University of Edinburgh
School of Informatics

Plan Recognition in RISK

4th Year Project Report
Artificial Intelligence and Software Engineering

Jibran Abdus Zaahir Khan

March 15, 2013

Abstract: This paper presents the design and implementation of a plan recognition agent based on an algorithm published by Christopher Geib and Robert Goldman in 2009[ref to work] called The Probabilistic Hostile Agent Task Tracker (PHATT). The plan recognition agents goal is to attempt to infer the unknown plan of a player from observations of their behaviour in the RISK environment.

Acknowledgements

(Distinguish between academic and non-academic work.)

Contents

1 Introduction **1**

1.1 Motivations . . . 1

1.1.1 What is game AI . . . 1

1.1.2 Difference between academic and game AI . . . 1

1.1.3 Computer Graphics previously getting more attention . . . 1

1.1.4 AI getting more attention now . . . 2

1.1.5 Plan Recognition . . . 2

1.1.6 Applications of Plan Recognition . . . 2

1.1.7 Application of plan recognition algorithms to games . . . 2

1.2 Artificial Intelligence and Board Games . . . 3

1.3 Aims . . . 3

1.4 Objectives . . . 4

1.5 Hypothesis . . . 4

1.6 Paper Structure . . . 4

2 Background **7**

2.1 Previous Work in Plan Recognition . . . 7

2.1.1 An Example of Plan Recognition . . . 8

2.2 Introduction to RISK . . . 10

2.2.1 Equipment . . . 10

2.2.2 Rules . . . 10

2.3 Why Plan Recognition in Board Games . . . 14

3 Design **15**

3.1 Environment Modelling . . . 15

3.1.1 Actions . . . 15

3.1.2 Territory . . . 17

3.1.3 Continent . . . 17

3.1.4 Player . . . 17

3.2 PHATT Modelling . . . 18

3.2.1 Computing an Explanation's Probability . . . 19

3.2.2 RISK Explanation Modelling . . . 19

3.2.3 Prediction Agent Pseudo Code . . . 21

3.2.4 Conceptual Issues . . . 24

3.2.5 Summary . . . 25

4 Implementation **27**
4.1 Modifications to Open Source Project 27
4.2 System Construction 27
4.3 System Architecture Concepts 28
4.3.1 Event Driven System 28
4.3.2 Component Based Class Architecture 28
4.3.3 Google Guava Libraries 28
4.3.4 Replaying Games 28
4.4 Summary 28

5 Evaluation **29**
5.1 Experiments 29
5.1.1 AI Games 29
5.1.2 Constrained Play 29
5.1.3 Free Play 30
5.2 Experimental Format 30
5.3 Data Collection 30
5.4 Experimental Findings 31
5.5 Outcomes 31
5.6 Critism 32

6 Conclusions **33**
6.1 Future Work 33
6.2 Final 33

7 Appendix **35**
7.1 Class Diagram 35
7.2 Experimental Results 35

Bibliography **37**

1. Introduction

AI has the potential to become the new driving force behind computer game innovation.

John David Funge, Artificial Intelligence for Computer Games

1.1 Motivations

1.1.1 What is game AI

Termed game Artificial Intelligence I.the application of A.I. techniques to games has been an area of research ever since the beginning of significant work on A.I.

Game A.I. is a term used used to differentiate A.I. applied in games from academic AI, and though the techniques have typically come from academia it is significantly different in both its scope and application.

1.1.2 Difference between academic and game AI

As the primary goal of academia is to further human understanding the scope of developed A.I. techniques are preferably general in application. Game A.I. on the other hand is built to create the illusion of intelligence and provide good game play for those playing the game, it therefore needs not be as general and is built with a singular purpose in mind which is usually considered to be any kind of control problem in a game.

Reference - difference between academic and game A.I [AI and computer games]

1.1.3 Computer Graphics previously getting more attention

Companies in the games industry who utilize game A.I. (which today is the vast majority) tirelessly seek an edge over their competition. This edge has typically come from computer graphics, effects such as dynamic rendering, the move from two dimensional to three dimensional to keep their customers interest.

1.1.4 AI getting more attention now

Emerging though is the idea that as users become accustomed to high quality graphics, developers will need something new to give their product an edge over their competitors. That edge will hopefully be better quality game A.I.

There has already been some notable examples such as the (name of the A.I technique from F.E.A.R) from F.E.A.R which used (describe the A.I. technique briefly), the game gained much acclaim and is referenced as a good example of game A.I. by the industry and others.

With this shift in focus, there is a greater incentive for the continued development of more sophisticated game A.I. likely with techniques developed in academia. The application of plan recognition algorithms may provide such an opportunity.

1.1.5 Plan Recognition

Plan recognition definition - plan recognition is the problem of inferring an agents plans from observations. Charniak, Goldman 1992

Research in plan recogniton began in the (find out)

1.1.6 Applications of Plan Recognition

According to Nate Blaylock a few of the most prominent of the application of plan recognition

User modelling, multi agent interaction and natural language processing

Currently plan recognition applications range from computer security to help systems (more detail).

1.1.7 Application of plan recognition algorithms to games

Plan recognition can perform automated game analysis which can be used by players to improve their performance.

Plan recognition can lead to AI responses that are personalized.

1.2 Artificial Intelligence and Board Games

In 1915 Leonardo Torres y Quevedo's built a chess automaton[ref]. It was considered the worlds first computer program and arguably the beginning of Artificial Intelligence and board games.

Thirty five years later Alan Turing published a landmark paper[ref], marking what many consider to be the 'birth of Artificial Intelligence'. Soon after John McCarthy officially coined the term Artificial Intelligence at a conference in Dartmouth College[ref]. He defined it as "the science and engineering of making intelligent machines"[ref].

E.g There existed Christopher Stracheys Game AI for the board game draughts built in 1951[ref]. As AI continued to develop so did the complexity of the A.I's seen in computer games, first AI to appear which used enemies was arcade game [ref].

Since then notable achievements such as Garry Kasparovs defeat to IBMs chess computer Deep Blue in 1997 [ref] have continued to impress the public.

From history it seems clear that AI and its application in games are intertwined becoming a growing area of interest in both commercial and academic fields.

Games are often considered a good metric for testing the quality of an AI [Ref]..

Some academic work done on RISK in particular. Development of an Intelligent Artificial Player for the Game of Risk[ref]. The author Michael Wolf[ref] claims RISK is generally well known but under recognised by academia.

1.3 Aims

What do we want to achieve?

An aim is a general statement of intent. It describes the direction in which the learner will go in terms of what they might learn or what the teacher/training will deliver.

To design a plan recognition agent for the board game RISK based on the PHATT algorithm developer by Geib and Goldman.

To implement a plan recognition agent for the board game RISK that can make successful predictions.

To further understanding of the complexities of performing plan recognition in the board game RISK.

1.4 Objectives

An objective is a more specific statement about what the learner should or will be able to do after the training experience.

To show plan recognition algorithms can be used successfully to predict an agent's plan in the board game RISK with an accuracy better than randomly picking a mission card.

To provide insight into the complexities of creating a plan recognition models for the RISK environment.

To provide insight into the nature of making plans in RISK.

1.5 Hypothesis

What do you want to answer?

Are plan recognition algorithms beneficial in games? Specifically RISK?

If the probability of a mission card is the highest among all the other mission cards of an agent, then it will be the correct mission card.

The predictions of the winners will be better than that of lowers.

Why because they are an artificially derived benefit which can be used by a player to optimize their behaviour, not solely , but in conjunction with the results of a plan recognition system.

Using more data from risk environment in the computation of the likelihood of explanations, the better the prediction accuracy will be.

1.6 Paper Structure

The structure of the paper is as follows:

The following chapter presents a background to the project where plan recognition and the board game RISK are introduced. Reasons for using plan recognition in board games are then discussed.

Chapter 3 describes the methodology of the project, it is split into two sections. The first is design, in this section PHATT is introduced and the design of the plan recognition agent is detailed. The subsequent section is implementation. Code related issues such as modifications to the open source project and any relevant design concepts are discussed.

Finally an an evaluation plan and conclusion is presented. In these, the experimental findings are presented, discussed and any conclusions derived from the experimental findings are presented.

2. Background

2.1 Previous Work in Plan Recognition

Many consider one of the earliest projects on plan recognition to have been in 1978. Having identified plan recognition as a problem in itself, Schmidt et al [ref - Schmidt Plan Recognition Problem] conducted experiments to determine whether or not people inferred the plans of other agents. From their results they created a rule based system called BELIEVER which attempted to capture the process of plan recognition.

Three years later Cohen, Perrault and Allen identified two different types of plan recognition *keyhole* and *intended* [ref - cpa strategies in natural language processing].

They defined each as:

- *Keyhole plan recognition* is the recognition of an agent's plan through unobtrusive observation".
- *Intended plan recognition* is the recognition of the plan of a cooperative agent who wishes to be understood.

In 1986 Kautz and Allen published a paper titled "Generalized Plan Recognition"[ref] which set the frame work of many plan recognition projects that followed and formed the basis of plan recognition through logic and reasoning. They defined keyhole plan recognition as involving the identification of a set of top-level goals from possible plans, which could be decomposed into related sub goals and basic actions, thus creating an event hierarchy also known as a *plan library*.

It was Charniak and Goldman[ref - 1991 paper] who first argued that plan recognition was largely a problem of reasoning under uncertainty and that any system which did not account for uncertainty would be inadequate. They went on to propose a probabilistic rather than logic based approach to plan recognition using a Bayesian model. Their research continues to be popular (check statement?) in many avenues of research including its application in games.

Albrecht, Zukerman and Nicholson [ref 1998 A Bayesian model for keyhole plan recognition in an adventure game] performed research on keyhole plan recognition using dynamic Bayesian networks to represent features of an adventure game. The result of their experiments they claimed "showed promise" for some domains.

More recently Synnaeve and Bessiere [ref - RTS Bayesian model plan recognition in Starcraft 2011] published a paper of the design and implementation of a plan recognition agent that through observations of building construction in the game Starcraft, can predict the types of units a player intends to produce based on what buildings they construct.

2.1.1 An Example of Plan Recognition

We can introduce common concepts, assumptions and the process of plan recognition with an example of an agent attempting to infer the plan of another agent in a non-adversarial environment.

Person A is cooking a meal for Person B, in other words A has a single *root goal* (a state they wish to reach), such as cooked beef hamburgers. This root goal can be decomposed into a number *sub goals* of such as cook meat patties or *actions* such as make meat patties which can be further decomposed into single actions

A wanting to surprise B, will not tell B what his root goal is. B cannot know what A is thinking but B must then somehow infer what A's root goal is likely to be by *observing* A cook in this way A's behaviour can be modelled as a Hidden Markov Model. Person B then models A's plan as follows.

Person B assumes A is a rational person, and that A has a *plan* to achieve this root goal. Whatever A's root goal is, it can likely be decomposed into *sub goals* such as cook hamburger meat. These sub goals can then be broken down into basic *actions* to achieve those sub goals, such as take hamburger meat out of packet. An important point to note is that these basic actions are not limited to only being part of a sub goal.

To simplify this example, let us make some assumptions:

- A believes that they can cook a meal.
- B can only infer the cooking plan of A by observing what A is cooking.
- B knows everything that A can cook.
- Through observing A cook, B can predict with absolute certainty what A will cook.
- A cannot hide any observations.
- A only wishes to cook one meal.
- A has no preferences of what to cook, in other words given a choice the probability of choosing is equally likely.
- One A does not change cooking plan.

Given these assumptions and B's knowledge of A's cooking abilities we can model a *plan library*, which is the set of all of A's possible cooking plans in the decomposed form that was previously detailed.

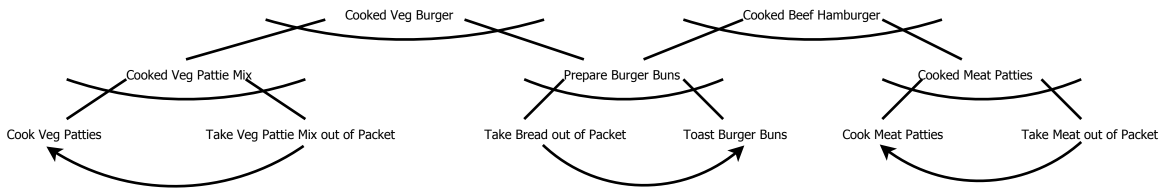

Figure 2.1: Root goals are nodes that have no parent nodes, sub goals are nodes that have both a parent and children and actions are nodes that have no children. And-nodes are represented by an undirected arc across the lines connecting the parent node to its children, or-nodes do not have this arc. Actions or goals that are dependant on a previous action are represented by an arc with an arrow.REPHRASE

The top level goals or root goals are cooked vegetarian (veg) burgers and cooked beef burger. These root goals can be broken down into sub goals, for the case of cooked veg burger the sub goals are cooked veg pattie mix and prepare burger buns.

A takes bread out of its packet and toasts the burger bun, looking at the plan library we have two possible *explanations* for As behaviour, cooked veg burger or cooked beef hamburger.

Each of these explanations are equally likely at this point.

A takes the meat of the packet, now given our assumptions we can conclude that A's plan is to cook beef hamburgers as A does not change his plan. In this way B has recognised A's plan.

2.2 Introduction to RISK

Developed and released by french movie director Albert Lamorisse in 1957 as La Conquête du Monde, RISK is a turn-based board game for two to six players. There exist many variations but the standard version, which this paper is concerned with, is an adversarial environment where players vie to control a fully observable board portraying a static geographical map of the Earth.

2.2.1 Equipment

Figure 2.2: Risk Equipment

The game consists of three pieces of equipment:

- A board portraying a geographical map of the earth.
- Different coloured tokens called *armies*.
- Two sets of regular six-sided dice.

2.2.2 Rules

Games are played following a set of rules which define the purpose of each piece of equipment and how they relate to each other. The following is a brief summary of these rules.

The board is divided into forty two *territories*. Each territory is a representation of a real country on Earth. These territories are partitioned into six *continents* usually corresponding to their real continent grouping.

Armies are placed in countries. If a player places an army in a territory, this declares that the player *occupies* that territory. Players must have at least one army placed in a territory they own at all times. Players can then choose to place as many additional armies as they wish in that territory.

How a player wins largely depends on the *Game mode*. Game modes significantly impact the behaviour of players and is decided by players beforehand. In standard RISK they are:

Game Mode	**Description**
Domination	Players aim to conquer a certain number of territories,
Mission	Each player is given a single unique mission card. A mission card describes a state they must reach e.g. Occupy North America and Africa, for the rest of the game this is their *root goal* which only they know. In order to win they can either complete this root goal or eliminate all other players.
Capital Risk	Each player is given a Capital territory and to win a player must occupy all other Capital territories.

This paper is concerned with the mission game mode **only**, therefore the following sections describe rules only for that game mode.

After an initial setup, each player's turn is split into three distinct phases which always occur in the same fixed order. These phases are:

1. Reinforcement
2. Attack
3. Movement

In each phase a player performs atleast one discrete *action* which helps to further the players goals, thus forming a sequential task environment.

2.2.2.1 Initial Setup

The game begins with an an initial setup, it involves:

- Territories being divided equally between players.

- Players being given a starting number of armies which is inversely proportionally to the number of players.
- Players distributing their starting armies over their territories.
- Each player being given a mission card.

2.2.2.2 Reinforcement Phase

At the start of a player's turn, they receive *reinforcements* in the form of additional armies. The act of placing an army in a territory is called *reinforcement*. The number of additional armies received is based on the number of territories a player occupies and whether they occupy any continents.

Occupied Continent	**Number of Bonus Armies**
Asia	7
North America	5
Europe	5
Africa	3
Australia	2
South America	2

Table 2.1: Occupied Continent Army Reinforcement Bonus

2.2.2.3 Attack Phase

Once a player has distributed their armies from the reinforcement phase, they can choose to *attack*.

Territories occupied by a player that contain more than one army can attack adjacent territories occupied by any other player. An attack consists of several *battles*. The outcome of a battle is decided by means of rolling two sets of dice, thus making it a stochastic environment. One set is rolled for the *defender* of the territory and the other for the *attacker*.

The number of dice each player receives for a battle is dependant on the number of armies placed in each of the players respective territories. The defender receives a die per army up to two armies. The attacker receives a die per army upto three armies not including the single army they are required to have in that territory while occupying it.

The general rules of engagement are, dice rolls of each player are compared on a one to one basis in descending order of values. The player with the lower value at each comparison loses a single army, if the die are equal the attacker loses an army. The number of comparisons per battle are set by the number of dice the defending player has rolled. The attacking player can commence as many battles as they wish during an attack provided they have more than one army in the attacking territory.

An attack of a territory has three outcomes:

- A *failedAttack* by the attacker as they have only one army remaining in which case they must retreat and a *successfulDefence* by the defender who retains the territory.
- A *failedAttack* by the attacker as they choose to retreat before having only one army remaining and a *successfulDefence* by the defender who retains the territory.
- A *successfulAttack* by the attacker who occupies the territory and a *failed-Defence* by the defender who has no armies remaining and so loses the territory. The attacking player, leaving atleast one army behind, must then move armies from the territory they attacked from into the newly occupied territory.

A player can perform any number of attacks from any territory they own during their turn, provided they have more than one army in the territory they choose to attack from.

2.2.2.4 Army Movement Phase

When either the player chooses to end the attacking phase or can no longer attack because they do not occupy a territory which contains more than one army their movement phase begins.

During their movement phase a player may move armies from one territory to an adjacent territory they own, provided they leave atleast one army in the territory the armies were moved from. This action can only be done once per turn in this phase, after which the movement phase is finished.

After the movement phase has been completed the players turn ends and another players reinforcement phase begins.

2.3 Why Plan Recognition in Board Games

Algorithmic landmark - such as the solution of the game of checkers

In the realm of board games such as chess, there have for many years dominated a number of algorithms such as the:

Min-max Algorithm with Alpha-Beta Pruning

Many games today have a large number of alternative moves are stochastic and have hidden state attributes. "Applying traditional game tree search algorithms designed for perfect information games that act on the raw state representation is infeasible" [ref Monte Carlo Planning in RTS Games], this has been said because the search space for such games is large and that finding the best move would take an unreasonable amount of time (Find a ref for this)

Instead of a brute force method of finding the best alternative the idea of plan recognition offers another method. Use plan recognition algorithms one can weed out unlikely explanations before committing to a search of the

Main benefit is that responses can be personalized.

This issue prompted research into the development of variants of the original algorithms that could cope but this

Using these algorithms people like Kabanza are starting to be able to tackle certain aspects which are harder to model?

3. Design

3.1 Environment Modelling

3.1.1 Actions

In RISK the actions players perform are:

- Attacking a Territory.
- Defending a Territory.
- Occupying a Territory.
- Losing a Territory.
- Reinforcing a Territory.
- Moving armies into a Territory.
- Moving armies out of a Territory.

Each action must be modelled in a manner that contributes towards explaning a player's behaviour.

3.1.1.1 Attacking and Defending

Any territory a player does not own but is adjacent to a territory that a player does own, can be successfully or unsuccessfully attacked. Additionally by occupying a territory a player can successful or unsuccessfully defend it.

Defence can be seen as a 'passive' action because an attack is required before a defence can occur. In that way it is only inconsistent with the explanations it is directly related to,

Attack is a choice and when a player chooses to attack a territory they have chosen not to attack another territory therefore is consistent with the mission that is affects but is inconsistent with all that it does not.

The actions of attacking and defending a territory only ever occur together and so given the previously defined outcomes of a battle, these can be modelled as follows:

MODEL PROBLEM - PLAYERS MAY BE IN A SITUATION WHERE THEY CANT PERFORM A CONSISTENT ACTION

Action	Consistent	Reasoning
SuccessfulAttack	Yes	Arguably the best indication of a players plan in mission card explanations is the territories they attack and successful attacks is in itself the best outcome.
FailedAttack	Yes	Is indictive of a playes intention to occupy a territory though is less significant than a successfull attack.
SuccessfulDefence	Yes	Players investing in the defence of a territory is a strong sign that they have an plan involving that territory.
FailedDefence	No	This is an inconsistent action because a player would not allow a territory to be lost of it is important to their plan.

Table 3.1: Modelling Attack and Defence Actions

3.1.1.2 Reinforce

To *reinforce* a territory is the act of placing an army in that territory. The reinforce actions that a particular player p can perform at any turn t is based solely on the territories that p owns during turn t. The pending set of any player for the reinforce actions is therefore modelled as follows.

For each territory T that p owns at a certain turn t. In p's pending set is an action to reinforce T. If a territory T_{lost} is lost by p (it is attacked and then occupied by another player), its corresponding reinforce action is removed from the players pending set for turn $t+1$. Conversely if another territory $T_{occupied}$ is occupied by P then a reinforce action for $T_{occupied}$ is added to the players pending set at turn $t+1$

3.1.1.3 Movement

Movement is the action of a player p moving armies from one territory T_1 that they occupy, to another adjacent territory T_2 they occupy.

The pending set actions is neighbouring countries where a player has more than one army.

Initially the movement model was based on the attack model where the smaller the number of actions the bigger the weighting should be, this is an incorrect model as the number of territories a player owns increases as a player gains more consistent movement and reinforce actions as they continue to be successful in their goal.

3.1.2 Territory

Territories form a vital part of the RISK environment and it is necessary to define a model for them. Each Territory is modelled as a state and by owning Each state has a number available actions which together form that territories associated pending set.

(Diagram of territory and associated actions)

T = Territory Object

In the form of the pending set of that Territory and the countries name T = TerritoryName, PS

3.1.3 Continent

Each continent C contain's atleast two territories T and so can be modelled simply as a tuple of the name of the continent and the set of territories that are contained in that continent.

$C = \{\text{continentName}, <T_1 \ldots T_n>\}$

3.1.4 Player

The term *Player* can be used inter changeably with the term agent that was introduced in previous descriptions. A data structure of a player must contain a number of things:

Each player has a list of countries they own. Each player has a name. Each player must have a history of active pending sets. Each player must have a list of actions they have taken. Each player must maintain a list of explanations that has been assigned to them.

Each player must be manageable.

The active pending set of an agent is decided *a priori* based on the territories a player owns.

3.2 PHATT Modelling

PHATT was published by Christopher Geib and Robert Goldman in 2009. In their paper they presented the central realization of the PHATT approach being "that plans are executed dynamically and the actions that an agent takes at each step depend critically on the actions that the agent has previously taken".

They went on to introduce a model of plan execution based on the notion of *pending sets* of actions. They defined a pending set as the actions an agent could take based on the actions that the agent had already performed, calling it "actions that are pending execution by the agent".

In summary, plan execution according to PHATT was modelled as the following. An agent would first choose a root goal, then a set of plans to achieve that root goal. Any actions of those plans that had no pre-requisite actions would form the initial pending set of the agent. The agent would then perform an action from the initial pending set. This would result in some actions being added to the pending set and others being removed to form a new pending set. The agent would then continue to perform actions until an outcome such as the the agent concluding the root goal had been achieved.

From this model of plan execution, Geib and Goldman proposed an algorithm utilizing a Bayesian approach to perform probabilistic plan recognition. It computed $Pr(g|obs)$, the conditional probability of a goal g given a set of observations obs, by computing $Pr(exp|obs)$, the conditional probability of a particular explanation exp of the likelihood of an agent having that root goal, given a set of observations obs.

Using Bayes Rule they defined $Pr(exp|obs)$ as:

$$Pr(exp|obs) = Pr(exp \wedge obs) \ / \ Pr(obs)$$

They then (as other practical Bayesian systems do) exploited the equivalent formulae

$$Pr(exp_0|obs) = Pr(exp_0 \wedge obs) \ / \ \sum_i Pr(exp_i \wedge obs)$$

This they described as the conditional probability of the specific explanation exp_0 being computed by dividing the probability of that explanation and the observations by the sum of the probability mass associated with all possible explanations.

3.2.1 Computing an Explanation's Probability

To go on to compute *Pr(exp $\wedge$ obs)* requires three probabilistic features:

- The probability of the root goal.
- The respective probabilities of choosing any sub goals.
- The probabilities of picking actions from the agents pending set.

The probability of a given explanation is then calculated by multiplying together the priors for each goal, the probabilities of each of the sub goals, and the probability of the observed actions being chosen as follows:

$$Pr(exp \wedge obs) = P(goals)P(plans|goals)P(obs|exp)$$

3.2.2 RISK Explanation Modelling

3.2.2.1 Root Goals

The root goals of this environment are the mission cards, these are:

- Occupy Europe, Australia and one other continent.
- Occupy Europe, South America and one other continent.
- Occupy North America and Africa.
- Occupy North America and Australia.
- Occupy Asia and South America.
- Occupy Asia and Africa.
- Occupy 24 territories.
- Occupy 18 territories and occupy each with atleast two troops.
- Eliminate a player.

The above list all the mission cards from the RISK environment, due to the modelling choices which will be presented later, a subset where the mission involved conquering continents were focused on as due to the high level of overlap other missions which would likely hamper predictions.

As mission cards are handed out to players' at random, the prior probability of each root goal is $1/N$ where N is the number of mission cards.

3.2.2.2 Sub Goals

For root goals that involve occupying atleast two continents it follows by definition that the sub goals of that root goal is occupying each continent. Two root goals allow an option to occupy a continent of choice provided they also occupy two particular continents. These two types of root goals therefore make the occupation of any single continent part of atleast one mission card.

3.2.2.3 Explanation Modelling

An explanation must be designed to contain all the necessary data required to compute its probability.

It therefore must contain these features:

A name A list of root goals A list of method choices A list of consistent actions A list of inconsistent actions

One may ask that why store inconsistent actions, the model, some actions are not defence actions of a territory that is not in the mission is not inconsistent but would be if we where to use a blank if not consistent then inconsistent AND why include inconsistent actions because there is less overhead than having to search through an entire list of actions

RG is the root goal MC is method choice CA is consistent action

Explanation data structure is $E = \{$ explanationName, RL, ML, CA $\}$

Given the list of root goals and sub goals, the full list of explanations that will be considered are:

- Occupy Europe, Australia and Africa.
- Occupy Europe, Australia and North America.
- Occupy Europe, Australia and South America.
- Occupy Europe, Australia and Asia.
- Occupy Europe, South America and Asia.
- Occupy Europe, South America and Africa.
- Occupy Europe, South America and North America.

- Occupy Europe, South America and Australia.
- Occupy North America and Africa.
- Occupy North America and Australia.
- Occupy Asia and South America.
- Occupy Asia and Africa.

Each of these are considered as possible explanations of a players behaviour.

3.2.3 Prediction Agent Pseudo Code

In this section is presented the pseudo code of the most significant operations of the plan recognition agent.

3.2.3.1 Building the Set of Explanations

Algorithm 3.2.1: GENERATEEXPLANATIONLIST(−)

```
forall the C ∈ CL do
    forall the E ∈ EL do
        if C is subGoal of E then
            forall the T ∈ CT do
                addAction(ReinforceT) to playerPendingSet
                addAction(FailedDefenceT) to playerPendingSet
                addAction(SuccessfulDefence) to playerPendingSet
                addAction(FailedAttackT) to playerPendingSet
                addAction(MovementT) to playerPendingSet
            end
        end
    end
end
```

Though this is an n operation, it only needs to be done once at the beginning of the game after the map has been loaded.

3.2.3.2 Explanation Computation

Given a list of all environment explanations EL each player is simply allocated a single instance of each explanation. Since we know that the computation of the probability of an explanation requires three features two of which we have already, we then require a function that generates all the available actions a player can perform to facilitate computing the probability of a player choosing that action.

Algorithm 3.2.2: GENERATEPENDINGSET(AT)

```
playerPendingSet = ∅
forall the T ∈ AT do
    addAction(ReinforceT) to playerPendingSet
    addAction(FailedDefenceT) to playerPendingSet
    addAction(SuccessfulDefence) to playerPendingSet
    forall the N ∈ T do
        if playerOwnsN then
            addAction(MovementT) to playerPendingSet
        else
            addAction(FailedAttackT) to playerPendingSet
            addAction(SucessfulAttackT) to playerPendingSet
        end
    end
    return newPendingSet
end
```

After initializing an empty *newPendingSet* which is a data structure that can only contain performable actions in the environment. The above pseudo first loops through each territory T in the list of all the territories AT that a player owns. For each territory a $ReinforceT$ action and a $LoseT$ action is added to the newPendingSet.

Before proceeding onto the next territory another loop is done through the list of neighbours N of that territory with a simple if-then-else statement. If the player owns that territory a $MovementT$ action is added to the newPendingSet, if not then a $OccupyT$ action is added to the newPendingSet, at the end the newPendingSet is returned and can be cached if necessary.

Algorithm 3.2.3: ComputeExplanation(−)

$expProbability = 1.0$
$expProbability = R * expProbability$
forall the $O \in OL$ **do**
| $expProbability = computeObservationProbability(O) * expProbability$
end
explanationProbability = normalizedexplanationProbability
return *explanationProbability*

The above pseudo code details the computation of an explanation. After initialising the float *explanationProbability* we multiply the explanation by the root probability. Now according to the specifications of PHATT we would normally multiply the method choice probabilties next but since each player has the full set of explanations we have eliminated the need for this term as we measure each explanation against the observations.

Using the method computeObservationProbability, the last loop computes the probability of choosing each observation O by the player from the list of observations OL.

The computeObservationProbability method is a vitally important operation as it computes one of the three probabilistic features required by the algorithm.

3.2.3.3 Equal Distribution

NOT COMPLETE

This is where each feature of an explanation is modelled as equally likely amongst its alternatives.

3.2.3.4 Simple Weighted Distribution

NOT COMPLETE

Following an idea found from an earlier paper by Geib[ref] which proposes a weighting towards consistent actions [ref - new model of plan recognition] that contribute to a plan. A different method of assigning probabilities to pending set actions was deduced.

Actions are separated into consistent and inconsistent actions. A hard coded weight is then equally distributed between each action.

3.2.3.5 Proportional Weighted Distribution

NOT COMPLETE

Write up explanation of new method of probability distribution.

3.2.3.6 Example of Operation

This can be clarified with an example.

As we know the probability of being a mission card is uniform and though there are six mission cards, two of those have four derivatives resulting in 12 explanations of a players behaviour. Thefore the probability of a single explanation is 1/12.

There are no method choice probabilties as at all times we consider the full set of explanations.

Imagine the risk map every territory has a border with every other territory

That the probability of a player choosing to attack any territory is uniform.

3.2.4 Conceptual Issues

Another difference is that I do not store or manipulated derivative trees as the original paper includes in the pseudo code it presents, this is due to hard coding derivative trees into explanations.

Main change is that PHATT works in an action space whereas I have had to adapt the implementation to represent the state of the 'risk world' for the algorithm to operate.

Narrowing large search space Initially the model was based on the plan library proposed in the PHATT paper. This was not good therefore utilized the constraints of countries to narrow the number of possible explanations and actions?

Calibrating Mechanisms of algorithm Uniform distribution Simple weighting based on reasoning about implications of an action to an explanation Weighting based on number of consistent and inconsistent actions available to a player when an action is performed.

Manual Simple Weighting Positively weighted probabilities towards consistent actions and negatively towards inconsistent actions

Initially had a simple system of choosing a weight to distribute

3.2.4.1 Probability Assignment

Finding Equal Distribution does not work very well for simple explanations

so made up:

Weighted Distribution based on consistent and inconsistent actions

Though question is How to assign weights to consistent and inconsistent actions?

Total number of actions are counted, and are separated into inconsistent actions and consistent actions and a weight is split among inconsistent actions and consistent actions. Problem is that it seems this system is too sensitive.

Because of sensitivity had to make up a more evenly weighted automatic system.

3.2.5 Summary

The summary for this section is

4. Implementation

4.1 Modifications to Open Source Project

Initialised plan recognition agent and processing class within

Initialisation of plan recognition agent and processing class in risk game.

Code that injected events into processing which passed it to the plan recognition agent in various places namely:

- Player Initialisation
- Player Removal
- Territory Occupation
- Attacks
- Army Movements

4.2 System Construction

The architecture of the plan recognition agent was built around an existing open source RISK game developed by Yura Mamyrin (verify name) and (anyone else) in (year?)[ref - to website?].

Initially the architecture was designed as a distributed system where the system would store its own subset of data structures from the RISK environment.

Agent data was generated such as countries they owned. Agent explanations computed by the plan recognition agent based on events from the environment.

The reason for this choice was the simplicity of its implementation at the time.

(Diagram of data structure of local copy)

Adding data to this hard coded data structure proved to be overly time consuming and therefore was changed to an automated system of the generation of a data structure allowing the plan recognition agent to work on any RISK map that could be successfully played in the open source game.

Through the modelling process the realization that I could utilize the data structures of the developer rather than storing a copy within the plan recognition agent advantage is less overhead, disadvantage is that this makes the agent more dependant on the methods of the developers implementations.

was could use mirrored those that had already been defined by the developers of the open source game.

4.3 System Architecture Concepts

4.3.1 Event Driven System

Plan recognition agent receives event objects that are fired from the previously mentioned modifications to the original source code. These event objects contain important information from the environment which the plan recognition agent uses to compute the likelihood of explanations.

4.3.2 Component Based Class Architecture

Classes are modelled as an aggregation of components allowing

4.3.3 Google Guava Libraries

Extensive use of the freely available Google Guava libraries to allow the prediction agent to operate concurrently with the game as well as more efficiently in its operations.

4.3.4 Replaying Games

Used an inbuilt tool by the developer of saying "Debug Logs" which could be used to reproduce a game exactly, this allows the algorithm to be tested with different parameters on the exact same game thus ruling out any environmental difference when analysing data.

4.4 Summary

5. Evaluation

Two questions to answer,

1) Does it work?

It performs better than randomly guessing but still at a low probability

2) If not why?

Nature of game, simplicity of model.

Three types of games to test the algorithm in:

5.1 Experiments

5.1.1 AI Games

The open source project has built in AI to play for the mission game mode. This provided an opporunity for the collection fo a large number of data samples to test the performance of the plan recognition agent.

5.1.2 Constrained Play

Where players are asked only to perform actions that are consistent with their root goal.

This is defined as given a situation a player can only:

attack a country that is directly consistent with their plan or a country that is on the shortest route to a country that is directly consistent with their plan. reinforce a country that is directly consistent with their plan or a country that is on the shortest route to a country that is directly consistent with their plan. move armies to a country that is directly consistent with their plan or a country that is on the shortest route to a country that is directly consistent with their plan.

Why did I choose to do this?

To establish that the algorithm works given that players behave only a consistent manner with the mission they have been given.

5.1.3 Free Play

Where players are asked to play as normal.

5.2 Experimental Format

Explained the rules then

There are two ways to win in mission RISK, one is to eliminate all the other players, the other is to complete your mission.

Explained to players the rules of the game

5.3 Data Collection

How did I get the data?

After the game is finished I automatically generate a log file that allows me to reply a game exactly and a csv file recording features such as:

a players actual mission player explanation probabilites over time which player won and lost

Used a tool created by developers to replay games in-order to be able to re-test the algorithm with changes in modelling/calibration.

For games of A.I. players used a program called Autohotkey which allowed me to create macros which automated the data collection process.

Using an system built by the original developers was able to easily reproduce a game for the purposes of evaluating the algorithm.

Questions to answer:

What did I tell players?

Explained to players the rules of the game, that they had two ways of winning the game, either by eliminating their opponents or by completing their mission card. Had the mantra, "play to win"

Experimental De-briefing, asked players to give a summary of what they had done in the game, recorded this down to use for comparison purposes at a later point.

Used python scripts to analyse the csv files.

5.4 Experimental Findings

Sampling of probability needs to be more frequent rather than just at end of term as many actions occur between turns and the game may end in the middle of a lot of actions during one turn. Players do a lot of significant things in one turn.

Players who spend a game fighting over a single continent raise the probability of all explanations associated with that continent and the result is a prediction of several explanations being equally likely. This also applies to fighting over two continents that are part of a three continent explanation.

Misclassification by Association - Explanations appear likely because players do things related to them even though they haven't done anything in one of the continents e.g Europe SA, Asia appears likely even though there has been occupation of territories in Europe because of lots of activity in SA and Asia. This issue is slightly negated by how I have modelled the attack actions probability contribution to explanations.

Main questions to answer.

Whether is converges?

It does converge.

How fast it converges?

Converges quite quickly after the attacks begin - use data from average data over time.

Players don't ignore their mission card like I first thought, as long as the mission seems easier to complete than eliminating all other players then players will try achieve their mission.

5.5 Outcomes

In evaluation DO NOT JUST RELY ON MEASUREMENTS! DISCUSS OUTCOMES! MAKE INFERENCES FROM DATA!

THE WHOLE PURPOSE OF ACADEMIA IS TO LEARN, SHOW THAT YOU HAVE SOMETHING TO CONTRIBUTE TO THAT AIM TO LEARN!

Like what did you find hard? What did you do that you found easy? What did you do that you found hard?

Would you recommend doing something, what would you recommend not doing?

TALK ABOUT THE OUTCOMES OF YOUR WORK!

Inherent issues with PHATT is unable to deal with deception[ref] and often players must perform actions unrelated to a plan to be able to complete their plan e.g survive, have to conquer other non-relevant continents, this confuses the algorithm.

The problem of prediction by association, this is where an explanation which involves a continent that a player has not done anything at all to becoming likely because the explanation contains two other continents that the algorithm has observed many things happen to.

How to decide weighting of

How to deal with this problem?

Using the plan recognition software given a programs a plan to detect how well a program performs a plan.

Using genetic algorithms to optimize this by looking at the numbers returned by the plan recognition algorithm and choosing the best configuration that survived and won.

5.6 Critism

Essentially attack has been modelled following PHATT but the rest of the actions are simple given a weighting of 0.98 for inconsistent and 1.0 for consistent. This attack is the most significant action in the vironment and this we can intuitvely tell is not true.

6. Conclusions

6.1 Future Work

The application of more sophisticated computation models based on the idea of generating a pending set from the state of the world then performing calculations with that pending set.

Only attack actions have been modelled in a matter

Probability model changed to proportion of consistent actions * 0.1 0.9 - proportion for consistent probability 1 - proportion for inconsistent probability

Due to time constraints did not use it. Why?

CRITICISM - ASSUMPTIONS ARE NOT ALWAYS TRUE BECAUSE OF NON-DETERMINISTIC NATURE OF BATTLES! AND OF MULTIPLE ATTACKS PER TURN FOR PLAYERS

6.2 Final

I would dare to say that I believe that the benefit of such systems would be such much that they would be considering cheating if used in events such as official tournaments.

7. Appendix

7.1 Class Diagram

Write up class descriptions with a diagram here

7.2 Experimental Results

Findings

Table of correct predictions and incorrect predictions in each game type.

Graph of the line of the correct explanation for each agent in a single graph.

GRAPHS to make Probabilities of each explanation over rounds, how to combine into a single graph?

Table of correctness (an percentage of correct predictions by end of game for all data)

Bibliography