Automatic Generation of Alternative Starting Positions for Simple Traditional Board Games

Umair Z. Ahmed (IIT Kanpur)
Krishnendu Chatterjee (IST Austria)
Sumit Gulwani (MSR, Redmond)

Simple Traditional Board Games

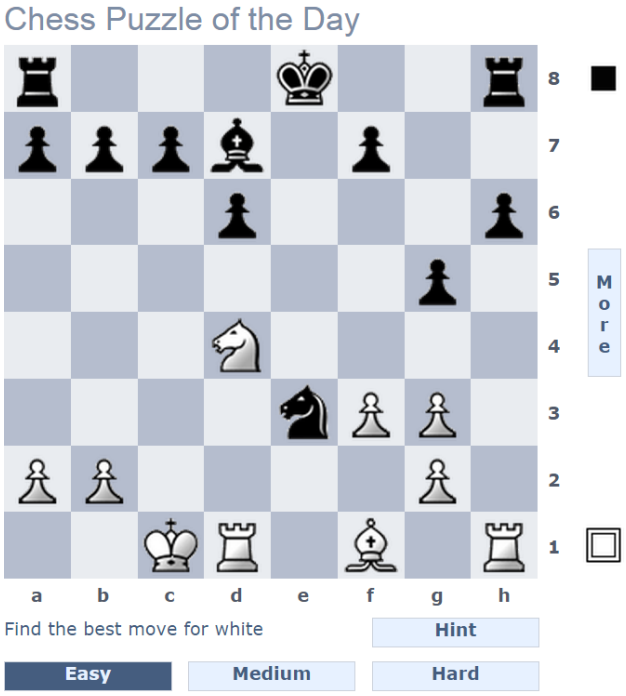
- Board games such as Tic-Tac-Toe and Connect-4
- Default start state is the empty board
- The traditional research question: who is the winner and optimal strategies from the empty starting state

Novel Problem: Alternative Starting Positions

- Customizing hardness level of starting state
  - Default start state can be biased (Tic-Tac-Toe 4x4)
  - Even if unbiased not conducive for novice players
- Generating multiple fresh start states
  - Strategies can be memorized
- Customizing length of play
  - Long plays can be boring.

Alternative Starting Positions

- Automatically generate **interesting** starting states
- Our results:
  - *Theoretical* : A search strategy that applies to all graph games.
  - *Experimental*: For simple board games



Source: *www.shredderchess.com*

Alternating Graph Game

- $G = \big((V, E), (V_1, V_2)\big)$
  - G – finite graph
  - V – vertex set
  - V_1 – player-1 vertices, partition of V
  - V_2 – player-2 vertices, partition of V
  - E – edge set, $E \subseteq \big((V_1 \times V_2) \cup (V_2 \times V_1)\big)$
- T_1- winning/target set for player-1, $T_1 \subseteq V$
- T_2- winning/target set for player-2, $T_2 \subseteq V$

Our Approach

- First, compute W_j from where player 1 can ensure to win in atleast j-steps.

- For positions in W_j, determine the hardness of positions using min-max evaluation.

Two Issues

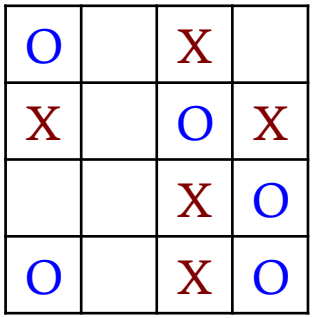
- Game graphs are large: Computing W_j is computational expensive.

- For large depth strategies evaluation using min-max strategies is also computationally expensive.

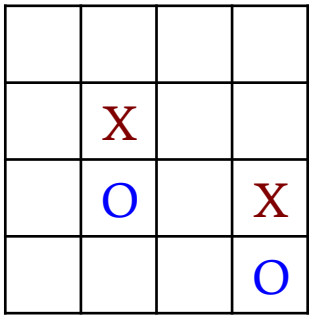
Solution Ideas

1. **Symbolic Methods**: Represent game symbolically using variables and compute W_j symbolically (details in paper)
2. **Iterative Simulation**: Determine hardness using lightweight small-depth strategies (details in paper).

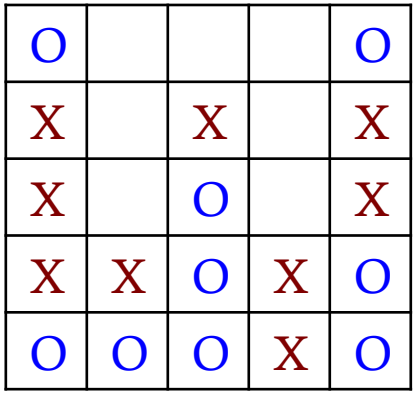
Board Games Variants



Tic-Tac-Toe



Bottom-2



Connect-4

Variable parameters

1. **Board Size**: 3x3, 4x4, 4x5, …
2. **Winning Condition**: RCD, RC, CD, RD
3. **Gravity**: None, Partial, Full

Experimental Results

Connect-4 5x5 with 5000 Random Sampling, against $k_2 = 3$

j	State Space	Win Cond	# States $\lvert W_j \rvert$	$k_1 = 1$			$k_1 = 2$			$k_1 = 3$		
				E	**M**	**H**	**E**	**M**	**H**	**E**	**M**	**H**
2	6.9×10^7	RCD	1.2×10^6	*	184	215	*	141	129	*	0	0
	8.7×10^7	RC	1.6×10^6	*	81	239	*	70	186	*	0	0
	1.0×10^8	RD	1.1×10^6	*	106	285	*	151	82	*	0	0
	9.5×10^7	CD	5.3×10^5	*	364	173	*	209	96	*	0	0
3	6.9×10^7	RCD	2.8×10^5	*	445	832	*	397	506	*	208	211
	8.7×10^7	RC	7.7×10^5	*	328	969	*	340	508	*	111	208
	1.0×10^8	RD	8.0×10^5	*	398	1206	*	464	538	*	179	111
	9.5×10^7	CD	1.5×10^5	*	146	73	*	171	110	*	120	72

Experimental Results

- Existence of vertices of different hardness levels

Game	Size	Player-1 depth		
		$k_1 = 1$	$k_1 = 2$	$k_1 = 3$
Tic-Tac-Toe	3x3	Only RCD		
Tic-Tac-Toe	4x4			
Bottom-2	3x3	Only RD		
Bottom-2	4x4		Except RCD	
CONNECT-3	4x4			
CONNECT-4	5x5			

Experimental Results

- Existence of vertices of different hardness levels
- Number of interesting vertices are rare – negligible fraction of huge state space
- Interesting games now possible in games with heavily biased starting states, like Tic-Tac-Toe 4x4

Auto Generated Starting States

- Positions Player-1 (X) can win in $j = 2$ turns, which are difficult against $k_2 = 3$

O		X	
X		O	X
		X	O
O		X	O

Tic-Tac-Toe
RC, k_1=1

	X		
	O		X
			O

Bottom-2
RCD, k_1=1,2

O				O
X		X		X
X		O		X
X	X	O	X	O
O	O	O	X	O

Connect-4
RCD, k_1=1,2

Conclusion

- Novel problem definition: automatic generation of interesting starting states
- Search technique: Utilizing symbolic methods and iterative simulation
- Present results for simple traditional board games
- Future work: Whether can be extended to more complicated games

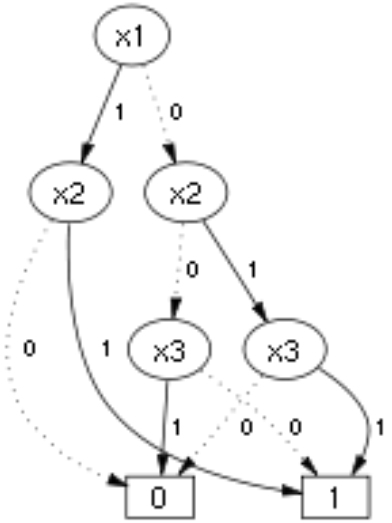
Acknowledgment

- AAAI-15 travel partially supported by
  - AAAI-2015 student scholarship
  - Microsoft Research India travel grant
  - Indian Institute of Technology (IIT) Kanpur

- Contact email: **umair@iitk.ac.in**

Binary Decision Diagrams (BDD)

$$f(x_1, x_2, x_3) = \overline{x_1}.\overline{x_2}.\overline{x_3} + x_1.x_2 + x_2.x_3$$



Binary Decision Diagram

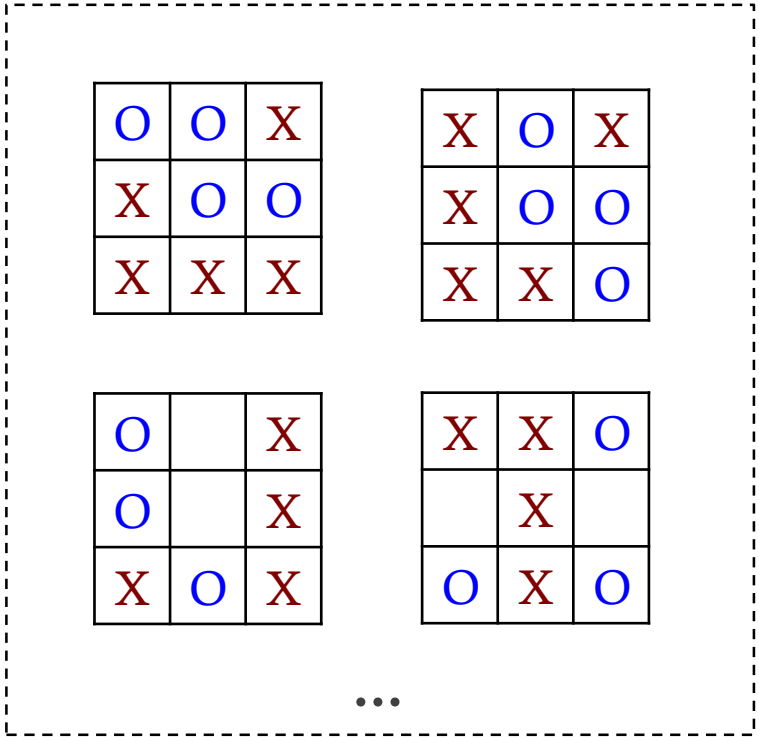
x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Truth Table

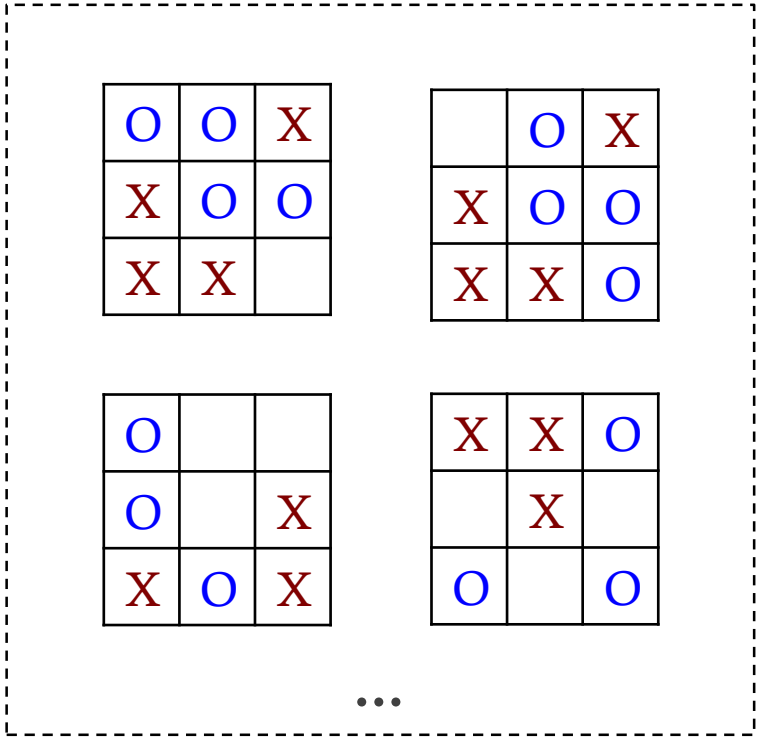
Source: *Wikipedia*

BDD Operations
EPre (Existential Predecessor)

$V = T_1$ (Winning states of Tic-Tac-Toe)

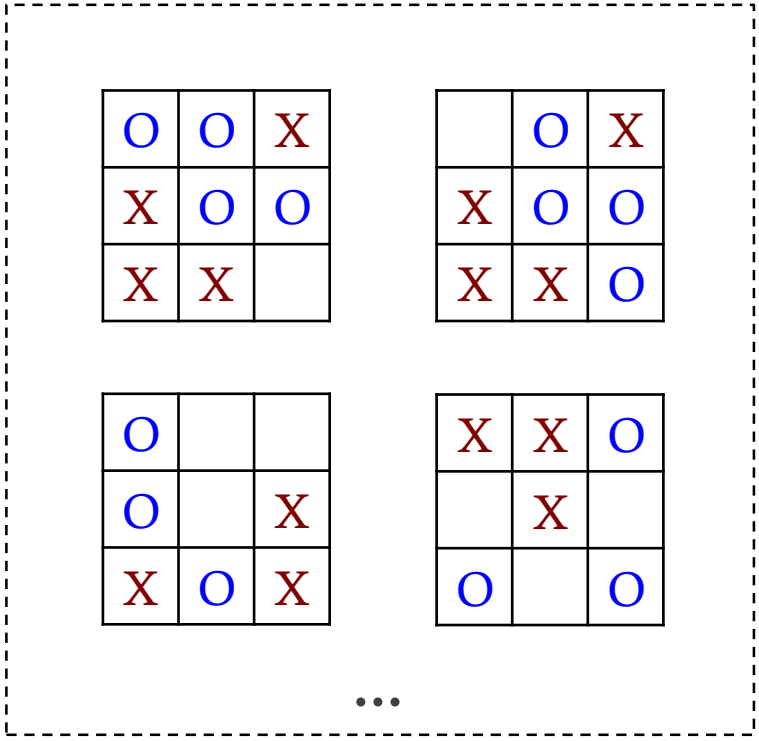


$EPre(V) = \{u \in U; \exists v \in V, (u, v) \in E\}$

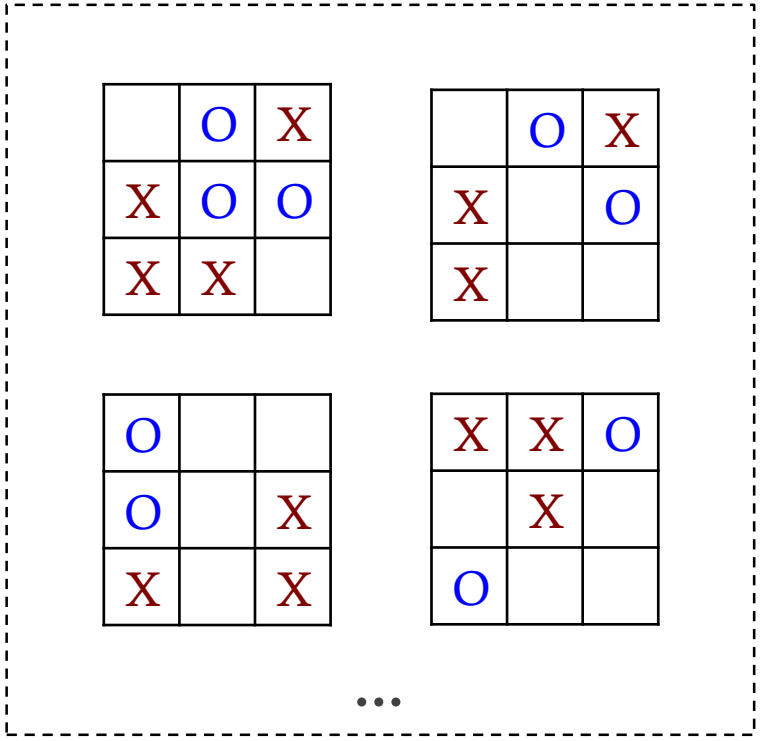


BDD Operations
APre (Universal Predecessor)

$V = EPre(T_1)$



$APre(V) = \{u \in U;$
$\forall (u,v) \in E, v \in V\}$



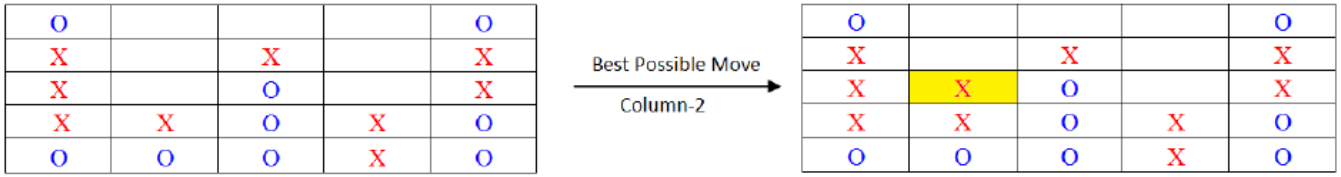
Search Strategy

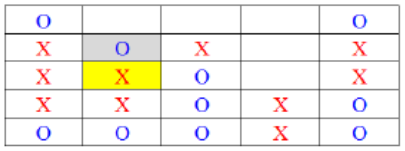
1. **Symbolic Methods**: Represent game symbolically using variables and compute starting states using BDD operations
2. **Iterative Simulation**: Determine hardness (Easy, Medium or Hard) of the starting states, using Min-Max

1. Symbolic Methods

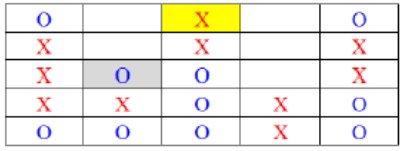
- Obtain Target/Winning set – T_1
- Compute j-steps win set – W_j
  - $W_0 = EPre(T_1)$
  - $W_{i+1} = EPre(APre(W_i))$

2. Iterative Simulation: Min-Max

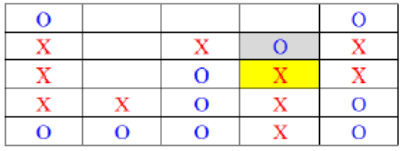




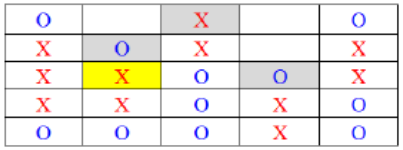
(a) $k_1 = 1$, column-2 fetches Player-X a reward of +5



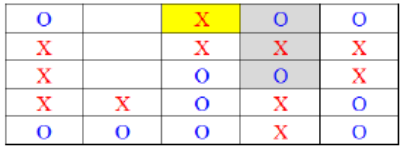
(b) $k_1 = 1$, column-3 fetches Player-X a reward of +6



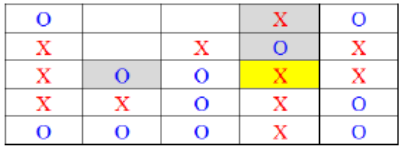
(c) $k_1 = 1$, column-4 fetches Player-X a reward of +2



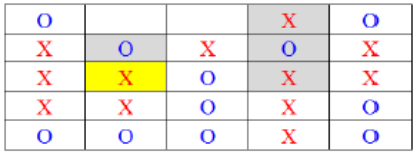
(d) $k_1 = 2$, column-2 fetches Player-X a reward of +3



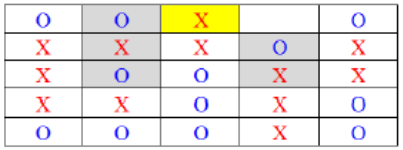
(e) $k_1 = 2$, column-3 fetches Player-X a reward of +6



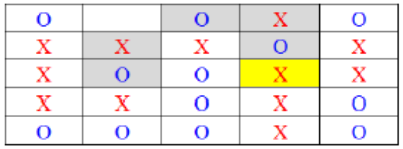
(f) $k_1 = 2$, column-4 fetches Player-X a reward of +0



(g) $k_1 = 3$, column-2 fetches Player-X a reward of $+\infty$



(h) $k_1 = 3$, column-3 fetches Player-X a reward of +0



(i) $k_1 = 3$, column-4 fetches Player-X a reward of +0