

Suspicion: A Graph Based Approach to Knowledge Tracking During Game Play

Jack Vuturo

Abstract—Suspicion is a hidden-identity board game in which players collect gems while attempting to deduce the secret guest identities of their opponents from indirect, ambiguous observations. We frame the game as a partially observable sequential decision problem and investigate whether structured logical inference can substitute for explicit belief-state reinforcement learning. Each agent represents its identity uncertainty as a bipartite graph between players and candidate guests, prunes the graph with Hall’s Marriage Theorem, and produces end-game identity guesses by combining exact (or Monte Carlo) edge marginals with a Graph Convolutional Network operating on the pruned graph. We compare four agents—a stealth heuristic (S3GI), two graph-based agents driven by a two-ply alpha-beta search (Logical and Heuristic), and a learned value-network agent (Smarty)—against a random baseline across two 1000-game five-player tournaments. In head-to-head matches against four random opponents, the graph-based agents and S3GI win 45–50% of games, roughly three to four times the head-to-head random baseline of about 13%, while Smarty, trained online with return-to-go targets, wins only 14% and falls below random play. The results suggest that in Suspicion the dominant lever for performance is deduction quality rather than gameplay policy refinement, and that bipartite graphs with Hall propagation capture most of the available identity information without a learned belief model.

Index Terms—Bipartite graphs, game-playing agents, graph convolutional networks, Hall’s marriage theorem, hidden-information games, partial observability.



1 INTRODUCTION

SUSPICION is a hidden-identity deduction board game in which each of n players is secretly dealt a guest card from a deck of ten characters. On each turn, players roll dice, move guest pieces around a mansion, and execute card actions that reveal partial information — a gem pickup constrained by room color, a question about who can see whom, or a glimpse at the deck. The game ends when the gem supply is exhausted, at which point each player simultaneously guesses every opponent’s identity. Final score combines gems collected during play with points awarded for correct guesses.

The game is a natural testbed for sequential decision making under partial observability. The true state — the assignment of guests to players — is never directly observed; an agent must instead maintain a belief over the $\binom{10}{n} \cdot n!$ possible assignments and refine it from indirect, often ambiguous observations. The decision problem has two coupled phases. During play the agent balances a low-risk, low-reward objective (gem collection) against a high-risk, high-reward objective (gathering evidence that will sharpen end-game guesses), all while leaking as little information about its own identity as possible. At game end, the accumulated

belief must be collapsed into a single hard assignment for each opponent.

This paper investigates whether structured logical inference can serve as an effective substitute for explicit belief-state reinforcement learning (RL) in this setting. We represent identity uncertainty as a bipartite graph between players and candidate guests, prune it with Hall’s Marriage Theorem, and produce final guesses with a Graph Convolutional Network applied to the resulting graph. We compare this approach against a stealth-oriented heuristic agent (S3GI), a learned value-network agent (Smarty), and a random baseline across two 1000-game tournaments. The graph-based agents win three to four times more often than random play in head-to-head matchups, and a learned value network — trained online with return-to-go targets — fails to match even the hand-crafted heuristic, suggesting that in Suspicion the dominant lever is deduction quality rather than gameplay policy.

2 METHODOLOGY

Five agents are evaluated in a round-robin tournament: Random (baseline), S3GI, Logical, Heuristic, and Smarty. Table ?? summarizes their design

choices across the three core components: game-play policy, identity-deduction machinery, and end-game guessing procedure. Agents are compared by win rate, average gem score, average guess score, and average total score across 1000 five-player games.

TABLE 1

Agent design summary across gameplay, deduction, and guess components.

Agent	Gameplay	Deductions	Guess
Random	Random	List (Gem Pickup)	Random
S3GI	Stealth Heuristic	List (Gem, Ask, ViewDeck)	Consistent Matching
Logical	Alpha-Beta	BipartiteGraph + HMT	GCN
Heuristic	Alpha-Beta	BipartiteGraph + HMT + Guest-Axis	GCN
Smarty	Value Network	BipartiteGraph + HMT	GCN

2.1 Identity Tracking using Bipartite Graphs

The Logical and Heuristic agents model identity uncertainty with a bipartite graph (Figure ??). One node set contains the five players (including the agent itself); the other contains the ten possible guest identities. An edge (P, G) means guest G is still a candidate identity for player P . All edges are present at game start and are removed as evidence accumulates.

Edges are either *hard* (definitively possible) or *conditional*. A conditional edge records the set of game turns on which guest G appeared in player P 's candidate set during an ambiguous gem-pickup observation. Conditional edges are promoted to hard when singleton resolution collapses them, and are removed when subsequent evidence rules them out. For the GCN adjacency matrix, hard edges have weight 1.0; conditional edge weights are proportional to the fraction of total observation turns attributable to that player, providing graded confidence.

2.2 Identity Pruning using Hall's Marriage Theorem

Hall's Marriage Theorem (HMT) states that a bipartite graph admits a perfect matching if and only if for every subset S of one side, its neighborhood satisfies $|N(S)| \geq |S|$. Applied in reverse: if a subset of k players shares a combined neighborhood of exactly k candidate guests, those guests cannot belong to any player outside the subset and are pruned from their edge sets (see Figure ?? for an example). The implementation (`applyHallPropagation`) iterates over subsets of size $k \in \{2, 3\}$ until no further removals are possible, then runs singleton propagation: when a player has exactly one remaining candidate, that guest is removed from all other players' edge sets.

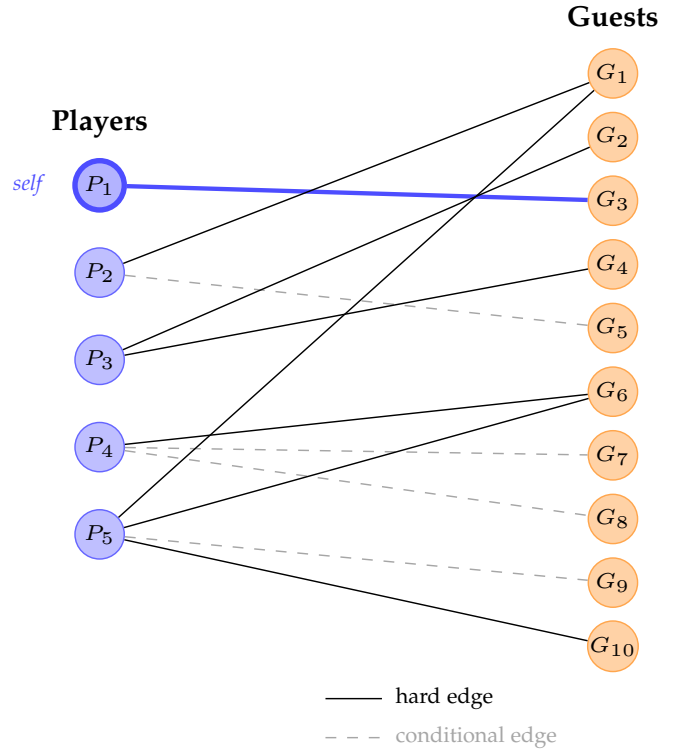


Fig. 1. Mid-game bipartite graph (5 players, 10 guest identities). P_1 (self, bold blue) holds a single confirmed hard edge to G_3 , established at initialization. Solid black edges are hard constraints; dashed gray edges are conditional, with weights proportional to per-turn gem-pickup observation evidence. Guests with no edges have been eliminated by prior observations or Hall propagation.

2.3 Heuristic Guest-Axis Deduction

The Heuristic agent additionally applies a guest-column heuristic (`applyGuestAxisDeduction`). For each guest G , the algorithm collects all players with conditional edges to G and their associated observation-turn sets. If exactly one player's turn-set covers the union of all other participants' turn-sets for G —and the union contains at least four turns of evidence—that player is committed to identity G as a hard assignment. This deduction is *not* logically sound, since multiple players can occupy the same room, but it enables aggressive identity commitment when evidence is strong.

2.4 Gameplay: Alpha-Beta Card Search

The Logical and Heuristic agents select cards and card actions via a two-ply alpha-beta search. For each card, its two constituent actions are expanded into candidate moves (gem color choices, ask targets, or piece moves) and scored by a heuristic:

$$h = 5 \cdot \text{colors} + 1 \cdot \text{gems} - 0.7 \cdot \text{edges} + 4 \cdot \text{confirmed} + 1.5 \cdot |\text{selfPossible}|$$

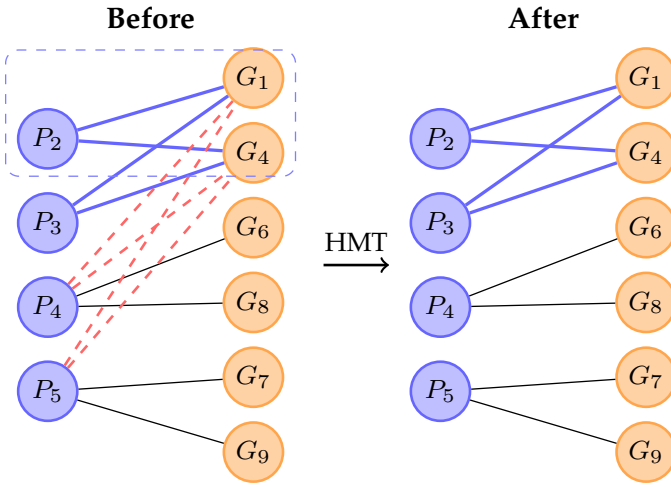


Fig. 2. HMT pruning example. $\{P_2, P_3\}$ form a Hall pair sharing exactly $\{G_1, G_4\}$ (blue box, blue edges). Those two guests are exclusive to the pair, so all edges from P_4 and P_5 to G_1 and G_4 (red dashed) are removed, reducing search space without any loss of valid matchings.

where *edges* is the total remaining edge count in the bipartite graph (lower is better), *confirmed* is the number of opponents with a single remaining candidate, and *selfPossible* is the *camouflage set*—the number of guest pieces opponents could still mistake for the agent after its gem pickups. For *ask* actions the search branches on both possible responses and computes an expected score. Dice moves are handled separately: the agent’s own piece is steered toward rooms containing uncollected gem colors; other pieces move randomly.

2.5 S3GI: Stealth Heuristic Gameplay

The S3GI agent uses an information-theoretic stealth strategy. Instead of a bipartite graph, it maintains a list-based candidate set per opponent and a *possibles* map tracking which guests could still be mistaken for itself (initially all guests, shrinking after each gem pickup reveals a gem color constraint).

For move selection, S3GI enumerates all legal board states reachable after the two dice rolls. Each resulting board state is scored by (1) how many guests in the *possibles* set share the same gem-taking room as the agent (*camouflage*) and (2) the expected entropy reduction from the best available *ask* action on that board. Moves that preserve more confusion about the agent’s own identity are preferred; among equally safe moves, the one offering the greatest information gain is selected.

2.6 Smarty: Value Network Gameplay

The Smarty agent (`SmartGraphBot`, Python) uses the same bipartite graph, HMT propagation, and GCN guessing as the Logical agent, but replaces the hand-crafted heuristic function with a learned value network. The network is a 3-layer MLP ($75 \rightarrow 128 \rightarrow 64 \rightarrow 1$, ReLU activations) that estimates the expected future score from the current game state. Its 75-dimensional input feature vector concatenates: the GCN player→guest probability matrix ($5 \times 10 = 50$ values), own gem counts normalized by 10 (3 values), all 10 guest board positions normalized to $[0, 1]$ (20 values), the camouflage set size normalized by $|\text{guests}|$ (1 value), and the total remaining bipartite edge count normalized by 50 (1 value).

Training is online and return-to-go based. At each turn, the agent snapshots its current feature vector alongside the partial gem score accumulated so far. After the game ends, each snapshot is assigned a target of (final gem score – partial score at snapshot) + guess score, representing the remaining future value from that turn onward. The value network is updated by MSE loss over these snapshot–target pairs for four gradient passes per game using Adam, then the updated weights are saved.

2.7 Identity Guessing

GCN Iterative Inference (Logical and Heuristic)

At game end the Logical and Heuristic agents combine the bipartite graph with a Graph Convolutional Network (GCN), shown in Figure ?? . The GCN has two graph convolutional layers ($7 \rightarrow 32 \rightarrow 32$, ReLU activations) followed by a linear readout ($32 \rightarrow 10$) applied only to the player nodes, with per-player softmax. Seven-dimensional node features encode: a two-element one-hot type indicator (player vs. guest), self-identity flag, maximum edge probability, normalized distribution entropy, conditional-edge fraction, and NPC indicator.

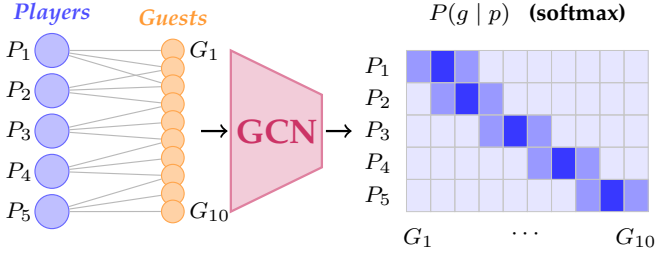


Fig. 3. GCN architecture. Player and guest nodes (7 features each) pass through two GCN layers operating on all 15 nodes. Only the 5 player embeddings feed a linear readout and per-row softmax, yielding the 5×10 probability matrix at right. Darker cells indicate higher probability; the darkest cell per row is the predicted identity.

Inference proceeds iteratively (Figure ??): the GCN produces per-player distributions, which are combined with graph-derived edge marginals (exact enumeration when the number of consistent matchings is ≤ 200 , otherwise Monte Carlo sampling) via the score $\text{edgeProb}[p][g] \cdot (1 + \text{gcProb}[p][g])$. The highest-scoring unassigned (*player*, *guest*) pair is committed as a hard assignment, the graph is updated, and the process repeats until all players are assigned.

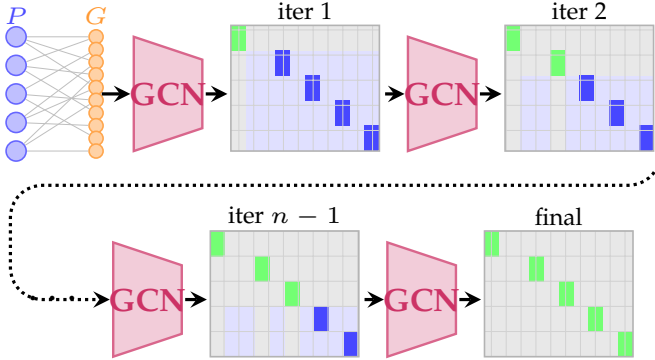


Fig. 4. Iterative GCN inference. At each step the highest-scoring unassigned (*player*, *guest*) pair (darkest blue cell) is committed as a hard match (green), and the graph is updated before the next GCN pass.

Consistent Matching (S3GI)

S3GI enumerates all injective player-to-guest assignments consistent with each player’s candidate list, counts how many consistent matchings contain each (*player*, *guest*) pair, and reports the most frequent assignment for each player.

3 EVALUATION

3.1 Experimental Setup

All agents were evaluated in 1000 five-player games. Because S3GI is implemented only in Java

and Smarty only in Python, we ran two separate tournaments: a Java tournament covering Random, S3GI, Logical, and Heuristic (with one extra Random to fill the fifth seat), and a Python tournament covering Random, Logical, Heuristic, and Smarty (again with one extra Random). Each game proceeds until the gem supply is exhausted; the winner is the player with the highest total score (gem score + guess score). Statistics are averaged per player per game. All GCN and value-network weights were loaded from checkpoints trained during prior on-line sessions; no additional training occurred during evaluation.

3.2 Results

Table ?? reports the Java tournament. All three graph-based agents substantially outperform the Random baseline ($\sim 7\%$ win rate). S3GI achieves the highest win rate (30.1%), narrowly leading Logical (29.1%) and Heuristic (26.5%). This edge comes primarily from guess score: S3GI’s consistent-matching enumeration produces slightly higher guess rewards despite gathering fewer gems on average than the graph bots. The graph bots collect marginally more gems (~ 7.3 vs. 7.0 for S3GI), reflecting the alpha-beta card selection that prioritizes gem diversity.

TABLE 2
Java tournament: 1000 games, 5 players

Agent	Win%	Avg Gems	Avg Guess	Avg Total
Random	7.1	6.28	17.34	23.63
S3GI	30.1	6.98	22.96	29.94
Logical	29.1	7.33	22.11	29.43
Heuristic	26.5	7.31	21.16	28.48

Table ?? reports the Python tournament, which introduces Smarty. Logical (37.8%) and Heuristic (33.7%) again dominate, while Smarty (5.8%) falls below even the Random baseline (11.4%). Smarty’s value network guides both card selection and movement, but the return-to-go training signal is noisy given the limited number of online training games, and the network appears not to have converged to a policy superior to the hand-crafted heuristic. Gem collection (5.47 on average) and guess accuracy are the lowest of any agent, suggesting the value network directs the bot toward states that are neither gem-rich nor identity-informative.

TABLE 3
Python tournament: 1000 games, 5 players

Agent	Win%	Avg Gems	Avg Guess	Avg Total
Random	11.4	6.13	17.94	24.08
Logical	37.8	7.16	22.42	29.58
Heuristic	33.7	7.15	21.90	29.05
Smarty	5.8	5.47	16.92	22.39

3.3 Individual Agent vs. Random Baseline

Table ?? pits each agent against four Random opponents over 1000 games. Horizontal rules separate matchup groups; bold rows are the evaluated agent and Random rows are averaged across the four opponents.

TABLE 4
Agent vs. 4 Random: 1000 games per matchup

Agent	Win%	Avg Gems	Avg Guess	Avg Total
<i>Random vs. 4 Random</i>				
Random (avg)	20.0	6.09	18.17	24.25
<i>S3GI vs. 4 Random</i>				
S3GI	50.0	6.94	22.97	29.91
Random (avg)	12.5	6.20	17.32	23.51
<i>Logical vs. 4 Random</i>				
Logical	45.8	7.20	22.69	29.88
Random (avg)	13.6	6.25	18.48	24.73
<i>Heuristic vs. 4 Random</i>				
Heuristic	45.2	7.19	22.12	29.31
Random (avg)	13.7	6.17	18.33	24.50
<i>Smarty vs. 4 Random (Python)</i>				
Smarty	14.1	5.30	16.50	21.80
Random (avg)	21.5	5.90	17.80	23.80

4 DISCUSSION

S3GI matches the graph-based agents in win rate (30.1% vs. 29.1% for Logical in the Java tournament) despite using a list-based candidate representation, no graph propagation, and a guess procedure based purely on enumerating consistent matchings. Its edge comes almost entirely from guess score: S3GI averages 22.96 guess points against 22.11 for Logical and 21.16 for Heuristic, while collecting roughly 0.3 fewer gems on average. Two factors plausibly account for this. First, when the candidate lists admit a tractable number of injective player-to-guest matchings, exact enumeration is unbiased — it counts every valid assignment and reports the per-player mode — whereas the iterative GCN procedure commits one (player, guest) pair at a time and can propagate early errors through the remaining inferences. Second, S3GI’s stealth-oriented gameplay preserves a larger camouflage set across

the game, which keeps the global identity structure more constrained because fewer players are leaking gem-color evidence. The result is that simpler deduction machinery, paired with information-preserving play, can match a more sophisticated graph-based pipeline.

The Heuristic agent extends Logical with the guest-axis deduction described in Section ??, yet wins 2.6 percentage points less often in the Java tournament and 4.1 percentage points less in the Python tournament. The deduction is acknowledged as unsound: it commits an identity to a single player when a turn-set covering condition is met, which is logically valid only if room occupancy is exclusive—which it is not. When the rule fires correctly it adds little, since Hall propagation usually arrives at the same conclusion shortly after; when it fires incorrectly it injects a hard edge that contradicts the true matching, distorting both the GCN’s input features and subsequent Hall pruning. The penalty is small in absolute terms but consistent in direction across both tournaments, supporting the view that the bipartite graph with Hall propagation alone already extracts most of the identity information available from gem-pickup observations.

Smarty’s failure is more dramatic than the conclusion alone suggests and deserves attribution to specific aspects of the design rather than to training volume in isolation. The value network drives both card selection and piece movement, so a poorly calibrated estimator misroutes the agent along two independent axes simultaneously. The training signal — final gem score minus partial score, plus guess score — is well-defined but noisy: in five-player games with stochastic dice and opponent variance, per-turn return-to-go is dominated by factors the 75-dimensional feature vector cannot explain, particularly opponents’ future deductions and movements. Without a separate policy head, a target network, or a curriculum to bootstrap a sensible estimator, online MSE updates against this signal appear to converge to a near-flat value function that prefers low-variance states regardless of their strategic content. Smarty’s average gem count of 5.47 — well below Random’s 6.13 — is consistent with this: the network is not directing the agent toward gem-rich rooms, and its guess score (16.92, the worst of any agent) shows it is not directing the agent toward identity-informative states either.

A methodological caveat applies to the cross-tournament comparison. Logical wins 29.1% in the Java tournament and 37.8% in the Python tournament, but this should not be read as a real change

in agent strength: the opponent field differs, with Python replacing S3GI with Smarty, the latter being effectively a guaranteed loss against any competent agent. The head-to-head matches in Table ??, where each evaluated agent faces four Random opponents, provide a cleaner per-agent measurement and place Logical (45.8%), Heuristic (45.2%), and S3GI (50.0%) within five percentage points of one another. The round-robin tournaments are therefore more useful for ranking peer agents than for measuring absolute strength.

5 CONCLUSION

This paper evaluated five agents on the hidden-identity board game Suspicion, combining gem-collection gameplay with end-game identity guessing. The results show that structured deduction clearly dominates random play: S3GI, Logical, and Heuristic all achieve win rates roughly three to four times higher than the random opponents they face in head-to-head matchups. Among the graph-based agents, S3GI's consistent-matching enumeration yields the highest guess scores despite simpler gem-collection heuristics, while Logical and Heuristic benefit from alpha-beta card selection at the cost of marginally lower guess accuracy. The guess-axis heuristic in the Heuristic agent did not improve performance over Logical, and in fact slightly reduced win rate in both tournaments, suggesting that the bipartite graph with Hall propagation alone captures most of the available identity information. Smarty's value network underperformed and failed to surpass Random, which we attribute to specific design choices—a single value head driving both card and movement decisions, return-to-go targets without a separate target network or curriculum, and a 75-dimensional feature vector that cannot account for opponent variance—rather than to training volume alone. Future work should explore a separate policy head, target-network stabilization, and curriculum play to allow the value network to converge. Overall, the results confirm that deliberate, logic-driven identity inference is the dominant factor in Suspicion performance.

REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.