Carole of Mini

Alex Joel Crain and Thomas Mason

April 30, 2018

Consider a two-player game on a directed acyclic graph (DAG).

The edges of the DAG represent moves in the game. The players alternate moves from a node to an adjacent node. For a given DAG G, the game always begins at the same root node r and ends once a leaf node is reached. Each leaf has a different value. All leaf values and paths to the leaves are known to the players at the start of the game. The payoff at the end of the game is the value of the leaf on which the game ends.

Player Paul will receive the payoff, so he wants to reach the leaf of greatest value. Player Carole must pay Paul the payoff, so she wants to reach the leaf of least value. They each know the other's objective, and their strategies account for this: Paul maximizes Carole's minimizations of the available values, and Carole minimizes Paul's maximizations of the available values.

Neither player can benefit from changing strategies while the other player's strategy remains the same. Paul and Carole thus find themselves locked in a Nash equilibrium.

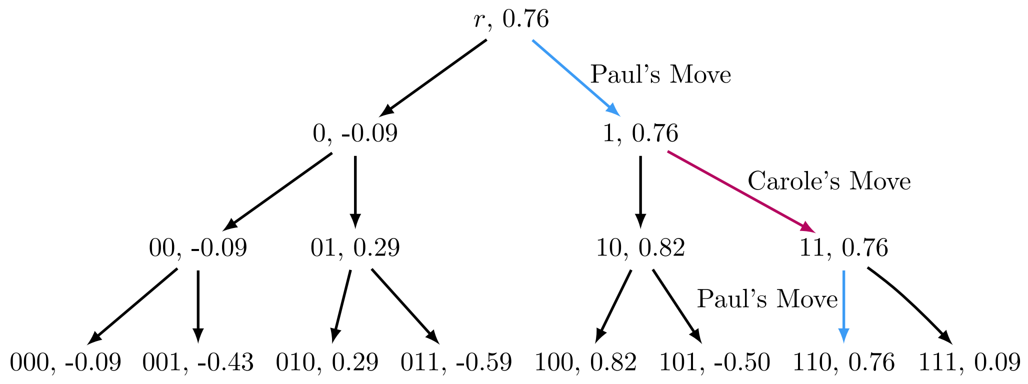

Figure 1: The equilibrial path of a game instance. Paul moves first and gets a payoff of \$0.76. Each position is depicted as ⟨node string⟩, ⟨dollar value⟩.

1 Algorithm

We devised a variant of depth-first search to backward-induct the payoffs for both potential starting players. Let u, v be nodes in a DAG G.

```
DFS-NASH[$u$]
    $MINIMAX[u] \leftarrow \infty$
    $MAXIMIN[u] \leftarrow -\infty$
    FOR $v \in ADJ[u]$
        DFS-NASH[$v$]
        IF $MINIMAX[u] > MAXIMIN[v]$
            $MINIMAX[u] \leftarrow MAXIMIN[v]$
        IF $MAXIMIN[u] < MINIMAX[v]$
            $MAXIMIN[u] \leftarrow MINIMAX[v]$
    IF $ADJ[u] = NIL$
        $MINIMAX[u] \leftarrow VALUE[u]$
        $MAXIMIN[u] \leftarrow VALUE[u]$
```

We assume that every node object w in G has a $MINIMAX$ and a $MAXIMIN$ field, as well as a field ADJ to hold a linked list of pointers to other nodes. If w is a leaf, it must also have a $VALUE$ field.

To obtain the payoff of the game starting at r, the root of G, we run DFS-NASH[r]; when the algorithm is complete, the payoff when Carole moves first is stored in $MINIMAX[r]$ and the payoff when Paul moves first is stored in $MAXIMIN[r]$.

DFS-NASH is $\Theta(E)$, where E is the number of edges in G that can be reached from r. (We rely here on the acyclicity of G—if there are cycles, DFS-NASH will never halt.)

For an n-move game, we initialized G as an array of $2^{n+1} - 1$ node objects. With $G[1] = r$, the adjacency lists and leaf values were assigned like so:

```
GRAPH-ASSIGN[$G, n$]
    FOR $I = 1$ TO $2^n - 1$
        $ADJ[G[I]].APPEND(G[2I + 1])$
        $ADJ[G[I]].APPEND(G[2I + 2])$
    FOR $I = 2^n$ TO $2^{n+1} - 1$
        $VALUE[G[I]] \leftarrow RAND[-1, 1]$
```

where $RAND$ draws a real number at random from a uniform distribution with the specified bounds. We coded the project in Python 3 and have included our pythonic implementations of these algorithms separately from this report.

2 Results

We let the number of moves n range from 1 to 21, inclusive. For each n, we randomized the leaf values 1,000 times; for each randomization, we obtained the payoff when Paul moved first and when Carole moved first.

2.1 First Things First

For each n, we observed that the average payoff when Paul moves first is, as expected, the additive inverse of the average payoff when Carole moves first. If the first player to move is decided by a fair coin flip, this is a fair game. The variance of payoffs shrinks as n increases, as shown below.

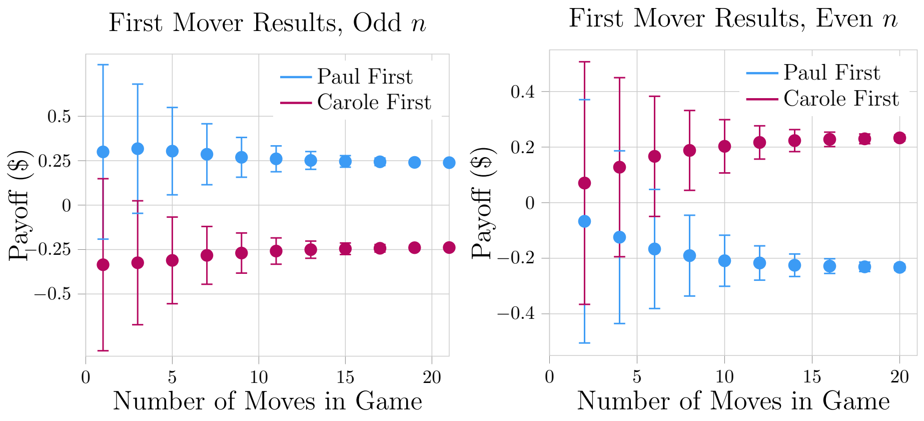

Figure 2: Payoff by who had the first move. Each dot represents the average payoff from 1,000 trials. Error bars give the standard deviation.

2.2 Last-Mover Advantage

The player who gets the *last* move appears to be at a massive advantage. When Paul moves last, the average payoff is *always* positive, and when Carole moves last, the average payoff is *always* negative. The parity of n does matter—if Paul is moving first, he is much better off playing a game with an odd n, so that he gets the last move; if he is moving second, he is better off playing a game with an even n, for the same reason.

Interestingly, from $n = 2$ onward, the average payoff when Paul moves last exhibits a damped oscillatory pattern (the same occurs, reflected, when Carole moves last). This pattern suggests that it also matters who moves first—but only for small n. When n is small and Paul gets the last move but not the first move, he gets a smaller payoff than when he gets both the last and the first move. There are two factors that could contribute to this effect, each of which would attenuate as n grows. One is that the player who moves both first and last gets one more move than the adversary! The other is that having the first move allows for halving the number of reachable leaves.

As n increments, the number of leaves grows exponentially. By the law of large numbers, the first mover is less and less likely to find an appreciably "better half" of leaves as the number of leaves grows exponentially. Concordantly, the advantage of moving first decays. The last move, on the other hand, is always a one-move subgame, regardless of how large n gets; the last-mover advantage does not decay. This explains the saturating disadvantage of moving first at even n: the first-mover advantage decays, and the adversary's last-mover advantage pushes the payoff further from the first mover's goal.

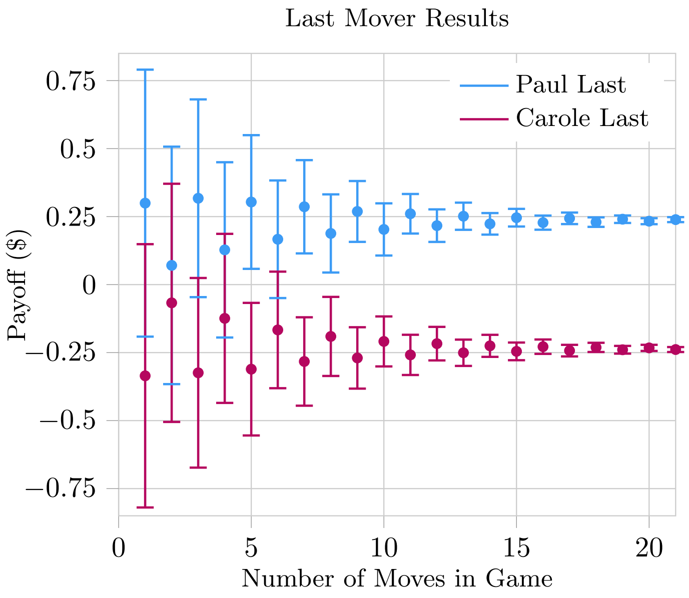

Figure 3: Payoff by who had the last move. Each dot represents the average payoff from 1,000 trials. Error bars give the standard deviation.

Our data suggest that, as $n \to \infty$, the expected payoff when Paul moves last $\mathbb{E}[P(n)] \approx 0.236$, regardless of whether he moves first. Similarly, the expected payoff when Carole moves last $\mathbb{E}[C(n)] \approx -0.236$ as $n \to \infty$. If we assume that Paul and Carole are equally likely to get the last move, then the expected payoff of an n-move game $\mathbb{E}[n] = (\mathbb{E}[P(n)] + \mathbb{E}[C(n)])/2 = 0 \ \forall \ n \in \mathbb{N}$.

In terms of when Paul moves first, the payoff alternates between ~ 0.236 and ~ -0.236 for odd and even n, respectively, with n large. (We obtained 0.236, rounded to 3 digits, by averaging the payoff magnitudes at $n = 20$ with those at $n = 21$. We believe that the lower variance of payoff magnitudes at larger n renders them more reliable indicators of limiting behavior. $n = 20$ provides a lower bound of 0.233 and $n = 21$ provides an upper bound of 0.239.)

2.3 Oh, the Gaussianity!

As n increases, the set of payoff distributions when Paul moves last $P(n)$ and when Carole moves last $C(n)$ takes on a decidedly bimodal character. As Figure 4 illustrates, a larger n leads also to the reduction of variance and of higher-order moments in each of $P(n)$ and $C(n)$. If we assume that this variance reduction continues and converges to 0, we can say that:

$$\lim_{n\to\infty} P(n) = \lim_{n\to\infty} \mathbb{E}[P(n)] \approx 0.236$$

Through the symmetry between the distributions of $P(n)$ and $C(n)$, we see that the distribution of payoffs is zero-centered and normal when the first or last mover is decided by a fair coin flip, even when n is small.

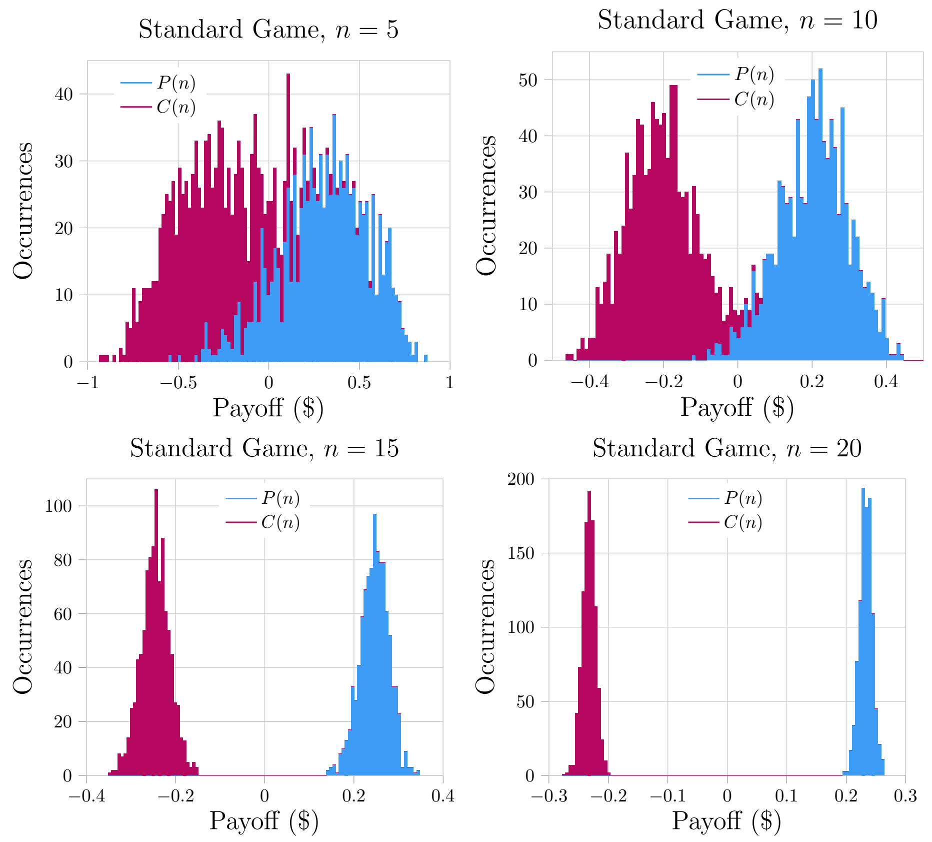

Figure 4: The stacked histograms of $P(n)$ and $C(n)$ exhibit less overlap, lower variance, and greater gaussianity as n increases.

2.4 Putting a Damper on Things

We can do more than approximate $P(n)$ and $C(n)$ as $n \to \infty$: we can obtain a function for $\mathbb{E}[P(n)]$ and for $\mathbb{E}[C(n)]$ at every $n \in \mathbb{N}$ by fitting the damped oscillations from Figure 3. The case where $n = 1$ is trivial (if Paul moves first, he immediately gets to grab the biggest random number), so we ignore it in our function estimation.

Since n is natural, we want the function to have period 2; for the case where Paul moves last, we used $-\cos(\pi n)$. We sought to fit a function of the form,

$$\mathbb{E}[P(n)] = -\cos(\pi n) A e^{-\lambda n} + \lim_{n\to\infty} \mathbb{E}[P(n)],$$

so we had to find the decay rate λ and initial amplitude A. We determined them by subtracting $\lim_{n\to\infty} \mathbb{E}[P(n)] \approx 0.236$ from our mean payoff magnitudes at each n and then fitting an envelope to the logarithm of the difference (using the negative difference at even n). We found $\lambda \approx 0.18$ and $A \approx 0.17$. Since $\mathbb{E}[C(n)] = -\mathbb{E}[P(n)]$, we ballpark both players' payoffs using the same constants.

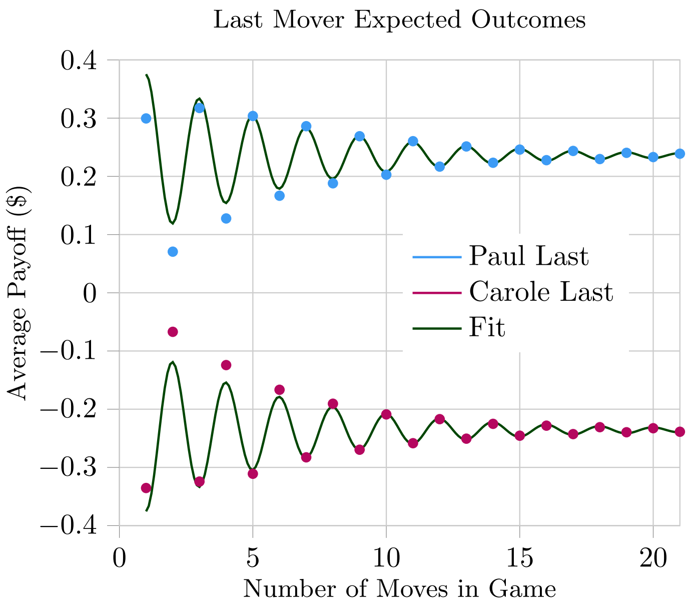

Figure 5: Fit of $\mathbb{E}[P(n)] = -\mathbb{E}[C(n)]$ as damped oscillatory functions, using the same coefficients for both. The expected payoffs are only defined for $n \in \mathbb{N}$; the fitted function is plotted over $\mathbb{R}$ as a visual aide.

While our estimate does not quite capture some high-variance transients at $n \leq 4$, it does decently describe the expected payoffs for $5 \leq n$. To get the expected payoff when Paul moves *first* rather than last, we can just multiply our already fitted function $\mathbb{E}[P(n)]$ by $(2(n \bmod 2) - 1)$.

2.5 All Uniforms Are the Same

In all experiments previously described, we drew leaf values from a uniform distribution in the interval $[-1, 1]$. We wanted to see what would happen if we extended this to $[-2, 2]$ and $[-3, 3]$. In particular, we hoped to find game properties that could be defined relative to the interval length. For each of the distribution intervals $[-2, 2]$ and $[-3, 3]$, for each n, we randomized the leaf values 1,000 times, and, for each randomization, we obtained the payoff when Paul moved first and when Carole moved first.

We found that almost all properties do scale as the ratio of interval lengths, including the average and variance of $P(n)$ at each n, the initial amplitude A, and $\lim_{n\to\infty} \mathbb{E}[P(n)]$. However, we also found something better: an invariant. The decay rate λ stays the same for the game regardless of the interval length.

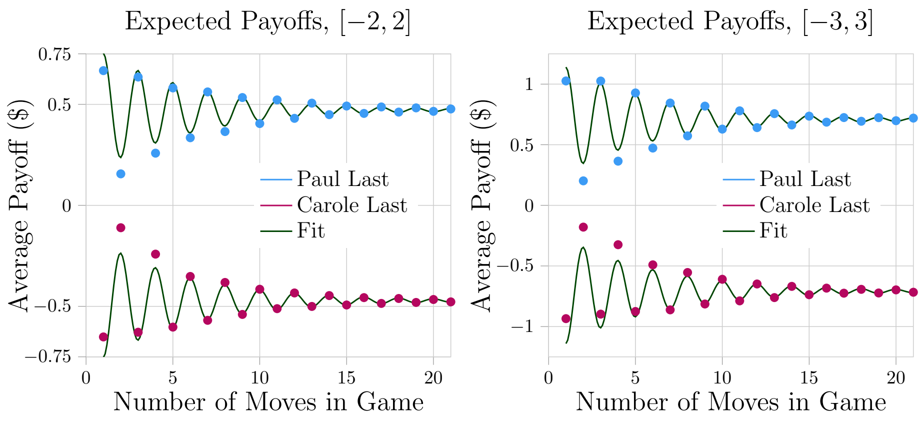

Figure 6: Fit of $\mathbb{E}[P(n)] = -\mathbb{E}[C(n)]$ for the game with random leaf values from intervals $[-2, 2]$ and $[-3, 3]$. Decay rate λ stays ~ 0.18; A scales as the ratio of interval lengths. Each dot represents the average payoff of 1,000 trials.

This experiment justifies some claims about game constants. Let L be the length of the interval on a uniform distribution from which leaf values are drawn. For now, we will assume a zero-centered interval.

$$\lim_{n\to\infty} \mathbb{E}[P(n)] \approx 0.118L \tag{1}$$

$$A \approx 0.085L \tag{2}$$

$$\lambda \approx 0.180 \tag{3}$$

We can use this to estimate the expected payoff of the game at any n for any zero-centered interval. For the expected payoff when Paul moves last, we have:

$$\mathbb{E}[P(L, n)] \approx \left(0.118 - 0.085 \cos(\pi n) e^{-0.18n}\right) L$$

For Carole last, it's $-\mathbb{E}[P(L, n)]$; for Paul first, it's $(2(n \bmod 2) - 1)\, \mathbb{E}[P(L, n)]$; and, for Carole first, it's $(1 - 2(n \bmod 2))\, \mathbb{E}[P(L, n)]$.

We can extend the above to any interval with ease. Let σ be the midpoint of the interval; for example, if we draw leaf values from $[0, 1]$, then $\sigma = 0.5$. Then $\lim_{n\to\infty} \mathbb{E}[P(n)] \approx 0.118L + \sigma$, and:

$$\mathbb{E}[P(L, \sigma, n)] \approx \left(0.118 - 0.085\cos(\pi n)e^{-0.18n}\right) L + \sigma.$$

3 Variations

We devised two variations of the game. The key difference of both from the standard game is that adjacencies are assigned at random. Acyclicity is maintained, but there may be many paths to the same leaf. Further, nodes need not end up in adjacency lists reachable from r, the root of DAG G, or in any adjacency lists at all; such nodes are effectively excluded from the game. For each n, we are no longer guaranteed to have a game with $2^{n+1} - 1$ positions.

In the standard game, the time complexity of `DFS-NASH` was equivalent to $\Theta(V) = \Theta(2^{n+1})$, since the standard game calls for V nodes and $V - 1$ edges. In these variations, its complexity *cannot* be described in those terms, but rather must be considered in terms of the number of randomly assigned adjacencies that can be reached from r.

The two variations differ in the range of nodes randomly assigned to each adjacency list. In the `No-Skip` variant, the game on G still consists of n moves. In the `Go-Skip` variant, the game on G consists of at least 1 and at most n moves.

We adapted `GRAPH-ASSIGN` for the two variations. `DFS-NASH` needed no changes to play each optimally.

3.1 No Skip

Let $RANDINT[X, Y]$ return a random integer between X and Y, inclusive. Let $CHOICE[B, M]$ return a linked list of elements randomly sampled M times from array B without replacement.

For an n-move game, we initialized G as an array of $2^{n+1} - 1$ node objects. With $G[1] = r$, the adjacency lists and leaf values were assigned like so:

```
GRAPH-ASSIGN-NO-SKIP$[G, n]$
    FOR $I = 1$ TO $2^n - 1$
        $COUNT \leftarrow 2^{\lfloor \log_2 I \rfloor + 1}$
        $M \leftarrow RANDINT[1, COUNT]$
        $ADJ[G[I]] \leftarrow CHOICE[G[COUNT, \ldots, 2 \times COUNT - 1], M]$
    FOR $I = 2^n$ TO $2^{n+1} - 1$
        $VALUE[G[I]] \leftarrow RAND[-1, 1]$
```

3.1.1 It's Complicated

`GRAPH-ASSIGN` is $\Theta(2^{n+1})$ since it does $2^{n+1} - 1$ iterations of constant-time operations. `GRAPH-ASSIGN-NO-SKIP`, on the other hand, is $\Omega(2^{n+1})$. In the best case, $RANDINT$ returns 1 at each iteration, and $CHOICE$ only needs to sample 1 element from G. In the worse case, $RANDINT$ returns $2^{\lfloor \log_2 I \rfloor + 1}$ at each iteration, and $CHOICE$ is stuck sampling that many elements at each iteration. $2^{1+\log_2 I} = 2I$, so we have $\sum_{I=1}^{2^n-1} 2I = (2^n - 1)(2^{n+1})/2 = O(2^{2n})$.

In `No-Skip`, the worst-case time complexity for `DFS-NASH` is the same as for `GRAPH-ASSIGN-NO-SKIP`. Since every adjacency assigned in the worst-case scenario will be reachable from r, DFS-NASH is $O(2^{2n})$. However, `DFS-NASH` has a different best case—in fact, it's better than its best case in the standard game! Say $RANDINT$ returns 1 each time and that, for all nodes at each "move depth", $CHOICE$ selects the same node at the next move depth. In that case, only n adjacencies will be reachable from r, so `DFS-NASH` is $\Omega(n)$.

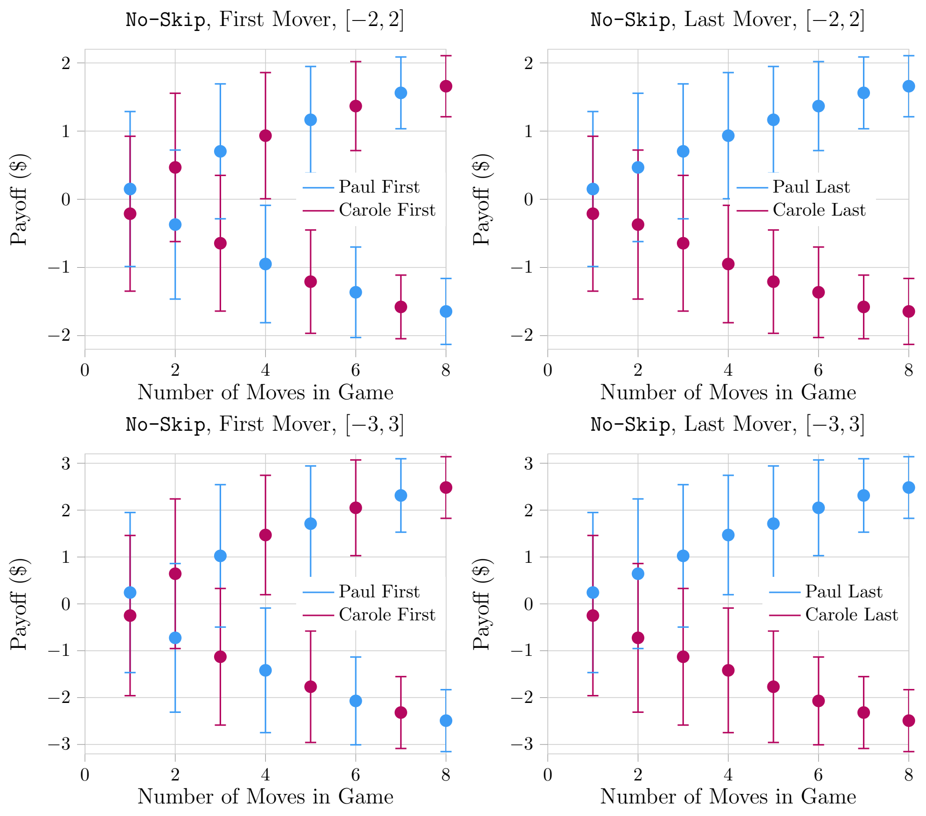

Figure 7: `No-Skip` results by mover and distribution. Each dot represents the average payoff from 1,000 trials. Error bars give the standard deviation.

3.1.2 Last-Mover Omnipotence

We let the number of moves n range from 1 to 8, inclusive. For each n, we randomized the leaf values 1,000 times with a uniform distribution in the interval

$[-2, 2]$; for each randomization, we obtained the payoff when Paul moved first and when Carole moved first. We repeated the process for the interval $[-3, 3]$.

Unlike in the standard game, having the first move in `No-Skip` does not typically allow halving the number of reachable leaves. The random adjacencies to the next move level also give the last mover an even greater advantage: now, instead of only getting to choose between two leaves, the last mover may have a wealth of leaves from which to select.

As n increases, the last mover's expected payoff *appears* to saturate toward a value with magnitude less than 90% that of the best possible leaf value. The variance also begins to shrink at this point, but it is unclear from Figure 7 how much of this is due to clipping at the best possible leaf value versus a symptom of convergence to some lesser value. The histograms answer that question.

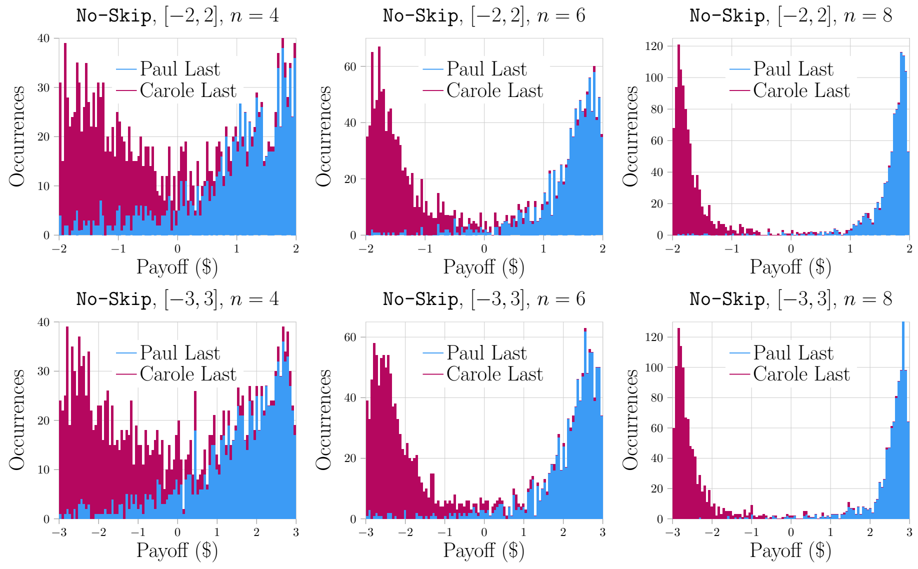

Figure 8: `No-Skip` payoffs by last mover. Payoffs grow more normal with increasing n, but are clipped at the support of the leaf values' dsitribution.

As n increases, the last mover's expected payoff approaches the best possible leaf value (from his or her point of view). The question is whether it will converge to the absolute best or to some fixed distance from it as $n \to \infty$. Even over the limited domain of n that we could test, it is clear that each last mover's payoff distribution becomes a *truncated* normal: you can't get a better value than the best, so there is no room for deviation on that side! This will skew the mean payoff away from the best value. The variance shrinks as n grows, though, so this effect will attenuate.

This experiment motivates an argument from first principles. As $n \to \infty$, the number of nodes reachable by the last mover in `No-Skip` will range over $[1, \infty)$. If we assume that any number in that range is equally likely to be returned by $RANDINT$, we tend toward giving the last mover an infinite number of random leaf values to choose from! Thus, we expect this to converge to the best possible leaf value.

3.2 Go Skip

Where `No-Skip` takes last-mover advantage to the max, `Go-Skip` erodes that advantage entirely. We still parameterize the initialization of DAG G with n, but the game may end up being anywhere from 1 to n moves, depending upon what adjacencies are assigned and who moves first.

Let $RANDINT$ and $CHOICE$ be defined exactly as for `No-Skip`. For an n-parameterized game, we initialized G as an array of $2^{n+1}-1$ node objects. With $G[1]=r$, the adjacency lists and leaf values were assigned like so:

```
GRAPH-ASSIGN-GO-SKIP$[G, n]$
    FOR $I = 1$ TO $2^n - 1$
        $START \leftarrow 2^{\lfloor \log_2 I \rfloor + 1}$
        $M \leftarrow RANDINT[1, 2^{n+1} - START]$
        $ADJ[G[I]] \leftarrow CHOICE[G[START, \ldots, 2^{n+1} - 1], M]$
    FOR $I = 2^n$ TO $2^{n+1} - 1$
        $VALUE[G[I]] \leftarrow RAND[-1, 1]$
```

3.2.1 It's (Also) Complicated

Just like in `No-Skip`, for the best-case complexity of `GRAPH-ASSIGN-GO-SKIP`, $RANDINT$ will return 1 at each iteration and $CHOICE$ will only have to sample 1 element per iteration, so the algorithm is $\Omega(2^{n+1})$. In the worst case, $RANDINT$ will return $2^{n+1}-2^{\lfloor \log_2 I \rfloor+1}$ at every iteration I, forcing $CHOICE$ to look at that many elements. We have $\sum_{I=1}^{2^n-1} 2^{n+1}-2I = 2^{2n}-2n = O(2^{2n})$.

In `Go-Skip`, the worst-case time complexity for `DFS-NASH` is the same as for `GRAPH-ASSIGN-GO-SKIP`. Since every adjacency assigned in the worst-case scenario will be reachable from r, `DFS-NASH` is $O(2^{2n})$. However, `DFS-NASH` has a different best case here—even better than in `No-Skip`! Say that when $I=1$ $RANDINT$ returns 1 and $CHOICE$ then picks a leaf. In that case, only 1 adjacency will be reachable from r, so `DFS-NASH` is $\Omega(1)$.

3.2.2 First Mover's Revenge

We have finally found a game where a first-mover advantage holds no matter the parity of n—further, the advantage increases with n, and at a greater rate than the last-mover advantage in `No-Skip`.

We let the number of moves n range from 1 to 7, inclusive. For each n, we randomized the leaf values 1,000 times with a uniform distribution in the interval $[-1, 1]$; for each randomization, we obtained the payoff when Paul moved first and when Carole moved first. We repeated the process for the interval $[-2, 2]$.

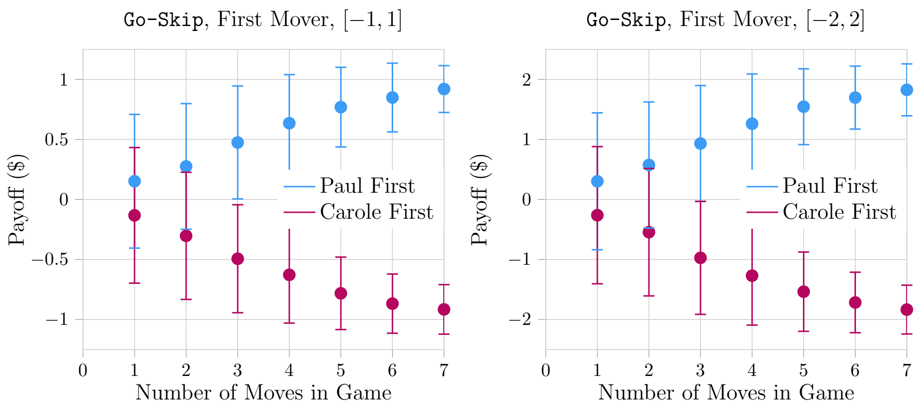

Figure 9: `Go-Skip` payoff by first mover and interval. Each dot represents the average payoff from 1,000 trials. Error bars give the standard deviation.

We observe the same payoff clipping in `Go-Skip` as in `No-Skip`, with the mean converging toward the best possible leaf value for the *first* mover this time. Parity makes little difference here: the first mover is likely to find a way to obtain the last move by skipping moves—sometimes, the last move even is the first move!

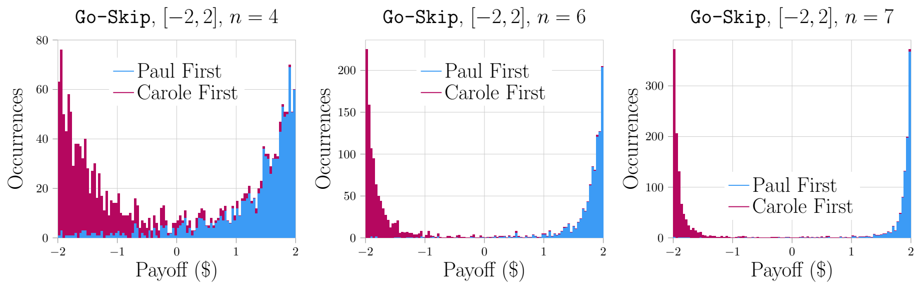

Figure 10: `Go-Skip` payoffs by first mover, played in interval $[-2, 2]$. The payoffs' distributions grow more normal with increasing n, but are clipped at ± 2.

Comparing Figure 10 with Figure 8, it is clear that the first mover's payoff in `Go-Skip` converges faster than the last mover's payoff in `No-Skip`. The "last-mover omnipotence" in `No-Skip`—the idea that the last mover will have infinite leaf choices as $n \to \infty$—also applies to the first mover in `Go-Skip`. However, since the first mover in `Go-Skip` has more paths to an optimal subgame, we here see omnipotence approached faster than in `No-Skip`.

4 Fun Fact

An author of this report was lucky enough to meet John Nash back in 2010.

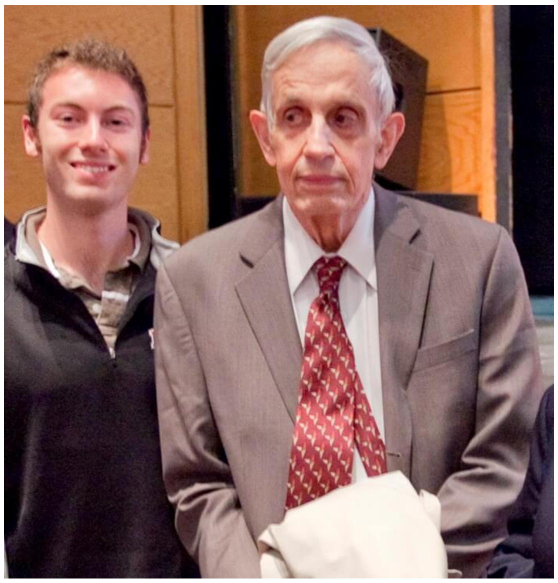

Figure 11: Alex Crain (left) grins as John Nash (right) glances over. Crain is unaware that he'll find Nash's equilibrium in a grad school project 8 years later.

5 Coda

Our implementations of `DFS-NASH` and the graph structures, and other code to run the experiments, are included separately as a program listing. Thanks for a fun project!