Success, strategy and skill: an experimental study *

Christopher Archibald
Computer Science Department
Stanford University
cja@stanford.edu

Alon Altman
Computer Science Department
Stanford University
epsalon@stanford.edu

Yoav Shoham
Computer Science Department
Stanford University

Microsoft Israel R&D Center
Herzliya Pituach, Israel
shoham@stanford.edu

ABSTRACT

In many AI settings an agent is comprised of both action-planning and action-execution components. We examine the relationship between the precision of the execution component, the intelligence of the planning component, and the overall success of the agent. Our motivation lies in determining whether higher execution skill rewards more strategic playing. We present a computational billiards framework in which the interaction between skill and strategy can be experimentally investigated. By comparing the performance of different agents with varying levels of skill and strategic intelligence we show that intelligent planning can contribute most to an agent's success when that agent has *imperfect* skill.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Plan execution*; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*Games*; I.2.11 [**Artificial Intelligence**]: Multiagent Systems

General Terms

Experimentation, Performance

Keywords

Skill, Precision, Computational Billiards, Strategy, Planning

1. INTRODUCTION

Skill, or the ability of a player to perform the mental and physical tasks necessary to succeed in a particular game or undertaking, has many dimensions. Among these are *strategic skill* and *execution* (or *raw*) *skill*. An example of these distinct facets of skill can be seen in the game of golf. A golfer, often with the aid of a caddy, determines which shot to attempt from a given position. The golfer then attempts to execute the shot as planned. The ability to accurately execute this planned shot is the golfer's execution skill. The ability of the golfer and caddy to decide which shot to attempt, taking into account both the golfer's execution skill and the desired shot result, is the golfer's strategic skill. The success of a golfer depends on both the ability to plan appropriate shots and on the ability to perform shots as planned.

These two dimensions of skill are particularly evident in computational pool competitions. In these competitions a computer agent is granted a certain time limit in which to plan a shot for execution on a virtual pool table. The chosen shot is then perturbed by the addition of noise. This noise represents the agent's raw skill level in the game. In tournaments held to date, the execution skill has been uniform among all players and also quite high, comparable to the skill level of expert human players. We begin with the deceptively simple-sounding question: if the tournament were played at a lower raw skill level, would the game become more strategic or less so?

There has been relatively little previous work done on modeling or reasoning about skill in games. In [10] Larkey and colleagues define skill as "the extent to which a player, properly motivated, can perform the mandated cognitive and/or physical behaviors for success in a specific game". They use the game of sum poker to analyze the impact of different levels and types of skill on player success. Agents with intuitively differing skill levels are presented, and their performance compared. They separate skill in playing a specific game into two separate components: planning skill and execution skill. Planning skill refers to the ability of the agent to plan and decide on a strategy in the game, as well as the quality of the strategy chosen. Execution skill refers to the ability of an agent to realize its chosen strategy in the game. They focus their investigation on elements of planning skill and conclude that skill is an important feature of real games, but that it is messy to represent and reason about skill, even for simple games.

Other work which involves skill [6, 7, 5] has as its main focus the skill of the game instead of the skill of the players. The goal in such work is to classify games either as games of skill or as games of chance. Each of these papers presents a different method for doing this. The common idea throughout these papers is that in a game of skill a player should have more influence on the outcome of the game, while in a game of chance a player has less control. This distinction is of legal importance in communities where games of chance

*This work was supported by NSF grant IIS-0205633-001 and in part by a BSF grant.

Cite as: Success, strategy and skill: an experimental study, Christopher Archibald, Alon Altman and Yoav Shoham, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

require a special license to operate. As in [10], skill is used in these works to include any characteristics of players that can influence their performance in a game.

Our work is closer to that in [10] as we are also concerned with the impact of a player's skill on their success. We differ in our focus on execution skill, whereas Larkey only minimally considered this aspect. Our motivating domains involve a specific form of execution skill: the ability of the player to accurately execute a desired action.

In the remainder of this paper we seek to answer our motivating question empirically, using a computational billiards framework. We vary three parameters: raw skill (represented by the noise added to agents' shots), speed of thought (represented by the amount of time allocated for computing the next shot), and sophistication of strategy (represented by using different planning agents, ranging from the current world champion billiards agent to agents with obviously inferior strategies), and we examine the impact of their combination on the success of the pool-playing agent.

2. BACKGROUND

In this section the setting of the experiment is described more concretely, including detail about the game played in the experiment and background on computational pool.

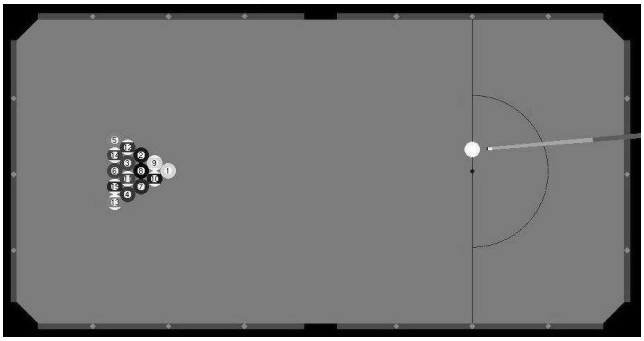


Figure 1: Pool table racked for 8-ball

2.1 Rules of 8-ball

The game played in computational pool tournaments to date, and used as a basis for our experiment, is 8-ball, based on the official rules of the Billiards Congress of America [3]. 8-ball is played on a rectangular pool table with six pockets which is initially racked with 15 object balls (7 solids, 7 stripes, and one 8-ball), and a cue ball (see Figure 1). The play begins with one player's break shot. If a ball is sunk on the break shot, the breaking player keeps his or her turn, and must shoot again. Otherwise, the other player gets a chance to shoot.

For a shot to be legal, the first ball struck by the cue ball must be of the shooting player's type, and the cue ball must not enter a pocket itself. The first ball legally pocketed after the break determines which side (solids or stripes) each player is on. Players retain their turn as long as they call (in advance) an object ball of their type and a pocket and proceed to legally sink the called ball into the called pocket. After all object balls of the active player's side have been sunk that player must attempt to sink the 8-ball. At this point, calling and legally sinking the 8-ball wins the game.

2.1.1 *Winning-off-the-break*

A perfect game by an 8-ball player consists of the player first breaking and then proceeding to sink, in consecutive shots, all the balls of one type (stripes or solids), followed by the 8-ball. This results in the breaking player winning the game without the opponent having a chance to take a shot. We call such a win a *win-off-the-break*. It is clear that an agent's win-off-the-break percentage, which is the fraction of the time they are expected to win-off-the-break, is correlated with superior play. What is less clear is how the win percentage of an agent in a full 8-ball match against an opponent is correlated with the win-off-the-break percentages of the two agents. It seems reasonable that the agent with the higher win-off-the-break percentage would be expected to win the match, but this ignores the possibility that agents could play a defensive style, rarely winning off the break, but forcing the opponent into tough situations, which could then lead to victory for the defensive player. Reliably establishing that win-off-the-break percentage can be used to predict actual performance in head-to-head matches is a sizable task, one which we intend to pursue in the future. For current purposes we consider the win-off-the-break percentage to be fully descriptive of an agent's success level. This can be thought of as having the agents compete in a single player game where the goal is to win-off-the-break. An agent then beats another agent if it wins off-the-break more frequently. This setting is closely related to more single agent games like golf.

2.2 Computational pool

Computational pool is a relatively new game, and was introduced by the International Computer Games Association (ICGA) in recent years as a new game to the Computer Olympiad. Billiards games have several characteristics that make them unique among games played by computer agents, and indeed among games in general [2]. In particular, they have continuous state and action spaces, actions are taken at discrete-time intervals, there is a turn-taking structure, and the results of actions are stochastic. This combination of features is unique and differs from other competitive AI domains such as chess [12], poker [4], robotic soccer [13], or the Trading Agent Competition [14]. Thus, the challenge of billiards is novel and invites application of techniques drawn from many AI fields such as path planning, planning under uncertainty, adversarial search [9] and motion planning [11].

Our experiments were run in a manner similar to past computational pool tournaments [8]. We use a client-server model where a server maintains the state of a virtual pool table and executes shots sent by client software agents on a physics simulator. Each agent has access to an identical version of the physics simulator which they can use to simulate shots internally as part of their shot selection. In the ICGA tournaments, each agent was given a 10 minute time limit per game during which to choose shots.

An action, or shot, in computational pool is represented by five real numbers: v, φ, θ, a, and b. v represents the cue stick velocity upon striking the cue ball, φ represents the cue stick orientation, θ represents the angle of the cue stick above the table, and a and b designate the position on the cue ball where the cue stick strikes, which plays a big role in imparting spin, or "english", to the cue ball. φ and θ are measured in degrees, v in m/s, and a and b are measured in millimeters.

2.2.1 Modeling execution skill

Since the physics simulator is deterministic, zero-mean Gaussian noise is added to each input shot parameter on the server side to simulate imperfect execution skill on the part of the agents. The result of the perturbed shot is then communicated back to the clients.

The ICGA tournaments have utilized two different Gaussian noise models over the years, with the most recent being Gaussian noise with standard deviations of $\sigma_\theta = 0.1$, $\sigma_\varphi = 0.125$, $\sigma_V = 0.075$, $\sigma_a = 0.5$, $\sigma_b = 0.5$. The specific level of noise was a much discussed topic prior to each tournament. The critical issue in this debate was deciding which raw skill level would reward and encourage the type of strategic play desired by the tournament organizers. The lack of justification for this decision was one of the main reasons we undertook the experiment we now describe.

3. DESCRIPTION OF THE EXPERIMENT

In this section we describe the agents used in the experiment and the design of the experiment itself.

3.1 The different agents

In order to test a variety of strategy types and also to ensure that experimental results obtained were not specific to the design of a single agent, we used four separate agents in our experiments. Two of these agents were designed specifically for this experiment, while the other two were variations of the defending champion of the ICGA computational pool tournament. We include brief overviews of the agent designs here, with more comprehensive details in the Appendix.

3.1.1 CueCard *(CC)*

CueCard won the gold medal at the 2008 ICGA computational pool tournament, and as such is used here as the most intelligent agent. It is described in some detail in [1]. From a given table state CueCard considers straight-in shots (shots where the cue ball hits an object ball directly into a pocket), more complex shots (multiple ball and/or rail collisions), and special shots designed to break up clusters of balls. Random shot variants (varying a, b and θ) are attempted for each feasible direction and velocity considered. Each of these shots is simulated with noise between 25 and 100 times, depending on available time. Resulting states are evaluated based on the number and quality of straight-in shots feasible in that state. The value of a shot is the average evaluation of the resulting states of all simulations of that shot. In order to refine the value estimates for the best 20 shots, another level of shots are generated and evaluated beginning from the states which were the results of the simulations of those 20 shots. The best shot found overall, after having its value refined by this second level of look-ahead search, is chosen for execution.

3.1.2 SingleLevel *(SL)*

The second agent we used in our experiments is SingleLevel, which is exactly the same agent as CueCard, except with the second level of search disabled. SingleLevel instead spends that time exploring more shot variants from the initial table state.

3.1.3 OptimisticPlanner *(OP)*

This agent is a good planner, but does not reason effectively about the effects of imperfect raw skill. OptimisticPlanner tries only straight-in shots and assumes that if a shot succeeds then the resulting state will be the same as the result of the noiseless shot. Straight-in shots are ranked using a shot difficulty look-up table. States are evaluated using the same evaluation function used by CueCard. OptimisticPlanner uses a depth-first search approach to refine the evaluation of the top 5 shots from each state, searching as deep as time will allow. Since only the noiseless shot is simulated, OptimisticPlanner can plan further ahead than CueCard and ensure that its entire plan (as much as it had time to search) is theoretically feasible.

3.1.4 MachineGunner *(MG)*

This agent is arguably of low sophistication. MachineGunner does not plan ahead, but does ensure that selected shots are robust with respect to its execution skill level. MachineGunner repeatedly selects an aiming direction at random. It then simulates a full velocity shot in that direction, seeing if the cue ball contacts a legal target ball. If legal contact is made, then MachineGunner attempts variants of this shot with different velocity values, simulating any non-fouling shots 50 times. Each shot is evaluated based only on the number of times it was successful out of the 50 times it was simulated. When enough potential shots have been found, MachineGunner randomly attempts small variations of the best 10 shots. The most successful shot found overall is selected for execution.

3.1.5 Agent comparison

CueCard and the other three agents, each of which was created specially for this experiment, were chosen to represent extreme types of strategic skill. MachineGunner and OptimisticPlanner were designed to be of lower strategic skill, but in two different dimensions. OptimisticPlanner is a very good planner, planning many shots ahead and only choosing shots that are the first steps of long-term successful plans. On the other hand, OptimisticPlanner is ignorant of its own execution skill level. It doesn't utilize knowledge of the raw skill level at which it is competing, but instead only weighs shots using their relative difficulty. MachineGunner, in contrast, utilizes its knowledge of its raw skill level to its advantage, simulating each shot 50 times to determine that shot's quality. Often, when many shots are available, MachineGunner is able to find a shot that is successful on all 50 of its simulations. MachineGunner's shortcomings are in planning. Shots are evaluated based only on their likelihood of success, without considering the strategic possibilities that exist in the resulting table state.

CueCard and SingleLevel are agents of higher strategic skill. They utilize some forward planning and also simulate shots to deal with the effects of noise. The only difference in their strategies is the depth of the forward planning they perform.

3.2 Topics of investigation

As stated earlier, the high level goal of our experiment was to investigate the effects that strategic skill and execution skill have on the success of an agent. In this section we briefly describe how we represented and varied both types of skill in our experiment. We then discuss the specific questions which we sought to answer in this experiment.

3.2.1 Strategic skill

Strategic skill in this experiment was varied in two dimensions. First, we varied the sophistication of the strategy. This was done by using the four agents described in Section 3.1, each of which has a different level of strategic sophistication. The speed of thought was also varied for each of these agents. This was done by modifying the time limit that an agent had for computing its shots during a single game. Additional time during which to plan a shot generally increases the strategic skill of an agent.

3.2.2 Execution Skill

To vary the execution skill of the agents, we varied the standard deviations of the Gaussian noise that was added to each of their shots. Since there are five standard deviations, each of which can be varied individually, there are a large number of ways to change the raw skill of the agents. To simplify the range of possible raw skills, we decided to use the ICGA noise model as a baseline and modify it by scaling each standard deviation by the same factor.

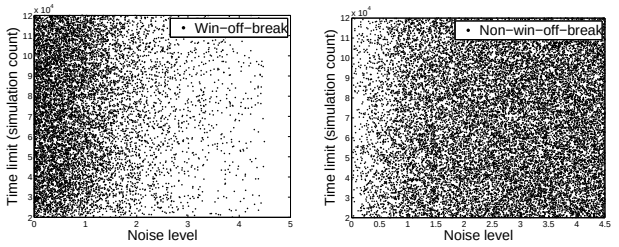
3.2.3 Motivating Questions

The experimental design and analysis were driven by finding answers and providing insight into the following questions:

- How does changing the execution skill of an agent affect its success? It seemed clear from the beginning that less execution skill would lead to less success, but the question still remained: how quickly does performance drop off? Are all agents equally affected by reductions in raw skill?
- Does a high level of raw skill reward a high level of strategic skill? Alternatively, is strategic skill less important at lower skill levels? Does perfect raw skill maximize the importance of strategic skill to an agent's performance?
- If our goal is to identify the agent with the highest level of strategic skill, at which raw skill level should a tournament be held? Can we show that the decisions reached for the ICGA tournaments were reasonable, or does something else make more sense?

3.3 Data generation and processing

Our experiment consisted of having each agent play games with different raw skill levels and different amounts of time available. Since break shots are typically generated offline and fine-tuned for a specific noise level, we modified the rules of 8-ball slightly to ensure that any difference in break shot success at the different raw skill levels didn't impact the overall results. Each agent used the same break shot, and each agent was allowed to re-break until the break shot was successful. Each agent also used the same method for dividing up the total game time limit into time limits for the individual shot[1]. For each game the agent played until either they won the game or lost their turn through a missed shot. After each game we recorded whether or not the agent won that game off-the-break. On the order of 20,000 games were played for each agent, with raw skill levels randomly chosen from 0 to 4.5 times the tournament noise model, and time limits randomly chosen from 20000 to 120000 simulation counts. We used simulation counts as a measure of time for our experiment so that the timing would be consistent across the different machines on which the experiments were run. The physics simulation is by far the most time-consuming portion of any of these agent strategies. An average simulation takes about 3 ms, so our simulation count limits correspond roughly to granting the agents between 1 and 6 minutes of computation time per game. An example plot of this raw data for CueCard is shown in Figure 2. Each black dot in the figure corresponds to a game played by CueCard with that noise level and time limit until it either won or lost its turn.



(a) CueCard's wins (b) CueCard's non-wins

Figure 2: Raw data showing where CueCard did and didn't win-off-the-break

The raw data for each agent was smoothed using convolution with a two-dimensional Gaussian with standard deviations equal to 6000 simulation counts in the time limit dimension and 0.25 in the noise dimension. This gives a smoothed estimate of the win-off-the-break percentage for each agent at each combination of time limit and raw skill level.

4. RESULTS OF THE EXPERIMENT

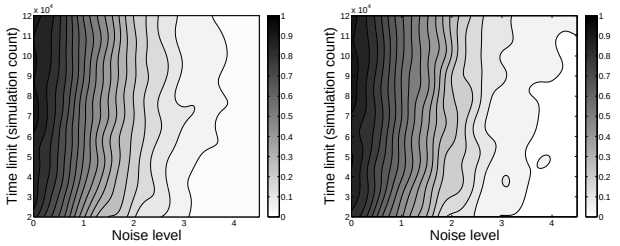
In this section we describe the results of our experiment. In Section 4.1 we discuss the smoothed results and then in Section 4.2 the effects of execution skill on player performance. We then discuss perhaps the most surprising of our results, concerning the relationship between raw skill and the value of strategic skill, in Sections 4.3 and 4.4.

4.1 Winning-off-the-break

After the raw data was smoothed, the resulting dataset contained an estimate of the win-off-the-break percentage for each agent competing at each time limit and raw skill level. Figure 3 contains a contour graph for each agent displaying this smoothed data.

Since a win-off-the-break is simply a binary indicator of agent success, we also investigated using the number of balls left on the table when the agent loses its turn to measure success. For a win-off-the-break this number would be 0, and for a miss on the first post-break shot it would be 7. The smoothed data using this measure of agent success is shown in Figure 4. It is apparent that each of these two data sources display the same high level characteristics, with the main difference being in the resolution of the data in those cases where the win-off-the-break percentage is low. The shape and location of the gradient curves remain consistent for each agent across both types of contour graphs. Since the two data sources are generally in agreement with each

[1] $\text{CurrentShotTime} = \frac{\text{Total time left}}{\#\text{ balls left} + 3}$

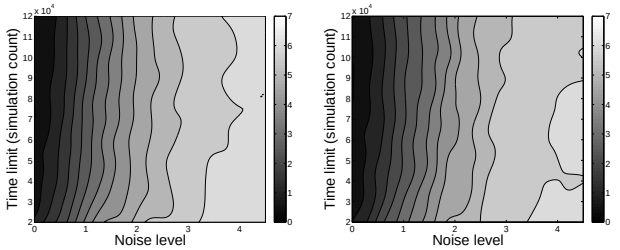


(a) Contour for CC (b) Contour for SL

(c) Contour for OP (d) Contour for MG

Figure 3: Average win-off-the-break percentage for each agent

other, in what follows we utilize only the win-off-the-break percentage estimates.



(a) Contour for CC (b) Contour for SL

(c) Contour for OP (d) Contour for MG

Figure 4: Average number of balls left when turn is lost for each agent

4.2 The value of execution skill

One of our first questions was simply to see how skill impacts the performance of the players. From the contour plots in Figures 3 and 4 it is clear that while increased skill and increased time each benefit the agents, raw skill variations in the range of the experiment (from 0 to 4.5) have a much larger impact on player performance than time limit variations in the simulation count range of the experiment (20000 to 120000 simulation counts). From this data we cannot conclusively say that raw skill is more important than time, since we do not know what happens as we increase the time limit dramatically (to hours or days, say), but certainly for reasonable time limits and raw skill levels this is true.

In Figure 5 the maximum[2] win-off-the-break percentage for each agent at each raw skill level is displayed. This gives a sense of how the agents' expected performance in the win-off-the-break game compare as the raw skill level is varied.

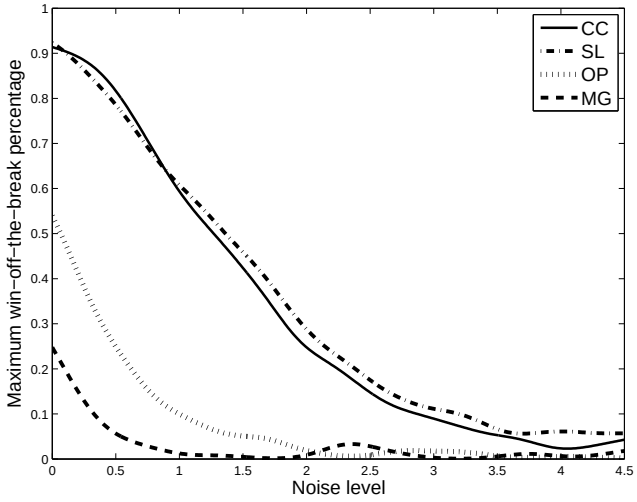


Figure 5: Maximum win-off-the-break percentage for each agent

Clearly, greater raw skill has a positive impact on agent performance in this game. For example, a CueCard agent with particularly low raw skill (a noise level greater than 2.5) would be expected to lose to a MachineGunner agent with perfect raw skill in a win-off-the-break contest. This is despite the gross difference in their levels of strategic skill. This matches the intuition of such situations, that great raw skill can overcome strategic deficiencies. Obviously, raw skill is important for an agent, and an agent can reap large dividends by improving its execution skill level.

4.3 The value of strategic skill: thought speed

Imagine that an agent designer has the ability to increase the amount of time that her agent has to make decisions. For example, perhaps the designer determines that some component of the agent could be optimized in order to run faster and be more efficient. For most agent designs, granting the agent this extra planning time should only increase its performance. But the question remains: how much difference would this added time make to the performance of the agent? Is this difference the same for different agents? Is it the same for the same agent at different raw skill levels? If not, then at which raw skill level is this added strategic skill of most value to the agent?

For a specific raw skill level, we can determine the difference that time can make to that agent's win-off-the-break percentage by finding the difference between that agent's

[2]Over all time limits

maximum and minimum win-off-the-break percentages for each raw skill level. In each case the maximum and minimum were computed over all of the experiment time limits. In this way, we can see how much the strategic skill of an agent, as measured in number of simulations allowed, matters to the performance of an agent at each raw skill level. These differences in performance due to time variations for each agent at each raw skill level are shown in Figure 6.

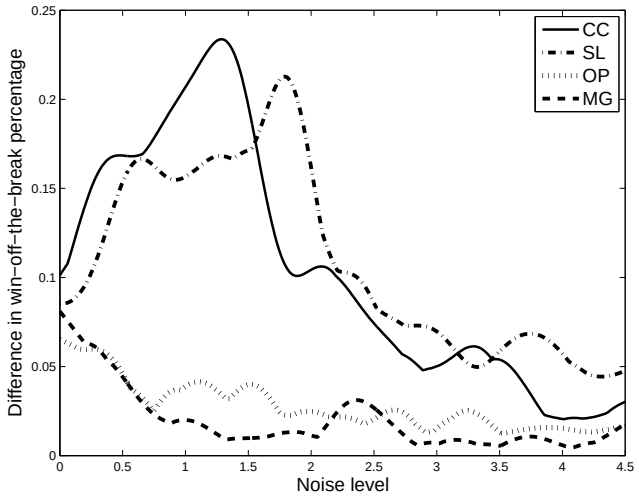


Figure 6: Difference that time makes to each agent at each raw skill level.

We can see that for the more sophisticated agents CueCard and SingleLevel, time is most valuable when some level of imperfect raw skill is present: their plot lines peak at raw skill levels of approximately 1.4 and 1.9 respectively. For the less sophisticated agents MachineGunner and OptimisticPlanner, time is most valuable when raw skill is perfect. We hypothesize that this is largely due to the fact that these less sophisticated agents are not making good use of the extra time within their algorithm. For example, in a situation where MachineGunner finds a perfectly successful shot early on in the random angle generation, extra time will never change the decision of the agent, since no shot could be valued higher.

We return to the problem of selecting a noise level for the computational billiards tournament. The intuition argued by some was that extremely low noise levels would place the highest value on intelligent planning, which was in line with the goals of the tournament to increase the strategic ability of computational billiards software. The fact that for the more sophisticated agents, CueCard and SingleLevel, time is most valuable with some level of noise sheds light on this debate.

For example, imagine that two slightly different programs are entered in the tournament. One is the basic version of CueCard. The other uses the same high level strategy as CueCard, but through innovative techniques has managed to run much faster than CueCard, giving this other agent the ability to simulate more shots that CueCard. It is clear that the more efficient agent has higher strategic intelligence. If the goal of the tournament is to identify the player with the highest strategic skill then this data shows that the ideal noise level for the tournament would be a noise level of approximately 1.4. Holding the tournament without noise, or at a significantly higher noise level would decrease the value of the difference in strategic skill that the more efficient CueCard agent has, and would increase the chance that the original CueCard would win, despite not having the most strategic skill amongst the competitors. The fact that the value of superior strategic skill is maximized at an imperfect raw skill level is somewhat unintuitive, although the raw skill levels that maximize this value are of high raw skill and close to the original tournament noise level (1.0 on our scale), surprisingly justifying the final decision they reached.

4.4 The value of strategic skill: sophistication of strategy

For agents competing in a competition, the objective is to outperform the opposition. In our setting, agents would want to win-off-the-break more frequently than their opponent in a head-to-head match. Consider a setting in which an agent could select the raw skill level at which the competition was played. The question naturally arises: which raw skill level would they pick, so as to maximize their chances of winning? Assuming that an agent knew the win-off-the-break percentage at different raw skill levels for itself and its opponent, it would select the raw skill level at which the difference between the two win-off-the-break percentages was maximized. For the agent with the higher expected win-off-the-break percentage this raw skill level is the one at which the difference in strategic skill between the two agents is of greatest value. In Figure 7 the average difference (over all time limits) in win-off-the-break percentage between CueCard and the other 3 agents is shown for each raw skill level.

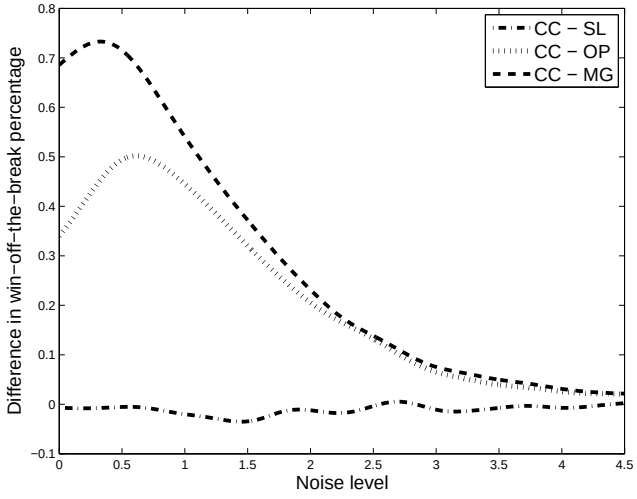


Figure 7: Difference between CC and other agents

This figure shows that CueCard, in facing OptimisticPlanner and MachineGunner, would prefer that the tournament be held with some small level of noise (around 0.5), since this maximizes the chance that CueCard will beat them. In other words, the value of CueCard's superior strategic skill is maximized when the game is played at

this small noise level. Perfect raw skill compensates somewhat for the strategic shortcomings of the less sophisticated agents. Figure 7 shows that CueCard's planning is more robust to increases in the raw skill level of play. Against SingleLevel, an opponent with comparable strategic skill, the raw skill level makes almost no difference, as the two agents are evenly matched at all raw skill levels.

On the other hand, Figure 8 shows the same information for the case of OptimisticPlanner against MachineGunner. Against MachineGunner, the worst performing of the four agents, OptimisticPlanner would prefer the game to be played without noise. This is natural, since OptimisticPlanner's forward planning will be most useful when the noiseless shot predictions are close to what really occurs in the game. Against better opponents, OptimisticPlanner would prefer the game be played at as high a noise level as possible, since this will limit the chance for the opponent to beat him, as both agents will have a very low win-off-the-break percentage .

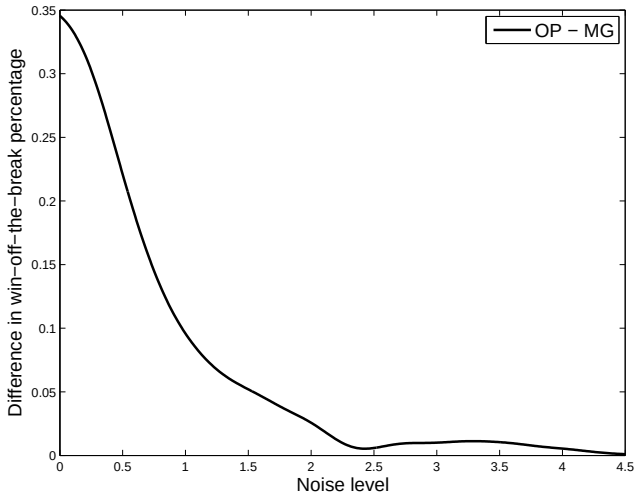


Figure 8: Difference between OP and MG

Interestingly, in considering the value of superior strategic skill, we see again that the highest value is achieved when there is some noise in the game. Agents with different levels of strategic sophistication differ in the rate at which their performance decreases as their raw skill is decreased. Agents with less strategic sophistication could prefer a tournament without noise to a tournament with some level of noise. Assuming again that the goal of a tournament is to identify the agent with the highest strategic skill level, this supports the decision to conduct the tournament with some level of noise. It also suggests the possibility of having separate tournaments at different noise levels, since agents relative performance can differ greatly as the raw skill level changes.

5. CONCLUSIONS

We used a computational billiards framework to experimentally analyze the effects of varying strategic and execution skill on the success of an agent in a game of win-off-the-break 8-ball. Our most striking finding is that superior strategic skill is most identifiable when agents have imperfect execution skill. This result has implications for the design of methods for identifying agents of high strategic skill, and we hypothesize that as a general principle it applies to other domains with the action-planning/action-execution dichotomy.

Two directions for future work appear particularly promising. The first is to investigate skill from a theoretical standpoint. A rigorous treatment of this topic could shed considerable light on the role that raw skill plays in agent interaction. The second, and perhaps more immediate direction, is to use the results here to make informed decisions about the running of future computational billiard tournaments. Careful and justified design of future competitions should lead to the development of new agent strategies which are more robust to changes in noise level, and perhaps even new strategies which can significantly outperform previous agents at certain noise levels.

6. REFERENCES

[1] C. Archibald, A. Altman, and Y. Shoham. Analysis of a winning computational billiards player. In *Proceedings of IJCAI 2009*, 2009.

[2] C. Archibald and Y. Shoham. Modeling billiards games. In *Proceedings of AAMAS 2009*, pages 193–199, 2009.

[3] Billiards Congress of America. *Billiards: The Official Rules and Records Book*. The Lyons Press, New York, New York, 1992.

[4] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence Journal*, 134:201–240, 2002.

[5] P. Borm and B. Genugten. On a relative measure of skill for games with chance elements. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 9(1):91–114, June 2001.

[6] M. Dreef, P. Borm, and B. v. d. Genugten. On strategy and relative skill in poker. Technical report, 2002.

[7] M. Dreef, P. Borm, and B. van der Genugten. A new relative skill measure for games with chance elements. *Managerial and Decision Economics*, 25(5):255–264, 2004.

[8] M. Greenspan. PickPocket wins the pool tournament. *International Computer Games Association Journal*, 29:152–156, 2006.

[9] R. Korf. Heuristic evaluation functions in artificial intelligence search algorithms. *Minds and Machines*, 5(4):489–498, 1995.

[10] P. Larkey, J. B. Kadane, R. Austin, and S. Zamirm. Skill in games. *Management Science*, 43(5):596–609, 1997.

[11] J.-C. Latombe. *Robot Motion Planning*. Springer, 1991.

[12] D. Levy and M. Newborn. *How Computers Play Chess*. Computer Science Press, 1991.

[13] P. Stone. *Intelligent Autonomous Robotics: A Robot Soccer Case Study*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2007.

[14] M. P. Wellman, A. Greenwald, and P. Stone. *Autonomous Bidding Agents: Strategies and Lessons from the Trading Agent Competition*. MIT Press, 2007.

APPENDIX

A. AGENT DESCRIPTIONS

Given a state of the table, each agent uses their corresponding steps to select a shot.

A.0.1 CueCard *(CC)*

1. For each legal ball and pocket, a set of directions φ, each with a minimum velocity v_0, is generated in attempt to sink the ball into the pocket. In this step we generate both straight-in shots (where the object ball goes directly into the pocket), more complex shots (involving more than one collision), and special shots designed to disperse clusters of balls.
2. For each of these (φ, v_0) pairs, discrete velocity values, v_i, between v_0 for this shot and the maximum allowed velocity v_{MAX}, are generated. The (φ, v_i) pairs that are deemed feasible (i.e. pocket a ball with no Gaussian noise added) are passed to the next step.
3. For each feasible (φ, v_i) pair, variants are generated by randomly assigning feasible values to a, b and θ.
   (a) Each such variant is simulated between 25 and 100 times, depending on available time.
   (b) The resulting states (projected table state after shot simulations) are scored using an evaluation function, allowing the calculation of an average score for each shot variant. The evaluation function estimates the success probability (using a lookup-table) for each straight-in shot from that state. A weighted sum of the success probability for the three best shots is the value of the state [3].
   (c) The top two shot variants for each (φ, v_i) pair are selected.
   (d) For these top two variants, the states resulting from the simulations in Step 3a are clustered into groups of similar table states. A representative state is chosen for each cluster, and a weighted set of representative states is formed.
4. The top 20 shot variants among all shots tested are selected for further evaluation.
5. To refine the evaluation of each of these 20 shots, we execute a second level of search starting with the representative resulting states of these 20 shots. The search method used at this second level essentially repeats the above process (Steps 1–3) with smaller constants, returning the average evaluation for the best shot.
6. After the representative state evaluations have been adjusted, a new evaluation for each of the 20 shot variants is generated, and the best variant overall is chosen.

A.0.2 SingleLevel *(SL)*

SingleLevel is the exact same agent as CueCard, except with the second level of search disabled (Section A.0.1, Step 5). SingleLevel instead spends that time exploring more shot variants from the initial table state.

[3]State value $= 1 * p_1 + 0.33 * p_2 + 0.15 * p_3$, where p_i is the success probability of the i-th most probable shot

A.0.3 OptimisticPlanner *(OP)*

1. For each legal ball and pocket, a set of directions φ, each with a minimum velocity v_0, is generated in attempt to sink the ball into the pocket. In this step only straight-in shots are generated.
2. Same as Step 2 for CueCard
3. For each feasible (φ, v_i) pair, variants are generated by randomly assigning feasible values to a, b and θ.
   (a) Each such variant is simulated a single time, without noise being added.
   (b) The resulting state is scored using the same evaluation function as CueCard. This score is then multiplied by the estimated success probability of the shot (using a lookup-table).
4. The top 5 shot variants among all shots tested are selected for further evaluation.
5. For each of these 5 shot variants, we divide the time equally and perform a depth-first search as deep as time allows, repeating Steps 1 - 4. The value of a state is set to be the value of the best shot available from that state.
6. The best shot found after each states value has been refined by the depth-first search is returned.

A.0.4 MachineGunner *(MG)*

1. Repeat the following Steps 1a - 1c until either 10 successful shots have been found, or at least 1 successful shot has been found and 80 percent of the shot time has elapsed.
   (a) Randomly generate a direction φ.
   (b) For this φ, simulate a full-power shot (with no noise added) in that direction. If the cue ball makes legal contact with another ball, continue to Step 1c, otherwise return to Step 1a.
   (c) Discretize v between its minimum and maximum value. For each feasible shot with this φ and v, simulate the shot 50 times and record the frequency of success for that shot. Add to a priority queue of shots, ordered by success rate.
2. Divide the remaining time equally among the best 10 shots found. For each shot do 50 times:
   (a) Randomly assign feasible values to a, b, and θ, to go with this shot's φ and v.
   (b) Simulate this shot 50 times and keep track of the most successful shot overall.
3. Return the most successful shot. If no successful shot has been found at all, then select the shot which fouled least. If no shots did not foul then intentionally scratch by tapping the ball.