Journal of the Operations Research Society of Japan
Vol. 62, No. 2, April 2019, pp. 64–82

A DYNAMIC PROGRAMMING ALGORITHM FOR OPTIMIZING BASEBALL STRATEGIES

Akifumi Kira
Gunma University

Keisuke Inakawa
Akita Prefectural University

Toshiharu Fujita
Kyushu Institute of Technology

(Received May 15, 2018; Revised October 1, 2018)

Abstract In this paper, baseball is formulated as a finite non-zero-sum Markov game with approximately 6.45 million states. We give an effective dynamic programming algorithm which computes equilibrium strategies and the equilibrium winning percentages for both teams in less than 2 second per game. Optimal decision making can be found depending on the situation—for example, for the batting team, whether batting for a hit, stealing a base or sacrifice bunting will maximize their win percentage, or for the fielding team, whether to pitch to or intentionally walk a batter, yields optimal results. Based on this model, we discuss whether the last-batting team has an advantage. In addition, we compute the optimal batting order, in consideration of the decision making in a game.

Keywords: Dynamic programming, OR in sports, Markov perfect equilibrium, advantage of the last-batting team, optimal lineup

1. Introduction

A dynamic programming (DP) approach to baseball is the main theme for this paper, and we first see a prototype of this idea in Howard's famous book [13]. He set maximization of the expected number of runs scored for one inning as a criterion, formulating baseball as a Markov decision process with 25 states. Orders from the manager, such as base stealing, sacrifice bunting, and batting for a hit, were also taken into consideration. Howard's work is based on the assumption that all nine batters on the team have equal abilities. The transition probabilities (the success rate of sacrificing, etc.) were artificially set and the optimal strategies were determined using a computer of that time. Bellman [2] proposes a more detailed formulation. He provided a subtle insight into strategies through a discussion based on the two criteria of maximizing the expected number of runs scored and the threshold probability of scoring at least k runs in one inning. However, due to the shortage of computing capacity at the time, Bellman's approach was not implemented.

On the other hand, D'Esopo and Lefkowitz [8] formulate baseball as a Markov chain with 25 states and propose a scoring index (SI) as an evaluation index for the expected number of runs scored in one inning, assuming that the same player steps up to the plate repeatedly. Cover and Keilers [7] also propose a similar index, the OERA (Offensive Earned-Run Average) value, as an index to evaluate the expected number of runs scored in a single game. While Howard and Bellman focused on strategy optimization, the goal of these evaluation indices is to express the contributions of each individual player in numerical form. Because of the tractability, this approach became popular and attracted the interest of many subsequent studies (e.g., Ano [1], Bukiet et al. [6]).

Now, the ability of computers has rapidly developed and approaches using DP have become possible. Hirotsu and Wright [10–12] formulate baseball as a Markov chain with

approximately 1.41 million states, and use DP to optimize player substitution strategies. Turocy [28] formulates baseball as a zero-sum Markov game with approximately 2.13 million states, and adopts the MLB rule to play extra innings until a winner is determined (i.e., the length of the game is finite with probability 1). Here, the manager of each opposing team is the game-theoretic player who maximizes the probability of their team winning. As an order from the manager, intentional walk is also taken into consideration, and a Markov perfect equilibrium (MPE) exists. Turocy performed numerical experiments using backward induction from the start of the game up to the completion of the eighth inning, and using a fixed-point approximation by a value-function iteration for the ninth inning. The details of the recursive formula and the algorithm are omitted in the paper, but he states that the values of the game (the equilibrium winning percentages for both teams) could be solved with high accuracy in less than a minute.

Kira and Inakawa [16], our previous paper, formulate baseball as a non-zero-sum finite Markov game with approximately 3.5 million states. We adopts Japanese NPB rule restricting extra innings to a maximum of three (i.e., the length of the game is finite, but the game may end in a draw.), and at least one pure-strategy MPE does exist. Our previous DP approach successfully realizes to compute a pure-strategy MPE in approximately 1 second per game. However, the values of the game depend on which MPE is obtained, because the NPB rule makes the game non-zero-sum. This non-uniqueness makes it difficult to measure the effects of strategies and players. In addition, another disadvantage is that the details of algorithm is omitted in our previous paper.

In this paper, we improve the disadvantage of our previous results. We take the intentional walk into consideration and formulate baseball as a finite non-zero-sum Markov game with approximately 6.45 million states. Our first contribution of this paper is reformulation to guarantee the uniqueness of the values of the game. When there are multiple choices that maximize their winning percentage, managers will prefer the one with the lowest losing percentage (i.e., the highest probability of ending in a draw) among them. By taking this into consideration, we define the class of lexicographic MPEs to be obtained. We derive a recursive formula that is satisfied by the lexicographic MPEs and the value functions of the game. Our second contribution is a detailed description of a depth-first search DP algorithm which realizes to calculate the value functions of the game and a lexicographic MPE in 2 second per game. Our algorithm can also calculate the values of the game under the MLB rule with high accuracy by increasing the extra inning limit. Our third contribution is an analysis of whether the last-batting team has an advantage. We discuss it based on the value of information in game theory. Our final contribution is a computation of the optimal lineup. So far as we know, there has been no study that has tried to optimize batting order, in consideration of strategy optimization such as a sacrifice bunt or a stolen base. Our effort reducing the computational time per game makes it possible.

In the theory of MDPs, the usual optimization criterion is to maximize the expected value of the total (discounted) sum of stage-wise rewards. It is well known that finite MDPs under this criterion can be solved in polynomial time. On the other hand, the threshold probability problem, which attempts to maximize the probability that the total sum of stage-wise rewards exceeds a specified value, has been extensively studied by many researchers (Boda et al. [3], Boda and Filar [4], Bouakiz and Kebir [5], Iwamoto et al. [14], Kira et al. [17], Ohtsubo [22], Ohtsubo and Toyonaga [23], Sakaguchi and Ohtsubo [24], Sobel [27], White [29, 30], Wu and Lin [31], and others), and this problem has been proved to be $\mathcal{NP}$-hard (Xu and Mannor [32]). So, handling with probability criteria is more difficult than that with the usual expectation criterion in general. The same applies to Markov games.

However, in baseball, our algorithm works very well. It will be clearly understood, through this paper, that baseball possesses some properties quite suitable for DP computation.

2. Formulation as A Markov Game

In this section, baseball is formulated as a finite non-zero-sum Markov game, where a Markov game is a multi-agent extension of MDPs (e.g., see Shapley [25] and Zachrisson [33]). The difference from our previous model [16] is taking the intentional walk into consideration. When a batter steps up to the plate, we always assume that the team in defense chooses their action whether to pitch to or to intentionally walk the batter before the team in offense chooses their action whether batting for a hit, stealing a base or sacrifice bunting. Hence our model is still a sequential game.

For convenience, we call the first-batting team and the last-batting team "team 0" and "team 1" respectively.

2.1. States

Let $\mathcal{S}$ be the state space. A state $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S}$ is made up of 7 components. Each component is defined as follows:

1. $\iota \in \{1, 2, \ldots, 12\}$ represents the current inning. $\iota = 9$ is the final inning, and for a tie, extra innings are played up to a maximum of $\iota = 12$.
2. $\tau \in \{0, 1\}$ represents offense in the top half of the inning ($\tau = 0$) or offense in the bottom half of the inning ($\tau = 1$).
3. $\omega \in \{0, 1, 2, 3\}$ represents the current number of outs.
4. λ is the current run difference and represents the value found by subtracting the runs of team 1 from the runs of team 0. For the purpose of determining the final winner, we store not the runs scored by each team but the current run difference. This is a state aggregation technique (e.g., see Sniedovich [26, Chap. 11]).
5. $\boldsymbol{r} = (r_3, r_2, r_1)$ represents the state of the runners.
   - $r_3 \in \{0, 1\}$ takes a value 0 if there is no runner on third base, and a value 1 if a runner is present.
   - $r_2 \in \{0, 1\}$ takes a value 0 if there is no runner on second base, and a value 1 if a runner is present.
   - $r_1 \in \{0, 1, \ldots, 9\}$ takes a value 0 if there is no runner on first base, and the same value as the batting order of the runner if a runner is present.

   Only r_1 distinguishes between runners, to take into account the success rate which is dependent on the runner when performing a stolen base from first to second base. In this paper, neither a stolen base from second to third base, nor a stolen base from third to home base, are considered.
6. $\boldsymbol{b} = (b_0, b_1)$ is made up of 2 components. $b_i \in \{1, 2, \ldots, 9\}$ indicates to which batter the batting order of team i rotates ($i = 0, 1$). It represents, when in offense ($\tau = i$), that the b_i-th batter steps up to the plate. When in defense ($\tau \neq i$), it means that the leadoff hitter in the next inning is the b_i-th batter.
7. $m \in \{0, 1\}$ is the index of the team which is on move. In this state, team m can choose their actions.

By the above definition, the initial state s_0 at the start time of the game is as follows:

$$s_0 = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m)_0 = (1, 0, 0, 0, (0, 0, 0), (1, 1), 1).$$

$\mathcal{S}_Q$ denotes the total states (absorbing states) at the end of the game:

$$\mathcal{S}_Q := \mathcal{S}_Q^0 \cup \mathcal{S}_Q^1 \cup \mathcal{S}_Q^{\mathrm{tie}} \cup \mathcal{S}_Q^{\mathrm{m}}(0) \cup \mathcal{S}_Q^{\mathrm{m}}(1) \subset \mathcal{S},$$

where

$$\begin{aligned}
\mathcal{S}_Q^0 &= \{(\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S} \,|\, \iota \geq 9,\ \tau = 1,\ \omega = 3,\ \lambda > 0\},\\
\mathcal{S}_Q^1 &= \{(\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S} \,|\, \iota = 9,\ \tau = 0,\ \omega = 3,\ \lambda < 0\}\\
&\quad \cup \{(\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S} \,|\, \iota \geq 9,\ \tau = 1,\ \lambda < 0\},\\
\mathcal{S}_Q^{\mathrm{tie}} &= \{(\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S} \,|\, \iota = 12,\ \tau = 1,\ \omega = 3,\ \lambda = 0\},\\
\mathcal{S}_Q^{\mathrm{m}}(0) &= \{(\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S} \,|\, \lambda \geq 30\}, \quad \text{(mercy-rule)}\\
\mathcal{S}_Q^{\mathrm{m}}(1) &= \{(\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S} \,|\, \lambda \leq -30\}. \quad \text{(mercy-rule)}
\end{aligned}$$

$\mathcal{S}_Q^0$ and $\mathcal{S}_Q^1$ correspond to a victory for team 0 and a victory for team 1, respectively. $\mathcal{S}_Q^{\mathrm{tie}}$ corresponds to a tie in the 12th inning after playing extra innings. We note that the mercy-rule refers to the establishment of a called game during the inning. By adopting both the NPB rules (i.e., the number of extra innings is finite) and the above mercy-rule, we get a finite Markov game. In addition, we equate the state s, such that $\omega = 3$ and $s \notin \mathcal{S}_Q$, with the corresponding state after the inning is over.

2.2. Actions

The manager of each team is the game-theoretic player maximizing the probability of their team winning. Here we let $\mathcal{S}_i$ be the set of states of moves for team i. Namely,

$$\mathcal{S}_i = \{(\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S} \setminus \mathcal{S}_Q \,|\, m = i\}, \quad i = 0, 1.$$

In $\mathcal{S}_0 \cup \mathcal{S}_1$, there are a lot of states that are not reachable from the initial state s_0. For example, $(\tau, r_1, b_0) = (0, 1, 2)$ is feasible, but $(0, 2, 1)$ is infeasible. We find out that the number of reachable states in $\mathcal{S}_0 \cup \mathcal{S}_1$ is $6,454,296$ by coding a computer program for counting them. In this paper, the action space is defined as

$$\mathcal{A} = \mathcal{A}^{\mathrm{defense}} \cup \mathcal{A}^{\mathrm{offense}},$$

where

$$\begin{aligned}
\mathcal{A}^{\mathrm{defense}} &= \{\text{pitching, intentional walk}\},\\
\mathcal{A}^{\mathrm{offense}} &= \{\text{batting, stolen base, sacrifice bunt}\}.
\end{aligned}$$

Let us consider a point-to-set valued mapping $\mathcal{A} : \mathcal{S} \setminus \mathcal{S}_Q \to 2^{\mathcal{A}} \setminus \{\phi\}$. $\mathcal{A}(s)$, called the feasible action space, represents the set of all actions in state s. In this paper, we define $\mathcal{A}(s)$, for any $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S} \setminus \mathcal{S}_Q$ in the following manner:

$$\begin{aligned}
&\text{pitching} \in \mathcal{A}(s) \Longleftrightarrow \tau \neq m,\\
&\text{intentional walk} \in \mathcal{A}(s) \Longleftrightarrow \tau \neq m,\\
&\text{batting} \in \mathcal{A}(s) \Longleftrightarrow \tau = m,\\
&\text{stolen base} \in \mathcal{A}(s) \Longleftrightarrow \tau = m, \quad r_2 = 0, \quad r_1 \geq 1,\\
&\text{sacrifice bunt} \in \mathcal{A}(s) \Longleftrightarrow \tau = m, \quad \omega \leq 1, \quad r_3 + r_2 + r_1 \geq 1.
\end{aligned}$$

Hence, stolen bases are feasible if and only if there is no second base runner and there is a runner present on first base, and sacrifice bunts are feasible if and only if a runner is present with 0 or 1 outs.

2.3. State transitions

For any $a \in \mathcal{A}$, let us define $\mathcal{X}(a)$, the set of all results that can occur stochastically when the action a is chosen, as follows:

$$\mathcal{X}(a) = \begin{cases} \{\text{game}\} & \text{if } a = \text{pitching}, \\ \{\text{walk}\} & \text{if } a = \text{intentional walk}, \\ \{\text{out, single, double, triple, home run, walk}\} & \text{if } a = \text{batting}, \\ \{\text{success, fail}\} & \text{otherwise.} \end{cases}$$

We denote the graph of $\mathcal{A}(\cdot)$ by $G_r(\mathcal{A})$. Namely,

$$G_r(\mathcal{A}) = \{(s,a) \,|\, a \in \mathcal{A}(s),\ \ s \in \mathcal{S}\}.$$

For any $(s,a) \in G_r(\mathcal{A})$ and any $x \in \mathcal{X}(a)$, $p(x|\, s,a)$ represents the conditional probability with which the result x occurs, given that action a is chosen in state s.

$$p(\cdot \,|\, s,a) : \mathcal{X}(a) \to [0,1], \quad \forall (s,a) \in G_r(\mathcal{A}),$$
$$\sum_{x \in \mathcal{X}(a)} p(x \,|\, s,a) = 1, \quad \forall (s,a) \in G_r(\mathcal{A}).$$

With our definition, the state $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m)$ includes information about the inning number, whether it is the top or bottom half, the out count, the run difference, the batting order of the batters, whether there are runners present or not, and also, the batting order of the runner if one is present on first base. Therefore, the transition probability generally depends on all of these states. However, in the numerical experiments carried out in Section 5 and Section 6, we assume that the transition probability depends only on the players that make a hit, or perform sacrifice bunts or stolen bases, and does not depend on those other components which comprise the state. Table 1 shows the probability parameters for the starting order of the Fukuoka Softbank Hawks, and was compiled based on the values achieved in Japan's professional baseball 2014 season [34, 35].

Table 1: Probability parameters

Name	AVG	Hitting						Stolen Base	Sacrifice Bunt
		Out	Single	Double	Triple	HR	Walk	Success	Success
1 Y. Honda	.291	.648	.217	.032	.016	.000	.087	.793	.941
2 A. Nakamura	.308	.627	.231	.035	.006	.006	.095	.833	.800*
3 Y. Yanagita	.317	.593	.211	.029	.007	.025	.136	.846	.000
4 S. Uchikawa	.307	.653	.199	.049	.002	.034	.063	.000	.000
5 Lee Dae-Ho	.300	.637	.195	.048	.000	.031	.090	.000	.000
6 Y. Hasegawa	.300	.624	.193	.056	.006	.011	.110	.500	.000
7 N. Matsuda	.301	.655	.185	.048	.007	.043	.062	.667	.500
8 S. Tsuruoka	.216	.750	.167	.024	.018	.000	.042	.000	.944
9 K. Imamiya	.240	.698	.174	.044	.002	.005	.077	.667	.873

* The minimum of 0.8 and the actual value was adopted for the sacrifice bunt success rate for players with extremely small numbers of attempted sacrifice bunts (less than 4) over the year.

This paper simplifies baseball in a similar manner to previous research. The simplifying rules used in this paper are as below.

"Simplifying rules"

1. With a mishit (an out), neither a batter nor a runner can advance bases.

2. A single advances a runner on first base to third base, and runners on second and third base reach the home plate.
3. A double and a triple allows all runners to reach the home plate.
4. It is assumed that there are no double plays.
5. For a successful stolen base, the runner on first base advances to second base.
6. For an unsuccessful stolen base, the runner on first base is out.
7. For a successful sacrifice bunt, the runners advance one base forward, and the batter performing the sacrifice bunt is out.
8. For an unsuccessful sacrifice bunt, the runner closest to the home plate is out, the other runners advance one base forward, and the batter is then the runner on first base.

If these simplifying rules are followed, then the next state s' is determined uniquely when in state s, action a is chosen, and result x occurs. We denote this next state by

$$s' = t(s, a, x).$$

Figure 1 illustrates the two-step transitions from a state with a runner on first with one out.

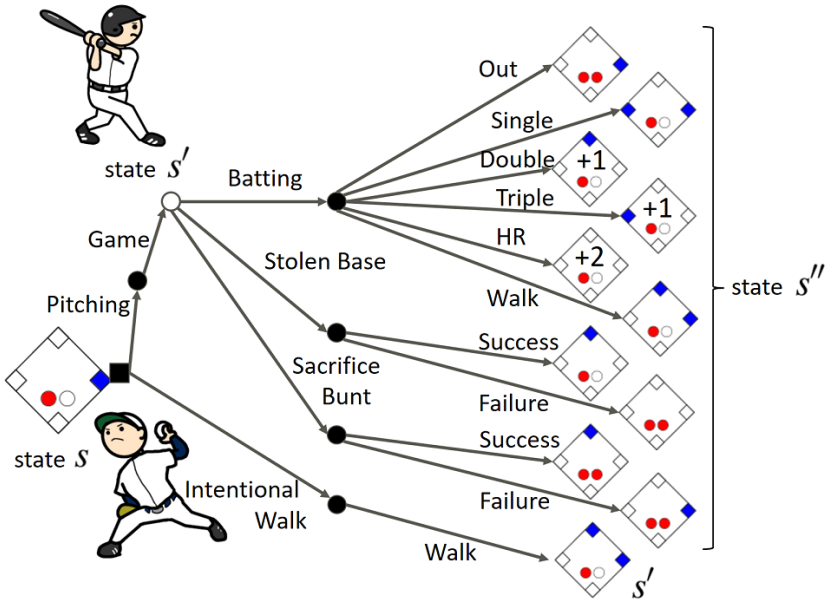


Figure 1: A part of the game tree

2.4. Markov policies

As a class of allowable policies, we consider the following class of Markov policies. The reason why we restrict our attention to this class will be stated in Section 3.

Definition 2.1 (Markov policy). *A mapping $\pi_i : \mathcal{S}_i \to \mathcal{A}$ is called a* (deterministic) *Markov policy for team i if $\pi_i(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}_i$ $(i = 0, 1)$. We denote the set of all deterministic Markov policies for team i by Π_i $(i = 0, 1)$.*

Suppose that Markov policy π_i is employed by team i. In this case, the Markov game commencing from each state s can be regarded as a Markov chain. In other words, if we let X_n be the state after n step transition from the initial state s_0, then $\{X_n\}$ is the Markov chain satisfying

$$\mathrm{P}^{\pi_0,\pi_1}(X_{n+1} = s' \,|\, X_n = s) = \begin{cases} p(x \,|\, s, \pi_i(s)) & \text{if } s \in \mathcal{S}_i,\ s' = t(s, \pi_i(s), x), \\ 1 & \text{if } s \in \mathcal{S}_Q,\ s' = s, \\ 0 & \text{otherwise,} \end{cases}$$

where P^{π_0,π_1} represents the conditional probability given that the policy π_i is employed by team i with $i = 0, 1$. Let T be the arrival time of $\{X_n\}$ to $\mathcal{S}_Q$. Namely,

$$T := \min\{n \,|\, X_n \in \mathcal{S}_Q\} < \infty.$$

We denote the probabilities of team i winning by $v_i(s; \pi_0, \pi_1)$.

$$v_i(s; \pi_0, \pi_1) := \mathrm{P}^{\pi_0,\pi_1}\left(X_T \in \mathcal{S}_Q^i \cup \mathcal{S}_Q^{\mathrm{m}}(i) \,|\, X_0 = s\right), \quad s \in \mathcal{S}, \quad (\pi_0, \pi_1) \in \Pi_0 \times \Pi_1, \quad i = 0, 1.$$

2.5. Payoff functions

For any Borel set $\mathcal{B}$ and any random variable X, we know the relation

$$\Pr(\, X \text{ is in } \mathcal{B}\,) = \mathrm{E}[\, \mathbf{1}_{\mathcal{B}}(X)\,], \tag{2.1}$$

where

$$\mathbf{1}_{\mathcal{B}}(X) = \begin{cases} 1 & \text{if } X \text{ is in } \mathcal{B}, \\ 0 & \text{otherwise}. \end{cases}$$

Thus, let us define team i's terminal payoff function $\psi_i : \mathcal{S}_Q \to \{0, 1\}$ as follows:

$$\psi_i(s) = \begin{cases} 1 & (-1)^i \lambda > 0, \\ 0 & \text{otherwise}, \end{cases} \qquad s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S}_Q, \quad i = 0, 1.$$

If the game is won, a payoff of 1 is acquired, whereas a loss or a tie is a payoff of 0. This value depends only on the current run difference λ. We note that

$$\psi_0(s) + \psi_1(s) = \begin{cases} 1 & \lambda \neq 0, \\ 0 & \lambda = 0, \end{cases} \qquad s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S}_Q.$$

Hence, our game is non-zero-sum game. Now, we can rewrite $v_i(s; \pi_0, \pi_1)$ as follows:

$$v_i(s; \pi_0, \pi_1) = \mathrm{E}^{\pi_0,\pi_1}\left[\psi_i(X_T) \,|\, X_0 = s\right], \quad s \in \mathcal{S}, \quad (\pi_0, \pi_1) \in \Pi_0 \times \Pi_1, \quad i = 0, 1,$$

where E^{π_0,π_1} represents the conditional expectation given that the policy π_i is employed by team i with $i = 0, 1$. The approach of using the relation (2.1) to reduce a probability criterion to the usual expectation criterion is often used in the field of MDPs (see Kira et al. [17]).

3. Lexicographic Markov Perfect Equilibria and Recursive Formula

In this section, we define lexicographic MPEs and the value functions of the game, and derive the recursive formula for effectively computing them.

An MPE is a profile of Markov policies that yields a Nash equilibrium in every proper subgame.

Definition 3.1 (Markov perfect equilibrium, MPE)**.** A profile of (deterministic) Markov policies (π_0^*, π_1^*) is called a (pure-strategy) MPE if it is a subgame perfect equilibrium. Namely, it satisfies

$$\begin{aligned} v_0(s; \pi_0, \pi_1^*) &\leq v_0(s; \pi_0^*, \pi_1^*), \quad \forall s \in \mathcal{S}, \quad \forall \pi_0 \in \Pi_0, \\ v_1(s; \pi_0^*, \pi_1) &\leq v_1(s; \pi_0^*, \pi_1^*), \quad \forall s \in \mathcal{S}, \quad \forall \pi_1 \in \Pi_1. \end{aligned}$$

Remark 3.1 (existence of the equilibria)**.** In the most general context, the action chosen by a policy in each state may be randomized. However, it is well-known that at least one pure-strategy MPE exists for a finite Markov game with perfect information (e.g., see Fundenberg and Tirole [9], Chap.13, p.516)*. We thus restrict our attention to the class of pure-strategy MPEs.

To guarantee the uniqueness of the equilibrium winning percentages for both teams, we introduce the following definition:

Definition 3.2 (lexicographic MPEs)**.** An MPE (π_0^*, π_1^*) is called a lexicographic MPE if it satisfies the following two conditions:

(i) For any $s \in \mathcal{S}$ and any $\pi_0 \in \Pi_0$ such that $v_0(s; \pi_0, \pi_1^*) = v_0(s; \pi_0^*, \pi_1^*)$,

$$v_1(s; \pi_0^*, \pi_1^*) \leq v_1(s; \pi_0, \pi_1^*).$$

(ii) For any $s \in \mathcal{S}$ and any $\pi_1 \in \Pi_1$ such that $v_1(s; \pi_0^*, \pi_1) = v_1(s; \pi_0^*, \pi_1^*)$,

$$v_0(s; \pi_0^*, \pi_1^*) \leq v_0(s; \pi_0^*, \pi_1).$$

Definition 3.3 (the value function of the game)**.** Let (π_0^*, π_1^*) be a lexicographic MPE, and for any state $s \in \mathcal{S}$, let $V_i(s)$ be the probability of team i winning when in state s. That is,

$$V_i(s) = v_i(s; \pi_0^*, \pi_1^*), \quad s \in \mathcal{S}.$$

Then the function V_i is called the value function of the game for team i.

Theorem 3.1 (Bellman equation)**.** The value functions and any lexicographic MPE (π_0^*, π_1^*) satisfy the following recursive formula:

$$V_i(s) = \begin{cases} \psi_i(s) & s \in \mathcal{S}_Q, \\ \displaystyle\max_{a \in \mathcal{A}(s)} \sum_{x \in \mathcal{X}(a)} V_i(t(s,a,x)) p(x|s,a) & s \in \mathcal{S}_i, \\ \displaystyle\sum_{x \in \mathcal{X}(\pi_j^*(s))} V_i(t(s, \pi_j^*(s), x)) p(x|s, \pi_j^*(s)) & s \in \mathcal{S}_j. \end{cases}$$

$$\pi_i^*(s) \in \arg\min_{a \in \mathcal{A}^*(s)} \sum_{x \in \mathcal{X}(a)} V_j(t(s,a,x)) p(x|s,a), \quad s \in \mathcal{S}_i,$$

$$\mathcal{A}^*(s) = \arg\max_{a \in \mathcal{A}(s)} \sum_{x \in \mathcal{X}(a)} V_i(t(s,a,x)) p(x|s,a), \quad s \in \mathcal{S}_i,$$

where $(i,j) = (0,1), (1,0)$.

Proof. As the initial condition for backward induction, we have

$$V_i(s) = \psi_i(s), \quad s \in \mathcal{S}_Q, \quad i = 0, 1.$$

Suppose that we are now in position to evaluate $V_0(s)$ and $V_1(s)$ for some state $s \in \mathcal{S}_0 \cup \mathcal{S}_1$, and suppose that we have evaluated $V_0(\cdot)$ and $V_1(\cdot)$ for all accessible states in one-step transition from s. If the team on move in the state s choses an action $a \in \mathcal{A}(s)$, and if

*This fact is an immediate consequence of the well-known Kuhn's theorem [18]. Kawasaki et al. [15] give another proof using a discrete fixed point theorem.

each team does their best in the subsequent subgame, then the winning percentages of both teams are

$$\sum_{x\in\mathcal{X}(a)} V_i(t(s,a,x))p(x|s,a), \quad i=0,1.$$

Therefore, any MPE (π_0^*, π_1^*) must satisfy

$$\pi_i^*(s) \in \mathcal{A}^*(s) = \arg\max_{a\in\mathcal{A}(s)} \sum_{x\in\mathcal{X}(a)} V_i(t(s,a,x))p(x|s,a),$$

where i is such that $s \in \mathcal{S}_i$. Furthermore, any lexicographic MPE (π_0^*, π_1^*) must satisfy

$$\pi_i^*(s) \in \arg\min_{a\in\mathcal{A}^*(s)} \sum_{x\in\mathcal{X}(a)} V_j(t(s,a,x))p(x|s,a), \tag{3.1}$$

where $j = 1 - i$. We thus obtain the result by backward induction. □

Remark 3.2 (existence of the lexicographic MPEs.)**.** The backword induction used in the proof of Theorem 3.1 guarantees the existence of at least one lexicographic MPE.

Remark 3.3 (uniqueness of the value functions)**.** The backword induction used in the proof of Theorem 3.1 also guarantees the uniqueness of the value functions. Namely, there exist multiple lexicographic MPEs if and only if the argument of the minimum in (3.1) is not a singleton for some state s, but all minimizers must result in the same probability of each team winning.

4. Dynamic Programming Algorithm

In the previous section, we have obtained the recursive formula satisfied by the optimal value functions and any lexicographic MPEs. By solving this, the optimal equilibrium strategies for each state, such as a sacrifice bunt or a stolen base, are obtained.

From the viewpoint of the theoretical framework of Markov games, the runs of team i may be treated as a reward system. In our model, to express the problem in the form of the usual expectation criterion, we store information about the runs scored as a component of the states. In the field of DP, such the state space $\mathcal{S}$ is called the enlarged (or augmented) state space. In the most general context of Markov games, the sum of rewards, for any state, earned up to that point depends on the history of the process. This indicates that the cardinality of the augmented state space increases exponentially with the length of the game. However, in our baseball model, it is sufficient and efficient to store the run difference λ and it only takes small integer values. This property is quite suitable for DP computation.

In the theory of finite Markov games (including finite MDPs), any general-purpose algorithm takes quadratic time with respect to the size of the state space in worst case. This time complexity is required because all the states must be evaluated and all the states may be accessible from all the states in one-step transition. However, in our baseball model, the number of all accessible states in one-step transition from any state s can be counted on both hands (See Figure 1). This indicates that we can construct a specialized algorithm which takes linear time with the size of the state space. We realize it by the use of memoized recursion. The algorithm can be implemented as described in Algorithm 1.

Data: an instance of the transition probabilities
$p(\cdot \mid s, a) : \mathcal{X}(a) \to [0, 1],\ (s, a) \in G_r(\mathcal{A})$.
Result: the optimal value functions V_0 and V_1 and a pure-strategy lexicographic MPE $\pi^* = (\pi_0^*, \pi_1^*)$
Initialize $V_0(s)$ and $V_1(s)$ to -1 for all $s \in \mathcal{S}$;
$s_0 = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m)_0 \leftarrow (1, 0, 0, 0, (0, 0, 0), (1, 1), 1)$;
Call Evaluate(arguments: s_0);

Algorithm 1: A dynamic programming algorithm for solving the baseball game

The memoized recursive function Evaluate(parameters: $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S}$), which evaluates $V_i(s)$ with $i = 0, 1$, can be implemented as follows:

if $s \in \mathcal{S}_Q$ **then**
  $V_i(s) \leftarrow \psi_i(s)$ for $i = 0, 1$;
else
  Declare local variables: an integer j, a floating-point variable *temp*, a state s', and a set of actions $\mathcal{A}^*$;
  $j \leftarrow 1 - m$;
  forall the $a \in \mathcal{A}(s)$ **do**
    $temp \leftarrow 0$;
    forall the $x \in \mathcal{X}(a)$ such that $p(x|s, a) > 0$ **do**
      $s' \leftarrow$ Call Transition(arguments: s, a, x);
      if $V_m(s') = -1$ **then**
        Call Evaluate(arguments: s');
      $temp \leftarrow temp + V_m(s')p(x|s, a)$;
    if $V_m(s) < temp$ **then**
      $V_m(s) \leftarrow temp, \quad \mathcal{A}^* \leftarrow \{a\}$;
    else if $V_m(s) = temp$ **then**
      $\mathcal{A}^* \leftarrow \mathcal{A}^* \cup \{a\}$;
  forall the $a \in \mathcal{A}^*$ **do**
    $temp \leftarrow 0$;
    forall the $x \in \mathcal{X}(a)$ such that $p(x|s, a) > 0$ **do**
      $s' \leftarrow$ Call Transition(arguments: s, a, x);
      $temp \leftarrow temp + V_j(s')p(x|s, a)$;
    if $V_j(s) = -1$ or $temp < V_j(s)$ **then**
      $V_j(s) \leftarrow temp, \quad \pi_m^*(s) \leftarrow a$;

Function Evaluate(parameters: $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S}$)

This memoized recursion solves the recursive formula just for reachable states from the initial state by implementing the depth-first search of the game tree. The function Transition(parameters: s, a, x) returns the next state $s' = t(s, a, x)$, and can be implemented as follows:

switch the value of a **do**
  case pitching
    (Do nothing)
  case intentional walk or batting
    $s \leftarrow$ Call TransBatting(arguments: s, x);
  case stolen base
    $s \leftarrow$ Call TransStolenBase(arguments: s, x);
  case sacrifice bunt
    $s \leftarrow$ Call TransSacrificeBunt(arguments: s, x);
if $\omega = 3$, $s \notin \mathcal{S}_Q$ **then**
  $s \leftarrow$ Call InningIsOver(arguments: s);
else if $a \neq$ intentional walk **then**
  $m \leftarrow 1 - m$;
return s;

Function Transition(parameters: $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S}$, $a \in \mathcal{A}(s)$, $x \in \mathcal{X}(a)$)

switch the value of x **do**
  case out
    $\omega \leftarrow \omega + 1$;
  case single
    $\lambda \leftarrow \lambda + (-1)^\tau (r_3 + r_2)$, $r_3 \leftarrow \mathbf{1}_{>0}(r_1)$, $r_2 \leftarrow 0$, $r_1 \leftarrow b_\tau$;
  case double
    $\lambda \leftarrow \lambda + (-1)^\tau (r_3 + r_2 + \mathbf{1}_{>0}(r_1))$, $r_3 \leftarrow 0$, $r_2 \leftarrow 1$, $r_1 \leftarrow 0$;
  case triple
    $\lambda \leftarrow \lambda + (-1)^\tau (r_3 + r_2 + \mathbf{1}_{>0}(r_1))$, $r_3 \leftarrow 1$, $r_2 \leftarrow 0$, $r_1 \leftarrow 0$;
  case home run
    $\lambda \leftarrow \lambda + (-1)^\tau (r_3 + r_2 + \mathbf{1}_{>0}(r_1) + 1)$, $r_3 \leftarrow 0$, $r_2 \leftarrow 0$, $r_1 \leftarrow 0$;
  case walk
    if $r_3 = r_2 = \mathbf{1}_{>0}(r_1) = 1$ **then**
      $\lambda \leftarrow \lambda + (-1)^\tau$;
    else
      Declare a local integer variable n;
      $n \leftarrow \min\{j \in \{1, 2, 3\} \mid r_j = 0\}$;
      if $n \geq 2$ **then**
        $r_n \leftarrow 1$;
    $r_1 \leftarrow b_\tau$;
$b_\tau \leftarrow b_\tau \mod 9 + 1$;
return s;

Function TransBatting(parameters: $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S}$, $x \in \mathcal{X}$(batting))

switch the value of x **do**
  case success
    $r_2 \leftarrow 1$;
  case fail
    $\omega \leftarrow \omega + 1$;
$r_1 \leftarrow 0$;
return s;

Function TransStolenBase(parameters: $s \in \mathcal{S}$, $x \in \mathcal{X}$(stolen base))

switch the value of x **do**
  case success
    $\lambda \leftarrow \lambda + (-1)^{\tau} r_3, \quad r_3 \leftarrow r_2, \quad r_2 \leftarrow \mathbf{1}_{>0}(r_1), \quad r_1 \leftarrow 0;$
  case fail
    Declare a local integer n and set $n \leftarrow \max\{i \,|\, r_i > 0\}$;
    if $n = 3$ **then**
      $r_3 \leftarrow r_2, \quad r_2 \leftarrow \mathbf{1}_{>0}(r_1)$;
    else if $n = 2$ **then**
      $r_2 \leftarrow \mathbf{1}_{>0}(r_1)$;
    $r_1 \leftarrow b_{\tau}$;
$\omega \leftarrow \omega + 1, \quad b_{\tau} \leftarrow b_{\tau} \mod 9 + 1;$
return s;

Function TransSacrificeBunt(parameters: $s \in \mathcal{S}$, $x \in \mathcal{X}$(sacrifice bunt))

if $\tau = 0$ **then**
  $\tau \leftarrow 1$;
else
  $\iota \leftarrow \iota + 1, \quad \tau \leftarrow 0;$
$\omega \leftarrow 0, \quad r_3 \leftarrow 0, \quad r_2 \leftarrow 0, \quad r_1 \leftarrow 0;$
return s;

Function InningIsOver(parameters: $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) \in \mathcal{S}$)

5. Advantage of The Last-Batting Team

In baseball, there is often talk of whether the last-batting team has an advantage. In Japanese professional baseball games, the visiting team bats first and the home team bats second. The wins and losses for home and visiting teams in the 2014 season are shown in Table 2.

Table 2: Win-loss records by home/road (2014 season)

TEAM	G	HOME				ROAD			
		W	L	D	PCT	W	L	D	PCT
Giants	144	44	27	1	.611	38	34	0	.528
Tigars	144	41	30	1	.569	34	38	0	.472
Carp	144	42	29	1	.583	32	39	1	.444
Dragons	144	35	35	2	.486	32	38	2	.444
Baystars	144	34	37	1	.472	33	38	1	.458
Swallows	144	32	39	1	.444	28	42	2	.389
Hawks	144	45	24	3	.625	33	36	3	.458
Buffaloes	144	46	26	0	.639	34	36	2	.472
Fighters	144	43	29	0	.597	30	39	3	.417
Marines	144	37	33	2	.514	29	43	0	.403
Lions	144	31	38	3	.431	32	39	1	.444
Eagles	144	30	42	0	.417	34	38	0	.472
Total		460	389	15	.532	389	460	15	.450

$\mathrm{PCT} := W/(\mathrm{W} + \mathrm{L} + \mathrm{D})$

Source: Nippon Professional Baseball Official Website [34]

In total, 864 games were played in the Central and Pacific leagues combined. The winning percentage for teams batting first was .450 and the winning percentage for teams batting last was .532, approximately 8 % higher. There are various advantages to being able to play a game at home, such as support from the home crowd. However, is there an advantage caused strictly by baseball rules?

Turocy [28] argued for the advantage of batting last by calculating the value of the game for the hypothetical situation where the same team plays itself. When doing this, the strategies that can be chosen by the manager are base stealing, sacrifice bunting, and intentionally walking a batter. These strategies can be turned "ON" or "OFF," and the value of the game is compared over a total of 8 different situations. In our paper, we have the Fukuoka Softbank Hawks (shown as in Table 1) play against themselves. We also switched each manager plan ON and OFF, both for the team batting first and the team batting last, and evaluated the value of the game at the point of the game starting in a total of 64 situations. We show the results in Table 3.

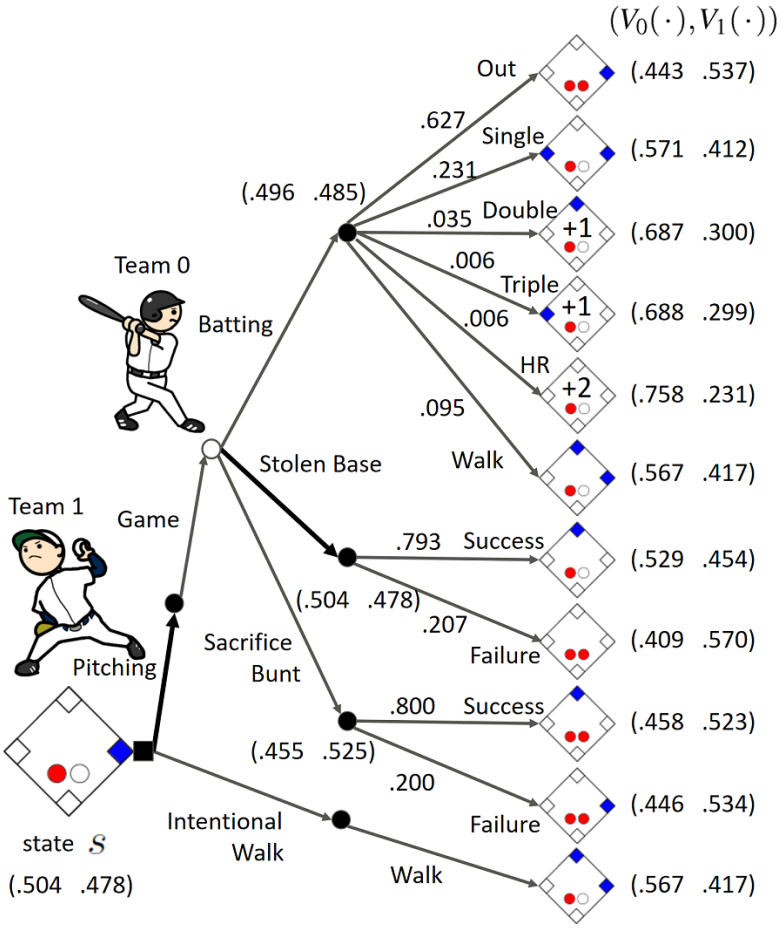
Table 3: Values of the games and the effects of the strategies

Batting-last / Batting-first	–, –, –		–, –, W		–, B, –		–, B, W		S, –, –		S, –, W		S, B, –		S, B, W	
–, –, –	.4932	.4932	.4930	.4933	.4699	.5180	.4698	.5181	.4690	.5190	.4689	.5191	.4458	.5437	.4456	.5438
–, –, W	.4935	.4929	.4933	.4930	.4713	.5167	.4712	.5168	.4694	.5186	.4692	.5187	.4475	.5417	.4473	.5419
–, B, –	.5167	.4710	.5160	.4715	.4933	.4957	.4926	.4962	.4925	.4966	.4918	.4971	.4689	.5213	.4682	.5219
–, B, W	.5170	.4706	.5163	.4711	.4947	.4944	.4940	.4949	.4929	.4961	.4922	.4967	.4707	.5195	.4700	.5199
S, –, –	.5177	.4700	.5176	.4701	.4942	.4949	.4941	.4950	.4934	.4957	.4933	.4958	.4698	.5206	.4697	.5207
S, –, W	.5180	.4696	.5179	.4697	.4956	.4935	.4955	.4936	.4938	.4953	.4937	.4954	.4715	.5187	.4714	.5188
S, B, –	.5409	.4480	.5402	.4485	.5174	.4727	.5167	.4732	.5167	.4734	.5160	.4739	.4929	.4983	.4922	.4988
S, B, W	.5412	.4477	.5405	.4481	.5188	.4714	.5181	.4718	.5171	.4730	.5164	.4735	.4947	.4964	.4940	.4968

- The left column displays the value of the game for the first-batting team, the right column for the last-batting team
- S = Base stealing is "ON", B = Sacrifice bunting is "ON", W = Intentional Walk is "ON"

We implemented our DP algorithm using C++ Language and executed it on a desktop PC with Intel® Core™ i7-3770K processor and 16GB memory installed. Computational time is longest when all strategies for both teams are ON. In this case, the calculation

for a lexicographic MPE and the values of the game was completed in 1.41 second per game. Figure 2 illustrates the computational result of $V_i(s)$ for $s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) = (7, 0, 1, 0, (0, 0, 1), (2, 1), 1)$.



$$s = (\iota, \tau, \omega, \lambda, \boldsymbol{r}, \boldsymbol{b}, m) = (7, 0, 1, 0, (0, 0, 1), (2, 1), 1)$$

- all strategies for both teams are ON.

Figure 2: Computational result: $V_i(s)$ for some state s

For the condition of only intentional walks being ON for both teams, the team batting first had a higher winning percentage. On the other hand, for the condition of only base stealing being ON for both teams and for the condition of only sacrifices being ON for both teams, the winning percentage of the team batting last was higher. Thus, intentional walks are most advantageous to the team batting first, while stolen bases and sacrifice bunting are most advantageous to the team batting last. Therefore, when the strategies of both teams are all switched ON, whether the winning percentage of the team batting first or of the team batting last is higher depends on the transition probabilities of chance moves. However, as Table 3 shows, the influence of walks is less than that of base stealing and sacrifice bunting. Thus, it seems safe to say that, normally, the winning percentage of the team batting last would be higher. These results correspond to the facts outlined by Turocy [28]. However, Turocy does state in his paper that "the disparity is slight, and does not make a big difference." Although this is true, when considering the fact that a .007

increase in winning percentage would amount to 1 additional win in a 144 game season, this difference should not be ignored.

We note that optimal decision also depends on the current run difference. Take a state with runner on first base with no outs in the bottom half of the final inning, for instance. Sacrifice bunt will probably maximize team 1's win percentage when the score is tied. However, in the case of 3 runs behind, sacrifice bunt should not be executed. Hence, the run difference between the opposing teams must be considered to win a game. In other words, the number of runs scored by the opposing team is an important piece of information. When the first-batting team makes decisions on batting for a hit, stealing a base, or sacrifice bunting (i.e., when in the top half of some inning), the manager can not know the runs scored in the bottom half of the inning by the last-batting team. However, when the last-batting team makes such decisions (i.e., when in the bottom half of some inning), the manager can know the runs scored in the top half of the inning by the first-batting team. Therefore, the last-batting team has an advantage of a half inning more observation. On the other hand, when making decisions on intentional walks, the stands of the first-batting team and the last-batting team is reversed. We believe this asymmetry due to the rules of baseball is the reason for the slight difference in winning percentage between the team batting first and that batting last.

6. Optimal Lineup

Since the paper by Bukiet et al. [6] was published, the hottest topic within research on Markov chain approaches with matrix analysis has been the calculation of optimal batting order. This topic has been addressed in many previous studies. In a DP approach, considerable computational time is required for the optimization of strategy itself; thus, the computational cost of completing an exhaustive search of batting lineup to find the optimal one is very high. However, in actuality, it is enough to search 8! permutations. A memoized value of a subgame can be reused for another, so the time taken to evaluate a single lineup $\sigma = (1, 2, 3, 4, 5, 6, 7, 8, 9)$ is nearly the same as the time taken to evaluate all 9 lineups that can be obtained by rotations (e.g., $\sigma' = (2, 3, 4, 5, 6, 7, 8, 9, 1)$, $\sigma'' = (3, 4, 5, 6, 7, 8, 9, 1, 2)$).

Tables 4 and 5 show our results for computing the optimal lineup for the Fukuoka Softbank Hawks. In this case, we also created a hypothetical game where the first-batting team and the last-batting team are the exact same team. The batting lineup for the first team was fixed as the default batting order, shown in Table 1. An exhaustive search was then conducted for the batting order of the last-batting team, and we found the optimal lineup and the worst lineup which maximizes and minimize the winning percentage of the last-batting team, respectively. Moreover, we changed the run difference established in a called game from 30 runs to 20 runs, because it was sufficient to be able to precisely calculate the values of the games at the start of the game. Since we conducted a simple exhaustive search on a single thread, the computation took approximately 10 hours.

Table 4: Optimal lineup

Default Lineup	Worst Lineup	Optimal Lineup
1 Y. Honda	1 S. Tsuruoka	1 A. Nakamura
2 A. Nakamura	2 Y. Hasegawa	2 Y. Yanagita
3 Y. Yanagita	3 K. Imamiya	3 S. Uchikawa
4 S. Uchikawa	4 Y. Honda	4 Lee Dae-Ho
5 Lee Dae-Ho	5 N. Matsuda	5 Y. Hasegawa
6 Y. Hasegawa	6 S. Uchikawa	6 N. Matsuda
7 N. Matsuda	7 Lee Dae-Ho	7 Y. Honda
8 S. Tsuruoka	8 A. Nakamura	8 S. Tsuruoka
9 K. Imamiya	9 Y. Yanagita	9 K. Imamiya

Table 5: Probability of the last-batting team winning

Batting-last / Batting-first	Default Lineup		Worst Lineup		Optimal Lineup	
Default Lineup	.4940	.4968	.5140	.4765	.4909	.5000

- The left column displays the value of the game for the first-batting team, the right column for the last-batting team

With the assumption that both teams make the best choices in terms of their game decision making on batting for a hit, stealing a base, or sacrifice bunting, the winning percentage difference between the optimal and worst batting order was only 2.35 %. However, in the context of a 144 game regular season, this would amount to a difference of 3.39 wins. Considering that more wins equate to more losses for other teams, the game difference with the other teams would be even greater.

7. Related Work

A generalization adding double play and sacrifice fly to our model is studied by Nakamura [20]. He analyzed detailed data of NPB provided by Data Stadium Inc., and won the excellence award in the 4th Sports Data Analysis Competition. Nishizawa [21] analyzes pinch hitting strategies under certain assumptions. In addition, he suggests using 2-opt to obtain a suboptimal batting order.

Maki et al. [19] propose an interesting approach to evaluate batting orders by simulations on video baseball games. They compare batting orders obtained by typical optimization approaches, and show the efficiency of the lineups provided by our model.

8. Summary

In this paper, baseball has been formulated as a finite non-zero-sum Markov game with approximately 6.45 million states. We demonstrated that the value functions of the games and lexicographic MPEs, where both teams' managers maximize the probabilities of their respective team winning, can be computed in less than 2 second per game. Based on our model and the perspective of the value of information in game theory, we have explained that intentional walks are most advantageous to the team batting first, while stolen bases and sacrifice bunting are most advantageous to the team batting last. Furthermore, we have successfully computed the optimal batting order, in consideration of strategy optimization such as a sacrifice bunt or a stolen base.

Acknowledgments

The authors wish to thank Professor Seiichi Iwamoto and Hidefumi Kawasaki for their valuable advice regarding this investigation. We are also grateful to Dr. Kotaro Ohori for his constant support. This research was supported in part by JSPS KAKENHI Grant Numbers 26730010, 17K12644, 15K01193, and 15K05004.

References

[1] K. Ano: Modified offensive earned-run average with steal effect for baseball. *Applied Mathematics and Computation*, **120** (2001), 279–288.

[2] R. Bellman: Dynamic Programming and Markovian Decision Processes, with Application to Baseball. In S.P. Ladany and R.E. Macol (eds.): *Optimal Strategies in Sports* (Elsevier-North Holland, New York, 1977), 77–85.

[3] K. Boda, J.A. Filar, Y. Lin, and L. Spanjers: Stochastic target hitting time and the problem of early retirement. *IEEE Transactions on Automatic Control*, **49-3** (2004), 409–419.

[4] K. Boda and J.A. Filar: Time consistent dynamic risk measures. *Mathematical Methods of Operations Research*, **63** (2006), 169–186.

[5] M. Bouakiz and Y. Kebir: Target-level criterion in Markov decision processes. *Journal of Optimization Theory and Applications*, **86**, (1995), 1–15.

[6] B. Bukiet, E.R. Harold, and J.L. Palacios: A Markov Chain Approach to Baseball. *Operations Research*, **45-1** (1997), 14–23.

[7] T.M. Cover and C.W. Keilers: An Offensive Earned-Run Average for Baseball. *Operations Research*, **25-5** (1977), 729–740.

[8] D.A. D'Esopo and B. Lefkowitz, The Distribution of Runs in the Game of Baseball, In S.P. Ladany and R.E. Macol(eds.): *Optimal Strategies in Sports* (Elsevier North-Holland, 1977), 55–62.

[9] D. Fudenberg and J. Tirole: *Game Theory* (MIT Press, Cambridge MA, 1991).

[10] N. Hirotsu and M. Wright: A Markov chain approach to optimal pinch hitting strategies in a designated hitter rule baseball game. *Journal of the Operations Research Society of Japan*, **46-3** (2003), 353–371.

[11] N. Hirotsu and M. Wright: Modeling a baseball game to optimize pitcher substitution strategies using dynamic programming. In: S. Butenko, J. Gil-Lafuente, and P.M. Pardalos (eds.): *Economics, Management and Optimization in Sports*, (Springer, Berlin, Heidelberg, 2004), 131–161.

[12] N. Hirotsu and M. Wright: Modelling a baseball game to optimise pitcher substitution strategies incorporating handedness of players. *IMA Journal of Management Mathematics*, **16-2** (2005), 179–194.

[13] R.A. Howard: *Dynamic Programming and Markov Processes* (M.I.T. Technology Press and Wiley, Cambridge, Mass, 1960).

[14] S. Iwamoto, T. Ueno, and T. Fujita: Controlled Markov chains with utility functions. In Z. Hou, J.A. Filar, and A. Chen (eds.): *Markov Processes and Controlled Markov Chains* (Springer US, 2002), 135-149.

[15] H. Kawasaki, A. Kira, and S. Kira: An application of a discrete fixed point theorem to a game in expansive form. *Asia-Pacific Journal of Operational Research*, **30** (2013), No. 3.

[16] A. Kira, and K. Inakawa: On Markov perfect equilibria in baseball. *Bulletin of Informatics and Cybernetics*, **46** (2014), 11–21.

[17] A. Kira, T. Ueno, and T. Fujita: Threshold probability of non-terminal type in finite horizon Markov decision processes. *Journal of Mathematical Analysis and Applications*, **386**, (2012), 461–472.

[18] H.W. Kuhn, Extensive games and the problem of information, In H.W. Kuhn and A.W. Tucker (eds.): *Contributions to the Theory of Games*, vol. **2** (*Annals of mathematics studies*, **28**) (Princeton University Press. Princeton, 1953), 193–216.

[19] T. Maki, T. Hanaka, and H. Ono: Evaluating batting orders via video baseball game simulations. *IPSJ SIG Technical Report*, (2016), 7 pages. (in Japanese)

[20] T. Nakamura: A simulation of baseball games by Markov games. Bachelor Thesis, (Supervisor: T, Matsui), Department of Social Engineering, School of Engineering, Tokyo Institute of Technology, (2015), 27 pages. (in Japanese)

[21] G. Nishizawa: Analysing pinch hitting strategies in baseball by Markov games. Bachelor Thesis, (Supervisor: T, Matsui), Department of Social Engineering, School of Engineering, Tokyo Institute of Technology, (2017), 29 pages. (in Japanese)

[22] Y. Ohtsubo: Optimal threshold probability in undiscounted Markov decision processes with a target set. *Applied Mathematics and Computation*, **149** (2004), 519–532.

[23] Y. Ohtsubo and K. Toyonaga: Optimal policy for minimizing risk models in Markov decision processes. *Journal of Mathematical Analysis and Applications*, **271** (2002), 66–81.

[24] M. Sakaguchi and Y. Ohtsubo: Optimal threshold probability and expectation in semi-Markov decision processes. *Applied Mathematics and Computation*, **126** (2010), 2947–2958.

[25] L.S. Shapley, Stochastic games, *Proceedings of the National Academy of Sciences of the United States of America*, **39** (1953), 1095-1100.

[26] M. Sniedovich: *Dynamic Programming: Foundations and Principles*, 2nd edn. (CRC Press, 2010).

[27] M.J. Sobel: The variance of discounted Markov decision processes. *Journal of Applied Probability*, **19** (1982), 794–802.

[28] T.L. Turocy: In search of the "last-ups" advantage in baseball: A game-theoretic approach. *Journal of Quantitative Analysis in Sports*, **4-2** (2008), Article 5.

[29] D.J. White: Mean, variance, and probabilistic criterion in finite Markov decision processes: A review. *Journal of Optimization Theory and Applications* **56** (1988), 1–29.

[30] D.J. White: Minimizing a threshold probability in discounted Markov decision processes. *Journal of Mathematical Analysis and Applications*, **173** (1993), 634–646.

[31] C. Wu and Y. Lin: Minimizing risk models in Markov decision processed with policies depending on target values. *Journal of Mathematical Analysis and Applications*, **231** (1999), 47–67.

[32] H. Xu, and S. Mannor: Probabilistic goal Markov decision processes. In T. Walsh (ed.): *Proceedings of the 22nd International Joint Conference on Artificial Intelligence* (IJCAI-11), (2011), 2046-2052.

[33] L.E. Zachrisson: Markov games. In M. Drescher, L.S. Shapley and A.W. Tucker (eds.): *Advances in Game Theory* (*Annals of Mathematical Studies*, **52**) (Princeton University Press, Princeton, 1964), 211-254.

[34] Nippon Professional Baseball Official Website, ⟨http://www.npb.or.jp/eng/⟩, Accessed 2018 Sep 25.

[35] Let's enjoy baseball data! (in Japanese), ⟨http://baseballdata.jp/⟩, Accessed 2018 Sep 25.

Akifumi Kira
Faculty of Social and Information Studies
Gunma University
4-2 Aramaki-machi
Gunma 371-8510, Japan
E-mail: a-kira@si.gunma-u.ac.jp