SOLVING CLUEDO

Planning and Bayesian Networks Approach

Michal Balaban
Michal.Balaban@mail.huji.ac.il

Nataly Milman
Nataly.Milman@mail.huji.ac.il

Nimrod Lavie
Nimrod.Lavie@mail.huji.ac.il

Almog Argaman
Almog.Argaman@mail.huji.ac.il

Introduction to artificial intelligence – Final Project

The Hebrew University of Jerusalem
Jerusalem, Israel

***Abstract* – finding the murderer, weapon and room of a crime is the main goal of Cluedo. This paper explores two agents for the game. One using planning and the other using Bayesian Networks.**

I. INTRODUCTION – DEFINING THE PROBLEM AND OUR GOALS

A. The game

Clue is a murder mystery game for three to six player that consists of a board which shows the rooms, corridors and secret-passages of an English country house. The game box includes several colored playing pieces to represent characters, two six-sided dice, three sets of cards (room cards, character cards and weapon cards), Murder envelope to contain one card from each set of cards, and a Detective's Notes pad on which are printed lists of rooms, weapons and characters, so players can keep detailed notes during the game.

At the beginning of play, three cards – one suspect, one room, and one weapon – are chosen at random and put into the Murder envelope, so that no one can see them. These cards represent the facts of the case. The remainder of the cards are distributed among the players.

The objective of the game is to deduce the details of the murder, i.e. the cards in the envelope. There are six characters, six murder weapons and nine rooms, leaving the players with 324 possibilities. As soon as a player enters a room, they may make a suggestion as to the details, naming a suspect, the room they are in, and weapon. For example: "I suggest it was Professor Plum, in the Dining Room, with the candlestick." The player's suggestion must include the room they are currently in. The token for the suggested suspect is immediately moved into that room. A player's suggestion may name themselves as the murderer and may include cards in their own hand.

Once a player makes a suggestion, the others are called upon to disprove it. If the player to their left holds any of the three named cards, that player must privately show one (and only one) of the cards to them. If not, the process continues clockwise around the table until either one player disproves the suggestion, or no one can do so. A player's turn normally ends once their suggestion is completed.

A player who believes they have determined the correct elements may make an accusation on their turn. The accusation can include any room and may be made immediately following a suggestion. The accusing player privately checks the three cards in the envelope. If they match the accusation, the player shows them to everyone and wins; if not, they lose and the other players continue in this way until the game is won.

In our version of the game we implemented a relaxation of the board which is 25*25 matrix with a 7*7 block in the center with room markings, figure 1 - board, every room has only one entry (secret passages and multiple entries are not supported).

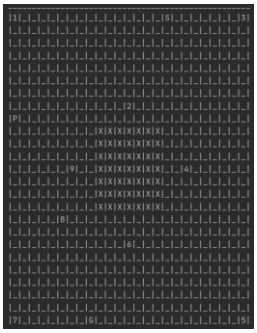


Fig. 1. Board. Numbers are rooms, letters are players and Xs are the envelope location that cannot be crossed.

Our Suggested Solutions

The deductive nature of the game and the possibility of learning both from your own actions as a player and other players'

actions made us choose planning and Bayesian networks as agents.

II. Approach and Methods

A. Planing

We have implemented our planning agent based on online re-planning principles.
Making an accusation in Clue ends with winning or losing, therefore it requires a high level of certainty. To reach it, there are certain options to be "crossed off" the list, one might even think of them as goals to be achieved. This is exactly why Planning was our first choice for solving Clue. However early on we encountered a problem: in classical planning the entire world is known and visible, while in our case discovering the world is the goal. At this point we turned to the book and learned about an online re-planner. We use an online re-planner that monitors its plan and changes it in response to new information that renders the current plan no longer useful.
Our Planner works as a detective, asking questions to receive information and eliminate suspects. It follows a predetermined plan which is designed (using heuristics) to reach the most information with a minimal amount of questions.
The planner starts by making a guess about who the murder triplet is (character, room, weapon), and based on this guess it builds a plan to verify that all the other cards are indeed in the hands of other players. After receiving an answer to a question the planner checks if it's knowledge about the world is contradicted by the new information. If some new information has arrived, the planner will re-plan.

Building the plan
For each card we assigned two propositions: one is with the ending “unknown” and the other is either “player” or “murder”, depending on whether we assumed it to be in part of the murder triplet or not. We also assigned three goal propositions character_found, room_found and weapon_found. The initial state is all the cards as “unknown”.
The actions are combinations of character card, weapon card and room card. The preconditions are the combination cards as unknown. Each combination creates three actions, each of which with different add and delete, and every time precisely one card is removed as “unknown” and added as “player” in case some player shows it to us. This causes every shown card to be added once since after this question is asked the proposition of this card with unknown will be deleted.
The murder triplet cards can only be marked as “murder” as a result of a question that has all these cards together.
We used our answers to exercise 3 to create the plan, using Planning_Problem with A* and not graph plan because it was a lot faster.
We also added a special question that adds the three found goal propositions only if all the other cards were turned to player or murder. We used a heuristic to build the plan:

Heuristics
Our heuristic has two parts.
The first part is the distances. For each successor we calculate the distance from our current location (in the beginning it is the start location and afterwards the location of the last room our player was in) to the room the successor question is asked in, and we add the distance from the successor question room to the room of the murder guess.
This part is used to direct the movement of the planner so it would not have to jump all over the board from question to question. Even though we can be moved because of other players questions, the planner takes this into account and continues asking questions from the next closest room according to the plan, which will still direct it toward the guessed murder room.
The second part is repeated questions. With each successor we also send a dictionary containing all the cards asked before, and how many times they were asked. We give a heavy penalty for repeated questions to make the planner ask questions that will give us the most new information (even though we will need to ask about these cards later since they were not confirmed as belonging to players before, we prefer to first ask completely new combinations in case we stumble upon the murder triplet by accident).

Re-planing
As we have said our online re-planner changes its plan in response to new information that make its plan irrelevant. It can be either a contradiction of our initial murder assumption or a contradiction of the result of an action in the plan.
We receive information in two ways – through our own questions and from other player's questions.
When the planner asks a question if the card shown was not the expected result of the question, a new plan will be created. It also keeps information on which player has the card that was shown.
When a different player asks a question, we only know what the question was, who answered and who didn’t, but we do not know the precise answer. For the player who answered, we keep these cards as a triplet of which only one card is owned by the player who answered. From this triplet we remove cards we already know the player they belong to, including our own. If we know the player that answered does not have some of the cards, we also remove them. If only one card remains, we know this card belongs to the player that answered, otherwise, we keep the combination and will reduce it to one card after receiving more answers.
For the players that did not answer, we keep the information that they do not have these cards and remove them from all the combinations added to these players before.
If because of such deductions we learn definite new information, we create a new plan, but we only update the plan once even if we get information on a few cards from one answer.

B. Bayesian networks agent

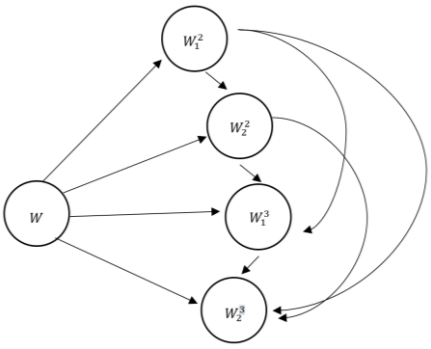
In Clue, there is one stochastic event at the beginning of the game – which is, drawing one card of each type to be a murder card, and then dealing the remaining cards to the players. From that point onward, although the amount of information one

might gain at each step is still not deterministic (because of the cubes), the agents basically gather partial predetermined information, trying to understand what the result of the event was. Given this nature of the game, we decided to implement a Bayesian Network (BN) solution to the problem. The main idea is to translate the evidence an agent typically observes to partial assignment of variables, from which the value of the main hidden variables can be inferred, based on known probability tables.

1. *The first approach to Bayesian networks*

The first BN implementation we tried was following Cai, C., & Ferrari, S. (2006, June) On the development of an intelligent computer player for CLUE, and included a critical simplifying assumption: each player has a fixed number of cards of each type in its hand. That is, the cards are being dealt from a separate deck for each type. Now, only cards of the same type may depend on each other, as seen in figure 2. During the game, we use basic inference algorithms on the BN, and we update it online as described below.

At the beginning of each game, three BN models are being initialized as the graph in figure 2, with CPTs of uniform distribution over all unseen cards, namely, the cards that are not in the agent's hand. If at any point in the game the agent sees that a player has a card, the CPT of the murder card is being updated by a query with an assignment of the seen card to the last unassigned variable in the model's chain. Then, the CPTs of all other variables are being updated forward, and the CPT of the variable that has been assigned will be updated with probability of 1 on its assignment. If the agent sees that a player failed to disprove some other player's suggestion, this means that the responding player does not have this suggestion's cards. Let a card in this suggestion be t_i, so the relevant CPTs are being updated s.t the probability of t_i is 0. At any turn, the agent moves in a greedy manner towards the most probable room according to the BN model, and if this room is accessible, makes a suggestion of this room and the most probable weapon and suspect. This way, whenever no other player can disprove the agent's suggestion, this suggestion is the correct one, and it makes an accusation.



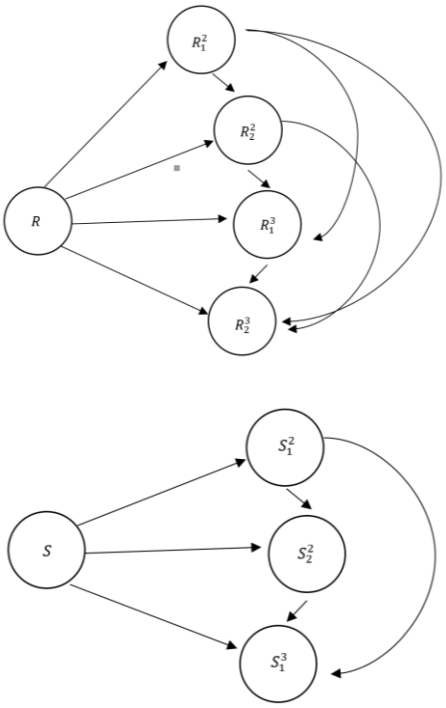


Fig. 2. Bayesian network graph

The BN model for the simplified variation of Clue, where T_i^j *is the* i^{th} *drawn card of type* T *of player* j *for all* $T \in \{S, W, R\}$ is representing suspects, weapons, and rooms]. In practice, the three independent parts are three different models, as described in Methods.

2. *The second approach to Bayesian networks*

Assumptions of the previous approach are made because the problem of updating probabilities in the Bayesian Network is NP-hard. But one of the assumptions basically tells us how many cards of each type the players have, which makes the game a bit less challenging. Challenge equals interest, so we've decided to build a BN player that will operate without any assumptions.

For each card, we create a variable $Card_i \in [0, numOfPlayers]$. So that the values in the CPT are indexes of the players and one more number that means that this card does not belong to any of the players – that is, it belongs to the murder triplet. Before the distribution of cards, all values have the same probabilities.

Instead of modeling the unknown connections between the cards, we decided to model possible question results.

For each player there are three types of situations:

1. The player answered a question player asked and now we know for sure that this player has that specific card.
2. The player answered a question another player asked and now we know the player who answered has at least one of three cards (S, W, R) that were in the question.
3. The player did not answer a question (ours or any other player's) and now we know for sure he has none of the three cards (S, W, R) that were in the question.

The first situation:

- For a pair of player and card, if the player shows it then the card surely belongs to him: this card's variable is equal to this player's index.
- Formally:

$$P\left(Q1_{P_j}\middle|\, Card\right) = 1 \Leftrightarrow Card = j$$
$$0\ otherwise.$$

Q1_Pj is a name in format "CardName + P + player_index", e.g. HallP1.

The last two situations are one and the same:

- For a specific player Pj and S, W, R cards (one from each category), Pj answers True to Q3 = SWR question if and only if the value of at least one of the cards corresponds to the given player's index j.
- Formally:

$$P\left(Q3_{P_j}\middle|S,R,W\right) = 1 \Leftrightarrow (S = j \vee W = j \vee R = j)$$
$$0\ otherwise.$$

Q3_Pj is a name in format "CharacterName + WeaponName + RoomName + P + player_index", e.g. ProfPlumCandlestickLibraryP1

This represents the world fully as it shows all in-game dependencies between cards and questions. At the same time, it builds a big search space: all card variables, for each of them and for each player Q1_Pj variables as well as for all players all combinations of Q3_Pj.

Let us go through an example of a query saying that Player_0 answered positively on someone else's question about (MissScarlette, Dagger, Hall). We expect it to update the probabilities of MissScarlette = Player_0, Dagger = Player_0, and Hall = Player_0 and stop there.

But in our Bayesian network, each of these cards has successors in the form of other triplets.

So it will try to update all of them. And they have parents in the form of three-card variables each.

The time it takes to update changes is not what we want from a player in a multi-player game.

So wait, we make a new network without the assumptions that save it from being NP-hard and get… An NP-hard problem?

Sounds about right and extremely far from our goal. Yes, we can have more players now, but one step takes more than half an hour to update.

Minimizing the search space

To understand what can be done we turned to the literature.

According to Zhang, N. L., & Poole, D. (1994) and Choi, A., Chan, H., & Darwiche, A. (2012) and the general math of probability calculations, the independence or irrelevance of the variables (nodes) as exists in our implementation, is what we need to be able to use edge deletion or nodes pruning.

We argue that given a Q3_Pj variable, the only relevant variables are its three parent cards, so we can remove Q3 nodes after the updates of their parent nodes and even more, that we do not need to create all of them at the beginning.

The reasoning behind it, simply put, is that all variables that cannot be updated should not be considered in the update at all. Given Q3_Pj, after their 3 parent card-nodes are updated their CPTs do change as they are what makes Q3_Pj be able to have True and False values. And the nodes that BN goes to update after it – all their children – both Q1_Pj and Q3_Pj are already fully calculated.

Each of these questions is a variable in {True, False} built according to the game rules. The values of their CPTs are strictly 0s and 1s.

According to probability theory, in CPT after some combination reached 0 probability, it would not become probable ever again and the sum for all values should be 1 so the probability of the second value will always stay 1.

Thus, updates these nodes do not change their CPTs and as a result, there is no need to check on their parent nodes as well.

So, we can use the algorithm and delete either edges or the nodes themselves before updating the card (parent) nodes. In our implementation, we create and connect the nodes given the question and delete it after the update. We tested it thoroughly and it indeed gave the same changes to the CPTs and as expected from the articles their deletion did not influence any other node.

Player and the model

BNPlayer2 creates the model, asks questions over a triplet of Suspect, Weapon, and Room when Suspect and Weapon are the most probable murder case cards and Room is the most probable card that is reachable from the current location. If none of them are reachable, it moves towards the most probable murder room.

It updates Q1_Pj node to True according to the player index that answered his question and shown card.

It updates Q3_Pj node to True according to the player index that answered some other player's question.

It updates Q3_Pj nodes to False according to the indexes of the players that didn't answer a question (both when the question is his or some other player's).

Remember that these are different nodes that have different names according to our explanation.

When the probabilities are high enough, we firstly ask a question as a suggestion and accuse only if no-one answered, otherwise update the tables and look for the most probable cards again.

III. Experiments and Results

The random player is designed to enter any unseen room it can and make a suggestion with any unseen weapon and suspect from within this room. If no room is available, the random player will move to another position on the board. Whenever its suggestion cannot be disproved, it makes an accusation of this very suggestion.

In order to evaluate and compare our different approaches, we ran hundreds of simulations in various set-ups. At first, we let our agents compete independently, each against two random players, and found out that all agents typically win, with winning rates of ~ 58% for Planner, ~72% for BN and ~76% for BN2 out of 200 games (figure 3). Then, we ran a simulation

of BN, Planner and Random, and found that BN wins ~58% and Planner wins ~36% (figure 4). In a simulation of BN2, Planner and Random, BN2 won 61% and Planner agent won 26%. Lastly, in the contest between BN, BN2 and Planner, BN won ~40%, BN2 won ~43% and Planner agent won ~17% of the games (figure 5).

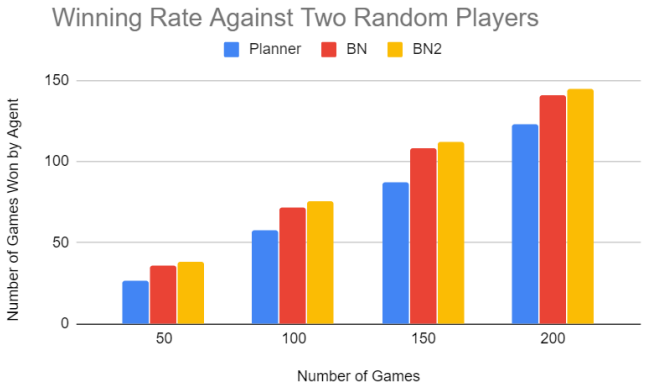


Fig. 3. Winning rate of each type of agent against two random players.

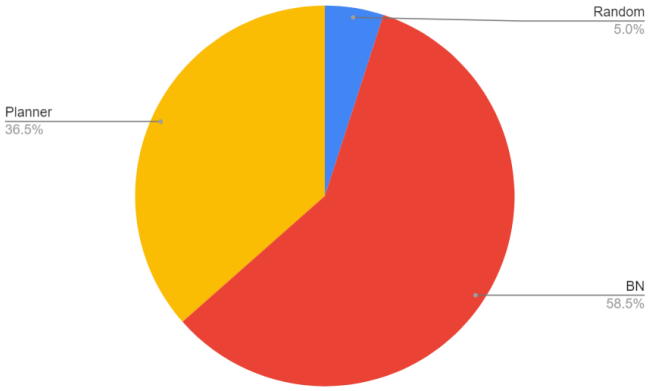


Fig. 4. Winning rates in a simulation of 200 games between BN, Planner and Random.

	Agents comparison		
	BN1	***BN2***	***Planner***
Winning Rate	40%	43%	17%

Fig. 5. Winning rates in a simulation of 200 games of a head-on contest between BN, BN2 and Planner.

CONCLUSIONS:

Both Planning and the probabilistic Bayesian Networks approaches were proved useful in the Clue game. The first BN agent has an inherent advantage with its simplifying assumption, so the fact that it is empirically better, figure 4, than the Planner does not imply by itself that this approach is preferable. But as we observe a consistent and coherent empirical advantage in winning rate of the two Bayesian Networks implementations over the Planning agent, we conclude that the probabilistic nature of the game makes it more suitable to model as a Bayesian Network, which is designed for these kind of stochastic environments. We consider the BN2 better than BN, in the sense that it gets slightly better results without the significant simplifying assumption of the first BN agent. We also suggest that Online Panning could be more useful for non-deterministic systems that change insignificantly along time, as opposed to Clue, where there is one unpredictable event at the beginning.

FURTHER WORK

The approach taken in the first implementation of a BN agent can be easily extended to more than 3 players, if the assumption on the number of cards of each type holds. So further research could be trying to see if the agent is better against more than 2 players, and how significant is this factor.

Currently, our approach to the second BN player does not take into account the number of cards each player has. Variables for balancing may be added to keep the sum of the cards strictly equal to the number of cards of each player (even if they are different). Say, if we know that some player has 6 cards and we know which are his, we may use the balancing table to update that there is no chance that any other card belongs to him. Maybe balancing tables can be created per category or for all of them and they are definitely going to take their sweet time updating, but these calculations still don't need any assumptions and may be useful to win faster.

REFERENCES

[1] Cai, C., & Ferrari, S. (2006, June). On the development of an intelligent computer player for CLUE: A case study on preposterior decision analysis. In 2006 American Control Conference (pp. 4350-4355). IEEE

[2] Cai, C., & Ferrari, S. (2008, June). A Q-learning approach to developing an automated neural computer player for the board game of CLUE®. In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence) (pp. 2346-2352). IEEE.

[3] Stuart, R., & Peter, N. (2016). Artificial intelligence-a modern approach 3rd ed.

[4] Choi, A., Chan, H., & Darwiche, A. (2012). On Bayesian network approximation by edge deletion. arXiv preprint arXiv:1207.1370.

[5] Zhang, N. L., & Poole, D. (1994). A simple approach to Bayesian network computations. In Proc. of the Tenth Canadian Conference on Artificial Intelligence.