Pyramid Poker Heuristic Validation Implementation Plan

🎯 Project Overview

Goal: Build and validate a heuristic system that predicts optimal hand arrangements without running full `findBestSetup()` optimization.

Target: Match consensus decisions between Points/Tiered2 methods (NetEV often outlier due to edge case bugs)

Key Insight: Focus on "multiple strong hands" scenarios where arrangement strategy matters most.

📋 Implementation Phases

Phase 1: Core Heuristic Framework

Files to Create:

- `hand-strength-heuristic.js` - Main heuristic calculation
- `heuristic-validator.js` - Testing and comparison framework
- `consensus-analyzer.js` - Multi-method agreement analysis

Core Functions:

```
javascript
```

```
// Main heuristic predictor
function heuristicPredictor(player1Cards, player2Cards, gameVariant) {
  return {
    predictedWinner: 'player1' | 'player2',
    confidenceLevel: 0.0-1.0,
    strengthDifference: number,
    reasoning: "why this prediction"
  };
}

// 17-card strength estimator
function estimate17CardStrength(cards, gameVariant) {
  return {
    premiumHandCount: number,
    strongHandCount: number,
    wildCardFlexibility: number,
    overallScore: number
  };
}
```

Phase 2: Multi-Method Testing Framework

Validation System:

```
javascript
```

```javascript
function comprehensiveValidation(testCases = 10000) {
    const results = {
        consensus: [], // Cases where 2+ methods agree
        outliers: [],  // Cases where NetEV disagrees
        perfect: [],   // Cases where all 3 + heuristic agree
        bugs: []       // Suspected bug cases
    };

    for (let i = 0; i < testCases; i++) {
        // Generate test case
        const p1Cards = deal17Cards();
        const p2Cards = deal17Cards();

        // Run all methods
        const heuristic = heuristicPredictor(p1Cards, p2Cards);
        const points = actualBattle(p1Cards, p2Cards, 'points');
        const tiered2 = actualBattle(p1Cards, p2Cards, 'tiered2');
        const netEV = actualBattle(p1Cards, p2Cards, 'netEV');

        // Analyze consensus
        const analysis = analyzeMethodAgreement(points, tiered2, netEV);

        // Categorize result
        if (analysis.perfectConsensus) {
            results.perfect.push({...});
        } else if (analysis.netEVOutlier) {
            results.outliers.push({...});
        } else if (analysis.suspectedBug) {
            results.bugs.push({...});
        } else {
            results.consensus.push({...});
        }
    }

    return results;
}
```

Phase 3: Heuristic Development Strategy

3.1 Basic Strength Assessment

```
javascript
```

```
function basicHandStrength(cards, gameVariant) {
    const analysis = analyzeCards(cards);

    return {
        // Premium hands (4K, SF, 7K, 8K)
        premiumPotential: countPremiumHands(analysis),

        // Strong hands (FH, Flush, Straight, 3K)
        strongPotential: countStrongHands(analysis),

        // Pair strength and distribution
        pairStrength: evaluatePairs(analysis),

        // Wild card flexibility
        wildOptions: assessWildCardPotential(analysis),

        // Overall difficulty assessment
        arrangementComplexity: estimateComplexity(analysis)
    };
}
```

3.2 Multi-Strong-Hand Strategy

```javascript
function multiStrongHandHeuristic(cardAnalysis) {
    // Key insight: Multiple strong hands = arrangement strategy critical

    if (cardAnalysis.premiumPotential >= 2) {
        return "PREMIUM_MULTIPLE"; // Easy case - spread the wealth
    }

    if (cardAnalysis.strongPotential >= 3) {
        return "STRONG_MULTIPLE"; // Medium difficulty - strategic placement
    }

    if (cardAnalysis.premiumPotential === 1 && cardAnalysis.strongPotential >= 2) {
        return "MIXED_STRONG"; // Hard case - premium placement critical
    }

    return "STANDARD"; // Regular optimization
}
```

3.3 Consensus Target Strategy

```javascript
function consensusTargeting() {
  // Target: Match Points + Tiered2 consensus (ignore NetEV outliers)
  // Rationale: NetEV has edge case bugs, Points/Tiered2 more reliable

  const targetMethod = "CONSENSUS_POINTS_TIERED2";
  const ignoreMethod = "NETEV_OUTLIERS";

  return {
    successMetric: "80%+ accuracy vs Points/Tiered2 consensus",
    debugTarget: "Cases where Points/Tiered2 disagree",
    bugDiscovery: "Cases where all 3 methods disagree"
  };
}
```

🎯 Key Focus Areas

1. Multi-Strong-Hand Scenarios

Challenge: "When you have 2+ strong hands, how do you arrange them?"

- Premium hand in front vs back position?
- Spread multiple good hands vs concentrate power?
- Wild card allocation for maximum impact?

Testing Priority: Generate test cases with multiple premiums/strong hands

2. Wild Card Complexity

Challenge: Wild cards create exponential arrangement possibilities

- Which position benefits most from wild card?
- Make new premium vs improve existing hand?
- Risk assessment with wild flexibility?

Heuristic Approach: Pre-calculate wild card "value add" for different scenarios

3. Edge Case Discovery

Benefit: Find and fix bugs in existing optimization methods

- Systematic disagreement analysis
- Pattern recognition in outlier cases
- Validation through consensus building

📊 Success Metrics

Primary Metrics

1. **Consensus Accuracy:** 80%+ match with Points/Tiered2 agreement
2. **Confidence Correlation:** High confidence predictions more accurate
3. **Category Performance:**
    - Easy cases: 95%+ accuracy
    - Medium cases: 75%+ accuracy
    - Hard cases: 60%+ accuracy

Secondary Benefits

1. **Bug Discovery:** Identify optimization method edge cases
2. **Strategy Insights:** Learn what makes arrangement decisions difficult
3. **Monte Carlo Foundation:** Fast heuristic enables massive simulation

---

🛠️ Implementation Steps

Step 1: Framework Setup (1 session)

- Create core heuristic structure
- Build multi-method testing framework
- Set up result analysis and logging

Step 2: Basic Heuristic (2 sessions)

- Implement card analysis functions
- Create simple strength scoring
- Test against obvious cases (premium vs weak hands)

Step 3: Refinement Cycle (3-5 sessions)

- Run 1,000 test cases, analyze failures
- Adjust heuristic weights and logic
- Focus on multi-strong-hand scenarios
- Iterate based on consensus matching

Step 4: Edge Case Analysis (1-2 sessions)

- Deep dive into method disagreement cases
- Bug discovery and reporting
- Wild card scenario optimization

Step 5: Monte Carlo Integration (Future)

- Replace `findBestSetup()` with heuristic in Monte Carlo
- Generate fresh win probabilities for game variants
- Validate new probability system

🎲 Expected Outcomes

Immediate Value:

- Fast hand strength estimation without full optimization
- Bug discovery in existing optimization methods
- Better understanding of arrangement strategy

Long-term Value:

- Foundation for game variant probability calculations
- Monte Carlo simulation enablement
- Machine learning training data generation

Risk Mitigation:

- Start with simple cases, build complexity gradually
- Use consensus targeting to avoid single-method bias
- Iterative development with constant validation