SOLVING THE OSHI–ZUMO GAME

M. Buro
University of Alberta, Edmonton, AB T6G 2E8, Canada
mburo@cs.ualberta.ca, http://www.cs.ualberta.ca/~mburo/

Abstract Kotani (2002) determined the part of the state space of the Japanese Oshi-Zumo game in which pure strategies suffice to win. This paper completes the analysis by computing and discussing a Nash-optimal mixed strategy for this game.

Keywords: Nash-optimal strategy, Oshi-Zumo, two-player game

1. Introduction

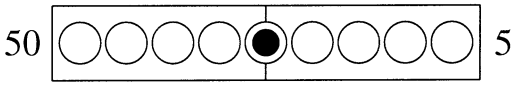
In this article the Japanese game Oshi-Zumo is analyzed. Moves in this game consist of simultaneous actions by two players who otherwise have complete information about the current game state. In general, such games can be represented by a collection of payoff matrix pairs whose entries define the expected amount paid to the players in case the respective action pair was chosen. It is well known that not knowing the opponent's action already makes it necessary to consider mixed strategies and that so-called Nash-optimal mixed strategies exist for any matrix game (Nash, 1950). A simple example is the Rock-Paper-Scissors game in which Rock beats Scissors, Scissors beats Paper, and Paper in turn beats Rock. The Nash-optimal strategy picks each of the actions with probability $\frac{1}{3}$.

In what follows, we first introduce the Oshi-Zumo game. It is more complex than Rock-Paper-Scissors, but considerably simpler than other popular incomplete information games such as Poker and Bridge. In fact, we will show how to compute a Nash-optimal strategy within seconds on ordinary PC hardware. We then highlight interesting properties of a Nash strategy and conclude the paper by discussing how the optimal player performs against reasonable, but sub-optimal strategies.

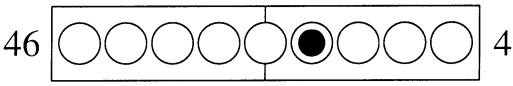
2. The Game

Oshi-Zumo – meaning "the pushing sumo (wrestler)" – is played by two players who both start off with N coins. At the beginning of a game, a sumo

H. J. Van Den Herik et al. (eds.), *Advances in Computer Games*



[50,4,∗]-Oshi-Zumo starting position – code $(50, 50, 0)$



Position after move (4,2) – code $(46, 48, 1)$

Figure 1. Oshi-Zumo positions and their triple representation.

wrestler is positioned at the center of a one-dimensional playing field which consists of $2K + 1$ locations (see Figure 1). Moves are played by secretly choosing a number of coins less or equal to the amount currently available to the respective player, but at least M. The bids are then revealed and the highest bidder pushes the wrestler one location towards the opponent's side. If the bids are equal, the wrestler does not move. Both bids are deducted and the game proceeds until the money runs out or the wrestler is pushed off the playing field. The final position of the wrestler determines the winner: if he is located at the center, the game result is a draw. Otherwise, the player in whose half the wrestler is located loses the game. We call this parameterized game an $[N, K, M]$-Oshi-Zumo game. In this paper we only consider the minimal bids $M = 0$ and $M = 1$ and declare a game over if both bids are 0. As before, the winner is determined by the current wrestler position.

3. Computing a Nash-Optimal Strategy

Certain Oshi-Zumo positions possess pure winning strategies. For example, all positions in which the opponent has no money left and the wrestler position is sufficiently advanced can be won by simply bidding one coin for the remainder of the game. Kotani (2002) determined all such positions for the standard $[50, 3, 1]$-Oshi-Zumo game. The following list specifies some more interesting $[50, 4, 0]$-positions that can be won by the first player with a pure strategy:

$$(n, n, 1) : 1 \leq n \leq 11 \text{ [bid 1]} \qquad (n, n+1, 2) : 1 \leq n \leq 12 \text{ [bid 1]}$$

$$(50, n, -4) : 1 \leq n \leq 16 \text{ [bid } n\text{]} \quad (49, n, -4) : 1 \leq n \leq 16 \text{ [bid } n\text{]}$$

All such positions can be computed by dynamic programming for small values of N and K because the size of the state space is only a polynomial $(N+1)^2 \times (2K + 3)$ in the parameters. First, we compute the payoffs P_i for both players at the boundary positions:

$$P_1(0, 0, k) = -P_2(0, 0, k) = \text{sign}(k), \text{ for } -K \leq k \leq K$$

$$P_1(n, m, \pm(K+1)) = -P_2(n, m, \pm(K+1)) = \pm 1, \text{ for } 0 \leq n, m \leq N$$

maximize Z such that

for all $M \leq j \leq n_2 : \; Z \leq \sum_{i=M}^{n_1} A_{i,j} x_i$,

for all $M \leq i \leq n_1 : \; x_i \geq 0$, and

$\sum_{i=M}^{n_1} x_i = 1$

minimize Z such that

for all $M \leq i \leq n_1 : \; Z \geq \sum_{j=M}^{n_2} A_{i,j} y_j$,

for all $M \leq j \leq n_2 : \; y_j \geq 0$, and

$\sum_{j=M}^{n_2} y_j = 1$

Figure 2. Linear programs (LPs) for determining Nash-optimal mixed strategies.

Then we search for positions with pure winning or drawing strategies, or ones that lose for sure no matter what. A position is won for player A if there exists an action such that for all actions of the opponent the expected payoff for A is 1. Declaring a position drawn or lost requires that all successor position values are known. We repeat this process until we do not find any new position values.

A Nash-optimal strategy can be computed similarly. Starting again with assigning values to the boundary positions, we iterate through all positions with unknown expected payoff until we find one for which all successor values have been established. At this time we make use of the fact that optimal strategies $\{(i, x_i) \mid M \leq i \leq n_1\}$ and $\{(j, y_j) \mid M \leq j \leq n_2\}$ for players MAX and MIN can be found by solving two linear programs (see Figure 2). MAX has move choices $M, \ldots n_1$ and MIN has actions $M, \ldots, n_2$. x_i and y_j denote the respective action probabilities. Matrix element $A_{i,j}$ defines the payment for MAX if action pair (i, j) is chosen. Because Oshi-Zumo is a zero-sum game, MIN receives the negated amount. Z denotes the expected payoff for MAX. This procedure eventually halts and computes the expected payoffs and mixed strategies for all positions.

We decided to not only create a table containing expected payoffs – which would be sufficient for computing values for all positions – but also to store the move distributions to speed up later game play and move analyses. Only one distribution needs to be computed and stored for each position because the move distribution for the second player in position (n, m, k) is identical to that of the first player at $(m, n, -k)$.

4. Implementation Issues

In our first implementation we adopted Michel Berkelaar's open-source software package LPSOLVE. Unfortunately, the solver ran into numerical problems which caused it to either give up on instances or report incorrect solutions. Implementing efficient LP solvers is by no means easy. In order to overcome the numerical problems we decided to replace floating-point by rational arithmetic in LPSOLVE – which turned out to be more complicated than expected. Finally,

we took the simpler LP solver code from Press et al. (1992) and combined it with GMP – the GNU arbitrary precision arithmetic library – by replacing the float/double data types by GMP's rational number C++ class. Solving LPs using rational arithmetic takes much longer than using floating-point values, even if the denominators are bounded. In order to speed up the Oshi-Zumo solver we therefore implemented a two-phase approach: whenever the fast LP solver reported problems or produced inconsistent results, we would start the slow solver based on rational arithmetic. We bounded denominators by 10^8 and normalized rational numbers whenever this limit was exceeded. Test runs on Oshi-Zumo games manageable by the floating-point based solver indicated that the results obtained by rational arithmetic only differed by a negligible amount. On a notebook PC with a 1-GHz Pentium-III CPU, solving the standard $[50, 3, 1]$ game takes just 12 seconds. The C++ source code can be downloaded from `http://www.cs.ualberta.ca/~mburo/sumo.tgz`.

5. A Nash-Optimal Oshi-Zumo Strategy

In what follows we concentrate on the $[50, 3, 0]$ and $[50, 3, 1]$ versions of the game and highlight interesting properties of their respective Nash-optimal strategies. We start by looking at the move distributions for the starting position:

$M = 0$ position=$(50, 50, 0)$ value= 0.0

bids:	0	1	2	3	4	5	6	7	8	9	10
prob:	.083	.077	.088	.083	.092	.088	.097	.092	.099	.094	.101

$M = 1$ position=$(50, 50, 0)$ value= 0.0

bids:	1	2	3	4	5	6	7	8	9
prob:	.139	.053	.146	.060	.152	.067	.156	.068	.156

Apparent is an "odd-even" effect in which higher and lower bid probabilities alternate. This probability pattern occurs in many positions. Why it occurs is an open question.

The smallest positions with randomization requirement are $(5, 2, -3)$ for $M = 0$ and $(6, 3, -3)$ for $M = 1$. The move distributions are as follows:

$M = 0$
position = $(5, 2, -3)$
$\text{value}_1 = -0.5$

bid_1:	1	2
prob:	.5	.5
bid_2:	0	2
prob:	.5	.5

$M = 1$
position = $(6, 3, -3)$
$\text{value}_1 = -0.5$

bid_1:	1	3
prob:	.5	.5
bid_2:	1	3
prob:	.5	.5

In 5,271 cases of the 23,409 possible $[50, 3, 0]$-positions, and in 4,057 cases for $M = 1$, more than one move has to be considered. To illustrate how complex the move decision can be, we present two positions with a high number of holes in the move distribution:

$M = 0$ position = $(17, 34, 3)$ $\text{value}_1 = 0.047$

bid_1:	0	2	3	4	5	6	7	8	10	12	14	17
prob:	.482	.015	.008	.017	.016	.020	.022	.026	.069	.087	.107	.126
bid_2:	1	2	3	4	5	6	7	17				
prob:	.066	.065	.086	.080	.082	.082	.058	.476				

$M = 1$ position = $(20, 32, 3)$ $\text{value}_1 = 0.333$

bid_1:	1	2	3	5	6	7	9	11	12	14	17	18	20
prob:	.369	.004	.038	.050	.007	.038	.125	.069	.048	.039	.080	.030	.096
bid_2:	2	3	4	5	6	9	10	11	12	20			
prob:	.136	.021	.079	.029	.059	.016	.188	.091	.043	.333			

Given such complex distributions, the question arises how well human players can play Oshi-Zumo.

6. How Good is Optimal?

Playing any mixed or pure strategy against a Nash-optimal player results in an expected payoff no better than the expected value E of a game between two Nash players. In contrast, the expected value of any pure strategy that picks actions from the set an optimal strategy considers, is exactly E when playing against the Nash-optimal player. This follows from the fact that all actions with non-zero probability have the same expected value. Therefore, the Nash-optimal solution is far from optimal with respect to exploiting simple (pure) strategies, such as playing Rock all the time in a sequence of Rock-Paper-Scissors games. In Rock-Paper-Scissors the Nash strategy cannot win anything against any other strategy in the long run. However, in more complex games – such as Oshi-Zumo or Poker – it can, because not all actions have non-zero probability in all situations.

A player who just memorizes one move from a Nash-optimal strategy for each position does not lose money against a Nash-player in the long run. How much does a player lose who occasionally plays moves not played by a Nash-player and how well do simple hand-crafted strategies play? To answer these questions we wrote a program that played a large number of games between a Nash-optimal strategy and several simple move selection algorithms. Figure 3 presents the tournament results. As expected, the completely random player loses almost every game. The player that randomly chooses bids in the interval formed by the minimal and maximum Nash bid performs much better and loses only about 0.035 units per game for $M = 0$ and 0.01 for $M = 1$. Simply choosing moves in a small fixed interval also leads to good results and shows how easy it is to look good against a Nash player. Also some fairly simple pure strategies perform surprisingly well.

A more interesting question is therefore how to adapt to players and exploit their weaknesses while minimizing the risk of being exploited. We think that

$M = 0$		$M = 1$	
random 0..#	−.97882	random 1..#	−.98216
random Nash range	−.035	random Nash range	−.0105
random 1..min(6,#)	−.31884	random min(2,#)..min(6,#)	−.3533
random 1..min(5,#)	−.16971	random min(2,#)..min(5,#)	−.21524
random 1..min(4,#)	−.05115	random min(2,#)..min(4,#)	−.05683
random 1..min(3,#)	−.00292	random min(2,#)..min(3,#)	−.00372
random 1..min(2,#)	+.000645	if #≥2 2 else 1	+.00039
1	−.002765	if #≥3 3 elif #≥2 2 else 1	−.02987
if #≥2 2 else 1	−.00156		

Figure 3. The average payoff of various simple move-selection algorithms playing 200,000 $[50, 3, M]$-games against a Nash-optimal strategy. # denotes the current number of coins left for the heuristic player.

using games simpler than say Poker but harder than Rock-Paper-Scissors as test domains can shed light into this interesting problem, which appears to be the last remaining hurdle on the way to Poker programs stronger than human players (Billings et al., 2003). Oshi-Zumo is a suitable candidate because its Nash-optimal strategy is non-trivial, but can be computed quickly.

Acknowledgement

Thanks go to Darse Billings for helpful discussions clarifying questions on Nash-optimal strategies.

References

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., and Szafron, D. (2003). Approximating Game-theoretic Optimal Strategies for Full-scale Poker. *Proceedings of the International Joint Conference on Artificial Intelligence*. To appear.

Kotani, Y. (2002). Analysis of the Pushing Sumo Game and its Application to Creativity Education. *IPSJ SIG-Notes Game Informatics*, Vol. 8.

Nash, J.F. (1950). Equilibrium Points in n-Person Games. *National Academy of Sciences*, Vol. 36, pp. 48–49.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992). *Numerical Recipes in C*. Cambridge University Press. 2nd edition.