Markov chain applications in the slot machine industry

Noelia Oses
NOF Consulting (http://www.nofcon.com/)
87 Windermere Rd, Lancaster LA1 3EZ, United Kingdom
Noelia@nofcon.com; Phone: +44(0)152463073; Mobile: +44(0)7811181721

Abstract
One of the important sectors within gambling is that of gaming machines. This industry used to favour approaches based on long-run simulations or complete enumerations of all possible outcomes but has now adopted mathematical modelling as the preferred method for calculating the probability distribution of the prizes and deriving the percentage return of the games. The latter is the percentage of the collected money the game will return to the players in the long term. In this context Markov chains are a very important tool for this industry. This paper analyses the different types of slot features modelled as Markov chains, discusses their particular characteristics, illustrates the type of problems that one might encounter in practice and provides suggestions for developing implementations of the models. The cases studied are games that have already been released and are commercially available.

Keywords: gaming; Markov processes; OR in gambling; slot machines

Introduction

Gambling has been popular since ancient times (the first games of chance date from as far back as c. 3500 B.C. (David, 1962)) and it is now more accessible than ever due to the popularity explosion of online casinos and the introduction of mobile gambling ([1]). One of the important sectors within gambling is that of the gaming machines. A gaming machine is any machine that, in exchange of money or tokens, offers a game of chance with prizes ([2]). Most gaming machines are of the reel-based type, also known as fruit or slot machines. Although slots have a variety of features (Oses and Freeman, 2006), the main one is the spin of the reels where the player wins by lining up symbols according to predetermined patterns. Additional features in the game are generally known as bonus games. Gambling makes a significant contribution to the UK economy with an estimated expenditure of £8,875 million (0.8% of UK GDP (Gambling Act, 2005)), of which £1.74 billion is made in costumer losses playing slot machines (Gaming Board, 2005).

Contrary to the approaches favoured in the old days, based on long-run simulations or complete enumerations of all possible outcomes (Freeman, 1998), nowadays mathematical modelling is preferred. The objective of the mathematical model is to calculate the probability distribution of the prizes and the percentage return is the main result that can be derived from this. The "percentage return" of a slot game is the percentage of the collected money the game will return to the players in the long term. From the prize distribution we can also derive hit rates, the volatility of the game and confidence intervals for the percentage return. Additionally, it allows performing risk analysis and gambler's ruin analysis. Manufactures, in order to license a game, must provide a model of the game and ensure that its percentage return is above the lower bound set by UK law ([3]). Slot games cannot be licensed without a model and cannot be sold without a licence. It is also important to set the percentage return in the right range because, if it is too low, players might decide against playing the game. If it is too high, on the

other hand, the game provider might not obtain the desired revenue. The model is also necessary to have control over the hit rates and the volatility. If prizes do not occur often enough players might get bored and stop playing the game.

Slot machine games nowadays often share the same game features, or a combination of them. Detailed descriptions of these features with corresponding modelling solutions are presented in (Oses and Freeman, 2006). This paper analyses the types of slot game features or bonuses that can be modelled as Markov chains, based on the industry experience of the author. Whereas the previous paper (Oses and Freeman, 2006) was a light introduction to the modelling of slot games, provided an overview of all of the most popular game features and roughly suggested modelling options, this paper focuses only on those features that can be modelled with Markov chains and goes much deeper into the modelling details. The following sections discuss the particular characteristics of these games and how they can be exploited when implementing solution algorithms. Selected case studies will illustrate these points.

Particular characteristics of Markov chain models in slots

Markov processes, and Markov chains in particular, are of interest in many different fields. Their use has been reported in manufacturing (Tan and Karabati, 2000; Feiring et al., 1998), marketing (Ching et al., 2004), operating system modelling (Stewart, 1991), genetics (Tan, 2002), etc. Discrete time Markov chain modelling is a very important tool for the slot machine industry, too.

The bonus games susceptible of being modelled as Markov chains usually share a set of particular characteristics. The game usually starts from a specific configuration or setting i.e. the initial distribution is fixed. This is, presumably, because these games are implemented as computer programs and the computer needs to be told, deterministically or probabilistically, where to start the game. Furthermore, the process usually starts from a specific, fixed state that represents the beginning of the bonus game. Additionally, the state transitions in these games are usually determined by some simple mechanism like the throw of a die or the spin of a reel. This means that, quite often, the number of possible transitions is quite small compared to the number of states and, therefore, the transition matrix is sparse. Last, states have rewards associated to them and the industry is exclusively interested in the expected prize value of the bonus game, this being the only attribute of the game necessary for the calculation of the percentage return.

Among the game features that can be modelled with Markov chains, the author has come across two types. Bonus games that start and end every time they are played and features that are on going. The following sections analyse each of these in depth.

On-going bonus games

These are bonus games that, when they are accessed, they are in the same state as they were when they were left the last time they were played. The state or configuration of the game for the very first time the feature is accessed (e.g. the first time after the machine is switched on) is specified in the game manual. In other words, the initial distribution is fixed and this will usually specify one, fixed starting state. This type of game does not have an end; it will keep jumping from one state to the next until the machine is reset (e.g. when the machine is switched off due to malfunction or another reason). When the state transition only depends on the current state and not the past states, this feature can be modelled as a Markov chain.

Percentage returns are always calculated for the long-term. In this case, this means that it is necessary to calculate the expected prize of this feature once the machine has been running long enough to overcome any effects of the starting state and it has reached its normal running mode

or steady state. Therefore, the problem becomes one of finding the steady state distribution of the Markov chain. Typically in this type of game, the number of states is finite, they are all recurrent and the chain is irreducible, therefore the steady state distribution exists.

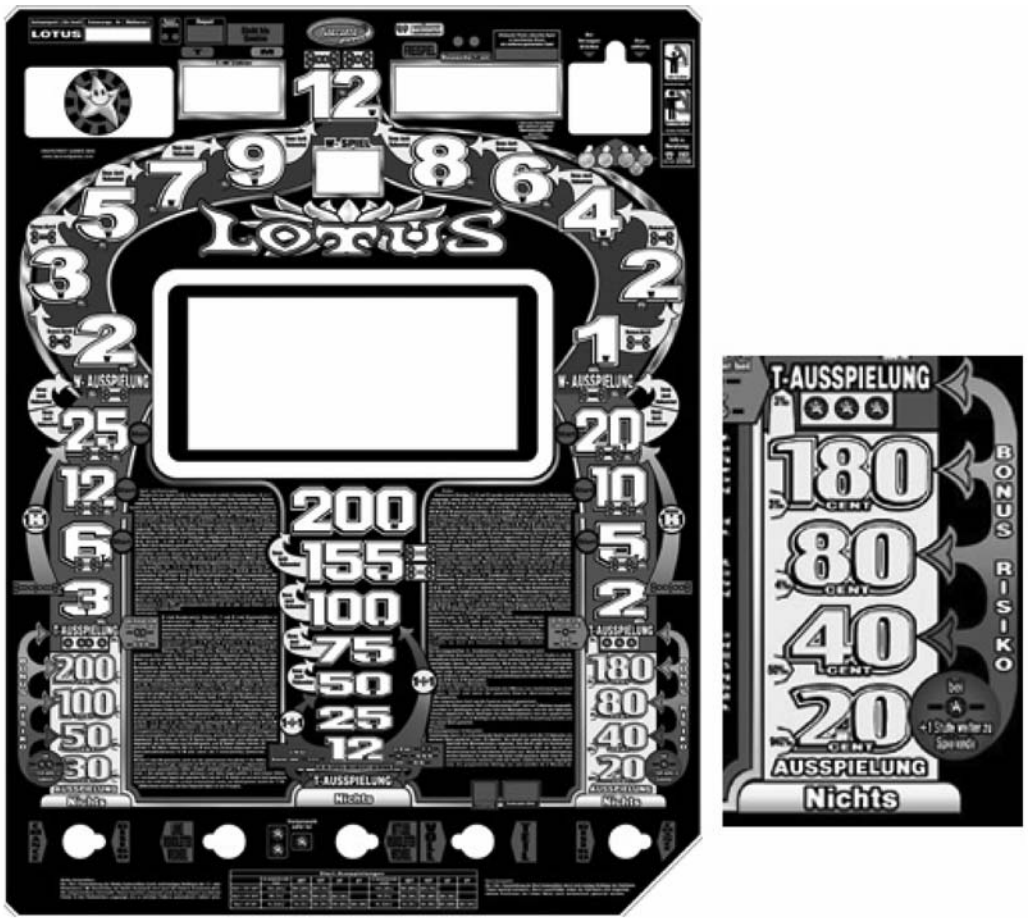
Due to the on-going nature of these bonus games, they are usually quite simple so as to not obstruct the clarity of the overall game. This simplicity means these features are easy to model with a Markov chain that has a small state space, as the Bonus Risiko analysed in the next section shows.

Bonus Risiko (Lotus)

The Lotus game has been released in 2006 by Barcrest Games and it is being marketed in Germany ([4]). This game has a bonus feature called Bonus Risiko (Figure 1). This feature is triggered via an award obtained in the main feature, the spin of the reels. If the award occurs, the player may gamble from the 20 cents position to the position or prize indicated with the lit arrow or down to lose. The probability of winning the gamble is always 0.5. At the end of the gamble the arrow will move up one position (regardless of the gamble result or even if the player collects) to the next level (unless at the top). The arrow will remain at the top until a successful gamble result is achieved before resetting back down to the bottom position.

There are 4 states in this chain: State 1 is when the arrow pointing to 40 is lit; State 2 is when the arrow pointing to 80 is lit; State 3 is when the arrow pointing to 180 is lit; and, State 4 is when the arrow pointing to T-Ausspielung is lit. If the top arrow is lit and the player wins the gamble, he will go on to play the T-Ausspielung feature.

Figure 1 Lotus (left) and Bonus Risiko (right)



T-Ausspielung accesses another feature, the expected value of which is calculated independently and which will be the reward associated to winning this gamble in this model. For the purpose of this paper, we will assume that the player does not have to make any choices. If this bonus is triggered, we will assume the player has to gamble. After the gamble, the player leaves this feature with the prize won, if he was successful in the gamble, or with no prize, otherwise. In this model, he does not have a choice of staying in this feature gambling further. In order to calculate the expected value, it is necessary to calculate the steady state probability distribution for the arrows. If we did not make the above assumptions and the player had choices then, in addition to calculating the steady state distribution, we would have to calculate the optimal strategy, but that is beyond the scope of this paper.

The transition matrix is $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0.5 & 0 & 0 & 0.5 \end{pmatrix}$ and the initial distribution is

(1 0 0 0). The stationary distribution can be calculated by inputting these into a Markov chain package (e.g. MARCA (Stewart, 1991)) or a general mathematics package (e.g. MATLAB (Hunt et al., 2001)). The result for this game is (0.2 0.2 0.2 0.4). Alternatively, since this type of applications are generally simple like the present case, the stage distributions can also be calculated directly on a spreadsheet until they converge to a limiting distribution (see Table 1 where p is the probability of winning the gamble, 0.5, and q is the probability of losing it, 1-p). This is quite easy and quick to do as the formulas need only be entered in the first stage, and then dragged right onto all other stages. Once we have the limiting distribution it is easy to calculate the expected prize value of the feature.

Start-and-end bonus games

These are bonus games that start and end every time they are played. Every time the player gets an outcome of the spinning reels that takes him to play this feature, it starts from the same starting position. The game progresses according to some random mechanism, with no input required from the player, until the event that marks the end of the game occurs. The prize might be determined by the path the process follows or by events that happen during the game.

The industry, as mentioned before, is only interested in the long-term percentage return i.e. in the expected prize the player will get. In this case, it makes no sense to think of the steady state distribution because the maximum number of stages the feature will go on for is limited and every time the game is played it will start from the initial stage. If the process does not have a natural end within a limited number of stages then the rules of the game will set a limit to, say, N throws of the die or spins of a reel. This limit will restrict the maximum number of stages. Depending on how the game is modelled the maximum number of stages might coincide, or not, with N. Therefore, the objective now is to calculate the transient distributions to determine the expected prize gained from each stage and, adding these up, the expected prize of the feature.

There is a fixed initial distribution, which usually means there is a fixed starting state, and a very simple and general mechanism for state transitions, i.e. very few possible transitions for each state. These two factors mean that in the first few stages the set of states for which the transient probability (the probability of being in that state at that stage) is non-zero is small. Finally, the industry's interest is exclusively focused in calculating the expected prize. These particularities can be exploited when implementing solutions by focusing entirely on

Table 1 Spreadsheet calculation of steady state

	B	C	D	E	F	G	H	I	J	
2	State \ Stage	0	1	2	3	4	5	6	…	183
3	Auss.	0	=C4+q*C3	=D4+q*D3	=E4+q*E3	=F4+q*F3	=G4+q*G3	=H4+q*H3	…	0.4
4	180	0	=C5	=D5	=E5	=F5	=G5	=H5	…	0.2
5	80	0	=C6	=D6	=E6	=F6	=G6	=H6	…	0.2
6	40	1	=p*C3	=p*D3	=p*E3	=p*F3	=p*G3	=p*H3	…	0.2
2	State \ Stage	0	1	2	3	4	5	6	…	183
3	Auss.	0	0	0	1	0.5	0.25	0.125	…	0.4
4	180	0	0	1	0	0	0	0.5	…	0.2
5	80	0	1	0	0	0	0.5	0.25	…	0.2
6	40	1	0	0	0	0.5	0.25	0.125	…	0.2

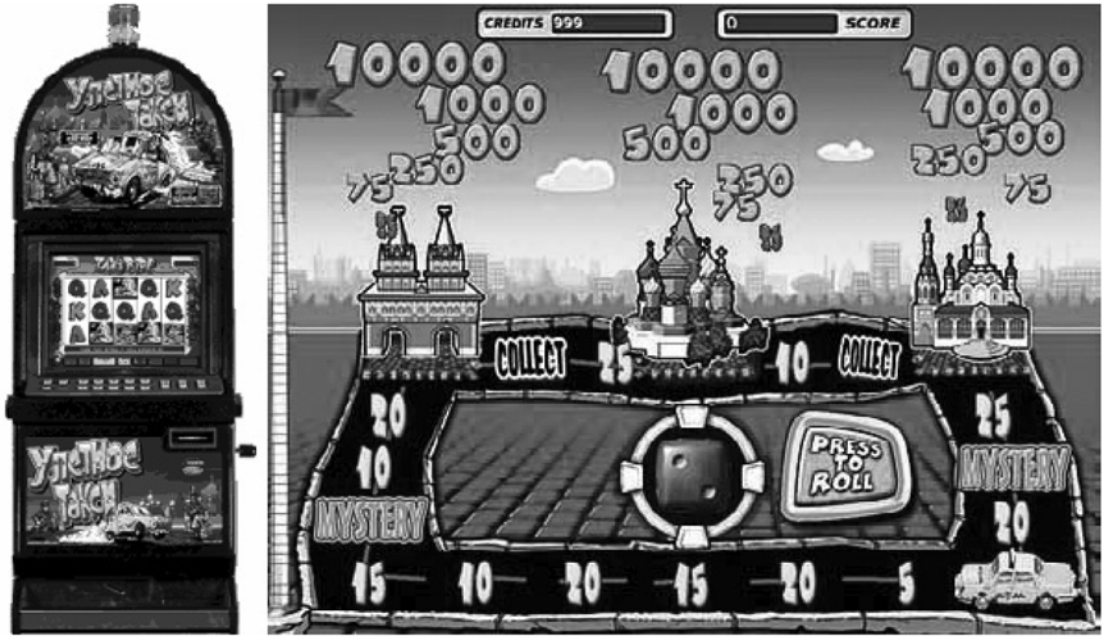
building the sets of states with transient probability non-zero for each of the stages and, from these, calculating the expected value. The main advantage of this approach is that it saves memory by only storing the information that is absolutely necessary.

The following two sub-sections analyse cases in which Markov chains have been used to model bonus features. The first, Taxi Ride, illustrates the modelling of a trail. Trails are a very popular feature in this industry and, therefore, no study of this industry that omitted them would be complete. An example of a very basic trail was given in (Oses and Freeman, 2006). The Taxi Ride bonus game illustrates a more sophisticated trail, highlighting the issues one might encounter when implementing the model as a computer program. The size of the state space prevents a spreadsheet solution for this game. The second example, Funny Fruits, illustrates the problems faced when the state space is very large. This is often the case with games in this category, which, as a direct consequence of this, are more difficult to implement in practice than the games in the previous section.

Taxi Ride

Taxi Ride is a 5-reel video slot game with a "Russian taxi" theme (Figure 2) released in December 2005 by the Russian manufacturer Unicum Group and designed and developed by the British Barcrest Games ([5]). The game was modelled by the author while working for the latter. Its features include line wins, gambles, two different bonuses, free spins and super free spins (the latter are special-feature free spins won during regular free spins) ([6]). This paper is only concerned with the "Taxi Ride" bonus triggered by three or more «Taxi» symbols on a pay line.

Figure 2 Taxi Ride screen shots



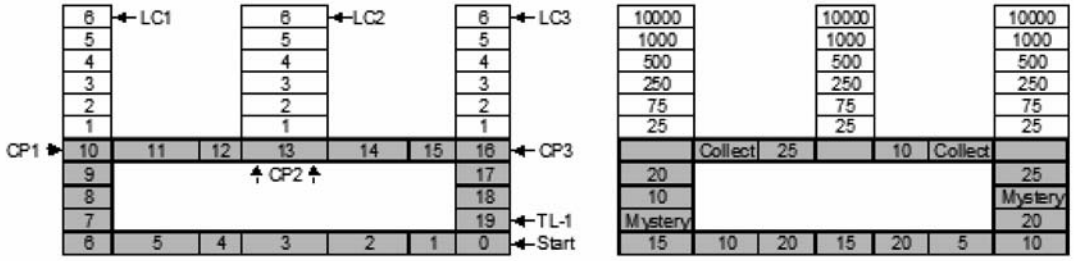
The "Taxi Ride" bonus feature is a wraparound trail represented on screen by a cartoon road map of Moscow (see right-hand side of Figure 2). The player moves clockwise around the trail, starting at the bottom right corner position for an opening award of 10 credits. Any value landed on is added to the player's win. The random roll of a die determines the moves. Most of the positions on the trail have a fixed prize associated with them. 'Collect' symbols mean that if the player falls on one of these the game finishes; these do not add to the accumulated prize. Positions 7 and 18 are 'Mystery' symbols. When the player falls on one of these symbols, he then carries on, collects or gets 'Extra Cash' according to a predetermined probability distribution. The 'Carry On' and 'Collect' outcomes of the mystery do not add to the accumulated prize. If the player falls on a 'Mystery' symbol and gets the 'Extra Cash' option, then the number of extra credits won will depend on a predetermined distribution. Positions 10, 13 and 16 on the trail have columns of values on top of them. The number of credits the player gets when he falls on one of these symbols depends directly on the number of times he has fallen on that same position before. The first time he falls on one of those positions the player gets the first value in the column. The second time he gets the second value in the column, and so on.

The player will carry on until one of the following three events happens. Landing on a 'Collect' position terminates the bonus and awards all credits accumulated to date. Landing on a 'Mystery' position and receiving the 'Collect' outcome will also terminate the bonus and award all credits accumulated to date. Last, surviving 50 throws of the die terminates the bonus game to receive a bonus of 10,000 credits plus any additional credits accumulated to date.

Mathematical model

Let TL denote the length of the trail, thus the trail positions go from 0 (starting position) to TL-1. There are three positions (positions 10, 13 and 16) that have prize columns above them. Let's call these trail positions CP_j for j = 1, 2, 3 (CP_1=10, CP_2=13 and CP_3=16). Let LC_j denote the number of positions in the prize column for j = 1, 2, 3. If the player falls on CP_j more than LC_j times then, after the LC_j-th time, the LC_j-th prize in the column will be awarded always, for j = 1, 2, 3. See Figure 3 for a diagram of the trail (right) and its key positions (left).

Figure 3 Trail indexes and trail



Let TP_n be the random variable that represents the player's position on the trail at stage n and let's denote the current position in the prize column above the CP_j trail position at stage n by $C_{j,n}$ for j = 1, 2, 3. Then, the state of the game at stage n is specified by the vector $S_n = (TP_n, C_{1,n}, C_{2,n}, C_{3,n})$. We will also define a state that will symbolise the end of the game when the player falls on a 'Mystery' symbol and the 'Mystery' results in 'Collect'. We will call this state 'MysteryCollect' and will be represented by (TL,0,0,0). $P_{Mystery}$ denotes the probability distribution of the 'Mystery' resulting in 'CarryOn', 'ExtraCash' or 'Collect'. D is the random variable that represents the outcome of a spin of the die, for which P(D = j) = p_j for j = 1,…,6 where $p_j \geq 0$ for all j and $p_1+...+p_6=1$. Finally, N is the maximum number of throws of the die the player is allowed to have; if the game has not finished after N throws then he is awarded a bonus prize in addition to the accumulated prize and the game ends. For Taxi Ride, N is 50.

The state space is too large to show the transition matrix as such. The general formulae are as follows. For any two states s and r where $s = (i,c_1,c_2,c_3)$ and r = (j,c'_1,c'_2,c'_3), $i, j \in \{0,..., TL - 1\}$ and $c_1, c'_1 \in \{0,1...,LC_1\}$, $c_2,c'_2 \in \{0,1,...,LC_2\}$, $c_3, c'_3 \in \{0,1,...,LC_3\}$, $0 < n+1 \leq N$:

- If s is a 'Collect' state, i.e. a 'Collect' position or the 'MysteryCollect' state, then:

$$P(S_{n+1} = r \mid S_n = s) = \begin{cases} 0 & r \neq s \\ 1 & r = s \end{cases}$$

It is possible to model the 'Collect' positions as absorbing states because they do not have a reward associated with them. Otherwise, a new 'Game Over' state with no reward associated to it would have to be introduced in the model.

- If s is a 'Prize' state, i.e. it is not a 'Mystery' or a 'Collect' state, then:
  - o Transitioning to a position on the trail that does not have a prize column above it, i.e. $j \neq CP_k \, \forall k =$ 1,2,3, the pointers to the positions on the prize columns do not change value:

$$P(S_{n+1} = r \mid S_n = s) = \begin{cases} P(D = k : (k+i) \bmod TL = j) & c'_1 = c_1, c'_2 = c_2, c'_3 = c_3 \\ 0 & \text{any other } r \end{cases}$$

  - o Transitioning to a position on the trail that has a prize column above it, i.e., $j = CP_k$ for some k=1,2,3, the pointer to the position on the k-th prize column increases in one position if it has not reached the top already and the other pointers do not change. We define .

$$k_l = \begin{cases} c_l & l \neq k \\ c_l + 1 & l = k \quad c_k + 1 \leq LC_k \\ c_l & l = k \quad c_k = LC_k \end{cases}$$

The state transition is as follows:

$$P(S_{n+1} = r \mid S_n = s) =$$
$$\begin{cases} P(D = l : (l+i) \bmod TL = CP_k) & c'_1 = k_1, c'_2 = k_2, c'_3 = k_3 \\ 0 & \text{any other } r \end{cases}$$

- If s is a 'Mystery' state, then:
  - The result of the 'Mystery' could be 'Collect', and it would transition to the 'MysteryCollect' state:

$$P(S_{n+1} = \text{'MysteryCollect'} \mid S_n = s) = P_{Mystery}(Collect)$$

  - Otherwise, for a 'Carry On' or 'Extra Cash' outcome, it will transition to either a position on the trail that does not have a prize column above it, i.e.:

i.e. $j \neq CP_k \; \forall k = 1,2,3$:

$$P(S_{n+1} = r \mid S_n = s) = \begin{cases} P_{Mystery}(CarryOn) + P_{Mystery}(ExtraCash) & * \\ 0 & \end{cases}$$

$$P(D = k : (k+i) \bmod TL = j) \quad \begin{matrix} c'_1 = c_1, c'_2 = c_2, c'_3 = c_3 \\ \text{any other } r \end{matrix}$$

  - Or it will transition to a position on the trail that has a prize column above it, i.e., $j = CP_k$ for some k=1,2,3. For k_l defined as above, the state transition is as follows:

$$P(S_{n+1} = r \mid S_n = s) = \begin{cases} P_{Mystery}(CarryOn) + P_{Mystery}(ExtraCash) & * \\ 0 & \end{cases}$$

$$P(D = l : (l+i) \bmod TL = CP_k) \quad \begin{matrix} c'_1 = k_1, c'_2 = k_2, c'_3 = k_3 \\ \text{any other } r \end{matrix}$$

This chain is stationary or homogenous and the 'Collect' and 'MysteryCollect' positions are absorbing states; all other states are transient. But it is the expected value that this industry is concerned with. Each state has a prize or reward associated with it, which depends on its trail position. From the above we can derive the probability distribution for each stage n, π_n ,and this multiplied by the prize vector (which is the same for all stages) yields the expected value of each stage. Adding the expected value for the first N stages (including stage 0) and the expected value of the bonus for surviving the N stages then we obtain the expected value of this game.

Analysis of the computer implementation

The solution could be implemented following "traditional" methods, that is, specifying the transition matrix and initial distribution and using Markov chain software, off-the-shelf or custom-built, to obtain the transient distributions. There are efficient methods to develop custom software for the solution of Markov chains with sparse matrices, like Compact Row Storage (CRS), to store these and templates or algorithms for the computation of numerical solutions (Barret et al., 1994; Stewart, 1994). However, the particular circumstances in which the problem is set can be exploited.

In MARCA (Stewart, 1991), for example, to generate the transition matrix the user can generate the state space by beginning in a given state and determining the set of all the states that may be reached from this initial state. The transition matrix is built simultaneously. Tan and Karabati (2000) also use a similar approach. However, this industry is not interested in learning about any of the characteristics of the Markov chain. Therefore, the transition matrix is not strictly necessary. The idea of generating the state space, nevertheless, will form the basis of the method developed for these games.

It is possible to have an iterative algorithm to generate the states and transient distributions without storing or using the transition matrix because transitions follow the same general mechanism for all states. This will be the 'generative' approach chosen for these games. In this case, for example, the transition mechanism consists of three different cases and is as follows. If the current state is a 'game over' state (either 'Collect' or 'MysteryCollect'), the game has finished and it transitions to itself (it is an absorbing state). If the current state is a 'Mystery' position, it will transition to 'MysteryCollect' with a certain probability and, with the complementary probability, it will transition to other states according to the result of the throw of the die. All other states, i.e. 'Prize' positions, will transition along the trail according to the result of the throw of the die.

The fact that the initial distribution is known and fixed also facilitates this approach, as we can take the fixed state from which the game starts as the first state.

This generative method is especially convenient since it is very intuitive, easy to program and saves memory. In Taxi Ride, the starting position is the (0,0,0,0) state, i.e. the probability that the chain is in this state in stage 0 is 1. The throw of a die will determine the transitions. Therefore, in stage 1 the chain will be one position further along the trail in state (1,0,0,0) with probability 1/6, two positions further in state (2,0,0,0) with probability 1/6, in state (3,0,0,0) with probability 1/6, in (4,0,0,0) with probability 1/6, in (5,0,0,0) with probability 1/6 or in (6,0,0,0) with probability 1/6. The same idea applies to the generation of other stages i.e. for each state that could be reached in the previous stage generate the states to which it can transition and the probability of reaching each of these in this stage. Each new transient distribution is stored and is used for the calculation of the transient distribution of the next stage. For each transient distribution, only the states with corresponding probability non-zero are stored. For each "new" state generated in a transition, the stage distribution generated so far is searched looking for this state. If it is already there, the probability is added to the probability already generated for transitioning to this new state from other states. If it is not, then the new state is added to the stage distribution with its associated probability. It is not necessary to store all the transient distributions in memory. At any given time, it is only indispensable to have in store the last transient distribution, which is needed to generate the next one, and the expected prize value of the stages generated so far. The expected value of each stage can be calculated as the stage distribution is being generated.

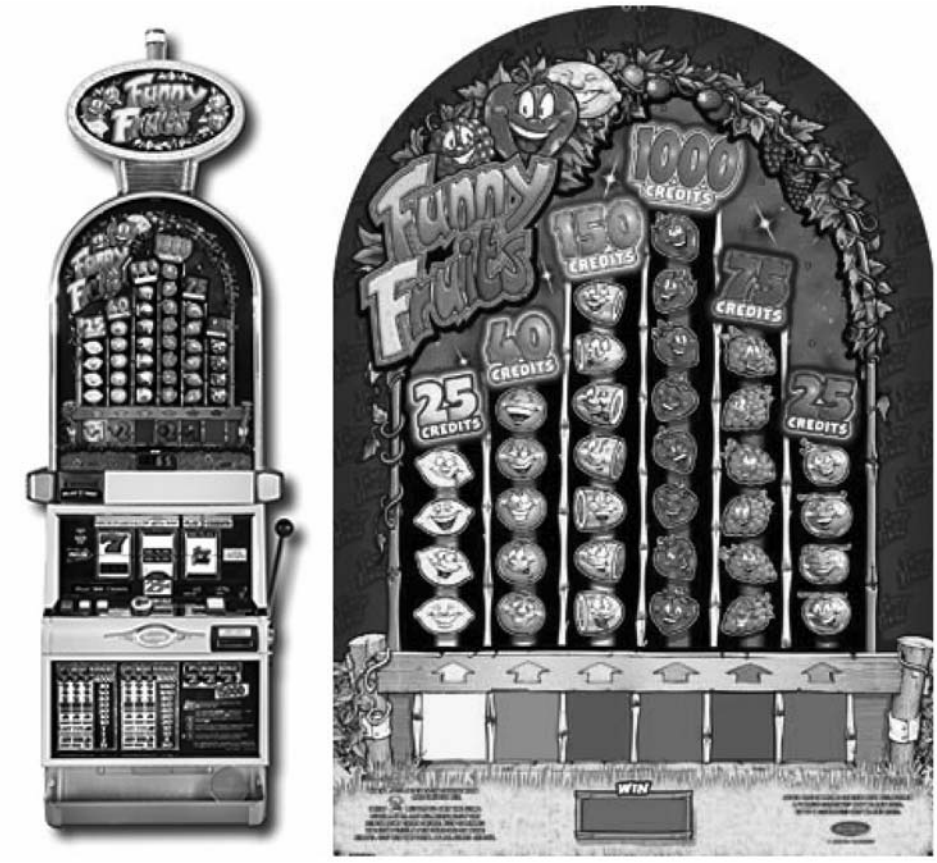
Funny Fruits

Funny Fruits is a 3-reel slot machine (Figure 4) released in 2005 by Barcrest USA ([7]). The features of this machine include line wins, a top award and the novel *Funny Fruits*™ bonus. The latter was modelled by the author while working for Barcrest Games. This paper is only concerned with the bonus game, which is triggered when a *Funny Fruits*™ symbol lands on the payline when playing maximum bet.

In this feature there are six stacks of fruits and a reel underneath (right-hand side of Figure 4). The reel underneath the stacks has 16 positions and six stops in view, each stop coinciding with one of the stacks. Pressing the 'Spin Reel' button causes the spinning of the bonus strip or reel. Numbers (0, +1 or +2) are revealed under the fruit stacks when the bonus strip stops spinning and the same number of fruits light up in the corresponding stack. This process continues until a spin fills one or more fruit stacks. When this happens, the player is awarded the prize(s) displayed on top of the filled stack(s). Then, the game continues or ends according to a probability distribution. Each stack has a set of probability distributions, called the "Carry On" distributions, associated to it. The probability distribution used depends on how many times stacks have been filled so far. If more stacks than one are filled simultaneously, the "Carry On" distribution of the highest stack filled is used. If the game carries on, the stack(s) that was (were) filled is (are) emptied. In Funny Fruits, there are a maximum of 3 carry ons, the fourth time a stack is filled the probability of carrying on is 0 for all stacks. When the bonus ends the player is awarded the sum of all the prizes accumulated so far.

This game is similar to a trail in the sense that the prize is cumulative and it is a random game that requires no input from the player. At each stage in the game, the progress only depends on the current positions of the stacks and the number of times tops have been reached so far, and it is determined by the outcome of the reel spin. Therefore, it can be modelled as a Markov chain.

Figure 4 Funny Fruits machine and bonus game



Mathematical model

$X_{1,n},\ldots, X_{6,n}$ are the random variables that represent the number of fruits lit up in each stack at time n. Let the $XT_1,\ldots,XT_6$ constants denote the total number of positions or fruits in each stack. NT_n is the random variable that represents the number of times that stack tops have been reached by stage n. Note that if more than one top is reached at stage n NT_n will only be increased by 1 in respect of NT_{n-1}. Let the constant N (equal to 4 in Funny Fruits) represent the maximum number of times that stack tops can be reached. Thus, the state of the chain at time n is represented by the vector $S_n = (X_{1,n}, X_{2,n}, X_{3,n}, X_{4,n}, X_{5,n}, X_{6,n}, NT_n)$ where $0 \le X_{i,n} < XT_i$ for $1 \le i \le 6$ and $0 \le NT_n \le N$.

The reel placed underneath the stacks has 16 positions ($R_0, R_1,\ldots, R_{15}$), and therefore, there are 16 possible outcomes of the spin. We will denote by R the random variable that represents the outcome of the spin of the reel, i.e. the reel index underneath the first stack after the spin. Thus, R takes values 0 to 15 where

$$P(R=i) = p_i,\ 0 \le p_i \text{ and } \sum_{i=0}^{15} p_i = 1.$$

If R = i, then the reel positions under the stacks will be the following: .

$R_i, R_{(i+1) \bmod 16},\ldots, R_{(i+5) \bmod 16}$. The game always starts with all the stacks empty. We will represent the "Game Over" state as (0,0,0,0,0,0,N+1).

Given two states, s and t, where:

$$s = \left(X_{1,n} = x_1, X_{2,n} = x_2, X_{3,n} = x_3, X_{4,n} = x_4, X_{5,n} = x_5, X_{6,n} = x_6, NT_n = nt\right) \text{ and}$$

$$t = \left(X_{1,n+1} = x_1', X_{2,n+1} = x_2', X_{3,n+1} = x_3', X_{4,n+1} = x_4', X_{5,n+1} = x_5', X_{6,n+1} = x_6', NT_{n+1} = nt'\right)$$

then the transition probabilities are as follows.

- If s is not the "game over" state and it has not reached any tops then the reel spin determines the transition:

$$P(S_{n+1} = t \mid S_n = s) =$$

$$P\left(\bigcup_{0 \le i < 16} \left\{ \begin{array}{l} R = i: \\ \left(\begin{array}{l} x_j' = \left(x_j + R_{i+j-1 \bmod 16}\right) \text{if } \left(x_j + R_{i+j-1 \bmod 16}\right) < XT_j \vee \\ x_j' = XT_j \text{ if } \left(x_j + R_{i+j-1 \bmod 16}\right) \ge XT_j \end{array} \right) \text{for j} = 1,\ldots,6 \wedge \\ \left(\begin{array}{l} nt' = nt \text{ if } \left(x_j + R_{i+j-1 \bmod 16}\right) < XT_j \ \forall \text{j} \vee \\ nt' = nt + 1 \text{ if } \left(x_j + R_{i+j-1 \bmod 16}\right) \ge XT_j \text{ for some j} \end{array} \right) \end{array} \right\} \right)$$

- If s is not the "game over" state and it has reached one or more stack tops then, if the "carry on" test is successful for the highest of the full stacks it will transition to emptying the full stacks, otherwise the game will end:

For t where $x_i' = \begin{cases} x_i & x_i < XT_i \\ 0 & x_i = XT_i \end{cases}$ and $nt' = nt$:

$$P(S_{n+1} = t \mid S_n = s) = P_{\text{highest top}}(\text{"Carry On"} \mid nt)$$

and

$$P(S_{n+1} = \text{"}GameOver\text{"} \mid S_n = s) = 1 - P_{\text{highest top}}(\text{"Carry On"} \mid nt)$$

- If s is the "game over" state then $P(S_{n+1} = t \mid S_n = s) = \begin{cases} 1 & t = s \\ 0 & t \ne s \end{cases}$

This Markov chain is also homogeneous and the "game over" state is an absorbing state, while all other states are transient.

Implementation of the model and practical issues

Much effort must be put into efficient memory management when implementing models with large state spaces. Therefore, since the transition matrix is not strictly necessary, the initial distribution is fixed and the transition mechanism is very simple, it is convenient to follow the 'generative' approach developed for the previous game, Taxi Ride.

Furthermore, to reduce the amount of memory needed we can reduce the number of states by changing the transitions when stacks are filled. If, after a transition, the new state has one or more stacks that have been filled, instead of storing this new state (that transitions to emptying the stacks or to "game over") we can store the two resulting states. The first resulting state is the state that results from emptying the full stacks with the associated probability equal to the product of the probability of reaching the state generated and the carry on probability. And the second is the end state with the associated probability equal to the product of the

probability of reaching the state generated and the probability of not carrying on. The number of stages, when modelling the game in this manner, will correspond to the number of reel spins, which is more intuitive.

Another lesson learned from this case is that the execution time of the model is also a concern. In Funny Fruits, the "arithmetic" involved in processing the transition (that is, the current state plus the reel outcome equals the new state) would be done again and again following the 'generative' algorithm. Due to the large size of the state space, this is not a time-efficient way to implement a solution. It is a better approach to generate all the stack-states and stack arithmetic first (current state of the stacks plus reel outcome, ignoring the number of times that tops have been reached). Then, generate the state space and, maybe, the state arithmetic if the memory can cope with this. And only then, generate the stage distributions. Repetitive operations or necessary information should be prepared before starting the generative algorithm.

Conclusions

The slot machine industry used to favour approaches based on long-run simulations or complete enumerations of all possible outcomes but has now adopted mathematical modelling as the preferred method. This paper has shown that Markov chain modelling is an important tool for this industry and that bonus games modelled as Markov chains usually share a set of characteristics. The process usually starts from a fixed initial distribution, transitions are generally determined by some simple random mechanism, states have rewards associated with them and the industry is only interested in the expected prize value of the game. It has been shown that these particularities can be exploited when implementing solution algorithms. In particular, it has been shown that the transition matrix is not strictly necessary when implementing these models, and its omission reduces the amount of memory needed. For start-and-end games a generative approach is proposed for the generation of states and transient distributions. It has been learned that repetitive operations and other necessary information should be prepared before starting the generative algorithm. Selected case studies have illustrated these points.

This paper has focused on games in which the player has no part. The next step is to look at games in which the player has to make decisions after each transition. These can be modelled as Markov decision processes.

References

Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and van der Vorst H. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition. SIAM: Philadelphia, PA; 1994.

Ching W-K, Ng M K, Wong K-K and Altman E. Customer lifetime value: stochastic optimisation approach. Journal of the Operational Research Society 2004; 55; 860–868.

David, F-N. Games, Gods and Gambling: The origins and history of probability and statistical ideas from the earliest times to the Newtonian era. Charles Griffin Co. Ltd., London; 1962.

Feiring B R, Sasfri T C P, Rao Tummala V M and Mak R W. Modelling the cost of poor quality in a five-state part manufacturing operation: a case study. Journal of the Operational Research Society 1998; 49; 1249-1253.

Freeman J M. Gambling on HI-LO: an evaluation of alternative playing strategies. Journal of the Operational Research Society 1998; 49; 1278-1287.

Gambling Act. Regulatory Impact Assessment. Department for Culture, Media and Sport 2005.

Gaming Board. Gaming Board for Great Britain 2004-5 Annual Report. The Stationery Office, London 2005.

Hunt B R, Lipsman R L, Rosenberg J and Coombes K R. A guide to MATLAB: for beginners and experienced users. Cambridge University Press, Cambridge, UK; 2001.

Oses N and Freeman J. Hitting the jackpot with OR. OR Insight 19 (3); 2006.
Stewart W J. Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Princeton, New Jersey 1994.
Stewart W J (editor). MARCA: Markov Chain Analyzer, a software package for Markoc chains. Numerical Solutions of Markov Chains, Marcel Dekker Inc; 1991.
Tan W-Y. Stochastic Models with Applications to Genetics, Cancers, AIDS and Other Biomedical Systems. World Scientific Pub.Co., Singapore; 2002.
Tan B and Karabati S. Modelling and scheduling of an asynchronous cyclic production line with multiple parts. Journal of the Operational Research Society 2000; 51; 1296-1308.

Web references

http://www.gamcare.org.uk/site.builder/onlinehelp.html accessed 30th March 2006
http://www.gamblingcommission.gov.uk/Client/detail.asp?ContentId=28 accessed 30th May 2006
http://www.gamblingcommission.gov.uk/Client/detail.asp?Contentid=46 accessed 31st May 2006
http://195.35.109.83/walberer/geldspieler/addsladder.html accessed 19th June 2006
http://eng.unicum.ru/news/113.html accessed 24th April 2006
http://eng.unicum.ru/catalogue/games/381.html?prod=535 accessed 24th April 2006
http://www.igt.com/GamingGroup/Games/game_detail.asp?toggle=ovr&pid=5.113.120&type_id=3373&pl=#cnt accessed 24th April 2006

Noelia Oses, Ph.D.

NOF Consulting -
www.nofcon.com

Noelia Oses is a freelance consultant for the slot machine industry. Her work focuses on developing probability models of slot games to calculate the probability distribution of the prizes and adjust the maximum percentage return in the long term or, in other words, to control the minimum house edge. Her research interests include applying OR, probability and stochastic processes techniques in this industry.

Noelia has been a member of The OR Society since 1999.

The OR Society,
Seymour House, 12 Edward Street
Birmingham, B1 2RX, UK
Tel: + 44 (0) 121 233 9300
Fax: + 44 (0) 121 233 0321
email@orsoc.org.uk