Machiavellian Agents: Player Modelling to Deceive and be Deceived

Scott Watson, Andrew Vardy, Wolfgang Banzhaf and Todd Wareham
Department of Computer Science
Memorial University of Newfoundland
St. John's, Newfoundland. A1B 3X5.
{saw104, av, banzhaf, harold}@mun.ca

***Abstract*—This paper explores several augmentations to the previously described Plantagenet model of computer game agents to give agents the deceptic capability to deceive the player and/or be deceived by the player. Augmented finite state machine controllers for agents in a simple role playing game are generated using an evolutionary algorithm. It is demonstrated that the proposed model is a practical option for generating populations of around 30 agents in which constraints on agent behaviour to ensure that actions are consistent with deception can be satisfied in substantial portions of the agent population.**

I. Introduction

Considerable research has been carried out into how to make better control architectures for agents in computer games [1], [2]. Some of these architectures are very sophisticated and boast impressive capabilities. Despite these achievements, very little of this research has made its way into commercial computer games [3], [4]. Previously, the Plantagenet system proposed by Watson et al. [5] addressed an important related issue on which relatively little work has been done — namely, improving the process by which existing agent controllers are made rather than improving the agent controllers themselves.

This paper continues that work by testing various augmentations to that model to support deceptic agents — that is, agents that have a capacity to deceive the player and/or be deceived by the player. Our ultimate goal is to enable Machiavellian agents: agents that have the theoretical deceptic capacity to deceive the player and be deceived by the player and are also perceived by human players as having these qualities. These capacities are enabled by equipping agents with a model of the player's objective that they can update and use in action selection.

This paper is organised as follows. Section II briefly covers background information and related work. Section III provides a summary of the controller generation model that underlies this work. Section IV describes three progressively more complex agent-model augmentations for enabling deceptic agents as well as the set-up of experiments designed to test the viability and performance characteristics of these augmentations. Section V contains both the results of these experiments and a discussion of these results. Finally, Section VI gives our conclusions as well as some planned directions for future work.

II. Background

A. Computer Role Playing Games

Role Playing Games (RPGs) are a subset of computer games that emphasize the development of the character controlled by the player, the importance of the player's character in the game world and the influence that the player's character has on that world. These games often place a lot of importance on carefully crafted storylines in which the player's character has a central role. The game worlds in RPGs can be extremely large in scope and complexity and the player often has great freedom to explore these worlds. RPG game worlds can be inhabited by thousands of non-player characters (NPCs).

B. Agents in Role Playing Games

Typically a significant part of the game experience in RPGs is based on the player's interaction with NPCs. This can vary from very simple interactions based on fighting and defeating a character to more complicated interactions such as conversation, trade or negotiation. In many cases, the social landscape of the game influences the interaction. For instance, a friendly agent will behave differently towards the player than an antagonistic agent. Because of the variety of interactions that are potentially required to be handled by the agent controllers in RPGs, and the scope for future experimentation that this offers, RPGs were chosen as the genre to focus on.

C. Finite State Models of Game Agents

Finite State Machines (FSMs) are models of computation [6] defined by a finite list of states and a finite list of transition rules. Each transition rule gives the possible state transitions for a given input. FSMs can be applied in a large variety of domains. Their strengths include conceptual simplicity, fast execution speeds, and ease of implementation.

Finite State Machines are used extensively in computer games [1]–[3], [7]–[9]. FSMs are easy to test, modify and customize [10]. FSMs can achieve good results but cannot deal with situations not explicitly prescribed for by the developer. Moreover human players are becoming adept at predicting behaviour by learning the rules encoded in the FSMs [3].

Efforts have been made to augment the classic FSM model to increase its functionality. Fuzzy State Machines (FuSMs) have come into fashion to give less deterministic behaviours [10]. Fuzzy logic allows unpredictable behaviours

978-1-4673-7921-2/15/$31.00 ©2015 IEEE

to be generated based on traits of the agents which are modeled as decision thresholds. FuSMs were used in Unreal, S.W.A.T.2 and Civilization: Call to Power [10]. Gruenwoldt et al. attempted to use a dynamic relationship graph to modulate basic FSM behaviour [11].

D. Player Modelling

An extensive literature on player modelling in computer games exists. In their 2011 survey Bakkes et al identify two broad goals of player modelling — "(1) providing an interesting or effective game AI on the basis of player models and (2) creating a basis for game developers to personalise gameplay as a whole, and creating new user-driven mechanics" [12]. They further identify four categories of modelling as being practically applicable to modern computer games — "(1) modelling actions, (2) modelling tactics, (3) modelling strategies, and (4) profiling a player" [12].

Researchers have shown that player modelling can be applied beneficially in computer game genres as diverse as first person shooter games [13], massively multiplayer online role playing games [14], real time strategy games [15], [16] and sports simulation games [17].

E. Deceptic Agents

To the best of our knowledge there is no literature specifically on computer game agents with capabilities to deceive the player or be deceived by the player. Research has been undertaken on deception in agents in more general settings. This includes subjects as diverse as how social agents can limit exploitation by deceptic agents in open environments [18], the modelling of trust and deceit in agent supply networks [19], examination of how deception affects group performance of agents simulating terraforming of Mars [20] and classical abstact cases of deception in game theory problems like the prisoner's dilemma [21]. These works are not directly relevant to the FSM-based computer game agents being used in this paper.

F. Previous Work

To the best of our knowledge there is no previous work pertaining to the evolution of FSMs to control game agents other than Watson et al. [5]. Certainly there is no mention of them in Togelius et al's 2011 survey of procedural content generation [22], Hocine and Gouaich's 2011 survey of agent programming in serious games [7] or Hendrikx et al's 2011 survey of procedural content generation [23]. The modeling of players in computer games is an established area of research, an excellent up-to-date summary is given in [24].

III. Plantagenet System

This section provides a brief overview of Plantagenet (PLayable AgeNT GENEraTor), the controller generation model introduced by Watson et al. [5]. This will facilitate understanding of how this model is augmented in Section IV-E to enable deceptic agents.

A. System Overview

The high level view of Plantagenet and its operation is given in Figure 1. A developer provides an input and output specification and an evolutionary algorithm uses these specifications to produce a set of controllers to be used by the agents in the game.

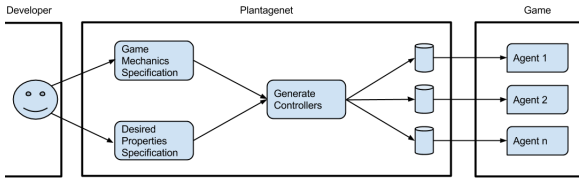

Fig. 1. System Overview

The most important constraint on a set of controllers is for the game that uses them to satisfy *minimal playability* — that is, to make it possible for the player to complete the game. In this context, completion is taken to mean the achievement of some pre-defined victory objective. Examples of victory objectives include slaying a particular NPC, obtaining a particular item or learning a particular fact. If it can be demonstrated that the interaction of the player with a set of candidate controllers results in a minimally playable game, we consider that controller set to be a *playable* controller set.

For every game victory objective, there may be multiple playable controller sets. For games featuring even a moderate number of agents, actions and items this number may be extremely large. It is therefore useful to consider how to compare playable controller sets in terms of desirability. Desirability is a measure of how many 'soft' constraints a controller set satisfies. Soft constaints are gameplay criteria that are desirable but not necessary for minimal playability. The nature of constraints is dependent on the nature of the game but examples might include specifying that no more than 20 agents should accept bribes from the player or the player should have to carry out no more than 20 actions to complete the game. The desirability of a given controller set can be explored by assessing the desirability of the game produced by using that controller set to control its agents. To assess this desirability, attributes of the game that is produced by using the controller set must be examined. These attributes can be inferred from the information gathered when checking if a controller set is playable.

To check if a controller set is playable, it is sufficient to check if there exists a sequence of actions that the player can carry out in order to achieve the victory objective. A sequence of such actions constitutes a *path* to victory. A controller set with at least one path is playable. A controller set may have many paths. The path(s) associated with a controller set can be used to compute certain attributes of the game-play experience that a player would get by playing a game formed using this controller set. The precise nature of the attributes that could be calculated is game-dependent. The soft constraints on the generation process are bounds on these attributes. Examples might include the number of actions the player must carry out to satisfy the victory objective, the number of agents the player must interact with to satisfy the victory objective or the types

of actions that the player must carry out to satisfy the victory objective.

B. Agents

Each agent has initial (possibly emply) sets of items and facts when a game starts. Items are unique and can only be held by one agent, while facts can be duplicated and can be held by many agents at a time. Each agent also belongs to a specified social group when a game starts, and this group may change throughout the game. Social groups are game-specific.

Each agent in the game is always in one of its respective internal states. Agents only change state in response to actions by the player. Their transitions between states are produced by the generation process. The inputs which trigger transition table lookup are of the form $<currentAgentState, playerAction, socialRelation>$ where $currentAgentState$ is the current state of the agent, $playerAction$ is the action the player carried out that the agent is reacting to and $socialRelation$ is the relation between the player and the social group the agent belongs to. The state an agent is in controls the outcome of any actions the player carries out involving that agent.

C. Input Specification

The input specification describes various properties of the game. These are wide-ranging but include the number and characteristics of the entities in the game and the actions that can manipulate them.

D. Output Constraints

Output here refers to the game created by using a given set of generated controllers. Constraints on this output are therefore constraints on the game-play produced. This allows the fitness function (see Section III-G below) to direct the search towards specific game-play objectives. The developer has the freedom to specify as many or as few constraints as they are interested in. There is a trade-off between the expressiveness and conciseness of the specification format. Constraints could include restrictions on state transitions, specification of which transitions are needed and desired frequencies of the actions a player should use.

E. Solution Encoding

If the transition table is considered to be a mapping from input tuple of the form

$$< currentAgentState, playerAction, socialRelation >$$

to output state $newState$, the solution is encoded as the $newState$ component of each mapping designated as required in the input specification. States are mapped to integers and the solution is represented as a list of integers.

F. Evolutionary Algorithm

A generic evolutionary algorithm was employed to evolve the controllers. Full details of the algorithm and its parameters are given in [5].

G. Fitness Function

A minimisation fitness function is employed. Solutions are evaluated by attempting to find all chains of actions (paths) leading to the completion of the specified victory objective that are possible under the agent configuration represented by the solution. Objectives can either be to acquire a fact, acquire an item or attain a state. Valid paths are determined by searching backwards from the victory objective in a depth-first manner. Once all the valid paths have been identified, the properties of the chains are evaluated and penalties applied as appropriate. An optimal solution will invoke no penalties and thus have a score of 0. We deliberately use an exhaustive fitness function to both guarantee that minimal playability will be detected if present and ensure reliable evaluation of soft constraints.. In works where these are not concerns, other evaluation schemes have been developed and applied [24].

IV. Experiment Set-up

Our aim in this work was to extend the model from [5] to support agents with four additional capabilities:

1) Give agents a rudimentary model of the player's objective;
2) Allow agents to select actions based on their model of the player's objective;
3) Allow agents to select honest/dishonest actions based on their model of the player's objective; and
4) Allow agents to infer a model of the player's objective from their observation of the player's actions.

These capabilities were chosen with a view to facilitating generation of agents such that games using those agents can include situations where:

1) Agents can give the player false information that hinders the player's objective. In other words, the player is deceived by the agents.
2) Players can 'fool' agents by acting in a way inconsistent with their true objective to gain the agents' help. In other words, the player deceives the agents.

In this section, we describe a game scenario (Sections IV-A – IV-D) as well as three progressively more complex augmentations to the finite-state agent model that can enable agents with progressively more complex deceptic abilities within this scenario (Section IV-E). This scenario and set of augmentations is the basis for a pair of experiments described in Section V that are designed to evaluate the viability and performance characteristics of our proposed augmentations.

A. Game Scenario

Each agent belongs to exactly one of three groups — Rebel, Loyalist or Undecided. Agents in the Rebel group will support the player, agents in the Loyalist group will support the Pretender and agents in the Undecided group will not offer any support. All agents in the Undecided group are randomly assigned the identification number of a fact or item that, if received, will cause them to change their social group.

In order to win the game, the player must obtain an item 'Crown' which is in the possession of the Pretender. It is theoretically possible that the player can simply attack the

TABLE I. ITEMS FEATURED IN OUR EXPERIMENTS

Item ID	Item Name
1	Crown
2 — (3N+1)	Placeholders for random distribution

TABLE II. FACTS FEATURED IN OUR EXPERIMENTS

Fact ID	Fact Name
1 — (3N)	Placeholders for random distribution

Pretender to gain the Crown but the Pretender has his N Guards with him at all times so in practice such an attack is extremely unlikely to succeed. Rather the player can use the 'Summon' action to call all the agents in the Rebel group to the Pretender's palace to join the battle. All the agents in the Loyalist group will also join the battle on the Pretender's side. The combat strength of all agents is identical. In order for the player to complete the game, they therefore need to expand the size of the Rebel group of agents and try to keep the size of the Loyalist group of agents as small as possible.

In this example, a playable controller set is any set that allows the player to obtain the 'Crown' item. Desirable properties are for no agents in the Loyalist or Rebel groups to have state transitions that lead them out of those groups and for all agents in the Undecided group to have transitions that lead them to change states to the Rebel or Loyalist groups.

B. Input Specification

The input specification describes the nature of the game, the entities present in it and the actions the player can use to manipulate the game state. It is broken down into the following components:

1) **List of Items**: The items used in the experiment are listed in Table I.
2) **List of Facts**: The facts used in the experiment are listed in Table II.
3) **List of Agents**: The agents used in the experiment are listed in Table III.
4) **Item Mappings**: This mapping of items to the agents that hold them can be observed in Table III.
5) **Fact Mappings**: This mapping of facts to the agents that know them can be observed in Table II.
6) **Trade Mappings**: The items that agents in the experiment will accept in exchange for items they hold are randomly generated. Note that these mappings can also be manually specified if desired.
7) **Action Descriptions**: The actions the player can carry out in the experiment are described in Table IV.

TABLE III. AGENTS FEATURED IN OUR EXPERIMENTS

Agent ID	Fact	Item	Note
1	N/A	Crown	Pretender
2 — (N+1)	N/A	N/A	Pretender's Guards
(N+2)—(6N+1)	N/A	N/A	Townsfolk

TABLE IV. ACTION DESCRIPTIONS USED IN OUR EXPERIMENTS

Action	Valid Start States	Valid Post States	Gains Fact	Gains Item	Change Group	Requires Fact/Item
Attack	Idle Fight Flee	Dead Slain	No	Yes	No	-1/-1
Ask	Idle Fight Flee	Answer	Yes	No	No	-1/-1
Trade	Idle	Accept Trade	No	Yes	Yes	-1/-1
Intimidate	Abstain	Answer	Yes	No	Yes	-1/-1
Pick Up	N/A	N/A	No	Yes	No	2/-1
Summon	N/A	N/A	No	No	No	2/-1
Pick Pocket	N/A	N/A	No	Yes	No	2/-1
Declare	N/A	N/A	No	No	Yes	2/-1
Persuade	N/A	N/A	No	No	Yes	2/-1

8) **States**: The states of the agent finite state machines are drawn from the following set: Idle, Flee, Attack, Fight, Lie, Answer, Abstain, Accept Trade, Refuse Trade, Invisible, Dead.
9) **Victory Objective**: In this experiment the player can only ever have one objective, to obtain the 'Crown' item.

C. Output Constraints

To help target the generation towards controller sets that produce the desired gameplay experience, constraints are placed on the search space. These are defined as follows:

1) **Required Transitions**: In this experiment, transitions are required for all combinations of states, actions and social relations.
2) **Valid Transition Mappings**:
   - $<?, Attack, ?> \rightarrow <Flee, Dead, Attack>$
   - $<?, Ask, ?> \rightarrow <Abstain, Answer, Lie>$
   - $<?, Trade, ?> \rightarrow <AcceptTrade, RefuseTrade>$
   - $<?, Intimidate, ?> \rightarrow <Abstain, Answer, Lie>$

   Note that $<?, attack, ?> \rightarrow <flee, dead, attack>$ denotes that for an agent in any state and in any social group, the valid states that they should transition to upon being attacked by the player are flee, dead or attack.
3) **Desired Actions**: None specified.

D. Fitness Function

The penalties that can be applied to a solution's fitness are listed in Table V.

TABLE V. OUTPUT CONSTRAINTS USED IN OUR EXPERIMENTS

ID	Reason	Penalty	Variant
0	No valid chains found	1000	All
1	An agent has a transition that would lead out of the Loyalist group	200	All
2	An agent has a transition that would lead out of the Rebel group	200	All
3	An agent has no transition that leads out of the Undecided group	200	All
4	An agent in the Rebel group should not respond to the player with actions that are unhelpful	200	1
5	An agent in the Loyalist group should not respond to the player with actions that are helpful	200	1
6	An agent in the Loyalist group should respond to player actions that can result in a change of group with an attack action	200	1,2
7	An agent in the Undecided group should change to the Loyalist group if it observes the player attacking another agent within its observation range	200	1,2
8	An agent in the Loyalist group should respond to the player carrying out an 'ask' action with a 'lie' action	200	1,2
9	An agent should update its player objective to gain possession of the item 'Crown' if it observes the player try to change an agent's group to Rebel	200	1,2,3
10	An agent should update its player objective to Pretender gaining possession of the item 'Crown' if it observes the player try to change an agent's group to Loyalist	200	1,2,3
11	An agent should update its player objective to Pretender gaining possession of the item 'Crown' if it observes the player carry out friendly actions toward agents in the Loyalist group	200	1,2,3

E. Machiavellian-enabling Augmentations

We introduce the following three augmentations to extend the base model given in [5] (see Section III) to generate agents with progressively more complex deceptic capabilities:

- **Variant 1:** The agent state tuple is expanded to include the player's victory objective (to acquire the 'Crown' item). Every agent is assigned this on initialisation. This represents a very trivial implementation of a player model. It is perfectly accurate and does not change. Penalties are added to model the agents acting according to their player model as listed in Table V. These penalties are the soft constraints discussed in Section III-A that allow playable solutions to be ranked in terms of desirability.
- **Variant 2:** All augmentations from Variant 1 are included. Additionally the model is changed to encourage agents in the Loyalist group to act in an aggressive way if the player tries to change their group. Additional changes are made to encourage agents in the Undecided group to switch to the Loyalist group if they observe the player in an aggressive encounter and to encourage agents in the Loyalist group to respond with the lie action in response to the player asking which groups agents belong to. The goal of these alterations is to give the impression that agents are trying to manipulate the player into alienating other agents. These changes are modelled with fitness penalties listed in Table V.
- **Variant 3:** All augmentations from Variant 1 and Variant 2 are included. Additionally the model is changed to allow agents to infer the objective of the player. Note that the player's true objective in this scenario is always to obtain the 'Crown' item. The agents can infer that the player's objective is either for the player to obtain the 'Crown' item or for the pretender to retain the 'Crown' item. The goal of this change is to facilitate a new aspect of gameplay by putting in place the mechanisms that would let the player fool agents in the Loyalist group into thinking that the player is on their side in order to obtain truthful information from the loyalist group. This change is modelled with fitness penalties listed in Table V.

To illustrate how deceptic agents based on the augmentations proposed above would function in practice, we give an example run relative to each augmentation-variant. For these examples we will use a population of 7 agents. This population is made up of the pretender, 1 guard, 1 townsfolk in the loyalist group, 1 townsfolk in the rebel group and 3 townsfolk in the undecided group. These agents are summarised in Table VI. Since there are three agents in the loyalist group and one agent (plus the player) in the rebel group, the player should seek to convert at least two of the townsfolk to the rebel group. This will mean their group outnumbers the loyalists and will give them the balance of power and a good chance of victory.

For each variant we list a set of actions the player carries out and the response of the agents. As far as is possible we use the same sequence of player actions in each case given below. This helps to illuminate the difference in agent behaviour between the variants.

TABLE VI. AGENTS USED IN AUGMENTATION-VARIANT EXAMPLES

Name	Social Group	Notes
Pretender	Loyalists	
Louise	Loyalists	Guard
Bill	Loyalists	
James	Rebels	
Tina	Undecided	
Roger	Undecided	
Boudica	Undecided	

1) Variant 1:

1) **Attack** James. James responds by **running** away.
2) **Ask** Louise which social group Bill belongs to. Louise responds by **answering** that Bill is part of the Loyalist group.
3) **Declare** your right to rule to Tina. Tina responds by **changing group** to Rebels.
4) **Declare** your right to rule to Roger. Roger responds by asking for the 'Hammer' item.
5) **Trade** item 'Gold' with Boudica for item 'Hammer.' Boudica responds by **accepting the trade** and by changing group to Rebels.
6) **Trade** item 'Hammer' with Roger to change group. Roger responds by **changing group** to Rebels.
7) Summon rebels to fight for the crown.

In this instance the player is able to recruit all three of the Undecied group to the Rebels group, which then outnumbers the Loyalists 5 to 3. The player therefore has an excellent chance of defeating the Loyalists.

2) Variant 2:

1) **Attack** James. James responds by **running** away.
2) **Ask** Louise which social group bill belongs to. Louise responds by **lying** that Bill is part of the Undecided group.
3) **Declare** your right to rule to Bill. Bill responds by **attacking** the player. Tina observes the fight and **changes group** to Loyalists.
4) **Declare** your right to rule to Tina. Tina responds by **attacking** the player. **Note that Tina's response to the player is different from her response to the same action in Variant 1 because in this variant her social group changed before the player acted.**
5) **Declare** your right to rule to Roger. Roger responds by asking for the 'Good Hunting Locations' fact. **Note that Roger's response to the player is different to his response to the same action in Variant 1 due to the generation process being non-deterministic and this being an equally valid response.**
6) **Trade** item 'Gold' with Boudica for fact 'Good Hunting Locations.' Boudica responds by **accepting the trade** and by changing group to Rebels. **Note that Boudica has a changed transition in this situation, this is an emergent result of the constraints on item/fact mappings and the random change in Roger's transitions.**
7) **Trade** fact 'Good Hunting Locations' with Roger to change group. Roger responds by **changing group** to Rebels.
8) Summon rebels to fight for the crown.

In this instance the player recruits 2 of the Undecided group to the Rebels but 1 of the Undecided group joined the Loyalists. The numbers are therefore split 4-4 and the player has an even chance of defeating the Loyalists. The key difference between the Variants 1 and 2 is that Louise lies, causing the player to try to recruit Bill. Bill starts a fight and Tina's observation of the fight causes her to change social group. In short the player has been *deceived* by Louise.

3) Variant 3:

1) **Attack** James. James responds by **running** away. Louise's model of the player's objective changes to believe that the player supports the Pretender.
2) **Ask** Louise which social group bill belongs to. Louise responds by **answering** that Bill is part of the Loyalist group.
3) Because the player knows that Bill is part of the Loyalist group, they don't try to recruit him.
4) **Declare** your right to rule to Tina. Tina responds by **changing group** to Rebels. **Note that Tina's response to the player is different to her response to the same action in Variant 2 due to the generation process being non-deterministic and this being an equally valid response.**
5) **Declare** your right to rule to Roger. Roger responds by **changing group** to Rebels. **Note that Roger's response to the player is different to his response to the same action in Variant 2 due to the generation process being non-deterministic and this being an equally valid response.**
6) **Declare** your right to rule to Boudica. Boudica responds by **abstaining** from answering. **Note that Boudica's response to the player is different to her response to the same action in Variant 2 due to the generation process being non-deterministic and this being an equally valid response.**
7) **Intimidate** Boudica. Boudica responds by **changing group** to Rebels.
8) Summon rebels to fight for the crown.

In this instance the player is able to recruit all three of the Undecied group to the Rebels group, which then outnumbers the Loyalists 5 to 3. The player therefore has an excellent chance of defeating the Loyalists. The key difference between Variant 3 and Variant 2 is that Louise changes her model of the player's objective after witnessing them act in a hostile way to someone in the Rebels. In short, Louise was *deceived* by the player. This deception causes Louise to tell the truth and in turn the player isn't tricked into attacking Bill and alienating Tina.

V. Results

Testing was done to verify (1) if our objectives could be satisfied at all and (2) what the performance characteristics of the generation process were. For each test, the base model from Watson et al [5] and the three variants defined in IV-E were executed for 30 test runs. All test runs were executed on a desktop 3GHz i5-2500K processor.

A. Viability Results

Viability tests were carried out on a population of 31 agents. The results demonstrating that the variants can produce vaiable output are summarised in Table VII. Each cell represents the percentage of agents that satisfied one of the constraints in Table V for each variant across all the test runs that satisfied minimal playability. Most of these constraints were not completely satisfied for every agent in the population but in all cases significant portions of the population satisfied the constraints. In particular, the results for Variants 2 and 3 showed that satisfying constraints 9-11 in a subset of the population came at the cost of reducing the proportion of the population that could satisfy constraints 6-8.

Variant 1 successfully generated populations of agents that allowed the player to manipulate their social groups to make the victory objective attainable. Variant 2 successfully generated agents with transitions that result in the lie action being used to respond to the player based on the agent's social group and player model. User evaluation will be required to evaluate if this agent behaviour is interpreted as manipulation by humans. Variant 3 successfully generated agents that update their player model. User evaluation will be required to evaluate if this agent behaviour is interpreted as being deceived by humans.

TABLE VII. DEGREE OF OUTPUT CONSTRAINT SATISFACTION

	Base	V1	V2	V3
Minimally Playable	100%	100%	100	%100
Constraint 1	-	95.53%	88.48%	83.5%
Constraint 2	-	95.12%	88.72%	82.12%
Constraint 3	-	100%	100%	97.07%
Constraint 4	-	98.27%	98.6	98.07%
Constraint 5	-	100%	100%	100%
Constraint 6	-	-	100%	98.07%
Constraint 7	-	-	100%	60%
Constraint 8	-	-	100%	60%
Constraint 9	-	-	-	42.5%
Constraint 10	-	-	-	37.81%
Constraint 11	-	-	-	45.13%

B. Performance Characteristic Results

Table VIII summarizes, for the base model and each augmentation-variant, the mean time taken to generate minimally-playable agent-groups of size 7, 19, and 31.

TABLE VIII. MEAN TIME TO ACHIEVE MINIMAL PLAYABILITY (SECONDS)

	Base	V1	V2	V3
7 agents	0.002	0.035	0.037	0.041
19 agents	0.03	0.086	0.101	0.113
31 agents	0.005	0.171	0.177	0.198

Table IX summarizes, for the base model and each augmentation-variant, the mean time taken to generate optimally-desirable agent-groups of size 7, 19, and 31.

TABLE IX. MEAN TIME TO ACHIEVE OPTIMAL DESIRABILITY (SECONDS)

	Base	V1	V2	V3
7 agents	0.127	1.996	1.958	1.984
19 agents	0.240	66.644	14.261	58.807
31 agents	0.621	241.767	159.605	225.892

Table X summarises test results for how successful the generation process was for fixed-time searches. For each specified time, the base model and the three variants were executed and the percentage of runs that satisfied minimal playability during that time limit are given.

TABLE X. DEGREE OF MINIMAL PLAYABILITY SATISFACTION

	Base	V1	V2	V3
5 seconds, 7 agents	100%	100%	100%	100%
5 seconds, 19 agents	100%	100%	100%	100%
5 seconds, 31 agents	100%	100%	100%	100%

Table XI summarises test results for how successful the generation process was for fixed-time searches. For each specified time, the base model and the three variants were executed and the percentage of runs that satisfied optimal desirability during that time limit are given.

TABLE XI. DEGREE OF OPTIMAL DESIRABILITY SATISFACTION

	V1	V2	V3
5 seconds, 7 agents	100%	100%	100%
15 seconds, 7 agents	100%	100%	100%
30 seconds, 7 agents	100%	100%	100%
60 seconds, 7 agents	100%	100%	100%
5 seconds, 19 agents	0%	0%	0%
15 seconds, 19 agents	70%	73.33%	20%
30 seconds, 19 agents	100%	86.67%	86.67%
60 seconds, 19 agents	100%	100%	90%
5 seconds, 31 agents	0%	0%	0%
15 seconds, 31 agents	0%	0%	0%
30 seconds, 31 agents	0%	0%	0%
60 seconds, 31 agents	0%	0%	0%

C. Discussion

In light of the importance of minimal playability as emphasized in Togelius et al [22] — "given the way most commercial games are designed, any risk of the player being presented with unplayable content is unacceptable" — minimal playability can be achieved very quickly 100% of the time for all augmentation-variants (Tables VII and X).

As expected, time to achieve desirability increases both with the complexity of the augmentation-variant and the wanted number of deceptic agents (Tables IX and XI).

These results reveal an unusual trend. In the viability results in Table VII we might expect to see constantly decreasing constraint satisfaction as we progress left-to-right from simple to complex variants. For constraints 4 and 5 this is not the case. This suggests that the input specification itself for this scenario makes satisfaction of those constraints very likely to the point that our evolutionary search has only marginal impact on their satisfaction.

Our experimental results have demonstrated the following:

- The controller generation model can be extended to model agents that hold a model of the player's objective.
- The controller generation model can be extended to model actions that can change an agent's social group.
- The controller generation model can be extended to model actions that are classified as helpful and unhelpful.
- The controller generation model can be extended to model actions that cause agents to change their player model.
- The controller generation model can be successfully employed to generate agent populations up to 31 in size.
- All of this can be done in a timescale of minutes.

These results show that we have succeeded in giving agents the deceptic capabilities we set as desired goals in Section IV.

These results should be considered with the caveat that the augmentations we have implemented are simple in design. We have successfully given agents deceptic capabilities but these capabilities are rudimentary. That being said, starting with simple augmentations has allowed us to more easily interpret the results and to demonstrate that even very limited deceptic capabilties can produce interesting results in game-play terms.

VI. Conclusions and Future Work

In this paper we have expanded on the Plantagenet system previously introduced by Watson et al [5] to facilitate the generation of controllers for agents that model the player's objective. We have shown that controllers can be generated which give agents the capabilities to select actions based on their model of the player's objective, in order to both attempt to deceive the player and be deceived by the player. The model of a player action has been expanded to incorporate classes of actions and actions that change agents's social group. Further, we have demonstrated that the model is a practical option for generating populations of around 30 deceptic agents.

User evaluation is planned to assess how the deceptic mechanics introduced here are perceived by human players. This will address the questions of whether these deceptic agents satisfy our definition of Machiavellian agents and, if so, whether Machiavellian agents lead to a perception of a better gameplay experience by players. Additional experiments and computational complexity analysis along the lines in Wareham and Watson [25] is planned to evaluate how feasible it would be in terms of required runtime and memory to extend the agent model to include higher-order (recursive) modelling of player objectives and modelling of other agents' objectives. To mitigate the rapid increases in complexity that this will likely lead to, a framework where groups of agents share a single model of the player will be explored.

VII. Acknowledgements

This work was supported by NSERC's Discovery Grant Program under RGPIN 283304-2012 (WB), RGPIN 228104-2010 (TW) and by a doctoral award from the Dean of the School of Graduate Studies at MUN to SW. The authors would also like to thank two anonymous reviewers whose comments helped improve the presentation of the paper.

References

[1] D. Cheng and R. Thawonmas, "Case-based plan recognition for real-time strategy games," in *Proceedings of the 5th International Conference on Intelligent Games and Simulation*, 2004, pp. 36–40.

[2] P. G. Patel, N. Carver, and S. Rahimi, "Tuning computer gaming agents using Q-learning," in *Proceedings of the Federated Conference on Computer Science and Information Systems*, 2011, pp. 583–590.

[3] C. Fairclough, M. Fagan, B. M. Namee, and P. Cunningham, "Research directions for AI in computer games," in *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science*, 2001, pp. 333–344.

[4] S. M. Lucas and G. Kendall, "Evolutionary computation and games," *Computational Intelligence Magazine*, vol. 1, pp. 10–18, 2006.

[5] S. Watson, W. Banzhaf, and A. Vardy, "Automated design for playability in computer game agents," in *Proceedings of the 2014 IEEE Conference on Computational Intelligence in Games*, 2014.

[6] J. E. Hopcroft, R. Motwani, and J. D. Ullman, "Introduction to automata theory, languages, and computation," *ACM SIGACT News*, vol. 32, no. 1, pp. 60–65, 2001.

[7] N. Hocine and G. A. Gouaich, "Agent programming and adaptive serious games: A survey of the state of the art," *Unpublished manuscript*, 2009.

[8] I. Szita, M. Ponsen, and P. Spronck, "Effective and diverse adaptive game AI," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, pp. 16–27, 2009.

[9] C. Thurau, G. Sagerer, and C. Bauckhage, "Imitation learning at all levels of game-AI," in *Proceedings of the 5th International Conference on Computer Games: Artificial Intelligence,Design and Education*, 2004, pp. 402–408.

[10] D. Johnson and J. Wiles, "Computer games with intelligence," in *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*. IEEE, 2001, pp. 1355–1358.

[11] M. Katchabaw, D. Elliott, and S. Danton, "Neomancer: An exercise in interdisciplinary academic game development," in *Proceedings of the DiGRA 2005 Conference*, 2005.

[12] S. C. Bakkes, P. H. Spronck, and G. van Lankveld, "Player behavioural modelling for video games," *Entertainment Computing*, vol. 3, no. 3, pp. 71–79, 2012.

[13] S. Hladky and V. Bulitko, "An evaluation of models for predicting opponent positions in first-person shooter video games," in *Proceedings of the 2008 IEEE Symposium on Computational Intelligence in Games (CIG'08)*. IEEE, 2008, pp. 39–46.

[14] C. Darken and B. Anderegg, "Particle filters and simulacra for more realistic opponent tracking," *AI Game Programming Wisdom*, vol. 4, pp. 419–428, 2008.

[15] S. C. J. Bakkes, P. H. M. Spronck, and H. J. van den Herik, "Opponent modelling for case-based adaptive game AI," *Entertainment Computing*, vol. 1, no. 1, pp. 27–37, 2009.

[16] F. Sailer, M. Lanctot, and M. Buro, "Simulation-based planning in rts games," *AI Game Programming Wisdom*, vol. 4, pp. 405–418, 2008.

[17] K. Laviers, G. Sukthankar, D. W. Aha, M. Molineaux, and C. Darken, "Improving offensive performance through opponent modeling." in *Artificial Intelligence for Interactive Digital Entertainment Conference*, 2009.

[18] A. Biswas, S. Sen, and S. Debnath, "Limiting deception in groups of social agents," *Applied Artificial Intelligence*, vol. 14, no. 8, pp. 785–797, 2000.

[19] W.-S. Kim, "Effects of a trust mechanism on complex adaptive supply networks: An agent-based social simulation study," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 3, p. 4, 2009.

[20] D. Ward and H. Hexmoor, "Towards deception in agents," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*. ACM, 2003, pp. 1154–1155.

[21] L. Mui, M. Mohtashemi, and A. Halberstadt, "Notions of reputation in multi-agents systems: a review," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. ACM, 2002, pp. 280–287.

[22] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 3, no. 3, pp. 172–186, 2011.

[23] M. Hendrikx, S. Meijer, J. van der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 9, no. 1, pp. 1–22, 2013.

[24] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player Modeling," in *Artificial and Computational Intelligence in Games*, ser. Dagstuhl Follow-Ups, S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013, vol. 6, pp. 45–59. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2013/4335

[25] T. Wareham and S. Watson, "Exploring options for efficiently evaluating the playability of computer game agents," in *Proceedings of the 20th International Computer Games Conference*, 2015.