Pyramid Poker Optimization Methods - Complete Guide

Method Evolution Timeline

Points → Empirical → Tiered1 → Tiered2 → NetEV

Each generation builds on and fixes problems from the previous version.

Method Descriptions

1. **Points (Original Foundation)**

- **Methodology:** Pure point maximization
- **Assumption:** 100% win rate on every hand
- **Data Source:** Raw hand strength calculations
- **Purpose:** Baseline optimization without win probability considerations
- **Key Insight:** Sometimes simple point maximization works well when fundamental card evaluation is sound

2. **Empirical**

- **Methodology:** Real win probability lookup
- **Data Source:** 10,000 rounds from Points method (6,000 hands total)
- **Problems:**
  - Sparse/missing data points (low sample sizes)
  - Some inferior hands had higher win probability than better hands
  - Required exact tuple matches in lookup table
- **Key Issue:** Missing data caused suboptimal arrangements (e.g., couldn't find 4K front match, fell back to trips)

3. **Tiered (Tiered1)**

- **Methodology:** Hierarchy-respecting Pure EV
- **Data Source:** Empirical's win probabilities
- **Improvement:** Fixed hand type hierarchy - never lets lower hand be valued higher than superior hand
- **Foundation for:** Tiered2 development

4. Tiered2

- **Methodology:** Refined Pure EV with edge case fixes
- **Data Source:** Tiered1 probabilities, rebuilt lookup tables
- **Improvements:**
  - Fixed bugs from Tiered1
  - Better handling of missing hands using 2-3 tuple elements
  - Based on 11,000 round dataset
- **Focus:** Pure EV (only counts winning scenarios)

5. NetEV

- **Methodology:** True Expected Value including loss penalties
- **Formula:** `(Win Rate × Points) - (Loss Probability × Loss Points)`
- **Data Source:** 1,463-row lookup table (real + extrapolated data)
- **Key Innovation:** Accounts for catastrophic loss scenarios that pure EV methods ignore
- **Philosophy:** Risk-adjusted optimization

Score Interpretation Guidelines

❌ Don't Compare Absolute Scores Between Methods

Different methods use different win rate assumptions:

- **Points:** Assumes 100% win rate → inflated scores
- **Empirical/Tiered/Tiered2:** Use realistic win probabilities
- **NetEV:** Includes loss penalties → more conservative scores

✅ Focus on Arrangement Differences

- **Which method finds superior card allocations?**
- **Do arrangements make strategic sense?**
- **Are there clear bugs or suboptimal choices?**

✅ Head-to-Head Analysis

When methods choose different arrangements, calculate:

- **Position-by-position comparison**

- **Total point potential**
- **Risk/vulnerability assessment**

Example:

```
Method A: Back(11) + Middle(12) + Front(15) = 38 total
Method B: Back(8) + Middle(16) + Front(18) = 42 total
→ Method B wins head-to-head by 4 points
```

Key Performance Patterns

Strong Hands (Test Cases 1-2)

- **All methods converge** to similar arrangements
- **Clear optimal plays** exist
- **NetEV shows mathematical ceiling** around 35.72

Weak/Normal Hands (Test Cases 1001-1003)

- **NetEV shows lower scores** (risk-adjusted thinking)
- **Tiered methods show higher scores** (pure point maximization)
- **Different optimization objectives** causing score divergence

Wild Card Scenarios (Test Cases 2001+)

- **NetEV performs better** with wild cards
- **Correctly values flexibility** in risk/reward calculations
- **More sophisticated** at evaluating complex scenarios

Method Selection Guidelines

For Game Play Optimization:

- **Use Tiered/Tiered2** for maximum point scoring
- **Players want to maximize points** on every hand
- **Even bad hands should get best possible arrangement**

For Mathematical Validation:

- **Use NetEV** for risk-adjusted analysis

- **Accounts for loss scenarios** ignored by other methods
- **More conservative but mathematically complete**

For Debugging/Analysis:

- **Use Points** as baseline reference
- **Use Empirical** to identify lookup table gaps
- **Compare all methods** to spot anomalies

Common Issues & Debugging

Empirical Problems:

- **Missing exact tuple matches** → falls back to suboptimal choices
- **Sparse data** → unreliable win probabilities
- **Can't evaluate premium hand combinations** that rarely occur in training data

NetEV Anomalies:

- **May choose inferior arrangements** due to lookup gaps or bugs
- **Over-conservative** choices when loss penalties dominate
- **Edge cases** where risk calculation fails

Investigation Process:

1. **Identify anomalous behavior** in test results
2. **Enable logging** for specific test case
3. **Trace decision path** → lookup hits, fallbacks, calculations
4. **Compare across all methods** to isolate the problem
5. **Fix root cause** and verify across test suite

Critical NetEV Bug Fix (Session: Aug 20, 2025)

🚨 Major Bug Found and Fixed:

NetEV was getting incorrect point values due to lookup table flaw:

The Problem:

- **NetEV formula:** `(Win Rate × Points) - (Loss Probability × Loss Points)`

- **Win rates were correct** (~0.95 for 5K)
- **Point lookup was broken** (returning 1 instead of 6)
- **Result:** 5K getting 0.95 EV instead of 5.7 EV

Root Cause:

- **NetEV called:** `get_point_value(hand_type_name, position)` with strings
- **Method expected:** numeric `hand_type_code`
- **Lookup failed** → defaulted to 1 point for all hands

The Fix Applied:

1. **Updated point lookup method** to use numeric handType codes (1-16)
2. **Fixed two callers** to pass `hand_rank_tuple[0]` instead of string names
3. **Complete mapping created** for all handTypes 1-16 across all positions

Expected Impact:

- **NetEV should now make proper strategic decisions**
- **Premium hands get correct point values** (5K = 6 points, not 1)
- **Arrangement choices should improve dramatically**

Next Session Plan:

Re-run all 30 test cases with fixed NetEV to see:

- **How arrangements change** with correct point calculations
- **Whether NetEV now makes sensible strategic choices**
- **If the "flawed lookup table" issues are resolved**
- **True performance comparison** between all methods

Test Case Logging

All methods support detailed logging for systematic investigation:

- **Turn on logging** for specific test cases
- **Trace exact decision paths** and calculations
- **Compare lookup vs fallback usage**
- **Identify where logic diverges** from expected behavior

Perfect for debugging anomalies like NetEV choosing flush over straight flush in front position.