Advanced Goal Measurement Report

1. Executive Summary

This report measures and compares round scores of **Greedy**, **Balanced**, and **Mixed compositions** using the standalone harness advanced_measurement.py.

Lineups tested:

- All Greedy
- All Balanced
- B2G2 (2 Balanced, 2 Greedy)
- B3G1 (3 Balanced, 1 Greedy)
- B1G3 (1 Balanced, 3 Greedy)

Key findings:

- In **symmetric lineups** (all greedy, all balanced), results are trivial: each side wins 100% by definition.
- In **mixed lineups**, outcomes strongly depend on composition:
  - **B2G2**: roughly **parity**, small tilt toward Balanced (~51–52% wins at large N).
  - **B3G1**: strongly favors Balanced (~75–76% wins).
  - **B1G3**: strongly favors Greedy (~73–74% wins).
- **Conclusion**: Neither heuristic dominates globally; the relative composition of bots determines the winner.

2. Introduction

Goal: Compare total round scores across different bot lineups and quantify the balance between Greedy and Balanced strategies.

Scope:

- Compare strategies across increasing numbers of games (N=20, 50, 100, 1000, 10000).
- Generate score distribution histograms per lineup and strategy.
- Report win counts, win rates, and mean scores.

3. Methodology

- **Tool:** advanced_measurement.py harness
- **Engine:** Poker module, bots auto-arranged with greedy or balanced heuristics
- **Lineups tested:** all_greedy, all_balanced, b2g2, b3g1, b1g3
- **Metrics collected:**
    - Win counts per strategy
    - Win rate %
    - Mean score per strategy
- **Visualization:** Histograms saved to plots/ for each N and lineup

4. Results

4.1 Small Samples (N=20, 50, 100)

Lineup	N=20 (Win %)	N=50 (Win %)	N=100 (Win %)
All Greedy	Greedy 100%	Greedy 100%	Greedy 100%
All Balanced	Balanced 100%	Balanced 100%	Balanced 100%
B2G2	Greedy 54%, Balanced 46%	Greedy 49%, Balanced 51%	Greedy 45%, Balanced 55%
B3G1	Greedy 32%, Balanced 68%	Greedy 25%, Balanced 75%	Greedy 29%, Balanced 71%
B1G3	Greedy 83%, Balanced 17%	Greedy 73%, Balanced 27%	Greedy 68%, Balanced 32%

4.2 Larger Samples (N=1000, 10000)

Lineup	N=1000 (Win %)	N=10000 (Win %)
All Greedy	Greedy 100%	Greedy 100%
All Balanced	Balanced 100%	Balanced 100%
B2G2	Greedy 48%, Balanced 52%	Greedy 49%, Balanced 51%

Lineup	N=1000 (Win %)	N=10000 (Win %)
B3G1	Greedy 24%, Balanced 76%	Greedy 24%, Balanced 76%
B1G3	Greedy 74%, Balanced 26%	Greedy 74%, Balanced 26%

Observation:

- Results stabilize at large N.
- **B2G2** shows near-parity, with Balanced slightly stronger.
- **B3G1** heavily favors Balanced.
- **B1G3** heavily favors Greedy.

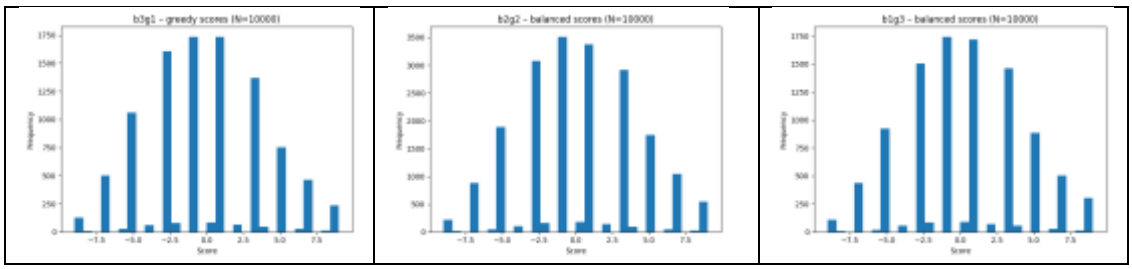
5. Discussion

Greedy vs Balanced is composition-dependent:

- Equal numbers (B2G2) → near parity, Balanced slightly ahead.
- Balanced-heavy (B3G1) → Balanced dominant (~3:1).
- Greedy-heavy (B1G3) → Greedy dominant (~3:1).

Score means confirm this: Balanced strategies tend to hold slightly positive averages when they are in the majority, Greedy when they dominate.

Histograms: distributions widen with larger N but converge to stable averages, demonstrating normal-like spread across all mixes (b3g1, b2g2, b1g3)



6. Balanced Exploration Stats

Balanced exploration was measured across N = 20, 50, 100, 1000, and 10,000 deals (×4 hands per deal). Two counts were tracked:

- **Valid hands** → candidate plays generated
- **Pruned hands** → candidates discarded (valid − 1)

N Deals	Hands	Valid Total	Pruned Total	Avg Valid	Avg Pruned
20	80	298	218	3.73	2.73
50	200	759	559	3.80	2.80
100	400	1519	1119	3.80	2.80
1000	4000	14,784	10,784	3.70	2.70
10000	40000	148,846	108,846	3.72	2.72

Discussion

- The Balanced heuristic consistently finds **~3.7 candidates per hand** and prunes **~2.7** of them.
- These averages stay stable across small and large runs, showing the search explores a predictable number of alternatives.
- Compared to Greedy (which produces only one arrangement), Balanced's extra exploration explains its higher runtime cost but also its more defensive, flexible play.

Appendices

```
@echo off

REM ============================================

REM Environment Info

REM ============================================

ver

python --version


REM ============================================

REM Warmup (import overhead)

REM ============================================

echo.

echo === Warmup ===
```

```
python advanced_measurement.py --games 1 --seed 111 --plots-out plots


REM ============================================

REM Main Runs (score comparisons + histograms)

REM Sizes: 20, 50, 100, 1000, 10000

REM ============================================


echo.

echo === GAMES=20 ===

python advanced_measurement.py --games 20 --seed 2020 --plots-out plots


echo.

echo === GAMES=50 ===

python advanced_measurement.py --games 50 --seed 5050 --plots-out plots


echo.

echo === GAMES=100 ===

python advanced_measurement.py --games 100 --seed 100100 --plots-out
plots


echo.

echo === GAMES=1000 ===

python advanced_measurement.py --games 1000 --seed 10001000 --plots-out
plots


echo.

echo === GAMES=10000 ===

python advanced_measurement.py --games 10000 --seed 1000010000 --plots-
out plots
```

```
REM ============================================

REM Balanced Exploration Stats (valid/pruned)

REM Runs across Ns: 20, 50, 100, 1000, 10000

REM ============================================

echo.

echo === Balanced Stats: N=20 ===

python advanced_measurement.py --games 20 --seed 2020 --balanced-stats


echo.

echo === Balanced Stats: N=50 ===

python advanced_measurement.py --games 50 --seed 5050 --balanced-stats


echo.

echo === Balanced Stats: N=100 ===

python advanced_measurement.py --games 100 --seed 100100 --balanced-
stats


echo.

echo === Balanced Stats: N=1000 ===

python advanced_measurement.py --games 1000 --seed 10001000 --balanced-
stats


echo.

echo === Balanced Stats: N=10000 ===

python advanced_measurement.py --games 10000 --seed 1000010000 --
balanced-stats
```