# NetEV Hand Tuple Strategy and Extrapolation Requirements

## Executive Summary

This document captures the strategic framework for NetEV's hand tuple system and the requirements for fixing the extrapolation algorithm. The key insight is that **it always matters** - every hand lookup is critical because small competitive differences create large point deltas (8-point swings between similar hands).

## Hand Tuple Element Structure

### 2-Element Hand Types

**No kickers possible - only primary rank matters:**

- **5K (Type 10), 6K (Type 12), 7K (Type 14), 8K (Type 16)** (Same rank hands)
- **Front Position Trips** (3 cards total, no room for kickers)

**Format:** `(hand_type_code, primary_rank)`

### 3-Element Hand Types

**Multiple ranks create competitive differences:**

#### Standard 3-Element Types

- **Four of a Kind:** `(8, quad_rank, kicker_rank)`
- **Full House:** `(7, trip_rank, pair_rank)`
- **Flushes:** `(6, highest_card, second_highest, third_highest)` → Reduced to `(6, highest, second)`
- **Trips (Middle/Back):** `(3, trip_rank, kicker_rank)`
- **Pairs:** `(2, pair_rank, kicker_rank)`
- **Two Pair:** `(position_dependent)`
- **High Card:** `(1, highest, second_highest)`

#### Special 3-Element Types (Sequence-Based)

**Straights (Hand Type 5):**

- **Regular:** `(5, highest_card, highest_card - 1)`
  - A-K-Q-J-10: `(5, 14, 13)`
  - K-Q-J-10-9: `(5, 13, 12)`
  - Q-J-10-9-8: `(5, 12, 11)`

- **Wheel:** `(5, 14, 5)` for A-2-3-4-5

**Straight Flushes (Hand Types 9, 11, 13, 15):**

- **5-Card SF (Type 9):** Same as straights
  - Regular: `(9, highest_card, highest_card - 1)`
  - Wheel: `(9, 14, 5)`
- **6-Card SF (Type 11):** `(11, highest_card, highest_card - 1)` or `(11, 14, 6)` for wheel
- **7-Card SF (Type 13):** `(13, highest_card, highest_card - 1)` or `(13, 14, 7)` for wheel
- **8-Card SF (Type 15):** `(15, highest_card, highest_card - 1)` or `(15, 14, 8)` for wheel

## Position Dependencies

### Position-Specific Element Counts

- **Front Trips:** 2 elements (no kickers in 3-card front)
- **Middle/Back Trips:** 3 elements (kickers matter)
- **All other hand types:** Consistent 3 elements across positions

### Strategic Relevance by Position

**Back Position:**

- High card, pair, two pair, trips: Valid but never strategically played
- Algorithm still evaluates them (then rejects)

**Middle Position:**

- High card: Never gets played strategically
- Algorithm uses "incomplete hands" to represent best possible high card arrangement
- Gets evaluated and almost always rejected

**Front Position:**

- More variety in playable hands due to 3-card constraint

## Incomplete Hand Strategy

### Algorithm Design

- **Generates incomplete high card hands** for all 17 possible high cards
- **Algorithm evaluates each** using NetEV lookup table

- **Almost always rejected** in favor of complete hands (pairs, trips, etc.)
- **Requires lookup entries** for algorithm to function properly

### Coverage Requirements

Must provide lookup entries for:

- **All complete hands** (pairs, trips, flushes, straights, etc.)
- **All incomplete hands** (17 high card variations by position)
- **Even strategically irrelevant combinations** that get generated then rejected

## Current Extrapolation Problems

### Problem 1: Invalid Hand Generation

**Root Cause:** `_generate_all_possible_hands()` creates impossible poker combinations

#### Examples of Invalid Tuples:

- `(5, 14, 2)` - Ace-high straight with 2 as second card (impossible)
- `(5, 13, 14)` - King-high straight with Ace as second card (impossible)

#### Valid Straight Patterns:

- Regular: `(5, high, high-1)` only
- Wheel: `(5, 14, 5)` only
- No other combinations possible

### Problem 2: Hierarchy Violations

**Root Cause:** Extrapolated values don't respect poker hand hierarchy

#### Example:

- King-high straight (real data): 2.238 points
- Invalid "Ace-high" straights (extrapolated): 1.92 points
- **Result:** Inferior hands scoring higher than superior hands

## Strategic Solution Framework

### Core Principle: Gap Analysis Over Junk Generation

**Instead of:** Generate all possible tuples → Filter out invalid ones **Better:** Analyze 640 real data rows → Identify actual gaps → Fill real gaps only

### Two-Part Solution Strategy

#### Part 1: Real Gap Identification

1. **Analyze existing 640 lookup rows** by hand type
2. **Identify missing hands** within logical hand type sequences
3. **Focus on actual gaps** not impossible combinations
4. **Hand-type-specific gap analysis** using poker logic

#### Part 2: Bounded Extrapolation

1. **Find hierarchy neighbors** for each gap
2. **Identify upper/lower bounds** from existing data
3. **Extrapolate within bounds** to preserve hierarchy
4. **Guarantee no hierarchy violations**

### Coverage Philosophy: "It Always Matters"

#### Why Complete Coverage is Required:

- Small differences create huge competitive deltas (8-point swings)
- 4 Aces with K vs Q kicker: +4 vs -4 points difference
- King-high vs Queen-high flush: Same 8-point differential
- NetEV's competitive advantage comes from recognizing subtle distinctions
- Fallback logic loses nuanced competitive analysis

**Implication:** Every missing lookup potentially loses critical competitive intelligence

## Implementation Requirements

### Valid Tuple Generation Rules

#### 2-Element Types:

- `(hand_type, primary_rank)` for ranks 2-14

#### 3-Element Types by Category:

#### Rank-Based:

- `(hand_type, primary_rank, secondary_rank)` with poker constraints

- Full House: trip_rank must be valid, pair_rank must be different
- Four of a Kind: quad_rank must be valid, kicker_rank must be different

**Sequence-Based (Straights/Straight Flushes):**

- Regular sequences: `(hand_type, high, high-1)` for high ∈ {9,10,11,12,13,14}
- Wheel sequences: `(hand_type, 14, wheel_length)` for appropriate wheel_length
- **No other combinations valid**

### Hierarchy Preservation Algorithm

For each gap hand:
1. Find next stronger hand in hierarchy
2. Find next weaker hand in hierarchy
3. Extract EV bounds: weaker.ev ≤ gap.ev ≤ stronger.ev
4. Extrapolate within bounds using appropriate methodology
5. Validate hierarchy preservation across entire table

### Validation Requirements

- **No impossible poker combinations** in tuple generation
- **Strict hierarchy preservation** in all extrapolated values
- **Complete coverage** for algorithm functionality
- **Position-appropriate** element counts and constraints

## Success Criteria

1. **Zero invalid hand tuples** in lookup table
2. **Perfect hierarchy preservation** across all hands
3. **Complete coverage** for optimization algorithm needs
4. **NetEV strategic decisions** based on accurate competitive intelligence
5. **Elimination of lookup gaps** that could affect strategic choices

## Future Considerations

- **Potential expansion** from 3 to 4 elements for flushes if precision needed
- **Position-specific optimizations** for hands that never get played
- **Algorithm efficiency improvements** based on strategic relevance patterns
- **Lookup table size optimization** while maintaining complete coverage