Friedrich-Schiller-Universität Jena

seit 1558

The Secretary Problem Applied to Rocket Launch Models

MASTERARBEIT

zur Erlangung des akademischen Grades
Master of Science (M. Sc.)
im Studiengang Wirtschaftsmathematik

Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik

eingereicht von Christian Preßel
geb. am 21.03.1993 in Haßfurt

Betreuer: Prof. Dr. Ingo Althöfer

Jena, 26. Oktober 2016

Abstract

This thesis takes a new approach to a very well known problem of stopping theory: We will take a look at a new kind of secretary problem, which is based on rocket launches.

In the first chapter, we will remind ourselves of the classic secretary problem and build a bridge towards rocket launches with a simple one-player-model. Next, the focus will be on a one-player model where multiple successful launches are required for a mission (Section 3: The n-k-t-model) and examine optimal strategies and their expected return while running into numerical problems.

Then, we move on to a two-player-model in which the players have a different number of rockets (Section 4: Rich vs. Poor) and study the effects of three distinct rule variations. In the fifth section, we explore models inspired by the space travel organization SpaceX, a multi-player model in which players work together as a team, and a two-player-model inspired by the scoring in Judo. In the sixth and final section of this thesis, we will vary the turn order of the two players to examine possible effects on the outcome of games, notably by randomizing the order and examining the model with snake order.

Contents

List of Acronyms

CSP	Classic Secretary Problem
LV	Launch value (expected amount of points achieved by attempting a launch)
NLNW	No Launch, No Win
PSP	Personal success probability
RMS	Regular mission success (Section 5.2)
TMS	Total mission success (Section 5.2)

List of Tables

List of Figures

List of Theorems / Lemmas / Definitions / Examples

1 Introduction

For as long as work and jobs have existed, companies and employers have been dealing with the question of „Who is the best person for this job?“. Nowadays, most large companies have specific departments just to find the best people for their purposes (often called Human Resource or Recruiting departments). Mathematically, this is what we call a selection problem - picking one or a subset of candidates out of a larger group, trying to find the optimal result. A simple problem that is based on this type of question is the Secretary problem, which will be presented in its most classic version in the following section.

But for this thesis, we will quickly move on from picking secretaries and go to space - at least theoretically. Based on the CSP, we will first define a one-player model dealing with rocket launches, which has many similarities to the secretary problem. Then, we will vary this model further to open up a whole new field of selection problems. We are going to examine one-player models, like the one inspired by the innovations of *SpaceX* in Section 5.1 and two-player models, most notably a model with players having a different number of rockets, dramatically named „Rich vs. Poor “ which can be found in Section 4.

The number of rocket launches needed will also vary, in most cases a single successful launch will suffice, but we also discuss a model in which multiple starts are required, like the $n - k - t$-model in Section 3. The possibilities for models of this type are endless - much like space itself. So, let's start by taking a look at the Classic Secretary Problem and then go for some rocket launches.

Note: In the thesis, we will often use „he“ when referring to the player(s). Despite this, all players can be seen as gender-neutral.

2 The Secretary Problem And A Simple Rocket Variant

The secretary problem is one of the most well-known problems in the field of optimal stopping theory. The classic version of this problem is laid out below:

Classic secretary problem

- A recruiter is searching for a new secretary, and has n candidates to fill the vacancy. The recruiter knows n and is able to rank candidates from 1st to nth.
- He calls up the candidates one-by-one, and has to decide whether to pick the current candidate or send him home.
- Once a candidate is sent home, he can no longer be called back to the office.
- Once a candidate is picked, all remaining candidates are sent home.
- The candidates' skill levels are independent and uniformly distributed.
- The recruiter wants to maximize the chance of picking the best candidate.

The optimal solution for this problem is defined by a **stopping rule**: The recruiter lets a certain number r of the n candidates pass to get information - after that he picks the first candidate who is better than those r candidates - or the last one, if none of the $n-r$ candidates is better. The search for the optimal cutoff r is done by calculating the probability of picking the correct candidate while letting the first $r-1$ candidates pass:

$$P(r) = \sum_{i=1}^{n} P(\text{Candidate } i \text{ selected} \cap \text{Candidate } i \text{ is best })$$
$$= \sum_{i=1}^{n} P(\text{Candidate } i \text{ selected} \mid \text{Candidate } i \text{ is best }) \cdot P(\text{Candidate } i \text{ is best})$$
$$= [\sum_{i=r}^{n} P(\text{best of the first } i-1 \text{ cand. is in the first } r-1 \mid \text{Cand. } i \text{ is best})] \cdot \frac{1}{n}$$
$$= \sum_{i=r}^{n} \frac{r-1}{i-1} \cdot \frac{1}{n} = \frac{r-1}{n} \sum_{i=r}^{n} \frac{1}{i-1}$$

This sum isn't defined for $r=1$, but it is clear to see that $P(1) = \frac{1}{n}$, i.e. picking the first candidate yields a 1-in-n chance of picking the best, given uniform distribution. The sum can be approximated by setting x as the limit of $\frac{r}{n}, t = \frac{i}{n}$ and $dt = \frac{1}{n}$.

$$P(x) = x \cdot \int_{x}^{1} \frac{1}{t}\ dt = -x \cdot ln(x)$$

Setting the derivative of P(x) to 0 and solving for x, we gather the optimal cutoff parameter $x = \frac{1}{e}$. Therefore, the recruiter should let approximately $\frac{n}{e}$ (around 37 percent) of candidates pass before picking the first candidate better than those. The probability of finding the best candidate by using this „37 percent rule " also tends towards $\frac{1}{e}$ for large n.

Overcoming the gap between choosing secretaries and launching rockets, we now take a look at a very simple decision problem: An organization is trying to launch a rocket successfully. For this they have a time window of $t \in \mathbb{N}$ days (the organization knows t), with one launch opportunity for each day. They only have one rocket though, and a failed launch would equal the destruction of said rocket and a mission failure. For each day, the probability of a successful launch is given by a variable $x_t \in (0, 1)$, and we define x_i as the probability on the i-last day (e.g. x_1 is the value on the last day of the time window). Comparing this to the CSP, we find several similarities:

- The organization has to make a decision on a day-by-day basis, just like the recruiter has to make a decision for each candidate coming to his office.
- Skipping a day will be the equivalent of sending home a candidate - both can't be brought back.
- Attempting the launch will be the equivalent of picking a candidate - further days / candidates cannot be considered.
- The deciders want to maximize their chances of making the best possible decision (successful launch / best candidate)

For this model, we define x_t as a independent, uniform random number ($x_t \sim U(0, 1)$). We will evaluate the success chances of the organization with E_t (expected success rate with t remaining days). For $t = 1$, we get

$$E_1 = E(x_1) = \frac{1}{2}$$

This is simple: On the last day, there has to be a launch attempt. Therefore, the organization will attempt a launch with any x_1, which has the expected value of 0.5.

$$E_2 = P(x_2 < E_1) \cdot E_1 + P(x_2 \geq E_1) \cdot E(x_2|x_2 \geq E_1) = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{3}{4} = \frac{5}{8}$$

Here, the decider sees x_2. If it is higher than the expected return for waiting for $t = 1$, he will attempt a launch - getting the conditional expectation $E(x_2|x_2 > E_1)$ as a return, otherwise he waits for the final day. For any $t \geq 2$, the recursion formula is the same:

$$\begin{aligned} E_t &= P(x_t - 1 < E_{t-1}) \cdot E_{t-1} + P(x_t \geq E_{t-1}) \cdot E(x_t|x_t \geq E_{t-1}) \\ &= E_{t-1} \cdot E_{t-1} + (1 - E_{t-1}) \cdot \frac{1 + E_{t-1}}{2} \\ &= \frac{1 + E_{t-1}^2}{2} \end{aligned}$$

This is a very simple recursion formula, and it's fairly obvious that $E_t \to 1$ for large t, as $\frac{1 + x^2}{2}$ is monotonically increasing for $x \geq 0$, and $E(x_t)$ has a maximum value of 1.

This is the model that is most similar to the classic secretary problem, the models in the next sections are all a bit more complex and sometimes less comparable to the CSP.

3 The n-k-t Rocket Model

3.1 Problem and Modeling

The first proper model examined in this thesis is that of a single player launching multiple rockets: A space travel organization is planning a mission which involves a total of k successful rocket launches. Knowing that some attempts might fail, they have a certain number of rockets $n \geq k$. The time window for the mission consists of t days (usually $t \geq n$), with a maximum of one launch attempt per day. n, k, t are all natural numbers and for later calculations, we only consider $n, k, t \geq 1$.

Definition 3.1. (Launch Success Probability)
In the model, the probability for a successful launch on a certain day is given by a random number $x_i \in (0, 1)$. Therefore, the probability of a failed launch attempt,which equals the loss of the rocket, is given by $1 - x_i$. For clarity, we define that x_i refers to the success probability on the ith-to-last day, e.g. x_1 being the probability of a successful launch on the final day.

The organization is aware of the current value of x_i and will decide whether to attempt a launch or wait for a better day. Now we take a look at the distribution of the x_i: In the simplest version of the model, we will set them as independent, uniformly distributed numbers, i.e. $x_i \overset{iid}{\sim} U(0, 1)$.

Definition 3.2. (Mission Success/Failure)
The mission is a success if the organization manages to successfully launch the required number of rockets before time runs out. The organization will receive 1 point for a successful mission.

The mission is a failure if time runs out or the rocket supply is destroyed before the required number of successful launches, e.g. $n = 0$ or $t = 0$ while $k > 0$. The organization will get no points in this case.

To have a succinct notation for the upcoming recursion formulas, we also define the following two values:

Definition 3.3. (Expected Mission Success, Decision Parameter)
*$E(n, k, t)$ describes the **expected mission success probability** given n rockets, k successful launches needed and t remaining days. $E(n, k, t) \in (0, 1)$. It can also be seen as the expected points the organization will receive.*

*$S(n, k, t) \in [0, 1]$ is the **decision parameter** the player makes his launching decision. If x_t is larger than this value, he will launch a rocket. For smaller x_t, he will choose to not attempt a launch and wait for the next day.*

As we have now defined all relevant notations and rules for this model, we will now examine the recursion for the organizations optimal strategy, and evaluate the results.

3.2 Recursion for Independent Uniform x_i

We examine the recursion and mission success probability in case the organization makes optimal decisions. We want to calculate the mission success probability for any given numbers $n, k, t \in \mathbb{N}$, particularly for $t \geq n \geq k$. Some values are fairly obvious: For $n < k$ and $t < k$, the value $E(n, k, t)$ is 0 because there are not enough rockets or not enough time respectively. Also, we can easily see that

$$E(n, k, t) = (\frac{1}{2})^t, \text{ for } k = t.$$

Here, the organization is launching a rocket every day regardless of x_i, because time is running out. They would need $k = t$ successful launches in a row for an overall mission success. Another situation that can be easily calculated is $(1, 1, 2)$:

$$\begin{aligned} E(1,1,2) &= P[x_2 \leq E(1,1,1)] \cdot E(1,1,1) + P[x_2 > E(1,1,1)] \cdot E(x_2|x_2 > \frac{1}{2}) \\ &= \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{3}{4} = \frac{5}{8} \end{aligned}$$

Here, the organization either attempts a launch on the penultimate day, if $x_2 > \frac{1}{2}$, or on the final day with an expected success rate of $\frac{1}{2}$. In general, the organization will attempt a launch rocket if the expected points for making the attempt are higher than for not making a launch attempt/skipping the day.

For programming the recursion, we will have to look at four different cases before we formulate the general recursion:

Case 1: $k = t = 1$
In this case, we are left with the final day and need to have a successful launch. The probability is equal to the expected value of a random $x_1 \sim U(0, 1)$, $\frac{1}{2}$.
Therefore we get $E(n, 1, 1) = \frac{1}{2}$ for all $n \geq 1$.

Case 2: $k = n = 1$
In this case, we have one rocket and need one successful launch. By Case 1, we know that $E(1, 1, 1) = \frac{1}{2}$, and given $E(1, 1, t-1)$, we can calculate

$$\begin{aligned} E(1,1,t) &= P[x_t \leq E(1,1,t-1)] \cdot E(1,1,t-1) + P[x_t > E(1,1,t-1)] \cdot \frac{1 + E(1,1,t-1)}{2} \\ &= E(1,1,t-1) \cdot E(1,1,t-1) + [1 - E(1,1,t-1)] \cdot \frac{1 + E(1,1,t-1)}{2} \\ &= \frac{1}{2} + \frac{1}{2} \cdot E(1,1,t-1)^2. \end{aligned}$$

This case is equal to the single-player model we briefly discussed in Section 2.

Case 3: $n \geq t$
While we are mostly interested in $n \leq t$, this case is also relevant to the general recursion (and programming). In this case, we have at least as many rockets as days. Because there is no added value in having rockets left over, the decider will simply attempt a launch every day, regardless of the value x_i:

$$E(n,k,t) = \begin{cases} 1-(\frac{1}{2})^t & k=1 \\ \frac{1}{2}\cdot(E(n-1,k-1,t-1)+E(n-1,k,t-1)) & k>1 \end{cases}$$

Case 4: $n > k = 1, n < t$

$E(n,1,t) = P[x_t \leq E(n,1,t-1)] \cdot E(n,1,t-1)$

$+P[x_t > E(n,1,t-1)] \cdot [\frac{1+E(n,1,t-1)}{2} \cdot 1 + [1-\frac{1+E(n,1,t-1)}{2}] \cdot E(n-1,1,t-1)]$

$= E(n,1,t-1)^2 + [1-E(n,1,t-1)] \cdot [\frac{1+E(n,1,t-1)}{2} \cdot 1 + [1-\frac{1+E(n,1,t-1)}{2}] \cdot E(n-1,1,t-1)]$

After preparing with these cases, we can now focus on the general recursion for $n,k,t > 1, k \leq n$ (enough rockets), $k \leq t$ (enough days) and $n < t$. First off, we take a look at the decider's two options:

- The current x_t is too small, and he chooses to not attempt a launch today. Therefore, the value of $E(n,k,t-1)$ is achieved. To keep the formula short, we will rename this value **w** („wait").
- Upon seeing x_t, the decider chooses to make a launch attempt. With a probability x_t this attempt is successful and $E(n-1,k-1,t-1)$ is achieved (renamed **s**, „successful launch"), with a probability of $1-x_t$ the attempt fails, the rocket is lost, and $E(n-1,k,t-1)$ is achieved (renamed **f**, „failed launch").

Thus, the decider chooses the maximum of the known success rate of waiting w, and the expected success rate in case of a launch attempt. The latter is also equal to the previously defined decision parameter and can be calculated as

$S(n,k,t) = x_t \cdot E(n-1,k-1,t-1) + (1-x_t) \cdot E(n,k-1,t-1) = x_t \cdot s + (1-x_t) \cdot f.$

$S(n,k,t)$ is strictly increasing in x_t because of $s > f$. For $E(n,k,t)$, we therefore receive the following general recursion formula:

$$E(n,k,t) = P[S(n,k,t) < w] \cdot w + [P(S(n,k,t) \geq w] \cdot S(n,k,t)$$

We now look for the optimal value x^*, for which waiting and attempting a launch yield the same (expected) success probability. For all $x_t > x^*$, a launch attempt is better than waiting.

$w = x^* \cdot s + (1-x^*) \cdot f \Leftrightarrow x^* = \frac{w-f}{s-f} = P(S(n,k,t) < w)$, resulting in

$$\begin{aligned} E(n,k,t) &= \frac{w-f}{s-f} \cdot w + (1-\frac{w-f}{s-f})(\frac{1+\frac{w-f}{s-f}}{2} \cdot s + \frac{1-\frac{w-f}{s-f}}{2} \cdot f) \\ &= \frac{-w^2+2wf-s^2}{2(f-s)} \end{aligned}$$

with $w = E(n,k,t-1), s = E(n-1,k-1,t-1), f = E(n-1,k,t-1)$.

3.3 Numerical Instability and Results

The recursion was first programmed in Scilab, the three parameters of *RocketNKT (nmax,kmax,tmax)* indicating the maximum n, k, t to which E is calculated. Now, testing it for $nmax = kmax = tmax$ quickly showed a problem. For $tmax = 37$, the calculation still worked without prompting an error, but in $t = 38$, there was the first division by 0. The first (n, k, t) in which this happened was (36,3,38), which is part of the general recursion. To understand what happened, let's look at how it is calculated:

$$E(36,3,38) = \frac{-E(36,3,37)^2 + 2 \cdot E(36,3,37) \cdot E(35,3,37) - E(35,2,37)^2}{2 \cdot [E(35,3,37) - E(35,2,37)]}$$

Division by zero occurs due to both $E(35,3,37)$ and $E(35,2,37)$ equaling 1 in Scilab's double-precision calculation. The difference is thus zero, and the program ends with an error.

We can still calculate accurate values for larger t, if we limit *nmax* and *kmax* to smaller values. Assuming $nmax = kmax$, here's how far the program runs without errors:

Table 3.3.1: Maximum t with successful calculation, n-k-t model

nmax/kmax	5	10/15/20	25	30	32
tmax	40485	32908	345	47	32
(n, k, t) with 1st error	(5,2,40486)	(6,2,32907)	(25,2,346)	(29,3,48)	(32,2,33)
First error	$E > 1$	$E > 1$	$E > 1$	$E > 1$	$E > 1$

We see that the „division by 0“ error is not really the first one to occur during calculation - before this, there are already values greater than 1, which is of course not possible by definition of the model. This is most likely due to rounding inaccuracies during the division while calculating the general recursion, which can be seen in Table 3.3.1 as well (the first values that excess 1 are all ones calculated in the general formula). The division by 0 error is probably connected to these inaccuracies as well. If we add another if-clause to alter values greater than 1 to be *exactly* 1, we will run into division by zero quite quickly as well.

Another implementation
Scilab uses the data type *double* for its calculations, which is precise enough for many applications. In our case, it can only be considered satisfying if we aren't too interested in large combinations of n, k, t. Therefore, a second implementation of the recursions was done in Java, which has a data type named *BigDecimal*, that basically allows infinite precision (for division, there are multiple rounding mode options.) Sadly, this is traded for a vast increase in calculation time, which makes it hardly useful as well - and it's still unclear what rounding mode will not get problematic for large n, k, t.

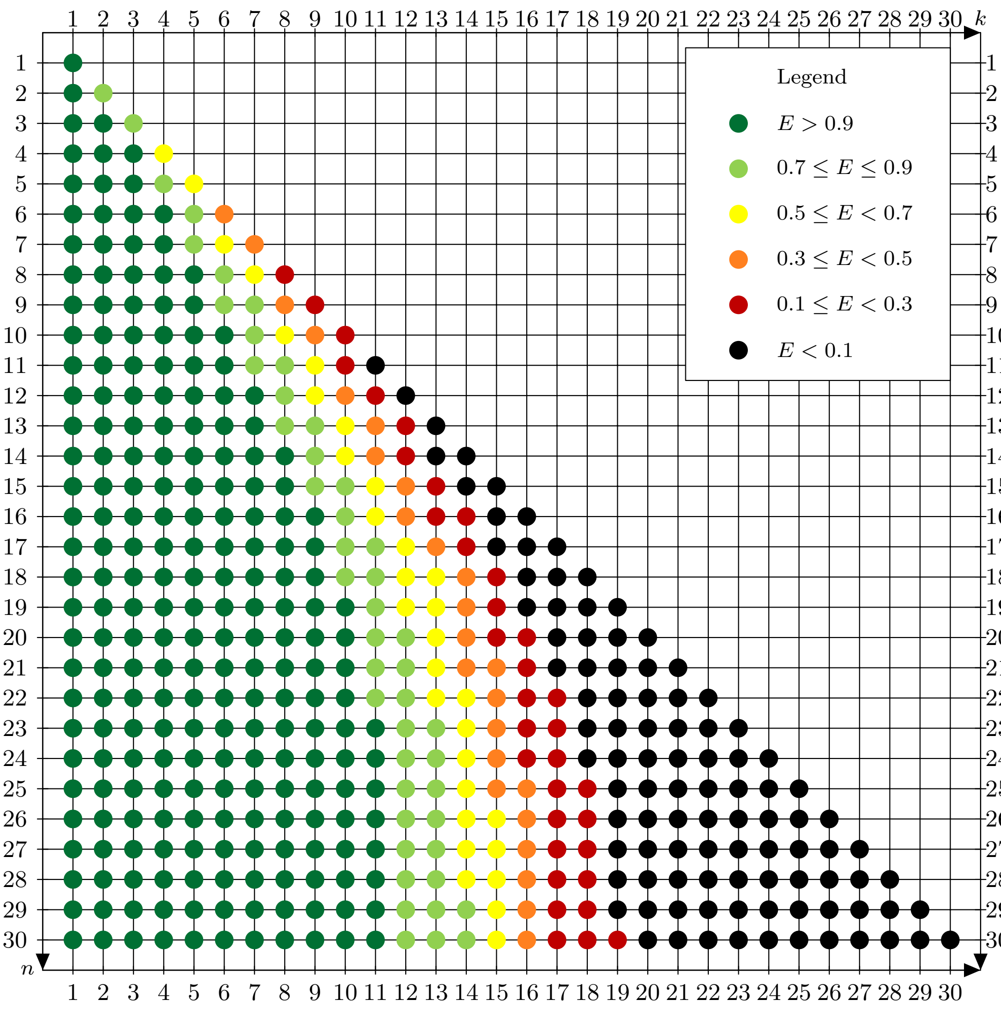


Figure 3.3.1: Visual representation of $E(n, k, 30)$

Some results:
In Figure 3.3.1 one can see a visualization of the results for $t = 30$. We see that in many of the (n, k) the expected success rate is either larger than 90 or less than 10 percent, with only about a third of the k's delivering an expected success rate in $(0.1, 0.9)$ per each n. It seems that for $n = t$, the following pattern can be deducted:

$$E(n, k, t) > 0.9 \text{ for } k \le \frac{n}{3}, E(n, k, t) < 0.1 \text{ for } k \ge \frac{2n}{3}$$

This works for $n = 3 \cdot i, i = 2, ..., 10$. It is unclear whether this continues for larger n as well, as the rounding inaccuracies begin to pop up at $n = k = t = 33$.
Another interesting case to examine is the one of $n = k$, i.e. a scenario in which all remaining rockets have to be launched successfully. Now, for $t = k$, we know that $E(n, k, t)$ simply becomes $(\frac{1}{2})^t$, equal to the probability of guessing the results of t fair coin-flips correctly. But what happens if the player has more time on his hands? This was calculated for $E(5, 5, t), E(10, 10, t), E(15, 15, t)$ and $E(20, 20, t)$, where we looked for the t in which certain barriers were passed. Each cell denotes the t in which $E(n, k, t)$ is larger then a barrier b. We chose $b = 0.5, 0.6, 0.7, 0.8, 0.9, 0.95, 0.99$ for this. The results are below in Table 3.3.2:

Table 3.3.2: Barriers in nkt-model, $n = k$

b \ (n,k)	(5,5)	(10,10)	(15,15)	(20,20)
≥ 0.5	28	97	205	350
≥ 0.6	39	133	279	476
≥ 0.7	57	193	402	684
≥ 0.8	93	311	646	1,097
≥ 0.9	201	664	1,375	2,330
≥ 0.95	417	1,369	2,831	4,794
≥ 0.99	2,152	7,017	14,478	24,500

If, for example, the player has to launch all his 20 rockets to succeed, and wants to ensure he has an 80% chance of doing so, he needs at least 1,097 days - which pretty much equals three years. To have a 90% success chance, this rises to over six years already. On the other hand, a player who has to launch all five of his rockets successfully already has a larger than 50% chance if he only has four weeks to do so. For comparison, let's look at the barriers when the player has $n = 5, 10, 15, 20$ rockets, but only has to start $k = 4, 9, 14, 19$ of them:

Table 3.3.3: Barriers in nkt-model, $n = k + 1$

b \ (n,k)	(5,4)	(10,9)	(15,14)	(20,19)
≥ 0.5	10	39	83	143
≥ 0.6	13	48	102	175
≥ 0.7	17	61	129	221
≥ 0.8	24	83	175	298
≥ 0.9	40	134	278	469
≥ 0.95	67	210	426	715
≥ 0.99	217	572	1,104	1,807

We see that for our previous example, the player with 20 rockets who now only needs to launch 19 successfully, will only need around ten moths to have an 80% success rate. The barriers for having a $> 99\%$ chance have been cut down by a factor larger than 10 as well. Having one „free“ failed start therefore helps the player dramatically if he wants to ensure a high success rate. Finally, lets look at the case when the player only needs to launch half his rockets, where the effect is even stronger:

Table 3.3.4: Barriers in nkt-model, $n = 2k$

b \ (n,k)	(4,2)	(10,5)	(16,8)	(20,10)
≥ 0.5	3	9	15	19
≥ 0.6	4	10	17	21
≥ 0.7	5	11	18	22
≥ 0.8	6	13	20	24
≥ 0.9	12	18	24	28
≥ 0.95	22	24	29	33
≥ 0.99	104	82	46	46

4 A Two-Player Model: Rich vs. Poor

In this section, we will explore a model in which two players are competing against each other. The rules of the game are as follows:

- Each player has a certain amount of rockets available. Player 1 has got n_1 rockets, while Player 2 has n_2 rockets. If $n_1 > n_2$, we will refer to Player 1 as the „rich“ player (or short „Rich“) and to Player 2 as the „poor“ player (short: „Poor“)
- The players take alternating turns as time counts down from a fixed $T \in \mathbb{N}$. Every day, one of the players has the opportunity for a launch attempt. If the launch attempt fails, the rocket is lost.
- The probability for a successful launch on the t-last day will be $x_t \sim U(0,1)$, independent for all t.
- The winner of the game receives 1 point, the loser no points. Both players want to maximize their expected points.

There are three possible ways of the game ending in the base model:

1. One of the players launches a rocket successfully. He wins the game (1 point), the opponent loses (0 points).
2. One player wastes all his rockets without a single successful launch. He loses the game (0 points), the opponent wins 1 point.
3. Time runs out on the players, i.e. t reaches 0. To give them incentive to make launch attempts, we declare them both losers in this scenario. By this, players should always launch rockets when time is running out. Any player who is in $t = 1$ in the basic scenario has to launch his rocket with any given x_1.

By these constraints, it is clear that this is not a constant sum game, especially if n_1 or n_2 is larger then t, where it is possible that both players receive no points because they fail to launch within time (although this will almost surely not happen for large T). If both n_1 and n_2 are smaller than $\lfloor \frac{t}{2} \rfloor$, both players should be forced to use up their rocket supply in time, as there is no additional reward for saving rockets.

Possible rule variants:

- **Lucky loser:** When one player launches a rocket successfully in $t > 1$, the other player has one more chance of launching the rocket on the following day, to get a certain fraction of the point $p \in (0,1)$ (if he is successful, the player with the first launch will receive $1 - p$ points instead of 1). It seems sensible to suggest that $p \leq \frac{1}{2}$.
- **No launch, no win:** When one player wastes his rockets, the other player has all remaining days to try a successful launch. If he doesn't get a successful start either, both lose as stated in the third ending.
- **Consolation points:** If time runs out, both players get a fixed $c \in (0,1)$ points. $c < \frac{1}{2}$ should be a reasonable assumption.

With these three variants to the base model, and given they can be combined with each other, we will have a total of (at least) eight different scenarios to examine in this section. Mostly of interest is the expected points for both players if t is large and both players play optimally. We will also look for possible anomalies or strange situations the players might face under certain rule sets, and evaluate the advantage Rich has over Poor.

By using recursion, we will calculate the expected points in any scenario, defining the following:

Definition 4.1. (Expected points in the Rich vs. Poor model)
The value $E_d(n_1, n_2, t)$ gives the expected points in every situation. The d stands for the player currently having the launch opportunity, thus $d \in \{1, 2\}$. n_1 and n_2 are the amount of rockets both players have left, and t represents the number of remaining days.

Similar to the decision parameter in Section 3, we also define:

Definition 4.2. (Decision parameter in the Rich vs. Poor model)
The parameter $s_d(n_1, n_2, t)$ is the ***decision parameter*** *for player d, when the players have n_1, n_2 rockets and t days left. Optimal players will make a launch attempt for any x_t larger than this value, and skip the day else.*

In each of the eight scenarios, we will go through all necessary cases to determine the recursion formula. We will find that in some scenarios, we can use the same formulas as calculated in previous scenarios. To illustrate the decision-making of the players, there will also be a selection of decision trees in each section.

Note: Each of the eight scenarios was programmed in Scilab separately, though most of the *RichVsPoorX* programs are redundant. For running all eight scenarios laid out in this thesis, only *RichVsPoor6* and *RichVsPoor8* are really necessary. With *RichVsPoor6* one can also analyze Scenario 1 ($p = 0, c = 0$), Scenario 2 ($c = 0$) and Scenario 4 ($p = 0$). With *RichVsPoor8* it is possible to analyze Scenario 3 ($p = 0, c = 0$), Scenario 5 ($c = 0$) and Scenario 7 ($p = 0$), too. With a bit more programming effort, there is obviously the possibility of having a „one-for-all“ program to run all eight scenarios.

4.1 Scenario 1: Standard Rule Set

In this scenario, we will use the rules laid out in the beginning of the chapter, without any variations. Whoever launches the first rocket successfully will gain the full point, and if neither of the players get into orbit, there will also be no consolation points. To determine the recursion, let's look at $t = 1$ first:

$$E_1(n_1, n_2, 1) = E_2(n_1, n_2, 1) = \frac{1}{2} \qquad \text{for all } n_1, n_2 \geq 1$$

With only one day left, both players are forced to make a final launch attempt (otherwise they are guaranteed to get 0 points). Therefore, they're expected points is equal to the expected value of x_1, which is uniformly distributed on $(0, 1)$, giving them half a point each on average. For $t = 2$, we will look at the decision tree for Rich:

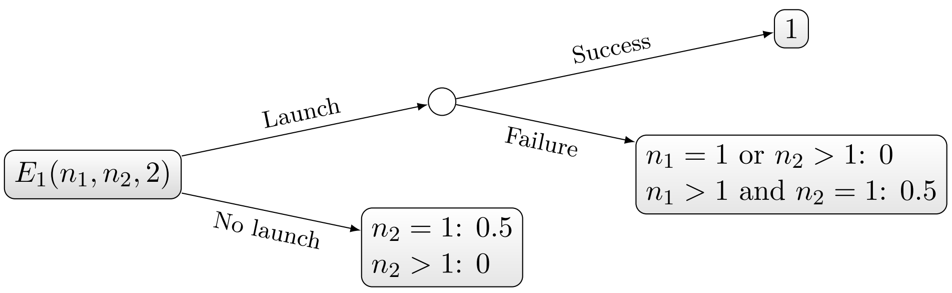


Figure 4.1.1: Decision tree, Scenario 1, $t = 2$

In case Rich attempts a launch in $t = 2$, he will receive 1 point if the launch is successful. If the attempt fails, he will receive 0 points in case he destroyed his last rocket or Poor not being at risk of destroying.*his* last rocket. If Poor only has one rocket left, Rich still has a 50% chance of getting the point when Poor makes his final attempt in $t = 1$. If Rich doesn't launch a rocket, he can only get points if Poor only has one rocket left as well. If Poor has more than one rocket remaining and thus can't destroy his rockets before time runs out, Rich has to attempt a launch given any x_2, therefore

$$E_1(n_1, n_2, 2) = \frac{1}{2} \text{ for all } n_2 > 1, \text{ and } E_2(n_1, n_2, 2) = \frac{1}{2} \text{ for all } n_1 > 1$$

Should Poor have only one rocket left, Rich has two winning options: Making his own successful launch in $t = 2$, or waiting for Poor's forced(!) launch attempt with his final rocket in $t = 1$, which fails half the time. If Rich has only one rocket left, he will make a launch attempt if x_2 is greater than $\frac{1}{2}$, i.e. the expected points if he does not launch a rocket and lets Poor make a forced attempt on the last day. Thus, we get

$$\begin{aligned}&E_1(1, 1, 2) = P[x_2 < 1 - E_2(1, 1, 1)] \cdot [1 - E_2(1, 1, 1)] \\ &+ P[x_2 \geq 1 - E_2(1, 1, 1)] \cdot E[x_2 | x_2 \geq 1 - E_2(n_1, 1, 1)] = \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{3}{4} = \frac{5}{8}\end{aligned}$$

and the analogous result $E_2(1, 1, 2) = \frac{5}{8}$ for Poor. If Rich has at least two rockets, he will simply launch with any x_2 because a failed launch will not lose him the game (yet). He therefore receives

$E_1(n_1,1,2) = \frac{1+0.5}{2} = \frac{3}{4}$ for $n_1 > 1, E_2(1,n_2,2) = \frac{3}{4}$ for $n_2 > 1$.

Next, we take a look at situations in which one of the players has only one rocket remaining, with the other player having at least two rockets left - this and all of the following cases are examined for $t > 2$.

In the first two cases, a failed launch attempt by the active player leads to an instant loss. The decision tree for both Rich and Poor in this case can be found in Figure 4.1.2 just below. The values s_1, s_2 on which basis the player decides if he attempts to launch a rocket is equal to the expected value of „no launch“. A higher x_i will lead to an attempt, otherwise the player will skip the day:

$$s_1 = s_1(1,n_2,t) = 1 - E_2(1,n_2,t-1) \text{ and } s_2 = s_2(n_1,1,t) = 1 - E_1(n_1,1,t-1)$$

For the expected points, we receive

$$E_1(1,n_2,t) = s_1^2 + (1-s_1)\cdot(\frac{1+s_1}{2}) \text{ and } E_2(n_1,1,t) = s_2^2 + (1-s_2)\cdot(\frac{1+s_2}{2})$$

where $\frac{1+s_i}{2}$ is the conditional expectation of x_t, $E(x_t|x_t \geq s_i)$.

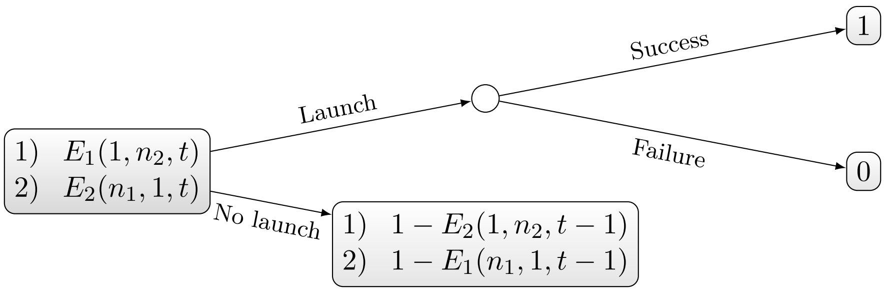


Figure 4.1.2: Decision tree, Scenario 1, active player with single rocket

In the other two situations, a failed launch still gives the active player a chance for an overall win (see Fig. 4.1.3). For the decision parameter, we demand:

$$\begin{aligned} 1 - E_2(n_1,1,t-1) = & \quad s_3 \cdot 1 + (1-s_3)\cdot(1-E_2(n_1-1,1,t-1)) \\ \Leftrightarrow \quad s_3 = & \quad \frac{E_2(n_1-1,1,t-1) - E_2(n_1,1,t-1)}{E_2(n_1-1,1,t-1)} \end{aligned}$$

for 1) and

$$\begin{aligned} 1 - E_1(1,n_2,t-1) = & \quad s_4 \cdot 1 + (1-s_4)\cdot(1-E_1(1,n_2-1,t-1)) \\ \Leftrightarrow \quad s_4 = & \quad \frac{E_1(1,n_2-1,t-1) - E_1(1,n_2,t-1)}{E_1(1,n_2-1,t-1)} \end{aligned}$$

The expected points for Rich and Poor can then be calculated as:

$$\begin{aligned} E_1(n_1,1,t) = & \quad s_3 \cdot (1 - E_2(n_1,1,t-1)) \\ + & \quad (1-s_3)\cdot[\frac{1+s_3}{2}\cdot 1 + \big(1 - \frac{1+s_3}{2}\big)\cdot(1-E_2(n_1-1,1,t-1)\big] \\ E_2(1,n_2,t) = & \quad s_4 \cdot (1 - E_1(1,n_2,t-1)) \\ + & \quad (1-s_4)\cdot[\frac{1+s_4}{2}\cdot 1 + \big(1 - \frac{1+s_4}{2}\big)\cdot(1-E_1(1,n_2-1,t-1)\big] \end{aligned}$$

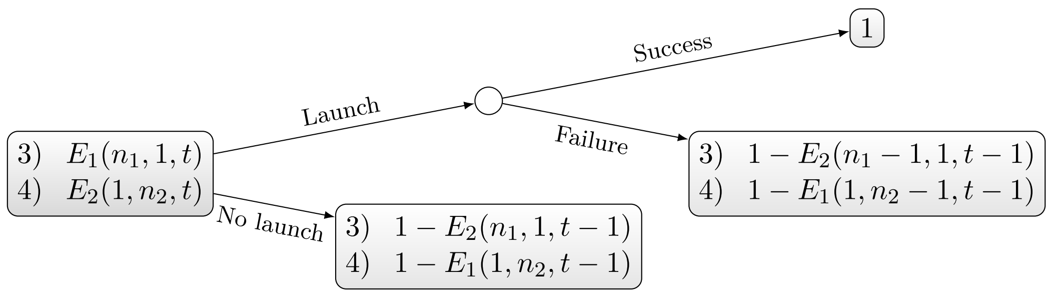


Figure 4.1.3: Decision tree, Scenario 1, passive player with single rocket

Finally, we can also look at the general recursion for all other scenarios, i.e. $n_1, n_2 > 1, t > 2$, the decision tree can be found below. For the decision parameters, we demand:

$$\begin{aligned}
1-E_2(n_1,n_2,t-1) = \quad & s_5 \cdot 1 + (1-s_5)\cdot(1-E_2(n_1-1,n_2,t-1)) \\
\Leftrightarrow \quad & s_5 = 1 - \frac{E_2(n_1,n_2,t-1)}{E_2(n_1-1,n_2,t-1)} \\
1-E_1(1,n_2,t-1) = \quad & s_6 \cdot 1 + (1-s_6)\cdot(1-E_1(n_1,n_2-1,t-1)) \\
\Leftrightarrow \quad & s_6 = 1 - \frac{E_1(n_1,n_2,t-1)}{E_1(n_1,n_2-1,t-1)}
\end{aligned}$$

and the expected points are therefore

$$\begin{aligned}
E_1(n_1,n_2,t) = \quad & s_5 \cdot (1-E_2(n_1,n_2,t-1)) \\
+ \quad & (1-s_5)\cdot\Big[\frac{1+s_5}{2} + \big(1-\frac{1+s_5}{2}\big)\cdot(1-E_2(n_1-1,n_2,t-1)\Big] \\
E_2(n_1,n_2,t) = \quad & s_6 \cdot (1-E_1(n_1,n_2,t-1)) \\
+ \quad & (1-s_6)\cdot\Big[\frac{1+s_6}{2} + \big(1-\frac{1+s_6}{2}\big)\cdot(1-E_1(n_1,n_2-1,t-1)\Big]
\end{aligned}$$

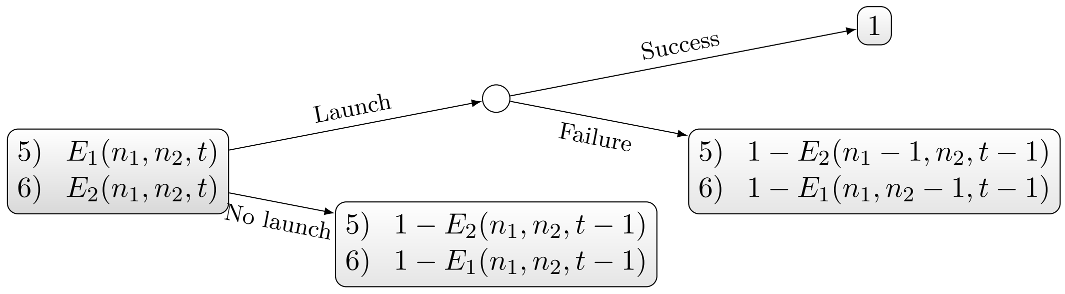


Figure 4.1.4: Decision tree, Scenario 1, both players with more than one rocket

We can now summarize all the formulas in one theorem:

Theorem 4.3. (Recursions in the Rich vs. Poor model, Scenario 1)

The following recursions describe the two-player-model with standard rule set:

$E_1(n_1,n_2,1) = E_2(n_1,n_2,1) = \frac{1}{2}, n_1, n_2 \geq 1$

$$E_1(1,1,2) = E_2(1,1,2) = \frac{5}{8}$$

$$E_1(n_1,1,2) = E_2(1,n_2,2) = \frac{3}{4} \qquad \textit{for } n_1, n_2 > 1$$

$$E_1(n_1,k,2) = E_2(k,n_2,2) = \frac{1}{2} \qquad \textit{for } n_1, n_2 \geq 1, k > 1$$

$$E_1(1,n_2,t) = s_1^2 + (1-s_1)\cdot(\frac{1+s_1}{2}) \quad \textit{for } n_2 \geq 1, t > 2$$

$$E_2(n_1,1,t) = s_2^2 + (1-s_2)\cdot(\frac{1+s_2}{2}) \quad \textit{for } n_1 \geq 1, t > 2$$

$$\begin{aligned}
E_1(n_1,1,t) = &\ s_3 \cdot (1 - E_2(n_1,1,t-1)) \\
+ &\ (1-s_3)\cdot\big[\frac{1+s_3}{2} + \big(1 - \frac{1+s_3}{2}\big)\cdot(1 - E_2(n_1-1,1,t-1)\big], n_1 \geq 2, t > 2 \\
E_2(1,n_2,t) = &\ s_4 \cdot (1 - E_1(1,n_2,t-1)) \\
+ &\ (1-s_4)\cdot\big[\frac{1+s_4}{2} + \big(1 - \frac{1+s_4}{2}\big)\cdot(1 - E_1(1,n_2-1,t-1)\big], n_2 \geq 2, t > 2 \\
E_1(n_1,n_2,t) = &\ s_5 \cdot (1 - E_2(n_1,n_2,t-1)) \\
+ &\ (1-s_5)\cdot\big[\frac{1+s_5}{2} + \big(1 - \frac{1+s_5}{2}\big)\cdot(1 - E_2(n_1-1,n_2,t-1)\big], n_1, n_2 \geq 2, t > 2 \\
E_2(n_1,n_2,t) = &\ s_6 \cdot (1 - E_1(n_1,n_2,t-1)) \\
+ &\ (1-s_6)\cdot\big[\frac{1+s_6}{2} + \big(1 - \frac{1+s_6}{2}\big)\cdot(1 - E_1(n_1,n_2-1,t-1)\big], n_1, n_2 \geq 2, t > 2
\end{aligned}$$

with

$$s_1 = 1 - E_2(1,n_2,t-1) \qquad s_2 = 1 - E_1(n_1,1,t-1)$$

$$s_3 = \frac{E_2(n_1-1,1,t-1) - E_2(n_1,1,t-1)}{E_2(n_1-1,1,t-1)} \qquad s_4 = \frac{E_1(1,n_2-1,t-1) - E_1(1,n_2,t-1)}{E_1(1,n_2-1,t-1)}$$

$$s_5 = 1 - \frac{E_2(n_1,n_2,t-1)}{E_2(n_1-1,n_2,t-1)} \qquad s_6 = 1 - \frac{E_1(n_1,n_2,t-1)}{E_1(n_1,n_2-1,t-1)}$$

Now the only question remaining is the behavior of $E_d(n_1,n_2,t)$ for large value of n_1, n_2, t and the advantage of the rich player. We can easily find an upper bound for the rich player's expectation and a lower bound for the poor player with this thought experiment: Given infinite rockets, the rich player will **always** launch a rocket, regardless of the current x_i. He only has to avoid the case in which he fails his first launch attempt and the poor player launches his first rocket successfully. Both of these events happen with probability $\frac{1}{2}$, therefore $E_1(n_1,n_2,t)$ should not exceed $1-(\frac{1}{2}\cdot\frac{1}{2}) = \frac{3}{4}$. On the other hand, the poor player will (on his turn) always have the possibility to just launch the rocket, regardless of x_i, thus $E_2(n_1,1,t)$ should never drop below $\frac{1}{2}$.
Now, these recursions can be programmed to be calculated automatically. We use *Scilab* running the program *RichVsPoor1*, which can be found on the added CD as

all other source codes. The program has three starting parameters *n1max, n2max, tmax*, whose purpose should be fairly self-explanatory. Via testing the program, the following results were gathered:

Theorem 4.4. (Results - Rich vs. Poor, basic rule set (Scenario 1)) *i)* $\lim_{t\to\infty} E_{1,2}(1,1,t) = 2-\sqrt{2}$, *expected points of waiting player are* $\sqrt{2}-1$

ii) $\lim_{n_1=t\to\infty} E_1(n_1,1,t) = \sqrt{3}-1 \approx 0.732051$ *(rich player's advantage)*

iii) $\lim_{n_2=t\to\infty} E_1(1,n_2,t) = 4-2\sqrt{3} \approx 0.535860$ *(poor player's disadvantage)*

iv) $\lim_{t\to\infty} E_1(n_1=n_2=t) = \frac{2}{3}$

As we can see, the findings ii) and iii) comply with our upper and lower bound.

Proof.
i) Assuming there is a limit to which the expected points in the 1-vs.-1-situation converge, we take a closer look at $E_1(1,1,t)$. For brevity, we rename it $E_{1,t}$, $E_2(1,1,t-1)$ will be shortened as $E_{2,t-1}$ and $E_1(1,1,t-2)$ becomes $E_{1,t-2}$. We therefore get

$E_{1,t} = E_{2,t-1}^2 + (1-E_{2,t-1}) \cdot \frac{1+E_{2,t-1}}{2}$, and
$E_{2,t-1} = E_{1,t-2}^2 + (1-E_{1,t-2}) \cdot \frac{1+E_{1,t-2}}{2}$.

Combining both equations gives

$$\begin{aligned} E_{1,t} &= [E_{1,t-2}^2 + (1-E_{1,t-2}) \cdot \frac{1+E_{1,t-2}}{2}]^2 \\ &+[1-(E_{1,t-2}^2 + (1-E_{1,t-2}) \cdot \frac{1+E_{1,t-2}}{2})] \cdot \frac{1+(E_{1,t-2}^2 + (1-E_{1,t-2}) \cdot \frac{1+E_{1,t-2}}{2})}{2} \\ &= E_{1,t-2}^4 - 4 \cdot E_{1,t-2}^3 + 4 \cdot E_{1,t-2}^2 - 7 \cdot E_{1,t-2} + 4. \end{aligned}$$

Assuming convergence, $E_{1,t}$ should equal $E_{1,t-2}$ for large t, thus

$$E_{1,t}^4 - 4 \cdot E_{1,t}^3 + 4 \cdot E_{1,t}^2 - 8 \cdot E_{1,t} + 4 = 0.$$

This equation of degree 4 has as many solutions, two of them real, two complex. The two real solutions are $2+\sqrt{2}$, which is outside of $(0,1)$ and therefore cannot be the limit, and $2-\sqrt{2}$, as above in the results:

$(2-\sqrt{2})^4 - 4 \cdot (2-\sqrt{2})^3 + 4 \cdot (2-\sqrt{2})^2 - 8 \cdot (2-\sqrt{2}) + 4$
$= 68 - 48\sqrt{2} - 80 + 56\sqrt{2} + 24 - 16\sqrt{2} - 16 + 8\sqrt{2} + 4 = 0.$

---

iii) $E_1(1,n_2,t) = s_1^2 + (1-s_1)(\frac{1+s_1}{2})$
$= (1-E_2(1,n_2,t-1))^2 + (1-(1-E_2(1,n_2,t-1))) \cdot (\frac{1+(1-E_2(1,n_2,t-1))}{2})$
$= \frac{1}{2}E_2(1,n_2,t-1)^2 - E_2(1,n_2,t-1) + 1.$

Now, we set $E_1(1, n_2 - 1, t - 2) = E_{1,n_2-1}$ and $E_1(1, n_2, t - 2) = E_{1,n_2}$ and receive

$E_2(1, n_2, t - 1) = s_4 \cdot (1 - E_{1,n_2-1}) + (1 - s_4) \cdot [\frac{1 + s_4}{2} + (1 - \frac{1 + s_4}{2}) \cdot (1 - E_{1,n_2-1})]$

as outlined in Theorem 4.3, with $s_4 = \dfrac{E_{1,n_2-1} - E_{1,n_2}}{E_{1,n_2-1}}$ the term equals

$E_2(1, n_2, t - 1) = \dfrac{E_{1,n_2-1} - E_{1,n_2}}{E_{1,n_2-1}} \cdot (1 - E_{1,n_2})$

$+ (1 - \dfrac{E_{1,n_2-1} - E_{1,n_2}}{E_{1,n_2-1}}) \cdot [\dfrac{1 + \frac{E_{1,n_2-1} - E_{1,n_2}}{E_{1,n_2-1}}}{2} + (1 - \dfrac{1 + \frac{E_{1,n_2-1} - E_{1,n_2}}{E_{1,n_2-1}}}{2} \cdot (1 - E_{1,n_2-1})]$

$= \dfrac{E_{1,n_2}^2}{2 \cdot E_{1,n_2-1}} - E_{1,n_2} + 1$

Similar to the proof for i), assuming $E_1(1, n_2, t)$ converges for $n_2, t \to \infty$, $E_1(1, n_2, t)$ will equal $E_1(1, n_2, t - 2)$ for large enough t.

The term $\dfrac{E_{1,n_2}^2}{2 \cdot E_{1,n_2-1}} - E_{1,n_2} + 1$ can thus be simplified as $1 - \dfrac{E_1(1.n_2 - 1, t - 2)}{2}$.

Finally, we put everything together and receive

$$\begin{aligned} E_1(1, n_2, t) &= \frac{1}{2} \cdot (1 - \frac{E_1(1.n_2 - 1, t - 2)}{2})^2 - (1 - \frac{E_1(1.n_2 - 1, t - 2)}{2}) + 1 \\ &= \frac{1}{8} \cdot E_1(1.n_2 - 1, t - 2)^2 + \frac{1}{2}. \end{aligned}$$

Under the assumption above, $E_1(1, n_2, t) = E_1(1, n_2 - 1, t - 2)$ for large n_2, t, and the equation is equivalent to

$$\frac{1}{8} \cdot E_1(1.n_2, t)^2 - E_1(1.n_2, t) + \frac{1}{2} = 0.$$

This quadratic equation has two solutions, with $4 - 2\sqrt{3}$ as the limit for $E_1(1.n_2, t)$.

---

ii) We already saw a example of the recursion for player Rich in the proof for iii):

$E_2(1, n_2, t - 1) = \dfrac{E_{1,n_2-1} - E_{1,n_2}}{E_{1,n_2-1}} \cdot (1 - E_{1,n_2}) + ... = \dfrac{E_{1,n_2}^2}{2 \cdot E_{1,n_2-1}} - E_{1,n_2} + 1$

This can also be simplified by setting $E_{1,n_2-1} = E_{1,n_2}$, giving $1 - \dfrac{E_1(1.n_2 - 1, t - 2)}{2}$

The analogous formula can be found for $E_1(n_1, 1, t)$:

$E_1(n_1, 1, t) = 1 - \dfrac{E_2(n_1 - 1, 1, t - 1)}{2}$ by assuming convergence. We already found the limit of the numerator, therefore we conclude:

$$\lim_{n_1 = t \to \infty} E_1(n_1, 1, t) = \lim_{n_2 = t \to \infty} 1 - \frac{E_2(n_1 - 1, 1, t - 1)}{2} = 1 - \frac{4 - 2\sqrt{3}}{2} = \sqrt{3} - 1.$$

---

iv) We can make a case for iv) by looking at a scenario in which both players have sufficient rockets to always attempt a launch: The player with launch rights will either win right on the first day (probability $\frac{1}{2}$), on the third day after the opponent failed the launch on day 2 (probability $\frac{1}{8}$), on the fifth day ($\frac{1}{32}$) and so on:

$$\begin{aligned}
E_1(n_1,n_2,t) &= \frac{1}{2}+\frac{1}{8}+\frac{1}{32}+\frac{1}{128}+...\\
\Rightarrow \frac{1}{4}\cdot E_1(n_1,n_2,t) &= \frac{1}{8}+\frac{1}{32}+\frac{1}{128}+...\\
\Rightarrow E_1(n_1,n_2,t)-\frac{1}{4}\cdot E_1(n_1,n_2,t) &= \frac{1}{2}\\
\Leftrightarrow E_1(n_1,n_2,t) &= \frac{2}{3}
\end{aligned}$$

□

Taking a look at the results for the decision parameter, the results are as follows

i) $\lim\limits_{t\to\infty} s_{1,2}(1,1,t)=\sqrt{2}-1\approx 0.414214$

ii) $\lim\limits_{n_1,t\to\infty} s_1(n_1,1,t)=0$ ($\to$ always launch)

iii) $\lim\limits_{n_2,t\to\infty} s_1(1,n_2,t)=2-\sqrt{3}\approx 0.267949$ (poor player's decision parameter)

iv) $\lim\limits_{n_1,n_2,t\to\infty} s_{1,2}(n_1,n_2,t)=0$ (for $n_1,n_2=O(t)$)

Proof.
(ii) and (iv) were already explained previously.
(i): Decision parameter $E_1(1,1,t)=1-E_2(1,1,t-1)=1-(2-\sqrt{2})=\sqrt{2}-1$.
(iii): Theorem 4.3 and $s_1=1-E_2(1,n_2,t-1)=1-(\sqrt{3}-1)=2-\sqrt{3}$.

□

Note: In this model, examining the case in which the passive player only holds one rocket is actually superfluous, as it is included in the case „both players with more than one rocket“ which means $s_3=s_5$ and $s_4=s_6$. This redundancy was due to the way the recursions were programmed in Scilab, and also occurs in most other scenarios in this section.

4.2 Scenario 2: Lucky Loser

For the lucky-loser-rule, we will only consider scenarios in which the losing player still has time left to launch a rocket. This means that if the winner successfully launches a rocket on the last day, he will get the full point for his win, and the loser will get none. If the first successful launch happens while $T \geq 2$, the lucky-loser-rule comes into effect.

In the first scenario, we have seen that Rich has a distinct advantage over Poor, due to being able to attempt more launches. His win-rate during his turn for large t was approaching around 73% or $\sqrt{3}-1$, while Poor's winning chance during his turn for large t was approaching $4-2\sqrt{3}$ or around 54%. Now we will introduce a variable $p \in (0,1)$ which the losing player can still win by successfully launching a rocket the day after his opponent's first launch. Questions to discuss in this scenario:

- Does the lucky-loser rule help to reduce Rich's advantage, or does it benefit both players equally?
- If the loser bonus helps Poor, which $p \in (0,1)$ leads to a fair game, i.e. for which p is $\lim_{n_1,t\to\infty} E_1(n_1,1,t) = \lim_{n_2,t\to\infty} E_1(1,n_2,t)$? This would mean that no matter how many rockets Rich has, his expected points would be the same during his turn.
- If there is an equilibrium as in point 2, which expected points value do both players receive when it's their turn?

First, we take a theoretical look at the extreme values of p, just outside $(0,1)$. If $p=0$, the lucky-loser rule would have no consequence, and we receive the same results as in Scenario 1. For $p=1$, and giving Rich infinite rockets, he will always attempt a launch. But his expected points are limited to $\frac{1}{2}$, because Poor can still 'steal' the full point by launching a rocket after Rich's first launch, succeeding half the time on average. Looking at the strategy of Poor (giving him 1 rocket and Rich infinite rockets), his best option for large t is to wait for Rich's first successful launch, and then attempt a launch himself, leading to $\frac{1}{2}$ expected points. Any other strategy will yield $\leq \frac{1}{2}$ for Poor, because of the possibility of destroying his only rocket if he goes first.

By this thought experiment, we can already deduct the assumption that a rising p will indeed help Poor, with $p=1$ making the game fair and $E_{1,2} = \frac{1}{2}$ as the equilibrium expectation in this case. It can also be assumed that p increasing towards 1 will make the game more fair, until the balance between Rich and Poor is achieved. Furthermore, the game will likely become more of a waiting-game the higher p is, as the reward for having a successful launch decreases considerably, especially if $p > \frac{1}{2}$. We will now focus on the recursion for this scenario for $p \in (0,1)$.

For $t=1$, we receive $E_{1,2}(n_1,n_2,1) = \frac{1}{2}$ for all $n_1, n_2 > 0$, just like in Section 4.1. For $t=2, p$ now comes into consideration:

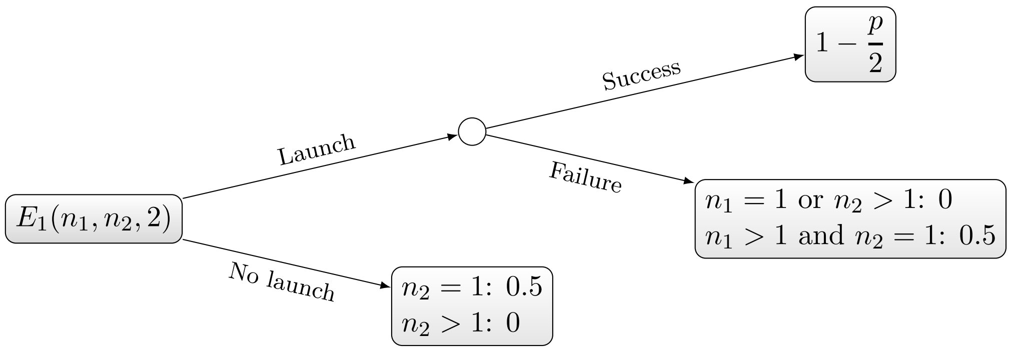


Figure 4.2.1: Decision tree, Scenario 2 (lucky loser), $t = 2$

$$\begin{aligned}&E_{1,2}(1,1,2)\\ &= P\big[x_2\cdot(1-\tfrac{p}{2})<\tfrac{1}{2}\big]\cdot\tfrac{1}{2}+P\big[x_2\cdot(1-\tfrac{p}{2})\geq\tfrac{1}{2}\big]\cdot E\big[x_2\cdot(1-\tfrac{p}{2})|x_2\cdot(1-\tfrac{p}{2})\geq\tfrac{1}{2}\big]\\ &= \big[\tfrac{1}{2}\cdot\frac{1}{1-p/2}\big]\cdot\tfrac{1}{2}+\big[1-\tfrac{1}{2}\cdot\frac{1}{1-p/2}\big]\cdot\big[\tfrac{1}{2}\cdot\big(1+\tfrac{1}{2}\cdot\frac{1}{1-p/2}\big)\cdot(1-p/2)\big]\\ &= \frac{-0.25p^2+p-1.25}{p-2}.\end{aligned}$$

This term is monotonically decreasing for $p \in (0,1)$, and equals $\frac{5}{8}$ for $p = 0$ and $\frac{1}{2}$ for $p = 1$. Furthermore we get

$$E_1(n_1,1,2) = E_2(1,n_2,2) = \frac{1-p/2+1/2}{2} = \frac{3-p}{4}$$

and similar to Scenario 1, we also have a must-launch-case for $t = 2$:

$$E_1(n_1,k,2) = E_2(k,n_2,2) = \frac{1-p/2}{2} = \frac{1}{2}-\frac{p}{4}, n_1,n_2\geq 1, k>1$$

Next, we take a look at the situation in which a player with one rocket has the launch option for the day:

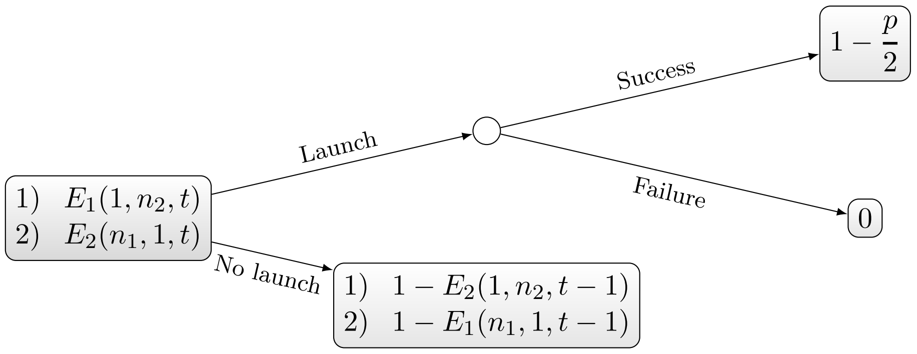


Figure 4.2.2: Decision tree, Scenario 2, active player with single rocket

The parameter p will obviously also influence the decision making. For optimal results, the decision parameter solves:

$1-E_2(1,n_2,t-1) = s_1(1,n_2,t)\cdot(1-\frac{p}{2})$ and $1-E_1(n_1,1,t-1) = s_2(n_1,1,t)\cdot(1-\frac{p}{2})$

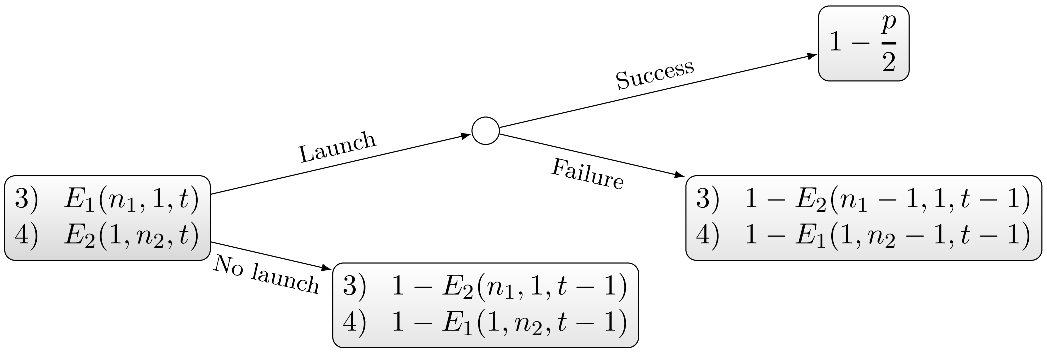
which gives us $s_1 = \dfrac{1 - E_2(1, n_2, t-1)}{1 - p/2}$, $s_2 = \dfrac{1 - E_1(n_1, 1, t-1)}{1 - p/2}$. For $p = 0$, we get the analogous results of 4.1. The expected points of Poor can then be calculated by

$$E_1(1, n_2, t) = s_1 \cdot (1 - E_2(1, n_2, t-1)) + (1 - s_1) \cdot [(1 - \frac{p}{2}) \cdot \frac{1 + s_1}{2}]$$

$$E_2(n_1, 1, t) = s_2 \cdot (1 - E_1(n_1, 1, t-1)) + (1 - s_2) \cdot [(1 - \frac{p}{2}) \cdot \frac{1 + s_2}{2}]$$

We see that p changes the value of the conditional expectation when x_t is larger than the expected points in $t-1$. The decision parameters will also be dependent on p. We will focus on the decision parameters, the formulas for the expected points can be found in Theorem 4.4 below. Now, we can take a look at the situations where the player with launch right has his opponent down to one rocket, and a launch failure doesn't equal an instant loss. The decision tree is analogous to Figure 4.1.3, except the payout for a successful launch will only be $1 - \frac{p}{2}$, instead of 1. (Figure 4.2.3) The decision parameters $s_3 = s_1(n_1, 1, t)$ and $s_4 = s_2(1, n_2, t)$ for both players in this situation are given by the equations:

$$\begin{aligned} 1 - E_2(n_1, 1, t-1) &= s_3 \cdot (1 - \frac{p}{2}) + (1 - s_3) \cdot (1 - E_2(n_1 - 1, 1, t-1)) \\ \Leftrightarrow s_3 &= \frac{E_2(n_1 - 1, 1, t-1) - E_2(n_1, 1, t-1)}{E_2(n_1 - 1, 1, t-1) \textbf{ - p/2}} \\ 1 - E_1(1, n_2, t-1) &= s_4 \cdot (1 - \frac{p}{2}) + (1 - s_4) \cdot (1 - E_1(1, n_2 - 1, t-1)) \\ \Leftrightarrow s_4 &= \frac{E_1(1, n_2 - 1, t-1) - E_1(1, n_2, t-1)}{E_1(1, n_2 - 1, t-1) \textbf{ - p/2}} \end{aligned}$$



$1 - \frac{p}{2}$
Success
Launch
Failure
3) $E_1(n_1, 1, t)$
4) $E_2(1, n_2, t)$
3) $1 - E_2(n_1 - 1, 1, t-1)$
4) $1 - E_1(1, n_2 - 1, t-1)$
No launch
3) $1 - E_2(n_1, 1, t-1)$
4) $1 - E_1(1, n_2, t-1)$

Figure 4.2.3: Decision tree, Scenario 2, passive player with single rocket

with $E_1(n_1, 1, t), E_2(1, n_2, t)$ being calculated the same way as in 4.1. For the general recursion ($n_1, n_2 > 1, t > 2$) we only need to find the new $s_5 = s_1(n_1, n_2, t)$ and $s_6 = s_2(n_1, n_2, t)$.

$$\begin{aligned} 1 - E_2(n_1, n_2, t-1) &= s_5 \cdot (1 - \frac{p}{2}) + (1 - s_5) \cdot (1 - E_2(n_1 - 1, n_2, t-1)) \\ \Leftrightarrow s_5 &= \frac{E_2(n_1 - 1, n_2, t-1) - E_2(n_1, n_2, t-1)}{E_2(n_1 - 1, n_2, t-1) \textbf{ - p/2}} \end{aligned}$$

$$\begin{aligned} 1 - E_1(1, n_2, t-1) &= s_6 \cdot (1 - \frac{p}{2}) + (1 - s_6) \cdot (1 - E_1(n_1, n_2 - 1, t-1)) \\ \Leftrightarrow s_6 &= \frac{E_1(n_1, n_2 - 1, t-1) - E_1(n_1, n_2, t-1)}{E_1(n_1, n_2 - 1, t-1) \mathbf{\ - p/2}} \end{aligned}$$

We see that the parameter p has a direct influence all throughout the recursions (unlike in Scenario 4 later). Now, these recursions were once again programmed in Scilab (*RichVsPoor2*). This program has the additional parameter p in addition to the three previous variables. Running *RichVsPoor2* with parameter $p = 0$ equals running *RichVsPoor1*. In Theorem 4.5, one can find all the recursion formulas in an overview.

Theorem 4.5. (Recursions in the Rich vs. Poor model, Scenario 2)
The following recursions describe the rich-vs-poor-model with 'lucky-loser' rule defined by parameter p:

$E_{1,2}(n_1, n_2, 1) = \frac{1}{2}$ *for all* $n_1, n_2 > 0$

$E_{1,2}(1, 1, 2) = \frac{-0.25p^2 + p - 1.25}{p - 2}$.

$E_1(n_1, 1, 2) = E_2(1, n_2, 2) = \frac{3}{4} - \frac{p}{4}$, *for* $n_1, n_2 > 1$

$E_1(n_1, k, 2) = E_2(k, n_2, 2) = \frac{1}{2} - \frac{p}{4}$, *for* $n_1, n_2 \geq 1, k > 1$

$$E_1(1, n_2, t) = s_1 \cdot (1 - E_2(1, n_2, t-1)) + (1 - s_1) \cdot [(1 - \frac{p}{2}) \cdot \frac{1 + s_1}{2}] \qquad n_2 \geq 1, t > 2$$

$$E_2(n_1, 1, t) = s_2 \cdot (1 - E_1(n_1, 1, t-1)) + (1 - s_2) \cdot [(1 - \frac{p}{2}) \cdot \frac{1 + s_2}{2}] \qquad n_1 \geq 1, t > 2$$

$$\begin{aligned} &E_1(n_1, 1, t) = s_3 \cdot (1 - E_2(n_1, 1, t-1)) \\ &+ (1 - s_3) \cdot \big[\frac{1 + s_3}{2} \cdot (1 - \frac{p}{2}) + \big(1 - \frac{1 + s_3}{2}\big) \cdot (1 - E_2(n_1 - 1, 1, t-1)\big] \qquad n_1 \geq 2, t > 2 \end{aligned}$$

$$\begin{aligned} &E_2(1, n_2, t) = s_4 \cdot (1 - E_1(1, n_2, t-1)) \\ &+ (1 - s_4) \cdot \big[\frac{1 + s_4}{2} \cdot (1 - \frac{p}{2}) + \big(1 - \frac{1 + s_4}{2}\big) \cdot (1 - E_1(1, n_2 - 1, t-1)\big] \qquad n_2 \geq 2, t > 2 \end{aligned}$$

$$\begin{aligned} &E_1(n_1, n_2, t) = s_5 \cdot (1 - E_2(n_1, n_2, t-1)) \\ &+ (1 - s_5) \cdot \big[\frac{1 + s_5}{2} \cdot (1 - \frac{p}{2}) + \big(1 - \frac{1 + s_5}{2}\big) \cdot (1 - E_2(n_1 - 1, n_2, t-1)\big] \qquad n_1, n_2 \geq 2, t > 2 \end{aligned}$$

$$\begin{aligned} &E_2(n_1, n_2, t) = s_6 \cdot (1 - E_1(n_1, n_2, t-1)) \\ &+ (1 - s_6) \cdot \big[\frac{1 + s_6}{2} \cdot (1 - \frac{p}{2}) + \big(1 - \frac{1 + s_6}{2}\big) \cdot (1 - E_1(n_1, n_2 - 1, t-1)\big] \qquad n_1, n_2 \geq 2, t > 2 \end{aligned}$$

with the following decision parameters:

$$s_1 = \frac{1 - E_2(1, n_2, t-1)}{1 - p/2} \qquad s_2 = \frac{1 - E_1(n_1, 1, t-1)}{1 - p/2}$$

$$s_3 = \frac{E_2(n_1 - 1, 1, t-1) - E_2(n_1, 1, t-1)}{E_2(n_1 - 1, 1, t-1) - p/2} \qquad s_4 = \frac{E_1(1, n_2 - 1, t-1) - E_1(1, n_2, t-1)}{E_1(1, n_2 - 1, t-1) - p/2}$$

$$s_5 = \frac{E_2(n_1 - 1, n_2, t-1) - E_2(n_1, n_2, t-1)}{E_2(n_1 - 1, n_2, t-1) - p/2} \qquad s_6 = \frac{E_1(n_1, n_2 - 1, t-1) - E_1(n_1, n_2, t-1)}{E_1(n_1, n_2 - 1, t-1) - p/2}$$

Now, it is time to examine the results of the recursion. Once again, most of interest is the value of $\lim_{n_1,t\to\infty} E_1(n_1,1,t)$ (advantage of Rich) and $\lim_{n_1,t\to\infty} E_2(n_1,1,t)$ (disadvantage of Poor). For $p=0$, these were $\sqrt{3}-1$ and $4-2\sqrt{3}$ respectively. It was first assumed that both these values would monotonically decrease for growing p and merge in $p=1$ with the result $\frac{1}{2}$ indicating a fair game. But as one can see in Figure 4.2.4, this is not the case:

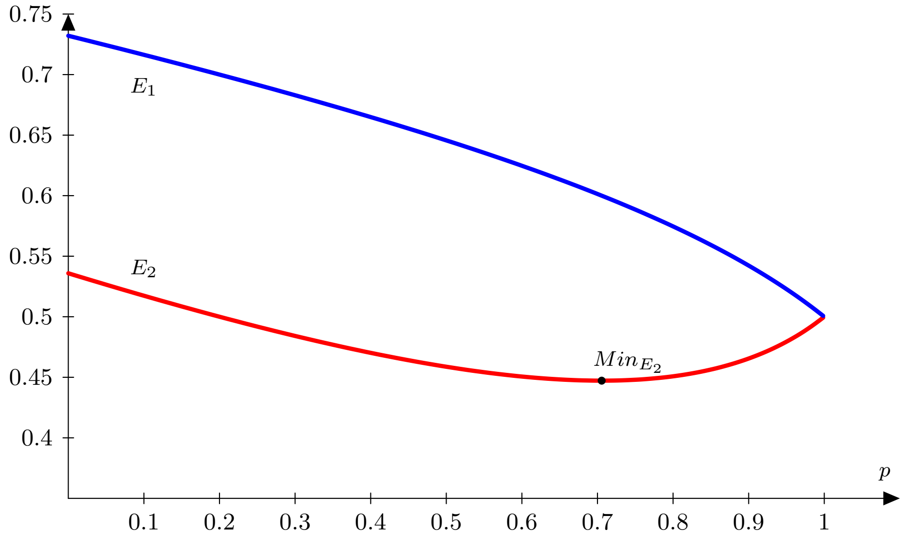


Figure 4.2.4: Rich vs. Poor, Scenario 2, $E_1(n_1,1,t)$ blue, $E_2(n_2,1,t)$ red, n_i,t large

Poor's disadvantage goes down even more than previously thought, with him expecting less than half a point for $0.2 < p < 1$. Interesting, the values in $p=\frac{1}{5}$ are very even, with Rich receiving 0.7 and Poor receiving 0.5 expected points when it's their turn. Looking at the graph, the next question has to be: Where is the minimum of Poor's expected points? This was evaluated by trying out different p-values in *RichVsPoor2*:

Theorem 4.6. (Minimal expected points for Poor, Scenario 2) *In the lucky loser scenario, the minimum of Poor's expected points is reached in* $p \approx 0.7055728 = \frac{8}{5} - \frac{2}{\sqrt{5}}$. *In this p, the expected points of Rich are* $\frac{3}{5}$ *and Poor's expectation for using optimal strategy is* $\frac{1}{\sqrt{5}}$.

Proof. See appendix. $\square$

The proof for Theorem 4.6 also gives us functions to calculate Rich's and Poor's expected points for large t directly, given any p:

$$E_2(p) = \frac{1}{2}(-2\sqrt{5p^2-16p+12} - 5p + 8) \quad \Big| \quad E_1(p) = 1 - \frac{E_2(p)}{2} - \frac{p}{4}$$

The approximate plot of the two functions is already shown Figure 4.2.4 above. But after seeing that E_2 does indeed drop below $\frac{1}{2}$, the other assumption made before establishing the recursions has to be questioned as well: Does a rising p actually make the game fairer? We define two metrics to examine this:

Definition 4.7. (Relative/Absolute Advantage)
Relative advantage: $RA(p) = \dfrac{E_1(p)}{E_2(p)} - 1$

Absolute advantage: $AA(p) = E_1(p) - E_2(p)$

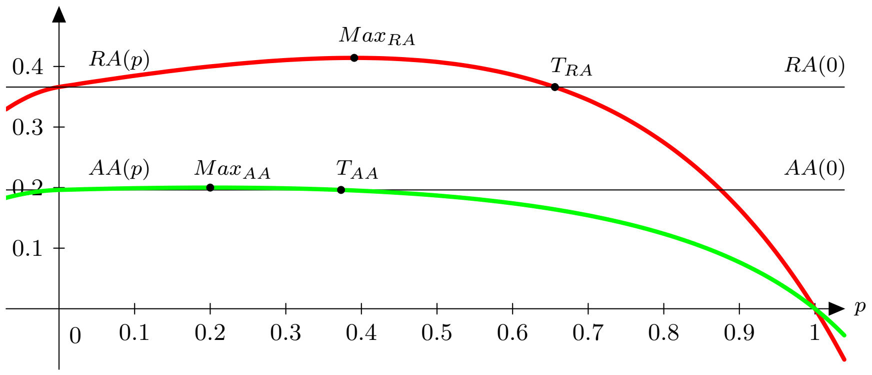


Figure 4.2.5: Rich vs. Poor, relative and absolute advantage (Scenario 2)

We see that for small p, RA is actually increasing, whereas AA seems to be nearly flat. For higher p, both functions begin decreasing towards 0. By analyzing both functions, we gather exact results:

Lemma 4.8. (Properties of $RA(p), AA(p)$)

- $\max_{p\in(0,1)}(RA(p)) = \sqrt{2} - 1$ *for* $p = \dfrac{4-2\sqrt{2}}{3} \approx 0.39052$

- $\min_{p>0}(RA(p) \leq RA(0)) = \dfrac{28-12\sqrt{3}}{11} \approx 0.65594$

- $\max_{p\in(0,1)}(AA(p)) = \dfrac{1}{5}$ *for* $p = \dfrac{1}{5}$

- $\min_{p>0}(AA(p) \leq AA(0)) = 21\sqrt{3} - 36 \approx 0.37307$

Proof. Derivatives for the maxima, equations for the „turning points“. □

In summary, increasing p does not necessarily help Poor to reduce the advantage Rich has over him. To reduce the relative advantage, Poor will need $p \gtrsim 0.65594$, whereas smaller p-values even diminish his chances even further. If he wants to increase his own expected points, $p \gtrsim 0.70557$ is required. If he only wants to reduce the absolute difference between him and Rich, $p \gtrsim 0.37307$ will suffice.

4.3 Scenario 3: No Launch, No win

In the first two scenarios of this model, we looked at contests in which both players have a secondary option of winning, besides launching a rocket successfully. That is, if Poor empties his rocket supply without any successful launch, Rich wins - and vice versa. Now, this secondary option might not be too realistic in real-life scenarios: Imagine two companies competing to send a satellite into space. In this situation, it is unlikely that one company will be considered victorious just because the other company has run out of rockets, as the common goal of satellite deployment is more important than the contest between the companies itself.

This is where the 'No launch, no win' rule comes into play. The secondary way of victory is scrapped from the model in this scenario. To be declared winners, a player **must** launch a rocket successfully. Waiting for the failure of the opponent will not yield positive results - as we will see in the early stages of recursion. This also becomes a scenario in which it is more likely that neither of the players wins.

We also have to decide what happens when one of the players wrecks his last rocket without a successful launch. In this thesis, the other player will receive **all** other days to make his launch attempts. If, for example, Poor crashes his final rocket in $t = 8$, Rich will have all seven remaining days to get a launch success (and not just four if the launching right was still alternating).
For the recursion, we will once again use Definition 4.1, and note that in this scenario n_1 or n_2 can equal zero (in Scenario 1 and 2 we had $n_1, n_2 \geq 0$). For $t = 1$, everything stays the same as in Scenario 1 with one small difference:

$$E_1(n_1, k, 1) = E_2(k, n_2, 1) = \frac{1}{2} \qquad \text{for all } n_1, n_2 > 0, k \in \mathbb{N}_0$$

This means that the passive player does not need to have rockets left in $t = 1$ - he will have lost at this point, though. Regardless of that, the current player still has to get a successful launch on the last day in order to win.
For $t = 2$, the differences in the decision-making become even clearer. Let's look at it from Rich's (Player 1) perspective in Figure 4.3.1: Now, instead of getting an

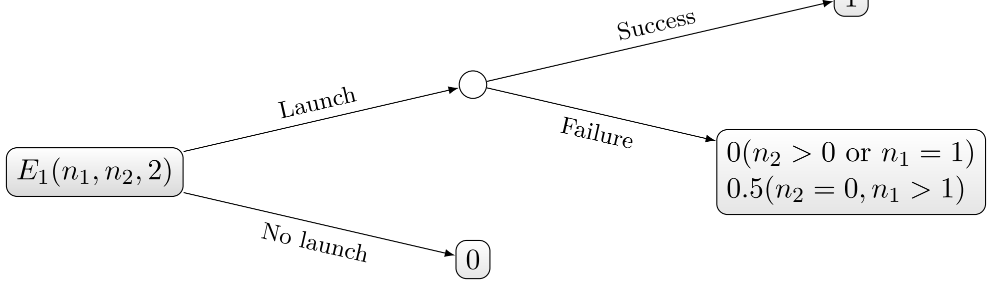


Figure 4.3.1: Decision tree, Scenario 3, $t = 2$

expected points value of $1 - E_2(n_1, n_2, 1)$, the player receives absolutely nothing by waiting for the last day, because he can't win through a failed launch of his opponent. Therefore, he does the same thing as he would do in $t = 1$: Launch a rocket with any x_2 the day gives to him. Therefore, we receive:
$E_1(n_1, 0, 2) = E_2(0, n_2, 2) = \dfrac{3}{4}$ for $n_1, n_2 > 1$, else $E_d(n_1, n_2, 2) = \dfrac{1}{2}$.

In this scenario, there are also situations in which the opposing players has no rockets left, as pictured in the decision tree in figure 4.3.2.

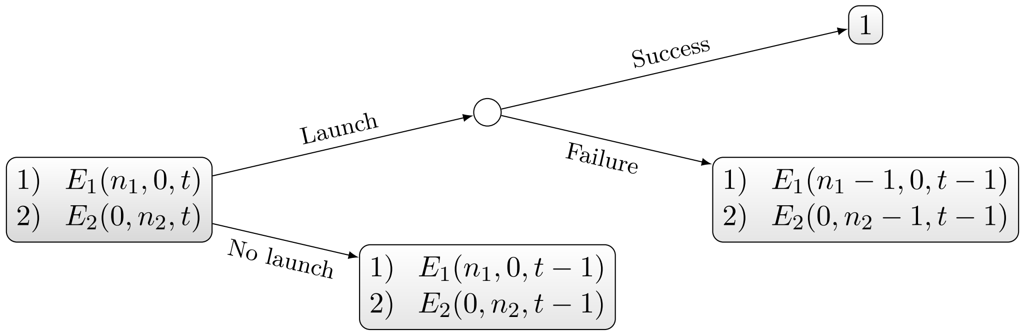


Figure 4.3.2: Decision tree, Scenario 3, passive player without rockets

Note that the expected value for failure will equal 0 if $n_1 = 1$ or $n_2 = 1$. The calculation for the expected points for both players in this case is found below. For this, we simplify $E_1(n_1, 0, t-1) = E_2(0, n_2, t-1) =: E_n$ and $E_1(n_1 - 1, 0, t-1) = E_2(0, n_2 - 1, t-1) =: E_{n-1}$:

$$\begin{aligned} E_1(n_1, 0, t) &= E_n^2 + (1 - E_n) \cdot [\frac{1 + E_n}{2} + (1 - \frac{1 + E_n}{2}) \cdot E_{n-1}] \\ &= \frac{E_n^2 \cdot E_{n-1} + E_n^2 - 2E_nE_{n-1} + E_{n-1} + 1}{2} \text{ for } n_1 > 1 \end{aligned}$$

For $n_1 = 1$, the model is equal to the one-player, one-rocket model in Section 2:

$$E_1(1, 0, t) = ... = \frac{1 + E_1(1, 0, t-1)^2}{2}$$

Looking at the situations in which one of the players has a single rocket left, we can see that there is no difference to the decision-making in Scenario 1, and the decision trees apply as well as the optimal decision parameter, see Figure 4.1.2/4.1.3 and calculations of s_1 to s_4. This is because destroying the the last rocket still loses that player the game, just like in the base scenario. Therefore, the potential outcomes for $E_1(1, n_2, t)$ and $E_2(1, n_2, t)$ don't change. The same can be said about the general recursion for $n_1 > 1, n_2 > 1, t > 2$, for which the decision parameters s_5 and s_6 found in Theorem 4.3 apply as well.

Theorem 4.9. (Recursions in the Rich vs. Poor model, Scenario 3)
The following recursions describe the rich-vs-poor-model with 'no launch, no win' rule.

$$E_1(n_1, n_2, 1) = E_2(n_1, n_2, 1) = \frac{1}{2} \quad \textit{for all } n_1, n_2 \geq 0 \textit{ except } n_1 = n_2 = 0.$$

$$E_1(n_1, 0, 2) = E_2(0, n_2, 2) = \frac{3}{4} \quad \textit{for all } n_1, n_2 > 1$$

$$E_1(n_1, n_2, 2) = E_2(n_1, n_2, 2) = \frac{1}{2} \quad \textit{for all } n_1, n_2 \geq 1$$

$$E_1(1, 0, t) = \frac{1 + E_1(1, 0, t-1)^2}{2} \qquad E_2(0, 1, t) = \frac{1 + E_2(0, 1, t-1)^2}{2}$$

$$E_1(n_1,0,t) = E_2(0,n_2,t) = \frac{E_n^2 \cdot E_{n-1} + E_n^2 - 2E_nE_{n-1} + E_{n-1} + 1}{2} \; \textit{for } n_1, n_2 > 1,$$
$$E_n = E_1(n_1,0,t-1) \textit{ or } E_2(0,n_2,t-1), E_{n-1} = E_1(n_1-1,0,t-1) \textit{ or } E_2(0,n_2-1,t-1)$$

and for all other scenarios, the formulas and decision parameter s_1,...,s_6 stated in Theorem 4.3 in Section 4.1 apply for the reasons mentioned above.

Looking at this scenario from a programming point of view, we now see that the extra rule essentially adds rows/columns to our multi-dimensional array already used in *RichVsPoor1* to determine expected points of both players in the basic scenario. Now, because numbering of rows/columns in Scilab starts with 1, we need to be careful to mind the index offset when altering the code to fit the scenario. This was done in *RichVsPoor3*, which has the same input variables as *RichVsPoor1* and can be found on the CD. We will also have to watch out for the indices when comparing the results of the two scenarios, as for example $E(2,3,1,5)$ of the output of *RichVsPoor1* should be compared to *RichVsPoor3*'s output of $E(2,4,2,5)$ due to index offset.

Letting the program calculate the recursions for us, we find that the limits found in Section 4.1/Theorem 4.4 apply in this scenario as well. The fact that both players have to successfully launch a rocket to get the full point doesn't change the advantage Rich has over Poor for large t.

Comparing the results of Scenario 1 with those in Scenario 3 (*CompareRVP*), we see that for smaller t, the active player's expected points are slightly reduced for even t in scenario 3 - whereas there is a slight increase for the active player's expectation in odd t. This is due to the fact that the only value different in this model compared to Scenario 1 is $E_1(n_1,1,2), E_2(1,n_2,2)$: In Scenario 1, this equals $\frac{3}{4}$ for $n_1, n_2 > 1$ because the active player can hope for a failed attempt of his opponent in $t = 1$, whereas in this Scenario he is forced to attempt a launch himself, giving him $\frac{1}{2}$ expected points.

This small difference causes the fluctuations for small t. These increases/decreases vanish rapidly though, and for large t, we end up with the same results as in the basic scenario in Section 4.1.

4.4 Scenario 4: Consolation Points

So far, we have only looked at models where both players are penalized for letting time run out, i.e. both losing and getting zero points when t reaches 0. Now, we take a look at a model in which both players are awarded a compensation if neither of them manage to successfully launch a rocket in the given time window. In real life, this could be seen as a payment of an insurance company.
The players will receive $c \in (0,1)$ if they *both* still have rockets left at $t = 0$. We still uphold the rule that if one player runs out of rockets, the other player wins the point. This means that players aren't forced to always attempt a launch when time is running out, if they see their chance at compensation is higher.
Firstly, let's take a look at the extreme values of c again. If $c = 0$, we receive the same results as in the basic scenario in section 4.1. For $c = 1$, the sky will be clear of rockets - because neither of the players has any incentive of attempting launches, as they will both receive the full point anyway. So, these to values aren't really of interest. So, we go ahead and study the scenario for $c \in (0,1)$
For $t = 1$ we had a forced attempt in the other scenarios. Now, the decision of the player depends on the number of rockets he has:

$$E_1(1, n_2, 1) = c \cdot c + (1-c) \cdot \frac{1+c}{2} \text{ for } n_2 \geq 1$$

$$E_1(n_1, n_2, 1) = \frac{1+c}{2} \text{ for } n_1 > 1.$$

If he only has a single rocket remaining, he will not get a compensation if he has a failed launch attempt on the final day. With more than one rocket remaining, he will have the compensation guaranteed and attempts a launch for any x_1 to take a chance of winning the full point. Player 2 will decide the same way.

For $t = 2$, we will take a look at the decision tree for Rich:

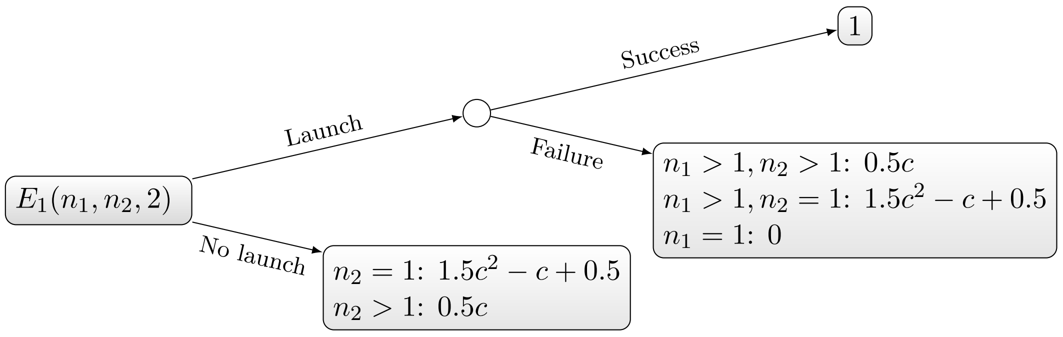

Success
1
Launch
Failure
$n_1 > 1, n_2 > 1$: $0.5c$
$n_1 > 1, n_2 = 1$: $1.5c^2 - c + 0.5$
$n_1 = 1$: 0
$E_1(n_1, n_2, 2)$
No launch
$n_2 = 1$: $1.5c^2 - c + 0.5$
$n_2 > 1$: $0.5c$

Figure 4.4.1: Decision tree, Scenario 4, $t = 2$

We see the same situation for both players: If the active player has more than one rocket left, he will attempt a launch regardless of x_2, since he has nothing to lose as the expected points for not launching and failing the launch attempt are identical. Let's look at Rich's decision in $t = 2$ in particular: He knows that if n_2¿1, Poor will make an attempt for any x_1 in $t = 1$, succeeding half the time on average. Therefore, Rich receives an expected $\frac{c}{2}$ if he doesn't launch or fails his attempt (if $n_1 > 1$).

If $n_2 = 1$, Poor will either not launch in $t = 1$, giving both players c, or risk a launch and either get the full point or none. Rich's expected return in this case is $c \cdot c + (1-c) \cdot (1 - \frac{1+c}{2}) = 1.5c^2 - c + 0.5$.

All in all, we receive for $n_1, n_2 > 1$:

$$\begin{aligned}
E_{1,2}(1,1,2) &= (1.5c^2 - c + 0.5)^2 + (1 - (1.5c^2 - c + 0.5)) \cdot \frac{1 + (1.5c^2 - c + 0.5)}{2} \\
&= 1.125c^4 - 1.5c^3 + 1.25c^2 - 0.5c + 0.625 \\
E_1(n_1,1,2) &= E_2(1,n_2,2) = 0.5 \cdot (1 + (1.5c^2 - c + 0.5) = 0.75c^2 - 0.5c + 0.75 \\
E_1(1,n_2,2) &= E_2(n_1,1,2) = (\frac{c}{2})^2 + (1 - \frac{c}{2}) \cdot \frac{1+0.5c}{2} = \frac{c^2}{8} + \frac{1}{2} \\
E_{1,2}(n_1,n_2,2) &= \frac{1}{2} + \frac{c}{4}
\end{aligned}$$

Moving forward to the case where the player with launch option has a single rocket with $t > 2$, with $E_2(n_1, 1, t)$ calculated analogously:

$$E_1(1,n_2,t) = (1 - E_2(1,n_2,t-1))^2 + E_2(1,n_2,t-1) \cdot \frac{1 + (1 - E_2(1,n_2,t-1))}{2}$$

In this situation, a failed launch will lead to 0 points, just like in Section 4.1. Now we examine the recursion when the roles are reversed, i.e. the player with more rockets having the launch option. We find that the decision making is the same as in Section 4.1, as well as the calculation of decision parameters (see Figure 4.1.3 for the decision tree):

$$s_1(n_1,1,t) = \frac{E_2(n_1-1,1,t-1) - E_2(n_1,1,t-1)}{E_2(n_1-1,1,t-1)} = s_3$$

$$s_2(1,n_2,t) = \frac{E_1(1,n_2-1,t-1) - E_1(1,n_2,t-1)}{E_1(1,n_2-1,t-1)} = s_4$$

with the corresponding expectations being calculated as in Theorem 4.3. The same can be said about the decision parameters s_5 and s_6, which are the same as in Theorem 4.3:

$$s_5 = 1 - \frac{E_2(n_1,n_2,t-1)}{E_2(n_1-1,n_2,t-1)}, s_6 = 1 - \frac{E_1(n_1,n_2,t-1)}{E_1(n_1,n_2-1,t-1)}.$$

We now quickly summarize the recursions for Scenario 4:

Theorem 4.10. (Recursions in the Rich vs. Poor model, Scenario 4) *The following recursions describe the rich-vs-poor-model with consolation rule:*

$$E_1(1,n_2,1) = E_2(n_1,1,1) = \frac{1+c^2}{2} \quad E_1(n_1,k,1) = E_2(k,n_2,1) = \frac{1+c}{2}, k \geq 1, n_i > 1$$

$$E_{1,2}(1,1,2) = \frac{9c^4}{8} - \frac{3c^3}{2} + \frac{5c^2}{4} - \frac{c}{2} \quad E_1(1,n_2,2) = E_2(n_1,1,2) = \frac{c^2}{8} + \frac{1}{2}, n_i > 1$$

$$E_1(n_1,1,2) = \frac{3c^2}{4} - \frac{c}{2} + \frac{3}{4}, n_1 > 1 \quad E_1(n_1,n_2,2) = \frac{c}{4} + \frac{1}{2}, n_i > 1$$

For all other cases, the formulas to calculate decision parameters s_1 to s_6 and expected points found in Theorem 4.3 apply.

Using *RichVsPoor4* we calculate the recursions in Scilab. Just as in Scenario 3, the findings in Theorem 4.4 also apply in this scenario. Even a c very close to 1 has no effect on the effected points of Rich and Poor for large t. But why does c not effect the results more? To find an easy answer, we can take a look at Rich's decision making in $t = 1$, when he has more than one rocket left: As one can see in Figure 4.4.2, Rich will get $c = 1 - \varepsilon$ ($\varepsilon > 0$), for either a failed launch or skipping the day. But no matter how small ε becomes, Rich will always attempt a launch to get it - and if he is successful, Poor is left with nothing.

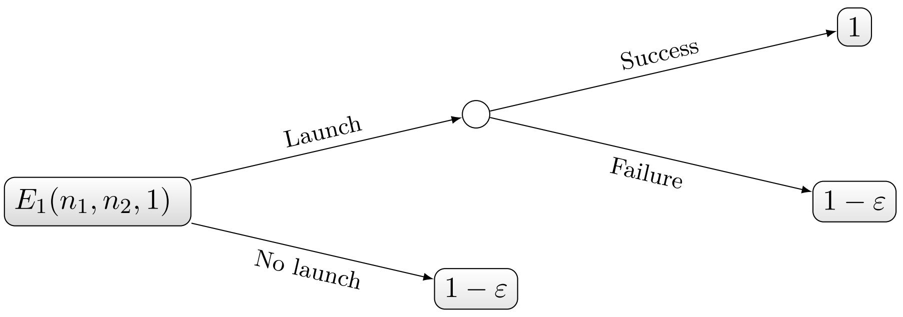


Figure 4.4.2: Decision tree, Scenario 4, $n_1 > 1, c = (1 - \varepsilon), t = 1$

For smaller t, we can also compare this model to the base scenario without consolation (Section 4.1). Using *compareRVP*, we take at look at the difference between running *RichVsPoor(10,10,20)* and *RichVsPoor4(10,10,0.5,20)*. We find mixed results: For even t, Rich's expectation is reduced for $n_2 < \frac{t}{2}$ and increased for $n_2 > \frac{t}{2}$ (and vice versa for Poor), whereas for odd $t(t > 1)$, Rich's expectation is increased for $n_1 < \lceil \frac{t}{2} \rceil$ and reduced for $n_1 > \lceil \frac{t}{2} + 1 \rceil$. Similarly to Scenario 3, these differences quickly converge to 0 for rising t: The maximum absolute difference in our example is always found in $n_1 = n_2 = 10$:

Table 4.4.1: Difference Scenario 1 − Scenario 4 (c=0.5) , E_1

t	1	2	3	4	5	6	7	8	9	10
Diff(10,10,t)	$-\frac{1}{4}$	$-\frac{1}{8}$	$\frac{1}{16}$	$-\frac{1}{32}$	$\frac{1}{64}$	$-\frac{1}{128}$	$\frac{1}{256}$	$-\frac{1}{512}$	$\frac{1}{1024}$	$-\frac{1}{2048}$

The fact that the maximum difference gets halved for every increase of t is not a „special effect“ of choosing $c = \frac{1}{2}$, it seems to work for all other $c \in (0, 1)$ as well, and can probably be proven using the recursion formulas (not part of this thesis).

After the three scenarios with one rule variant, we now move on to the models where two rules have been changed.

4.5 Scenario 5: Lucky Loser + No Launch, No Win

After looking at three scenarios with a single rule change, we now take a step further and combine two of the three rule variants to examine possible effects this might have on the model. First off, we will combine the Lucky loser rule with parameter p (as seen in Section 4.2) and the "No launch, no win" rule (as previously discussed in Section 4.3).

For $t = 1$, we noticed in the previous sections that both these rules have no influence on the players' decision-making or expected points, so we can state that:

$$E_1(n_1, k, 1) = E_2(k, n_2, 1) = \frac{1}{2} \qquad \text{for all } n_1, n_2 > 0, k \in \mathbb{N}_0$$

For $t = 2$, we notice a similar change to the decision-making as in Figure 4.3.1, with the pay-out for a successful launch being reduced to $1 - \frac{p}{2}$ due to the lucky loser rule. The player will also receive nothing if he waits for $t = 1$. Therefore, the player will always attempt a launch in $t = 2$ and will receive

$$E_1(n_1, n_2, 2) = E_2(n_1, n_2, 2) = \frac{1}{2} \cdot (1 - \frac{p}{2}) = \frac{1}{2} - \frac{p}{4} \qquad \text{for all } n_1, n_2 \geq 1.$$

$$E_1(n_1, 0, 2) = E_2(0, n_2, 2) = \frac{1}{2} \cdot (1 - \frac{p}{2}) + \frac{1}{2} \cdot \frac{1 - \frac{p}{2}}{2} = \frac{3}{4} - \frac{3}{8}p \qquad \text{for all } n_1, n_2 > 1.$$

Because the NLNW rule is in effect, we examine the situations in which one of the players has zero rockets next ($t > 1$). With this rule in place, the Lucky Loser rule does not affect the decision-making in these cases, because the player with 0 rockets can't have another try if his opponent succeeds in launching a rocket. Therefore, the decision tree is the same as in Section 4.3 and is depicted in Figure 4.3.2 - if $n_1 = 1$ or $n_2 = 1$, the expected points for a failed launch are reduced to 0, just as in Scenario 3. The recursion formula once again can be calculated as

$$E_1(n_1, 0, t) = ... = \frac{1 + E_1(n_1, 0, t-1)^2}{2}, E_2(0, n_2, t) = ... = \frac{1 + E_2(0, n_2, t-1)^2}{2}$$

Next up, the cases in which one of the players has a single rocket left: Here, the NLNW rule doesn't have any effect on the recursion formulas. The decision tree for $E_1(1, n_2, t)$ and $E_2(n_1, 1, t)$ can be found in Figure 4.2.2, the tree for $E_1(n_1, 1, t)$ and $E_2(1, n_2, t)$ is shown in Figure 4.2.3. The decision parameters s_1 to s_6 are calculated as outlined in Section 4.2.

As one might notice, these two rules don't really interact with each other. This is due to their different nature: The lucky-loser rule only really comes into play if both players have rockets, while the NLNW rule only has a visible effect when one player has no rockets left. For evaluating the recursion results, *RichVsPoor5* is used, with the standard parameter plus an additional parameters p for the lucky-loser rule and index offset to cover the NLNW rule.

Theorem 4.11. (Recursions in the Rich vs. Poor model, Scenario 5) *The following recursions describe the rich-vs-poor-model with 'no launch, no win' and 'lucky-loser' rule defined by parameter p:*

$$E_1(n_1,k,1) = E_2(k,n_2,1) = \frac{1}{2} \qquad \textit{for all } n_1,n_2 > 0, k \geq 0$$

$$E_1(n_1,n_2,2) = E_2(n_1,n_2,2) = \frac{1}{2} - \frac{p}{4} \qquad \textit{for all } n_1,n_2 \geq 1$$

$$E_1(n_1,0,2) = E_2(0,n_2,2) = \frac{3}{4} - \frac{3}{8}p \qquad \textit{for all } n_1,n_2 > 1$$

$$E_1(1,0,t) = \frac{1+E_1(1,0,t-1)^2}{2} \qquad E_2(0,1,t) = \frac{1+E_2(0,1,t-1)^2}{2}$$

$$E_1(n_1,0,t) = E_2(0,n_2,t) = \frac{E_n^2 \cdot E_{n-1} + E_n^2 - 2E_nE_{n-1} + E_{n-1} + 1}{2} \; \textit{for } n_1,n_2 > 1,$$

$$E_n = E_1(n_1,0,t-1) \textit{ or } E_2(0,n_2,t-1), E_{n-1} = E_1(n_1-1,0,t-1) \textit{ or } E_2(0,n_2-1,t-1)$$

For all other cases, the formulas to calculate decision parameters s_1 to s_6 and expected points found in Theorem 4.5 apply.

As one could expect, the limits for Rich's and Poor's expectation for large t are the same as pointed out in Theorem 4.6, where

$$E_2(p) = \frac{1}{2}(-2\sqrt{5p^2 - 16p + 12} - 5p + 8)$$

$$E_1(p) = 1 - \frac{E_2(p)}{2} - \frac{p}{4}$$

or every $p \in (0,1)$ and large t.

For smaller t, we can also compare the numbers of Scenario 5 and Scenario 2, finding a similar effect as during the comparison of Scenario 1 and Scenario 3: Adding the NLNW rule to the game reduces the expectation of the active player in even t, whereas it increases his expectation in odd t. Once again, these differences quickly converge to 0 for rising t: For $t = 20$, the maximum absolute difference between the two scenarios in our examples approximately $1.82 \cdot 10^{-5}$.

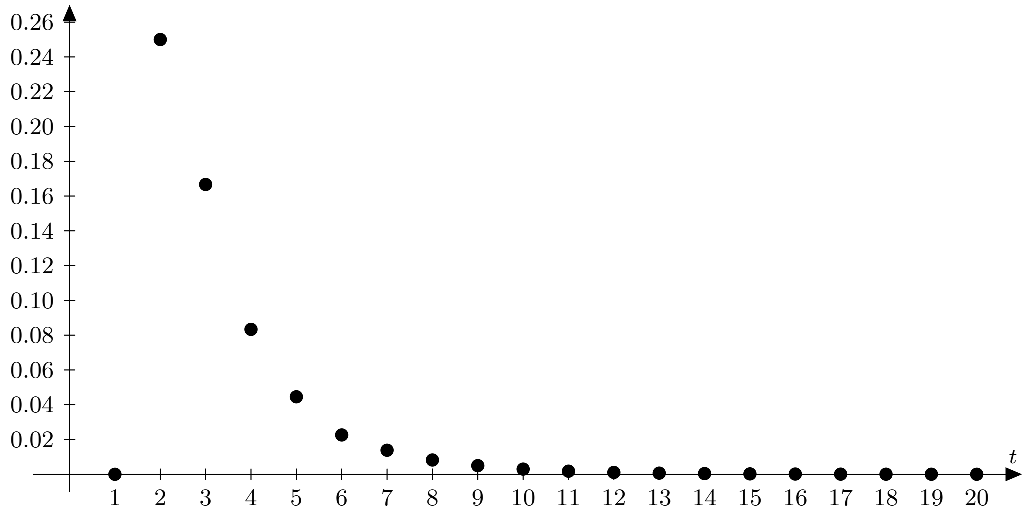


Figure 4.5.1: Maximum absolute difference of E_1 between Scenario 2 and 5, $p = \frac{1}{2}$

4.6 Scenario 6: Lucky Loser + Consolation Points

In this section we combine the two rule variations that can be adjusted with a parameter: The "lucky loser" rule with it's parameter $p \in (0,1)$ defining how many points a losing player can still win after his opponent's successful launch, and the consolation rule, with $c \in (0,1)$ being the amount of points both players receive if time runs out without any successful launch.

With two parameters to control the model, this should - at least in theory - be the most interesting scenario with two rule variations. But first, we will take a look at the recursion.
For $t = 1$, the lucky-loser rule doesn't affect the player. The consolation rule on the other hand does influence the expected points and we receive the same formula as in Section 4.4:

$$E_1(1, n_2, 1) = E_2(n_1, 1, 1) = \frac{1 + c^2}{2}, n_i \geq 1$$

$$E_1(n_1, k, 1) = E_2(k, n_2, 1) = \frac{1 + c}{2}, k \geq 1, n_i > 1$$

For $t = 2$, we look at the decision tree for player 1, which is very similar to the one in Figure 4.4.1:

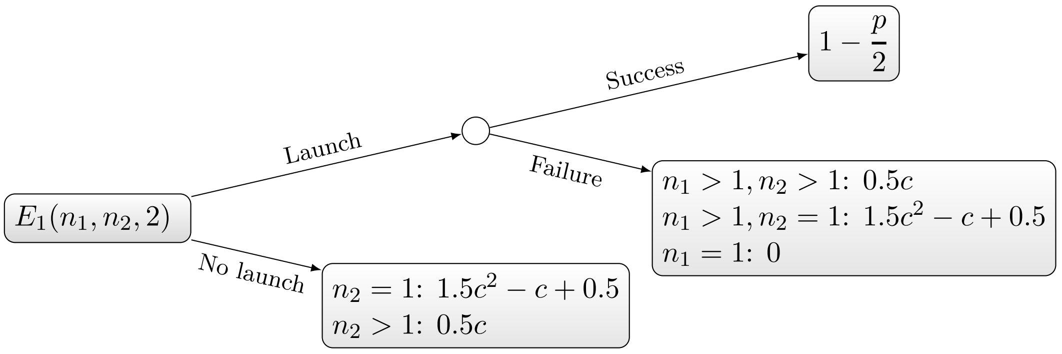


Figure 4.6.1: Decision tree, Scenario 6, $t = 2$

$$\begin{aligned}
E_{1,2}(1,1,2) &= P(x_2 \cdot (1 - \frac{p}{2}) < 1.5c^2 - c + 0.5) \cdot (1.5c^2 - c + 0.5) \\
&\quad + P(x_2 \cdot (1 - \frac{p}{2}) \geq 1.5c^2 - c + 0.5) \cdot E(x_2 \cdot (1 - \frac{p}{2}) | x_2 \cdot (1 - \frac{p}{2}) \geq 1.5c^2 - c + 0.5) \\
&= \frac{-2.25c^4 + 3c^3 - 2.5c^2 + c - 0.25p^2 + p - 1.25}{p - 2} \\
E_1(n_1, 1, 2) &= E_2(1, n_2, 2) = \frac{(1.5c^2 - c + 0.5) + (1 - \frac{p}{2})}{2} = \frac{3c^2}{4} - \frac{p}{4} - \frac{c}{2} + \frac{3}{4} \\
E_1(1, n_2, 2) &= E_2(n_1, 1, 2) = (\frac{c}{2})^2 + (1 - \frac{c}{2}) \cdot [(1 - \frac{p}{2}) \cdot \frac{1}{2}(1 + \frac{c}{2})] = \frac{pc^2}{16} - \frac{p}{4} + \frac{c^2}{8} + \frac{1}{2} \\
E_{1,2}(n_1, n_2, 2) &= \frac{1}{2}(1 - \frac{p}{2} + \frac{c}{2}) = \frac{c - p + 2}{4}
\end{aligned}$$

Theorem 4.12. (Recursions in the Rich vs. Poor model, Scenario 6)
The following recursions describe the rich-vs-poor-model with consolation' and 'lucky-loser' rule defined by parameters c and p:

$$\begin{aligned}
E_1(1,n_2,1) &= E_2(n_1,1,1) = \frac{1+c^2}{2}, n_i \geq 1 \\
E_1(n_1,k,1) &= E_2(k,n_2,1) = \frac{1+c}{2}, k \geq 1, n_i > 1 \\
E_{1,2}(1,1,2) &= \frac{-2.25c^4+3c^3-2.5c^2+c-0.25p^2+p-1.25}{p-2} \\
E_1(n_1,1,2) &= E_2(1,n_2,2) = \frac{3c^2-p-2c+3}{4} \\
E_1(1,n_2,2) &= E_2(n_1,1,2) = \frac{pc^2-4p+2c^2+8}{16} \\
E_{1,2}(n_1,n_2,2) &= \frac{c-p+2}{4}
\end{aligned}$$

For all other cases, the formulas to calculate decision parameters s_1 to s_6 and expected points found in Theorem 4.5 apply.

The recursion was programmed in *RichVsPoor6*, which has 5 parameters: p, c and the boundaries for calculation $n1max, n2max, tmax$. Running the program shows that the results for large t resemble those found in Scenario 2, meaning that the compensation once again doesn't influence the „big picture“ (similar to the comparison between scenarios 1 and 4).

Comparing Scenario 6 and Scenario 2 for the variables $p = c = 0.5$ and smaller t gives us results similar to the comparison of Scenarios 1 and 4: For even t, Rich's expectation is reduced for $n_2 < \frac{t}{2}$ and increased for $n_2 \geq \frac{t}{2}$ (and vice versa for Poor and n_1), whereas for odd $t (t > 1)$, Rich's expectation is increased for $n_1 < \lfloor \frac{t}{2} \rfloor$ and reduced for $n_1 \geq \lfloor \frac{t}{2} \rfloor$. It isn't clear whether this is the case for all combinations of c and p, which could be a topic of further research.

4.7 Scenario 7: No Launch, No Win + Consolation Points

Combining these two rules may seem contradictory at first glance - given that with the first rule, nobody wins by not launching, and the other one gives each player a compensation $c \in (0,1)$ if they both can't launch a rocket. This paradox might make for some interesting results though, so we will examine the recursion for this scenario as well.

In $t = 1$, the NLNW rule does not affect the expected points the players will receive (as pointed out in Section 4.3), but the compensation comes into play, so the result is equal to the one gathered in Section 4.4:

$$E_1(1, n_2, 1) = c \cdot c + (1-c) \cdot \frac{1+c}{2}\ ,\ E_1(n_1, n_2, 1) = \frac{1+c}{2} \text{ for } n_1 > 1.$$

and

$$E_1(n_1, 0, 1) = E_2(0, n_2, 1) = \frac{1}{2}, E(n_1 > 1, 0, 2) = E_2(0, n_2 > 1, 2) = \frac{3}{4}$$

For $t = 2$, things now get a little bit tricky, as both rule variations on their own had an effect on the expected points: In Section 4.3 (only NLNW), we found that $E_{1,2}(n_1, n_2, 2) = \frac{1}{2}$ for all $n_1, n_2 \geq 1$, as both players would lose if they didn't manage to launch a rocket in $t = 2$. During the consolation scenario in Section 4.4, we found that the consolation c did influence the decision in $t = 2$ as well. So let's look at the decision tree for Rich when we combine both rules:

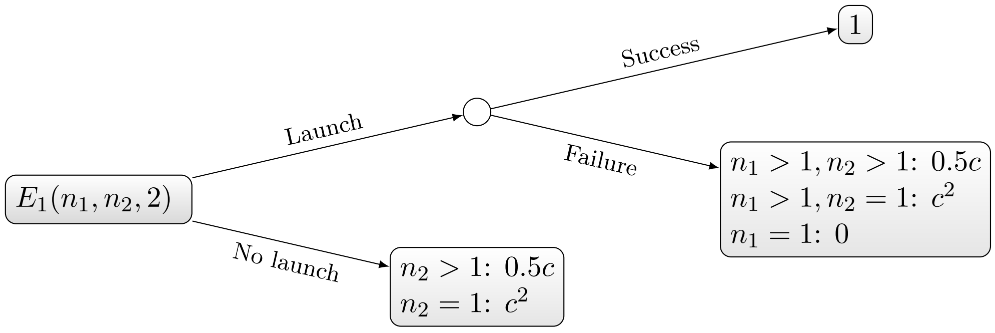


Figure 4.7.1: Decision tree, Scenario 7, $t = 2$

If Poor only has one rocket, he will receive $c \cdot c + (1-c) \cdot \frac{1+c}{2}$. Rich will in this case only get c if Poor doesn't launch, which happens with probability $P(x_1 < c) = c$. If Poor attempts a launch with his last rocket, he will either fail - and lose the compensation for **both** players - or succeed and win the full point. When Poor has more than one rocket in $t = 1$, he will launch for any x_i, failing the attempt half the time, thus giving Rich $\frac{c}{2}$.

$$E_{1,2}(1,1,2) = (c^2)^2 + (1-c^2)\cdot(\frac{1+c^2}{2}) = \frac{1+c^4}{2}$$

$$E_1(n_1,1,2) = E_2(1,n_2,2) = \frac{1+c^2}{2}$$

$$E_1(1,n_2,2) = E_2(n_1,1,2) = (\frac{c}{2})^2 + (1-\frac{c}{2})\cdot(\frac{1+c/2}{2}) = \frac{c^2}{8}+\frac{1}{2}$$

$$E_{1,2}(n_1,n_2,2) = \frac{1+c/2}{2} = \frac{1}{2}+\frac{c}{4}$$

With the NLNW rule in place, we will also have to look at the cases in which one player has no rockets left. Because the consolation is only given out when both players still have rockets, it has no effect here, therefore we also apply the formulas first presented in Section 4.3 in this scenario. Looking back to Sections 4.3 and 4.4, we can see that there are no further cases to examine. In these cases, neither rule effects the nature of the recursion formulas or calculation of decision parameters.

Theorem 4.13. (Recursions in the Rich vs. Poor model, Scenario 7) *The following recursions describe the rich-vs-poor-model with 'no launch, no win' and 'consolation' rule defined by parameter c:*

$$E_1(1,n_2,1) = E_2(n_1,1,1) = \frac{1+c^2}{2} \qquad E_1(n_1,k,1) = E_2(k,n_2,1) = \frac{1+c}{2}, k \geq 1, n_i > 1$$

$$E_1(n_1,0,1) = E_2(0,n_2,1) = \frac{1}{2} \qquad E(n_1,0,2) = E_2(0,n_2,2) = \frac{3}{4}, n_i > 1$$

$$E_{1,2}(1,1,2) = \frac{1+c^4}{2} \qquad E_1(n_1,1,2) = E_2(1,n_2,2) = \frac{1+c^2}{2}$$

$$E_{1,2}(n_1,n_2,2) = \frac{1}{2}+\frac{c}{4} \qquad E_1(1,n_2,2) = E_2(n_1,1,2) = \frac{c^2}{8}+\frac{1}{2}$$

$$E_1(1,0,t) = \frac{1+E_1(1,0,t-1)^2}{2} \qquad E_2(0,1,t) = \frac{1+E_2(0,1,t-1)^2}{2}$$

$$E_1(n_1,0,t) = E_2(0,n_2,t) = \frac{E_n^2\cdot E_{n-1} + E_n^2 - 2E_nE_{n-1} + E_{n-1} + 1}{2} \quad \textit{for } n_1,n_2 > 1,$$

$$E_n = E_1(n_1,0,t-1) \textit{ or } E_2(0,n_2,t-1), E_{n-1} = E_1(n_1-1,0,t-1) \textit{ or } E_2(0,n_2-1,t-1)$$

For all other cases, the formulas to calculate decision parameters s_1 to s_6 and expected points found in Theorem 4.3 apply.

Comparing Scenarios 3 and 7, the added consolation rule leads to the active player's expected points increasing for even t and decreasing for odd t, which these differences quickly converging to 0. Comparing Scenario 4 to 7, the added NLNW rule leads to a decrease in the active player's expectation in $t = 2$, which then alternates - giving an increase in odd t. All these difference converge to 0 quickly as well.

4.8 Scenario 8: Lucky loser + No Launch, No Win + Consolation Points

Finally, we combine all three rule variants in the model. Exploring the recursion, we know that the only rule applying for $t = 1$ is the consolation rule, thus:

$$E_1(1, n_2, 1) = E_2(n_1, 1, 1) = \frac{1 + c^2}{2}, n_i \geq 1$$

$$E_1(n_1, k, 1) = E_2(k, n_2, 1) = \frac{1 + c}{2}, k \geq 1, n_i > 1$$

$$E_1(n_1, 0, 1) = E_2(0, n_2, 1) = \frac{1}{2}, n_i \geq 1$$

For $t = 2$, the decision tree for Rich looks as follows:

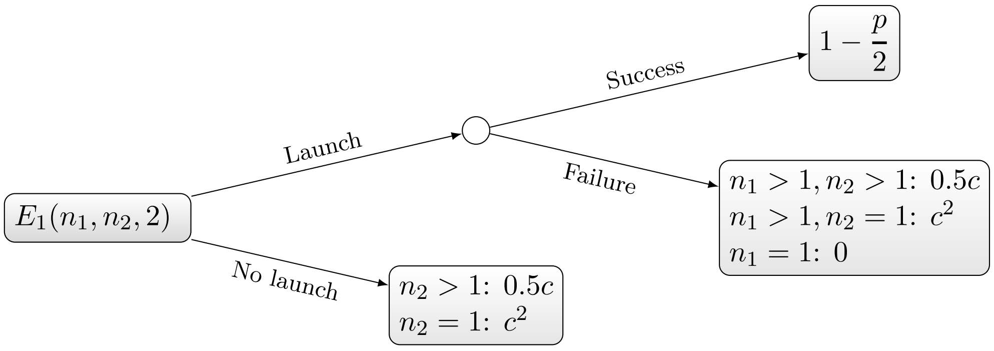


Figure 4.8.1: Decision tree, Scenario 8, $t = 2$

$$\begin{aligned}
E_{1,2}(1, 1, 2) &= (c^2)^2 + (1 - c^2) \cdot (\frac{(1 - p/2) + c^2}{2}) = \frac{2c^4 + c^2p - p + 2}{4} \\
E(n_1, 0, 2) &= E_2(0, n_2, 2) = \frac{3}{4}, n_i > 1 \\
E_1(n_1, 1, 2) &= E_2(1, n_2, 2) = \frac{1 - p/2 + c^2}{2} \\
E_1(1, n_2, 2) &= E_2(n_1, 1, 2) = (\frac{c}{2})^2 + (1 - \frac{c}{2}) \cdot (\frac{1 - p/2 + c/2}{2}) = \frac{c^2 + cp - 2p + 4}{8} \\
E_{1,2}(n_1, n_2, 2) &= \frac{(1 - p/2) + c/2}{2} = \frac{2 - p + c}{4}
\end{aligned}$$

Due to the NLNW rule, we will also consider cases in which one of n_1, n_2 equals zero. For $t \geq 3$, the recursion formulas found in the Lucky loser model (Scenario 2) apply.

Theorem 4.14. (Recursions in the Rich vs. Poor model, Scenario 8)

$$E_1(1,n_2,1) = E_2(n_1,1,1) = \frac{1+c^2}{2} \qquad E_1(n_1,k,1) = E_2(k,n_2,1) = \frac{1+c}{2}, k \geq 1, n_i > 1$$

$$E_1(n_1,0,1) = E_2(0,n_2,1) = \frac{1}{2}, n_i \geq 1 \quad E(n_1,0,2) = E_2(0,n_2,2) = \frac{3}{4}, n_i > 1$$

$$E_{1,2}(1,1,2) = \frac{2c^4 + c^2p - p + 2}{4} \qquad E_1(n_1,1,2) = E_2(1,n_2,2) = \frac{1 - p/2 + c^2}{2}$$

$$E_{1,2}(n_1,n_2,2) = \frac{2 - p + c}{4} \qquad E_1(1,n_2,2) = E_2(n_1,1,2) = \frac{c^2 + cp - 2p + 4}{8}$$

$$E_1(1,0,t) = \frac{1 + E_1(1,0,t-1)^2}{2} \qquad E_2(0,1,t) = \frac{1 + E_2(0,1,t-1)^2}{2}$$

$$E_1(n_1,0,t) = E_2(0,n_2,t) = \frac{E_n^2 \cdot E_{n-1} + E_n^2 - 2E_nE_{n-1} + E_{n-1} + 1}{2} \ \textit{for } n_1, n_2 > 1,$$

$$E_n = E_1(n_1,0,t-1) \ \textit{or } E_2(0,n_2,t-1), E_{n-1} = E_1(n_1-1,0,t-1) \ \textit{or } E_2(0,n_2-1,t-1)$$

For all other cases, the formulas to calculate decision parameters s_1 to s_6 and expected points found in Theorem 4.5 apply.

Comparing the results to the two-rule-change scenarios didn't give us new information.

5 The SpaceX Model

„SpaceX designs, manufactures and launches advanced rockets and spacecraft. The company was founded in 2002 to revolutionize space technology, with the ultimate goal of enabling people to live on other planets.“[1]

A significant part of the „revolution“SpaceX is working on are re-usable rocket/ rocket stages. Many recent missions have been satellite deployments, in which first, the flagship Falcon 9 rocket was launched and brought the payload into orbit via the Dragon spacecraft. Afterwards, the first stage of the Falcon 9 is directed back to Earth and landed either on land or on a drone-ship in the ocean. Milestones included:

- First soft water landing: April 2014[2]
- First successful landing on ground: December 21, 2015[3]
- First successful landing on drone-ship: April 8, 2016[4]

The development of re-usable rockets is the central part of reducing space travel costs to make repeated Mars flights financially feasible - these are planned for late 2022.[5] Based on rocket launches and landings, we will now build another model.

5.1 Two-Parameter Model

In the simplest version of this model, we will use two variables to describe the decision-making of the starter, who has only a single rocket. One, called x_t, is the probability for a successful rocket launch. It is equivalent to the x_t used in Sections 3 and 4, and will be uniformly distributed in $(0, 1)$. The second value, y_t describes the probability of a successful landing of the rocket, whether on land or on sea. In the first version of the model, y_t will be independent from x_t with the same uniform distribution on $(0, 1)$.

The starter will gain 1 point if both launch and landing of the rocket are a success, and receive $a \in [0, 1]$ points if only the landing fails. If the launch fails, he will receive no points. Analogously to Sections 2 and 3, we define x_1 and y_1 as the success probabilities on the last day before time runs out. The starter knows the values of the current day and we will once again use recursion to gather the expectation E_t of the starter if he makes optimal decisions every day. For $t = 1$, we get

In $t = 1$, the starter has no more incentive to wait, given he will get no points if time runs out. Therefore, he will try launching with any (x_1, y_1) and his expected points return is

[1]`http://spacex.com/about/`, opened 1st October 2016
[2]Video: `https://www.youtube.com/watch?v=CQnR5fhCXkQ`
[3]Video: `https://www.youtube.com/watch?v=1B6oiLNyKKI`
[4]Video: `https://www.youtube.com/watch?v=sYmQQn_ZSys`
[5]`http://www.spacex.com/sites/spacex/files/mars_presentation.pdf`, page 47

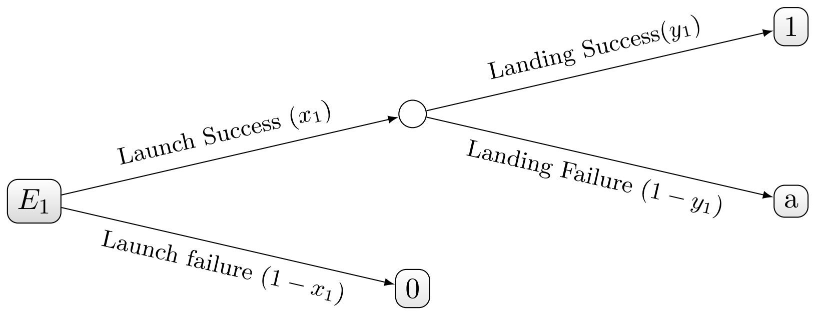


Figure 5.1.1: SpaceX model, $t = 1$

$$\begin{aligned} E_1 &= P(\text{launch failure}) \cdot 0 \\ &+P(\text{launch success}) \cdot (P(\text{landing success}) \cdot 1 + P(\text{landing failure}) \cdot a) \\ &= (1 - x_1) \cdot 0 + x_1 \cdot (y_1 \cdot 1 + (1 - y_1) \cdot a), \text{and given } E(x_1) = E(y_1) = \frac{1}{2} \\ &= \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot (\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot a) = \frac{1}{4} + \frac{1}{4} \cdot a \end{aligned}$$

For $t = 2$, the starter now also has the option to not attempt a launch at all. The decision tree can be found below - to calculate the expected points, we will once again compare the value of waiting, i.e. E_1, to the expected points of launching the rocket LV, given by:

$$LV = (1 - x_2) \cdot 0 + x_2 \cdot (y_2 \cdot 1 + (1 - y_2) \cdot a) = x_2 \cdot (y_2 + (1 - y_2) \cdot a)$$

As we can see, the expected launch value will always be 'constant' for any t. Nevertheless we still have to consider LV as a random value depending on both input variables.

$$\begin{aligned} E(LV) &= E[x_i \cdot (y_i + (1 - y_i) \cdot a)] \\ &= E[x_i \cdot y_i + x_i \cdot (1 - y_i) \cdot a] \\ &= E(x_i \cdot y_i) + E(x_i \cdot y_i) \cdot a = \frac{1}{4} + \frac{1}{4} \cdot a. \end{aligned}$$

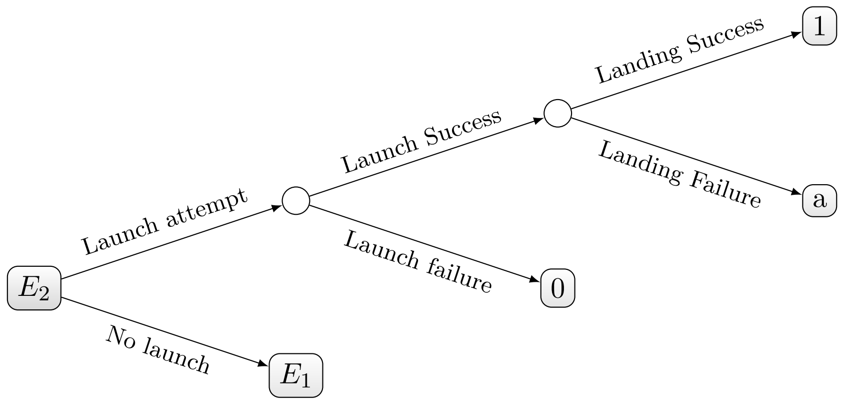


Figure 5.1.2: SpaceX model, $t = 2$

The recursion formula for $t > 2$ is given by

$$E_t = P(LV < E_{t-1}) \cdot E_{t-1} + P(LV \geq E_{t-1}) \cdot E(LV|LV > E_{t-1})$$

For the recursion, we have to examine the distribution of LV. Given both x_t and y_t and random, uniformly distributed variables, the cumulative distribution function of LV is pretty complicated.
$F_M(m) = P(M \leq m) = E[1_{\{M \leq m\}}] = \int_{M \leq m} f_(x, y) d(x, y)$
An alternative for evaluating the distribution and density functions of LV is based in the use of Monte-Carlo algorithm to get an approximation of the expected launch value. The pseudo-code for the algorithm can be found below, the Scilab implementation on the CD. This Monte-Carlo method has another advantage besides avoiding having to calculate complicated density functions: We can also easily substitute the uniformly, independent random variables with other, 'customized' distributions, and check the (approximated) results there. This can be done easily by editing the line in which the random numbers are generated.

Algorithm 1 Monte Carlo Algorithm for SpaceX Model

```
function SpaceX($a, t, mc$)    ▷ $a$ payout for successful launch, $t$ number of recursions, $mc$ number of random numbers generated for each day.
    $E \leftarrow zeros(t, 1)$
    $E_1 \leftarrow 1/4 + (1/4) * a$
    $P \leftarrow zeros(t, 1)$
    $CE \leftarrow zeros(t, 1)$
    for $i \leftarrow 2$ to $t$ do
        $Z \leftarrow$ zeros(mc,1)
        $k \leftarrow 0$                                    ▷ Counter for $LV < E_{t-1}$
        $s \leftarrow 1$                                    ▷ Counter for $LV \geq E_{t-1}$
                                                            ▷ Calculating P(Launch value > Wait value)
        for $j \leftarrow 1$ to $mc$ do
            $R \leftarrow rand(2, 1)$                       ▷ Randomize $x_t$ and $y_t$
            $LV \leftarrow R_1 * (R_2 + (1 - R_2) * a)$     ▷ Calculate launch value
            if $LV < E_{i-1}$ then                          ▷ Launch value under 'wait' value
                $k \leftarrow k + 1$
            else
                $s \leftarrow s + 1$
                $Z_s \leftarrow LV$;                        ▷ Save launch value for later calculation
            $j \leftarrow j + 1$
        $CE_i \leftarrow (\sum Z)/s$                        ▷ Calculate the conditional expectation
        $P_i \leftarrow k/mc$                               ▷ Calculate $P(LV < E_{i-1})$
        $E_i \leftarrow P_i * E_{i-1} + (1 - P_i) * B_i$
    return $E, P, Z, B$
```

Let's take a look at the results: First a comparison between the expected points E_t, $t = 1, ..., 50$ for $a = 0$ (blue), $a = 0.25$ (green), $a = 0.5$ (violet), $a = 0.75$ (red) and $a = 1$ (black) in Figure 5.1.3. For up to $t = 100$ and with a Monte-Carlo parameter of 10^5, the algorithm took about 35 seconds for each variation of a on a standard PC.
We can see that E_t converges to 1 more quickly for higher a, which is obvious since it increases the players expected points right from $t = 1$. In $a = 1$, in which the landing doesn't matter, we receive the same model as already discussed in Section 2. To illustrate the different growth of E_t for different a, *SpaceXbarrier* was used.

It calculates $\min(t \mid E_t > b)$ for any given barrier(s) b. The results for various a and barriers $b = 0.8, 0.9, 0.95, 0.99, 0.999$ with $mc = 10^5$ can be found in Table 5.1.1. For $a = 1$, we used the formula found in Section 5.2 to evaluate E_t exactly. The estimations for $b = 0.999$ are based on the seemingly linear growth of the „barrier E_t for $b = 0.8, 0.9, 0.95$ and 0.99.
This means that for low a, any player that wants a 99% chance of getting the full point must have a lot of time at his hands - for $a = 0.4$ and one launch opportunity per day, he would need to have around eighteen-thousand days, or about *forty-nine years*. This is an eternity compared to the just over six months he needs to have for a 99% chance of succeeding when $a = 1$, i.e. when landing is irrelevant.

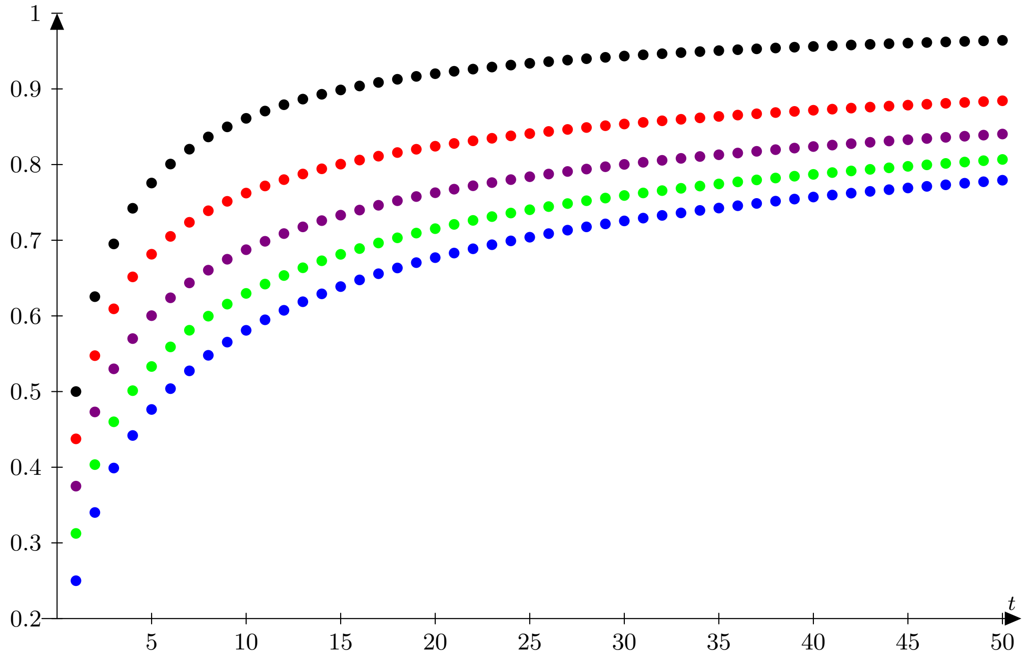


Figure 5.1.3: SpaceX model, E_t for t up to 50, $a = 0, 0.25, 0.5, 0.75, 1$, $mc = 10^5$

Table 5.1.1: Barriers in the SpaceX model (*=exact calculation)

a \ b	0.8	0.9	0.95	0.99	0.999
0*	63	279	1,163	29,840	2,998,486
0.2	53	226	931	23,906	$2.4 \cdot 10^6$ (est.)
0.4	40	170	698	17918	$1.8 \cdot 10^6$ (est.)
0.6	28	114	468	11,922	$1.2 \cdot 10^6$ (est.)
0.8	16	59	235	5,973	$0.6 \cdot 10^6$ (est.)
1*	6	16	35	193	1,991

5.2 Multi-Player-Teams With Majority Decision

The next model features multiple players collaborating on a space travel mission. Imagine a rocket to deploy multiple satellites in orbit, or delivering multiple rovers or probes to another object in space. To save costs, companies might choose to work together to maximize efficiency and reduce costs. For this scenario, we define the following model:

Definition 5.1. (Team model with majority decision)
- n players are using a single rocket for their individual missions
- t days to launch the rocket, otherwise mission fails
- Common parameter $x_i \sim U(0,1)$ denotes launch success probability for i-last day
- $y_{i,k} \sim U(0,1)$ denotes probability of player k succeeding in his individual mission
- Each player only sees x_i and his own $y_{i,k}$
- Regular Mission success: Launch and majority of individual players successful.
- E_t probability of regular mission success with t remaining days.
- M_t personal mission success probability for each player with t days left.
- All $x_i, y_{i,k}$ are independent.

In this thesis, we will focus on the case $n = 3$. This gives us:

Regular mission success (RMS):
Launch and at least two of the three players' missions are successful.

Total mission success (TMS):
Launch and all three players' missions are successful.

The team will try to maximize E_t. We can also define two voting modes:

Launch by simple majority:
Launch attempt if at least two of the three players agree to it.

Launch by unanimous decision:
Launch attempt only if all three players agree to it.

Before we take a look at the decision making in this model, we can already evaluate E_t (the expected chance of a RMS) for $t = 1$. Given that the mission will fail when time runs out, there is no voting necessary and players are forced to attempt a launch given any $x_1, y_{1,1}, y_{1,2}, y_{1,3}$:

$$\begin{aligned} E_1 &= P(RMS) = P(\text{at least two players and launch successful}) \\ &= E(x_1) \cdot (P(\text{three missions successful}) + P(\text{two missions successful})) \\ &= \frac{1}{2} \cdot [(\frac{1}{2})^3 + 3 \cdot (\frac{1}{2} \cdot \frac{1}{2} \cdot (1 - \frac{1}{2})] = \frac{1}{2} \cdot [\frac{1}{8} + \frac{3}{8}] = \frac{1}{4}. \end{aligned}$$

For $t \geq 2$, we now have to take a look at the decision making: For this, we have to analyze what parameters are known to each player. In the definition, we stated that he only sees x_t, the launch parameter, and his individual mission parameter y_i. One could argue that each player also knows E_{t-1}, i.e. the mission success rate if the day is skipped.

The player can use x_t and $y_{t,i}$ and calculate his personal success probability (PSP):

$$PSP(t,i) = x_t \cdot y_{t,i}, \text{ for } i = 1,2,3.$$

Now he has a value on which base he can make a decision. In the previous chapters, it was always quite clear how a player makes the „right“ decision. Now, it's not quite as clear.
He can now compare his PSP to his personal expected win rate in t, given by M_t, for $t = 1$ this gives us

$$M_1 = E(x_1) \cdot E(y_{1,i}) = \frac{1}{4} \text{ for } i = 1,2,3.$$

This result is easy to explain: With only the last day left, the player will make an attempt at launch and individual mission in any case, similarly to the decisions in the previous sections. He therefore will have the expected value of x_1 and $y_{1,k}$, which given their distribution both equal $\frac{1}{2}$, getting multiplied with each other due to their independence. For $t = 2$, things get a bit more complex - we will leave out the k parameter for now, as M_t will be the same for each player:

$$M_2 = P(x_2 \cdot y_2 \leq M_1) \cdot M_1 + P(x_2 \cdot y_2 > M_1) \cdot E(x_2 \cdot y_2 | x_2 \cdot y_2 > M_1)$$
$$M_2 = P(x_2 \cdot y_2 \leq \frac{1}{4}) \cdot \frac{1}{4} + P(x_2 \cdot y_2 > \frac{1}{4}) \cdot E(x_2 \cdot y_2 | x_2 \cdot y_2 > \frac{1}{4})$$

In the first part of the formula, the PSP is lower than the expected value of the last day, the player will therefore vote to skip the penultimate day and try on the last day.
In the second part, the player will vote for a mission attempt, getting the conditional expectation of $x_2 \cdot y_2$ under the condition that this is greater than $M_1 = \frac{1}{4}$. To find this expectation, we will have to calculate the probability density function (PDF) f and cumulative distribution function (CDF) F of $z = x \cdot y$, with $x, y \sim U(0,1)$. We get the latter by using its definition:

$$F_Z(z) = P(Z \leq z) = \int\limits_{x=0}^{1} P(y \leq \frac{z}{x}) \cdot f_X(x) dx = \int\limits_{x=0}^{1} dx + \int\limits_{x=z}^{1} \frac{z}{x} dx = z - z \cdot \log z$$

Therefore, the PDF of $z = x \cdot y$ is given by $f(z) = -\log z, z \in (0,1]$

With these two functions, we can calculate both $P(x_2 \cdot y_2 \leq M_1)$ and $E(x_2 \cdot y_2 | x_2 \cdot y_2 > M_1)$. With $M_1 = \frac{1}{4}$, we get

$$P(x_2 \cdot y_2 \leq \frac{1}{4}) = F(\frac{1}{4}) = \frac{1}{4} - \frac{1}{4} \cdot \log(\frac{1}{4}) \approx 0.59657$$

$$E(x_2 \cdot y_2 | x_2 \cdot y_2 > \frac{1}{4}) = \frac{\int_{1/4}^{1} x \cdot f(x)}{1 - F(\frac{1}{4})} = \frac{\int_{1/4}^{1} -\log(x) \cdot x dx}{\frac{3}{4} + \frac{1}{4} \cdot \log(\frac{1}{4})} = \frac{0.19105...}{0.40342...} \approx 0.47358$$

Finally, for M_2 we receive

$$M_2 = 0.59657 \cdot \frac{1}{4} + 0.40343 \cdot 0.47358 \approx 0.340197$$

This value will also be the base for the decision of each player in $t = 3$. The general recursion for $M_t, t > 1$ can be evaluated as

$$M_t = P(x_t \cdot y_t \leq M_{t-1}) \cdot M_{t-1} + P(x_t \cdot y_t > M_{t-1}) \cdot E(x_t \cdot y_t | x_t \cdot y_t > M_{t-1})$$

$$= F(M_{t-1}) \cdot M_{t-1} + (1 - F(M_{t-1})) \cdot E(x_t \cdot y_t | x_t \cdot y_t > M_{t-1})$$

and with $F(x) = x - x \cdot \log x$ this becomes

$$= (M_{t-1} - M_{t-1} \cdot \log M_{t-1}) \cdot M_{t-1} + (1 - (M_{t-1} - M_{t-1} \cdot \log M_{t-1})) \cdot \frac{\int\limits_{M_{t-1}}^{1} -\log z \cdot z \, dz}{1 - (M_{t-1} - M_{t-1} \cdot \log M_{t-1})}$$

$$= (M_{t-1} - M_{t-1} \cdot \log M_{t-1}) \cdot M_{t-1} + (1 - (M_{t-1} - M_{t-1} \cdot \log M_{t-1})) \cdot \frac{[\frac{z^2}{4} - \frac{1}{2} z^2 \log z]_{M_{t-1}}^{1}}{1 - (M_{t-1} - M_{t-1} \cdot \log M_{t-1})}$$

$$= \ldots = -\frac{M_{t-1}^2}{4} + \frac{1}{2} M_{t-1}^2 \log M_{t-1} + M_{t-1}(M_{t-1} - M_{t-1} \log M_{t-1}) + \frac{1}{4}.$$

Similar to the simple one player model in Section 2, M_t should asymptotically equal 1 for $t \to \infty$. We can easily see that this is the case: If we put in $M_{t-1} = 0$ into the expression, all parts of the expression equal 0, plus the constant $\frac{1}{4}$. If we enter $M_{t-1} = 1$, we get

$$M_t = -\frac{1^2}{4} + \frac{1}{2} 1^2 \log 1 + 1(1 - 1 \cdot \log 1) + \frac{1}{4} = -\frac{1}{4} + 0 + 1 + \frac{1}{4} = 1.$$

Now, given $M_t \in (0, 1)$, the only thing left to show is that the expression is monotonically increasing on $(0, 1)$ by evaluating the derivative:

$$(-\frac{M_{t-1}^2}{4} + \frac{1}{2} M_{t-1}^2 \log M_{t-1} + M_{t-1}(M_{t-1} - M_{t-1} \log M_{t-1}) + \frac{1}{4})' = M_{t-1} - M_{t-1} \cdot \log M_{t-1}.$$

This is positive for $0 < M_t < e$, which is more than enough for the interval $(0, 1)$.

After we found the decision parameter for each of the players, all there's left to analyze is the decision making in the team. Because we work with simple majority voting, at least two votes for launching will trigger the launch attempt, whereas at least two votes for waiting will mean that no launch is made on this day. Let's look at three examples to illustrate how it works:

Table 5.2.1: Decision making, Team model, $t = 2$

x_2	M_1	$y_{2,1}$	$y_{2,2}$	$y_{2,3}$	Vote 1	Vote 2	Vote 3	Overall
0.75	0.25	0.4	0.6	0.3	Yes	Yes	No	Yes by 2-1
0.5	0.25	0.2	0.8	0.1	No	Yes	No	No by 2-1
0.9	0.25	0.7	0.3	0.4	Yes	Yes	Yes	Yes by 3-0

Example 5.2. (Decision in Team model, examples for $t = 2$)
In the first row, there's a 2-to-1 majority for a launch, as player 1 has a personal success probability (PSP) of $0.75 \cdot 0.4 = 0.3 > 0.25$ and player 2 votes Yes due to $0.75 \cdot 0.4 = 0.3 > 0.25$, whereas player 3 votes No because $0.75 \cdot 0.3 = 0.2375 < 0.25$. In the second row, we find two players voting against a launch attempt, as their PSP is less than M_1. In row 3, a large x_2 leads to all players having a relatively low threshold for their personal parameter, and all three vote for launching.

Finally, we will examine the recursion for E_t: Let's first go through the Vote result probabilities step-by-step, starting with $P(0 \text{ Yes})$, which is the probability for an unanimous „No" vote:

$$\begin{aligned} P(0 \text{ Yes}) &= P(x_t \leq M_{t-1}) \\ &+ P(x_t \cdot y_{t,1} < M_{t-1} \wedge x_t \cdot y_{t,2} < M_{t-1} \wedge x_t \cdot y_{t,3} < M_{t-1} \wedge x_t > M_{t-1}) \end{aligned}$$

The first part represents the case in which the launch parameter x_t is smaller than the decision parameter M_{t-1}. Because the individual parameters of the players are distributed in $(0, 1)$, this automatically means that no player will vote for a start, as $x_t \cdot y_{t,i} < M_{t-1}$. The second part denotes the case in which x_t is sufficiently large, but all three PSPs are smaller than M_{t-1} anyway, resulting in 0 Yes votes.

$$\begin{aligned} P(0 \text{ Yes}) &= M_{t-1} + P(y_{t,1} < \frac{M_{t-1}}{x_t} \wedge y_{t,2} < \frac{M_{t-1}}{x_t} \wedge y_{t,3} < \frac{M_{t-1}}{x_t} \wedge x_t > M_{t-1}) \\ &= M_{t-1} + F_y(\frac{M_{t-1}}{x_t}) \cdot F_y(\frac{M_{t-1}}{x_t}) \cdot F_y(\frac{M_{t-1}}{x_t}) \cdot (1 - M_{t-1}) \\ &= M_{t-1} + (\frac{M_{t-1}}{x_t})^3 \cdot (1 - M_{t-1}), \text{ with } x_t \sim U(m, 1). \end{aligned}$$

This can be simulated by using Monte-Carlo algorithms. Same goes for $P(1 \text{ Yes})$, which can be calculated similarly - the factor 3 comes in due to any of the three players giving the single Yes vote:

$$\begin{aligned} P(1 \text{ Yes}) &= 3 \cdot (P(y_{t,1} > \frac{M_{t-1}}{x_t} \wedge y_{t,2} < \frac{M_{t-1}}{x_t} \wedge y_{t,3} < \frac{M_{t-1}}{x_t} \wedge x_t > M_{t-1}) \\ &= 3 \cdot [(1 - F_y(\frac{M_{t-1}}{x_t})) \cdot F_y(\frac{M_{t-1}}{x_t}) \cdot F_y(\frac{M_{t-1}}{x_t}) \cdot (1 - M_{t-1})] \\ &= 3 \cdot [(1 - \frac{M_{t-1}}{x_t}) \cdot (\frac{M_{t-1}}{x_t})^2 \cdot (1 - M_{t-1})], \text{ with } x_t \sim U(m, 1). \end{aligned}$$

Now we simply put together both results to receive the probability of a simple majority vote for „no launch" :

$$P(< 2 \text{ Yes}) = P(0 \text{ Yes}) + P(1 \text{ Yes}) = \frac{2(m^4 - m^3)}{x^3} + \frac{3(m^2 - m^3)}{x^2} + m$$

with $m = M_{t-1}$ and $x_t \sim U(m, 1)$. As one might expect, this probability quickly rises to 1 along with M_t, because it becomes infinitesimally unlikely that the players agree to starting when all their thresholds for voting 'Yes' are near 1.

The probability of a simple majority voting for the launch is then given by the complementary probability

$$P(\geq 2 \text{ Yes}) = 1 - [P(0 \text{ Yes}) + P(1 \text{ Yes})]$$

For the unanimous decision voting, it is useful to know $P(3 \text{ Yes})$, which is

$$\begin{aligned} P(3 \text{ Yes}) &= P(y_{t,1} > \frac{M_{t-1}}{x_t} \wedge y_{t,2} > \frac{M_{t-1}}{x_t} \wedge y_{t,3} > \frac{M_{t-1}}{x_t} \wedge x_t > M_{t-1}) \\ &= (1 - F_y(\frac{M_{t-1}}{x_t})) \cdot (1 - F_y(\frac{M_{t-1}}{x_t})) \cdot (1 - F_y(\frac{M_{t-1}}{x_t})) \cdot (1 - M_{t-1}) \\ &= (1 - \frac{M_{t-1}}{x_t})^3 \cdot (1 - M_{t-1}), \text{ with } x_t \sim U(m, 1). \end{aligned}$$

We have now established the voting probabilities for both simple majority and unanimous voting. Now the question remains: How is E_t calculated for $t > 1$. For E_t, which is the expected rate of regular mission successes (RMS), we get the following formula:

$$E_t = P(< 2 \text{ Yes}) \cdot E_{t-1} + P(\geq 2 \text{ Yes}) \cdot E(P(\text{RMS})| \geq 2 \text{ Yes votes})$$

If fewer than two players vote Yes, the day is skipped and therefore E_{t-1} is the expected return. Now, if at least two players vote Yes, a launch attempt is made. The return is the expected probability of at least a regular mission success under the condition that two or more „Yes“ votes are casted by the players. We know that $P(\text{RMS or TMS})$ equals

$$P(\text{RMS}) = x_t \cdot y_{t,1} \cdot y_{t,2} \cdot y_{t,3} + 3 \cdot (x_t \cdot (1 - y_{t,1}) \cdot y_{t,2} \cdot y_{t,3}).$$

It's easy to see that $P(\text{RMS}) \to 1$ for $x_t, y_{t,i} \to 1$.
Previously, we found: $M_t \to 1$ for $t \to \infty$.

For any chance at a simple majority vote for a launch it was necessary that $x_t > M_{t-1}$. This led us to the formula

$$P(\geq 2 \text{ Yes}) = 1 - [P(0 \text{ Yes}) + P(1 \text{ Yes})] = 1 - (\frac{2(m^4 - m^3)}{x^3} + \frac{3(m^2 - m^3)}{x^2} + m)$$

with $x_t \sim U(M_{t-1}, 1) \to 1$ for $t \to \infty$.
Under the assumption that $y_{t,i}$ rise proportionally - which they should, as otherwise the PSP would quickly become smaller than M_{t-1} - we therefore get:

$$P(\geq 2 \text{ Yes votes}, t) \to 0 \text{ for } t \to \infty$$

This means that for rising t, a launch attempt becomes infinitesimally unlikely. Same goes for the probability of a unanimous „Yes“ vote for a launch, which should converge to 0 even quicker, as $P(3 \text{ Yes}) < P(\geq 2 \text{ Yes})$. For reference, let's take a look at some values of $P(3 \text{ Yes})$ and $P(\geq 2 \text{ Yes})$, approximated for a Monte-Carlo parameter of 10^4: This can be found in Table 5.2.2 below.
Now, what is the conditional expectation of $P(\text{at least RMS})$? This seems to be difficult to calculate given that you have to first find the density/probability function of it and then work in the condition, which in itself is at least very tricky to work out.

Table 5.2.2: Probability of majority Yes Vote in Team Model

t	2	5	10	100	1000
P(3 Yes) - unanimous	0.1548	0.0374	0.0121	0.0002	$2 \cdot 10^{-6}$
$P(\geq$ 2 Yes) - simple majority	0.4203	0.1703	0.0816	0.0043	0.00015

What we *can* do is to find an upper bound for E_t. As we already pointed out, the probability of a Yes vote, no matter if by simple or unanimous majority, converges to 0 for rising t. Now, what happens in the rare cases when indeed a majority votes for a launch? The two (or three) players voting must feel extremely safe to take the risk of launching, meaning that for large t, their mission part will almost surely be successful, meaning that E(P(RMS)) should also converge to 1 for rising t. And this is exactly the upper bound we were looking for:

$$E_t^S \leq P(< 2 \text{ Yes}) \cdot E_{t-1} + P(\geq 2 \text{ Yes}) \cdot 1$$

for simple majority voting and

$$E_t^U \leq P(< 3 \text{ Yes}) \cdot E_{t-1} + P(3 \text{ Yes}) \cdot 1$$

for unanimous voting. Below you can see a plot of both upper bounds.

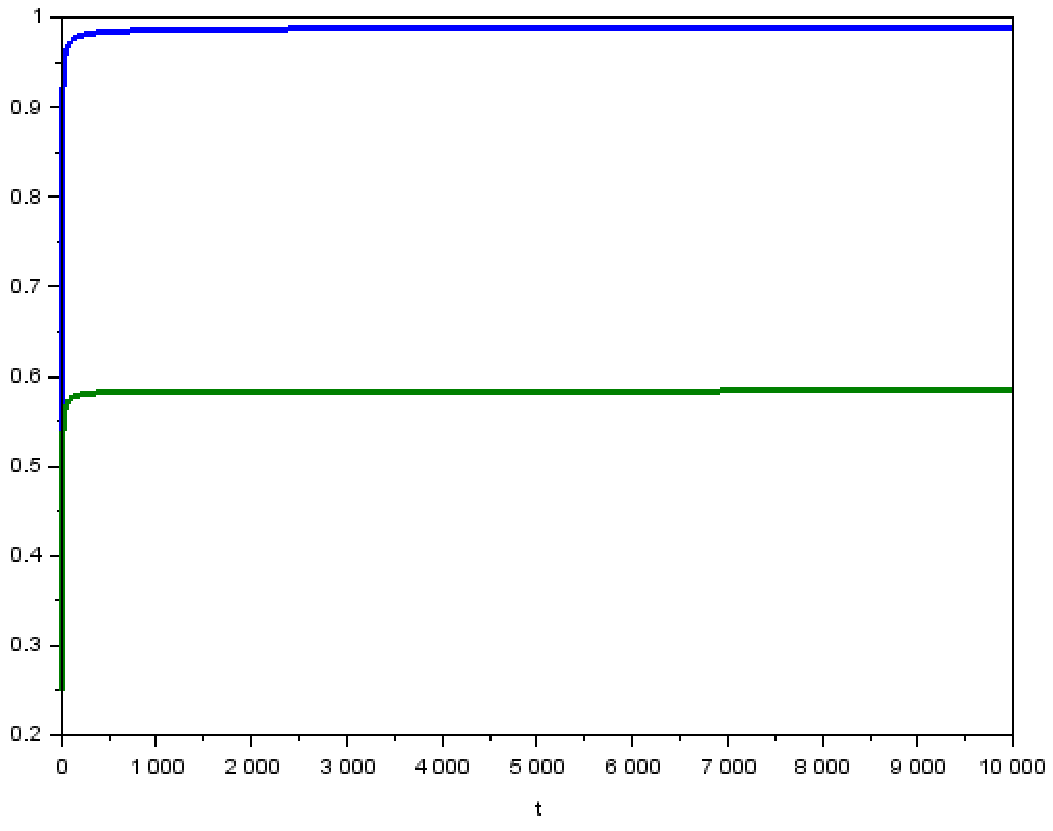


Figure 5.2.1: Upper bound for E_t^U (green) and E_t^S (blue)

As we can see, the upper bound of expected value of the team voting by simple majority is way bigger than that of the unanimously-voting team. Looking at limits for large t, it seems that E_t^S does not quite reach 1. This seems suspicious, as one would think that for large enough t, the team would always find a spot in which a mission success is almost-surely guaranteed.

But looking at the upper bound of E_t^U, we see that it doesn't even seem to reach 0.6. A assumed limit of $2 - \sqrt{2} getspassedforlargeenought$. For $t = 10000$, the values calculated with a Monte-Carlo parameter of $mc = 10^4$ are

$$E_t^S \approx 0.98854, E_t^U \approx 0.58381$$

For $t = 10^5, mc = 10^3$ it is

$$E_t^S \approx 0.98928, E_t^U \approx 0.58415$$

Why is the expectation of the unanimous team so low? The assumption is that it's due to the infinitesimally small $P(3 \text{ Yes})$ for large t, which leads to the team letting time tick down to a (comparably) very small t and then attempting a „panic-launch“ with a lower chance of success, whereas the team with the simple majority voting starts more often for larger t and is more successful. As only two of three players have to succeed in their mission for it to be an overall success, the mission will most likely succeed when two players are almost-surely succeeding. If three players have to vote unanimously, they probably stand in each other's way more than they are helping the team. Or as a German saying goes: Too many cooks spoil the broth!

A potential limit of E_t^U could be $2 - \sqrt{2} \approx 0.5878$, but this is just speculation. It is also unclear how sharp this upper bound actually is, as the. This is something that can perhaps be determined in future research as well.

5.3 SpaceX meets Judo - Three-Parameter-Model

The two things might not have anything in common at first glance, but the scoring in the Japanese combat sport Judo during the 2016 Summer Olympics gave inspiration for this model. For those not familiar with the sport, here is how the scoring works:[6]

Scoring in Judo (simplified)

- **Ippon:** If one of the judokas throws his opponent on his back "with impetus and control", they score an ippon, and instantly win the contest.
- **Waza-ari:** If one of the judokas throws his opponent on his back with less force than necessary for an ippon, they score a waza-ari (half-point). If one judoka gets two waza-aris, they win the contest.
- **Yuko:** This is the small tiebreaker, given to a judoka who manages to throw his opponent on his side. Yukos will be the tiebreaker in case both judokas are tied in waza-aris.
- **Shido:** There are also penalties/infractions for minor offenses against the rules, for example too passive behavior, or leaving the fighting area without permission. If one of the judokas receives **four** shidos, he is disqualified and his opponent wins the match. If at the end of the match both judokas are tied in waza-aris and yukos, the one with less shidos (infractions) will be declared winner.

If the judokas are tied in all four categories at the end of the fight, the "Golden Score" rule comes into effect. The match continues until the first contestant gets any score and wins the match.

Table 5.3.1: Judo scoring examples

Player	Ippon	Waza-ari	Yuko	Shido	Result
A	1	0	0	2	Winner by Ippon
B	0	1	3	1	Loser by Ippon
A	0	1	0	1	Winner by Waza-ari
B	0	0	2	0	Loser by Waza-ari
A	0	0	1	3	Loser by Shido
B	0	0	1	1	Winner by Shido

Example 5.3. (Three scoring scenarios)
In Example 1, judoka A is behind on all three smaller categories, but has managed to score an ippon, winning the match. In Example two, judoka A wins because he has achieved more waza-aris than his opponent. In Example 3, both judokas are tied on actual fight scoring, with judoka B winning the match by receiving fewer shidos than A.

[6] `http://www.intjudo.eu/upload/2015_04/20/142952199282442702/2015_ijf_refereeing_rules_english.pdf`,
or `http://www.nbcolympics.com/news/judo-101-rules-scoring`

Now we can build an analogous model with rocket launches. This can be done as a single-player or two-player model, we will focus on the model with two contestants in this thesis. First, we split the mission of the space organization in three distinct parts: In real life, these could be rocket launch, satellite deployment and rocket landing. Time is once again counting down from a fixed $T \in \mathbb{N}$. All three parts have independent success probabilities: $x_i, y_i, z_i \sim U(0,1)$. Now, the scoring will work as follows:

- **Ippon / Full mission success**: All three parts of the mission succeed. Probability given by $x_i \cdot y_i \cdot z_i$.
- **Waza-ari / Deployment success:** Launch and satellite deployment are successful, but the landing fails. Probability given by $x_i \cdot y_i \cdot (1 - z_i)$.
- **Yuko / Launch success:** The launch is successful, but the deployment of the satellite goes wrong. The state of the landing will be irrelevant in this case. Probability given by $x_i \cdot (1 - y_i)$.
- **Shido / Launch failure:** If the launch fails, the organization will receive an infraction. Probability given by $(1 - x_i)$.

Now, the rules can be varied or be kept the same as in Judo:

- Number of full mission successes resulting in win: m (standard value 1).
- Number of deployment successes resulting in win: d (standard value 2).
- Number of launch failures resulting in disqualification: f (standard value 4).
- Later added: *Number of launch successes resulting in win: s*

The turn order is once again strictly alternating. We can vary between giving both organization a fixed or infinite number of rockets/satellites. Looking at the number of parameters we have to take into account for the recursion, we see that even without a finite number of rockets and winning by having a full mission success, there are already seven parameters in the expected points, as we see in the following definition:

Definition 5.4. (Expected points in the Judo model)
$E_i(d_1, d_2, s_1, s_2, f_1, f_2, t)$ gives the expected points for player i ($i = 1, 2$) when the players have d_1, d_2 successful deployments, s_1, s_2 successful launches and f_1, f_2 failed launches with t days left.

Note: A successful deployment does not influence s_1 and s_2. This definition assumes that one full mission success is enough for a win. If it isn't, there needs to be two more parameters, m_1 and m_2 counting the number of mission successes by the two players. If we gave the players a finite amount of rockets, there would be another two parameters n_1, n_2 for the number of rockets they have left.

For the standard model, we can see that $d_i \in \{0, 1\}$ and $f_i \in \{0, 1, 2, 3\}$, because higher values would lead to a win ($d_i > 1$) or a disqualification ($f_i > 3$). The number of successful launches s_i is not limited because simple launches are only used as a tiebreaker in $t = 0$. Therefore $s_i \in \mathbb{N}$, as well as obviously $t \in \mathbb{N}$. Instead of using the golden score rule, we will give each player half a point if they are tied by all four categories. Now, let's examine how a winner gets determined in $t = 0$:

$$E_1(d_1, d_2, s_1, s_2, f_1, f_2, 0)$$
$$= \begin{cases} 1 & (d_1 > d_2) \vee (d_1 = d_2 \wedge s_1 > s_2) \vee (d_1 = d_2 \wedge s_1 = s_2 \wedge f_1 < f_2) \\ \frac{1}{2} & (d_1 = d_2) \wedge (s_1 = s_2) \wedge (f_1 = f_2) \\ 0 & else \end{cases}$$

$E_1(...)$ will also become 1 for $d_1 = d$ and 0 for $f_1 = f$. We can also examine the decision making for player 1 in $t \in N$:
$E_1(d_1, d_2, s_1, s_2, f_1, f_2, t)$

$$= \begin{cases} 1 & \text{(1) full mission success} \\ 1 - E_2(d_1 + 1, d_2, s_1, s_2, f_1, f_2, t-1) & \text{(2) deployment success, landing failure} \\ 1 - E_2(d_1, d_2, s_1 + 1, s_2, f_1, f_2, t-1) & \text{(3) launch success, deployment failure} \\ 1 - E_2(d_1, d_2, s_1, s_2, f_1 + 1, f_2, t-1) & \text{(4) launch failure} \\ 1 - E_2(d_1, d_2, s_1, s_2, f_1, f_2, t-1) & \text{(5) no launch attempt} \end{cases}$$

Similar to all other models, all of these cases happen with a certain probability, with the expected payouts sorted (1) $\geq$ (2) $\geq$ (3) $\geq$ (5) $\geq$ (4). The player sees the daily parameters (x_t, y_t, z_t) and decides between the 'fixed' expected points for not launching (5) and

$$\text{LV} := (1) \cdot (x_t y_t z_t) + (2) \cdot (x_t y_t (1 - z_t)) + (3) \cdot (x_t (1 - y_t)) + (4) \cdot (1 - x_t).$$

The payout for (2) will be 1 if $d_1 = d - 1$, because another successful launch and deployment results in a win. Payout for (4) can be 0 if $f_1 = f - 1$, with the fth launch failure leading to a disqualification/loss. Figuring out the distribution of LV would yield an even more complicated formula than in Section 5.1, therefore we will once again use a Monte-Carlo algorithm to determine an approximated value for LV in each situation. This will be compared to the „waiting value "$1 - E_2(..., t-1)$ and the expected value can then be calculated as in the other models:

$$E_1(..., t) = P(LV < WV) \cdot WV + (1 - P(LV < WV)) \cdot E(LV | LV \geq WV)$$

. Coming to the programming of the recursion, an unlimited and thus potentially infinite s_i is really not feasible. Therefore, one had to find a way to limit s_i without influencing the model too much. The program calculates towards any given maximum T, and for $t = T, s_i$ does not have to exceed $\{0, 1, ..., T+1\}$. In this case, all potential „Yuko advantages", (more successful launches), are depicted, in addition to a T+1-to-0 lead, which can not be caught up to in T periods.
Because the recursions count s_i upwards while t goes down, the needed interval for s_i becomes larger as t decreases. To be able to evaluate the desired interval in $t = T$, the interval in $t = 0$ has to be $s_i \in \{0, 1, ...2T + 1\}$. Even this restriction becomes problematic for larger t, as the programs runtime is increasing substantially. The volume of the matrix used for the recursions has the volume

$$V(E) = 2 \cdot 2 \cdot 2 \cdot (2T + 2) \cdot (2T + 2) \cdot 4 \cdot 4 \cdot (T + 1) = 512T^3 + 1536T^2 + 1536T + 512$$

which already exceeds 600,000 for $T = 10$. A recursion with a runtime of $O(t^3)$ is really not desirable. Thus, it was decided to add another rule to the model: s

successful launches resulting in a win, with $s > d$. If, for example, s is limited to 10, the volume of the matrix is reduced to

$$V(E) = 2 \cdot 2 \cdot 2 \cdot 10 \cdot 10 \cdot 4 \cdot 4 \cdot (T + 1) = 12800T + 12800$$

which is linear and therefore can be calculated faster. And even with this, the calculation on a standard PC with $mc = 10.000, s = 10$ takes about 35 minutes per t. With $mc = 1000$, the calculation up to $t = 100$ took a little under six hours. The programs *RocketJudo* and *JudoMC* used for examining the recursions can be found on the CD as well. Here are a few results:

Table 5.3.2: Judo model, E_1 with $d_1, d_2, f_1, f_2 = 0$ and $t = 100$

s_1 \ s_2	0	1	2	3	4	5	6	7	8	9
0	0.537	0.541	0.537	0.542	0.535	0.535	0.526	0.508	0.448	0.300
1	0.540	0.538	0.540	0.540	0.534	0.536	0.524	0.506	0.445	0.297
2	0.532	0.536	0.533	0.538	0.539	0.536	0.525	0.504	0.443	0.300
3	0.540	0.541	0.535	0.537	0.546	0.535	0.528	0.506	0.454	0.297
4	0.543	0.534	0.538	0.536	0.542	0.540	0.535	0.511	0.448	0.304
5	0.540	0.541	0.540	0.539	0.543	0.540	0.532	0.510	0.454	0.302
6	0.551	0.546	0.550	0.546	0.554	0.546	0.541	0.514	0.452	0.295
7	0.580	0.571	0.575	0.574	0.574	0.573	0.567	0.550	0.486	0.308
8	0.642	0.646	0.640	0.651	0.647	0.650	0.642	0.630	0.552	0.384
9	0.811	0.814	0.811	0.813	0.814	0.806	0.807	0.800	0.777	0.629

Table 5.3.3: Judo model, E_1 with $d_1, d_2, s_1, s_2 = 0$ and $t = 100$

f_1 \ f_2	0	1	2	3
0	0.537	0.563	0.596	0.686
1	0.514	0.53	0.569	0.672
2	0.471	0.485	0.525	0.625
3	0.360	0.373	0.412	0.507

Table 5.3.4: Judo model, E_1 with $d_1, d_2 = 0, s_1, s_2 = 9$ and $t = 100$

f_1 \ f_2	0	1	2	3
0	0.629	0.631	0.64	0.705
1	0.610	0.612	0.630	0.705
2	0.577	0.575	0.604	0.668
3	0.473	0.458	0.496	0.558

It's clear that $E_i(d_1, d_2, s_1, s_2, f_1, f_2, t)$ should be at least $\frac{1}{8}$, as this is the expected probability of a full mission success for any given x_t, y_t, z_t - this is comparable to winning three fair coin-flips in a row. In $t = 1$, there a many scenarios in which the active player is in the lead, and his expectation is 1 as he won't launch a rocket and lets time run out.

6 Two-Player Models with Non-Trivial Turn Orders

After examining SpaceX and Judo models, we now return to more "basic" models of rocket launching. So far, we have only looked at models where there is a strictly alternating turn order between multi players. For this section, we will go a step further and look at other possible turn orders to see if they have a notable effect on the players' chances.
There is also another way of tweaking the model besides altering the turn order: Models with non-deterministic time have been examined by Döring (2015), e.g. models where time does not count down in simple steps of 1, but randomly in larger or smaller steps (two day or one day with probability of 50% each, for example). This type of model might be considered for further research, but isn't examined in this thesis.

6.1 Random Order

We start by looking at a two-player model similar to the one described in Section 4. These two players are once again competing against each other, with a player winning the game by either launching his rocket successfully, or his opponent destroying his own rocket. The winner will receive 1 point, the loser 0 points. If the time, which counts down from a certain $T > 0$, runs out without a successful launch by either player, both of them lose and receive no points. The 'daily value' x_i is once again uniformly distributed on $(0, 1)$.

In Section 4, the turn order was strictly alternating. Now, before each new day, a coin-flip decides whether Player 1 or Player 2 get the launch opportunity for that given day. Let $p \in (0, 1)$ be the probability of Player 1 getting the launch opportunity for that day, therefore $1 - p$ will be Player 2 probability of receiving the launch opportunity.
With each player only having one rocket, both of them have only got one attempt at launching - if it fails, they lose. We define $E_d(t)$ as the expectation of the active player d with t days left. Similarly to Section 4, for $t = 1$ and $t = 2$ it is clear that

$$E_1(1) = E_2(1) = \frac{1}{2}, E_1(2) = E_2(2) = \frac{5}{8}$$

The value for $t = 2$ can be seen by looking at the decision tree for any of the two players: As we can see in Figure 6.1.1, the player will have a expectation of half a point on the last day, regardless of the parameter p because both players have to launch on the final day and have a 50-50 chance of success. Therefore, both players expectation on the penultimate day is $\frac{5}{8}$ as in Section 4.
For $t > 2$, we see the influence of the parameter p.
The decision parameters (see Definition 4.2) for the players are given by

$$s_1(t) = p \cdot E_1(t-1) + (1-p) \cdot (1 - E_2(t-1))$$

$$s_2(t) = p \cdot (1 - E_1(t-1)) + (1-p) \cdot E_2(t-1)$$

Out of these, we can now calculate the expected points

$$E_1(t) = s_1(t) \cdot s_1(t) + (1 - s_1(t)) \cdot \frac{1 + s_1(t)}{2} = \frac{1}{2} + \frac{s_1(t)^2}{2}$$

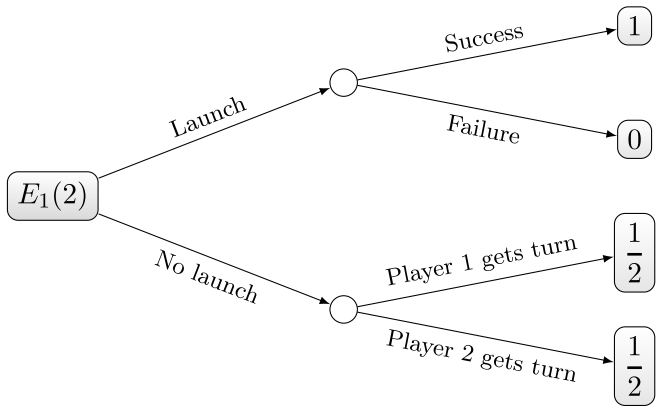


Figure 6.1.1: Decision tree, random-order, 1-vs-1, $t = 2$

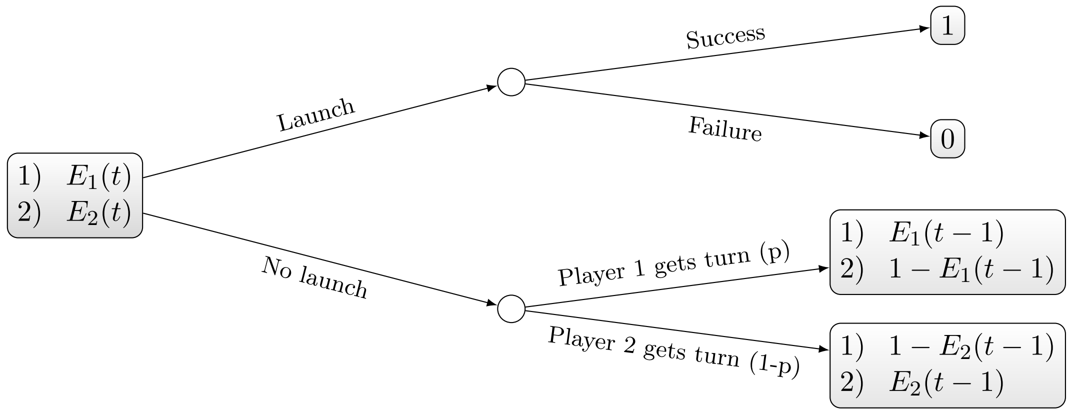


Figure 6.1.2: Decision tree for model with random launching right

$$E_2(t) = s_2(t) \cdot s_2(t) + (1 - s_2(t)) \cdot \frac{1 + s_2(t)}{2} = \frac{1}{2} + \frac{s_2(t)^2}{2}$$

One can already notice a certain symmetry, which can also be found in the results, as $E_1(t, p) = E_2(t, 1 - p)$ for large t. The recursion was programmed in Scilab (*RocketRandomOrder*) and tested for various p. Here are a few of the results found:

Table 6.1.1: Expectation for large t, random order, single rocket

p	E_1 (decimal)	E_2 (decimal)	E_1	E_2
0	0.5	1	1/2	1
1/10	0.53125	0.78125	17/32	25/32
1/5	0.55556	0.72222	10/18	13/18
1/4	0.56699	0.70096	$1 - \sqrt{3}/4$	$2 - \frac{3}{4}\sqrt{3}$
1/3	0.58786	0.67157	$2 - \sqrt{2}$	$7/2 - 2\sqrt{2}$
2/5	0.60102	0.65153	11/2 - $2\sqrt{6}$	$8 - 3\sqrt{6}$
1/2	0.625	0.625	5/8	5/8

Now, some of those values represented simple fractions. 0.58786 was identified as $2 - \sqrt{2}$ quickly as well. The other values were identified by using WolframAlpha.

These findings point to the assumption, that for rational $p = \dfrac{a}{a+b}, a, b \in \mathbb{N}$ the expected value of E_1 for large t has the form $x + y \cdot \sqrt{z}$, with all three variables depending on p. With a bit of trial and error, all three values were identified:

Theorem 6.1. *For the launch model with two players having one rocket each and random order given by a rational parameter* $p = \frac{a}{a+b} \in (0,1)$*, the expected value of player 1 for large* t *is given by*

$$E_1(t) = x - y \cdot \sqrt{z},$$

with

$$x = \frac{2a^2 + b^2 - ab}{2 \cdot (a-b)^2}, y = \frac{a}{(a-b)^2}, z = a \cdot b, \textit{with } a \neq b$$

Proof.
Similar to other proofs, we assume convergence and prove that the presented value is the correct one. Assuming convergence, $E_1(t)$ should asymptotically equal $E_1(t-1)$ as well. Therefore, we will just speak of E_1 (and E_2) here.

$$\begin{aligned}
E_1 &= \frac{1 + (\frac{a}{a+b}E_1 + \frac{b}{a+b}(1-E_2))^2}{2} \\
\Leftrightarrow E_1^* &= \frac{a^2 + ab + b^2 + abE_2 \pm \sqrt{b(a+b)^2(2aE_2 + b)}}{a^2} \\
E_2 &= \frac{1 + (\frac{a}{a+b}(1-E_1) + \frac{b}{a+b}E_2))^2}{2} \\
\Leftrightarrow E_2^* &= \frac{a^2 + ab + b^2 + abE_1 \pm \sqrt{a(a+b)^2(2bE_1 + a)}}{b^2}
\end{aligned}$$

The solutions with $-\sqrt{\ldots}$ are the important ones: Putting E_2^* into E_1^* and solving for E_1^*, we receive:

$$E_1 = \frac{\pm 2a^{3/2}\sqrt{b} + 2a^2 - ab + b^2}{2(a^2 - 2ab + b^2)} = \frac{2a^2 - ab + b^2}{2(a-b)^2} - \frac{a}{(a-b)^2}\sqrt{ab} \text{ for } a \neq b$$

.

This can analogously be done to receive E_2, which behaves symmetrical to E_1. □

Note: This can also be done for any $p \in (0,1)$, yielding $E_1(p) = \frac{1 - 3p + 4p^2 - 2\sqrt{p^3 - p^4}}{2 - 8p + 8p^2}$ for $p \in (0,1)$. Because the recursion was mostly tested for rational p, the focus of the above theorem lies on this. A plot of $E_1(p)$ and $E_2(p)$ can be found in the appendix to this chapter.

6.2 Snake Order

Definition 6.2. (Snake order)
A snake order for N players means that in the first round, player 1 has his turn first, then player 2, ..., until player N. In the next round, player N goes first, followed by player $N-1$, $N-2$,..., and finishing with player 1, after which the order is reversed again for the next round.

Example 6.3. (Snake order for 2 players)
Let's assume the game starts in $t = T$, counts down and player 1 has the first turn.

Time left	T	T-1	T-2	T-3	T-4	T-5	T-6	...
Player	1	2	2	1	1	2	2	...

By the example above, we see that for large T and two players, this essentially means that each player always has two subsequent days to launch his rocket. For this model, we will give player 1 n_1 and player 2 n_2 rockets, and force them to launch attempts by declaring them both losers if time runs out - similar to the Scenario 3 in the Rich-vs.-Poor model, laid out in Section 4.3. If one player runs out of rockets, the other player will also get all remaining days to try a successful launch. We define the following notation for expected value and decision parameters:

Definition 6.4. (Expected points/Decision parameters, Snake Order)
$E_d(n_1, n_2, p, t)$ denotes the expected points of (active) player d, with player 1 and 2 having n_1 and n_2 remaining rockets, player p being the next active player and t days remaining. $s_d(n_1, n_2, p, t)$ denotes the corresponding decision parameter.

First, for $t = 1$ we get $E_1(n_1, k, p, 1) = E_2(k, n_2, p, 1) = \frac{1}{2}$ for all $n_1, n_2 \geq 1, k \geq 0$. Due to the rules, it does not matter whether the passive player has any rockets left, and in $t = 1$, the next player is also irrelevant.

For $t = 2$, we need to examine two different situations:

Active player has launch right for $t = 1$:

$$E_1(n_1, k, 1, 2) = E_2(k, n_2, 2, 2) = \begin{cases} 1 - (\frac{1}{2})^2 = \frac{3}{4} & \text{for } n_i > 1 \\ \frac{5}{8} & \text{for } n_i = 1 \end{cases}$$

When a player has more than one rocket, he will launch two of them in the final two days. Otherwise, he receives the same expectation E_2 as in the single-player model outlined in Section 2.

Passive player has launch right for $t = 1$: $E_{1,2}(n_1, n_2, 2, 2) = \frac{1}{2}$ for $n_i > 1$
The active player has no incentive to wait, he launches a rocket with any x_2.

Now, we take a look at scenarios in which one player has no rocket left:

$$E_1(1, 0, 1, t) = \frac{1 + E_1(1, 0, 1, t-1)^2}{2}, E_2(0, 1, 2, t) = \frac{1 + E_2(0, 1, 2, t-1)^2}{2}$$

$$E_1(n_1, 0, 1, t) = \frac{E_n^2 \cdot E_{n-1} + E_n^2 - 2E_nE_{n-1} + E_{n-1} + 1}{2}, n_1 > 1, \text{ with}$$

$E_n = E_1(n_1, 0, 1, t-1), E_{n-1} = E_1(n_1 - 1, 0, 1, t-1)$

$$E_2(0, n_2, 2, t) = \frac{E_n^2 \cdot E_{n-1} + E_n^2 - 2E_nE_{n-1} + E_{n-1} + 1}{2}, n_2 > 1, \text{ with}$$
$E_n = E_2(0, n_2, 2, t-1), E_{n-1} = E_2(0, n_2 - 1, 2, t-1)$

This is an analogous result already discussed in Section 4.3. Next up, the situations in which one of the players has a single rocket left:

Active player with single rocket, active player has next turn:

$$\begin{aligned} E_1(1, n_2, 1, t) &= E_1(1, n_2, 2, t-1)^2 + (1 - E_1(1, n_2, 2, t-1)) \cdot \tfrac{1+E_1(1,n_2,2,t-1)}{2} \\ &= \frac{1 + E_1(1, n_2, 2, t-1)^2}{2}, \text{ analogous result for } E_2(n_1, 1, 2, t-1). \end{aligned}$$

Active player with single rocket, passive player has next turn:

$$\begin{aligned} E_1(1, n_2, 2, t) &= (1 - E_2(1, n_2, 2, t-1)^2 + (1 - (1 - E_2(1, n_2, 2, t-1)) \cdot \tfrac{1+(1-E_2(1,n_2,2,t-1))}{2} \\ &= \frac{1 + (1 - E_2(1, n_2, 2, t-1))^2}{2}, \text{ analogous result for } E_2(n_1, 1, 1, t-1). \end{aligned}$$

Passive player with single rocket, active player has next turn:

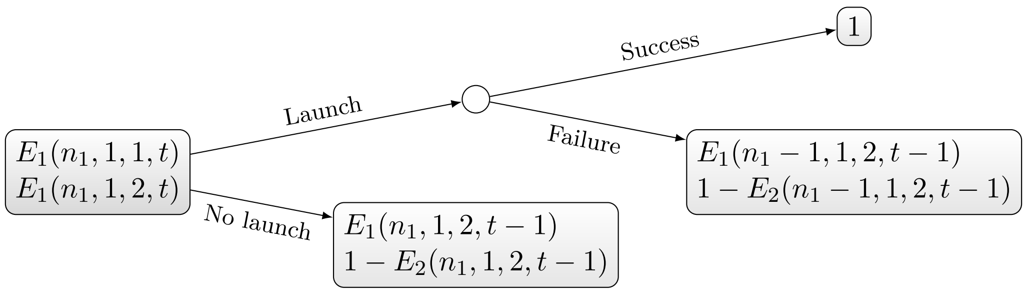


Figure 6.2.1: Decision tree, snake-order model, passive player with single rocket

Before this, all decision parameter were equal to the expected value of not launching. Now, we have to calculate them specifically:

$$s_1(n_1, 1, 1, t) \cdot 1 \cdot (1 - s_1(n_1, 1, 1, t)) \cdot E_1(n_1 - 1, 1, 2, t-1) = E_1(n_1, 1, 2, t-1)$$

$$\Leftrightarrow s_1(n_1, 1, 1, t) = \frac{E_1(n_1 - 1, 1, 2, t-1) - E_1(n_1, 1, 2, t-1)}{E_1(n_1 - 1, 1, 2, t-1) - 1} =: s_1$$
$$\begin{aligned} E_1(n_1, 1, 1, t) &= s_1 \cdot E_1(n_1, 1, 2, t-1) \\ &+ (1 - s_1) \cdot [\frac{1 + s_1}{2} \cdot 1 + (1 - \frac{1 + s_1}{2}) \cdot E_1(n_1 - 1, 1, 2, t-1)] \end{aligned}$$

Passive player with single rocket, passive player has next turn:

$$s_1(n_1, 1, 2, t) \cdot 1 \cdot (1 - s_1(n_1, 1, 1, t)) \cdot (1 - E_2(n_1 - 1, 1, 2, t-1) = 1 - E_2(n_1, 1, 2, t-1)$$

$$\Leftrightarrow s_1(n_1,1,2,t) = \frac{E_2(n_1-1,1,2,t-1) - E_2(n_1,1,2,t-1)}{E_2(n_1-1,1,2,t-1)} =: s_2$$

$$\begin{aligned} E_1(n_1,1,2,t) &= s_2 \cdot (1 - E_2(n_1,1,2,t-1) \\ &+ (1-s_2)\cdot [\frac{1+s_2}{2} + (1-\frac{1+s_2}{2})\cdot (1-E_2(n_1-1,1,2,t-1) \end{aligned}$$

The calculation for $E_2(1,n_2,1,t)$ and $E_2(1,n_2,2,t)$ is similar and left as an exercise. Finally, we look at the case in which both players have more than one rocket:

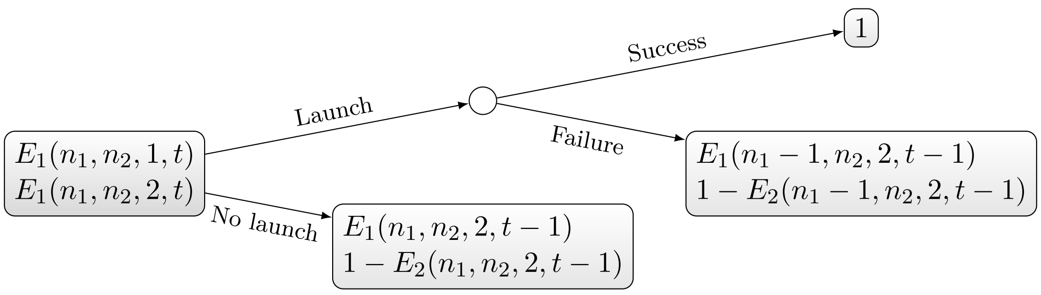


Figure 6.2.2: Decision tree, snake-order model, both players with > 1 rocket

$$s_1(n_1,n_2,1,t)\cdot 1 \cdot (1-s_1(n_1,n_2,1,t)) \cdot E_1(n_1-1,n_2,2,t-1) = E_1(n_1,n_2,2,t-1)$$

$$\Leftrightarrow s_1(n_1,n_2,1,t) = \frac{E_1(n_1-1,n_2,2,t-1) - E_1(n_1,n_2,2,t-1)}{E_1(n_1-1,n_2,2,t-1) - 1} =: s_3$$

$$\begin{aligned} E_1(n_1,n_2,1,t) &= s_3 \cdot E_1(n_1,n_2,2,t-1) \\ &+(1-s_3)\cdot \frac{1+s_3}{2}\cdot 1 + (1-\frac{1+s_3}{2})\cdot E_1(n_1-1,n_2,2,t-1) \end{aligned}$$

$$s_1(n_1,n_2,2,t)\cdot 1\cdot (1-s_1(n_1,n_2,1,t))\cdot (1-E_2(n_1-1,n_2,2,t-1) = 1-E_2(n_1,n_2,2,t-1)$$

$$\Leftrightarrow s_1(n_1,n_2,2,t) = \frac{E_2(n_1-1,n_2,2,t-1) - E_2(n_1,n_2,2,t-1)}{E_2(n_1-1,n_2,2,t-1)} =: s_4$$

$$\begin{aligned} E_1(n_1,n_2,2,t) &= s_4 \cdot (1 - E_2(n_1,n_2,2,t-1) \\ &+ (1-s_4)\cdot [\frac{1+s_4}{2} + (1-\frac{1+s_4}{2})\cdot (1-E_2(n_1-1,n_2,2,t-1) \end{aligned}$$

Now we have all relevant recursion formulas. These were coded into *RocketSnake*, and results were evaluated.

Theorem 6.5. (Results - Snake-Order Model)

1) $\lim_{t\to\infty} E_1(1,1,1,t) \approx 0.6562805$ 2) $\lim_{t\to\infty} E_1(1,1,2,t) \approx 0.5590715$

3) $\lim_{n_1=t\to\infty} E_1(n_1,1,1,t) \approx 0.8421738$ 4) $\lim_{n_1=t\to\infty} E_1(n_1,1,2,t) \approx 0.6843476$

5) $\lim_{n_2=t\to\infty} E_1(1,n_2,2,t) \approx 0.5124546$ 6) $\lim_{n_2=t\to\infty} E_1(1,n_2,1,t) \approx 0.6313048$

7) $\lim_{t\to\infty} E_1(n_1=n_2=t, p=1) = \frac{4}{5}$ 8) $\lim_{t\to\infty} E_1(n_1=n_2=t, p=2) = \frac{3}{5}$

Proof.
1) $\mathbf{E_1(1,1,1,t)} = \dfrac{1+E_1(1,1,2,t-1)^2}{2}, E_1(1,1,2,t-1) = \dfrac{1+(1-E_2(1,1,2,t-2))^2}{2},$

$E_2(1,1,2,t-2) = \dfrac{1+E_2(1,1,1,t-3)^2}{2}, E_2(1,1,1,t-3) = \dfrac{1+(1-\mathbf{E_1(1,1,1,t-4)})^2}{2}.$
Assuming convergence, we put all four terms into each other and receive this big equation of degree 16:

$\dfrac{x^{16}}{2^{15}} - \dfrac{x^{15}}{2^{11}} + \dfrac{x^{14}}{2^8} - \dfrac{21x^{13}}{2^{10}} + \dfrac{5x^{12}}{2^6} - \dfrac{29x^{11}}{2^7} + \dfrac{131x^{10}}{2^8} - \dfrac{29x^9}{32} + \dfrac{321x^8}{256} - \dfrac{43x^7}{32} + \dfrac{9x^6}{8} - \dfrac{13x^5}{16} + \dfrac{5x^4}{8} - \dfrac{x^3}{2} + \dfrac{x^2}{4} + \dfrac{5}{8} = x$ with $x = E_1(1,1,1,t)$.

This equation has 16 solutions, of which two are real, and one is approx. 0.6562805.

---

2) $E_1(1,1,1,t) = \dfrac{1+E_1(1,1,2,t)^2}{2}$, and with $E_1(1,1,1,t) = 0.6562805$ for large t, we receive $E_1(1,1,2,t) \approx 0.5590715$.

---

3) Player 1 has enough rockets for always launching, therefore his recursion formula is more simple:
$\mathbf{E_1(n_1,1,1,t)} = \dfrac{1+E_1(n_1-1,1,2,t-1)}{2},$
$E_1(n_1-1,1,2,t-1) = \dfrac{1+(1-E_2(n_1-2,1,2,t-2))}{2}$
$E_2(n_1-2,1,2,t-2) = \dfrac{1+E_2(n_1-2,1,1,t-3)^2}{2},$
$E_2(n_1-2,1,1,t-3) = \dfrac{1+(1-\mathbf{E_1(n_1-2,1,1,t-4)})^2}{2}.$

We end up with an equation of degree 4: $x^4 - 4x^3 + 8x^2 + 24x - 24 = 0$, which is solved by $x \approx 0.8421738$.

---

4) $E_1(n_1,1,1,t) = \dfrac{1+E_1(n_1-1,1,2,t-1)}{2}$, with $E_1(n_1,1,1,t) \approx 0.8421738$

we receive $\mathrm{E}_1(n_1,1,2,t) \approx 0.684348$ for large t.

---

5) $E_1(1,n_2,2,t) = \dfrac{1+(1-E_2(1,n_2,2,t-1))}{2} \approx \dfrac{1+(1-0.8421738)^2}{2} = 0.5124546$

---

6) $E_1(1,n_2,1,t) = \dfrac{1+E_1(1,n_2,2,t-1)^2}{2} \approx \dfrac{1+(0.5124546)^2}{2} = 0.6313048$

---

7) When both players have enough rockets to always launch, Player 1 goes first and he has the next turn as well, his chances of winning are:

$$\begin{aligned}
E &= \frac{1}{2} + \frac{1}{4} + \frac{1}{32} + \frac{1}{64} + \frac{1}{512} + \frac{1}{1024} + ... \\
&= \frac{3}{4} + \frac{3}{64} + \frac{3}{1024} + ... \qquad\qquad | \text{ divide by } 16 \\
\Leftrightarrow \frac{1}{16}E &= \frac{3}{64} + \frac{3}{1024} + ... \\
\Rightarrow \frac{15}{16}E &= \frac{3}{4} \Leftrightarrow E = \frac{4}{5}.
\end{aligned}$$

---

8) If player 1 only has the first turn, and then fourth/fifth,..., his chances are:

$$\begin{aligned}
E &= \frac{1}{2} + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{512} + \frac{1}{4096} + ... \\
&= \frac{9}{16} + \frac{9}{256} + \frac{9}{4096} + ... \qquad\qquad | \text{ divide by } 16 \\
\Leftrightarrow \frac{1}{16}E &= \frac{9}{256} + \frac{9}{4096} + ... \\
\Rightarrow \frac{15}{16}E &= \frac{9}{16} \Leftrightarrow E = \frac{3}{5}.
\end{aligned}$$

□

If we compare this to the results of the base scenario in Section 4.1/4.3, we see that „Rich“ has an even bigger advantage over Poor when he is active and also has the next turn, whereas when he is active with the next two launch opportunities going to Poor, his expectation is decreased:

$$\lim_{n_1=t\to\infty} E_1(n_1,1,1,t) \approx 0.8421738 > \sqrt{3}-1 > 0.6843476 \approx \lim_{n_1=t\to\infty} E_1(n_1,1,2,t)$$

The analogous result can be found for Poor as well:

$$\lim_{n_2=t\to\infty} E_1(1,n_2,2,t) \approx 0.5124546 < 4-2\sqrt{3} < 0.6313048 \approx \lim_{n_2=t\to\infty} E_1(1,n_2,1,t)$$

And for the situation in which both players always launch:

$$\lim_{t\to\infty} E_1(n_1=n_2=t, p=1) = \frac{4}{5} > \frac{2}{3} > \frac{3}{5} = \lim_{t\to\infty} E_1(n_1=n_2=t, p=2)$$

For a large number of players, their decision-making becomes more simple. As they are unlikely to have another attempt at launching, they will asymptotically launch as soon as it's their turn. For N players, the player first to act has a a winning chance of $\frac{1}{2}$, the second player has a chance of $\frac{1}{4}$ and the i-th player a chance of $\frac{1}{2^i}$. This was found in one of Prof. Althöfer's lectures for standard order, and applies here as well if the number of player is large enough.

7 Conclusions and Open Questions

We have now examined various different models, all more or less based on rocket launches. Let's summarize the results:

Section 2: In the simplest version of rocket launch models, which was also the one most comparable to the Classic Secretary Problem, we found that for rising t, E_t quickly converges to 1 for a single player.

Section 3: In the n-k-t model, we found that the recursion had numerical problems due to rounding inaccuracies and subsequent division by 0. We also gathered some results, namely that any player that doesn't have to launch all his rockets, needs significantly less time to have a large chance at succeeding in his mission. If he has to launch them all and has only little time buffer to do so, his chances are only slim.

Section 4: In this Rich-Vs-Poor-model, we used three variants to alter potential results of a base scenario. In this base scenario, we found that the rich player has a significant advantage when he is the active player (winning chance around 73%) whereas the poor player does not have much more than hope to get lucky (53% as the active player).
Of the three rule variants, only the lucky-loser rule with parameter p had a lasting influence on the limits for both Rich and Poor. Although we assumed that any $p > 0$ would make the game fairer, we saw that this wasn't the case, depending on how advantages were defined. Indeed, Poor had his minimum expectation of $\frac{1}{\sqrt{5}}$ for $p = \frac{8}{5} - \frac{2}{\sqrt{5}}$, only increasing it towards a fair game in $p = 1$ for larger p. The other two rule variants sadly didn't have notable effects.

Section 5: In the simple SpaceX model, we noted that the player has a very long way towards a high win rate if a, the reward for just launching, is small. For a nearing 1, the model converges with the one described in Section 2, where the convergence to an almost-surely success is much quicker. In the team model (Section 5.2), we found strong indications that simple majority voting is superior to unanimous voting, if only a simple majority of successful launches is needed.

Section 6: In the final two models, we altered the turn order to see if id had any effect on the expected outcome of the games. With random order, the first player's expectation increases pretty much linear up until $p = 0.8$ and quickly moves up to 1 for $p = 1$ afterwards (see Figure A.3.1 for reference), at least in a scenario where both players had a single rocket. Finally, in the snake-order model, we found that the player with more rockets had an even bigger advantage when having two turns in a row compared to the Rich-Vs-Poor model, but had a decreased expectation when having only one launch attempt before Poor.

Open questions and potential fields of future research:

Due to the potential variety of these models, this thesis only gives a small glimpse of what can be researched. In most models presented in this thesis, independent, uniformly distributed random numbers were used to describe the probability of successful launches. By opening up this restriction, e.g. using non-independent variables that are not necessarily distributed in $U(0,1)$, a whole new landscape of models becomes viable. These models might also reflect reality a lot closer - as for example weather on a particular day usually has a non-zero correlation with the weather on the day before. In two-player models, we could also give both players different distributions. Now to some more section-specific topics:

In Section 3: Finding a model that is close to the one presented in the section, but has no numerical problem due to a division by 0 in some cases. This might be achieved by somehow limiting the cases to „interesting“ ones, e.g. cases in which it is not instantly clear if the expected success rate is close to 0 or 1.

In Section 4: The consolation rule didn't really affect the long-run results of Rich and Poor. Here, a model in which the two players are working together could be an interesting focus for future research - in this, the consolation could be more of effect here, when both players see no realistic chance of launching and rather share the consolation c. Finding other rules other than the three documented in this thesis might also be worthy of consideration.

In Section 5: In all three models presented in this section, we used Monte-Carlo algorithms to determine approximate results. For quantitative examination, this was deemed enough - but for proof of potential results, exact results would be even more useful. In the simple SpaceX model, finding the density and probability function of LV can be done by splitting it into two independent factors. In the other two models, the calculation of exact conditional expectations seems a lot more complex. Finding other models based on scoring in certain sports might be a potentially inspiring task as well. A short description of a model based on basketball (and not necessarily on rocket launches) came to mind just before completion of this thesis and can be found in the appendix of Section 5.

In Section 6: It could be of potential interest to combine the Rich-vs.-Poor-model with random order, to examine possible effects on Rich's advantage over Poor. Depending how p is chosen, this could similar to snake order make the game fairer or increase Rich's advantage even further.

Overall, we have explored part of the rocket launch model landscape, and found both expected and surprising results. It seems like there could be even more to discover, as this was just a small part of possible games/models based on launching rockets. It is not quite clear if some of these results have an application to the real world yet, or if the models have to get more „realistic“ and complex to provide a larger amount of realism. This should be determined in future works.

A Appendix

A.1 Appendix to Chapter 4

A.1.1 Proof of Theorem 4.5

First we show that $E_2(n_1,1,t)$ is minimal in $p = \frac{8}{5} - \frac{2}{\sqrt{5}}$ and equals $\frac{1}{\sqrt{5}}$. After that, it is easy to show that $E_1(n_1,1,t)$ equals $\frac{3}{5}$ in this point.

The relevant recursion formula is found in Theorem 4.4:
$E_2(n_1,1,t) = s_2 \cdot (1 - E_1(n_1,1,t-1)) + (1-s_2)\cdot[(1-\frac{p}{2})\cdot\frac{1+s_2}{2}]$
and $s_2 = \dfrac{1 - E_1(n_1,1,t-1)}{1-p/2}$
Assuming convergence, we can equal
$E_1(n_1,1,t-1) = E_1(n_1-1,1,t-1) = E_1(n_1,1,t) = E_1$ for large n_1 and t and
$E_2(n_1,1,t-1) = E_2(n_1-1,1,t-1) = E_2(n_1,1,t) = E_2$ for large n_1 and t.

For reference, we also take a look at the decision tree for Rich again: See Figure 4.2.3 on page 27.
Rich will always launch, and therefore $E_1 = \frac{1}{2}\cdot(1-\frac{p}{2}+1-E_2) = 1 - \frac{E_2}{2} - \frac{p}{4}$.
For E_2, we use the formula of Theorem 4.4:

$$\begin{aligned}
E_2 &= \frac{1-E_1}{1-p/2}\cdot(1-E_1) + (1-\frac{1-E_1}{1-p/2})\cdot[(1-p/2)\cdot\frac{1}{2}\cdot(1+\frac{1-E_1}{1-p/2})]\\
&= \frac{(1-E_1)^2}{1-p/2} + \frac{2\cdot E_1 - p}{2-p}\cdot[(1-p/2)\cdot(\frac{E_1-1}{p-2}+\frac{1}{2})]\\
&= \frac{p^2-4p+4\cdot E_1^2 - 8\cdot E_1 + 8}{8-4p}
\end{aligned}$$

and with $E_1 = 1 - \frac{E_2}{2} - \frac{p}{4}$ this becomes

$$E_2 = -\frac{5p^2 + 4\cdot p\cdot E_2 - 16p + 4\cdot E_2^2 + 16}{32p-4}$$

This quadratic equation has two solutions:

$$E_2 = \frac{1}{2}(\pm\, 2\sqrt{5p^2-16p+12} - 5p + 8)$$

with $\frac{1}{2}(-2\sqrt{5p^2-16p+12} - 5p + 8) =: f(p)$ giving the limit of Poor's expected points for a given p.
The derivative equals $f'(p) = \frac{1}{2}\big(-\dfrac{2(5p-8)}{\sqrt{5p^2-16p+12}} - 5\big)$,
which equals 0 for $p = \frac{8}{5} - \frac{2}{\sqrt{5}}$.

Therefore, we have identified the p in which E_2 is minimal.

The value of E_2 is $f(\frac{8}{5} - \frac{2}{\sqrt{5}}) = \frac{1}{\sqrt{5}}$, as stated in the Theorem.

Finally, E_1 is given by $E_1 = 1 - \frac{E_2}{2} - \frac{p}{4} = 1 - \frac{2}{5} + \frac{1}{2\sqrt{5}} - \frac{2}{4\sqrt{5}} = \frac{3}{5}$.

A.1.2 $E_1(p), E_2(p), RA(p), AA(p)$ with relevant points

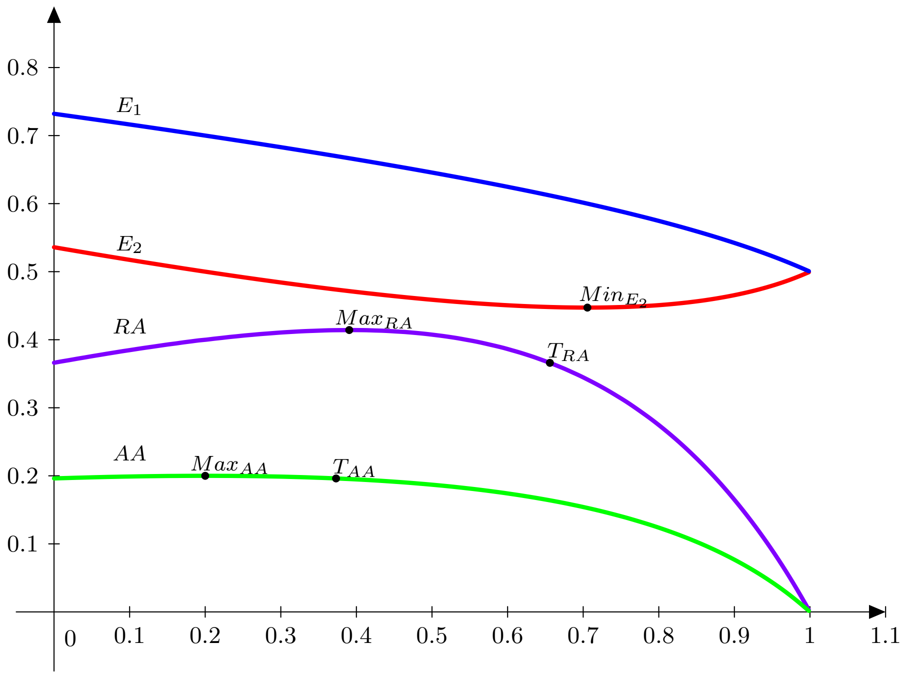


Figure A.1.1: Rich vs. Poor, Scenario 2, $E_{1,2}$ and advantages

A.2 Appendix to Chapter 5

Idea for a Basketball model:
- Two players
- Time ticking down from T to 0
- Players take turns in their throws
- Two random numbers x_t, y_t for each turn
- The larger value gives success rate of 2-point-throw
- The smaller value gives success rate of 3-point-throw
- Each player decides if he goes for two or three points in his turn
- Player with most points in $t = 0$ wins the game.
- $E_d(p_1, p_2, t)$ expected win rate of player d when players have p_1, p_2 points and t periods left.

A.3 Appendix to Chapter 6

$E_1(p), E_2(p)$ in the random-order model with one rocket each

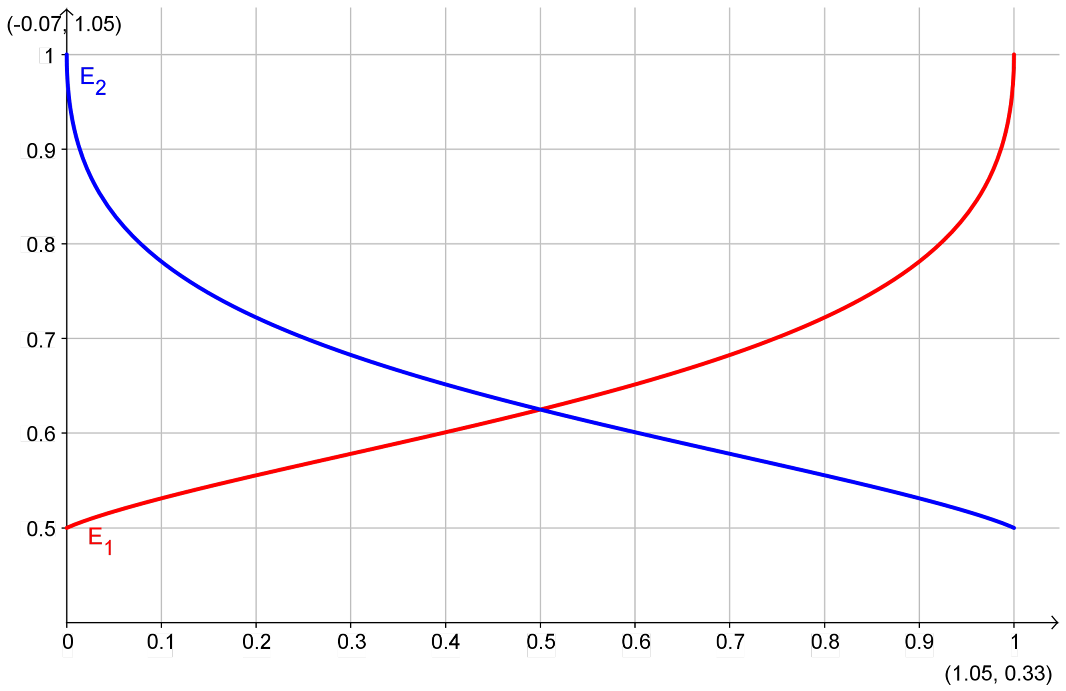


Figure A.3.1: Random-order model, $E_{1,2}$

A.4 CD with additional content

A CD with additional content is attached to the physical copy of this thesis. On this CD, one can find all the source codes for the programs used in this thesis, the two non pgf/tikz-pictures (Figures 5.2.1 and A.3.1, to be exact) , and additional results of both Rich vs. Poor and Judo model - and a PDF version of this paper.

References

Most of the models are based on Prof. Althöfer's and my own ideas. Here are the sources for the non-mathematical informations in this thesis:

- For computation of several equations: WolframAlpha, available at `http://wolframalpha.com`
- For SpaceX-related information: `http://spacex.com/`
- Judo rules: `http://www.intjudo.eu/upload/2015_04/20/142952199282442702/2015_ijf_refereeing_rules_english.pdf` This was apparently deleted some time after October 1st. Another summary can be found at `http://www.nbcolympics.com/news/judo-101-rules-scoring`.

Acknowledgments

First of all, I would like to thank Prof. Althöfer for the support during this thesis, delivering new ideas, models and variants throughout the writing of this paper, and giving helpful tips at several points.

Furthermore, I'd like to thank Chris Irrgang for his tips concerning the Java implementation of the n,k,t-model.

Another person I'd like to thank is André Prater for his help during the first few semesters of studying: Without his help, I probably wouldn't have come so far.
Several online sites have been very helpful throughout the years of studying, most notably WolframAlpha, StackExchange and MatheRaum. Thanks to the members of the latter two sites!

Finally, thanks to my mother who always encouraged me to move onwards during the five years of studying.

German Summary

Die vorliegende Arbeit wirft einen Blick auf eine neue Art von Sekretärinnenproblem bzw. Auswahlproblemen, basierend auf Raketenstarts. In Kapitel 2 wird zunächst das klassische Sekretärinnenproblem dargelegt und anschließend ein sehr ähnliches Ein-Spieler-Modell mit Raketenstarts präsentiert.

In Kapitel 3 findet sich ein weiteres Ein-Spieler-Modell, bei dem das Starten mehrerer Raketen in einem gegebenen Zeitfenster untersucht wird - hierbei kommt es allerdings zu numerischen Problemen. Anschließend befasst sich Kapitel 4 mit einem Zwei-Spieler-Modell, bei dem die beiden Spieler eine unterschiedliche Zahl von Raketen vorweisen, das sog. „Reich-Gegen-Arm-Modell“. Hier wird das Basisszenario außerdem durch drei Regeln variiert, und Effekte dieser Regeländerungen untersucht.

Abschnitt 5.1 ist inspiriert von den technischen Innovationen von SpaceX und bezieht neben Raketenstarts auch -landungen mittels eines zweiten Parameters ein. In Abschnitt 5.2 wird dann ein Team-Modell untersucht, bei dem drei Spieler eine gemeinsame Rakete starten wollen, um individuelle Missionen zu erfüllen, wobei sich herausstellt, dass eine einfach Mehrheit bei der Abstimmung bessere Resultate zu erzielen scheint als eine einstimmige. Anschließend wird noch ein Modell mit drei Parametern vorgestellt, dessen Idee auf dem Scoringverfahren im Kampfsport Judo basiert.

Im letzten Kapitel wird abschließend untersucht, welche Effekte eine Änderung der Zugreihenfolge weg von strikt abwechselnder Reihenfolge hat. In Abschnitt 6.1. wird die Zugreihenfolge randomisiert, und im Abschnitt 6.2 durch eine sogenannte „Snake-Order“ ersetzt.

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfen angefertigt habe.
Seitens des Verfassers bestehen keine Einwände die vorliegende Bachelorarbeit für die öffentliche Benutzung im Universitätsarchiv zur Verfügung zu stellen.

Jena, den 26. Oktober 2016

Christian Preßel