Solving the Game of Darts

Chun Wang
Department of Management Science and Engineering
School of Economics and Management, Tsinghua University
wangch5@sem.tsinghua.edu.cn

January 7, 2021

1 The Rules of Darts

In order to understand the game of darts we need to understand how a single dart scores. A dartboard is displayed in Figure 1 and depending on where a dart lands on the board, a particular score is realized. The small concentric circles in the middle of the board define the "double bulls-eye" (DB) and the "single bulls-eye" (SB) regions. If a dart lands in the DB region (the small red circle) then it scores 50 points while a dart landing in the SB region (the green annulus surrounding the DB region) scores 25.

Beyond the SB region the dartboard is divided into 20 segments of equal area corresponding to the numbers 1 to 20. If a dart lands in the "20" segment, for example, then it will land in the treble twenty (T20) region, the double twenty region (D20) or the single twenty region (S20) for scores of 60, 40 or 20, respectively. The double region is the region between the two outermost circles on the dartboard whilst the treble region is the region between the two circles beyond the SB region. The single region is then the union of the two disjoint regions between the SB and treble region and between the treble and double regions. If a dart lands beyond the double region then it scores zero.

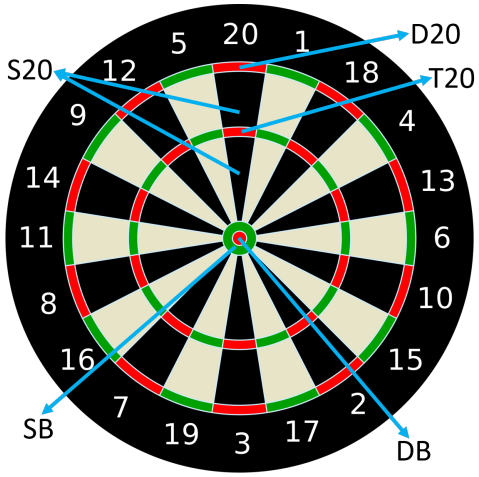


Figure 1: A standard dartboard

The rules of darts that we describe here[1] are for the most commonly played form of the game, namely "501". This is the form of the game typically played in pubs and in most professional tournaments. In 501, each player starts on a score of 501 and takes turns in throwing three darts. These three darts constitute a "turn". After a player's turn his score on the three darts are added together and subtracted from his score at the beginning of the turn. The first player to reach a score of *exactly* zero wins the game. In order to win, however, a player's final dart must be a double,

[1]The rules for 501 and other less common forms of the game are described at https://www.mastersofgames.com/rules/darts-rules.htm. Furthermore the geometry of the dartboard, i.e. the distances from the center to the various rings on the dartboard, are provided in Appendix A.

i.e. it must land on D1, D2, ..., D20 or DB. If a player's turn would result in an updated score of 1 or a negative score, then the turn is invalidated, the player remains on the score he had prior to the turn and his opponent then takes his turn. Note that it's possible for a player to win without having to throw all three darts in his turn. For example suppose a player has a score of 20 just prior to his turn. If he then scores D10 with the first dart of his turn he wins. Alternatively if he missed D10 with his first dart and scored S10, then he could still win with the second dart of his turn by throwing a D5.

The game we have just described is known as a "leg" and in practice darts matches are typically played over many legs with the winner being the first to win some fixed number of legs. Alternatively, some tournaments have a legs and "sets" structure whereby the winner of the match is the first to win a fixed number of sets and the winner of a set is the first to win a fixed number of legs. This latter structure then is similar to other sports such as tennis, for example. Finally, we note that because the player that throws first has an advantage, players alternate in starting legs (and sets). For example, if player A begins the first leg of the first set, then player 2 will begin the first leg of the second set etc. Similarly within a set, players alternate in starting each leg. If a match is only played over legs then the players alternate in starting legs.

Regardless of the match structure, however, the key component of a darts match is the leg and it should be clear that a player will maximize his chances of winning a match if he optimizes his strategy for playing each leg.

2 A Mathematical Model for Dart Throwing

Taking the center of the DB region to be the origin, we will use $\boldsymbol{\mu} \in \mathbb{R}^2$ to denote[2] the target of the dart throw and $(x, y) \in \mathbb{R}^2$ the outcome of the throw. We then let $p(x, y; \boldsymbol{\mu})$ denote the probability of the outcome (x, y) given the target $\boldsymbol{\mu}$. Given $p(x, y; \boldsymbol{\mu})$ it will be easy to compute the distribution $p(z; \boldsymbol{\mu})$ where $z = g(x, y)$ maps $(x, y) \in \mathbb{R}^2$ to the dart score z, e.g. D16, SB, T20, S7 etc. We will also let $h(z) \in \mathbb{N}$ denote the actual numerical score achieved by the dart so for example, $h(\text{D16}) = 32$, $h(\text{T20}) = 60$, $h(\text{SB}) = 25$, $h(\text{S7}) = 7$ etc. Our goal then is to construct a model for $p(x, y; \boldsymbol{\mu})$ that can also be easily estimated with the available data.

2.1 A Simple Gaussian Model

Tibshirani et al. (2011) (hereafter TPT) proposed several models for capturing the skill of a darts player. Their first model assumed

$$[x\ y]^\top \sim \text{N}_2(\boldsymbol{\mu}, \sigma^2 \mathbf{I}_2) \tag{1}$$

where $\text{N}_2(\boldsymbol{\mu}, \sigma \mathbf{I}_2)$ denotes the bivariate Gaussian distribution with mean $\boldsymbol{\mu}$ (the intended target) and covariance matrix $\sigma^2 \mathbf{I}_2$ where $\mathbf{I}_2 \in \mathbb{R}^{2,2}$ is the identity matrix. They also proposed the more general model

$$[x\ y]^\top \sim \text{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2}$$
$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_x^2 & \sigma_{x,y} \\ \sigma_{x,y} & \sigma_y^2 \end{pmatrix}.$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{2,2}$ is an arbitrary covariance matrix. TPT proposed the use of the EM algorithm to estimate the parameters σ and $\boldsymbol{\Sigma}$ of (1) and (2), respectively. They assumed the intended target $\boldsymbol{\mu}$ was known for each data-point but that only the result $z = g(x, y)$ was observed rather than the realized location (x, y). That only the z's were observed seems like a reasonable assumption given the difficulty of measuring (x, y)-coordinates of dart throws and indeed this is also true of the data-set that we use in this paper.

[2] In all of our calculations we represent dartboard measurements on a scale of millimeters.

2.2 A Conditional Gaussian Model

A feature of professional darts players is that simple models like (1) and (2) are unlikely to be sufficiently rich for modeling their skills. This is because professional darts players tend to focus on (and practice throwing at) specific parts of the darts board, e.g. T20, T19, DB etc. This means that their skill levels, as determined by σ^2 or $\boldsymbol{\Sigma}$, is likely to be a function of $\boldsymbol{\mu}$. Indeed this is what we observe in the data. For example, the Dutch player Michael van Gerwen[3] is successful 45.3% of the time when targeting T20 but only 30.2% of the time when targeting T17.

In light of the real data that we do have, it therefore seems appropriate to assume there exists a finite number of regions of the board, say $R_1, \ldots, R_M$, so that (2) can then be generalized to

$$[x\ y]^\top \quad \sim \quad \mathrm{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma}_m), \quad \text{for } \boldsymbol{\mu} \in R_m. \tag{3}$$

We note that (3) is as easy to estimate as (2) as we can partition our data-set by regions with each region consisting of one or more target regions. In this project, we chose $M = 7$ with $R_1 = \{\mathrm{T20}\}$, $R_2 = \{\mathrm{T19}\}$, $R_3 = \{\mathrm{T18}\}$, $R_4 = \{\mathrm{T17}\}$, $R_5 = \{\mathrm{DB}\}$, $R_6 = \{\mathrm{D1}, \ldots, \mathrm{D20}\}$ and $R_6 = \{\text{others}\}$.

3 The Dynamic Programming Formulation for the Race to Zero

Once we have a model $p(x, y; \boldsymbol{\mu})$ we can then easily compute $p(z; \boldsymbol{\mu})$, the corresponding distribution over scores and $p(h(z); \boldsymbol{\mu})$, the corresponding distribution over numerical scores. With these distributions we are in a position to formulate and solve various dynamic programs.

3.1 Ignoring the "Turn" Feature

An interesting problem in its own right is the problem whereby a player aims to minimize the expected number of throws to get to zero. Such a player pays no attention to the score of his opponent and is therefore non-strategic. It can be formulated as a stochastic shortest path DP and we refer to it as the *non-strategic* (NS) problem. To the best of our knowledge Kohler (1982) was the first[4] to solve this problem.

The state space for this DP is (s, i) where $s \in \{1, \ldots, D\}$ denotes the player's current score (typically $D = 501$) and i denotes how many throws have already been taken. The Bellman equation for this DP satisfies

$$V_{\mathrm{NS}}(s_i, i) = \min_{\boldsymbol{\mu}_{i+1}} \mathrm{E}\left[V(s_{i+1}, i+1) \mid (s_i, i), \boldsymbol{\mu}_{i+1}\right] \tag{4}$$

$$= \min_{\boldsymbol{\mu}_{i+1}} \sum_z V_{\mathrm{NS}}(f(s_i, z), i+1)\, p(z;\, \boldsymbol{\mu}_{i+1}) \tag{5}$$

where $V_{\mathrm{NS}}(s, i)$ is the value function when the current score is s and i darts have already been thrown. We have the terminal condition $V_{\mathrm{NS}}(0, i) = i$ and the state dynamics satisfy

$$s_{i+1} = f(s_i, z) := \begin{cases} s_i - h(z), & s_i - h(z) > 1 \\ 0, & h(z) = s_i \text{ and } z \text{ a double} \\ s_i, & \text{otherwise} \end{cases} \tag{6}$$

where we recall that $h(z) \geq 0$ is the random numerical score of the $(i+1)^{st}$ throw. In this non-strategic version of the game we want to compute $V_{\mathrm{NS}}(501, 0)$.

There is a special structure in this DP, however. Specifically the state s_i is non-increasing in i so that $s_{i+1} \leq s_i$ for all i. We can take advantage of this by redefining $V_{\mathrm{NS}}(s)$ to be the minimum

[3] As of writing, van Gerwen is the number one ranked darts player in the world according to the PDC rankings.

[4] Kohler (1982) used the skill model of (1) and due to limited computing power at the time, used various numerical approximations together with a branch-and-bound technique to solve the DP.

expected number of throws required to reach 0 starting from state $s \geq 2$. We then have

$$\begin{aligned} V_{\text{NS}}(s) &= \min_{\boldsymbol{\mu}} \{1 + \text{E}[V_{\text{NS}}(s^{\text{new}}) \mid s, \boldsymbol{\mu}]\} \\ &= \min_{\boldsymbol{\mu}} \left\{ 1 + \sum_z V_{\text{NS}}(f(s,z))\, p(z;\, \boldsymbol{\mu}) \right\} \end{aligned} \tag{7}$$

with $V_{\text{NS}}(0) := 0$. We can solve (7) successively for $s = 2, \ldots, 501$.

3.2 Including the "Turn" Feature

Here the goal is to minimize the number of turns where we recall that a turn consists of three throws. All three are used unless you win or go bust with an earlier throw in which case your score reverts to the score at the beginning of the turn. Working with turns is more accurate as this is how the game of darts is typically played in practice and the rules associated with turn have an important impact on the strategy of players.

When we work with turns the state space becomes more complicated and consists of (s, i, u) where s is the score at the beginning of the turn, $i \in \{1, 2, 3\}$ is the number of throws remaining in the turn and u is the score-to-date *within* the turn so for example if $i = 3$ then $u = 0$. The state transition now satisfies $(s^{\text{new}}, i^{\text{new}}, u^{\text{new}}) = f((s, i, u), z)$ where

$$f((s,i,u),z) := \begin{cases} (0,3,0), & s = u + h(z) \text{ and } z \text{ a double}, \ i \in \{1,2,3\} \\ (s,3,0), & s - (u + h(z)) \leq 1 \text{ and not } (s = u + h(z),\ z \text{ a double}), \\ & \qquad \text{如果扔菜了 这轮算白扔} \ i \in \{1,2,3\} \\ (s, i-1, u + h(z)), & s > (u + h(z)) + 1,\ i > 1 \\ (s - u - h(z), 3, 0), & s > (u + h(z)) + 1,\ i = 1. \end{cases} \tag{8}$$

Note that the first two cases in (8) correspond to getting out and going bust, respectively. The third case corresponds to neither going bust nor getting out and still having at least one throw remaining in the turn. The final case corresponds to the last throw of a turn and neither getting out nor going bust. With these state dynamics the analog of (7) becomes

$$\begin{aligned} V_{\text{NS}}(s,i,u) &= \min_{\boldsymbol{\mu}} \left\{ \text{E}[1_{\{i^{\text{new}}=3\}} + V_{\text{NS}}(f((s,i,u),z)) \mid (s,i,u), \boldsymbol{\mu}] \right\} \\ &= \min_{\boldsymbol{\mu}} \left\{ \sum_z \left[1_{\{i^{\text{new}}=3\}} + V_{\text{NS}}(f((s,i,u),z)) \right] p(z;\, \boldsymbol{\mu}) \right\} \end{aligned} \tag{9}$$

with $V_{\text{NS}}(0, i, u) := 0$ for any i, u. Note also that a turn is counted only when the turn has been completed. This explains why after winning in the first case of (8) we set $i^{\text{new}} = 3$ so that the final turn is counted in (9). As with (7) there is also a monotonic structure to the state dynamics and we can use this to successively solve for $V_{\text{NS}}(2, i, u)$, $V_{\text{NS}}(3, i, u), \ldots, V_{\text{NS}}(501, i, u)$. Each of these 500 problems is more challenging than their analogs in (7) because of the additional states here.

4 What to do

1. In the example code *solve_dp_noturn*, I demonstrated how to solve the dart game without the turn feature. In particular, I directly solved the optimal value and policy through the Bellman equation (7). Now, please solve it using value iteration and policy iteration. See details in Appendix B.3.

2. Solve the dart game with turn feature by value iteration and policy iteration (this is optional). See details in Appendix B.4.

3. The example code *solve_dp_noturn* can take the input using either the action set of 984 aiming locations (a two-dimension array) or the full action set of 341×341 aiming locations (a three-dimension array). Your code should remain the same style and use the full aiming location set to resolve the game with the turn feature.

4. Write a report. Describe the Bellman equations for the dart game with the turn feature. The more important part is to identify some interesting behaviors in the game. For example, why the expected turns/throws is not monotonically increasing in s? How is the improvement for using the full aiming location set? How to make the computation more efficient/fast? (How to choose proper initial values or policies? How to implement the iteration computation in a better way in code?) How the player's Gaussian model impact his targeting choices? You are not required to answer all of these questions. Just pick one or two questions (or interesting issues found by yourself), and make some detailed analysis.

References

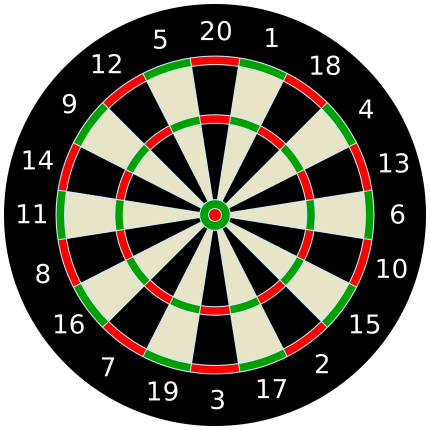
Kohler, David. 1982. Optimal strategies for the game of darts. *Journal of the Operational Research Society* **33**(10) 871–884.

Tibshirani, Ryan J., Andrew Price, Jonathan Taylor. 2011. A statistician plays darts. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* **174**(1) 213–226.

Appendix A Geometry of the Dartboard

A standard competition dartboard has the following measurements (in millimeters):

Distance	Measurement
Center to DB wire	6.35
Center to SB wire	15.9
Center to inner triple wire	99
Center to outer triple wire	107
Center to inner double wire	162
Center to outer double wire	170

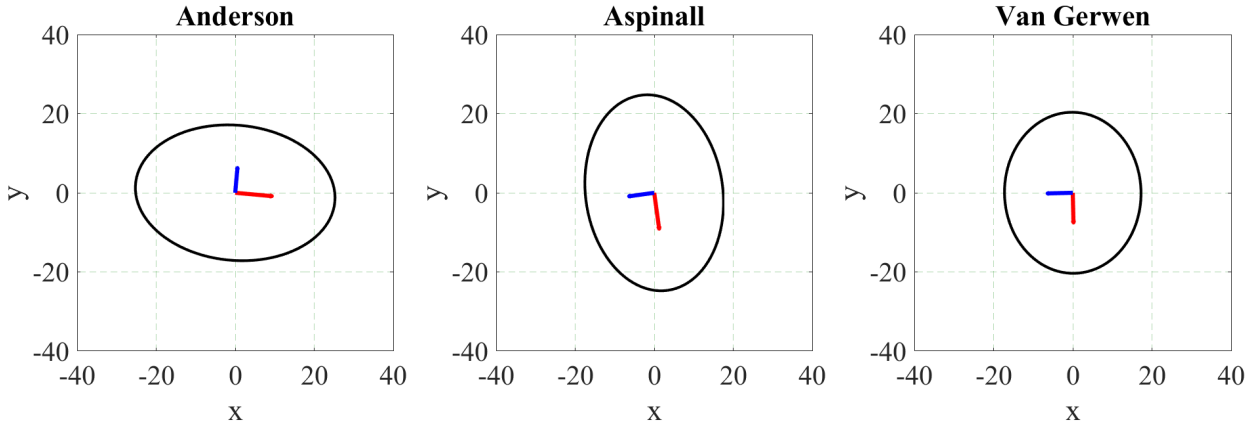


Given these measurements and taking the center of DB to be $(0, 0)$, it is straightforward to compute the map $z = g(x, y)$ that maps any (x, y)-coordinate on the dartboard to the corresponding dart score.

Appendix B More Details

B.1 Estimating the Conditional Skill Gaussian Models

As mentioned in Section 2, we use a conditional Gaussian model to fit the probability of outcome of dart throwing. Specifically, given an aiming location $\boldsymbol{\mu}$ on the dartboard, we identify the region R_m it belongs to, and then the dart landing location follows a bivariate Gaussian distribution $\mathrm{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma}_m)$ with mean $\boldsymbol{\mu}$ (the intended target) and covariance matrix $\boldsymbol{\Sigma}_m$. (The estimation of the mode parameters is not the focus of this project. We used an EM algorithm to estimate each component of the skill model, i.e., $\boldsymbol{\Sigma}_m$ for $m = 1, ..., 7$ from the real data for each player.)



(a) player1 (b) player2 (c) player7

Figure 2: Anderson, Aspinall, and van Gerwen in Fitted Double Models

For example, the covariance matrixes of the Gaussian models for Gary Anderson (player1), Nathan Aspinall (player2) and Michael van Gerwen (player7) aiming at the double areas $R_6 = \{\mathrm{D1}, \ldots, \mathrm{D20}\}$ are as following,

Anderson (player1)

$$\boldsymbol{\Sigma} = \begin{pmatrix} 106.56 & -5.25 \\ -5.25 & 49.18 \end{pmatrix},$$

Aspinall (player2)

$$\boldsymbol{\Sigma} = \begin{pmatrix} 51.41 & -7.18 \\ -7.18 & 102.40 \end{pmatrix},$$

van Gerwen (player7)

$$\boldsymbol{\Sigma} = \begin{pmatrix} 49.60 & -0.46 \\ -0.46 & 69.17 \end{pmatrix}.$$

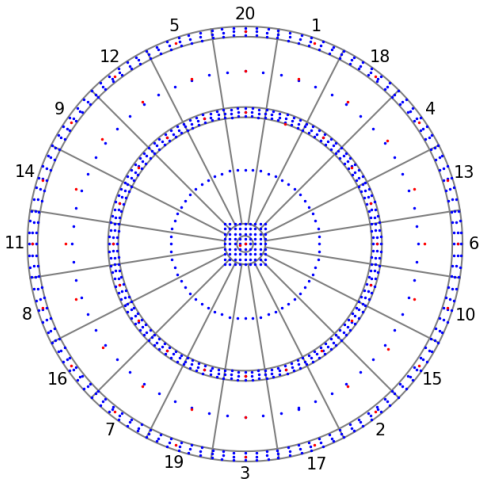
In Figures 2(a), 2(b) and 2(c) we display the 95% confidence ellipses corresponding to these models. (*How do we compare players' skills through the figures?*)

B.2 The Aiming Locations

Since the geometry of the dartboard is measured on a millimeter scale and the dartboard is a circle of radius 170mm, one possible design would be to consider a 340mm $\times$ 340mm square enclosing the board and build a mesh grid of width 1mm to be the space of the potential targets. Unfortunately this would result in $341 \times 341 = 116,281$ possible actions / targets and solving the game with such a large action space is time consuming. However it's clear a priori that a huge number of these actions would never be used and so we instead proceeded to define the space of permissable actions / targets as follows.

Eighteen targets in each of the double and treble regions were permitted for a total of 720 double / treble targets. A small square enclosing the SB and DB regions contained 81 additional targets. Three targets located in each of the inner and outer of single regions were also permitted. This led to a total of $720 + 81 + 120 = 921$ common targets for each of the players. For each particular player, we also include the target within each region, e.g. T20, S12 etc., that has the highest probability of being hit for that player. Finally we also include the single point on the board that has the highest expected score which is generally in T20. This meant an additional 63 targets for each player. Hence, there are 921+63=984 possible aiming locations considered.

A more detailed description of the permissible action space are provided here. (It may help to refer to the geometry of the dartboard as described in Appendix A.) We take a 32mm x 32mm square enclosing the DB and SB regions. Within this square we determined $9 \times 9 = 81$ equally spaced targets. Additional single-region targets are located at (θ, r) for all $\theta \in [0°, 6°, ..., 354°]$ and $r \in [58, 135]$ where (θ, r) represent polar coordinates with the origin located at the center of DB. For double regions we locate permissable targets at (θ, r) for all $\theta \in [0°, 3°, 6°, ..., 357°]$ and $r \in [163, 166, 169]$. For treble regions we locate permissable targets at (θ, r) for all $\theta \in [0°, 3°, 6°, ..., 357°]$ and $r \in [100, 103, 106]$. These target points are displayed in blue in Figure 3 and they are the same for all players.



Note: The blue points are the common aiming locations shared among all players. The red points within each region are the targets that maximize Van Gerwen's expected score within that reason and they are also included in van Gerwen's permissable action set.

Figure 3: The Permissable Action / Target Set of Michael van Gerwen

For each particular player, we also include the target within each region that has the highest probability of being hit according to that player's fitted skill model. Finally, we also include the single point on the board that has the highest expected score which is generally in T20. These points will vary (though not by much) from player to player. These points are plotted in red in Figure 3 which displays the permissable action set for Michael van Gerwen. [5]

B.3 Solve the Single Player Dart Game without the Turn Feature

When excluding the turn feature, the game state is the current score s, and the state dynamics are described by (6) in Section 3.1. Recall the notations defined in the beginning of Section 2, given an target $\boldsymbol{\mu}$, z is the score region where the dart lands, and $h(z)$ is the actual numerical score achieved. $V(s)$ denotes the optimal value at s, i.e., the minimum expected number of throws required to reach 0 starting from state s. We need to solve the Bellman equation (7) as following

$$V(s) = \min_{\boldsymbol{\mu}} \left\{ 1 + \sum_{z} V(f(s,z))\, p(z;\, \boldsymbol{\mu}) \right\},$$

where the summation is over all the score regions of the dartboard, namely S1, ..., S20, D1, ..., D20, T1, ..., T20, SB, and DB.

In coding, it may be more convenient to work with the actual numerical score. Suppose the player throws a dart by targeting $\boldsymbol{\mu}$ at state s, and then one of the three possible outcomes occur:

- If the player makes a numerical score $h \in \{1, 2, ..., s_{\max} = \max(60, s-2)\}$, the game transits to the state $s-h$. Let $p(h;\, \boldsymbol{\mu})$ denote the probability of making a score h when aiming at $\boldsymbol{\mu}$.
- If the player makes a double score $s/2$, the game reaches 0 and finishes. Let $p_D(s/2;\, \boldsymbol{\mu})$ denote the probability of making a double score $s/2$ when targeting at $\boldsymbol{\mu}$.
- If the player makes a score $h > s_{max}$, the game busts and returns to the same state s. Let $p(\text{bust};\, \boldsymbol{\mu}) = \sum_{h > s_{max}} p(h;\, \boldsymbol{\mu})$ denote the probability of bust when aiming at $\boldsymbol{\mu}$. When the player makes a score 0 with probability $p(0;\, \boldsymbol{\mu})$, the game also transits to the state s.

Note that above odds add up to 1, i.e., $\sum_{h=1}^{s_{\max}} p(h;\, \boldsymbol{\mu}) + p_D(s/2;\, \boldsymbol{\mu}) + p(0;\, \boldsymbol{\mu}) + p(\text{bust};\, \boldsymbol{\mu}) = 1$. Now, the Bellman equation (7) becomes

$$V(s) = \min_{\boldsymbol{\mu}} \left\{ 1 + \sum_{h=1}^{s_{\max}} p(h;\, \boldsymbol{\mu}) V(s-h) + p_D(s/2;\, \boldsymbol{\mu}) V(0) + \left(p(0;\, \boldsymbol{\mu}) + p(\text{bust};\, \boldsymbol{\mu})\right) V(s) \right\}, \quad (10)$$

which can be solved successively for $s = 2, ..., 501$ with $V(0) = 0$.

Solving Directly

When solving (10) for state s, suppose that we have already computed $V(s')$ for state of less score $s' < s$, then only $V(s)$ is unknown and all $V(s-h)$ are known for $h = 1, ..., s_{\max}$. Hence, we can directly solve $V(s)$ through the Bellman equation (10) as following.

Let $V(s;\, \boldsymbol{\mu})$ denote the value of s associated with an aiming location $\boldsymbol{\mu}$, i.e., how many throws are required in expectation to reach 0 if targeting $\boldsymbol{\mu}$ at state s and then aiming optimally in later states with score less than s. We can establish

$$V(s;\, \boldsymbol{\mu}) = 1 + \sum_{h=1}^{s_{\max}} p(h;\, \boldsymbol{\mu}) V(s-h) + p_D(s/2;\, \boldsymbol{\mu}) V(0) + \left(p(0;\, \boldsymbol{\mu}) + p(\text{bust};\, \boldsymbol{\mu})\right) V(s;\, \boldsymbol{\mu}), \quad (11)$$

$$V(s;\, \boldsymbol{\mu}) = \frac{1 + \sum_{h=1}^{s_{\max}} p(h;\, \boldsymbol{\mu}) V(s-h) + p_D(s/2;\, \boldsymbol{\mu}) V(0)}{1 - p(0;\, \boldsymbol{\mu}) - p(\text{bust};\, \boldsymbol{\mu})}. \quad (12)$$

[5] These implementation can be found in the code *function_get_aiming_grid_v2.get_aiming_grid_v2()*.

The optimal value are searched through the action set of 985 aiming locations

$$V(s) = \min_{\boldsymbol{\mu}} V(s;\, \boldsymbol{\mu}). \tag{13}$$

Usually we are not able to directly solve the Bellman equations for general stochastic shortest path DP problems, and we may use the following two methods.

Value Iteration
For the problem at state s, we start with an initial value $V_1(s)$, and then we do iterations

$$V_{k+1}(s) = \min_{\boldsymbol{\mu}} \left\{ 1 + \sum_{h=1}^{s_{\max}} p(h;\, \boldsymbol{\mu}) V(s-h) + p_D(s/2;\, \boldsymbol{\mu}) V(0) + \left(p(0;\, \boldsymbol{\mu}) + p(\text{bust};\, \boldsymbol{\mu})\right) V_k(s) \right\}, \tag{14}$$

for $k = 1, 2, ...$ to improve the value until it converges to the optimal one, i.e., when $|V_{k+1}(s) - V_k(s)|$ is very small.

Policy Iteration
For the problem at state s, we start with an initial policy $\boldsymbol{\mu}_1$, and generate iteratively a sequence of new policies $\boldsymbol{\mu}_2, \boldsymbol{\mu}_3, ...$ as follows.

Given the policy $\boldsymbol{\mu}_k$, i.e., which aiming location to choose at state s, we perform a policy evaluation step to compute $V(s;\, \boldsymbol{\mu}_k)$ which can be done through (11) and (12)

$$V(s;\, \boldsymbol{\mu}_k) = 1 + \sum_{h=1}^{s_{\max}} p(h;\, \boldsymbol{\mu}_k) V(s-h) + p_D(s/2;\, \boldsymbol{\mu}_k) V(0) + \left(p(0;\, \boldsymbol{\mu}_k) + p(\text{bust};\, \boldsymbol{\mu}_k)\right) V(s;\, \boldsymbol{\mu}_k),$$

$$V(s;\, \boldsymbol{\mu}_k) = \frac{1 + \sum_{h=1}^{s_{\max}} p(h;\, \boldsymbol{\mu}_k) V(s-h) + p_D(s/2;\, \boldsymbol{\mu}_k) V(0)}{1 - p(0;\, \boldsymbol{\mu}_k) - p(\text{bust};\, \boldsymbol{\mu}_k)}.$$

We then perform a policy improvement step to compute a new policy $\boldsymbol{\mu}_{k+1}$ as

$$\boldsymbol{\mu}_{k+1} = \arg\min_{\boldsymbol{\mu}} \left\{ 1 + \sum_{h=1}^{s_{\max}} p(h;\, \boldsymbol{\mu}) V(s-h) + p_D(s/2;\, \boldsymbol{\mu}) V(0) + \left(p(0;\, \boldsymbol{\mu}) + p(\text{bust};\, \boldsymbol{\mu})\right) V(s;\, \boldsymbol{\mu}_k) \right\}. \tag{15}$$

We repeat the process until it converges to the optimal policy when $\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k$.

B.4 Solve the Single Player Dart Game with the Turn Feature

When including the turn feature, the game state is (s, i, u) and the state dynamics are described by (8) in Section 3.2. The turn feature makes the dart game complicated by growing the state space significantly. For example, if the player starts the turn with $s = 6$, the possible states in this turn are $(s = 6, i = 3, u = 0)$, $(s = 6, i = 2, u = 0, 1, 2, 3, 4)$, and $(s = 6, i = 1, u = 0, 1, 2, 3, 4)$, and the state transitions in the turn are demonstrated in Figure 4.

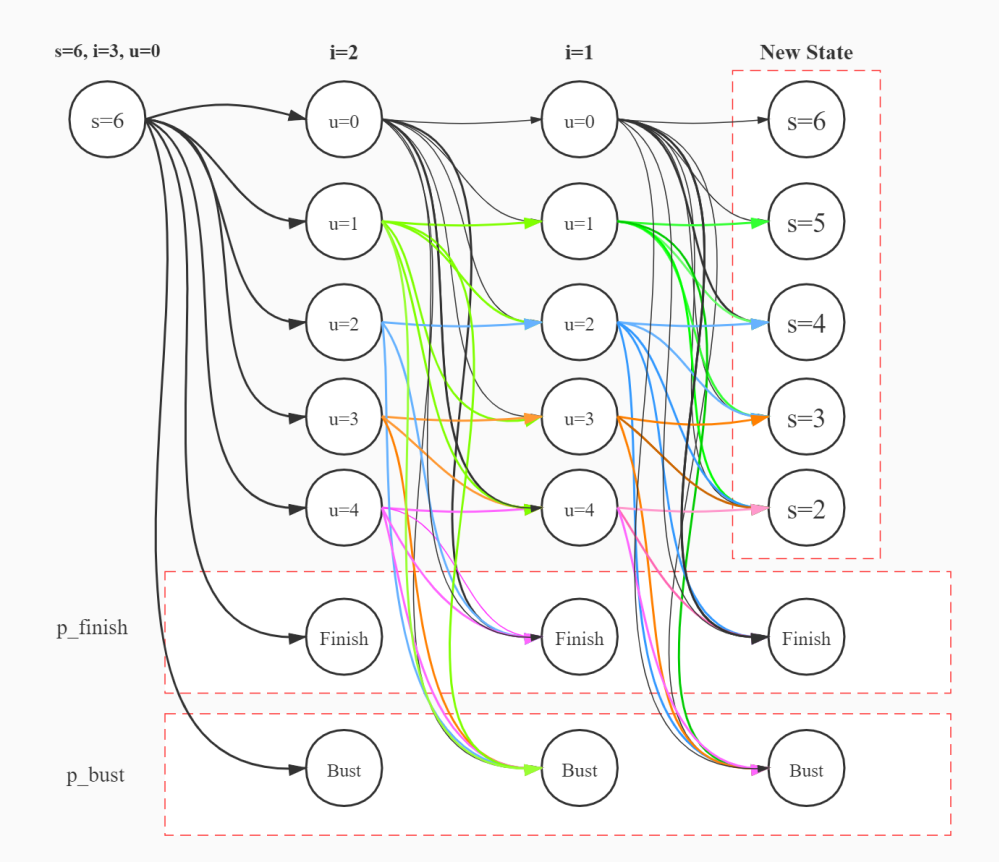


Figure 4: The State Transitions in the Turn Starting at $s = 6$

Similar to the analysis in Section B.3, after the player throws a dart by targeting $\boldsymbol{\mu}$ at state (s, i, u), the game may transit to the state of next throw (or next turn), reach 0, or bust depending on the score h achieved. The Bellman equations (9) can be reformulated as

$$V(s, i = 3, u) = \min_{\boldsymbol{\mu}} \left\{ \sum_{h=0}^{\max\{60, s-2\}} p(h; \boldsymbol{\mu}) \left[0 + V(s, i = 2, h)\right] + p_D\left(s/2; \boldsymbol{\mu}\right) \left[1 + V(0)\right] \right.$$
$$\left. + p(\text{bust}; \boldsymbol{\mu}) \left[1 + V(s, i = 3, 0)\right] \right\} \text{ for } u = 0, \tag{16}$$

$$V(s, i = 2, u) = \min_{\boldsymbol{\mu}} \left\{ \, ? \, V(s, i = 1, u + h) \, ? \, V(0) \, ? \, V(s, i = 3, 0) \right\} \text{ for } u = 0, 1, ..., \max\{60, s - 2\}, \tag{17}$$

$$V(s, i = 1, u) = \min_{\boldsymbol{\mu}} \left\{ \, ? \, V(s - u - h, i = 3, 0) \, ? \, V(0) \, ? \, V(s, i = 3, 0) \right\} \text{ for } u = 0, 1, ..., \max\{120, s - 2\}. \tag{18}$$

Recall that there is also a monotonic structure to the state dynamics where the beginning score of turns in the game is non-increasing, and we can use this to successively solve for $V(2, i, u)$, $V(3, i, u)$,...,$V(501, i, u)$. Suppose we already computed $V(s', i, u)$ for $s' < s$, however, we can not solve $V(s, i, u)$ directly through equations (16) , (17) and (18) in the way of (11)-(13). (why?)

Value Iteration

For the states in the turn beginning with score s, we start with an initial value $V_1(s,i,u)$, and then we do iterations

$$V_{k+1}(s,i=3,u) = \min_{\boldsymbol{\mu}} \left\{ \sum_{h=0}^{\max\{60,s-2\}} p(h;\,\boldsymbol{\mu})\,[0+V_k(s,i=2,h)] + p_D\,(s/2;\,\boldsymbol{\mu})\,[1+V(0)] \right.$$
$$\left. + p(\text{bust};\,\boldsymbol{\mu})\,[1+V_k(s,i=3,0)] \right\} \text{ for } u=0, \tag{19}$$

$$V_{k+1}(s,i=2,u) = \min_{\boldsymbol{\mu}} \left\{ ? \right\} \text{ for } u=0,1,...,\max\{60,s-2\}, \tag{20}$$

$$V_{k+1}(s,i=1,u) = \min_{\boldsymbol{\mu}} \left\{ ? \right\} \text{ for } u=0,1,...,\max\{120,s-2\}. \tag{21}$$

for $k=1,2,...$ to improve these values until they converges to the optimal one, i.e., when $|V_{k+1}(s,i,u) - V_k(s,i,u)|$ is very small for each possible (s,i,u).

Policy Iteration

For the states in the turn beginning with score s, we start with an initial policy $\boldsymbol{\mu}_1(s,i,u)$, and generate a sequence of new policies $\boldsymbol{\mu}_2(s,i,u), \boldsymbol{\mu}_3(s,i,u), ...$ iteratively.

Given the policy $\boldsymbol{\mu}_k(s,i,u)$, the policy evaluation step is to compute $V(s,i,u;\,\boldsymbol{\mu}_k(s,i,u))$. (Here we write the dependence of a policy $\boldsymbol{\mu}_k$ on the state (s,i,u) explicitly to remind that the aiming location could vary over different states in the same turn. In the following we simply write $\boldsymbol{\mu}_k(s,i,u)$ as $\boldsymbol{\mu}_k$ to save notations.)

$$V(s,i=3,u;\,\boldsymbol{\mu}_k) = \sum_{h=0}^{\max\{60,s-2\}} p(h;\,\boldsymbol{\mu}_k)\,[0+V(s,i=2,h;\,\boldsymbol{\mu}_k)] + p_D\,(s/2;\,\boldsymbol{\mu}_k)\,[1+V(0)]$$
$$+ p(\text{bust};\,\boldsymbol{\mu}_k)\,[1+V(s,i=3,0;\,\boldsymbol{\mu}_k)] \text{ for } u=0, \tag{22}$$

$$V(s,i=2,u;\,\boldsymbol{\mu}_k) = ? \text{ for } u=0,1,...,\max\{60,s-2\}, \tag{23}$$

$$V(s,i=1,u;\,\boldsymbol{\mu}_k) = ? \text{ for } u=0,1,...,\max\{120,s-2\}. \tag{24}$$

Equations (22), (23) and (24) form a linear equation system where the unknown variables are $V(s,i,u;\,\boldsymbol{\mu}_k)$. We may solve these equations by standard techniques, such as Gaussian elimination. If we take a careful look, we may compute $V(s,i=3,u=0;\,\boldsymbol{\mu}_k)$ from $V(s'<s,i=3,u=0)$ directly by considering the transition probability among turns.

The policy improvement step is to compute a new policy $\boldsymbol{\mu}_{k+1}(s,i,u)$ as

$$\boldsymbol{\mu}_{k+1}(s,i=3,u) = \arg\min_{\boldsymbol{\mu}} \left\{ \sum_{h=0}^{\max\{60,s-2\}} p(h;\,\boldsymbol{\mu})\,[0+V(s,i=2,h;\,\boldsymbol{\mu}_k)] + p_D\,(s/2;\,\boldsymbol{\mu})\,[1+V(0)] \right.$$
$$\left. + p(\text{bust};\,\boldsymbol{\mu})\,[1+V(s,i=3,0;\,\boldsymbol{\mu}_k)] \right\} \text{ for } u=0, \tag{25}$$

$$\boldsymbol{\mu}_{k+1}(s,i=2,u) = \arg\min_{\boldsymbol{\mu}} \left\{ ? \right\} \text{ for } u=0,1,...,\max\{60,s-2\}, \tag{26}$$

$$\boldsymbol{\mu}_{k+1}(s,i=1,u) = \arg\min_{\boldsymbol{\mu}} \left\{ ? \right\} \text{ for } u=0,1,...,\max\{120,s-2\}. \tag{27}$$

We repeat the process until the policy converges to the optimal one, i.e., when $\boldsymbol{\mu}_{k+1}(s,i,u) = \boldsymbol{\mu}_k(s,i,u)$ for each possible (s,i,u).