Tango Puzzle

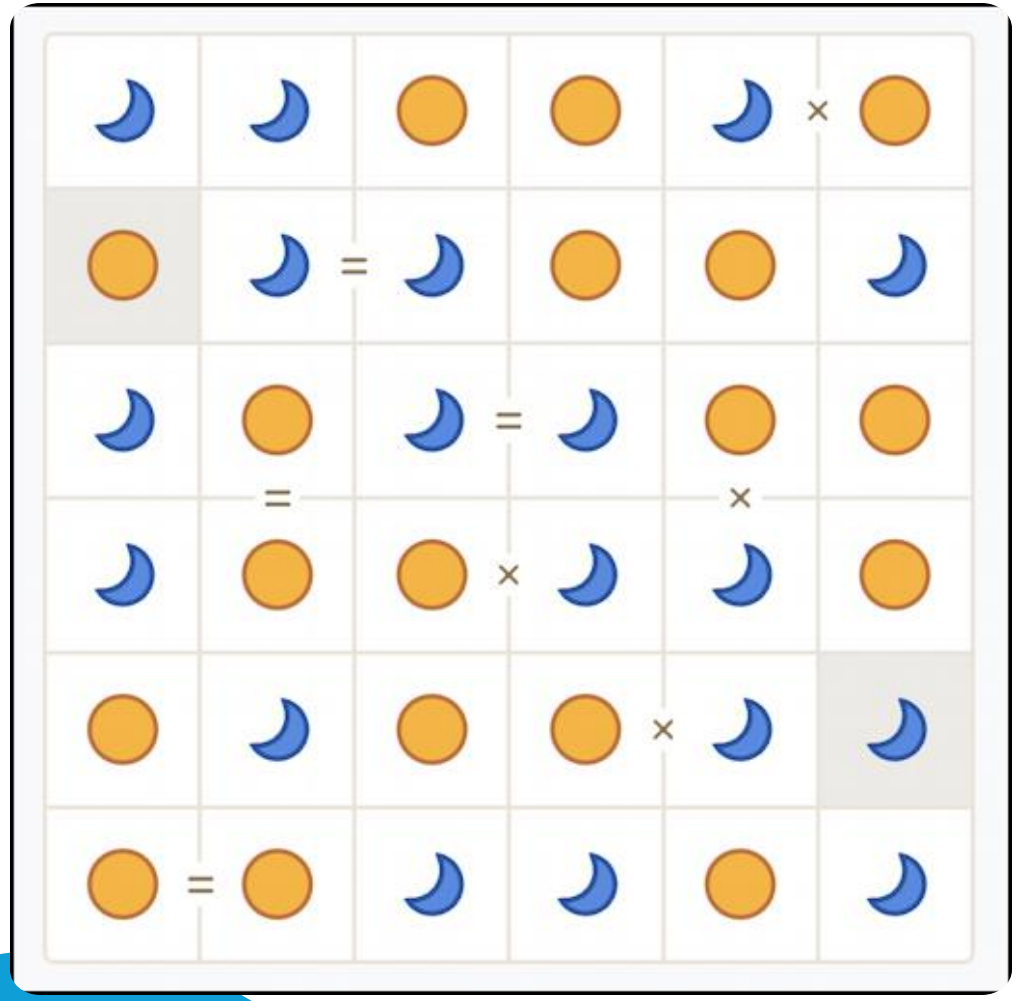
Team members:

Suraj Thapa

Sadiksha Chitrakar

Lal Bahadur Reshmi Thapa

May 04, 2025



Tango

A logic game where players fill a grid with alternating Sun and Moon symbols, ensuring each row and column follows specific placement rules and puzzle constraints.

How to play:

- Fill each cell with either a suns or moons (for instance) can be any two distinct symbols
- No more than 2 of the same symbol may be next to each other, vertically or horizontally
- Each row and column must have an equal number of suns and moons
- Cells separated by = must be the same type
- Cells separated by × must be opposite types
- Each puzzle has one right answer and can be solved via deduction

Existing Solutions

1. QUBO Formulation (Mathematical Approach)

- **Strategy**: Convert constraints into a quadratic energy function
- **Tool**: Solved via quantum-inspired or classical solvers (e.g., Quadratic Unconstrained Binary Optimization optimizers)
- **Strengths**:
  - Precise mathematical representation
  - Suitable for combinatorial optimization frameworks
- **Limitations**:
  - Hard to scale with dynamic or large grids
  - Lacks interactivity or step-by-step visualization
  - Requires specialized solvers

Existing Solutions

2. Classical Backtracking Approach

- **Strategy**: Try filling grid recursively; backtrack on invalid moves
- **Strengths**:
  - Simple and complete
  - Easy to implement for smaller grids
- **Limitations**:
  - Very slow for large or complex boards
  - No learning or optimization

Project Goals

Optimal Puzzle Solver Using A*

Develop and demonstrate an A* search-based solver that uses admissible heuristics to find the optimal solution efficiently.

Early Failure Detection with Arc Consistency

Implement arc consistency techniques to prune invalid choices early in the solving process, reducing computation time and enhancing performance.

Reinforcement Learning Agent with Q-Learning

Train a Q-learning agent to learn optimal symbol placement policies (Sun or Moon) by maximizing the expected cumulative reward across the grid.

A* Solver Approach

- A graph-based optimal pathfinding algorithm which selects paths that minimize total cost:

  $f(n) = g(n) + h(n)$, $g(n)$ = cost so far (number of steps taken)

  $h(n)$ = estimated cost to goal (heuristic)

- **Heuristic Used in Our Project**
  - Total constraint violations =
    - Triple violations (three same symbols in row/col)
    - Unequal sun/moon counts
    - Rule constraint mismatches (=, x)
  - Fewer violations → Lower heuristic → Better move

- **Why A* Works Well Here**
  - Prioritizes promising grid states
  - Efficiently avoids invalid or dead-end paths
  - Solves the puzzle optimally and quickly

AC-3 Solver Approach

- A constraint propagation algorithm used in CSPs that ensures every value in a cell's domain has a valid value in connected cells
- Eliminates impossible choices before making guesses

- **How It Works in Our Tango Game**
  - Each empty cell starts with domain: {sun, moon}
  - AC-3 removes values that lead to:
    - Triple violations
    - Unbalanced rows/columns
    - Constraint mismatches (=, x)

- **Backtracks Only When Needed**
  - If AC-3 can't find consistent assignments, backtracking begins
  - During backtracking:
    - Assign → Propagate → Undo if inconsistent
  - Reduces guesswork and speeds up solving

Q-Learning Approach

- A Reinforcement Learning algorithm where agent learns to play the game by trial and error
- Learns an optimal policy without prior knowledge of the environment
- **How It Works in Our Tango Game**
  - The game grid is the environment
  - Each cell assignment is an action
  - Agent selects moves based on Q-values (state-action values)
- **How the Agent Learns**
  - Solves the puzzle cell by cell, from top-left to bottom-right
  - At each step, assigns sun or moon to the next unfilled cell
  - Uses ε-greedy exploration to balance between exploring and exploiting
- **Reward System**
  - +100 for solving the puzzle
  - -5 for constraint violations
  - +1 for each valid move
- **Learning Over Time**
  - Updates Q-values after each move
  - Learns better strategies through trial-and-error
  - Eventually solves the puzzle consistently and intelligently

How Our Work Differs

Feature	Existing Solvers	Our Implementation
Solving Strategy	Mathematical optimization (QUBO), classical backtracking	Search-based (A*), CSP-based (AC-3), Learning-based (Q-learning)
Constraint Handling	Hard-coded or symbolic constraints	Constraint propagation and dynamic checking
Adaptability	Static logic, no learning	Q-learning adapts via trial-and-error
Backtracking Mechanism	Full puzzle reset on failure	AC-3 uses intelligent, partial backtracking
Modular Codebase	Often monolithic or fixed	Fully modular (grid, constraints, solvers separated)
Autonomous Learning Agent	Not applicable	Yes – Q-learning agent improves over time

Challenges and Solutions

Challenge	Solution
Encoding complex Tango constraints in Reinforcement Learning	Manually incorporated constraint checks during training and inference
Q-learning agent getting stuck in invalid states	Introduced undo mechanism with penalty (-5) and retried until solved
Maintaining visual consistency across solvers	Designed unified grid structure and reused common drawing logic
Keeping code maintainable and extendable	Followed modular architecture separating logic, UI, and solver engines

Demonstration of the project

1. A* Algorithm → Lal Bahadur Reshmi Thapa
2. AC3 Algorithm → Sadiksha Chitrakar
3. Q-Learning → Suraj Thapa

Conclusions

- Developed three AI-based solvers for the Tango Puzzle: A* Search, AC-3 with backtracking, and Q-learning
- AI methods demonstrated better adaptability than classical backtracking by incorporating heuristics, constraint propagation, and learning from experience

Future Work

- Enhance Q-learning with Deep Q-Networks (DQN)
  - Improve learning efficiency and generalization for larger state spaces
- Scale to larger grids (e.g., 8×8, 10×10)
  - Adapt constraint checking and solver logic for higher complexity
- Explore hybrid approaches
  - Combine constraint propagation with reinforcement learning for more robust solving
- Add difficulty levels and puzzle generator
  - Dynamically generate puzzles with adjustable complexity

References

[1] Mata Ali, Alejandro & Mencia, Edgar. (2024). A QUBO Formulation for the Generalized Takuzu/LinkedIn Tango Game.

[2] Foster, J. (2025, February 16). *It only takes one to Tango - Jeff Foster - Medium*. Medium. https://jeff-foster.medium.com/it-only-takes-one-to-tango-92665433017d

[3] *Russell, Stuart J. (Stuart Jonathan), 1962-. (2010). Artificial intelligence : a modern approach. Upper Saddle River, N.J. :Prentice Hall,*

[4] https://github.com/fffej/tango

Thank you!