Intermediate Goal Measurement Report

1. Executive Summary

This report presents measurement results for the Intermediate Goal of the Chinese Poker (Pusoy) Project.

Objective

"Replace Players 2, 3, and 4 with basic bot opponents using simple heuristics, and measure runtime and space usage of their arrangements."

Measurements

- Runtime execution was measured for five bot assignment strategies:
    1. All Greedy
    2. All Balanced
    3. 2 Greedy, 2 Balanced
    4. 1 Greedy, 3 Balanced
    5. 3 Greedy, 1 Balanced
- Sample sizes: 20, 50, 100, 1000, and 10000 games.
- Space usage was tracked via Python tracemalloc peak memory.

Key Findings

- Greedy bots remain the fastest and most memory-efficient.
- Balanced bots are consistently ~2× slower with slightly higher peak allocations.
- Mixed setups scale predictably depending on the number of balanced bots.
- The scoring phase contributes 30–40% of total runtime across all runs.

2. Introduction

The Intermediate Goals required three changes:

1. Replace Players 2–4 with bots using heuristics.
2. Enable a human to play against bots.
3. Validate bot arrangements and scoring.

To evaluate progress, runtime and memory usage were measured with a custom tool (measurement.py). The tool validates correctness with asserts while capturing performance metrics.

3. Methodology

Measurement tool: Simplified measurement.py, which runs only the bot suite.

Metrics collected:

- Runtime: average, p95, max (seconds).
- Memory: average and max peak (KB).

Experiments executed:

- Configurations: All Greedy, All Balanced, 2G/2B, 1G/3B, 3G/1B.
- Sample sizes: 20, 50, 100, 1000, and 10000 games.

4. Results

4.1 Summary – Average Overall Runtime (s)

Config	20 Games	50 Games	100 Games	1000 Games	10000 Games
All Greedy	0.0062	0.0064	0.0064	0.0138	0.0192
All Balanced	0.0150	0.0149	0.0148	0.0477	0.0476
2G / 2B	0.0105	0.0104	0.0184	0.0337	0.0333
1G / 3B	0.0125	0.0123	0.0417	0.0407	0.0404
3G / 1B	0.0079	0.0089	0.0262	0.0266	0.0267

4.2 Summary – Peak Memory Usage (KB)

Config	20 Games	50 Games	100 Games	1000 Games	10000 Games
All Greedy	2.34	2.31	2.32	2.34	2.34
All Balanced	2.37	3.30	2.79	3.12	3.09
2G / 2B	2.41	2.89	2.67	2.72	2.91
1G / 3B	2.36	2.89	2.67	2.89	2.91
3G / 1B	2.43	2.96	2.74	2.74	2.96

4.3 Observations

Greedy stays consistently fastest across all runs (6–19 ms avg).

Balanced stabilizes near 15 ms at small runs, but scales slower (≈48 ms by 1000–10000 games).

Mixed configs scale **linearly with balanced players**:

- 2G/2B: mid-range (~10–34 ms).
- 1G/3B: heaviest mixed (~40 ms).
- 3G/1B: faster (~27 ms).

Memory flat and predictable (~2–3 KB). Balanced-heavy configs occasionally spike (up to 3 KB).

5. Discussion

Greedy bots: Most efficient, best for stress runs or classroom demos.

Balanced bots: Consistently ~2× slower; appropriate for testing defensive arrangement strategies.

Mixed bots: Show linear scaling with number of balanced players; offer a middle ground.

Scoring: Takes 30–40% of runtime consistently, regardless of configuration.

Memory: Flat and predictable across all runs (2–3 KB); not a limiting factor.

Test Appendices

CLI Tests:

```
@echo off
REM ============================================
REM Environment Info
REM ============================================
ver
python --version

REM ============================================
REM Warmup (import overhead)
REM ============================================
echo.
echo === Warmup ===
python measurement.py --games 1 --seed 111

REM ============================================
REM Main Runs (all five configs each)
```

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

echo.
echo === GAMES=20 ===
python measurement.py --games 20 --seed 2020

echo.
echo === GAMES=50 ===
python measurement.py --games 50 --seed 5050

echo.
echo === GAMES=100 ===
python measurement.py --games 100 --seed 100100

echo.
echo === GAMES=1000 ===
python measurement.py --games 1000 --seed 10001000

echo.
echo === GAMES=10000 ===
python measurement.py --games 10000 --seed 1000010000
```