred) **then CNF is "UNSAT"**

Optimizations:

1. **Pure Literal Elimination**
   - If one variable only occurs in one state in ever clause (either entirely negated or unnegated), then you can assign the corresponding Boolean.
2. **Watched Literals**
   - Create a list for each clause of 2 of the variables in that clause that you will watch
     - If one is true, then it is satisfied
     - If both are false, conflict
     - If both are unassigned, continue
     - If one is false and one is unassigned, look for another unassigned to replace the false one with
       - If one cannot be found, it is a unit clause
3. **VSIDS**
   - Choose variables that appear often when assigning