

A Comparative Study of Solvers for Amazons Endgames

Kloetzer Julien
Iida Hiroyuki
Bouzy Bruno



The goal

- Find the best solver for Amazons endgames (in real-time)
- Solving: finding the exact value (and a possible best move) of a position. Useful for
 - Problem solving
 - Game solving
- Solving can be
 - Two-valued: Win/Lose (Chess and Chess problems, Shogi and Tsumeshogi, Amazons)
 - Multi-valued: Determined by a score (Amazons or Clobber sub-games)

Solving algorithms

DFPN

- Depth-first version of Proof-number Search (PNS): uses proof- and disproof-numbers to solve first in a tree the easiest nodes to prove/disprove

WPNS

- Recent version of PNS whose goal is to better handle the presence of DAGs (Directed Acyclic Graphs) in a game-tree

Game-playing methods

Alpha/Beta

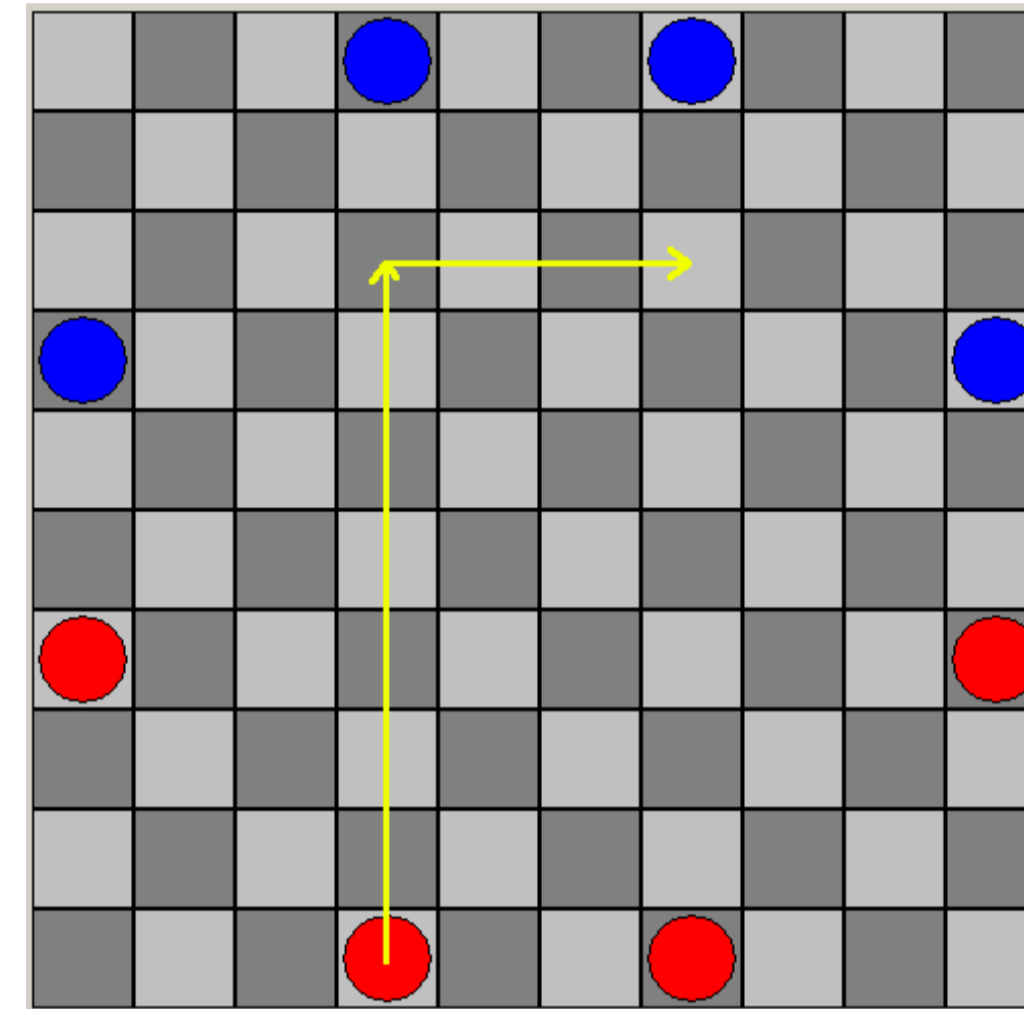
- Standard complement of iterative deepening, used to cut huge parts of a game-tree

Monte-Carlo Tree-Search (MCTS)

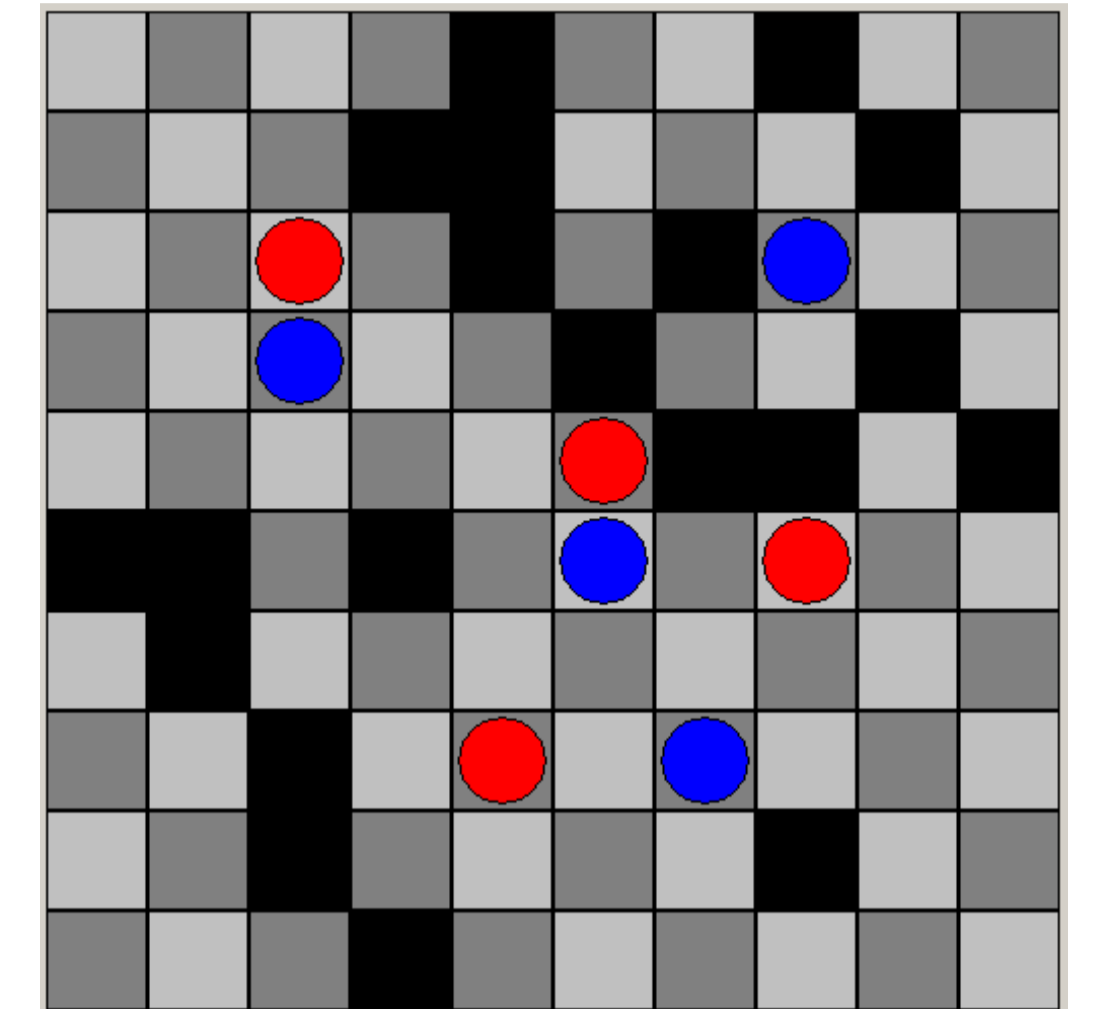
- Statistical method playing moves which maximize their average result. Adding TS to a MC engine allows it to focus its search on best moves and greatly increases its performance

The game of the Amazons

- Created by Walter Zamkuskas (Argentina) in 1988
- Intermediate between Chess (movement) and Go (territory)
- In turn, each player moves an Amazon (like a queen in Chess), then *shoots an arrow* from the landing position in the same way: this square is blocked
- No further shot or move can pass over or land on a blocked square
- The first player unable to move loses the game
- The score of the game is the size of the *territory* of the winner, that is the number of moves he can still play after his opponent has passed



Starting position and a move



A mid-game position

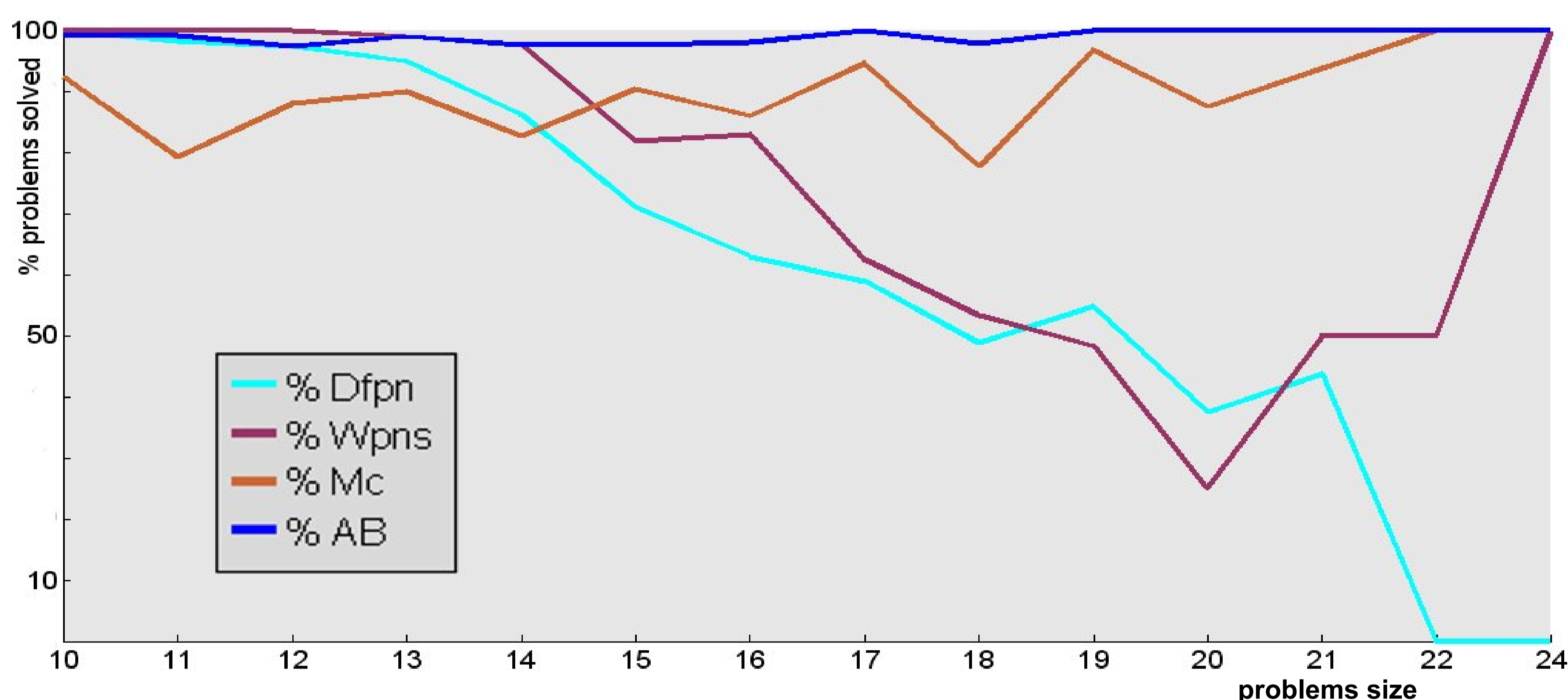
Results

Method	Percentage
DFPN	81.84
WPNS	87.74
Alpha/Beta	98.52
MCTS	87.51

Percentage of problems solved for each assessed method

Method	Percentage unsolved	Percentage badly solved	Average scoring error
DFPN	9,49	6,74	4,97
WPNS	6,16	5,59	5,43
Alpha/Beta	0	1,48	1,15
MCTS	0	12,49	1,67

Statistics on unsolved problems



Problem 1 : Use 2-valued solvers for a multi-valued problem?

DFPN and WPNS are 2-valued solvers; how to adapt?

- Instead of answering to *Is this position winnable?*, we answer to *Can I win this position with at least N points?* (or the reverse question)
- How? With an initial result R to prove
 - Consider leaf nodes with better results than R as YES nodes, others as NO nodes
 - Do an iterative search on all possible results, starting from the worst

Problem 2 : How to evaluate Game-playing methods?

Game-playing methods just give a move, not the exact value of a position; how to compute the value?

- Get a move from the method, then use a judge program (a « real » solver) to compute the exact value of the position given by playing the move in the initial position.
 - If the computed value is equal to the value of the initial position, the move is optimal
 - If it is inferior, the move has failed by <difference between the two evaluations> points

Problem 3 : The solving method's procedure needs more time

Set the limit of time for the game-playing methods to find their move-answer, and give that same period of time to each iteration of the solving method process

Problem 4 : Both classes of methods are not evaluated in the same way

Use the same judge program to evaluate both game-playing methods and solving methods

Results analysis

Considering its "standard" status, the performances of DFPN are disappointing

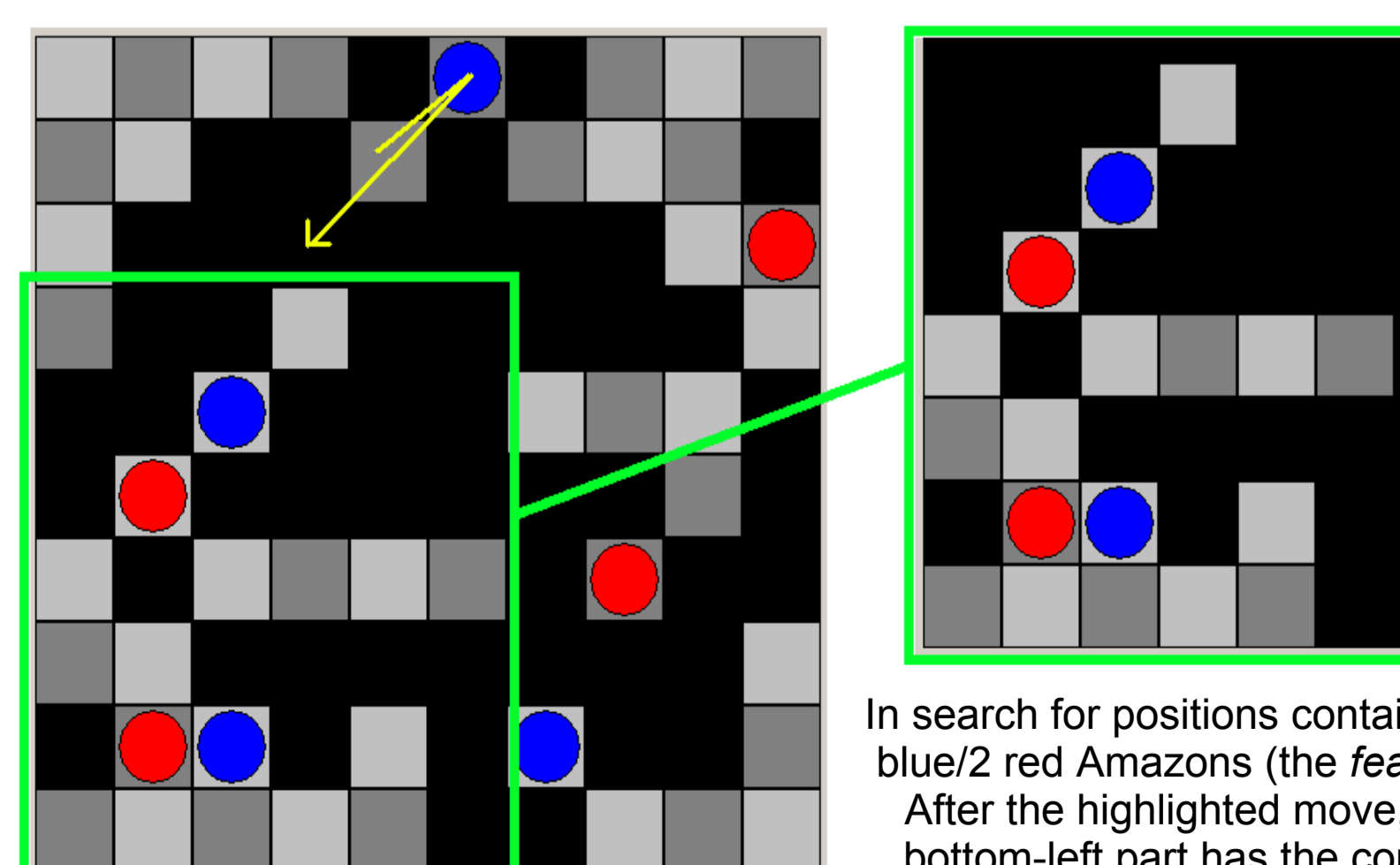
The performance of WPNS are encouraging, but needs further study

Alpha/Beta clearly outperforms every other method, confirming the necessity of an evaluation function for the game of Amazons

MCTS performs surprisingly well, considering it is not at all designed to solve problems

Amazons problems

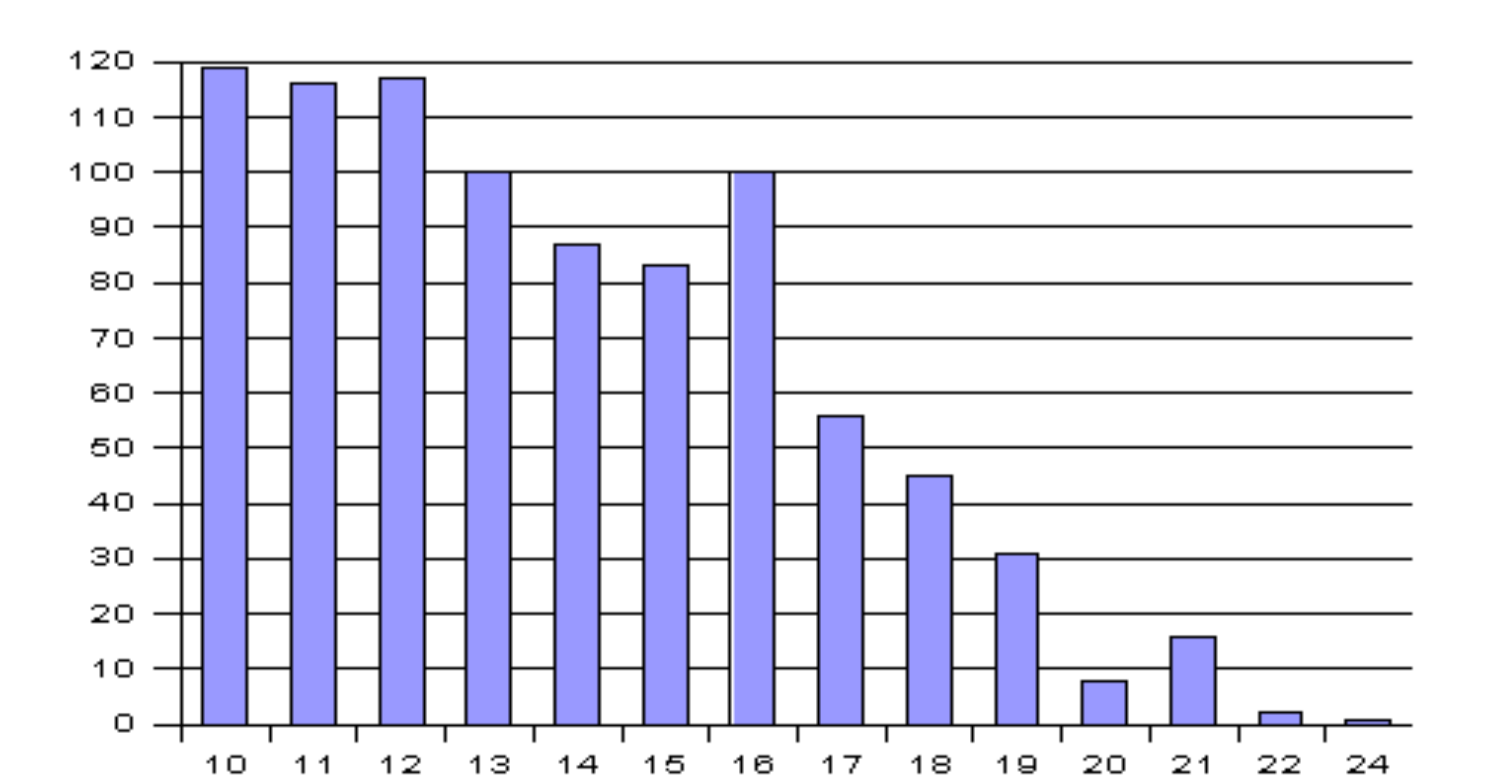
- Unlike other more famous games (Chess, Othello, Shogi...), there is few knowledge about the game of Amazons. If we want a problem database, we will have to build it
- What is an Amazon problem? It is a portion of the board containing Amazons of both players as well as arrows, considered isolated from the rest of the board (edges of the position should be either edges of the board or arrows). Its solution is a move or a set of move maximizing the final *territory* of the winning player
- How to get these sub-positions? By extracting them from game records



In search for positions containing 2 blue/2 red Amazons (the *feature*). After the highlighted move, the bottom-left part has the correct feature: it is copied and stored.

Extraction procedure

- Read a game record until a sub-part of the board is isolated
- Check if this sub-part has a set of pre-determined *features*; if yes, copy the sub-position and store it in the database



Number of extracted problems per size (number of empty squares)

Problems extracted

- Game record: auto-play from our program (Campya) with 3 minutes per side
- Problems that were Too small or too simple were discarded
- ~900 problems generated