Possibilities for Learning in Game Artificial Intelligence
Arild Jacobsen, Sindre Berg Stene, Sule Yildirim
Høgskolen i Gjøvik

Abstract
In our earlier research, we looked into the need for and use of AI in video games. Our survey on the existing literature on game artificial intelligence and our hands-on experience with some of the games which were developed through 1990s up to today have shown that the Artificial Intelligence in commercially available video games has made significant progress over the decades, but one area which commercial games have largely ignored is the use of learning AI. Meanwhile, game artificial intelligence research continues to look into and create examples of using such artificial intelligence techniques, e.g. reinforcement learning, evolutionary algorithms, in academic games. At the moment these techniques are largely employed only by game artificial intelligence research; however, considering that game environments in commercial games are becoming more dynamic and unpredictable, one would think that these techniques will be more capable of handling such environments and as such would be more widely used by commercial developers. Even so, it is still rare that commercial game developers employ these techniques in their games. In this paper, we will investigate the reasons behind that by looking at the possible benefits and problems, as well as the current state of learning in game artificial intelligence.

Introduction
Alexander Nareyek in his paper on Artificial Intelligence in Games (Nareyek, 2004) indicates that the percentage of CPU (central processing unit) cycles that developers are allowed to burn on AI computations is steadily growing. This might be because the speed of graphics cards has been increasing much faster than that of the CPU, which frees up lots of resources. These additional resources might open up possibilities for more sophisticated AI, for example the use of learning techniques.¹ With that purpose, in the section "The history of learning in video games", firstly we examine the status quo of several academic and commercial games in terms of how much learning is present in them, what learning is useful for in these games, and the nature of the techniques behind learning in them. Secondly, in the section "What can game agents learn?", we list actions and situations that the game agents (can) learn in games. Thirdly, having examined the status quo and the possible actions for learning, under the title "The benefits of using learning in games", we look into more general reasons for using learning in games and the benefits that might come from doing so. Are there disadvantages that overshadow the benefits, causing game developers to avoid using other AI techniques than a few common ones? What are these disadvantages? Under "The issues and techniques for implementing learning in games" title, we focus on the techniques that could be used to implement learning in commercial games, and also examine the problems with making use of these techniques. Next, in the last section before the conclusion, titled "Evaluation of game agents' learning capabilities: a commercial perspective", we look at real-world examples, first mentioned in the history section, of commercial games using learning techniques in light of the benefits and problems described in the previous sections. Finally, we bring our observations together to present our conclusions.

¹ Definition of Machine Learning: A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$ (Mitchell, 1999)

This paper was presented at NAIS 2009; see http://http://events.idi.ntnu.no/nais2009/
Overall, these sections try to find an answer to the following dilemma:

On one hand, there is a belief that learning can offer a number of benefits to the
game developers and the players, on the other hand, there are only a few game titles that
have learning as a feature in them. Where is the problem?

In this paper, by machine learning, we refer to the techniques such as decision tree
learning, artificial neural networks, Bayesian learning, genetic algorithms, learning sets
of rules, inductive learning, analytical learning, reinforcement learning etc. Learning is
needed for self improvement. Learning leads an agent to change its existing
behaviors/actions that would be employed in a given (game) state with new ones. As a
result an agent who believes in a rule might modify that rule and own a new rule after
its learning. For example, an agent who believes in a rule such as "if an enemy shoots
me, I will die", might modify it, after some experience, with the rule "if an enemy
shoots me and I cannot take cover, I will die". Rules, those do not change over time and
those that are not inferred as a result of a reasoning or a planning process lead to
reactive behaviors. Also, although there is a difference between learning and adaptation
in terms of biological processes, for game AI, we assume them to be both learning.

The history of learning in video games

There are several examples of game AI encompassing some amount of learning and
reasoning capabilities in commercial games already released. (e.g. learning capability of
creatures in Black & White). The earliest of our examples dates back to 1996
(Creatures), while the rest stay within the time frame of early 2000's till today.

In (Grand et al. 1997), an interactive entertainment product based on techniques
developed in Artificial Life and Adaptive Behavior research (e.g. Brooks and Maes
1994) is discussed. The authors indicate that the product, called Creatures, allows
human users to interact in real-time with synthetic agents which inhabit a closed
environment. The agents, known as “creatures”, have artificial neural networks for
sensory-motor control and learning, artificial biochemistries for energy metabolism and
hormonal regulation of behavior. Both the network and the biochemistry are
“genetically” specified to allow for the possibility of evolutionary adaptation through
sexual reproduction.

In (Evans, 2001), learning is stated to cover the following characteristics,
specifically for the game Black & White:

• Learning that (e.g. learning that there is a town nearby with plenty of food).
• Learning how (e.g. learning how to throw things, improving your skill over
time).
• Learning how sensitive to be to different desires (e.g. learning how low your
energy must be before you should start to feel hungry).
• Learning which types of object you should be nice to, which types of object you
should eat, etc. (e.g. learning to only be nice to big creatures who know spells).
• Learning which methods to apply in which situations (e.g. if you want to attack
somebody, should you use magic or a more straightforward approach?).

Learning can be initiated in a number of very different ways (Evans, 2001):

• From player feedback, stroking or slapping the creature (A Non Player Character
- NPC²).

² A Non Player Character (NPC) is a character not controlled by the player. It can be
either a companion or an enemy to the player.
From being given a command: when the creature is told to attack a town, the creature learns that that sort of town should be attacked.

From the creature observing others: observing the player, other creatures, or villagers.

From the creature reflecting on his experience: after performing an action to satisfy a motive, seeing how well that motive was satisfied, and adjusting the weights representing how sensible it is to use that action in that sort of situation.

In the Nero project, neural networks are used as controllers for NPCs and NPCs are evolved for learning combat behavior (Stanley et al., 2005b). The agents begin the game with no skills and only the ability to learn. In order to prepare for combat, the player must design a sequence of training exercises and goals. Skills that players have taught the agents include avoiding an enemy, dodging turret fire, and navigating complex mazes without any path-planning algorithm, that is, agents learn to navigate mazes on their own. The machine learning present in the game developed for this project is interactive since it is the player who designs training exercises for the agents, assigns it to them and watches the agents while learning.

In (Orkin and Roy, 2007), conversational virtual agents are envisioned as collaborators and communicators with humans. This article introduces a computational model of a common ground called a Plan Network, a statistical model that encodes context-sensitive expected patterns of behavior and language, with dependencies on social roles and object affordances. The authors specifically describe learning the Restaurant Plan Network from data collected from over 5,000 game-play sessions of a minimal investment multiplayer online (MIMO) role-playing game called The Restaurant Game. Their results demonstrate a kind of social common sense for virtual agents, and have implications for automatic authoring of content in the future.

Several commercial digital games have already made use of machine learning to enhance their AI, using these techniques during the development of the game to develop their AI agents and releasing the game with the resultant AI as static constructs. (Galway et al., 2008) This approach is referred to as offline learning, whereas the use of machine learning incorporated into the actual game-play is called online learning [also referred to as in-game learning (Stanley et al. 2005a)]. Both Colin McRae Rally 2.0 and Forza Motorsports made use of offline learning to train their driver AIs, although with somewhat differing goals; in Colin McRae Rally 2.0, the AI was trained using neural networks in order to drive similar to how a human would, as well as achieving a competitive level of competence (Generation5.org, 2009a). Whereas in Forza Motorsports, not only were AI agents trained prior to release to create opponent AI, but the players were allowed to train their own personalized agents which then emulated their specific driver's style (Vagamelabs.com, 2009), a task for which a learning AI seems more appropriate than one employing traditional techniques.

Galway et al, 2008 also make mention of online learning used in actual commercial digital games, of which there has only been a handful, including Creatures and Black & White. However, the point is made that the game-play in these games is heavily focused on the machine learning being controlled by the player, using machine learning as a separate feature, rather than using it to merely enhance the traditional game-play of an existing genre.

From this retrospective on the use of machine learning in digital games, we find that the use of learning techniques is quite limited, more so if we consider only commercial games, and even more so if we consider only online learning. As we move on, we will
look at learning techniques in greater depth, referring to the examples mentioned here where appropriate.

**What can game agents learn?**

At the most basic level, a learning agent alters its behavior to achieve optimal results. In practice, there are some additional considerations; how easy a behavior is to learn, how mutable an agent's behavior is allowed to be, as well as the sometimes difficult question of what is optimal. Unless a developer intends that a human supervisor makes judgments on whether a behavior is good or bad, some programmatical approach must be defined to determine the relative worth of a behavior. Depending on the situation, this can be as easy as whether the agent possesses a larger amount of some resource as a result of the behavior, while in others it might be so ethereal as how pleased the human player is. A learning opponent agent could possibly learn how to beat the human player every time, but while that could be considered optimal behavior by an academic researcher, for a game developer, the optimal behavior would be considered offering the right amount of resistance to challenge and entertain the player and subsequently lose. Furthermore, whether a behavior had good or bad results might be complex, dependant on a number of actions performed by the agent. This can make certain behaviors harder to learn than others as they depend on performing several actions which have no beneficial results until the last action is performed, which could lead to the agent quickly learning not to perform those first actions, since they offer no benefit, and therefore never learning the composite behavior. Finally, the greater the freedom an agent is allowed in modifying its behavior based on its learning, the greater the chances of unexpected behavior, an issue we address in a later section.

Examples for learning in games are learning paths, learning combat behaviors e.g. learn to jump, avoid fire, approach an enemy, chase an enemy, learning to avoid objects/obstacles, learning to adapt to conditions (Maslow, 2002) that cannot be anticipated prior to the games release, such as the particular styles, tastes, and dispositions of individual players. Also a bot in a First Person Shooter (FPS) game can learn where it has the greatest success to kill a player, it can learn how long a player takes to launch his first assault in a Real-Time Strategy (RTS) game so that the AI can decide whether to expand militarily or economically, or it can learn which fruits in an agent's environment are poisonous and can bias its food choice accordingly (Maslow, 2002).

At the most basic level, learning happens by adjusting parameters, like the width of the default "squad" to run through the pathfinder, optimizing the pathfinder for a certain size, or for a certain resolution of path.

Galway et al., 2008 presents an example learning scenario within a multi-agent test bed, entitled FlatLand, in which game agents compete to attain randomly positioned targets while avoiding collisions with each other. This research is conducted by (Yannakakis et al., 2003) and a feedforward neural network (FFN) is used for the generation of a game agent.

**The benefits of using learning in games**

In general, learning can be useful in building up soft goals for a game AI. Possible goals would be to form game agents that are:

- realistic
- able to play like a human
- able to compete
- able to communicate or interact with the player

**Realistic**
Learning is one of the most important aspects of being human; as such, while observing NPCs performing stupid actions, often makes players see them as foolish, if they continue to exhibit this behavior, apparently learning no lessons, it is easy to lose any empathy towards them, regarding them as automatons. NPCs with more human-like behavior and/or better AI skills might lead to a more immersive game-play for the players, as they will likely care more for the game characters and have an easier time becoming emotionally attached to them. This is a good thing from the perspective of game developers, as it aids in creating the suspension of disbelief needed to immerse the player in the game world. There are many techniques that are applicable in order to achieve this empathy for game characters, several of which do not even deal directly with AI, but what better tool to create the illusion of living, learning, human-like entities than having them actually learn?

**Able to play like human**
As opposed to the idea of the "realistic" in-game character, described above, which aids he suspension of disbelief when a player immerses him or herself in the game environment, we have the agent who attempts to act as if it is a human playing the game. The division between the two can be seen quite clearly in the FPS genre. There are opponents who oppose the player while he or she attempts to traverse the levels of a campaign, usually following a storyline, and these should act like humans in the game-world. While in the arena-style game mode commonly referred to as "deathmatch", players compete against other humans, or AI agents, referred to as bots, attempting to play like other humans. Sometimes even playing matches with a mix of human- and AI-controlled opponents. (Hirono and Thawonmas 2009) state that the entertainment value of a deathmatch is directly related to how human-like the opponents are. The fact that a games developer has held an open competition to create the most human-like bot, complete with a reward of several thousand dollars (Botprize.org, 2009a, 2009b) lends some measure of support to this statement. And while Steve Polge, lead programmer of Epic Games warns against making the bots too human: "Humans can be pretty annoying and obnoxious opponents.", he also claims that developers strive for AI which can challenge players with unexpected and surprising plans (Technologyreview.com, 2009).

On the same topic, Will Wright, creator of the original SimCity, as well as the monumentally popular The Sims, remarked "You want to build an emotional model for the agent you're competing with, [...] It's not just about having an accurate aim. It's about creating a bot that simulates a victory dance above your dead corpse." (Technologyreview.com, 2009).

**Able to compete**
Another point is to provide a more interesting challenge, by allowing the AI to adapt to the players behavior; i.e. if an AI has a limited number of tactics, and the player can learn the weaknesses of those tactics, then the player could be able to predict which tactic the AI will employ and exploit the weaknesses of that tactic, possibly reducing the challenge presented by the AI to the point where it provides no entertainment. Endowing the AI with tactics without weaknesses might not be desirable either, as an AI opponent which is too hard to beat will also lower the entertainment value of the game. However, (Galway et al., 2008) states that through the use of online learning,
game agents may be enhanced with a capability to dynamically learn from mistakes, player strategies and game-play behaviors in real-time, thus providing a more engaging and entertaining game-play experience. Essentially, if the game AI was able to alter its tactics over time, in response to a player's exploitation of its weaknesses, it could be possible to present the player with ever-changing tactics, retaining the challenge by preventing the player from simply repeating learned responses, without removing the possibility to analyze and find weaknesses with the AI's tactics.

A less dramatic approach is to have the learning AI evolve its behavior during the development phase, as mentioned in the history section above. While the contrasts between the performance of traditional or learning techniques are less pronounced in this case, there are those that claim learning techniques can perform this task as well as or better than non-learning ones, of course dependant on what kind of task the AI is intended for (Generation5.org, 2009a).

Able to communicate or interact with the player

It might be desirable that an NPC is able to communicate or interact with the player. This kind of interaction or communication can help an NPC to receive feedback from the player about its actions and as a result learn which action it will repeat in future and which actions it will forget about. Such an interaction is present in the game Black & White between the creatures and the player as stated in (Evans, 2001). Also, in the same game, through creatures' communication with the player, the creatures can be given commands such as attacking a particular town and hence this helps them learn that that sort of town should be attacked. In addition to these, companion NPCs can help the player through interaction with the player, e.g. the player can be informed by a companion about a grenade thrown at the player by the enemy in case the player is looking at somewhere else and is not aware of the approaching danger.

Issues and techniques for implementing learning in games

In this section we will refer to existing techniques currently used in game AI, and we will discuss issues that may arise from their use.

For the game AI to function with the game, there needs to be an interface defining the form of the input to the AI and the output from the AI, thus placing constraints on the data going between the AI and the rest of the game. As an example, an AI opponent in an RTS could at the most detailed level, have the entirety of the game state available as input and as its output might be issuing individual orders to each of its units, while at the other end of the scale it might be taking as its input a few high-level variables, e.g. the available resources and the prevalence of enemy air forces, and as its output choose between a limited list of complete behavior strategies. When the interface for the AI has a wide set of possible parameters and controls, such as giving orders for units to move to positions anywhere in a large game environment, the set of possible states is huge.

Any sufficiently advanced AI where the AI interacts in a complex environment is going to have an architecture that can be viewed as a layered structure (Yildirim and Stene, 2009). Lower layers handle most elementary tasks e.g. determining the optimal path to the target, playing appropriate sequences of character animation, or any number of minor updates of behavior according to a long term plan. Higher layers are used for tactical reasoning and selecting the behavior which is compatible with the agent’s present strategy. An example of a higher layer decision could be, should the agent patrol the area, enter combat or run through the map in search of an opponent? (Gryzb, 2005). The code for learning can be part of any of these layers.
It is unfeasible for an AI developer to predict every behavior a learning AI could produce. As such, if the agents use learning techniques not only during the development phase, but also after release, it is expected that they would occasionally behave in unexpected ways. However, while unexpected behavior is sought after, it raises a number of issues for the developer. Foremost among these is the matter of bug testing; no matter the amount of testing, if the AI can produce unexpected behavior, there is always the possibility that the consumer will experience unexpected bugs (Generation5.org, 2009a). With a large set of possible game states influenced by the game AI's decisions, there's a high likelihood that unwanted situations may arise. A common situation is where an NPC under active control by the game AI is unable to reach its optimal position, or is stuck in some loop of behavior, trapped like an animal pacing back and forth in a cage. The game developer will want to avoid the possibility of such situations arising, yet is hard pressed to test every potential game state. Furthermore, the more recent popularity of social sites has increased communication between the players of these games and their potential buyers. In this way, it's not uncommon to find discussions of situations where a player has recorded what the game looks like when he or she has encountered a game state where the AI is performing actions that look unrealistic (Aigamedev.net, 2009), hence increasing the risk involved in using game AI techniques which produce unpredictable behavior.

Earlier we referred to the distinction between two ways in which learning may be utilized, namely online and offline learning. While learning can be seen as a feature of the game-play - allowing the player to see the behavior of the game AI improve over time (online learning) - there's also the option of utilizing learning only in the design and development of a game, and turning the feature off before releasing the game. This is especially useful when using computationally heavy methods, such as subsymbolic methods (neural networks, genetic algorithms), but the option is also open for any AI method where parameters may be adjusted in an attempt to attain optimal behavior or even simply more variations of behavior. The purpose of constraining it to the development process of the game would be to enable thorough testing of any unpredictable behavior before the game is released. This approach alleviates the risk involved in letting the game AI's behavior change over time, since no new behavior should surface after release of the game.

In research conducted by Yannakakis et al. (2003), a feedforward neural network was used for the generation of a game agent controller within a multi-agent test bed. With behavior controlled by this form of neural networks, the game agents compete to attain randomly positioned targets while avoiding collisions with each other. The feedforward neural network utilized a feature vector consisting of the coordinates of a number of the nearest opponent agents and the target point, centered on the game agent, together with the distance to the target point. Training examples were obtained using a game agent controller with a set of near optimal behaviors, created using an artificial potential field designed specifically for the test bed. Evaluating the best performing controllers from both backpropagation and evolutionary learning, the resulting game agents were tested over a number of runs of the simulation, with multiple copies of the game agent being used in each simulation run. Results indicated that the controllers trained using backpropagation were less proficient, though required less computational processing than the controllers generated using evolutionary methods. This suggests that additional applications for neural network based techniques can be found in game AI.

Earlier in the paper, we mentioned that in Nero project, neural networks which are used as controllers for NPCs are evolved for learning combat behavior (Stanley et al., 2005b). After the training (evolution), the agents or squads who are fit enough to enter a
battle are chosen for action in the actual game. More about the rtNEAT method which the project is based on can be found in (Stanley et al., 2005c).

An NPC which executes Finite State Machines to mimic human behavior can learn from its mistakes and build up its own experience to be able to act smarter in time. Equipped with a Finite State Machine (FSM) and some historic data of past states, an NPC should be able to find out which of its actions has been beneficial and be able to consider what other action types it could employ under given circumstances. However, learning actions which is not in its repertoire and learning them on its own would be a real challenge for an NPC especially when it would obviously lack the observational and mimicking by imitation or natural language understanding capacities of a human player. Learning to create new rules, or in this case new states, remains a real challenge. The research in AI towards this direction can also find its application in the game development process, when designing new NPCs.

We have seen that where learning techniques are used, developers usually opt for either offline learning, or only using said techniques to improve elements that do not greatly modify the difficulty of the game. However, having robust learning algorithms improve game-play over time remains a goal. While we have not seen any realtime learning, The Restaurant Game (Orkin and Roy, 2007) is an example of continuous learning that has a major effect on how the player interacts with the AI. While this is not a commercial game, it is a step in the right direction.

In this section, we have pointed out several issues related to using learning in games. These issues include risk factors having to do with generated behavior as well as development time. As a result, the cases where learning techniques have been used in games AI are few, and even in those cases, the learning is usually done offline and tested before release, allowing no changes in behavior after that point. In addition to these issues, it seems to be the general opinion that the techniques they have now do the job well enough, i.e. the problems in current games have satisfactory solutions using traditional techniques at a cost equal to or lesser than learning techniques. As such, they would rather avoid complicating the development process with the issues mentioned above.

Evaluation of game agents’ learning capabilities: a commercial perspective

In this section we are giving an evaluation of games (both academic and commercial) with learning in them from a commercial perspective. As far as commercial titles using machine learning go, there are two titles that are often used as examples in research papers: Black & White and Creatures. The reason for this is quite simple that it is next to impossible to find any other examples. So, machine learning in commercial games is a scarce phenomenon; how does it affect those examples we have?

In Black & White, the player interacts with a Creature, which serves as an aide to him or her throughout the game and whose behavior is based on machine learning. The game-play itself falls fairly easily into the God Game genre, and the presence of the Creature does not change this in any significant way. The player could train the Creature to do tasks that will further his or her goals, but could also choose to ignore it and play the game in more or less the same way. From one perspective, the inclusion of machine learning in the game does not improve the quality of its agent behavior; rather it adds a separate element of game-play, unrelated to the traditional game-play of the genre, where the purpose of the game-play is simply watching the Creature learn. From another perspective, while human feedback is an important part of training a desirable
agent, the Creature can learn from a variety of other feedback, e.g. satisfying its desires, and the AI could be trained entirely by interacting with the game world (though the value a Creature trained in such a way would add to the game is dubious), and if well-trained, the Creature, while not strictly necessary, does mesh with the game-play at large.

In Creatures, the player directs the evolution of artificial life forms, Norns, each of which is controlled by a neural network. Unlike Black & White, the use of machine learning is an integral part of Creatures whose absence would leave the game pointless, but similar to Black & White, the machine learning is not so much used as a tool to enhance some aspect of game-play as it is the game-play itself.

Our examples for offline learning however, do attempt to use the learning techniques to enhance the traditional game-play; creating racing agents who rather than simply following the optimal course, race in a more human-like fashion (Videolectures.net, 2009), as mentioned in our section on the benefits of learning above, while still maintaining a competitive level. Colin McRae Rally 2.0 in particular received good reviews which made particular mention on the quality of the AI (Gamegenie.com, 2009), and on the agents behavior differing from those observed in games not using machine learning (Generation5.org, 2009b).

As it stands, we have found but a few examples of commercial game titles using learning techniques, and in those cases where online learning was used, it would not be hard to make the point that learning techniques were not used to challenge traditional techniques and create a better AI, but rather as a game-play element of its own. On the other hand, offline learning has been implemented to good effect in several commercial game, complementing traditional methods without compromising ease of use or quality of results (Generation5.org, 2009a). As for the future, we can refer to some notable figures in game development who mention learning as a feature they expect to see in the future of AI in games, e.g. (Imperial College, 2009), as well as the plethora of supporters who have it out with the detractors on message boards on a daily basis. It seems likely that machine learning will play a part in the future of game development, but it might not be as a complement to, rather than a replacement for the traditional techniques.

**Conclusion**

In the introduction, we posed the following dilemma:

On one hand, there is a belief that learning can offer a number of benefits to the game developers and the players, on the other hand, there are only a few game titles that have learning as a feature in them. Where is the problem?

As we see it now, there are actually a number of benefits to using learning. However, there are also issues relevant to using learning in games. The purpose of writing this article is to provide an insight into these issues and challenges related to the use of learning in games. The reasons for not implementing learning in games as we find them are the challenges of unpredictability, complexity (i.e. difficulty of implementation) and resource requirements. Ironically, though the goal of learning is to produce new behavior, and new behavior would be unpredictable, otherwise it wouldn't be learning, one of the big issues with using learning in commercial games is exactly that unpredictability.

Although there is much academic research on the use of learning in games, it is important that this research presents solutions to these challenges such that the solutions
are commercially viable. While it is hard to predict what may be viable in the future, further research in this area should consider the following questions:

- Which learning techniques that are known today are viable for use in commercial games, when we consider the challenges that need to be overcome?
- Which paths of research might be the ones that will lead to commercially viable machine learning in digital games?
- Do the benefits of using learning techniques make up for the difficulties of their implementation?
- May a game environment be suitable for testing learning techniques, and if so, which learning techniques and what kind of game environments?

In this way, commercial game development should be able to benefit from the research done in game AI learning techniques or even the topic of learning in general. If the use of these techniques should become more widespread and developed, it could in return open new possibilities for using these game environments as tools for testing theories of learning.

Yet our overall conclusion is that the use of offline learning has already been used to good effect, while online learning might become useful in the future, though perhaps in less obvious ways than simply replacing the traditional AI agent.

References

K.O. Stanley, B.D. Bryant, and R. Miikkulainen (2005c) Evolving Neural Network Agents in the NERO Video Game, Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05), Piscataway, NJ.