1 The model

I used the ship-based model. I first load all the ships, and the initial grid and set each variable with (row,column,orientation), where row/column represents the row/column number of the upperleft square of the ship, and orientation=0/1 means the ship is placed horizontally/vertically. I also used a dictionary to map the variable to the length of the corresponding ship.
Assume the variables are $v_1, ..., v_n$ where n is the number of ships.
The first constraint is that, for each row, the sum of squares occupied by ships in that row equals to the corresponding row constraint.
The second constraint is that, for each column, the sum of squares occupied by ships in that column equals to the corresponding column constraint.
The third constraint is that, no pair of ships can be too close. It consists of $\binom{n}{2}$ small constraints, where n is the number of total ships.
The fourth constraint is that, the final board must match the hint.
We dont intentionally check the number of each ships because we have set the ship variables of correct ships after we load the data.

2 Heuristics for speed up

I found that the Forward Checking Algorithm with MRV very fast. I implemented a function called MRV that outputs the unassigned variable with smallest CurDom. I called this function to select the variable to branch on right before the for loop.
In addition, I implement "stricter constraints" for constraint 1,2,4 for partial state so that if they are violated, then it is impossible to satisfy the final constraints (as described in paragraph 1) by adding more ships.
For example, for any row, if the sum of squares occupied by assigned ships exceed the row constraint, then it violates the "stricter constraints". Similar for columns.
There are many "stricter constraints" for constraint 4. For example, if the hint contains "L" and the square "L" hasnt been occupied by a ship of length¿=2 in partial state, then there cannot be ships near the square "L" (otherwise "L" cannot be occupied).