Strategy for bitches (a dice game)

Dave Fetterman

Obviously Unemployed

8/16/22

Abstract

This paper details a decision framework for playing well at "bitches (a dice game)", introduced to me by Larry Waldman. Mr. Waldman asked for a truly *optimal* strategy but, like blackjack, the state of the game is likely tractable for a computer to "solve" but too large for a human to remember during gameplay. If, however, we think of this as many simple, parallel games, we can produce a very memorizable "blackjack cheat card" for b. (a. d. g.) that still performs reasonably well.

1 Introduction

Following is a series of desperately important text messages from Larry "F." Waldman about the game "bitches (a dice game)", which is buyable here (`https://gluebunnygames.com/products/bitches-a-dice-game`), or easy to assemble from fifteen dice and the frantic instructions below in Fig. 1.

```
Gotta tell you about this game. I need a paper on the game theory optimized
    strategy

12 regular dice + 1 8 sided die, 1 10 sided die, 1 12 sided die.

Roll all the dice. You must take at least one die out each roll. If the die (or
    dice) you remove are not their max value (eg 6 for a regular die), you get max
     - <die value> amount of points. So for example if you take away a regular die
     showing four, you get two points.

Points are bad.

After all the dice are gone, add up your points and that is your score.
```

Figure 1: Larry has to talk to me

I find the use of "regular dice" to be six-normative but agree to proceed with the following two missions:

- Determine the strategy for getting the best expected score ("optimal strategy").
- Determine the expected number of points from playing that strategy.

Note: I generally proceed with solving an equivalent game: picking up a die gives you as many points as the face value. So, instead of getting one (bad) point when picking up a six-sided die showing five, you receive five (good) points. A score minimizing strategy using the b. (a. d. g.) scoring system and one maximizing this scoring system are equivalent; accepting a die with s sides gives you $s - p$ (bad) points in the first system and p in the second. We can convert "expected face value" to "expected b (a. d. g.) points" trivially when we're done.

The proof/algorithm sketch:

- Section 2: We solve the game definitively for one die of any size.
- Section 3: We show solving the game perfectly for the standard b. (a. d. g.) setup (15 dice of sizes indicated in Fig. 1) can be done perfectly, but requires evaluating way too many states and transitions to be feasible, fun, or useful in play.
- Section 4: We propose a simple strategy: We can reduce the game by considering 15 dice as each playing parallel and separate single-die games solved in Section 2 and bear the cost when our bubble of delusion is occasionally burst. We calculate a guess at a "typical" score with this strategy.
- Section 5: We use computers to validate the real expected score.

2 Solving the one die case

As the old saying goes, "if you cannot play chess well with 3 pieces, you cannot play well with 32." Therefore, in chess, we study endgames first. Here, **we first solve a similar game played with a single die** with unique incremental pip counts $1, 2, 3...s$ up to some arbitrary s, in which we try to maximize our current roll (which the then pick up, or keep rolling).

Though we make a misstep in solving this next (the Aside section), the following principle is rock solid.

Principle: Know when to quit

- We establish an optimal strategy for the base case $r = 1$ (one roll remaining).
- With $r > 1$ rolls remaining, if accepting the current value of the die has a higher expected value than the expected value using the optimal strategy going forward, accept the die value. Else re-roll.

If[1] we have an optimal strategy for $r = 1$, then we necessarily have one for $r = 2, 3...$ and so on.

2.1 Aside: (Incorrectly) using the expected maximum future roll

We will now take a misstep in finding the optimal strategy, an example of an intuitive but wrong move in probabilistic thinking. It helps illustrate why the best strategy is best, and so we include it.

The logic: We, of course, want to choose the best future roll from this die, so the **expected max strategy** is *if the current roll is worse than the expected future maximum among the r rolls remaining, keep rolling. Otherwise, pick up the die and accept the points.*

This *seems* like the right idea. After all, if we can expect our future rolls to include a better roll on average, we should keep rolling, right? This is true, but not the whole story.

At least it's easy to calculate. Given s (a singe die's number of sides) and r rolls remaining, the expected value of the max roll value is straightforward:

$$p_{max \leq k}(s, r) = (\frac{k}{s})^r \tag{1}$$

$$p_{max \leq k-1}(s, r) = (\frac{k-1}{s})^r \tag{2}$$

$$p_{max=k}(s, r) = (\frac{k}{s})^r - (\frac{k-1}{s})^r \tag{3}$$

$$E_{max}(s, r) = \sum_{k=1}^{s} k[\frac{k^r - (k-1)^r}{s}] \tag{4}$$

For any given number of pips k between 1 and s:

- (1) is the probability that all of the r rolls are between 1 and k.
- (2) is the probability that all of the r rolls are between 1 and $k - 1$.
- (3) subtracts these two for the probability that the max roll is exactly k (as in, there's at least one k roll in the space of (1)).

[1] We do: we only have the option of accepting the roll. Expected face value is clearly $\frac{s+1}{2}$.

- (4) is the familiar expected value formula.

Fig. 2.shows the results for 6-, 8-, 10-, and 12-sided dice.

r	$s = 6$	$s = 8$	$s = 10$	$s = 12$
1	3.5	4.5	5.5	6.5
2	4.472	5.812	7.15	8.486
3	4.958	6.469	7.975	9.479
4	5.245	6.858	8.467	10.072
5	5.431	7.115	8.792	10.465
6	5.56	7.295	9.022	10.744
7	5.654	7.428	9.192	10.952
8	5.724	7.529	9.323	11.111
9	5.778	7.608	9.426	11.238
10	5.82	7.67	9.509	11.34
11	5.853	7.721	9.576	11.424
12	5.88	7.763	9.633	11.495
13	5.901	7.798	9.68	11.554
14	5.919	7.827	9.72	11.605
15	5.933	7.851	9.754	11.648

Figure 2: Expected max roll of an s-sided die over r rolls

We then look to the expected value of the next state and pick up our die if the value equals or exceeds it. For example, if we're rolling a 6-sided die and we have three rolls left, after rolling we pick up if and only if the roll exceeds what we'd expect by taking our shot with two rolls left (4.472). This means we'd pick up a 5 or 6. If we continue, we would pick up anything equal to or exceeding 3.5 on the next roll ($r = 1, s = 6$).

2.2 Why doesn't this work?

Though we would like to choose the maximum roll across our future rolls, and we can be reasonably confident some high numbers are ahead of us if we have many rolls remaining, we have a problem: *we are not guaranteed to choose the max roll when we see it*, even when appropriarly we adjust our expectations of the max roll *down* as we near the end of the cliff as in Fig. 2.

Counterexample to max strategy

Consider this example with a 50-sided die and three rolls remaining, with $E_{max}(s, r)$ signifying the expected value of the **max** roll over r rolls of a s-sided die, which can be computed with formula (4) above.

- $E_{max}(50, 1) = 25.5$. You must pick up once rolled.

- $E_{max}(50, 2) = 33.83$. You should pick up if your roll equals or exceeds $\lfloor E_{max}(50, 1) \rfloor = 25$.
- $E_{max}(50, 3) = 37.995$. You should pick up if your roll equals or exceeds $\lfloor E_{max}(50, 2) \rfloor = 33$.

The reroll cutoff (inclusive) on $r = 2$ will be 25 for any optimal strategy, since no decisions are possible for $r = 1$. The expected value for this Max strategy $E_M(r = 2)$ is 31.75.

$$E_M(2) = \sum_{k=26}^{50} \frac{k}{50} + \frac{25}{50} * 25.5 = 31.75 \tag{5}$$

So we look at the expected value of five strategies (A, B(est), C, M(ax), D) then, which are exactly the different re-roll cutoffs when $r = 3$: (30, 31, 32, 33, 34), respectively:

$$E_A(3) = \sum_{k=31}^{50} \frac{k}{50} + \frac{30}{50}(31.75) = 35.25 \tag{6}$$

$$E_B(3) = \sum_{k=32}^{50} \frac{k}{50} + \frac{\mathbf{31}}{50}(31.75) = \mathbf{35.265} \tag{7}$$

$$E_C(3) = \sum_{k=33}^{50} \frac{k}{50} + \frac{32}{50}(31.75) = 35.260 \tag{8}$$

$$E_M(3) = \sum_{k=34}^{50} \frac{k}{50} + \frac{\mathbf{33}}{50}(31.75) = 35.235 \tag{9}$$

$$E_D(3) = \sum_{k=35}^{50} \frac{k}{50} + \frac{34}{50}(31.75) = 35.19 \tag{10}$$

We see that the expected value of our strategy peaks not at our supposed max strategy of rerolling at 33 or less, but rerolling at 31 or less (accepting 32 and 33).

This counterexample proves that max is not the optimal strategy, though not by a whole lot for this small (and easily calculable) example. The real intuition comes from considering that, for example, if we're three rolls out on a 50-sider, there is a *possibility* that there are one or even *two* fifties in my future; those will drive up the expected maximum roll. However, the likelihood that I will encounter them is diminished by the chance that I'm rolling high thirties or above before that; I may have big maxes in my future, but pretty big rolls now mean I'm going to take my marbles home now.

Three future in-order rolls of (35, 1, 50) and (50, 1, 35) both contribute 50 equally to the expected maximum. However, I will never encounter the first case, since I will have optimally picked up the 35 by then, which 'shadows' the high rolls. Note that (35, 1, 10) doesn't 'shadow' a future low roll, since the maximum is not the third number but 35.

If this isn't intuitive enough, consider that on a 1000-sided die with three rolls left, you have to exclude the possibility of the sequence (999, *, 1000). This part of the possibility space contributes 1000 to the max strategy, but to no reasonable strategy otherwise. This is why our EVs for the optimal strategy will be lower than the intuitive "pick the expected future maximum". Instead, we need to "pick the expected future maximum *that we will get to select*".

2.3 Solving the one-die case optimally

Though there doesn't appear to be a closed form for the expectations and cutoffs of the optimal strategy, determining it is no big shakes. If we have r rolls left for a die of size s, and we have the optimal expected value $E_B(s, r-1)$ for the version of the problem with one less roll, our optimal strategy, according to the *Know when to quit principle*, is to keep rolling unless we equal or exceed the value of continuing. For ease of reading, we introduce cutoff $c(s, r)$ to mean $\lfloor E_B(s, r-1) \rfloor - 1$. This is the highest number we'd reject (keep rolling with) when evaluating roll r.

$$E_B(s, 1) = \frac{s+1}{2} \tag{11}$$

$$E_B(s, r) = \sum_{c(s,r)<k\leq s} k(\frac{1}{s}) + E_B(s, r-1)\frac{c(s, r)}{s}, r > 1. \tag{12}$$

(13) is our known only (and therefore optimal) value of a game with one roll. The first term of (14) is the expected payoff we get by picking up if we exceed our cutoff. The second term is the expected payoff we get by continuing if we don't exceed our cutoff.

So, for playing a one-die version of this game with a die of size s, start by rejecting anything except s, and follow the blue colored entries in Figs. 3 through 6 for where you should start adjusting your cutoffs down.

For ease of use, we also present this as a "Blackjack Table" in Fig. 7, the kind of extraneous card often found in a 52-deck alongside promotional flotsam, which tells you when to hit or stay based on the dealer's card and your cards.[2] After reader feedback, we present these

[2] A blackjack table ignores the make up of cards left in the deck as a necessary simplification.

dice left (6-side)	max reject	expected face	exp badg points
1		3.5	2.5
2	3	4.25	1.75
3	4	4.667	1.333
4	4	4.944	1.056
5	4	5.13	0.87
6	5	5.275	0.725
7	5	5.396	0.604
8	5	5.496	0.504
9	5	5.58	0.42
10	5	5.65	0.35
11	5	5.709	0.291
12	5	5.757	0.243
13	5	5.798	0.202
14	5	5.831	0.169
15	5	5.859	0.141

Figure 3: Optimal 6-sided die strategy

dice left (8-side)	max reject	expected face	exp badg points
1		4.5	3.5
2	4	5.5	2.5
3	5	6.063	1.938
4	6	6.422	1.578
5	6	6.691	1.309
6	6	6.894	1.106
7	6	7.045	0.955
8	7	7.165	0.835
9	7	7.269	0.731
10	7	7.36	0.64
11	7	7.44	0.56
12	7	7.51	0.49
13	7	7.571	0.429
14	7	7.625	0.375
15	7	7.672	0.328

Figure 4: Optimal 8-sided die strategy

as *keeper cutoffs*, one more than $c(s, r)$ in the rest of the paper, and color the cells green to distinguish and annoy.

dice left (10-side)	max reject	expected face	exp badg points
1		5.5	4.5
2	5	6.75	3.25
3	6	7.45	2.55
4	7	7.915	2.085
5	7	8.241	1.76
6	8	8.492	1.508
7	8	8.694	1.306
8	8	8.855	1.145
9	8	8.984	1.016
10	8	9.087	0.913
11	9	9.179	0.821
12	9	9.261	0.739
13	9	9.335	0.665
14	9	9.401	0.599
15	9	9.461	0.539

Figure 5: Optimal 10-sided die strategy

dice left (12-side)	max reject	expected face	exp badg points
1		6.5	5.5
2	6	8	4
3	7	8.833	3.167
4	8	9.389	2.611
5	9	9.792	2.208
6	9	10.094	1.906
7	10	10.328	1.672
8	10	10.523	1.477
9	10	10.686	1.314
10	10	10.822	1.178
11	10	10.935	1.065
12	10	11.029	0.971
13	11	11.11	0.89
14	11	11.184	0.816
15	11	11.252	0.748

Figure 6: Optimal 12-sided die strategy

By building up from ground truth (one roll) to our state of r rolls remaining, using the Principle of Knowing When To Quit, we have shown that this is the optimal algorithm for one die of any size.

dice left	Keep (6)	Keep (8)	Keep (10)	Keep (12)
2	4	5	6	7
3	5	6	7	8
4	5	7	8	9
5	5	7	8	10
6	6	7	9	10
7	6	7	9	11
8	6	8	9	11
9	6	8	9	11
10	6	8	9	11
11	6	8	10	11
12	6	8	10	11
13	6	8	10	12
14	6	8	10	12
15	6	8	10	12

Figure 7: The "Blackjack Table": Pick up dice with green or higher.

3 The Architecture of a perfect solution

The state of the 15-die game is not quite as simple.

Whereas the one-die game's[3] state S can be summarized as $S = [0, r] \in \mathbb{N}$, that is, a single integer between 0 and r rolls left, inclusive, we now have state $S = [0, 12] \times [0, 1] \times [0, 1] \times [0, 1] \in \mathbb{N}^4$. This reads as the combination of the count of six-sided, eight-sided, ten-sided, and twelve-sided dice that remain, for a total of 104 states (including the "done" state of $(0, 0, 0, 0)$.)

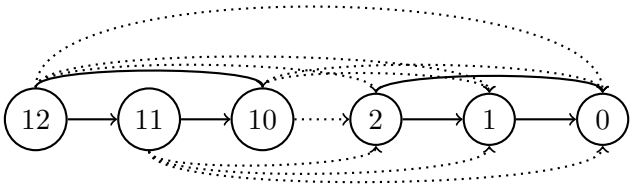


Figure 8: Transitions between 6-die substates

We can best visualize the transitions between these 104 states with the *two* graphs in Figs. 8 and 9. The total state is exactly the combination of having anywhere from 0 to 12

[3]Note that we expanded the one-die game a bit with r free rolls instead of a boring one-roll game. You will see why shortly.

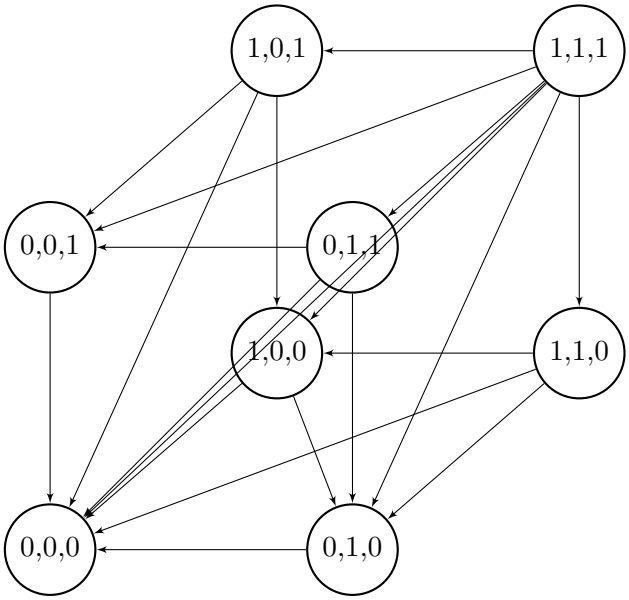


Figure 9: Transitions between (B,C,D) = (8-die,10-die,12-die) substates

6-dice left (Fig. 8) and having 0 or 1 (8-, 10-, 12-) dice left (Fig. 9). To conceptualize the whole graph in one figure, imagine breaking each of the $[0,1]^3$ nodes into its own copy of (Fig. 8), and further, that every combined node $S_a = (a_6, a_8, a_{10}, a_{12})$ has a directed edge to node $S_b = (b_6, b_8, b_{10}, b_{12})$ if and only if $a_6 \leq b_6, a_8 \leq b_8, a_{10} \leq b_{10}, a_{12} \leq b_{12}, S_a \neq S_b$. More direclty, these are the transitions where we remove one or more dice from our state to get to a new one.

Given a state S and the rolls we've gotten, we still want to choose to pick up the dice that give us the best expected value going forward. This means running this algorithm to determine the expected value of every state S before we even start:

Pre-compute Algorithm: Expected value of die state S:

1. Start with accumulator $z = 0$.
2. For each possible combination c of rolls of $S = (a_6, a_8, a_{10}, a_{12})$ dice counts of size 6,8,10,12 respectively:
   (a) For each non-empty member r of the power set 2^c, where we take points and remove those r dice:
      i. Compute points added $Points(r, c)$.
      ii. Determine future state from removing those dice $S_+ = S_a - r$.
      iii. Look up saved expected value E_{S_+} from previous computations.
      iv. Compute total value $E_S(r, c) = Points(r, c) + E_{S_+}$
   (b) Choose the r^* with the highest value $E_S(r^*, c)$.
   (c) Add $\frac{E_S(r^*,c)}{p(c)}$ to z, where $p(c)$ is the probability of combination c.
3. Store $E_S \leftarrow z$.

We can precompute each such $E_S, S \in ([0, 12] \times [0, 1]^3 \in \mathbb{N}^4)$, in a *reverse topological sort*, meaning crawling the graphs of Figs. 8 and 9 *upstream* against the arrows, so that all of their required E_{S_+} are ready for them. An order might be doing all states with 0 dice, then 1 die, then 2 dice, like so:

- $S = (0, 0, 0, 0)$ (known to be 0)
- $S = (0, 0, 0, 1)$ (known to be 6.5)
- $S = (0, 0, 1, 0)$ (known to be 5.5)
- $S = (0, 1, 0, 0)$ (known to be 4.5)
- $S = (1, 0, 0, 0)$ (known to be 3.5)
- $S = (2, 0, 0, 0)$
- $S = (1, 1, 0, 0)$
- $S = (1, 0, 1, 0)$
- $S = (1, 0, 0, 1)$
- $S = (0, 1, 0, 1)$
- ...
- $S = (3, 0, 0, 0)$
- ...
- $S = (12, 1, 1, 1)$

This idea is an example of *dynamic programming*, the idea of starting with less-complicated computations and saving the results to build to more complicated ones. If we can rely on the previous computations as producing the best expected value, starting from known answers (the value of the empty state (0); the expected value of rolling a single die (3.5, 4.5, 5.5, 6.5 for our sizes)), the principle of choosing the move with the best expected value will produce, by extension, the best expected value at any larger state (including the 15-die state at the beginning of b.a.d.g.).

Finally, once all the E_S are computed, in order to make the best choice during game time, the most straightforward correct algorithm is likely *running steps (2a) and (2b) from the Pre-Compute algorithm above.*

Doing the pre-computation would be doable, but a pain. Computing the truly best move in real-time would be a lot for a computer, and certainly impossible for a human..

3.1 Why so difficult?

Let's look at the volume of each of these steps in the **Algorithm** above to see the volume of these computations.

- Step 2. *... each possible combination c of rolls of* $S = (a_6, a_8, a_{10}, a_{12})$ *dice of size 6,8,10,12 respectively:*
  - At the beginning state, naively, with 12 six-sided dice, there are $6^{12} = 2,176,782,336$ possible rolls if the dice are all considered distinct.
  - Noting that rolling $(1,1,2,2,3,3,4,4,5,5,6,6)$, $(6,6,5,5,4,4,3,3,2,2,1,1)$ and $(1,6,1,6,2,5,2,5,3,4,3,4)$ are all the same, we can compute using the formula for "12 indistinct balls in 6 distinct bins"[4] to get 6188 truly different rolls.
  - Combined with the possibilities for the other dice, this makes $(6188)(8)(10)(12) = 5,940,480$ rolls to consider at the beginning.
  - Naturally this is the biggest number of these. There may be only 6 rolls to consider at the last roll, perhaps. Even in the middle (say, 6 6-sided dice, one 8-sided, one 10-sided), we're looking at 36,960 rolls.
- Step 2a: *... each non-empty member r of the power set* 2^c*, where we take points and remove those r dice*
  - If there are 15 dice, the power set of these (minus the empty set[5]) has $2^{15} - 1 = 32,767$ members.
  - Again, looking in the middle of the game, $2^8 - 1 = 255$ is still a lot to go through.

[4] $\binom{12+6-1}{6-1}$

[5] Gotta take at least one die!

  - We could mitigate this by not examining any choices of r where we accept multiple low-valued dice (half the pip count or less), which is never a good choice. If half the values are low, then we're still left with examining something like $2^7 = 128$ possibilities. Of course, we have to account for getting all ones in our expected value computation!

- The other computations are either simple (e.g. $p(c)$ in 2c) or already done (E_S for an earlier state S).
- And, of course, we need to do this 103 times[6], walking backward through the graphs in Figs. 8 and 9.
- If we took the average computation as the one where there are 6 6-sided dice, one 8-sided, and one 10-sided, that still totals $36960 * 255 * 103 = 970,754,400$ steps. Of course, a middle step is more like 7% as hard as the beginning step, not 50%!

In the most optimistic case, we can pre-compute the value of all 104 expected states. Then, for each of them we can generate a lookup table for the up to 6 million rolls possible. Again, we can perhaps cut out some of the space where the die are high enough (s) or low enough ($\leq \frac{s}{2}$) for the answer to be obvious, but even specifying this strategy fully will take a lot of data.

It is possible that there is a perfect and compressible heuristic with a lot less data. We present an imperfect but extremely small heuristic that should perform pretty well.

4 Splitting into parallel games

The main insight of this paper is: **We can reduce the game by considering 15 dice as each playing separate one-die games and bear the cost when this ends up being untrue.** This produces a game with a compressible, human-scale strategy table, and a pretty good score (we think).

The value of this should be clear. If we have n dice remaining, and each die is playing a one-die game with up to *about* n possible rolls[7] in its future, by the "Know When to Quit" principle, we should continue to roll that die until it's time to quit.

If this is the case, then our by linearity of expectation[8], the EV is clear from the lower-right entries in Figs. 3, 4, 5, and 6. 12 6-dice with an EV best face of 5.859 (.141 points), an 8-die with 7.762 (.328 points), a 10-die with 9.461 (.539 points), and a 12-die with 11.252 (.748 points), end up giving us an **expectation of 3.307 b. (a. d. g.) points.**.

[6] Empty set is a freebie: 0.

[7] So we're approximating rolls r with number of dice n.

[8] The expected value of a sum of variables is the same as the sum of the expected values of the variables.

However, this is not the case for two reasons:

- If we get lucky and start bagging lots of high rolls, we have fewer rolls ahead of us as we might "skip" from, say 15 dice left to 12 by picking up 3 instead of 1. This is *simultaneous fortune*.
- If we get unlucky, we will be forced to choose pick up a die at or below its cutoff. This is the main issue we'll face, or *simultaneous misfortune*. Within this:
  - How often will we encounter misfortune?
  - How much is it gonna cost us when we do?

It's clear that we cannot declare that our strategy is the best *only if nothing goes wrong*; just imagine we said "Only pick up a die if it's the maximum. If nothing goes wrong, your total score will be zero".

But we'll focus on approximating the amount of pain we'll endure with this strategy, on top of the 3.307 points.

Simultaneous fortune at least has two countervailing factors which may help cancel it out.

- Good: Simultaneously rolling a bunch of max rolls, say, helps us get a better score; it is clear we should always pick them up.
- Good: Big jumps make the game shorter, and a shorter game has a smaller chance of the misfortune that actually does damage to us (makes us eat points), *simultaneous misfortune*.
- Bad: At the same time, it reduces the number of at-bats for the other dice by a little bit.

For a game of 12 6-sided dice, rolling three sixes betters our expected (b. a. d. g.) score for those dice by (.243 * 3 = .729) points, while the other nine see their expected score rise to .42 from .291 (1.161 points).

In reality, simultaneous fortune is an unalloyed good. The only way it "hurts us" is merely accounting: it simply makes the remaining dice's expected values derived from the "separate games" simplification less accurate, as we go from having 15 rolls to somewhat fewer.

4.1 An Example Game

Each b. (a.d.g.) game is a walk through the 104-node graph of Figs. 8 and 9 from node (12,1,1,1) to node (0,0,0,0), taking between one and fifteen steps. The number of such paths bears a lot of similarity to the large numbers of the previous section. Calculating the

expected deviations (fortune and misfortune) from the parallel model is daunting. Instead, let's just look at a typical "seeming" path to get a sense of our chance of simultaneous fortune and misfortune.

Note that the chance of simultaneous misfortune for a set of dice $S = (a, b, c, d)$, with cutoffs (c_a, c_b, c_c, c_d) is $p_{sm}(a, b, c, d) = (\frac{c_a}{6})^a(\frac{c_b}{8})^b(\frac{c_c}{10})^c(\frac{c_d}{12})^d$; you can read this as "all dice are at or below their cutoff". Using the tables for our strategy, let's take a typical seeming walk.

The rules for this walk in Fig. 10 are:

- We pick up dice (second column) according to the tables above. Instead of using rolls remaining, however, we use dice count (first column). We could create a function to better estimate rolls remaining, but we're trying to create a human-runnable strategy here. We need to keep it simple.
- We calculate the expected number of simultaneous misfortunes along the way as $p_{sm}(S)$ in the third column. The expected sum is the sum of these turn-based expectations.
- At each roll we add to an expected number of "hits" (over the cutoff) for the 6, 8, 10, and 12 dice over all their rolls (fourth column). This is $1 - \frac{c_i}{s_i}$ for each die.
- Once we get to an integer threshold in hits in column four, we remove dice accordingly.

dice	State	$p_{sm}(S)$	Exp hits (cumulative)
15	$(12, 1, 1, 1)$	$(5/6)^{12} * 7/8 * 9/10 * 11/12 = 0.081$	$(2, 0.125, 0.1, 0.083)$
13	$(10, 1, 1, 1)$	$(5/6)^{10} * 7/8 * 9/10 * 11/12 = 0.117$	$(3.67, 0.25, 0.2, 0.166)$
12	$(9, 1, 1, 1)$	$(5/6)^9 * 7/8 * 9/10 * 10/12 = .127$	$(5.17, .375, .3, .33)$
10	$(7, 1, 1, 1)$	$(5/6)^7 * 7/8 * 8/10 * 10/12 = .163$	$(6.33, .5, .5, .67)$
9	$(6, 1, 1, 1)$	$(5/6)^6 * 7/8 * 8/10 * 10/12 = .195$	$(7.33, .625, .7, .84)$
8	$(5, 1, 1, 1)$	$(5/6)^5 * 7/8 * 8/10 * 10/12 = .234$	$(8.17, .875, 1.1, 1)$
5	$(4, 1, 0, 0)$	$(4/6)^4 * 6/8 = .148$	$(9.5, 1.125, X, X)$
3	$(3, 0, 0, 0)$	$(4/6)^3 = .296$	$(10.5, X, X, X)$
2	$(2, 0, 0, 0)$	$(3/6)^2 = .25$	$(11.5, X, X, X)$
1	$(1, 0, 0, 0)$	0	

Figure 10: A random walk down Waldman St.

4.2 Example Results

We have, for this walk, an expected 1.61 occurrences of simultaneous misfortune. Let's estimate that we only lift one die[9] and we pay a cost of between 1 (almost certain for early misses) and 2 (a bad final roll won't, on average penalize us all that much either), so a hand-wavy average penalty of 1.5, for a total of 1.5*1.61=2.415 extra b. (a.d.g.) points for misfortune. Adding this to our ideal 3.307 points gives us 5.722 points as our estimate.

There is no simple means to determine a truly average walk. The only reasonable way to evaluate either with any accuracy is just simulation with our strategy.

5 Simulation

The code for the simulation is at `https://github.com/fettermania/mathnotes/blob/main/bdice/bdice.py`. This an implementation of the algorithm we've implicitly taken for parallel runs:

1. Start with state $S \leftarrow (12, 1, 1, 1)$
2. If state $S = (0, 0, 0, 0)$, return number of points and interruptions and exit
3. Roll all dice in S.
4. If any dice are strictly above the thresholds in Figs. 3 through 6, take them out of S and add their points to the total.
5. Otherwise take the single die which differs from its cutoff by the least[10], remove it, add its points to the total. Add 1 to total interruptions.
6. Go back to step 2.

Results are in figure Fig. 11.

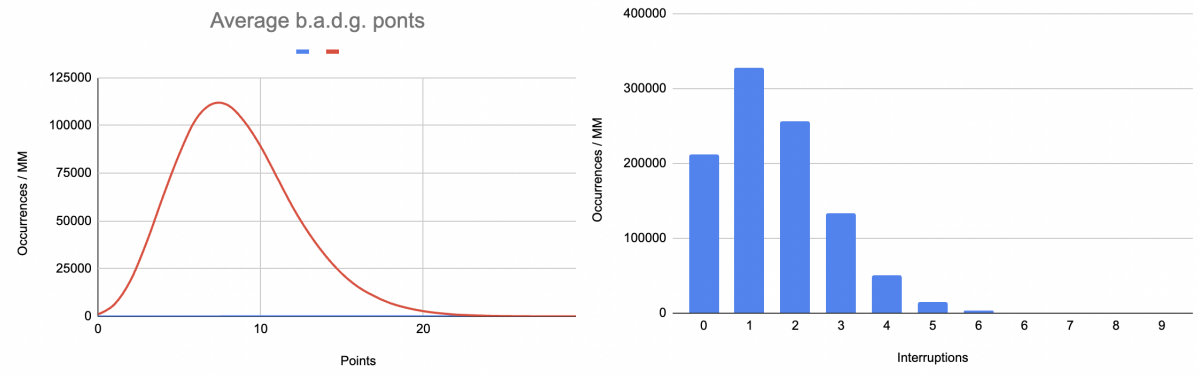
Summary across a million games using the parallel strategy:

- These are naturally right-tailed distributions (you can get 8 interruptions but you can't get -5) but look "normal".[11]
- Our average number of interruptions was **1.55**. This is pretty close to our guess of 1.61 in the Section 4.

[9] Though *maybe* if multiple dice are just a bit worse than their future EV, there's a corner case where lifting many dice under misfortune is best.

[10] Ties broken arbitrarily, meaning we don't suspect this is important.

[11] Stated without proof.



(a) Points

(b) Interruptions

Figure 11: Results: A million walks

- Our chance of getting zero points was 0.11%. This is likely strategy-independent (only happens if you always get a max somewhere in each roll).
- Our chance of getting no interruptions was 21.2%. So, our parallel delusion bubble didn't burst for those runs.
- However, our average score for those runs with no interruptions was 5.67, higher than our EV of 3.31, so *simultaneous fortune does have a real effect*, to the tune of 2.36 points on average. This revised number would bring our Section 4 estimate up to 8.1.
- **Our average number of points was 8.53**. This is higher than our new revised guess, which was based on guessing that the interruptions were about 1.5 points each. This makes their average 1.84 points each.

5.1 Among Friends

As much as b. (a. d. g.) has an objective, it is to beat the score of all of the other players at the table. The payoff structure could be pooling everyone's equal ante and giving it to the player with the lowest score.

This is different than trying to get the best expected score. The payoff for rewarding a low expected score would be more like everyone winning 20 dollars from an infinite pot and giving back their score in dollars.

Though the Blackjack Table in Fig. 7 helps us approximate playing for a low score easily, there are some adjustments we can consider:

- It's possible that, given the phenomena of *simultaneous fortune* and *simultaneous misfortune* of earlier sections, the configuration of Fig. 7 optimizing for the best **mean** may be slightly different. We could search nearby in our simulations to find better results.
- If you're playing to get the best of n players, the configuration in the table is likely not ideal as well, as winning among n requires tuning for higher **variance**.

We put aside the first consideration of *mean* for another time, perhaps. To get a sense of what score tends to win among n players, we run the simulation again in figure Fig. 12.

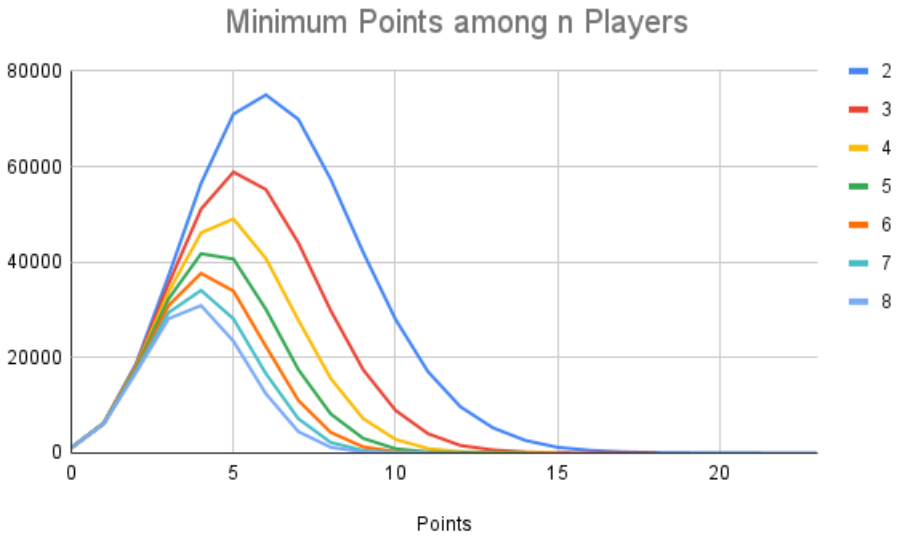


Figure 12: Winning points among N players

players	average minimum score
2	6.4732
3	5.548292
4	4.984596
5	4.58938
6	4.29199
7	4.05768
8	3.864

Figure 13: Average minimum winners

These assume everyone plays off of our table in Fig. 7. Our average score for our "parallel

strategy" is 8.5 and has some sizable variance, so it stands to reason that the winner among two players might have 6 or 7 points, among three players slightly less, and so on.

This of course presumes Larry can put together eight like-minded friends. The heuristics in Figs. 3 through 6 should be good enough for passable performance at the Waldman table, and memorizable enough even for 3-4 rounds of Crown and Schwepps Raspberry-Pomegranate Ginger Ale Zero.