Battleships

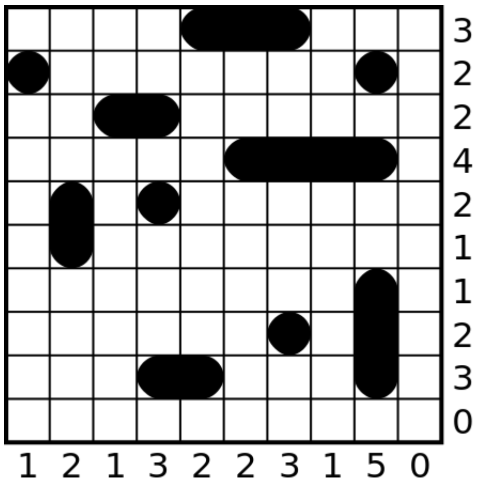
Ofir Mirkin, Ivanina Ivanova, Raphael Demoulin, Preshtibye Raggoo

March 2022

1 Introduction

Battleships also known as *Bimaru* is a logic puzzle in a square grid with simple rules and challenging solutions. The size of the grid can be from 1x1 to 10x10.

The rules of Battleships are simple:
You have to find the location of the battleships hidden in the grid.
The armada includes one battleship four squares long, two cruisers three squares long, three destroyers two squares long, and four submarines one square in size.
Each ship occupies a number of contiguous squares on the grid, arranged horizontally or vertically.
The ships are placed so that no ship touches any other ship, not even diagonally.
The numbers outside the grid show the number of cells occupied by battleships on that row/column.



2 Modeling

2.1 Variables

2.1.1 Boats

$n \in [\![1, 4]\!]$ where n is the size of the boat.
$k \in [\![1, 10]\!]$ is a boat such that :

- k $\in [\![1, 4]\!] \implies$ boat is of size n = 1
- k $\in [\![5, 7]\!] \implies$ boat is of size n = 2
- k $\in [\![8, 9]\!] \implies$ boat is of size n = 3
- $k = 10 \implies$ boat is of size n = 4

$B_{i,j,k}$: The upper left corner of the boat k at the coordinate (i,j)

2.1.2 Orientation

O_k : the boat k is vertical
$\neg O_k$: the boat k is horizontal

2.1.3 Cell

$X_{i,j,k}$: cell (i,j) occupied by boat k

2.2 Constraints

2.2.1 Boat Existence

The boat k occupies the cells such that:

$$(B_{i,j,k} \wedge O_k \implies \bigwedge_{l=0}^{n-1} X_{i+l,j,k}) \wedge (B_{i,j,k} \wedge \neg O_k \implies \bigwedge_{l=0}^{n-1} X_{i,j+l,k})$$

There is at least one boat k :

$$\bigvee_{(i,j)} B_{i,j,k}$$

There is at most one boat k :

$$\bigwedge_{(i,j)\neq(i',j')} \neg B_{i,j,k} \vee \neg B_{i',j',k}$$

2.2.2 In every cell there is only one boat or section of boat.

$$\neg(\bigvee_{i,j,k_1\neq k_2} X_{i,j,k_1} X_{i,j,k_2}) \equiv \bigwedge_{i,j,k_1\neq k_2} \neg X_{i,j,k_1} \vee \neg X_{i,j,k_2}$$

2.2.3 The numbers outside the grid show the number of cells occupied by battleships on that row/column.

$$\binom{10}{M}$$

where M is the number of occupied cells in each line or column.

2.2.4 The boats are placed so that no boat touches any other boat, not even diagonally.

$X_{i,j}$ is the initial placement of the boat.

Orientation = Horizontal

$$(B_{i,j,k} \land \neg O_k) \implies (\bigwedge_{r \in \{i-1,i+1\}} \bigwedge_{t \in [\![j-1,j+n]\!]} \neg X_{r,t}) \land \neg X_{i,j-1} \land \neg X_{i,j+n}$$

Orientation = Vertical

$$(B_{i,j,k} \land O_k) \implies (\bigwedge_{r \in \{j-1,j+1\}} \bigwedge_{t \in [\![i-1,i+n]\!]} \neg X_{t,r}) \land \neg X_{i-1,j} \land \neg X_{i+n,j}$$

3 Transform to CNF

3.1 Contiguous Cell

The boat k occupies the cells such that:

$$(B_{i,j,k} \land O_k \implies \bigwedge_{l=0}^{n-1} X_{i+l,j,k}) \land (B_{i,j,k} \land \neg O_k \implies \bigwedge_{l=0}^{n-1} X_{i,j+l,k})$$

$$\equiv$$

$$\bigwedge_{l=0}^{n-1} (\neg B_{i,j,k} \lor \neg O_k \lor X_{i+l,j,k}) \land \bigwedge_{l=0}^{n-1} (\neg B_{i,j,k} \lor O_k \lor X_{i,j+l,k})$$

3.2 No boat touches another boat

Orientation = Horizontal

$$(B_{i,j,k} \wedge \neg O_k) \implies (\bigwedge_{r\in\{i-1,i+1\}} \bigwedge_{t\in[\![j-1,j+n]\!]} \neg X_{r,t}) \wedge \neg X_{i,j-1} \wedge \neg X_{i,j+n}$$

$$\equiv$$

$$(\bigwedge_{r\in\{i-1,i+1\}} \bigwedge_{t\in[\![j-1,j+n]\!]} (\neg B_{i,j,k} \vee O_k \vee \neg X_{r,t}))\wedge$$
$$(\neg B_{i,j,k} \vee O_k \vee \neg X_{i,j-1}) \wedge (\neg B_{i,j,k} \vee O_k \vee \neg X_{i,j+n})$$

Orientation = Vertical

$$(B_{i,j,k} \wedge O_k) \implies (\bigwedge_{r\in\{j-1,j+1\}} \bigwedge_{t\in[\![i-1,i+n]\!]} \neg X_{t,r}) \wedge \neg X_{i-1,j} \wedge \neg X_{i+n,j}$$

$$\equiv$$

$$(\bigwedge_{r\in\{j-1,j+1\}} \bigwedge_{t\in[\![i-1,i+n]\!]} (\neg B_{i,j,k} \vee O_k \vee \neg X_{t,r}))\wedge$$
$$(\neg B_{i,j,k} \vee O_k \vee \neg X_{i-1,j}) \wedge (\neg B_{i,j,k} \vee O_k \vee \neg X_{i+n,j})$$

4 Implementation of programs

4.1 Programming languages

Battleship is C and Python project. Where we have mostly used C for the logical part and Python is used only for the graphical interface.

4.2 Transforming variables to literals

We have chosen the following formulas to transform the names of the variables into literals. The literals have to be of type integer because of the DIMACS format.

- $X_{i,j,k} \equiv 100 \times i + 10 \times j + k$
  - $if X_{i,j,k} = X_{0,0,0} \implies X_{i,j,k} = 2010$
- $B_{i,j,k} \equiv 1000 + 100 \times i + 10 \times j + k$
- $O_k \equiv 2000 + k$
- $\neg O_k \equiv -(2000 + k)$

4.3 Data Structures

We have worked with arrays and linked lists. Following the conventions:

- Literal is a variable of type int
- Every clause is an array if literals
- The CNF is a linked list of clauses

5 Source code

The two main parts of our Battleship project are the following:

- ***constraints.c & constraints.h*** In this part we code all the logical formulas explained above.
- ***types.c & types.h*** Here we define all the data structures that we use throughout our program and all the basic functions to manipulate those structures (e.g. add to a linked list/array).
- ***find_model_final.c*** This part of the code is taking the model and transforms it to a format that can be read by the graphical interface.

A complete guide how to use the project as well as the source code of the project is provided in the GitHub repository.