AI Development Plan: Medium Difficulty

🎯 Goal

Create a Medium AI that makes strategic decisions instead of random choices.

📊 Current State (Easy AI)

Strategy:

```javascript
// Easy AI: Random selection
_chooseRandomMove(possibleMoves) {
  const randomIndex = Math.floor(Math.random() * possibleMoves.length);
  return possibleMoves[randomIndex];
}
```

Characteristics:

- ✅ Fast (instant decision)
- ✅ Valid moves only
- ❌ No strategy
- ❌ No territory evaluation
- ❌ No opponent awareness

🎮 Medium AI Strategy

Core Principles:

1. **Territory Expansion**
   - Prefer moves that capture more cells
   - Prioritize contour captures
   - Maximize controlled area
2. **Position Evaluation**
   - Corner positions = High value

- Edge positions = Medium value
- Center positions = Low value (easier to surround)

3. **Opponent Blocking**
   - Identify opponent's large territories
   - Try to prevent opponent's contour captures
   - Cut opponent's expansion paths
4. **Risk Assessment**
   - Avoid creating pockets that opponent can surround
   - Prefer connected territories over scattered cells

Scoring System

Move Score Components:

```javascript
moveScore =
  cellsGained * 10 +          // Direct territory gain
  contourCapture * 50 +       // Captured cells via contour
  cornerBonus * 20 +          // Positioning value
  connectionBonus * 15 +      // Territory connectivity
  opponentBlockBonus * 25 -   // Blocking opponent expansion
  riskPenalty * 30            // Risk of being surrounded
```

Detailed Scoring:

1. **Cells Gained (Direct)**

```
Score = rectangleArea * 10
Example: 3×4 rectangle = 12 cells = 120 points
```

2. **Contour Capture (Predicted)**

```
Score = estimatedCapturedCells * 50
Example: Closing contour = 20 cells = 1000 points
```

3. **Corner Bonus**

If placement touches corner: +20 points per corner touched
Corners = harder to surround, more defensible

4. **Connection Bonus**

Score = connectedSides * 15
More connected sides = stronger territory
Example: 3 sides connected = 45 points

5. **Opponent Block Bonus**

If placement blocks opponent's expansion: +25 points
If prevents opponent's contour: +50 points

6. **Risk Penalty**

If creates isolated pocket: -30 points
If easily surroundable: -20 points

🏗️ Implementation Plan

Phase 1: Basic Evaluation

1. ✅ Calculate direct cell gain
2. ✅ Identify corner positions
3. ✅ Count connected sides
4. ✅ Select highest scoring move

Phase 2: Advanced Evaluation

1. ⏳ Predict contour captures
2. ⏳ Detect opponent threats
3. ⏳ Calculate risk factors

Phase 3: Optimization

1. ⏳ Add look-ahead (1 move)
2. ⏳ Improve performance
3. ⏳ Fine-tune scoring weights

💻 Code Structure

New Methods for Medium AI:

```javascript
```

```
/**
 * Evaluate a single move (Medium AI)
 */
_evaluateMove(move, gameState) {
  let score = 0;

  // 1. Direct cell gain
  score += this._scoreCellGain(move);

  // 2. Position value (corner/edge)
  score += this._scorePosition(move, gameState);

  // 3. Territory connection
  score += this._scoreConnection(move, gameState);

  // 4. Contour capture prediction (Phase 2)
  score += this._scoreContourPotential(move, gameState);

  // 5. Opponent blocking (Phase 2)
  score += this._scoreOpponentBlock(move, gameState);

  return score;
}

/**
 * Choose best move based on evaluation
 */
_chooseBestMove(possibleMoves, gameState) {
  let bestMove = null;
  let bestScore = -Infinity;

  for (let move of possibleMoves) {
    const score = this._evaluateMove(move, gameState);
    move.score = score;

    if (score > bestScore) {
      bestScore = score;
      bestMove = move;
    }
  }

  console.log(`AI Medium: Best move score=${bestScore} at (${bestMove.x},${bestMove.y})`);
  return bestMove;
}
```

📐 Evaluation Functions Detail

1. Cell Gain Scoring

javascript

```
_scoreCellGain(move) {
  const { width, height } = this._getDimensions(move);
  return width * height * 10;
}
```

2. Position Scoring

javascript

```
_scorePosition(move, gameState) {
  let score = 0;
  const { x, y, width, height } = move;
  const { width: gridWidth, height: gridHeight } = gameState;

  // Corner detection (any corner of rectangle touches grid corner)
  const corners = [
    {x: 0, y: 0},                    // Top-left
    {x: gridWidth - 1, y: 0},        // Top-right
    {x: 0, y: gridHeight - 1},       // Bottom-left
    {x: gridWidth - 1, y: gridHeight - 1}   // Bottom-right
  ];

  for (let corner of corners) {
    if (this._rectangleTouchesPoint(x, y, width, height, corner.x, corner.y)) {
      score += 20;
    }
  }

  // Edge bonus (less than corner, but still valuable)
  if (x === 0 || y === 0 || x + width === gridWidth || y + height === gridHeight) {
    score += 10;
  }

  return score;
}
```

3. Connection Scoring

javascript

```
_scoreConnection(move, gameState) {
  const { x, y, width, height } = move;
  const { grid, currentPlayer } = gameState;

  let connectedSides = 0;

  // Check 4 sides of rectangle for existing territory
  // Top side
  if (y > 0 && this._hasPlayerCellsInRange(grid, x, y - 1, width, 1, currentPlayer)) {
    connectedSides++;
  }

  // Bottom side
  if (y + height < gameState.height && this._hasPlayerCellsInRange(grid, x, y + height, width, 1, currentPlayer)) {
    connectedSides++;
  }

  // Left side
  if (x > 0 && this._hasPlayerCellsInRange(grid, x - 1, y, 1, height, currentPlayer)) {
    connectedSides++;
  }

  // Right side
  if (x + width < gameState.width && this._hasPlayerCellsInRange(grid, x + width, y, 1, height, currentPlayer)) {
    connectedSides++;
  }

  return connectedSides * 15;
}
```

🎲 Difficulty Comparison

Feature	Easy	Medium	Hard (Future)
Decision	Random	Scored	Minimax
Look-ahead	0 moves	0 moves	1-2 moves
Territory eval	❌ No	✅ Yes	✅ Advanced
Contour predict	❌ No	⏳ Basic	✅ Advanced
Opponent aware	❌ No	⏳ Basic	✅ Advanced

Feature	Easy	Medium	Hard (Future)
Speed	Instant	<100ms	<500ms
Win rate vs Easy	50%	~70%	~85%

🧪 Testing Plan

Test Scenarios:

1. **Corner Control Test**

> Grid: 10×10
> Turn 1: AI gets [3,3]
> Expected: Place in corner (high score)

2. **Territory Expansion Test**

> Grid: 10×10
> Turn 5: AI has 20 cells
> Expected: Maximize connected territory

3. **Contour Setup Test**

> Grid: 10×10
> Turn 10: AI can close contour
> Expected: Prioritize contour-closing move

4. **Avoid Isolation Test**

> Grid: 10×10
> Turn 8: AI has scattered cells
> Expected: Connect territories, not create new islands

📊 Performance Targets

Medium AI Goals:

- **Decision Time:** <100ms per move
- **Memory Usage:** <5MB additional
- **Win Rate vs Easy:** 65-75%
- **Moves Quality:** Strategic > Random

Benchmarks:

Grid Size	Positions Checked	Evaluation Time
10×10	~150	~50ms
20×20	~350	~80ms
50×50	~2000	~150ms

🔧 Implementation Checklist

Phase 1 (Basic Medium AI):

- [ ] Create _evaluateMove() method
- [ ] Create _scoreCellGain() method
- [ ] Create _scorePosition() method
- [ ] Create _scoreConnection() method
- [ ] Create _chooseBestMove() method
- [ ] Modify choosePlacement() to use difficulty
- [ ] Add helper methods for geometry
- [ ] Test basic scoring

Phase 2 (Advanced Features):

- [ ] Add _scoreContourPotential()
- [ ] Add _scoreOpponentBlock()
- [ ] Add risk assessment
- [ ] Fine-tune score weights
- [ ] Performance optimization

Phase 3 (Polish):

- [ ] Add logging for move scores
- [ ] Add debug visualization
- [ ] Balance scoring weights
- [ ] Documentation

🎯 Success Criteria

Medium AI is considered **successful** if:

1. ✅ Consistently beats Easy AI (>60% win rate)
2. ✅ Makes visibly better strategic choices
3. ✅ Prioritizes corners and edges
4. ✅ Connects territories intelligently
5. ✅ Decision time <100ms on 10×10 grid
6. ✅ No crashes or errors
7. ✅ Code is maintainable and documented

🚀 Next Steps

1. **Start with Phase 1** - Basic scoring system
2. **Test thoroughly** - Verify scoring works correctly
3. **Iterate** - Adjust weights based on testing
4. **Add Phase 2** - Advanced features
5. **Polish** - Optimize and document

💡 Design Notes

Why This Approach?

Greedy Evaluation (Medium) vs **Minimax (Hard)**:

- Medium: Evaluate current move only (greedy)
- Faster, simpler, still strategic
- Good for casual play
- Foundation for Hard AI

Scoring-based vs **Rule-based**:

- Flexible: Easy to tune weights
- Extensible: Easy to add new factors
- Debuggable: Can see exact scores

Ready to implement Phase 1! 🎮