Optimal Play of the Dice Game Pig

Todd W. Neller
Clifton G.M. Presser
Department of Computer Science
300 N. Washington St.
Campus Box 402
Gettysburg College
Gettysburg, PA 17325–1486
tneller@gettysburg.edu

Introduction to Pig

The object of the jeopardy dice game Pig is to be the first player to reach 100 points. Each player's turn consists of repeatedly rolling a die. After each roll, the player is faced with two choices: *roll* again, or *hold* (decline to roll again).

- If the player rolls a 1, the player scores nothing and it becomes the opponent's turn.
- If the player rolls a number other than 1, the number is added to the player's turn total and the player's turn continues.
- If the player holds, the *turn total*, the sum of the rolls during the turn, is added to the player's score, and it becomes the opponent's turn.

For such a simple dice game, one might expect a simple optimal strategy, such as in Blackjack (e.g., "stand on 17" under certain circumstances, etc.). As we shall see, this simple dice game yields a much more complex and intriguing optimal policy, described here for the first time. The reader should be familiar with basic concepts and notation of probability and linear algebra.

Simple Tactics

The game of Pig is simple to describe, but is it simple to play well? More specifically, how can we play the game optimally? Knizia [1999] describes simple tactics where each roll is viewed as a bet that a 1 will not be rolled:

The UMAP Journal 25 (1) (2004) 25–47.

> . . . we know that the true odds of such a bet are 1 to 5. If you ask yourself how much you should risk, you need to know how much there is to gain. A successful throw produces one of the numbers 2, 3, 4, 5, and 6. On average, you will gain four points. If you put 20 points at stake this brings the odds to 4 to 20, that is 1 to 5, and makes a fair game. . . . Whenever your accumulated points are less than 20, you should continue throwing, because the odds are in your favor.
>
> Knizia [1999, 129]

However, Knizia also notes that there are many circumstances in which one should deviate from this "hold at 20" policy. Why does this reasoning not dictate an optimal policy for all play? The reason is that

risking points is not the same as risking the probability of winning.

Put another way, playing to maximize expected score for a single turn is different from playing to win. For a clear illustration, consider the following extreme example. Your opponent has a score of 99 and will likely win in the next turn. You have a score of 78 and a turn total of 20. Do you follow the "hold at 20" policy and end your turn with a score of 98? Why not? Because the probability of winning if you roll once more is higher than the probability of winning if the other player is allowed to roll.

The "hold at 20" policy may be a good rule of thumb, but how good is it? Under what circumstances should we deviate from it and by how much?

Maximizing the Probability of Winning

Let $P_{i,j,k}$ be the player's probability of winning if the player's score is i, the opponent's score is j, and the player's turn total is k. In the case where $i + k \geq 100$, we have $P_{i,j,k} = 1$ because the player can simply hold and win. In the general case where $0 \leq i, j < 100$ and $k < 100 - i$, the probability of a player who plays optimally (an *optimal player*) winning is

$$P_{i,j,k} = \max\left(P_{i,j,k,\text{roll}}, P_{i,j,k,\text{hold}}\right),$$

where $P_{i,j,k,\text{roll}}$ and $P_{i,j,k,\text{hold}}$ are the probabilities of winning for rolling or holding, respectively. These probabilities are

$$\begin{aligned} P_{i,j,k,\text{roll}} &= \tfrac{1}{6}\left[(1 - P_{j,i,0}) + P_{i,j,k+2} + P_{i,j,k+3} + P_{i,j,k+4} + P_{i,j,k+5} + P_{i,j,k+6}\right], \\ P_{i,j,k,\text{hold}} &= 1 - P_{j,i+k,0}. \end{aligned}$$

The probability of winning after rolling a 1 or after holding is the probability that the other player will not win beginning with the next turn. All other outcomes are positive and dependent on the probabilities of winning with higher turn totals.

At this point, we can see how to compute the optimal policy for play. If we can solve for all probabilities of winning in all possible game states, we need only compare $P_{i,j,k,\text{roll}}$ with $P_{i,j,k,\text{hold}}$ for our current state and either roll or hold depending on which has a higher probability of resulting in a win.

Solving for the probability of a win in all states is not trivial, as dependencies between variables are cyclic. For example, $P_{i,j,0}$ depends on $P_{j,i,0}$ which in turn depends on $P_{i,j,0}$. This feature is easily illustrated when both players roll a 1 in subsequent turns. Put another way, game states can repeat, so we cannot simply evaluate probabilities from the end of the game backwards to the beginning, as in dynamic programming (as in Campbell [2002] and other articles in this *Journal*) or its game-theoretic form, known as the *minimax process* (introduced in von Neumann and Morgenstern [1944]; for a modern introduction to that subject, we recommend Russell and Norvig [2003, Ch. 6]).

Let $\mathbf{x}$ be the vector of all possible unknown $P_{i,j,k}$. Because of our equation $P_{i,j,k} = \max(P_{i,j,k,\text{roll}}, P_{i,j,k,\text{hold}})$, our system of equations takes on the interesting form

$$\mathbf{x} = \max(\mathbf{A_1 x} + \mathbf{b_1}, \mathbf{A_2 x} + \mathbf{b_2}).$$

The geometric interpretation of a linear system $\mathbf{x} = \mathbf{Ax} + \mathbf{b}$ is that the solution is the intersection of hyperplanes; but what does our system correspond to geometrically? The set of solutions to a single equation in this system is a (possibly) "folded" hyperplane (**Figure 1**); so a simultaneous solution to the system of equations is the intersection of folded hyperplanes.

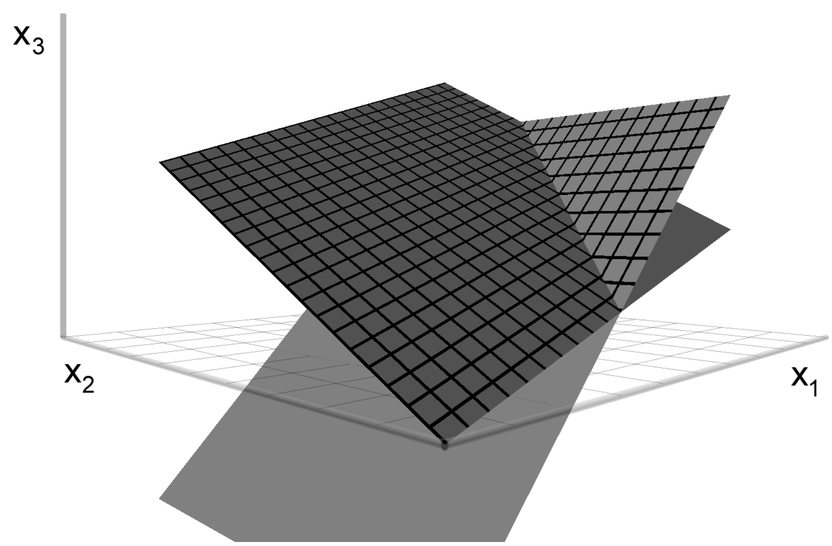


Figure 1. A folded plane.

However, our system has additional constraints: We are solving for probabilities, which take on values only in $[0, 1]$. Therefore, we are seeking the intersection of folded hyperplanes within a unit hypercube of possible probability values.

There is no known general method for solving equations of the form $\mathbf{x} = \max(\mathbf{A_1 x} + \mathbf{b_1}, \mathbf{A_2 x} + \mathbf{b_2})$. However, we can solve our particular problem using a technique called value iteration.

Solving with Value Iteration

Value iteration [Bellman 1957; Bertsekas 1987; Sutton and Barto 1998] is a process that iteratively improves estimates of the value of being in each state until the estimates are "good enough." For ease of explanation, we first introduce a simpler game that we have devised called "Piglet." We then describe value iteration and show how to apply it to Piglet as a generalization of the Jacobi iterative method. Finally, we describe how to apply value iteration to Pig.

Piglet

Piglet is very much like Pig except that it is played with a coin rather than a die. The object of Piglet is to be the first player to reach 10 points. Each turn, a player repeatedly flips a coin until either a tail is flipped or else the player holds and scores the number of consecutive heads flipped.

The number of equations necessary to express the probability of winning in each state is still too many for a pencil-and-paper exercise, so we simplify this game further: The winner is the first to reach 2 points.

As before, let $P_{i,j,k}$ be the player's probability of winning if the player's score is i, the opponent's score is j, and the player's turn total is k. In the case where $i + k = 2$, we have $P_{i,j,k} = 1$ because the player can simply hold and win. In the general case where $0 \leq i, j < 2$ and $k < 2 - i$, the probability of a player winning is

$$P_{i,j,k} = \max(P_{i,j,k,\text{flip}}, P_{i,j,k,\text{hold}}),$$

where $P_{i,j,k,\text{flip}}$ and $P_{i,j,k,\text{hold}}$ are the probabilities of winning if one flips or holds, respectively. The probability of winning if one flips is

$$P_{i,j,k,\text{flip}} = \tfrac{1}{2}\left[(1 - P_{j,i,0}) + P_{i,j,k+1}\right]$$

The probability $P_{i,j,k,\text{hold}}$ is just as before. Then the equations for the probabilities of winning in each state are given as follows:

$$\begin{aligned}
P_{0,0,0} &= \max\left\{\tfrac{1}{2}\left[(1 - P_{0,0,0}) + P_{0,0,1}\right],\ 1 - P_{0,0,0}\right\},\\
P_{0,0,1} &= \max\left\{\tfrac{1}{2}\left[(1 - P_{0,0,0}) + 1\right],\qquad 1 - P_{0,1,0}\right\},\\
P_{0,1,0} &= \max\left\{\tfrac{1}{2}\left[(1 - P_{1,0,0}) + P_{0,1,1}\right],\ 1 - P_{1,0,0}\right\},\\
P_{0,1,1} &= \max\left\{\tfrac{1}{2}\left[(1 - P_{1,0,0}) + 1\right],\qquad 1 - P_{1,1,0}\right\},\\
P_{1,0,0} &= \max\left\{\tfrac{1}{2}\left[(1 - P_{0,1,0}) + 1\right],\qquad 1 - P_{0,1,0}\right\},\\
P_{1,1,0} &= \max\left\{\tfrac{1}{2}\left[(1 - P_{1,1,0}) + 1\right],\qquad 1 - P_{1,1,0}\right\}.
\end{aligned} \tag{1}$$

Once these equations are solved, the optimal policy is obtained by observing which action maximizes $\max(P_{i,j,k,flip}, P_{i,j,k,\text{hold}})$ for each state.

Value Iteration

Value iteration is an algorithm that iteratively improves estimates of the value of being in each state. In describing value iteration, we follow Sutton and Barto [1998], which we also recommend for further reading. We assume that the world consists of *states*, *actions*, and *rewards*. The goal is to compute which action to take in each state so as to maximize future rewards. At any time, we are in a known state s of a finite set of states $\mathcal{S}$. There is a finite set of actions $\mathcal{A}$ that can be taken in any state. For any two states $s, s' \in \mathcal{S}$ and any action $a \in \mathcal{A}$, there is a probability $\mathcal{P}^a_{ss'}$ (possibly zero) that taking action a will cause a transition to state s'. For each such transition, there is an expected immediate *reward* $\mathcal{R}^a_{ss'}$.

We are not interested in just the immediate rewards; we are also interested to some extent in future rewards. More specifically, the *value* of an action's result is the sum of the immediate reward plus some fraction of the future reward. The *discount factor* $0 \leq \gamma \leq 1$ determines how much we care about expected future reward when selecting an action.

Let $V(s)$ denote the estimated value of being in state s, based on the expected immediate rewards of actions *and* the estimated values of being in subsequent states. The estimated value of an action a in state s is given by

$$\sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right].$$

The optimal choice is the action that maximizes this estimated value:

$$\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right].$$

This expression serves as an estimate of the value of being in state s, that is, of $V(s)$. In a nutshell, value iteration consists of revising the estimated values of states until they converge, i.e., until no single estimate is changed significantly. The algorithm is given as Algorithm 1.

Algorithm 1 repeatedly updates estimates of $V(s)$ for each s. The variable Δ is used to keep track of the largest change for each iteration, and ϵ is a small constant. When the largest estimate change Δ is smaller than ϵ, we stop revising our estimates.

Convergence is guaranteed when $\gamma < 1$ and rewards are bounded [Mitchell 1997, §13.4], but convergence is not guaranteed in general when $\gamma = 1$. In the case of Piglet and Pig, value iteration happens to converge for $\gamma = 1$.

Algorithm 1 Value iteration

For each $s \in \mathcal{S}$, initialize $V(s)$ arbitrarily.
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$,
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$
    $\Delta \leftarrow \max\left(\Delta, |v - V(s)|\right)$
until $\Delta < \epsilon$

Applying Value Iteration to Piglet

Value iteration is beautiful in its simplicity, but we have yet to show how it applies to Piglet. For Piglet with a goal of 2, the states are all (i, j, k) triples that can occur in game play, where i, j and k denote the same game values as before. Winning and losing states are terminal. That is, all actions taken in such states cause no change and yield no reward. The set of actions is $\mathcal{A} = \{\text{flip}, \text{hold}\}$.

Let us consider rewards carefully. If points are our reward, then we are once again seeking to maximize expected points rather than maximizing the expected probability of winning. Instead, in order to reward only winning, we set the reward to be 1 for transitions from nonwinning states to winning states and 0 for all other transitions.

The next bit of insight that is necessary concerns what happens when we offer a reward of 1 for winning and do *not* discount future rewards, that is, when we set $\gamma = 1$. In this special case, $V(s)$ is the probability of a player in s eventually transitioning from a nonwinning state to a winning state. Put another way, $V(s)$ is the probability of winning from state s.

The last insight we need is to note the symmetry of the game. Each player has the same choices and the same probable outcomes. It is this fact that enables us to use $(1 - P_{j,i,0})$ and $(1 - P_{j,i+k,0})$ in our Pig/Piglet equations. Thus, we need to consider only the perspective of a single optimal player.

When we review our system of equations for Piglet, we see that value iteration with $\gamma = 1$ amounts to computing the system's left-hand-side probabilities (e.g., $P_{i,j,k}$) from the right-hand-side expressions (e.g., $\max(P_{i,j,k,\text{flip}}, P_{i,j,k,\text{hold}})$) repeatedly until the probabilities converge. This specific application of value iteration can be viewed as a generalization of the Jacobi iteration method for solving systems of linear algebraic equations (see Burden and Faires [2001, §7.3] or Kincaid and Cheney [1996, §4.6]).

The result of applying value iteration to Piglet is shown in **Figure 2**. Each line corresponds to a sequence of estimates made for one of the win probabilities for our Piglet equations. The interested reader can verify that the exact values,

$$P_{0,0,0} = \frac{4}{7}, \quad P_{0,0,1} = \frac{5}{7}, \quad P_{0,1,0} = \frac{2}{5}, \quad P_{0,1,1} = \frac{3}{5}, \quad P_{1,0,0} = \frac{4}{5}, \quad P_{1,1,0} = \frac{2}{3},$$

do indeed solve the system of equations **(1)**.

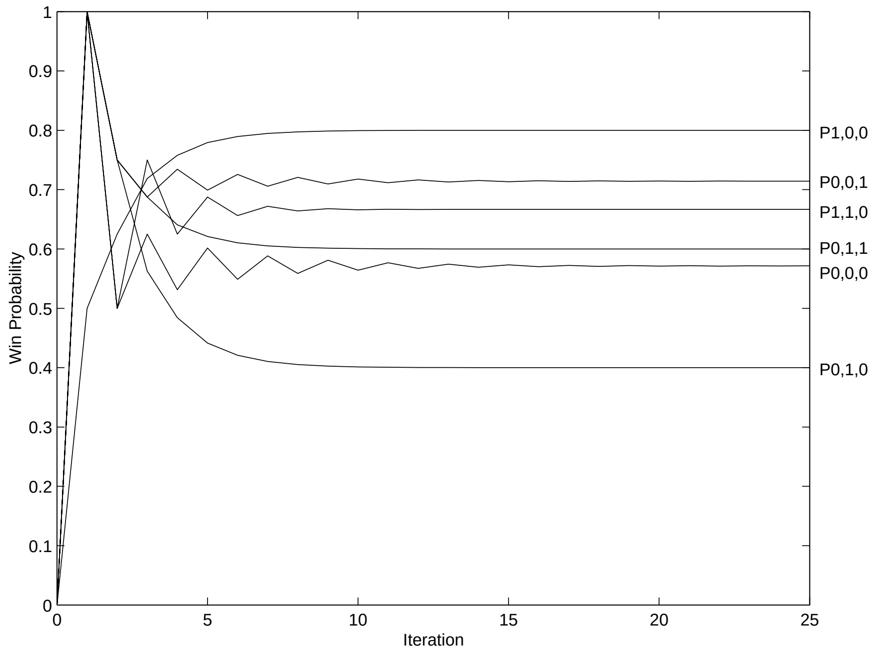


Figure 2. Value Iteration with Piglet (goal points = 2).

Finally, we note that the optimal policy is computed by observing which action yields the maximum expected value for each state. In the case of Piglet with a goal of 2, one should always keep flipping. For Pig, the policy is much more interesting. Piglet with a goal of 10 also has a more interesting optimal policy, although the different possible positive outcomes for rolls in Pig make its policy more interesting still.

Applying Value Iteration to Pig

Value iteration can be applied to Pig much the same as to Piglet. What is different is that Pig presents us with 505,000 equations. To speed convergence, we can apply value iteration in stages, taking advantage of the structure of equation dependencies.

Consider which states are reachable from other states. Players' scores can never decrease; therefore, the sum of scores can never decrease, so a state will never transition to a state where the sum of the scores is less. Hence, the probability of a win from a state with a given score sum is independent of the probabilities of a win from all states with lower score sums. This means that we can first perform value iteration only for states with the highest score sum.

In effect, we partition probabilities by score sums and compute each partition in descending order of score sums. First, we compute $P_{99,99,0}$ with value iteration. Then, we use the converged value of $P_{99,99,0}$ to compute $P_{98,99,0}$ and $P_{99,98,0}$ with value iteration. Next, we compute probabilities with the score sum 196, then 195, etc., until we finally compute $P_{0,0,k}$ for $0 \leq k \leq 99$.

Examining **Figure 2**, we can see that beginning game states take longer to converge as they effectively wait for later states to converge. This approach of performing value iteration in stages has the advantage of iterating values of earlier game states only after those of later game states have converged.

This partitioning and ordering of states can be taken one step further. Within the states of a given score sum, equations are dependent on the value of states with either

- a greater score sum (which would already be computed), or
- the value of states with the players' scores switched (e.g., in the case of a roll of 1).

This means that within states of a given score sum, we can perform value iteration on subpartitions of states as follows: For player scores i and j, value iterate together all $P_{i,j,k}$ for all $0 \leq k < 100-i$ and all $P_{j,i,k}$ for all $0 \leq k < 100-j$.

We solve this game using value iteration. Further investigation might seek a more efficient solution technique or identify a special structure in these equations that yields a particularly simple and elegant solution.

The Solution

The solution to Pig is visualized in **Figure 3**. The axes are i (player 1 score), j (player 2 score), and k (the turn total). The surface shown is the boundary between states where player 1 should roll (below the surface) and states where player 1 should hold (above the surface). We assume for this and following figures that player 1 plays optimally. Player 1 assumes that player 2 will also play optimally, although player 2 is free to use any policy.

Overall, we see that the "hold at 20" policy only serves as a good approximation to optimal play when both players have low scores. When either player has a high score, it is advisable on each turn to try to win. In between these extremes, play is unintuitive, deviating significantly from the "hold at 20" policy and being highly discontinuous from one score to the next.

Let us look more closely at the cross-section of this surface when we hold the opponent's score at 30 (**Figure 4**). The dotted line is for comparison with the "hold at 20" policy. When the optimal player's score is low and the opponent has a significant lead, the optimal player must deviate from the "hold at 20" policy, taking greater risks to catch up and maximize the expected probability of a win. When the optimal player has a significant advantage over the opponent, the optimal player maximizes the expected probability of a win by holding at turn totals significantly below 20.

It is also interesting to consider that not all states are reachable with optimal play. The states that an optimal player can reach are shown in **Figure 5**. These states are reachable regardless of what policy the opponent follows. The reachable regions of cross-sectional **Figure 4** are shaded.

To see why many states are not reachable, consider that a player starts a turn at a given (i, j, 0) and travels upward in k until the player holds or rolls a 1. An

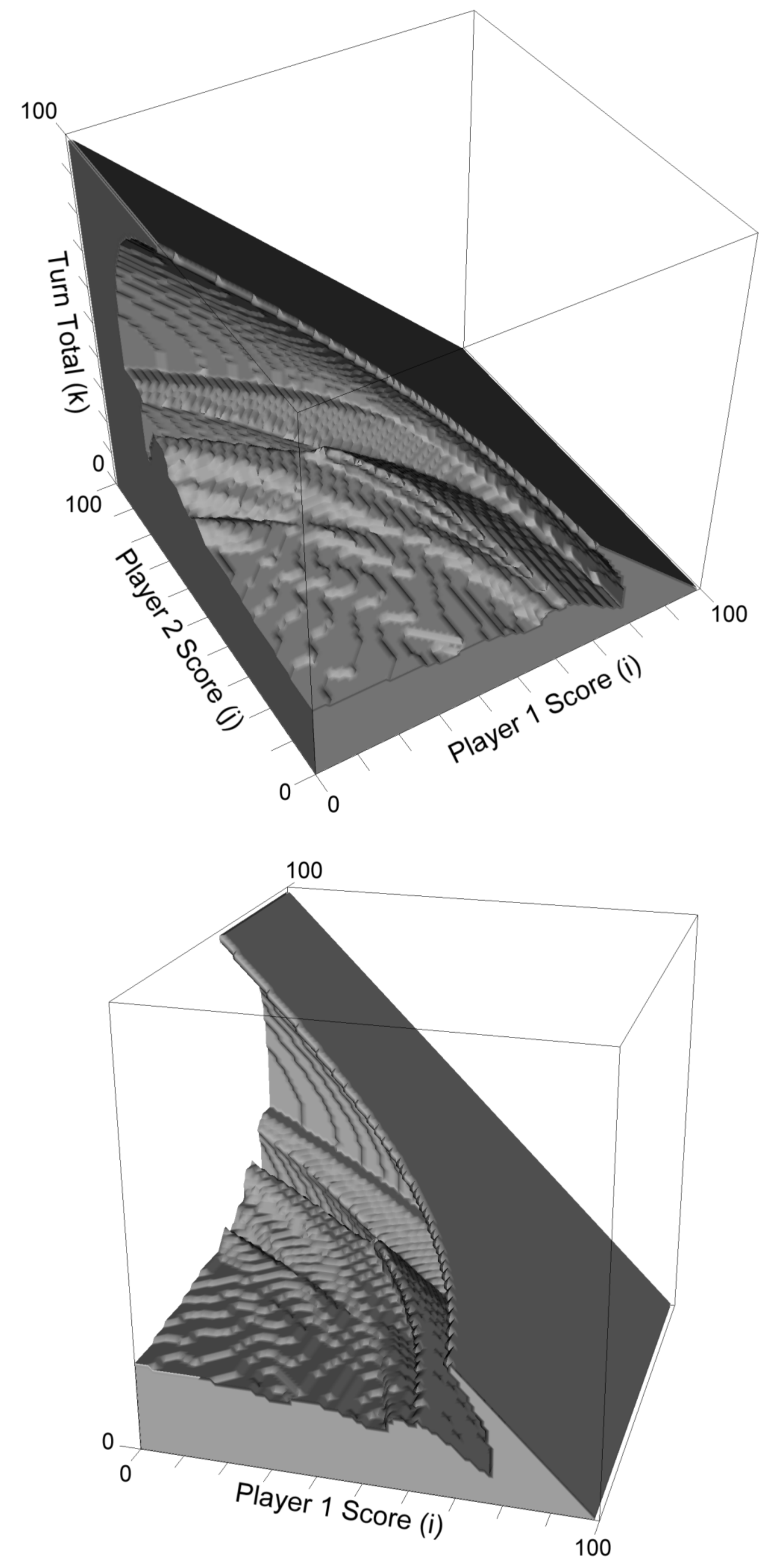


Figure 3. Two views of the roll/hold boundary for optimal Pig play policy.

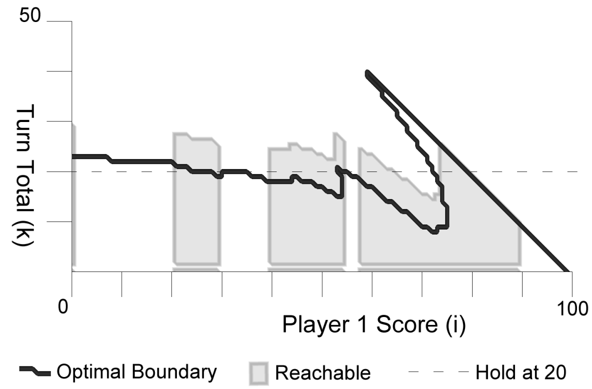


Figure 4. Cross-section of the roll/hold boundary, opponent's score = 30.

optimal player following this policy will not travel more than 6 points above the boundary. For example, an optimal player will never reach the upper-left tip of the large "wave" of **Figure 4**. Only suboptimal risk-seeking play will lead to most states on this wave, but once reached, the optimal decision is to continue rolling towards victory.

Also, consider the fact that an optimal player with a score of 0 will never hold with a turn total less than 21, regardless of the opponent's score. This means that an optimal player will never have a score between 0 and 21. We can see these and other such gaps in **Figure 5**.

Combining the optimal play policy with state reachability, we can visualize the relevant part of the solution as in **Figure 6**. Note the wave tips that are not reachable.

The win probabilities that are the basis for these optimal decisions are visualized in **Figure 7**. Probability contours for this space are shown for 3%, 9%, 27%, and 81%. For instance, the small lower-leftmost surface separates states having more or less than a 3% win probability.

If both players are playing optimally, the starting player wins 53.06% of the time; that is, $P_{0,0,0} \approx 0.5306$. We have also used the same technique to analyze the advantage of the optimal policy versus a "hold at 20" policy, where the "hold at 20" player is assumed to hold at less than 20 when the turn total is sufficient to reach the goal. When the optimal player goes first, the optimal player wins 58.74% of the time. When the "hold at 20" player goes first, the "hold at 20" player wins 47.76% of the time. Thus, if the starting player is chosen using a fair coin, the optimal player wins 55.49% of the time.

Conclusions

The simple game of Pig gives rise to a complex optimal policy. A first look at the problem from a betting perspective yields a simple "hold at 20" policy, but this policy maximizes expected points per turn rather than the probability of winning. The optimal policy is instead derived by solving for the probability

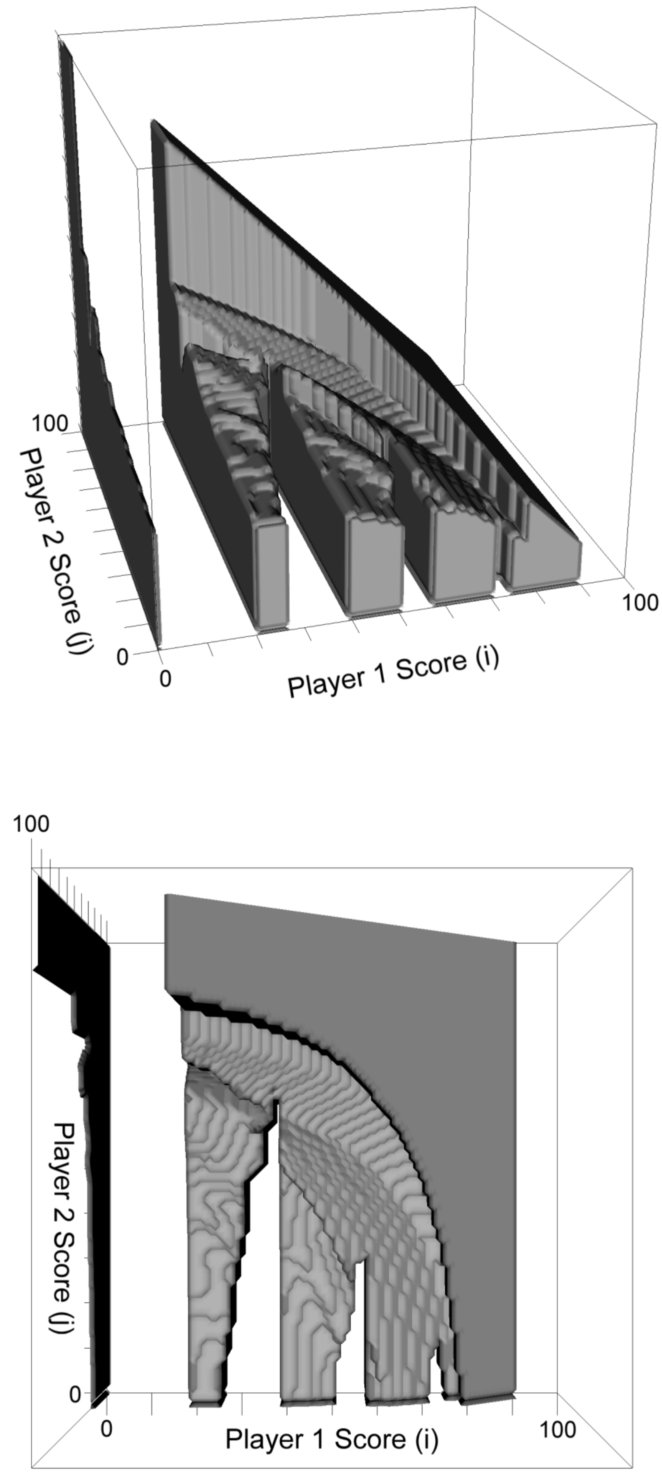


Figure 5. Two views of states reachable by an optimal Pig player.

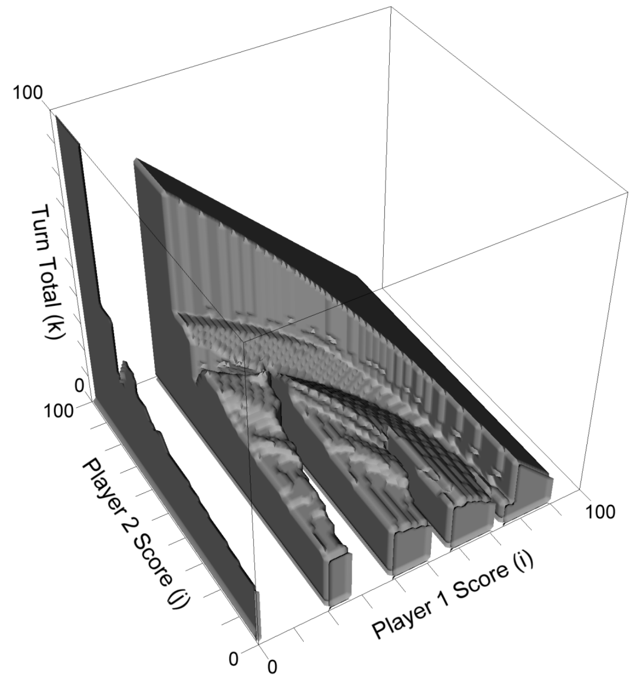


Figure 6. Reachable states where rolling is optimal.

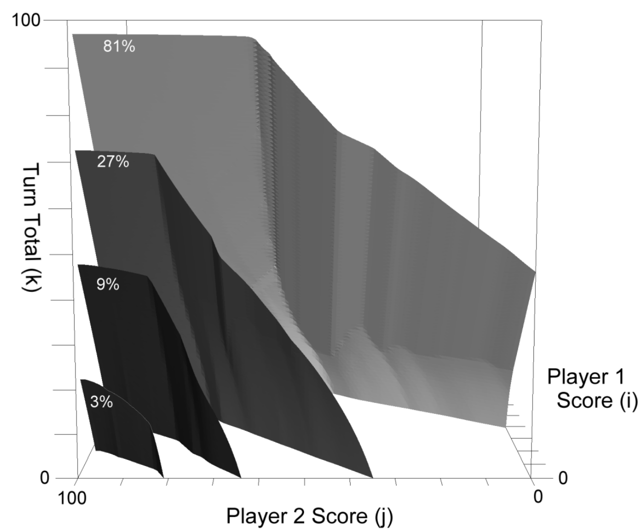


Figure 7. Win probability contours for optimal play (3%, 9%, 27%, 81%).

of winning for every possible game state. This amounts to finding the intersection of folded hyperplanes within a hypercube; the method of value iteration converges and provides a solution. The interested reader may play an optimal computer opponent, view visualizations of the optimal policy, and learn more about Pig at `http://cs.gettysburg.edu/projects/pig`.

Surprising in its topographical beauty, this optimal policy is approximated well by the "hold at 20" policy only when both players have low scores. In the race to 100 points, optimal play deviates significantly from this policy and is far from intuitive in its details. Seeing the "landscape" of this policy is like seeing the surface of a distant planet sharply for the first time having previously seen only fuzzy images. If intuition is like seeing a distant planet with the naked eye, and a simplistic, approximate analysis is like seeing it with a telescope, then applying the tools of mathematics is like landing on the planet and sending pictures home. We will forever be surprised by what we see!

Appendix: Pig Variants and Related Work

We present some variants of Pig. Although the rules presented are for two players, most games originally allow for or are easily extended to more than two players.

The rules of Pig, as we have described them, are the earliest noncommercial variant that we have found in the literature. John Scarne wrote about this version of Pig [1945], recommending that players determine the first player by the lowest roll of the die. Scarne also recommended that all players should be allowed an equal number of turns. Thus, after the turn where one player holds with a score ≥ 100, remaining players have the opportunity to exceed that or any higher score attained. This version also appears in Bell [1979], Diagram Visual Information Ltd [1979], and Knizia [1999]. Boyan's version [1998] differs only in that a roll of 1 ends the turn with a one-point gain.

Scarne's version also serves as the core example for a unit on probability in the first year of the high-school curriculum titled Interactive Mathematics Program® [Fendel et al. 1997]. However, the authors say about their activity, "The Game of Pig," that it " ... does not define a specific goal .... In this unit, 'best' will mean *highest average per turn in the long run* [italics in original]." That is, students learn how to analyze and maximize the expected turn score. Fendel et al. also independently developed the jeopardy coin game Piglet for similar pedagogical purposes, giving it the humorous name "Pig Tails." "Fast Pig" is their variation in which each turn is played with a single roll of n dice; the sum of the dice is scored unless a 1 is rolled.

Parker Brothers Pig Dice

Pig Dice® (©1942, Parker Brothers) is a 2-dice variant of Pig that has had surprisingly little influence on the rules of modern variants, in part because the game requires specialized dice: One die has a pig head replacing the 1; the other has a pig tail replacing the 6. Such rolls are called Heads and Tails, respectively. The goal score is 100; yet after a player has met or exceeded 100, all other players have one more turn to achieve the highest score.

As in Pig, players may hold or roll, risking accumulated turn totals. However:

- There is no undesirable *single* die value; rather, rolling dice that *total 7* ends the turn without scoring.
- Rolling a Head and a Tail doubles the current turn total.
- Rolling just a Head causes the value of the other die to be doubled.
- Rolling just a Tail causes the value of the other die to be negated.
- The turn total can never be negative; if a negated die would cause a negative turn total, the turn total is set to 0.
- All other non-7 roll totals are added to the turn total.

Two Dice, 1 is Bad

According to game analyst Butler [personal communication, 2004], one of the simplest and most common variants, which we will call "2-dice Pig," was produced commercially under the name "Pig" around the 1950s. The rules are the same as our 1-die Pig, except:

- Two standard dice are rolled. If neither shows a 1, their sum is added to the turn total.
- If a single 1 is rolled, the player's turn ends with the loss of the turn total.
- If two 1s are rolled, the player's turn ends with the loss of the turn total *and of the entire score*.

In 1980–81, Butler analyzed 2-dice Pig [2001], computing the turn total at which one should hold to reach 100 points in n turns on average given one's current score. However, he gives no guidance for determining n. Beardon and Ayer [2001b] also presented this variant under the name "Piggy Ones." They [2001a] and Butler also treated a variant where 6 rather than 1 is the undesirable roll value.

W.H. Schaper's Skunk® (©1953, W.H. Schaper Manufacturing Co.) is a commercial variant that elaborates on 2-dice Pig as follows:

- Players begin with 50 poker chips.
- For a single 1 roll, in addition to the aforementioned consequences, the player places one chip into the center "kitty," unless the other die shows a 2, in which case the player places two chips.

- For a double 1 roll, in addition to the aforementioned consequences, the player places four chips into the center kitty.
- The winning player collects the kitty, five chips from each player with a nonzero score, and ten chips from each player with a zero score.

Presumably, a player who cannot pay the required number of chips is eliminated from the match. The match is played for a predetermined number of games, or until all but one player has been eliminated. The player with the most chips wins the match.

Bonuses for Doubles

Skip Frey [1975] describes a 2-dice Pig variation that differs only in how doubles are treated and how the game ends:

- If two 1s are rolled, the player adds 25 to the turn total and it becomes the opponent's turn. (Knizia [1999] calls this a variant of Big Pig, which is identical to Frey's game except that the player's turn continues after rolling double 1s.)
- If other doubles are rolled, the player adds twice the value of the dice to the turn total, and the player's turn continues.
- Players are permitted the same number of turns. So if the first player scores 100 or more points, the second player must be allowed the opportunity to exceed the first player's score and win.

A popular commercial variant, Pass the Pigs® (©1995, David Moffat Enterprises and Hasbro, Inc.), was originally called "PigMania"® (©1977, David Moffat Enterprises). Small rubber pigs are used as dice. When rolled, each pig can come to rest in a variety of positions with varying probability: on its right side, on its left side, upside down ("razorback"), upright ("trotter"), balanced on the snout and front legs ("snouter"), and balanced on the snout, left leg, and left ear ("leaning jowler"). The combined positions of the two pigs lead to various scores.

In 1997, 52 6th-grade students of Dean Ballard at Lakeside Middle School in Seattle, WA, rolled such pigs 3939 times. In the order of roll types listed above, the number of rolls were: 1344, 1294, 767, 365, 137, and 32 [Wong n.d.]. In Pass the Pigs, a pig coming to rest against another is called an "oinker" and results in the loss of all points. Since the 6th-graders' data are for single rolls, no data on the number of "oinkers" is given. However, David R. Bellhouse's daughter Erika rolled similar "Tequila Pigs" 1435 times in sets of 7 with respective totals 593, 622, 112, 76, 27, and 2. The remaining 3 rolls were "oinkers," leaning on other pigs at rest in standard positions [Bellhouse 1999].

PigMania® is similar to 2-Dice Pig, in that a roll of a left side and a right side in PigMania has the same consequences as rolling a 1 in 2-Dice Pig (the turn ends with loss of the turn total), and a roll with pigs touching has the same consequences as rolling double 1s (the turn ends with loss of the turn total and of the entire score). PigMania is similar to Frey's variant in that two pigs in the same non-side configuration score double what they would individually.

Maximizing Points with Limited Turns

Dan Brutlag's SKUNK [1994]—not to be confused with W.H. Schaper's Skunk®—is a variant of Pig that follows the rules of 2-Dice Pig except:

- Each player gets only five turns, one for each letter of SKUNK.
- The highest score at the end of five turns wins.

Brutlag describes the game as part of an engaging middle-school exercise that encourages students to think about chance, choice, and strategy; in personal correspondence, he mentions having been taught the game long before 1994. He writes, "To get a better score, it would be useful to know, on average, how many good rolls happen in a row before a 'one' or 'double ones' come up" [1994].

However, students need to realize that optimal play of SKUNK is not conditioned on how many rolls one has made but rather on the players' scores, the current player's turn total, and how many turns remain for each player. It is important to remind students that dice do not "know" how many times they have been rolled, that is, dice are stateless. The false assumption that the number of prior rolls makes the probability of a 1 being rolled more or less likely is an example of the well-known *gambler's fallacy* [Blackburn 1994].

For example, suppose that a turn total of 10 has been achieved. The decision of whether to roll again or stay should not be affected by whether the 10 was realized by one roll of 6–4 or by two rolls of 3–2; the same total is at stake regardless. Although one might argue that the number of turn rolls is an easy feature of the game for young students to grasp, the turn total is similarly easy and moreover is a *relevant* feature for decision-making.

Falk and Tadmor-Troyanski [1999] follow Brutlag's SKUNK with analysis of optimizing the score for their variant THINK. In THINK, both players take a turn simultaneously. One player rolls the dice and both use the result, each deciding separately between rolls whether to hold or not. A turn continues until both players hold or a 1 is rolled. As Knizia [1999] did for the original Pig, Falk and Tadmor-Troyanski seek play that optimizes the expected final score, blind to all other considerations. They first treat the case where a player must decide before the game begins when to stop rolling for each turn, as if the player were playing blindfolded. This case could be considered a two-dice variant of n-dice Fast Pig where each player chooses n for each turn. In this circumstance, the number of turn rolls is a relevant feature of the decision because it is the only feature given. They conclude that three rolls are appropriate for all but the last turn, when one should roll just twice to maximize the expected score. They then remove the blindfold assumption and perform an odds-based analysis that shows that the player should continue rolling if $s + 11t \leq 200$, where s is the player score and t is the turn total.

Single Decision Analysis Versus Dynamic Programming

While the single-roll, odds-based analyses of Falk and Tadmor-Troyanski [1999] and Knizia [1999] yield policies optimizing expected score for a *single turn*, they do not yield policies optimizing score over an arbitrary number of turns. We illustrate by applying dynamic programming to a solitaire version of THINK.

Let $E_{r,s,t}$ be the player's expected future score gain if the turn is r, the player's score is s, and the player's turn total is t. Since the game is limited to five turns, we have $E_{r,s,t} = 0$ for $r > 5$. For $1 \leq r \leq 5$, for the optimal expected future score gain, we want

$$E_{r,s,t} = \max\left(E_{r,s,t,\text{roll}}, E_{r,s,t,\text{hold}}\right)$$

where $E_{r,s,t,\text{roll}}$ and $E_{r,s,t,\text{hold}}$ are the expected future gain if one rolls and holds, respectively. These expectations are given by:

$$\begin{aligned}
E_{r,s,t,\text{roll}} = \frac{1}{36}\ \big[&1(4 + E_{r,s,t+4}) + 2(5 + E_{r,s,t+5}) + 3(6 + E_{r,s,t+6}) \\
+\ &4(7 + E_{r,s,t+7}) + 5(8 + E_{r,s,t+8}) + 4(9 + E_{r,s,t+9}) \\
+\ &3(10 + E_{r,s,t+10}) + 2(11 + E_{r,s,t+11}) + 1(12 + E_{r,s,t+12}) \\
&+ 10(-t + E_{r+1,s,0}) + 1(-s - t + E_{r+1,0,0})\big], \\
E_{r,s,t,\text{hold}} = E_{r+1,s+t,0}&.
\end{aligned}$$

Since the state space has no cycles, value iteration is unnecessary. Computing the optimal policy π^* through dynamic programming, we calculate the expected gain $E^{\pi^*}_{1,0,0} = 36.2915\underline{3}313960543$. If we instead apply the policy $\pi^{\leq}$ of the single-turn odds-based analysis, rolling when $s + 11t \leq 200$, we calculate the expected gain $E^{\pi^{\leq}}_{1,0,0} = 36.2915\underline{1}996719233$. These numbers are so close that simulating more than 10^9 games with each policy could not demonstrate a significant statistical difference in the average gain.

However, two factors give support to the correctness of this computation. First, we observe that the policy $\pi^{\leq}$ is risk-averse with respect to the computed optimal policy π^*. According to the odds-based analysis, it does not matter what one does if $s + 11t = 200$, and the authors state "you may flip a coin to decide". The above computation assumed one always rolls when $s + 11t = 200$. If this analysis is correct, there should be no difference in expected gain if we hold in such situations. However, if we instead apply the policy $\pi^{<}$ of this odds-based analysis, rolling when $s + 11t < 200$, we compute $E^{\pi^{<}}_{1,0,0} = 36.291\underline{3}2666694349$, which is different and even farther from the optimal expected gain.

Second, we can more easily observe the difference between optimal and odds-based policies if we extend the number of turns in the game to 20. Then $E^{\pi^*}_{1,0,0} = 104.7\underline{8}360865008132$ and $E^{\pi^{\leq}}_{1,0,0} = 104.7\underline{2}378302093477$. After 2×10^8 simulated games with each policy, the average gains were $104.7\underline{8}4846975$ and $104.7\underline{2}618221$, respectively.

Of special note is the good quality of the approximation such an odds-based analysis gives us for optimal THINK score gain, given such simple, local considerations. For THINK reduced to four turns, we compute that policies π^* and $\pi^{\leq}$ reach the same game states and dictate the same decisions in those states. Similarly examining Knizia's Pig analysis for maximizing expected score, we find the same deviation of optimal versus odds-based policies for Pig games longer than eight turns.

Miscellaneous Variants

Yixun Shi [2000] describes a variant of Pig that is the same as Brutlag's SKUNK except:

- There are six turns.
- A roll that increases the turn total does so by the *product* of the dice values rather than by the sum.
- Double 1s have the same consequences as a single 1 in 2-Dice Pig (loss of turn total, end of turn).
- Shi calls turns "games" and games "matches". We adhere to our terminology.

Shi's goal is not so much to analyze this Pig variant as to describe how to form heuristics for good play. In particular, he identifies features of the game (e.g., turn total, score differences, and distance from expected score per turn), combining them into a function to guide decision making. He parametrizes the heuristics and evaluates parameters empirically through actual play.

Ivars Peterson describes Piggy [2000], which varies from 2-Dice Pig in that there is no bad dice value. However, doubles have the same consequences as a single 1 in 2-dice Pig. Peterson suggests comparing Piggy play with standard dice versus nonstandard *Sicherman dice*, for which one die is labeled 1, 2, 2, 3, 3, and 4 and the other is labeled 1, 3, 4, 5, 6, and 8. Although the distribution of roll sums is the same for Sicherman and standard dice, doubles are rarer with Sicherman dice.

Jeopardy Dice Games: Race and Approach

Pig and its variants belong to a class of dice games called *jeopardy dice games* [Knizia 1999], where the dominant decision is whether or not to jeopardize *all* of one's turn total by continuing to roll for a potentially better turn total. We suggest that jeopardy dice games can be further subdivided into two main subclasses: *jeopardy race games* and *jeopardy approach games*.

Jeopardy Race Games

In *jeopardy race games*, the object is to be the first to meet or exceed a goal score. Pig is the simplest of these. Most other jeopardy race games are variations of the game Ten Thousand (the name refers to the goal score; Five Thousand is a common shortened variant). In such games, a player rolls dice (usually six), setting aside various scoring combinations of dice (which increase the turn total) and re-rolling the remaining dice, until the player either holds (and scores the turn total) or reaches a result with no possible scoring combination and thus loses the turn total. Generally, if all dice are set aside in scoring combinations, the turn continues with all dice back in play.

According to Knizia [1999], Ten Thousand is also called Farkle, Dix Mille, Teutonic Poker, and Berliner Macke. Michael Keller [n.d.; 1998] lists many commercial variants of Ten Thousand: $Greed (1980, Avalon Hill), Zilch (1980, Twinson), Bupkis (1981, Milco), Fill or Bust (1981, Bowman Games)— also known as Volle Lotte (1994, Abacus Spiele), High Rollers (1992, El Rancho Escondido Ents.), Six Cubes (1994, Fun and Games Group), Keepers (Avid Press), Gold Train (1995, Strunk), and—the most popular—Cosmic Wimpout (1984, C3 Inc.). Sid Sackson also described the Ten Thousand commercial variants Five Thousand (1967, Parker Brothers) [Sackson 1969] and Top Dog (John N. Hanson Co.) [Sackson 1982].

Additional commercial jeopardy race games include Sid Sackson's Can't Stop® (1980, Parker Brothers; 1998, Franjos Spieleverlag), and Reiner Knizia's Exxtra® (1998, Amigo Spiele).

It is interesting to consider the relationship between jeopardy race dice games and primitive board "race games" [Bell 1979; Parlett 1999] that use dice to determine movement. Parlett writes, "It seems intuitively obvious that race games evolved from dice games" [1999, 35]. In the simplest primitive board games, the board serves primarily to track progress toward the goal, as a form of score pad. However, with the focus of attention drawn to the race on the board, Parlett and others suggest, many variations evolved regarding the board. Thus, jeopardy dice-roll elements may have given way to jeopardy board elements (e.g., one's piece landing on a bad space, or being landed on by another piece). In whichever direction the evolution occurred, it is reasonable to assume that jeopardy race games have primitive origins.

Jeopardy Approach Games

In *jeopardy approach games*, the object is to most closely approach a goal score without exceeding it. These include Macao, Twenty-One (also known as Vingt-et-Un, Pontoon, Blackjack), Sixteen (also known as Golden Sixteen), Octo, Poker Hunt [Knizia 1999], Thirty-Six [Scarne 1980], and Altars [Imbril n.d.]. Macao, Twenty-One, and Thirty-Six are most closely related to the card game

Blackjack. The playing-card version of Macao was very popular in the 17th and 18th centuries [Scarne 1980]; the card game Vingt-et-Un gained popularity in the mid-18th century as a favorite game of Madame du Barry and Napoleon [Parlett 1991]. Parlett writes, "That banking games are little more than dice games adapted to the medium of cards is suggested by the fact that they are fast, defensive rather than offensive, and essentially numerical, suits being often irrelevant" [1999, 76].

Computational Challenges

Optimal play for Macao and Sixteen has been computed by Neller and his students at Gettysburg College through a similar application of value iteration. Other non-jeopardy dice games have been solved with dynamic programming, e.g., Campbell [2002]. However, many dice games are not yet solvable because of the great number of reachable game states and the memory limitations of modern computers.

Memory requirements for computing a solution may be reduced through various means. For instance, the partitioning technique that we described can be used to hold only those states in memory that are necessary for the solution of a given partition. Also, one can make intense use of vast, slower secondary memory. That is, one can trade off computational speed for greater memory.

One interesting area for future work is the development of techniques to compute *approximately optimal* policies. We have shown that many possible Pig game states are not reachable through optimal play, but it is also the case that many reachable states are improbable. Simulation-based techniques such as Monte Carlo and temporal difference learning algorithms [Sutton and Barto 1998] do not require probability models for state transitions and can converge quickly for frequently occurring states. Approximately optimal play for more difficult games, such as Backgammon, can be achieved through simulation-based reinforcement learning techniques combined with feature-based state-abstractions [Tesauro 2002; Boyan 2002].

References

Beardon, Toni, and Elizabeth Ayer. 2001a. Game of PIG—Sixes. NRICH (June 2001 and May 2004). `http://www.nrich.maths.org/public/viewer.php?obj_id=1258` .

———. 2001b. Game of PIG—Ones: Piggy Ones and Piggy Sixes: Should you change your strategy? NRICH (July 2001). `http://www.nrich.maths.org/public/viewer.php?obj_id=1260` .

Bell, Robert Charles. 1979. *Board and Table Games from Many Civilizations*. Revised ed. New York: Dover Publications, Inc.

Bellhouse, David R. 1999. Il campanile statistico: What I did on my summer holidays. *Chance* 12 (1): 48–50.

Bellman, Richard. 1957. *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bertsekas, D.P. 1987. *Dynamic Programming: Deterministic and Stochastic Models*. Englewood Cliffs, NJ: Prentice-Hall.

Blackburn, Simon. 1994. *The Oxford Dictionary of Philosophy*. New York: Oxford University Press.

Boyan, Justin A. 1998. Learning evaluation functions for global optimization. Ph.D. thesis. Carnegie Mellon University. Pittsburgh, PA. Carnegie Mellon Tech. Report CMU-CS-98-152.

———. 2002. Least-squares temporal difference learning. *Machine Learning* 49 (2/3): 233–246.

Brutlag, Dan. 1994. Choice and chance in life: The game of "skunk". *Mathematics Teaching in the Middle School* 1 (1): 28–33.

Burden, Richard L., and J. Douglas Faires. 2001. *Numerical Analysis*. 7th ed. Pacific Grove, CA: Brooks/Cole Publishing Co.

Butler, Bill. 2001. Durango Bill's "Pig (Pig-out)" analysis. `http://www.durangobill.com/Pig.html`.

Campbell, Paul J. 2002. Farmer Klaus and the mouse. *The UMAP Journal* 23 (2): 121–134. 2004. Errata. 24 (4): 484.

Diagram Visual Information Ltd. 1979. *The Official World Encyclopedia of Sports and Games*. London: Paddington Press.

Falk, Ruma, and Maayan Tadmor-Troyanski. 1999. THINK: A game of choice and chance. *Teaching Statistics* 21 (1): 24–27.

Fendel, Dan, Diane Resek, Lynne Alper, and Sherry Fraser. 1997. *The Game of Pig. Teacher's Guide. Interactive Mathematics Program, Year 1.* Berkeley, CA: Key Curriculum Press.

Frey, Skip. 1975. *How to Win at Dice Games*. North Hollywood, CA: Wilshire Book Co. 1997. Reprint.

Imbril, Blacky. n.d. Altars. `http://members.aol.com/dicetalk/rules/altars.txt` .

Keller, Michael. n.d. Ten Thousand. `http://members.aol.com/dicetalk/rules/10000.txt` .

———. 1998. Ten Thousand games to play with dice. *WGR* 13: 22–23, 37. Published by Michael Keller, 1227 Lorene Drive, Pasadena, MD 21222; fomalhaut@earthlink.net .

Kincaid, David R., and E. Ward Cheney. 1996. *Numerical Analysis: Mathematics of Scientific Computing*. 2nd ed. Pacific Grove, CA: Brooks/Cole Publishing Co.

Knizia, Reiner. 1999. *Dice Games Properly Explained*. Brighton Road, Lower Kingswood, Tadworth, Surrey, KT20 6TD, U.K.: Elliot Right-Way Books.

Mitchell, Tom M. 1997. *Machine Learning*. New York: McGraw-Hill.

von Neumann, John, and Oskar Morgenstern. 1944. *Theory of Games and Economic Behavior*. Princeton, NJ: Princeton University Press.

Parlett, David. 1991. *A History of Card Games*. New York: Oxford University Press.

———. 1999. *The Oxford History of Board Games*. New York: Oxford University Press.

Peterson, Ivars. 2000. Weird dice. *Muse Magazine*. (May/June): 18.

Russell, Stuart, and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Sackson, Sid. 1969. *A Gamut of Games*. New York: Pantheon Books. 1982. 2nd ed. New York: Pantheon Books.

Scarne, John. 1945. *Scarne on Dice*. Harrisburg, PA: Military Service Publishing Co. 1980. 2nd ed. New York: Crown Publishers, Inc.

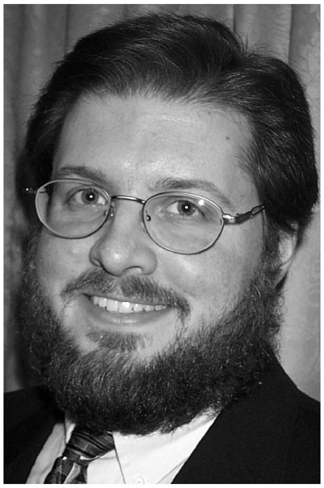
Shi, Yixun. 2000. The game PIG: Making decisions based on mathematical thinking. *Teaching Mathematics and Its Applications* 19 (1): 30–34.

Sutton, Richard S., and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Tesauro, Gerald J. 2002. Programming backgammon using self-teaching neural nets. *Artificial Intelligence* 134: 181–199.

Wong, Freddie. n.d. Pass the Pigs — Probabilities. http://members.tripod.com/~passpigs/prob.html .

About the Authors



Todd W. Neller is Assistant Professor of Computer Science at Gettysburg College. A Cornell University Merrill Presidential Scholar, he received a B.S. in Computer Science with distinction in 1993. He received a Stanford University Lieberman Fellowship in 1998, where he received the George E. Forsythe Memorial Award for excellence in teaching and a Ph.D. in Computer Science in 2000. His thesis concerned extension of artificial intelligence search algorithms to hybrid dynamical systems, and the refutation of hybrid system properties through simulation and information-based optimization. Recent works have concerned the application of reinforcement learning techniques to the control of optimization and search algorithms.

Clifton G.M. Presser is also Assistant Professor of Computer Science at Gettysburg College. He received a B.S. in Mathematics and Computer Science from Pepperdine University in 1993. Clif received his Ph.D. in Computer Science at the University of South Carolina in 2000, where he received the Outstanding Graduate Student Award in the same year. Clif's dissertation research was on automated planning in uncertain environments. Currently, his research concerns computer visualization of high-dimensional geometry, algorithms and information.