3 Games; Alternation; Exhaustive Search; Time vs. Space

3.1 How to Win

In this section we consider a more interesting *provably* intractable problem: playing games with full information, two players and zero sum. We will see that even for some simple games there cannot be a much more efficient algorithm than exhaustive search through all possible configurations.

The rules of an ***n-player game*** G are set by families f, v of ***information*** and ***value*** functions and a ***transition rule*** r. Each player $i \in I$ at each step participates in transforming a configuration (game position) $x \in C$ into the new configuration $r(x, m),\ m : I \to M$ by choosing a move $m_i = m(i)$ based only on his knowledge $f_i(x)$ of x. The game proceeds until a ***terminal*** configurations $t \in T \subset C$ is reached. Then $v_i(t)$ is the loss or gain of the i-th player. Our games will have ***zero sum*** $\sum v_i(t) = 0$ and ***full information:*** $f_i(x) = x$, $r(x, m) = r'(x, m_{a(x)})$, where $a(x)$ points to the ***active*** player. We consider binary, two-players, no-draw games, taking $0 \notin C \subset \mathbb{Z}$, $M \subset \mathbb{Z}$, $T = I = \{\pm 1\}$, $a(x) = \mathbf{sign}(x)$, $v_i(t) = a(t)i$, and $|r(x, m)| < |x|$.[5]

An example of such games is chess. Examples of games without full information are card games, where only a part $f_i(x)$ (player's own hand) of the position x is known. Each player may have a ***strategy*** providing a move for each position. A strategy S is ***winning*** at x if starting at a position x it guarantees victory whatever the opponent does, even if he knows S. We can extend v_1 on T to V on all positions with a winning strategy for one side so that $a(x)V(x) = \sup_m\{a(x)V(r(x, m))\}$. ($\sup\{\}$ taken as -1.)

Evaluating or ***solving*** a game, means computing V. This ability is close to the ability to find a good move in a modified game. Indeed, modify a game G into G' by adding a preliminary stage to it.
At this stage the player A offers a starting position for G and her opponent B chooses which side to play. Then A may either start playing G or decrement the counter of unused positions and offer another one. Obviously, B wins if he can determine the winning side of every position. If he cannot while A can, A wins. Also, any game can be modified into one with two moves: $M \subset \{0, 1\}$ by breaking a string move into several bit-moves. (A position of the new game consists of a position x of the old one and a prefix y of a move. The active player keeps adding bits to y until m is complete and the next position generated by $r(x, m)$.) Evaluating such games is obviously sufficient for choosing the right move.

Theorem. *Each position of any full information game has a winning strategy for one side.*

(This theorem [Neumann, Morgenstern 44] fails for games with ***partial*** information: either player may lose if his strategy is known to the adversary. E.g.: 1. Blackjack (21); 2. Each player picks a bit; their equality determines the winner.) The game can be solved by playing all strategies against each other. There are 2^n positions of length n, $(2^n)^{2^n} = 2^{n \times 2^n}$ strategies and $2^{n \times 2^{n+1}}$ pairs of them. For a 5-bit game that is 2^{320}. The proof of this Theorem gives a much faster (but still exponential time!) strategy.

Proof. Make the graph of all $\leq \|x\|$ -bit positions and moves; Set $V = 0$; reset $V = v$ on T.
Repeat until idle: If $V(x) = 0$, set $V(x) = a(x) \sup_m\{a(x)V(r(x, m))\}$.
The procedure stops with empty $V^{-1}(0)$ since $|r(x, m)| < |x|$ in our games keep decreasing. □

Games may be categorized by the difficulty to compute r. We will consider only r computable in linear space $O(\|x\|)$. Then, the $2^{2\|x\|}$ possible moves can be computed in exponential time, say $2^{3\|x\|}$. The algorithm tries each move in each step. Thus, its total running time is $2^{3\|x\|+1}$: *extremely* slow (2^{313} for a 13-byte game) but still *much* faster than the previous (double exponential) algorithm.

Exercise: the Match Game. Consider 3 boxes with 3 matches each: ! ! ! | ! ! ! | ! ! !.
The players alternate turns taking any *positive* number of matches from a *single* box. One cannot leave the table empty. Use the above algorithm to evaluate all positions and list the evaluations after each its cycle.

Exercise: Modify the chess game by giving one side the right to make (if it chooses to) an extra move out of turn during the first 10 moves. Prove that this side have a non-loosing strategy.

[5] Our examples will assure "$< |x|$" by implicitly prepending non-terminal configurations with a counter of remaining steps.