J. Appl. Prob. **53,** 91–105 (2016)
*doi:*10.1017/jpr.2015.11
© *Applied Probability Trust* 2016

A FINITE EXACT ALGORITHM TO SOLVE A DICE GAME

FABIÁN CROCCE,* *KAUST and Universidad de la República*
ERNESTO MORDECKI,** *Universidad de la República*

Abstract

We provide an algorithm to find the value and an optimal strategy of the Ten Thousand dice game solitaire variant in the framework of Markov control processes. Once an optimal critical threshold is found, the set of nonstopping states of the game becomes finite and the solution is found by a backwards algorithm that gives the values for each one of these states of the game. The algorithm is finite and exact. The strategy to find the critical threshold comes from the *continuous pasting condition* used in optimal stopping problems for continuous-time processes with jumps.

Keywords: Markov control process; dice game; optimal control; finite exact algorithm

2010 Mathematics Subject Classification: Primary 90C40
Secondary 60G40

1. Introduction

The emergence of probability theory is closely related to the practice of dice games, as masterly exposed by Hald [4]. Inspired by this fact, and by the power of the mathematical method, we consider a popular dice game known as 'Ten Thousand'.

In this game, played with five dice, several players, by turns, roll the dice several times. Each possible outcome of a roll has an assigned score, which may be 0. If after rolling the dice, a strictly positive score is obtained, the player can roll again some of his/her dice to increase his/her turn score or he/she can stop rolling to add that score into his/her general account, ending his/her turn; otherwise, if a null score is obtained, the player loses the accumulated turn score, thus ending his/her turn.

As a consequence, each turn consists of a sequence of rolls, ending either when the player obtains no score or when he/she decides to stop and add the turn score to his general score account. Here arises the problem of making an optimal decision. The goal of the game is to be the first player to attain a certain number of points, usually 10 000. This game, also known as Zilch and Farkle, among other names, has several versions with minor differences (some of them played with six dice).

In this paper we consider the problem that faces an individual player who aims to maximize his turn score, in what can be considered a solitaire variant of the game. By modelling this optimization problem within the framework of the Markov control processes (MCP) we provide a finite exact algorithm to obtain the value function and an optimal strategy for the considered game. We expect that this algorithm can constitute a first step in finding the optimal strategies for the original multiplayer game.

Received 12 June 2012; revision received 23 March 2015.

* Postal address: King Abdullah University of Science and Technology (KAUST), P. O. Box 4700, Thuwal, Jeddah, 23955-6900 Saudi Arabia. Email address: fcrocce@fing.edu.uy
** Postal address: Centro de Matemática, Facultad de Ciencias, Universidad de la República. Iguá 4225, CP 11400 Montevideo, Uruguay.

Despite the popularity of dice games and the interest of the probabilistic community in the topic, only a few references can be found concerning the family of dice games we consider. Roters [10] solved the solitaire version of the Pig game (a simpler variant with one die; see Section 5.2). Shortly afterward, Haigh and Roters [3] considered a variant of the same solitaire game consisting of minimizing the number of turns needed to reach a target. Concerning competitive games, Neller and Presser [7] solved the competitive version of the Pig game. More recently, Tijms [11] considered also the competitive Pig game and a simultaneous decision making variant, named Hog (see also [12] where variants of the game are analysed).

As mentioned before, we consider that the theoretical framework to study the solitaire game is the theory of MCPs. Our problem happens to be a transient MCP under the expected total cost criteria, with a countably infinite state space, a finite number of actions at each state, and an unbounded reward function.

The 1967 article by Blackwell [1] is among the first papers that considered the expected total cost criteria. A general reference for Markov decision processes can be found in Puterman [9]. Filar and Vrieze [2] considered both solitaire and competitive games with finite state space. Pliska [8] considered an MCP with infinite state space with bounded payoff function. Our example does not fit in this framework, and similar simpler examples (see Section 5.2) show that the *dynamic programming equations* (DPEs) are not expected to have a unique solution. This situation is considered by Hernández-Lerma *et al.* [5], where an ad hoc norm was proposed to restrict the search space for the value function. Nevertheless, the conditions required in [5] to prove the transience of the process under all possible strategies require certain uniformity; a condition that does not seem to be fulfilled in our case.

The rest of the paper is as follows. In Section 2 we describe the rules of the game and in Section 3 its mathematical model. In Section 4 we present our main result, which consists of a finite algorithm that provides an exact solution to the DPEs, leading to the solution of the game. In Section 5 we solve some related games and in Section 6 we present a brief discussion and some conclusions.

2. Ten Thousand game

In the solitaire version of the Ten Thousand game that we consider, one player aims to maximize the score of one turn when playing the game described above. This turn consists of a sequence of rolls.

2.1. Description of one roll

The player rolls n dice ($1 \le n \le 5$). A score is assigned to each one of the dice combinations described in Table 1. We introduce a *dice configuration* $i = [f, o, t]$ to summarize the outcome of a roll that has f 5s, o 1s, and three ts for $t = 2, 3, 4, 6$. By $t = 0$ we denote the absence of three of a kind for these numbers.

Mathematically equivalent, to simplify the treatment, we divide all point amounts by 50. To link our model with the real game, the reader can consider 'big points' or 'chips' costing 50 points each. Observe that three 5s or three 1s are counted in f and o, respectively. We denote by $\mathbf{0} = [0, 0, 0]$ the nonscoring configuration and by $\mathcal{I}$ the set of configurations with a nonvanishing score. The configurations are listed in Table 2. Note that, as we consider configurations obtained after rolling five or fewer dice, we get the restriction $d(i) \le 5$.

TABLE 1: Combination of dice and corresponding scores in the Ten Thousand dice game.

Combination	Points	Chips	Combination	Points	Chips
Each 5	50	1	Three 4s	400	8
Each 1	100	2	Three 5s	500	10
Three 2s	200	4	Three 6s	600	12
Three 3s	300	6	Three 1s	1000	20

To each configuration $i = [f, o, t]$ we associate a score $s(i)$, a number of scoring dice $d(i)$, and a number of scoring combinations $c(i)$, through the equations:

$$\begin{aligned} s(i) &= f + 2o + 14\,\mathbf{1}_{\{o\geq 3\}} + 7\,\mathbf{1}_{\{f\geq 3\}} + 2t, \\ d(i) &= f + o + 3\,\mathbf{1}_{\{t\neq 0\}}, \\ c(i) &= f + o + \mathbf{1}_{\{t\neq 0\}}, \end{aligned}$$

where $\mathbf{1}$ is the indicator function. Observe that three 5s or three 1s count as three scoring combinations.

Definition 2.1. We say that a configuration j is *smaller* than a configuration i when the set of scoring combinations of j is a nonvoid strict subset of the set of scoring combinations of i. In this case, we write $j \prec i$ and have $s(j) < s(i)$, $d(j) < d(i)$, and $0 < c(j) < c(i)$.

2.2. Description of the game

We represent by $i_1, \ldots, i_k, \ldots$ the successive outcomes of a sequence of rolls, by τ_k the points accumulated up to the kth dice roll, and by n_k the number of dice available to roll after the kth roll. The values of τ_k and n_k depend on τ_{k-1}, the random outcome i_k, and the action chosen by the player, as explained below. In the first roll the player begins with 0 points and five dice to roll, i.e. $\tau_0 = 0$ and $n_0 = 5$. Depending on the configuration i_k obtained in the kth roll, the player has the following available actions.

(i) If no score is obtained ($s(i_k) = 0$) the player loses the accumulated score ($\tau_k = 0$) ending the game.

(ii) If all the dice rolled give a score ($d(i_k) = n_{k-1}$) then the score increases to $\tau_k = \tau_{k-1} + s(i_k)$ and the player can either stop, earning τ_k points and ending the game, or he/she can roll the five dice again ($n_k = 5$).

(iii) If not all rolled dice give a score and only one scoring combination is obtained ($d(i_k) < n_{k-1}$ and $c(i_k) = 1$), then the score increases to $\tau_k = \tau_{k-1} + s(i_k)$ and the player can either stop, earning τ_k points, or he/she can roll $n_k = n_{k-1} - d(i_k)$ dice.

(iv) If not all the dice rolled give a score and more than one scoring combination is obtained ($d(i_k) < n_{k-1}$ and $c(i_k) > 1$), then, in addition to the actions described in (iii), the player can choose a configuration $j_k \prec i_k$ (see Definition 2.1), increasing the score only to $\tau_k = \tau_{k-1} + s(j_k)$ but obtaining $n_k = n_{k-1} - d(j_k)$ dice to roll again (the player resigns a score to roll more dice).

In Table 3 we summarize the actions described above. After each roll, the player continues rolling the nonscoring dice (and eventually part of the scoring dice in case (iv)), making decisions according to (ii)–(iv) until he/she decides to stop or until he/she gets no scoring dice (i). The goal

TABLE 2: Configurations obtained in one roll of the Ten Thousand game.

Score dice	Confidence	Combination	Score	Frequency				
$d(i)$	f, o, t	$c(i)$	$s(i)$	$f(1, i)$	$f(2, i)$	$f(3, i)$	$f(4, i)$	$f(5, i)$
0	0, 0, 0	0	0	4	16	60	204	600
1	1, 0, 0	1	1	1	8	48	240	1020
	0, 1, 0	1	2	1	8	48	240	1020
2	2, 0, 0	2	2		1	12	96	600
	1, 1, 0	2	3		2	24	192	1200
	0, 2, 0	2	4		1	12	96	600
3	3, 0, 0	3	10			1	16	160
	2, 1, 0	3	4			3	48	480
	1, 2, 0	3	5			3	48	480
	0, 3, 0	3	20			1	16	160
	0, 0, 2	1	4			1	13	106
	0, 0, 3	1	6			1	13	106
	0, 0, 4	1	8			1	13	106
	0, 0, 6	1	12			1	13	106
4	4, 0, 0	4	11				1	20
	3, 1, 0	4	12				4	80
	2, 2, 0	4	6				6	120
	1, 3, 0	4	21				4	80
	0, 4, 0	4	22				1	20
	1, 0, 2	2	5				4	65
	1, 0, 3	2	7				4	65
	1, 0, 4	2	9				4	65
	1, 0, 6	2	13				4	65
	0, 1, 2	2	6				4	65
	0, 1, 3	2	8				4	65
	0, 1, 4	2	10				4	65
	0, 1, 6	2	14				4	65
5	5, 0, 0	5	12					1
	4, 1, 0	5	13					5
	3, 2, 0	5	14					10
	2, 3, 0	5	22					10
	1, 4, 0	5	23					5
	0, 5, 0	5	24					1
	2, 0, 2	3	6					10
	2, 0, 3	3	8					10
	2, 0, 4	3	10					10
	2, 0, 6	3	14					10
	1, 1, 2	3	7					20
	1, 1, 3	3	9					20
	1, 1, 4	3	11					20
	1, 1, 6	3	15					20
	0, 2, 2	3	8					10
	0, 2, 3	3	10					10
	0, 2, 4	3	12					10
	0, 2, 6	3	16					10

TABLE 3: Possible actions after rolling n dice and obtaining configuration i.

$s(i)$	$d(i)$	$c(i)$	Action
0	0	0	Stop
>0	n		Stop/Roll five dice
>0	$<n$	1	Stop/Roll $n-d(i)$ dice
>0	$<n$	>1	Stop/Roll $n-d(i)$ dice/Roll $n-d(j)$ dice $(j \prec i)$

of the game is to maximize the accumulated score at the end of the game. Observe that, unless situation (ii) is achieved, the number of rolling dice of the subsequent rolls strictly decreases, as at each step some scoring dice should be set aside. It should be noted that every time the player decides to roll, he/she risks losing the accumulated turn score and finishing the game with 0 points.

Remark 2.1. Observe that, according to (ii), the player can theoretically roll the dice an arbitrary number of times, accumulating an arbitrarily large score.

3. Mathematical model of the solitaire game

We model the solitaire Ten Thousand game as an MCP in which one player aims to maximize the nondiscounted sum of the rewards. To this end we define the state space $\mathcal{X}$, the sets $\mathcal{A}(\boldsymbol{x})$ of actions the player can take at each state $\boldsymbol{x} \in \mathcal{X}$, and for each pair $(\boldsymbol{x}, \boldsymbol{a})$ with $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{x})$ we define a reward $r(\boldsymbol{x}, \boldsymbol{a})$ and a probability distribution $q(\cdot \mid \boldsymbol{x}, \boldsymbol{a})$ over the state space $\mathcal{X}$ for the following state.

3.1. States

Each state $\boldsymbol{x}$ of the solitaire game includes three elements: an amount τ of accumulated chips; the last obtained configuration $i = [f, o, t]$; and a number $n = 1, \dots, 5$ of available dice to roll. Note that, as the amount of rolled dice is at most five, there is a restriction on the obtained configuration i after the roll, and the amount n of available dice to roll. We have $d(i) + n \leq 5$. An exception to this rule is when all rolled dice give a score; in this case $i \in \mathcal{I}$ and $n = 5$. It is necessary to add two singular states: the initial state $\iota = (0, \varnothing, 5)$ and a final absorbing state $\Delta = (0, \mathbf{0}, 0)$. In this way we define the state space of the MCP as

$$\mathcal{X} = \{\iota, \Delta\} \cup \{\boldsymbol{x} = (\tau, i, n) \in \{1, 2, \dots\}\mathcal{I}\{1, 2, 3, 4, 5\} \colon i \in \mathcal{I}(5 - n) \text{ if } n \neq 5\},$$

where $\mathcal{I}(m)$ represents the set of scoring configurations with up to m scoring dice, i.e.

$$\mathcal{I}(m) = \{i \in \mathcal{I} \colon d(i) \leq m\}.$$

We observe that the state space of the game is a countably infinite set, in accordance with Remark 2.1. When states $\boldsymbol{x}$, $\boldsymbol{y}$, or $\boldsymbol{z}$ are considered we assume that $\boldsymbol{x} = (\tau_x, i_x, n_x)$, $\boldsymbol{y} = (\tau_y, i_y, n_y)$, and $\boldsymbol{z} = (\tau_z, i_z, n_z)$ to denote their components.

To model the possibility of resigning some scoring dice to obtain more dice to roll (see (iv) in Section 2.2) the following definition is useful.

Definition 3.1. We say that the state $\boldsymbol{x} \in \mathcal{X}$ is smaller than $\boldsymbol{y} \in \mathcal{X}$, and denote $\boldsymbol{x} \prec \boldsymbol{y}$, if: $n_y \neq 5$; $\tau_x = \tau_y - s(i_y) + s(i_x)$; $i_x \prec i_y$ and $n_x = n_y + d(i_y) - d(i_x)$. The interpretation of this definition is that $\boldsymbol{m}_{\text{to}\,x}$ is an available action from state $\boldsymbol{y}$.

3.2. Actions, transition probabilities, and rewards

When from a state $\boldsymbol{x}$ ($\boldsymbol{x} \neq \Delta$) all the $n_{\boldsymbol{x}}$ available dice are rolled (an action that we call *roll* and denote by $\boldsymbol{r}$) the probability distribution for the next state $\boldsymbol{y}$ is determined by

$$q(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{r}) = \begin{cases} \dfrac{f(n_{\boldsymbol{x}}, \boldsymbol{0})}{6^{n_{\boldsymbol{x}}}} & \text{if } \boldsymbol{y} = \Delta, \\ \dfrac{f(n_{\boldsymbol{x}}, j)}{6^{n_{\boldsymbol{x}}}} & \text{if } \boldsymbol{y} = (\tau_{\boldsymbol{x}} + s(j), j, n_{\boldsymbol{x}} - d(j) + 5\,\mathbf{1}_{\{n_{\boldsymbol{x}} = d(j)\}}), \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

where $f(n_{\boldsymbol{x}}, j)$ are the frequencies listed in Table 2. Observe that the indicator $\mathbf{1}_{\{n_{\boldsymbol{x}} = d(j)\}}$ models the right to throw the five dice again when all the dice give a score (case (ii) above). The previous equation also applies in the $\boldsymbol{x} = \iota$ case, in which $\tau_{\boldsymbol{x}} = 0$ and $n_{\boldsymbol{x}} = 5$.

There are several actions the player can choose: to roll all the available dice $\boldsymbol{a} = \boldsymbol{r}$, to stop the game $\boldsymbol{a} = \boldsymbol{s}$, and to 'make a move' $\boldsymbol{a} = \boldsymbol{m}_\bullet$ (which could be a set of actions). We continue by describing the respective actions associated with each state, with their corresponding rewards and probability transitions.

1. In the initial state $\iota = (0, \varnothing, 5)$ the player has the only action $\boldsymbol{r}$, with null reward and probability transitions $q(\cdot \mid \iota, \boldsymbol{r})$ given in (3.1).

2. In a state $\boldsymbol{x}$ with one scoring combination (i.e. $c(i_{\boldsymbol{x}}) = 1$) or with five dice to roll ($n_{\boldsymbol{x}} = 5$), the player has two actions: $\boldsymbol{s}$, with reward τ and probability 1 of going to Δ; and $\boldsymbol{r}$, which has null reward and probability transitions $q(\cdot \mid \boldsymbol{x}, \boldsymbol{r})$ given in (3.1).

3. In a state $\boldsymbol{x}$ with more than one scoring combination (i.e. $c(i_{\boldsymbol{x}}) > 1$) and $n_{\boldsymbol{x}} < 5$, besides the same two actions described in item 2, the player has the possibility of making a *move* to a smaller state $\boldsymbol{y}$ (see Definition 3.1). This action, which is denoted by $\boldsymbol{m}_{\text{to } \boldsymbol{y}}$, has null reward and probability 1 of going to the state $\boldsymbol{y}$.

4. For the sake of completeness, we assume that the state Δ is absorbing and has a unique action *wait* $\boldsymbol{w}$ with null reward.

It should be noted that the only nonnull reward corresponds to action $\boldsymbol{s}$ and after taking this action the game ends. Thus, we have an MCP under the total-cost criteria, where (if $\boldsymbol{x}_j$ is the state at time j, and $\boldsymbol{a}_j$ is the action taken at that time)

$$\sum_{j=0}^{\infty} r(\boldsymbol{x}_j, \boldsymbol{a}_j) = \sum_{j=0}^{\infty} \tau_j\, \mathbf{1}_{\{\boldsymbol{a}_j = \boldsymbol{s}\}}\,.$$

Therefore, the objective function has only one nonvanishing addend (the accumulated score at the stopping moment) or vanishes after a nonscoring roll.

The possible moves $\boldsymbol{m}_\bullet$ the player can make from a given state can be described by enumerating the scoring dice he resigns, i.e. the difference between the original configuration $i_{\boldsymbol{x}}$ and the new configuration $i_{\boldsymbol{y}}$. With this observation in mind, the number of possible actions is reduced to 16, which are described in Table 4.

3.3. Markov control problem

Given a countable state space $\mathcal{X}$ and a countable set of actions $\mathcal{A}(\boldsymbol{x})$ for each $\boldsymbol{x} \in \mathcal{X}$, consider $\mathbb{K} := \{(\boldsymbol{x}, \boldsymbol{a})\colon \boldsymbol{x} \in \mathcal{X}, \boldsymbol{a} \in \mathcal{A}(\boldsymbol{x})\}$, $\mathbb{H}_0 := \mathcal{X}$, and $\mathbb{H}_n := \mathbb{K}^n \times \mathcal{X}$. An element $h_n = (\boldsymbol{x}_0, \boldsymbol{a}_0, \dots, \boldsymbol{x}_n) \in \mathbb{H}_n$ is called an n-history. A strategy is a sequence $\pi = (\pi_0, \pi_1, \dots)$,

TABLE 4: Possible actions.

Notation	Description
s	Stop
r	Roll all the nonscoring dice
m_5	Resign one 5
m_1	Resign one 1
m_{55}	Resign two 5s
m_{51}	Resign one 5 and one 1
m_{11}	Resign two 1s
m_{551}	Resign two 5s and one 1
m_{511}	Resign one 5 and two 1s
$m_{t_k}: k = 1, \dots, 6$	Resign a turn of k
w	Wait

where π_n is a stochastic kernel defined on $\mathbb{H}_n$ such that $\pi_n(\cdot \mid h_n)$ is supported on $\mathcal{A}(x_n)$. If for every n and for every $h_n \in \mathbb{H}_n$, the measure $\pi_n(\cdot \mid h_n)$ is supported in just one action $a \in \mathcal{A}(x_n)$, the strategy is said to be *pure*, and each π_n can be seen as a function that, given the history, chooses the next action. A strategy π is said to be *Markovian* if it satisfies $\pi_n(\cdot \mid h_n) = \pi_n(\cdot \mid h'_n)$ provided that the last state in both histories coincide, in this case the notation $\pi_n(\cdot \mid x_n)$ is used instead of $\pi_n(\cdot \mid h_n)$. A Markovian strategy is said to be *stationary* if $\pi_n(\cdot \mid x_n) = \pi_m(\cdot \mid x_m)$ provided that $x_n = x_m$; in this case one can avoid the n in the notation using just $\pi(\cdot \mid x_n)$.

Remark 3.1. Denoting $\mathcal{A} = \bigcup_{x \in \mathcal{X}} \mathcal{A}(x)$, a pure stationary strategy π can be understood as function $A\colon \mathcal{X} \to \mathcal{A}$ that satisfies the constraint $A(x) \in \mathcal{A}(x)$ for all $x \in \mathcal{X}$, where $A(x)$ is the action that has full probability to be picked when the system is in the state x, i.e. $\pi(A(x) \mid x) = 1$. In this case we say that the pure stationary strategy π has rule of decision A.

Given a strategy π and an initial state $x \in \mathcal{X}$, a probability measure $\mathbb{P}_x^\pi$ on $\mathbb{K}^\infty$ with the product σ-algebra is determined in a canonical way. Denoting by $\mathbb{E}_x^\pi$ the expectation under $\mathbb{P}_x^\pi$ the player expects to receive when starting from x and following the strategy π,

$$V(x, \pi) = \mathbb{E}_x^\pi\left(\sum_{j=0}^{\infty} r(x_j, a_j)\right). \tag{3.2}$$

For a more detailed exposition and subclasses of strategies we refer the reader to [5], and [9]. Denoting by Π the set of all possible strategies, our objective is to find the value function $V^*\colon \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ and an optimal strategy $\pi^* \in \Pi$ such that

$$V^*(x) = \sup_{\pi \in \Pi} V(x, \pi) = V(x, \pi^*) \quad \text{for all } x \in \mathcal{X}. \tag{3.3}$$

Observe that, from the definition of the MCP, $V(\Delta, \pi) = 0$ for all possible strategy π, then we have $V^*(\Delta) = 0$.

Denote by $F(\mathcal{X}, \mathbb{R} \cup \{\infty\})$ the set of extended real functions with domain $\mathcal{X}$, and consider the operator $L\colon F(\mathcal{X}, \mathbb{R} \cup \{\infty\}) \to F(\mathcal{X}, \mathbb{R} \cup \{\infty\})$ acting by

$$\mathrm{L}\, V(x) = \sup_{a \in \mathcal{A}(x)} \mathrm{L}_a\, V(x) \tag{3.4}$$

with $\mathrm{L}_a\, V(x) = r(x, a) + \sum_{y \in \mathcal{X}} V(y) q(y \mid x, a)$.

The value function V^* is the smallest nonnegative solution of the DPE

$$V = \mathrm{L}\, V. \tag{3.5}$$

Furthermore, defining the sequence of extended real functions $\{W_k : k \in \mathbb{N}\}$ with domain $\mathcal{X}$, such that W_0 is the null function and $W_{k+1} = \mathrm{L}\, W_k (k = 0, 1, \dots)$, it follows that the defined sequence is monotone nondecreasing for every $\boldsymbol{x}$ and

$$V^*(\boldsymbol{x}) = \lim_{k\to\infty} W_k(\boldsymbol{x}). \tag{3.6}$$

The previous assertions are proved in [1, Theorem 3] for the case of a bounded reward function r. To see that the result remains valid in our case we consider for any $m = 1, 2, \dots$ an MDP with the same state space, actions, and transition probabilities as the one already defined but with a bounded reward function r_m such that $r_m(\boldsymbol{x}, \boldsymbol{a}) = r(\boldsymbol{x}, \boldsymbol{a}) \wedge m$. For this new MDP we denote by $V_m(\boldsymbol{x}, \pi)$ and $V_m^*(\boldsymbol{x})$ the quantities defined in (3.2) and (3.3), respectively, and we denote by $W_{m,k}(\boldsymbol{x})$ the sequence on the right-hand side of (3.6).

From the aforementioned result in [1], we know that $W_{m,k}(\boldsymbol{x})$ is monotone nondecreasing in k and

$$\sup_{\pi\in\Pi} V_m(\boldsymbol{x}, \pi) = V_m^*(\boldsymbol{x}) = \lim_{k\to\infty} W_{m,k}(\boldsymbol{x}) = \sup_k W_{m,k}(\boldsymbol{x}).$$

Taking the supremum over all m and interchanging the order of supremum, we obtain

$$\sup_{\pi\in\Pi} \sup_m V_m(\boldsymbol{x}, \pi) = \sup_k \sup_m W_{m,k}(\boldsymbol{x}).$$

To verify (3.6), it remains to verify that

$$V(\boldsymbol{x}, \pi) = \sup_m V_m(\boldsymbol{x}, \pi), \qquad W_k(\boldsymbol{x}) = \sup_m W_{m,k}(\boldsymbol{x}). \tag{3.7}$$

The first statement in (3.7) follows from the monotonous convergence theorem, observing that $\sup_m r_m = r$. The second follows by induction on k. For $k = 0$ the result is trivial. Assuming that the equality holds for $k = l - 1$ observe that

$$\sup_m W_{m,l}(\boldsymbol{x}) = \sup_{\boldsymbol{a}} \biggl(r(\boldsymbol{x}, \boldsymbol{a}) + \sup_m \sum_{\boldsymbol{y}\in\mathcal{X}} W_{m,l-1}(\boldsymbol{y}) q(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{a}) \biggr).$$

The supremum on the right-hand side can be substituted into the sum since $W_{m,k}$ is nondecreasing with m. Using the inductive hypothesis the proof is completed. Up to this point we have proved (3.6). To see that this limit is indeed the smallest solution of (3.5), we refer to the proof of [1, Theorem 3] since the hypothesis of r being bounded is not used to prove this fact.

Introducing the notation

$$V_r(\tau, n) = \sum_{k\in\mathcal{I}} \frac{f(n,k)}{6^n} V(\tau + s(k), k, n - d(k) + 5\,\mathbf{1}_{\{n=d(k)\}}), \tag{3.8}$$

the DPEs of our model can be written as

$$V(\boldsymbol{x}) = \begin{cases} \tau_{\boldsymbol{x}} \vee V_r(\tau_{\boldsymbol{x}}, n_{\boldsymbol{x}}) \vee \max\{V(\tau_{\boldsymbol{y}}, i_{\boldsymbol{y}}, n_{\boldsymbol{y}}) : \boldsymbol{y} \in \mathcal{X}, \boldsymbol{y} \prec \boldsymbol{x}\} & \text{if } \boldsymbol{x} \notin \{\iota, \Delta\}, \\ V_r(0, 5) & \text{if } \boldsymbol{x} = \iota, \\ 0 & \text{if } \boldsymbol{x} = \Delta, \end{cases} \tag{3.9}$$

where the maximum in the first case is assumed to be 0 if the set is void. It is useful to observe that the mentioned maximum (let us denote it by M) coincides with

$$M' = \max\{\tau_{\boldsymbol{y}} \vee V_{\boldsymbol{r}}(\tau_{\boldsymbol{y}}, n_{\boldsymbol{y}}) \colon \boldsymbol{y} \in \mathcal{X}, \boldsymbol{y} \prec \boldsymbol{x}\}.$$

It is clear that $M' \leq M$. To see that the equality holds we need to observe that if an optimal action from state $\boldsymbol{x}$ is $\boldsymbol{m}_{\text{to}\,\boldsymbol{x}_1}$, and an optimal action from $\boldsymbol{x}_1$ is $\boldsymbol{m}_{\text{to}\,\boldsymbol{x}_2}$ then $\boldsymbol{x}_2 \prec \boldsymbol{x}$ and from the state $\boldsymbol{x}$ the action $\boldsymbol{m}_{\text{to}\,\boldsymbol{x}_2}$ is as good as $\boldsymbol{m}_{\text{to}\,\boldsymbol{x}_1}$. As the number of moves in a row that can be performed is bounded (by the amount of scoring dice minus 1), the previous situation finishes (after a bounded number of steps) in a state $\boldsymbol{y}$ in which the optimal action is $\boldsymbol{s}$ or $\boldsymbol{r}$. For that $\boldsymbol{y}$, we have $M = V(\tau_{\boldsymbol{y}}, i_{\boldsymbol{y}}, n_{\boldsymbol{y}}) = \tau_{\boldsymbol{y}} \vee V_{\boldsymbol{r}}(\tau_{\boldsymbol{y}}, n_{\boldsymbol{y}}) = M'$.

4. Main result

We state our main result, whose proof can be found at the end of this section.

Theorem 4.1. *There exists a pure and stationary optimal strategy π^* for the considered game. Calling A the rule of decision of π^* and V^* the value of the game, it follows that if $\tau_{\boldsymbol{x}} \geq 56$ then $V^*(\boldsymbol{x}) = \tau$ and $A(\boldsymbol{x}) = \boldsymbol{s}$; while for states $\boldsymbol{x}$ with $\tau_{\boldsymbol{x}} < 56$, $V^*(\boldsymbol{x})$ and $A(\boldsymbol{x})$ are exactly obtained by our algorithm. About the algorithm, for states with two available actions $V(\boldsymbol{x})$ is already initialized with its final value $V^*(\boldsymbol{x})$, while if moves are available $V(\boldsymbol{x})$ may need as many updates as states $\boldsymbol{y} \colon \boldsymbol{y} \prec \boldsymbol{x}$ are available. The value of the game from its initial state is* $V^*(\imath) = 5.872019$. *In Tables 5 and 6 we summarize the outcome of the algorithm. Actions $\boldsymbol{m}_{11}$, $\boldsymbol{m}_{551}$, $\boldsymbol{m}_{511}$, and $\boldsymbol{m}_{t_k} \colon k = 1, \dots, 6$ are never optimal.*

To construct a finite algorithm, given the fact that the number of states is infinite, we need to reduce the number of relevant states to a finite amount. For this aim, a first step in the algorithm design is the determination of a set of states in which it is optimal to stop immediately, i.e. states such that $V^*(\tau, i, n) = \tau$. The search is based on the fact that for large values of τ we expect it not to be optimal to risk the accumulated score, so the optimal action would be to stop and obtain $V^*(\tau, i, n) = \tau$. We mimic the search of the solution to optimal stopping problems for continuous-time processes with positive jumps that verify integral equations; see, for example, [6]). Assuming that there exists a critical threshold τ^* such that

$$\tau^* = \min\{\tau \colon V^*(\tau, i, n) = \tau \text{ for all } (i, n)\},$$

and based on the nature of the game, it is natural to suppose that the minimum is attained in a state with $n = 5$. We then compute

$$\tau^*(5) = \min\left\{\tau \in \mathbb{Z}^+ \colon \tau \geq \sum_{k \in \mathcal{I}} \frac{f(5, k)}{6^5}(\tau + s(k))\right\},$$

or, equivalently (using $\sum_{k \in \mathcal{I}} f(5, k) + f(5, \mathbf{0}) = 6^5$),

$$\tau^*(5) = \min\left\{\tau \in \mathbb{Z}^+ \colon \tau \geq \frac{\sum_{k \in \mathcal{I}} f(5, k) s(k)}{f(5, \mathbf{0})}\right\},$$

obtaining $\tau^*(5) = 56$. The following lemma includes the intuitive fact that for $\tau \geq 56$ the optimal action is to stop.

TABLE 5: Value function and optimal strategy for $n = 2, \dots, 4$. For $n = 5$ see Table 6. Cells with '–' are unreachable. The missing values mean that the optimal action is s, so the value coincides with τ (this is the case for the omitted rows $\tau \geq 19$). Column $n = 1$ (and also $n = 2$ for 'other i') is omitted, since the optimal action is always s. The optimal action for the present values on the table depends only on the column and is indicated in the last row. Note the following observations. All present values are greater than τ, otherwise the optimal action would be s. Values in columns for which the optimal action is a move appear also in other columns (the destination of the move). These columns could be completed 'by hand'.

τ	$n = 2$			$n = 3$			$n = 4$
	$[2, 1, 0]$	$[1, 2, 0]$	$[2, 0, 0]$ $[1, 1, 0]$	$[2, 0, 0]$ $[1, 1, 0]$	$[0, 2, 0]$	other i	all i
0	–	–	–	–	–	–	–
1	–	–	–	–	–	–	4.338
2	–	–	–	4.338	–	3.447	5.021
3	–	–	3.447	5.021	–	4.123	5.743
4	5.021	–	4.123	5.743	5.021	4.837	6.476
5	5.743	5.021		6.476	5.743	5.552	7.212
6	6.476			7.212	6.476	6.266	8.001
7	7.212			8.001	7.212		8.840
8	8.001			8.840	8.001		9.679
9				9.679			10.517
10				10.517			11.357
11				11.357			12.196
12				12.196			13.035
13				13.035			13.875
14							14.715
15							15.554
16							16.394
17							17.234
18							18.073
a	m_{55}	m_{51}	r	m_5	m_1	r	r

Lemma 4.1. *The following facts hold for a state* x:

(a) *if* $\tau_x \geq 56$ *then* $V^*(x) = \tau_x$;

(b) *if* $\tau_x < 56$ *and* $n_x = 5$ *then* $V^*(x) > \tau_x$;

(c) *if* $\tau_x < 56$ *then* $V^*(x) \leq 56$.

Proof. The proof is based on the fact that $V^*(x)$ is the limit of the nondecreasing sequence $W_k(x)$; see (3.6). Let us prove Lemmas 4.1(a) and 4.2(c). First, observe that for any x, we have $W_1(x) = \tau_x$. To see this, remember that $W_1 = \mathrm{L}\, W_0$, which is (see (3.4))

$$W_1(x) = \max_{a \in \mathcal{A}(x)} \mathrm{L}_a\, W_0(x).$$

The maximum is attained with the action $a = s$, which is the only action such that $\mathrm{L}_a\, W_0(x) > 0$; in fact, $\mathrm{L}_s\, W_0(x) = r(x, s) = \tau_x$. We continue by proving by induction that for any $k \in \mathbb{Z}^+$, $W_k(x) < 56$, if $\tau_x < 56$, and $W_k(x) = \tau_x$, if $\tau_x \geq 56$: the $k = 1$ case is already proved. Assume that the result holds in the $k < m$ case and let us see if it remains valid for m.

TABLE 6: Value function and optimal strategy for $n = 5$. See the caption of Table 5.

τ	$n=5$ all i	τ	$n=5$ all i	τ	$n=5$ all i	τ	$n=5$ all i
0	5.8721	14	17.413	28	30.181	42	43.075
1	6.6070	15	18.291	29	31.102	43	43.997
2	7.3550	16	19.170	30	32.022	44	44.919
3	8.1070	17	20.060	31	32.943	45	45.840
4	8.8830	18	20.972	32	33.864	46	46.762
5	9.7130	19	21.893	33	34.785	47	47.684
6	10.5530	20	22.814	34	35.706	48	48.607
7	11.3940	21	23.734	35	36.627	49	49.529
8	12.2350	22	24.655	36	37.548	50	50.452
9	13.0770	23	25.576	37	38.469	51	51.375
10	13.9180	24	26.497	38	39.390	52	52.298
11	14.7790	25	27.418	39	40.312	53	53.221
12	15.6550	26	28.339	40	41.233	54	54.144
13	16.5340	27	29.260	41	42.154	55	55.066
a	***r***	***a***	***r***	***a***	***r***	***a***	***r***

We have

$$W_m(\boldsymbol{x}) = \max_{a\in\mathcal{A}(\boldsymbol{x})} \{\mathrm{L}_a W_{m-1}(\boldsymbol{x})\}.$$

Let us prove that in the case $\tau_{\boldsymbol{x}} \geq 56$ the maximum is attained with the action $\boldsymbol{s}$ (giving $W_m(\boldsymbol{x}) = \tau_{\boldsymbol{x}}$): an action $\boldsymbol{m}_{\text{to } \boldsymbol{y}}$, if possible, would imply a transition with full probability to a state $\boldsymbol{y}$: $\boldsymbol{y} \prec \boldsymbol{x}$. In this case $\tau_{\boldsymbol{y}} = \tau_{\boldsymbol{x}} - s(i_{\boldsymbol{x}}) + s(i_{\boldsymbol{y}}) < \tau_{\boldsymbol{x}}$. Then we have $\mathrm{L}_{\boldsymbol{m}_{\text{to } \boldsymbol{y}}} W_{m-1}(\boldsymbol{x}) = W_{m-1}(\boldsymbol{y})$, which is less than or equal to $\tau_{\boldsymbol{x}}$ by induction hypothesis. On the other hand, for the action $\boldsymbol{r}$, we have (using the notation $\boldsymbol{x} = (\tau, i, n)$)

$$\begin{aligned}
\mathrm{L}_{\boldsymbol{r}} W_{m-1}(\boldsymbol{x}) &= \sum_{k\in\mathcal{I}(n)} \frac{f(n,k)}{6^n} W_{m-1}(\tau + s(k), k, n - d(k) + 5\,\mathbf{1}_{\{n=d(k)\}}) \\
&= \sum_{k\in\mathcal{I}(n)} \frac{f(n,k)}{6^n}(\tau + s(k)) \\
&= \tau + \sum_{k\in\mathcal{I}(n)} \frac{f(n,k)}{6^n} s(k) - \tau\frac{f(n,\mathbf{0})}{6^n} \\
&< \tau,
\end{aligned}$$

since, for $\tau \geq 56$,

$$\tau > \frac{\sum_{k\in\mathcal{I}(n)} f(n,k)s(k)}{f(n,\mathbf{0})},$$

not only for $n = 5$ but also for $n = 1, 2, 3, 4$ (this fact can be verified directly). If $\boldsymbol{x} = (\tau, i, n)$ with $\tau < 56$ then $\mathrm{L}_a W_{m-1}(\boldsymbol{x}) < 56$ for all possible actions. For actions $\boldsymbol{s}$ and $\boldsymbol{m}$ this fact is a direct consequence of the induction hypothesis, while for the action $\boldsymbol{r}$, we have

$$\begin{aligned}
\mathrm{L}_{\boldsymbol{r}} W_{m-1}(\boldsymbol{x}) &= \sum_{k\in\mathcal{I}(n)} \frac{f(n,k)}{6^n} W_{m-1}(\tau + s(k), k, n - d(k) + 5\,\mathbf{1}_{\{n=d(k)\}}) \\
&< \sum_{k\in\mathcal{I}(n)} \frac{f(n,k)}{6^n}(56 + s(k)) \\
&= \mathrm{L}_{\boldsymbol{r}} W_{m-1}(56, i, n) \\
&\le 56.
\end{aligned}$$

Now that we know the result for all $k \ge 1$ we can take limit to obtain Lemmas 4.1(a) and 4.1(c).

To prove Lemma 4.1(b) consider $\boldsymbol{x} = (\tau, i, 5)$, with $\tau < \tau^*(5) = 56$. We already observed (in general) that $W_1(\boldsymbol{x}) = \tau$. By the very definition of $\tau^*(5)$ we know that $\mathrm{L}_{\boldsymbol{r}} W_1(\boldsymbol{x}) > \tau$. So $W_2(\boldsymbol{x}) > \tau$. Finally, we can use the monotonicity of W_k and take the limit, to conclude the proof. □

Once we know $V^*(\boldsymbol{x}) = \tau_{\boldsymbol{x}}$ in the case $\tau_{\boldsymbol{x}} \ge 56$, we can use the fact that V^* is a solution of (3.9) to compute $V^*(\boldsymbol{x})$ for the remaining states. In Algorithm 4.1 we use a backward induction to make this computation. All states $\boldsymbol{x}$: $\tau_{\boldsymbol{x}} < 56$ are considered once. The 'currently considered state' $\boldsymbol{x}$ is assigned in line 3 in the $n_{\boldsymbol{x}} = 5$ case and in line 7 for the rest of the states. The initial state ι is considered separately in line 15. The following lemma proves that the values obtained by the algorithm complete a solution of (3.9). We use the notation $V_{\boldsymbol{r}}^*(\tau, n)$ with the same meaning as in (3.8) in the particular case in which the function V there considered is the value function V^*.

Algorithm 4.1.
```
1.  for $\tau = 55$ downto 1
2.    for $i \in \mathcal{I}$
3.      $\boldsymbol{x} \leftarrow (\tau, i, 5)$ ($\boldsymbol{x}$ is the currently considered state)
4.      $V(\boldsymbol{x}) \leftarrow V_{\boldsymbol{r}}(\tau, 5)$, $A(\boldsymbol{x}) \leftarrow \boldsymbol{r}$
5.    for $n = 4$ downto 1
6.      for $i \in \mathcal{I}(5 - n)$
7.        $\boldsymbol{x} \leftarrow (\tau, i, n)$ ($\boldsymbol{x}$ is the currently considered state)
8.        if $V_{\boldsymbol{r}}(\tau, n) \le \tau$ then
9.          $V(\boldsymbol{x}) \leftarrow \tau$, $A(\boldsymbol{x}) \leftarrow \boldsymbol{s}$
10.       else
11.         $V(\boldsymbol{x}) \leftarrow V_{\boldsymbol{r}}(\tau, n)$, $A(\boldsymbol{x}) \leftarrow \boldsymbol{r}$
12.         for states $\boldsymbol{y}$: $\boldsymbol{x} \prec \boldsymbol{y}$, $\tau_{\boldsymbol{y}} \le 55$
13.           if $V(\boldsymbol{y}) < V(\boldsymbol{x})$ then
14.             $V(\boldsymbol{y}) \leftarrow V(\boldsymbol{x})$, $A(\boldsymbol{y}) \leftarrow \boldsymbol{m}_{\text{to}\,\boldsymbol{x}}$
15. $V(\iota) \leftarrow V_{\boldsymbol{r}}(0, 5)$
```

Note that $V_r(\tau, n)$ should computed according to (3.8) and taking into account the boundary condition $V(\tau, i, n) = \tau$ for $\tau \geq 56$.

Lemma 4.2. *When the state x is considered by our algorithm.*

(a) *$V(x)$ is initialized with the maximum between τ_x and $V_r^*(\tau_x, n_x)$.*

(b) *The value of $V(x)$ is only updated if a move $(m_{to\ y})$ is available and gives more expected reward. In this case the update is performed when y is considered.*

(c) *$V(\imath)$ is initialized as $V^*(\imath)$ in line 15.*

Proof. Let us start by analysing the algorithm and proving the lemma under the assumption $V_r(\tau, n) = V_r^*(\tau, n)$ (which is proved below). In line 4 the value $V(x)$ is initialized for $x\colon n_x = 5$. It gets $V_r^*(\tau_x, 5)$ which is $\tau_x \vee V_r^*(\tau_x, n_x)$ (see the proof of Lemma 4.1(b)). In the $1 \leq n_x < 5$ case, the initialization of $V(x)$ is performed either in line 9 or 11 with the maximum between τ_x and $V_r^*(\tau_x, n_x)$. This completes the proof of Lemma 4.2(a). To prove Lemma 4.2(b) observe that the value $V(x)$ is not updated again during the consideration of state x itself. The only other lines in which V is modified are 14 and 15. The latter is where the initial state is considered. In line 14 the value of $V(y)$ is updated if $x \prec y$ and $\tau_x \vee V_r^*(\tau_x, n_x) > V(y)$, which is to say, if a move to x is available from y and gives a better expected reward (considering as available actions at x just s and r; the comment after (3.9) shows that this is enough). Observe that this possible update is considered just for states $x\colon 1 \leq n_x < 5$ if $V_r^*(\tau_x, n_x) > \tau_x$. In the $n_x = 5$ case the verification would be superfluous, since there is no $y\colon x \prec y$, while in the $\tau_x \geq V_r^*(\tau_x, n_x)$ case it would not make sense to update $V(y)$ since $V(y) \geq \tau_y > \tau_x$. Interchanging the roles of x and y, Lemma 4.2(b) is proved. Lemma 4.2(c) is trivial once we know that $V(x) = V^*(x)$ for the rest of the states.

It remains to prove that $V_r(\tau, n)$ computed in the algorithm coincides with $V_r^*(\tau, n)$. In the first iteration (of the for in line 1), in which $\tau = 55$, the assertion is valid, since $V_r(\tau, n)$ depends only on values that are known by the boundary condition and no update of values $V(y)$ for y with $\tau_y > 55$ is foreseen in the algorithm, which means that for $\tau = 55$ the values of $V_r(\tau, n)$ are never updated in later iterations. Assume that the result is valid in the first $k - 1$ iterations ($2 \leq k \leq 55$) and let us prove that it remains valid in the kth iteration (in which $\tau = 56 - k$). We need to prove that $V_r(\tau, n)$ depends on values of $V(y)$ that are already final ($V(y) = V^*(y)$). Taking into account the induction hypothesis, and the algorithm operation, the problem is reduced to proving that $V(y)$ is not updated (in line 14) when a state $z\colon z \prec y$ is considered. To prove that this is not the case, observe that if y is involved in the computation of $V_r(\tau_x, n_x)$ then, according to (3.8), we have $\tau_y = \tau_x + s(i_y)$. On the other hand, if $z \prec y$ then $\tau_z = \tau_y - s(i_y) + s(i_z)$ with $0 < s(i_z) < s(i_y)$, which renders $\tau_z = \tau_x + s(i_z) > \tau_x$. As the algorithm considers first states with greater τ, the fact that $\tau_z > \tau_x$ for every $z \prec y$ implies that $V(y)$ is not updated after x is considered. This concludes the proof of the lemma. □

Proof of Theorem 4.1. The claim about the value function has been proved in lemmas 4.1 and 4.2. It remains to prove that the strategy π^*, with rule of decision A attains the value V^* already found. For this purpose, we show that [9, Theorem 7.1.7(b)] can be applied. By construction of the algorithm, the optimal action $A(x)$ for states $x = (\tau, i, n)$ with $\tau \leq 55$ is such that

$$\mathrm{L}_{A(x)}\, V^*(x) = \mathrm{L}\, V^*(x) = V^*(x).$$

The same equality holds for states $\boldsymbol{x} = (\tau, i, n)$ with $\tau \geq 56$ and $A(\boldsymbol{x}) = \boldsymbol{s}$. The condition is also fulfilled for $\boldsymbol{x} = \Delta$ with $A(\Delta) = \boldsymbol{w}$, taking into account that $V^*(\Delta) = 0$. This means that the pure stationary strategy π^* is what in [9] is called *conserving*. It just remains to verify that the strategy is *equalizing*, which means that

$$\limsup_{N\to\infty} P_A^N V^*(\boldsymbol{x}) = 0 \quad \text{for all } \boldsymbol{x} \in \mathcal{X},$$

where the operator P_A acts on V^* by $P_A V^*(\boldsymbol{x}) = \mathrm{L}_{A(\boldsymbol{x})} V^*(\boldsymbol{x})$ and $P_A^N V^*$ refers to the application N times of the same operator. This condition is trivially fulfilled in our case, since no matter what the initial state is, when the player follows rule A, after at most 57 dice rolls the system will be in the final state Δ. This is due to the fact that in any roll a positive score must be accumulated, or the game will go to the final state. Assuming the worst case, that the player starts for ι and always gets just one chip, it will take at most 56 dice rolls to arrive to one of the states in which the optimal action is to stop. To compute a bound of the number of transitions of the Markov control problem that are necessary to arrive to the final state we could think that after any roll the player makes as many moves as possible before rolling again. The maximum number of moves that can be made in a row is strictly less than the number of scoring combinations, which is at most five. This means that $P_A^N V^*(\boldsymbol{x}) = 0$ for all $N \geq 56 \times 4 + 1$ and for all states $\boldsymbol{x} \in \mathcal{X}$. Taking the lim sup, we complete the proof. □

5. Other related games

5.1. Ten Thousand with restricted actions

A natural game related with the one studied in this paper is the *stop or roll* solitaire Ten Thousand game. It is essentially the same game but without the possibility of taking action $\boldsymbol{m}$ (the player can only take actions $\boldsymbol{s}$ and $\boldsymbol{r}$). It has value $V_{S/R} = 5.576\,326$ and the algorithm to solve it is Algorithm 4.1 without lines 12, 13, and 14.

Other games that require restricting the total number of possible actions of the original game can also be solved. The values (from the initial state) of some of these games are presented in Table 7.

5.2. The solitaire Pig game

The solitaire Pig game is similar, played with only one die. The only actions the player has are to roll again or to stop (for details see [10]). The state can be modelled with just τ, the accumulated score. The problem is, in this case, an optimal stopping problem. The corresponding DPEs are

$$V(\tau) = \max\{\tau, \tfrac{1}{6}(V(\tau+2) + V(\tau+3) + V(\tau+4) + V(\tau+5) + V(\tau+6))\}. \quad (5.1)$$

The critical threshold is $\tau^* = 20$. Computing backwards, we obtain $V^*(0) = 8.14$. In this example, we find it instructive to observe that the difference equation

$$V(\tau) = \tfrac{1}{6}(V(\tau+2) + V(\tau+3) + V(\tau+4) + V(\tau+5) + V(\tau+6))$$

has an infinite number of solutions, under the condition that $V(\tau) \geq \tau$, for instance $V(\tau) = V(0)1.0461^\tau$ for all large enough $V(0)$. The value function is the minimal solution of (5.1), which satisfies $V(\tau) = \tau$ for all $\tau \geq \tau^*$.

TABLE 7: Values of restricted actions games.

Number of actions	Possible actions[1]	Value of the game
2	s, r	5.576 326 278 2
3	s, r, m_5	5.801 218 003 7
4	s, r, m_5, m_1	5.815 334 063 9
5	s, r, m_5, m_1, m_{55}	5.870 748 432 6
6	$s, r, m_5, m_1, m_{55}, m_{51}$	5.872 018 918 5
15	all	5.872 018 918 5

6. Conclusions

The theory of MCPs is a powerful tool to analyse dice games. Nevertheless, although in general it is possible to write the DPEs of the game, in many concrete situations the number of states and actions make it very difficult to effectively solve the game. In these situations, when possible, it is necessary to take into account the particular characteristics of the game in order to solve the problem. Usually, these equations are solved by iteration. In this paper we obtain a very simple and exact algorithm to solve the solitaire Ten Thousand game. It is interesting to note that the value function is characterized as the smallest solution to the DPEs, in a framework where the uniqueness of the solution to this equation is not assured. The same idea is also used to solve some related simpler games. Whether a similar type of algorithm can be used in the competitive Ten Thousand game remains an open question.

Acknowledgements

The authors are indebted to an anonymous referee for providing insightful comments and an extremely careful reading of the different versions of this paper; this supportive work contributed significantly to its final version.

The first author is part of the SRI-UQ Center for Uncertainty Quantification, KAUST.

References

[1] BLACKWELL, D. (1967). Positive dynamic programming. In *Proc. Fifth Berkeley Symp. Math. Statist. Prob.* (Berkeley, CA, 1965/66), Vol. I, *Statistics*, University of California Press, pp. 415–418.

[2] FILAR, J. AND VRIEZE, K. (1997). *Competitive Markov Decision Processes.* Springer, New York.

[3] HAIGH, J. AND ROTERS, M. (2000). Optimal strategy in a dice game. *J. Appl. Prob.* **37,** 1110–1116.

[4] HALD, A. (1990). *A History of Probability and Statistics and Their Applications Before 1750.* John Wiley, New York.

[5] HERNÁNDEZ-LERMA, O., CARRASCO, G. AND PÉREZ-HERNÁNDEZ, R. (1999). Markov control processes with the expected total cost criterion: optimality, stability, and transient models. *Acta Appl. Math.* **59,** 229–269.

[6] MORDECKI, E. (1999). Optimal stopping for a diffusion with jumps. *Finance Stoch.* **3,** 227–236.

[7] NELLER, T. W. AND PRESSER, C. G. M. (2004). Optimal play of the dice game pig. *The UMAP J.* **25,** 25–47.

[8] PLISKA, S. R. (1978). On the transient case for Markov decision chains with general state spaces. In *Dynamic Programming and Its Applications* (Proc. Conf., University of British Columbia, Vancouver, 1977), Academic Press, New York, pp. 335–349.

[9] PUTERMAN, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley, New York.

[10] ROTERS, M. (1998). Optimal stopping in a dice game. *J. Appl. Prob.* **35,** 229–235.

[11] TIJMS, H. (2007). Dice games and stochastic dynamic programming. *Morfismos* **11,** 1–14.

[12] TIJMS, H. AND VAN DER WAL, J. (2006). A real-world stochastic two-person game. *Prob. Eng. Inf. Sci.* **20,** 599–608.