Dice games and stochastic dynamic programming *

Henk Tijms

Abstract

This paper uses dice games such as the game of Pig and the game of Hog to illustrate the powerful method of stochastic dynamic programming. Many students have difficulties in understanding the concepts and the solution method of stochastic dynamic programming, but using challenging dice games this understanding can be greatly enhanced and the essence of stochastic dynamic programming can be explained in a motivating way.

1 Introduction

It is a great pleasure to make as Dutch mathematician this invited contribution about stochastic dynamic programming at the occasion of the 10th anniversary of the journal Morfismos of Mexican mathematics students: where a Dutch school of researchers made significant contributions to the field stochastic dynamic programming in the seventies and the eighties of the last century, it was in the nineties the Mexican school of researchers under leadership of professor Onésimo Hernández-Lerma that was very influential in the further development of the field. In this contribution we consider stochastic problems that are fun and instructive to work on. These problems are the dice game Pig and the

---

related dice game Hog. The game of Pig and the game of Hog are
not only teaching treasures but involve challenging research problems
as well. These control problems are of pedagogical use for

- stochastic dynamic programming

- Markov chains

- game theory.

The dice games of Pig and Hog are simple to describe, but it is not
that simple at all to find the optimal strategies. Let us first describe
the games.

The game of Pig

The game of Pig involves two players who in turn roll a die. The
object of the game is to be the first player to reach 100 points. In
each turn, a player repeatedly rolls a die until either a 1 is rolled or
the player holds. If the player rolls a 1, the player gets a score zero
for that turn and it becomes the opponent's turn. If the player holds
after having rolled a number other than 1, the total number of points
rolled in that turn is added to the player's total score and it becomes
the opponent's turn. At any time during a player's turn, the player
must choose between the two decisions "roll" or "hold".

The game of Hog

The game of Hog (fast Pig) is a variation of the game of Pig in which
players have only one roll per turn but may roll as many dice as desired.
The number of dice a player chooses to roll can vary from turn to turn.
The player's score for a turn is zero if one or more of the dice come up
with the face value 1. Otherwise, the sum of the face values showing on
the dice is added to the player's score.

We will first analyze the *single-player* versions of the two stochastic
control problems. For various optimality criteria in the single player
problem, the stochastic dynamic programming approach for calculating
an optimal control rule will discussed. The optimal control rule is rather
complex and therefore its performance will also be compared with the
performance of a simple heuristic rule.

2   The game of Pig

We first consider the single-player version of the game of Pig before we discuss the dynamic programming approach the case with two players. In the two-player's case the goal is to be the first player reaching 100 points. For the single-player version the following two optimality criteria can be considered:

- minimal expected number of turns to reach 100 points

- maximal probability of reaching 100 points in a given number of turns.

The optimal control rules can be calculated from the optimality equations from stochastic dynamic programming, but these optimal rules are rather complex and difficult to use in practice. Therefore we also consider the simple "hold at 20" heuristic and compare the performance of this heuristic with the performance of the optimal rule. The "hold at 20" rule is as follows: after rolling a number other than 1 in the current turn, the player holds that turn when the accumulated number of points during the turn is 20 or more. The rationale of this simple heuristic is easily explained. Suppose that k points have been accumulated so far in the current turn. If you roll again, the expected number of points you gamble away is $\frac{1}{6} \times k$, while the expected number of additional points you gain is equal to $\frac{5}{6} \times 4$, using the fact the expected value of the outcome of a roll of a die is 4 given that the outcome is not 1. The first value of k for which $\frac{1}{6} \times k \geq \frac{5}{6} \times 4$ is $k = 20$.

It turns out that the "hold at 20" heuristic performs very well the criterion is to minimize the expected number of turns to reach 100 points. As will be shown below, the expected value of the number of turns to reach 100 point is 12.367 when the "hold at 20" heuristic is used and this lies only 0.7% above the minimal expected value 12.545 that results when an optimal control rule is used. The situation is different for the criterion of maximizing the probability of reaching 100 points within a given number of turns. Under the "hold at 20" heuristic the probability of reaching 100 points within N turns has the respective values 0.0102, 0.0949, 0.3597, 0.7714, and 0.9429 for $N =$5, 7, 10, 15, and 20, whereas this probability has the maximum values 0.1038, 0.2198, 0.4654, 0.8322, and 0.9728 when an optimal rule is used. Thus, the "hold at 20" heuristic performs unsatisfactorily for the second optimality criterion.

2.1 Analysis of the heuristic rule

The analysis for the "hold at 20" heuristic is based on recurrence relations that are derived by arguments used to analyze absorbing Markov chains. Define μ_i as the expected value of the number of turns needed to reach a total score of 100 points when starting a new turn with a score of i points and using the "hold at 20" rule. The goal is to find μ_0. For a=0, 20, 21, 22, 23, 24, and 25, denote by $\alpha_{0,a}$ the probability that the player will end up with exactly a points in any given turn under the "hold at 20" rule. Once the probabilities $\alpha_{0,a}$ have been computed, we calculate the μ_i by a backwards recursion. By the law of conditional expectations,

$$\mu_i = 1 + \mu_i \alpha_{0,0} + \sum_{a=20}^{25} \mu_{i+a} \alpha_{0,a} \qquad \text{for } i = 99, 98 \ldots, 0.$$

with the convention $\mu_k = 0$ for $k \geq 100$. Thus, initiating the recursion with $\mu_{99} = 1 + \mu_{99} \alpha_{0,0}$, we compute successively $\mu_{99}, \mu_{98}, \ldots, \mu_0$. How to calculate the probabilities $\alpha_{0,a}$? This goes along the same lines as the computation of the absorption probabilities in a Markov chain with absorbing states. For any fixed a, we use the intermediary probabilities $\alpha_{b,a}$ for $0 \leq b \leq 19$, where $\alpha_{b,a}$ is defined as the probability that the current turn will end up with exactly a points when so far b points have been accumulated during the current turn and the "hold at 20 rule" is used. For $a = 0$, we find by the law of conditional probabilities that

$$\alpha_{b,0} = \frac{1}{6} + \sum_{j=2}^{6} \alpha_{b+j,0} \frac{1}{6} \qquad \text{for } b = 19, 18, \ldots, 0$$

with the convention $\alpha_{k,0} = 0$ for $k \geq 20$. For any a with $20 \leq a \leq 25$, we find by conditioning that

$$\alpha_{b,a} = \sum_{j=2}^{6} \alpha_{b+j,a} \frac{1}{6} \qquad \text{for } b = 19, 18, \ldots, 0$$

with the convention $\alpha_{k,a} = 1$ for $k = a$ and $\alpha_{k,a} = 0$ for $k \geq 20$ and $k \neq a$. Applying these recursion equations, we find $\alpha_{0,0} = 0.6245$, $\alpha_{0,20} = 0.0997$, $\alpha_{0,21} = 0.0950$, $\alpha_{0,22} = 0.0742$, $\alpha_{0,23} = 0.0542$, $\alpha_{0,24} = 0.0352$, and $\alpha_{0,25} = 0.0172$. Next the value $\mu_{100} = 12.637$ is calculated for the expected number of turns needed to reach 100 points if the "hold at 20 rule" is used.

How do we calculate the probability of reaching 100 points in no more than N turns under the "hold at 20" heuristic? To do so, we define $Q_n(i)$ for $i < 100$ and $n \geq 1$ the probability $Q_n(i)$ as the probability of reaching 100 points in no more than n turns when the first turn is started with a score of i points and the "hold at 20 rule" is used. Also, let $Q_n(i) = 1$ for any $i \geq 100$ and $n \geq 1$. If no more than a given number of N turns are allowed, the desired probability is $Q_N(0)$. Using the law of conditional probabilities, it follows that the probabilities $Q_n(i)$ for $n = 1, 2, \ldots$ can be computed from the recursion

$$Q_n(i) = Q_{n-1}(i)\alpha_{0,0} + \sum_{a=20}^{25} Q_{n-1}(i+a)\alpha_{0,a}$$

for $i < 100$ and $n \geq 1$ with the boundary condition $Q_0(j) = 1$ for $j \geq 100$ and $Q_0(j) = 0$ for $j < 100$.

2.2 Dynamic programming for the single-player version

In the optimality analysis of the single-player version, a state variable should be defined together with a value function. The state s of the system is defined by a pair $s = (i, k)$, where

$i =$ the player's score at the start of the current turn
$k =$ the number of points obtained so far in the current turn.

We first consider the criterion of minimizing the expected number of turns to reach 100 points. For this criterion, the value function $V(s)$ is defined by

$V(s) =$ the minimal expected value of the number of turns including the current turn to reach 100 points starting from state s

We wish to compute $V(0, 0)$ together with the optimal decision rule. This can be done from Bellman's optimality equations. For $k = 0$,

$$V(i, 0) = 1 + \frac{1}{6}V(i, 0) + \sum_{r=2}^{6} \frac{1}{6}V(i, r).$$

For $k \geq 1$ and $i + k < 100$,

$$V(i, k) = \min\left[V(i+k, 0), \ \frac{1}{6}V(i, 0) + \sum_{r=2}^{6} \frac{1}{6}V(i, k+r) \right],$$

where $V(i, k) = 0$ for those (i, k) with $i + k \geq 100$. The first term in the right side of the last equation corresponds to the decision "hold" and the second term corresponds to the decision "roll". The optimality equation can be solved by the method of successive substitutions. Starting with $V_0(s) = 0$ for all s, the functions $V_1(s), V_2(s), \ldots$ are recursively computed from

$$V_n(i, 0) = 1 + \frac{1}{6} V_{n-1}(i, 0) + \sum_{r=2}^{6} \frac{1}{6} V_{n-1}(i, r), \quad n = 1, 2, \ldots$$

and

$$V_n(i, k) = \min \left[V_{n-1}(i + k, 0), \frac{1}{6} V(i, 0) + \sum_{r=2}^{6} \frac{1}{6} V_{n-1}(i, k + r) \right],$$

$$n = 1, 2, \ldots.$$

By a basic result from the theory of stochastic dynamic programming (see for instance [1] or [2]),

$$\lim_{n \to \infty} V_n(s) = V(s) \quad \text{for all } s.$$

In the literature bounds are known for the difference $|V_n(s) - V(s)|$, providing a stopping criterion for the method of successive substitutions.

Let us next consider the optimality criterion of maximizing the probability of reaching 100 points in no more than N turns with N a given integer. Then, we define for $m = 0, 1, \ldots, N$ the value function $P_m(s)$ by

$P_m(s)$ = the maximal probability of reaching 100 points from state s if no more than m turns can be used including the current turn,

where $P_m(s) = 1$ for all $s = (i, k)$ with $i + k \geq 100$. The desired probability $P_N(0, 0)$ and the optimal decision rule can be calculated from Bellman's optimality equation. For $k = 0$ and $i = 99, 98, \ldots, 0$

$$P_m(i, 0) = \frac{1}{6} P_{m-1}(i, 0) + \sum_{r=2}^{6} \frac{1}{6} P_m(i, r), \quad m = 1, \ldots, N$$

and for $i = 98, 97, \ldots, 0$ and $k = 99 - i, \ldots, 1$

$$P_m(i, k) = \min \left[P_{m-1}(i + k, 0), \frac{1}{6} P_{m-1}(i, 0) + \sum_{r=2}^{6} \frac{1}{6} P_m(i, k + r) \right],$$

$$1 \leq m \leq N.$$

The value functions $P_1(s), P_2(s), \ldots, P_N(s)$ can be recursively calculated, using the fact that $P_m(i, k) = 1$ if $i + k \geq 100$ and starting with

$$P_0(i, k) = \begin{cases} 1 & \text{if } i + k \geq 100 \\ 0 & \text{if } i + k < 100. \end{cases}$$

2.3   Dynamic programming for the two-players case

To conclude this section, we consider for the game of Pig the case of two players. The players *alternate* in taking turns rolling the die. The first player to reach 100 points is the winner. Since there is an advantage in going first in Pig, it is assumed that a toss of a fair coin decides which player begins in the game of Pig. Then, under optimal play of both players, each player has a probability of 50% of being the ultimate winner. But how to calculate the optimal decision rule. By the assumption that players alternate in taking turns rolling the die, the optimal decision rule can be computed by using standard dynamic programming techniques. In the final section of this paper we will consider a variant of the game of Hog in which in each round the two players have to decide simultaneously how many dice to roll, where the players cannot observe each other's decision. Such a variant with simultaneous actions of both players in the same turn can also be considered for the game of Pig. Then, methods from standard dynamic programming cannot be longer used but instead one should use much more involved methods from game theory.

The dynamic programming solution for the game of Pig with two players who alternate in taking turns proceeds as follows. The state s is defined by $s = ((i, k), j)$, where (i, k) indicates that the player whose turn it is has a score i and has k points accumulated so far in the current turn and j indicates that the opponent's score is j. Define the value function $P(s)$ by

$P(s)$ = the probability of the player winning whose turn it is given that the present state is state s,

where $P(s)$ is taken to be equal to 1 for those $s = ((i, k), j)$ with $i + k \geq 100$ and $j < 100$. To write down the optimality equations, we use the simple observation that the probability of a player winning after rolling a 1 or holding is one minus the probability that the other player will win beginning with the next turn. Thus, for state $s = ((i, k), j)$ with $k = 0$,

$$P((i, 0), j) = \frac{1}{6}[1 - P((j, 0), i)] + \sum_{r=2}^{6} \frac{1}{6} P((i, r), j).$$

For state $s = ((i, k), j)$ with $k \geq 1$ and $i + k, j < 100$,

$$P((i, k), j)) = \min\left[1 - P((j, 0), i + k),\right.$$
$$\left. \frac{1}{6}[1 - P((j, 0), i)] + \sum_{r=2}^{6} \frac{1}{6} P((i, k + r), j)\right],$$

where the first expression in the right side of the last equation corresponds to the decision "hold" and the second expression corresponds to the decision "roll". Using the method of successive substitution, these optimality equations can be numerically solved, yielding the optimal decision to take in any state $s = ((i, k), j)$.

3   The game of Hog

We first give the analysis for the single-player version of the game. In the game of Hog (Fast Pig) the player has to decide in each turn how many dice to roll simultaneously. A similar heuristic as the "hold at 20" rule manifests itself in the game of Hog (Fast Pig). This heuristic is the "five dice" rule that prescribe to roll five dice in each turn. The rationale of this rule is as follows: five dice are the optimal number of dice to roll when the goal is to maximize the expected value of the score in a *single* turn. The expected value of the total score in a single turn with d dice is $(1 - (5/6)^d) \times 0 + (5/6)^5 \times 4d$ and this expression is maximal for $d = 5$. The number of turns needed to reach 100 points has the expected value 13.623 when the "five dice" rule is used, while the expected value of the number of turns needed to reach 100 points has the value 13.039 when an optimal decision rule is used. Again, a very good performance of the heuristic rule when the criterion is to minimize the expected number of

turns. However, the story is different when the criterion is to maximize the probability of reaching 100 points in no more than N turns with N given. This probability has the respective values 0.0056, 0.0610, 0.2759, 0.6993, and 0.9159 for $N=5$, 7, 10, 15, and 20 when the "five dice" rule is used, while the respective values are 0.0869, 0.1914, 0.4240, 0.8004, and 0.9631 under an optimal rule.

3.1   Analysis for the single-player version

For both the criterion of the expected number of turns to reach 100 points and the criterion of the probability to reach 100 points in a given number of turns, we will give a unified analysis that covers both the heuristic rule and the optimal rule. Instead of taking the state as the current score of the player, it is convenient to define the state as the number of points the player still needs to reach the goal when a new turn is about to begin. The decision d in any state s prescribes to roll simultaneously d dice. Denoting the set of possible decisions in state s by $D(s)$, we can give a unified analysis by taking $D(s) = \{5\}$ for the analysis of the "five dice" rule and taking $D(s) = \{1, 2, \ldots, D\}$ for the analysis of an optimal rule, where D is finite but large number. A key ingredient in the computations are the probabilities $q_i^{(d)}$ to be defined by

$$q_i^{(d)} \quad = \quad \text{the probability of obtaining } i \text{ points in a turn}$$
$$\text{when the decision is to roll } d \text{ dice,}$$

To calculate this probabilities, we need the probability $r_i^{(d)}$ which is defined as the conditional probability that a roll of d dice gives i points given that no 1s are rolled. Using the fact that the conditional distribution of the outcome of the roll of a single die is uniformly distributed on the integers $2, \ldots, 6$ given that the outcome is not 1, it follows that the $r_i^{(d)}$ can be recursively calculated from the convolution formula

$$r_i^{(d)} = \sum_{j=2}^{6} \frac{1}{5} r_{i-j}^{(d-1)} \quad \text{for } i = 2d, 2d+1, \ldots, 6d, \quad \text{and} \quad r_i^{(d)} = 0 \text{ otherwise,}$$

with the convention $r_0^{(0)} = 1$ and $r_i^{(0)} = 0$ for $i \neq 0$. Next, the $q_i^{(d)}$ follow from

$$q_0^{(d)} = 1 - \left(\frac{5}{6}\right)^d \quad \text{and} \quad q_i^{(d)} = \left(\frac{5}{6}\right)^d r_i^{(d)} \quad \text{for } i, d = 1, 2, \ldots \ .$$

For the criterion of the expected number of turns to reach the goal, we define the value-function $V(i)$ as the minimal expected number of additional turns to get i additional points when using the decision sets $D(i)$ (in case $D(i) = \{5\}$ for all i, the minimal expected number should of course be read as the expected number). The goal is to calculate $V(100)$. Then, letting $V(i) = 0$ for $i \leq 0$, we have the dynamic programming equation:

$$V(i) = \min_{d \in D(i)} \left\{ 1 + q_0^{(d)} V(i) + \sum_{r=2d}^{6d} q_r^{(d)} V(i-r) \right\}$$

or, equivalently,

$$V(i) = \min_{d \in D(i)} \left\{ \frac{1}{1 - q_0^{(d)}} \left[1 + \sum_{r=2d}^{6d} q_r^{(d)} V(i-r) \right] \right\} .$$

The function values $V(i)$ can be computed recursively for $i = 1, \ldots, 100$.

For the criterion of the probability of reaching the goal within a given number of N turns, the value function $P_m(i)$ is defined as the maximal probability to get i additional points when no more than m turns are allowed, where m runs from 1 to N. We wish to find $P_N(100)$. Letting $P_m(i) = 1$ for $i \leq 0$, we have the dynamic programming equation:

$$P_m(i) = \min_{d \in D(i)} \left\{ q_0^{(d)} P_{m-1}(i) + \sum_{r=2d}^{6d} q_r^{(d)} P_{m-1}(i-r) \right\} .$$

The recursion is initiated with the boundary condition $P_0(i) = 1$ for $i \leq 0$ and $P_0(i) = 0$ for $i > 0$.

3.2   Analysis for the case of two players

To conclude this section, we consider for the game of Hog the original case of two players. The players *alternate* in taking turns rolling the die. The first player to reach 100 points is the winner. Since there is an advantage in going first in Hog, it is assumed that a toss of a fair coin decides which player begins in the game of Hog. The dynamic programming solution for the game of Hog with two players who alternate in taking turns proceeds as follows. The state defined as $s = (i, j)$, where i indicates the number of points the player whose turn it is still needs for the winning score and j indicates the number of points the opponent

still needs for the winning score. Define the value function $P(s)$ as the win probability of the player whose turn it is given that the present state is state s and both players act optimally in each turn. Then, for the states (i, j) with $i, j > 0$, the optimality equation is

$$P(i,j) = \max_{d=1,\ldots,D} \left\{ q_0^{(d)}[1 - P(j,i)] + \sum_{r=2d}^{6d} q_r^{(d)}[1 - P(j, i - r)] \right\},$$

with the convention $P(j, k) = 0$ for $j > 0$ and $k \leq 0$, where D denotes the largest number of dice that can be rolled.

4 A game-theoretic problem

This section considers a variant of the game of Hog, where the two players have to take *simultaneously* a decision in each round of the game. This variant deals with a television game show and the problem is as follows. At the end of the television game show the two remaining contestants have to play a dice game. The contestants each sit behind a panel with a battery of buttons numbered as $1, 2, \ldots, D$, say $D{=}10$. In each stage of the game, both contestants must simultaneously press one of the buttons, where the contestants cannot observe each other's decision. The number on the button pressed by the contestant is the number of dice that are thrown for the contestant. For each contestant the score of the throw for that contestant is added to his/her total, provided that none of the dice in that throw showed the outcome 1; otherwise no points are added to the current total of the candidate. The candidate who first reaches a total of 100 points is the winner. In case both candidates reach the goal of 100 points in the same move, the winner is the candidate who has the largest total. In the event of a tie, the winner is determined by a toss of a fair coin. At each stage of the game both candidates have full information about his/her own current total and the current total of the opponent. What does the optimal strategy look like?

The computation and the structure of an optimal strategy is far more complicated than in the problems discussed before. The optimal rules for the decision problems considered before were deterministic, but the optimal strategy will involve randomized actions for the problem of the television game show. In zero-sum games randomization is a key ingredient of the optimal strategy.

We will give only an outline of the solution procedure. The rules of the game state that in each round the two players have to decide at the same moment upon the number of dice to use, so without seeing what the opponent is doing but knowing and using the scores so far. So, after a number of rounds player 1 still needs a points and player 2 needs b points. This describes the state of the system. If now player 1 decides to use k dice and player 2 uses l dice, then the state changes from (a, b) into $(a - i, b - j)$ with probability $q_i^{(k)} q_j^{(l)}$. The game is a stochastic terminating zero-sum game. The value of the game is defined as the probability that player 1 will win minus the probability that player 2 will win, given that both players play optimally. Define

$$V(a, b) = \begin{cases} 1 & \text{if } a < b \text{ and } a \leq 0 \\ 0 & \text{if } a = b \leq 0 \\ -1 & \text{if } a > b \text{ and } b \leq 0. \end{cases}$$

We want to determine $V(a, b)$ for both a and b positive and the optimal, possibly randomized, actions that guarantee this value. The value of the game and the optimal moves of the two players can be computed by repeatedly solving the appropriate matrix games. Let $x = (x_1, x_2, \ldots, x_D)$ be a randomized move for player 1, i.e., player 1 rolls d dice with probability x_d, where

$$\sum_d x_d = 1.$$

The first approach to think off is to recursively compute $V(a, b)$ via a sequence of LP-problems, starting in $(a, b) = (1, 1)$ and working backwards, step by step, until $(a, b) = (G, G)$ with $G = 100$. This requires to solve the optimization problem:

$$\text{maximize } V \quad \text{subject to}$$

$$\sum_d x_d \left(\sum_{i+j>0} q_i^{(d)} q_j^{(l)} V(a - i, b - j) + q_0^{(d)} q_0^{(l)} V \right) \geq V, \quad l = 1, \ldots, D,$$

$$x_d \geq 0, \quad d = 1, \ldots, D, \quad \sum_d x_d = 1 \,,$$

where, for $i + j > 0$, the values $V(a - i, b - j)$ have been computed before and hence are known. (V is unrestricted in sign.) However,

this optimization problem is not exactly an LP-problem because of the nonlinear term

$$\sum_d x_d q_0^{(d)} q_0^{(l)} V.$$

To make an LP-approach possible, we proceed as follows. Define $V^{(n)}(a, b)$ as the value of the game if it is played at most n times with a terminal reward 0, if after n steps the game has not yet reached the payoff-zone. Thus, $V^{(0)}(a, b) := 0$ if $a > 0$ and $b > 0$. Also, define

$$V^{(n)}(a, x, b, l) = \sum_d x_d \sum_{i,j} q_i^{(d)} q_j^{(l)} V^{(n-1)}(a - i, b - j), \; n > 0 \;,$$

with the convention that, for $n \geq 0$ and $a \leq 0$ or $b \leq 0$, $V^{(n)}(a, b) = V(a, b)$. Then, in iteration n for state (a, b), the value of the game and an optimal move for player 1 can be obtained from the following LP-problem for a matrix game:

$$\text{maximize } V \quad \text{subject to}$$

$$V^{(n)}(a, x, b, l) \geq V, \quad l = 1, \ldots, D,$$

$$x_d \geq 0, \quad d = 1, \ldots, D, \quad \sum_d x_d = 1.$$

The optimal value V satisfies $V = V^{(n)}(a, b)$ and the optimal $x^{(n)}(a, b)$ represents an optimal move for player 1 in state (a, b) in iteration n. $V^{(n)}(a, x, b, l)$ converges exponentially fast to the value of the game, and $x^{(n)}$ is nearly optimal for n sufficiently large. Of course, for reasons of symmetry, the optimal move for player 2 in state (a, b) is the same as the optimal move for player 1 in state (b, a). The computations for an optimal strategy are formidable for larger values of D with D being the maximum number of dice that can be rolled. The computations reveal that the optimal strategy uses indeed randomized actions. For example, for the case of $D = 5$, player 1 uses 2, 4 or 5 dice with respective probabilities 0.172, 0.151 and 0.677 when player 1 still needs 1 point and player 2 still needs 3 points. Also, the numerical calculations reveal a kind of turnpike result: for states (i, j) sufficiently far from $(0, 0)$ the players use non-randomized decisions only (for example in state (5,13) in which player 1 still needs 5 points and player 2 still needs 13 points, player 1 uses 4 dice and player 2 uses 5 dice when $D = 5$). It would be nice to have a theoretical proof of this intuitively obvious turnpike result

as well to have a theoretical proof of certain monotonicity properties of the optimal strategy.

There are various modifications of the television game show possible. To mention a few:

1. Suppose that a player gets not only a score 0 but also loses all (or some of) the points collected so far if there is an outcome 1 in the throw of his dice.

2. Suppose the players know the outcomes of their own throws, but don't know what the other player has been doing at all. This is a game with imperfect information. Is it possible to determine an optimal strategy?

3. Suppose that, in addition to the previous situation, you also know how many dice your opponent has used. This too is a game with imperfect information.

Henk Tijms
Department of Econometrics and Operations Research,
Vrije University, Amsterdam,
The Netherlands,
tijms@feweb.vu.nl

References

[1] Derman C., Finite State Markovian Decision Problems, Academic Press, New York, 1970.

[2] Hernández-Lerma O., Adaptive Markov Control Processes, Springer Verlag, New York, 1989.

[3] Neller T. W.; Presser C. G. M., *Optimal play of the dice game Pig*, The UMAP Journal **25** (2004), 25–47. See also the material on the website http://cs.gettysburg.edu/projects/pig/

[4] Tijms, H. C., Understanding Probability, Chance Rules in Everyday Life, 2nd edition, Cambridge University Press, Cambridge, 2007.

[5] Tijms H. C.; Van der Wal J., *A real-world stochastic two-person game*, Probability in the Engineering and Informational Sciences **20** (2006), 599–608.