_/_/_

Alpha-Beta + Stimulated annealing Approach for NPC Behaviour

Notes on Alpha Beta approach:-
- optimizes minmax algo by reducing the number of nodes evaluated in game tree
- used in ~~adverstal~~ adversarial deterministic game env.

Notes on stimulated annealing:-
- Probabalistic optimization technique.
- find gloabal optimum.
- init accepts worse situations with high probability and gradually decreasing this prob over time

NPC Behaviour
- focuses on creating realistic, effective and interesting NPC

x features - strategic Plan, act with appropriate timing, react to ~~the~~ changes in env.

Recent application of Simulated annealing in NPC

① Stimulated Annealing for ghost agents in Pac-Man -> developing attacking Behaviours (flanking surrounding Players) - [Maze game]
(ghost agents - NPC)

Recent Applications of ~~Stimulated~~ Alpha-Beta Pruning:- used in chess, checkers, Tic Tac toe => 2 player games

Hybrid Decision Making for Imperfect Information Games: Integrating Simulated Annealing with Alpha Beta Pruning

(eg: of imperfect information game game - Rummy like card game)

Mission :- To develop a hybrid AI framework that integrates Simulated Annealing and Alpha-Beta Pruning for optimized decision-making in imperfect information games, enhancing the strategic depth and human-likeness of NPC

Vision:
To pioneer a novel AI decision-making paradigm in the field of Game AI, enabling intelligent agents that adapt, optimize and strategize effectively in dynamic, probabilistic and multi agent env. - paving the way for next generation digital gameplay experiences.

WORKFLOW

Phase1 :-
defining game env and representing all components needed for intelligent decision making

eg:
* Player's Hand -> List of cards held by NPC

* Discard Pile : Visible cards recently discarded by other players

_/_/_

* Opponent Possibilities: Inferred probabilities of cards in opponent hands based on visible action

* DrawPile Cards remaining in the deck (partial or fully uncertain)

Possible Actions at Each Turn

- Draw a card from the drawpile or discard pile
- discard a ~~pile~~ card
- Declare (if valid melds are formed)
- Rearrange hand into potential sets/sequences (melds)

Hand Encoding for Optimization

- represent as vectors, bitmasks or graphs
  Nodes = cards
  Edges = potential links b/w cards forming melds.

Phase 2 :- Simulated Annealing (SA) - Local Hand Optimization

- Simulated Annealing is used here to explore different configurations of the hand to determine most promising card combinations and the best discard

Goal: Maximize the hand's score potential and minimize risk through intelligent discard

Steps:-

1) Initial state: Current configuration of the NPC's hand

2) Neihbor Generation :- Slightly alter hand config (eg. swapping cards b/w sets, trying different melds)

3) Energy function (Evaluation metrics)

Lower Energy = better hand
Incorporate:
- No. of valid sets/sequences
- Deadwood points (unmatched cards)
- Risk factor of ~~high~~ discarding high-usefulness cards

4) Annealing scheduling :-

- Begin with high randomness (temp)
- Gradually reduce over iterations.
- Converge to an optimal or near-optimal discard strategy.

(5) Output:
- Best discard option
- Ideal meld configuration to aim for in future turns

SA = ~~out of all possible meld pattern, which one gives best potential in current hand~~

__/__/__

Phase 3 Alpha Beta Pruning strategic Decision Tree

⇒ Simulated Annealing focuses on optimizing the NPC's current hand by identifying best meld config and discard choices within single game state

⇒ ABP - for multi turn strategic level
- It simulates the progression of the game over time - considering possible future actions

⇒

Goal of ABP :- make intelligent decision by forecasting moves defenses or counter moves based on player strategies

Steps :-

- decision tree construction:
- Root :- current game state
- children : All legal future game states after potential draws/discards
- incl opponents likely actions via belief states or probabilistic modeling

Evaluation func

Derived partly from SA-optimized hand state

includes

- probability of forming complete sets

__/__/__

- opponent threats or advantages
- time to potential victory
- Risk / rewards of drawing from discard pile.

O/P:-
whether to draw from deck or discard pile
which sets to pursue (influenced by SA O/P)

Defensive vs Offensive plays
(eg. discards to prevent opponent from picking a card)

phase 4:- Action Exec & Environment Feedback :-

- action based on SA + ABP o/p.
- AI perform chosen moves (draws, discards, rearranges, declares)

Game Updates :
The env reflects changes to the game state + new info updates internal models

<u>Feedback Loop:-</u>
use the updated state to
- Re-run for next hand optimization
- Expand ABP decision tree for future rounds

_/_/_

~~Phases: Evaluation And Learning~~

~~To improve over time, RL~~

Components	Role
SA	Hand optimization, best discard meld detection
~~ABP~~ ABP	strategic foresight, opponent modeling, decision tree
Feedback Loop	cont learning & env adaptation
Tool Integration	Modular, scalable, easy to simulate and visualize.

ROLE CLARIFICATION

SA - out of all possible meld patterns which one gives me the best potential with my current hand

How it works

- Takes current hand
- Tries various combinations (sets, seq)
- Uses an energy func (meld completeness,
- ~~Slowly cools (reduces~~ deadwood etc)

Slowly cools (reduces randomness) to settle on one optimal meld config

o/p: Best hand struct + which card to discard (locally optimal choice)

ABP: Given the meld struct SA suggested how will this play ~~over~~ out over time against opponent

How it works

Builds a decision tree:
Root = current state with SA optimized hand
children → future possible moves (draw, discard, oppo, actions)

· uses ABP ~~to~~ prune bad futures & focus on promising paths.

O/p :- strategic actions like whether to pursue marse meld or switch if oppo blocks
which cards to hold or discard for best future impact
→ Defensive / offensive play.

NOVELTY :-

* dual-~~decision~~ layered decision Arch
  · SA – local hand optimization
  ABP – global strategic planning
  a unique separation of tactical & strategic reasoning

* Optimization + foresight synergy
Most existing AI agents use either optimization (like greedy or rule-based) or look ahead strategies (eg. ABP).
This ~~[illegible]~~ model combines both intelligently

__/__/__

* SA for meld formation (not common)
  • using SA to ~~form~~ form meld and reduce deadwood - rarely explored in card games where heuristic dominate
* ABP applied in Imperfect Info Domain
  • ABP mostly used in perfect info games Applying it in a probabilistic setting like Rummy is novel
* Opponent Modeling + Adaptive Planning
  ~~cost~~ model considers oppo possible moves during ABP, adding realism and game theoretic depth not commonly seen in casual card games AI's

PROCEDURE FOR IMPLEMENTATION

* Step1:- Game env setup
  To do:-
  • Choose target ~~game~~ card game
  • define game rules, player actions (draw, discard, meld) and opponent modeling assumptions
  Tools: Python
  Petting 200 (for multi agent card game situation)
  Pygame (Simple GUI)
* Step2:- State Representation
  • encode player hand as a vector or graph
  Track:
  → Discard pile (Known)

→ Draw history (partial info)
→ Melds/sets in hand
Tools :- Numpy or Pandas for data rep
NetworkX (for graph-based model of cards)

* Step 3 :- Simulated annealing - Hand optimization
To do :- Init with current hand
define :-
→ Neighbor func - eg swap cards, try new melds
→ Energy func :- based on meld completeness, deadwood score, potential
→ ~~Cool~~ cooling schedule : gradually reduce randomness
Tools :- Custom SA code (or - scikit optimize)
Matplotlib (visualize annealing convergence)

* Step 4 :- Alpha-Beta Pruning - Strategic planning
To do :-
• Build a decision tree:
Root : SA optimized hand state
children : draws, discards, predicted opponent actions
• Use belief state models to handle uncertainty
• Apply ABP to prune suboptimal paths
Tools :- python

__/__/__

* Step 5: Action Exec
  to do: Based on ABP decision exec
  → Draw from discard / deck
  → Discard card
  → Declare meld
  • update env accordingly
  Tools: Petting zoo or custom game engine logic

* Step 6: Evaluation:-
  Test agent against
  → Rule based players
  → Greedy agents
  → Random players
  Measure:-
  • win rate
  • Average deadwood score
  • Convergence time
  Tools: Matplotlib / Seaborn - visualizations
  Pandas / ~~Exce~~ Excel → results log