The Battles 2 Farming Problem

redlaserbm

January 31, 2024

1 Introduction

Bloons TD Battles 2 is competitive multiplayer tower defense video game in which each player, assigned one half of the screen, must pop balloons — referred to as bloons from here on out to maintain consistency with game lore — on their side of the screen before they reach the exit by placing monkeys (the towers) while sending bloons to their opponent in the hopes that they will be unable to pop those bloons. In order to build sufficient defense to pop incoming bloons and send enough outcoming bloons to overwhelm the opponent, players must take time in the early portion of the game to build up income that allows them to earn money to build defense and send more powerful bloons. For high-level Bloons TD Battles 2 players, a problem of utmost importance is that of determining, given multiple ways of generating income, the most optimal way to utilize all of those income sources. Although not always the case, a typical game of Battles 2 will involve players balancing *two* income sources:

1. **Eco** - This is income which is earned every 6 seconds. Players can send bloons to their opponent, spending some cash that they have right now in exchange for higher eco, which will allow them to earn most cash later. There are multiple types of bloons which the player can choose to send, and the choice of bloon influences the amount of eco gained or lost. Weaker bloons are unlikely to overwhelm the opponent but increase eco while stronger bloons are more powerful but may gain less eco or even cause the player to lose eco in some cases.

2. **Farms** - Banana Farms (typically) pay out a fixed amount of cash every round of the game. Since rounds can vary in length over the course of the game, farms are more effective during shorter rounds and less effective during longer rounds. The payout of a farm is (typically) spread out evenly over the course of a given round. For example, if a given round lasts 16 seconds, the uncrosspathed Banana Plantation, which pays out 40 dollars 16 times a round, would deliver the ith payment $i - 1$ seconds into the round. (More details on how farm payout times are computed later...)

Because round lengths influence the effectiveness of farms, and because players themselves can influence round lengths by popping bloons, players must decide in real-time whether to

focus on the time-based income source of eco or the round-based income source of farms. High-level players have arrived at strategies for handling this precarious between the two income sources, but generally have done so without the assigning their treatments formal mathematical rigor. This means that while top players have committed to strategies for earning money that are known to work well, we don't know whether they are truly optimal, and often times over the course of the game's lifespan, players have discovered better and better ways to earn money without there having been any prior balance changes to serve as a catalyst for such discovery.

The goal of this document is to solve the eco-farming problem in Battles 2, which asks roughly: What is the most optimal way to gain money given the ability to both gain eco and build farms? Restated with mathematical precision, the problem may be worded as follows: Suppose we are currently in a game in which t_0 seconds have elapsed, and that we have C cash, E eco, $F = \{f_{ijk}\}$ farms (where f_{ijk} denotes the amount of ijk farms currently on hand), and the round lengths are $\{r_i\}_{i=0}^{50}$ How do we utilize eco and farms to maximize the money gained in the time interval $[t_0, t)$?

2 Eco

It is common practice for farm players to mix farm play with yellow or even pink eco to maximize monetary gain. Hence, to begin our discussion, we will begin by performing an analysis of eco, and determine an explicit formula for the amount of money earned after an arbitrary amount of time spent eco'ing, assuming first that eco can be sustained non-stop and then removing this assumption in later subsections.

To keep computations tractable, we will make a simplifying assumption on the behavior of eco, treating it like a continuous stream of income generation. Given the eco send $S = (S_{cost}, S_{eco})$, if the player wishes to utilize this eco send, they will spend $S_{cost}/6$ dollars/second on the eco send and gain $S_{eco}/6$ eco/second while doing so.

2.1 Nonstop Eco

When the player has enough cash and eco on hand, they can constantly send bloons to their opponent. This scenario occurs more often than not, and we use the term *eco'ing* to describe the continuous sending of one type of bloon. Assume that you are non-stop yellow eco'ing and that you have enough cash and eco to sustain doing so without having to stop as a consequence of running out of money. A sufficient condition for this is having at least 1000 cash and 960 eco. This means that if your current eco is E, then after N complete eco ticks, you will have spent $1000N$ in sending yellows, while the amount of money awarded after the ith tick of doing so would be $E + 40i$. This means that the cash you gain after N ticks is given by

$$f(N) = -1000N + \sum_{i=1}^{N}(E + 40i) \tag{1}$$

$$= (E - 1000)N + 20N(N+1) \tag{2}$$

Now consider an arbitrary eco send $S = (S_{cost}, S_{eco})$ which, if sent non-stop, costs S_{cost} every 6 seconds to sustain and awards S_{eco} every eco tick. Then, we have

Proposition 2.1. *Assume the player currently has E eco. Then, the cash gain from N complete ticks of nonstop eco'ing with S is given by*

$$f(N) = (E - S_{cost})N + S_{eco}\frac{N(N+1)}{2} \tag{3}$$

2.2 Sustainable Eco

So far, we have assumed that the player can send eco non-stop, but this is not always possible if the player has too little cash or eco on hand.

Definition 2.2. *The eco send S is said to be **sustainable** given the tuple of cash and eco (C, E) if $C + f(N) \geq S_{cost}$ for all $n \in \mathbb{N}$, where f is given by equation 3.*

If hypothetically, it was possible to eco even with negative money, then formula 3 gives the money earned from N complete ticks of eco'ing. Thus, given some arrangement of (C, E) and time t^+ where t is the time of an eco tick, in order for S to be sustainable given (C, E), the value $f(N) + C$ for $N \in \mathbb{N}_0$ must never dip below S_{cost}. To check whether eco is sustainable, we just need to check whether $f(N^*) \geq S_{cost} - C$, where N^* minimizes $f(N)$. Since $f(N)$ is quadratic with a positive leading coefficient, N^* is unique and may be found as follows. The derivative of f (considered briefly in this context as a continuous function) is

$$f'(N) = (E - S_{cost}) + S_{eco}\Big(N + \frac{1}{2}\Big) \tag{4}$$

hence the minimum of f is achieved at

$$N^* = \max\Big(\mathcal{R}\Big(\frac{S_{cost} - E}{S_{eco}} - \frac{1}{2}\Big), 0\Big) \tag{5}$$

where $\mathcal{R}(x)$ is a function that rounds to the nearest integer. This permits us to make the following statements:

Definition 2.3. *A set $\Lambda \subset [0, \infty) \times [0, \infty)$ is a **sustainability region** for the eco send S if S is sustainable given (C, E) for all $(C, E) \in \Lambda$.*

Proposition 2.4. *The sustainability region of eco send S is the set*

$$\Big\{(C, E) \in [0, \infty) \times [0, \infty) : C \geq S_{cost} - (E - S_{cost})N^* - S_{eco}\frac{N^*(N^*+1)}{2}\Big\} \tag{6}$$

where

$$N^*(E) = \max\Big(\mathcal{R}\Big(\frac{S_{cost} - E}{S_{eco}} - \frac{1}{2}\Big), 0\Big) \tag{7}$$

To see this in action, we compute the sustainability region for some sets of eco in the game:

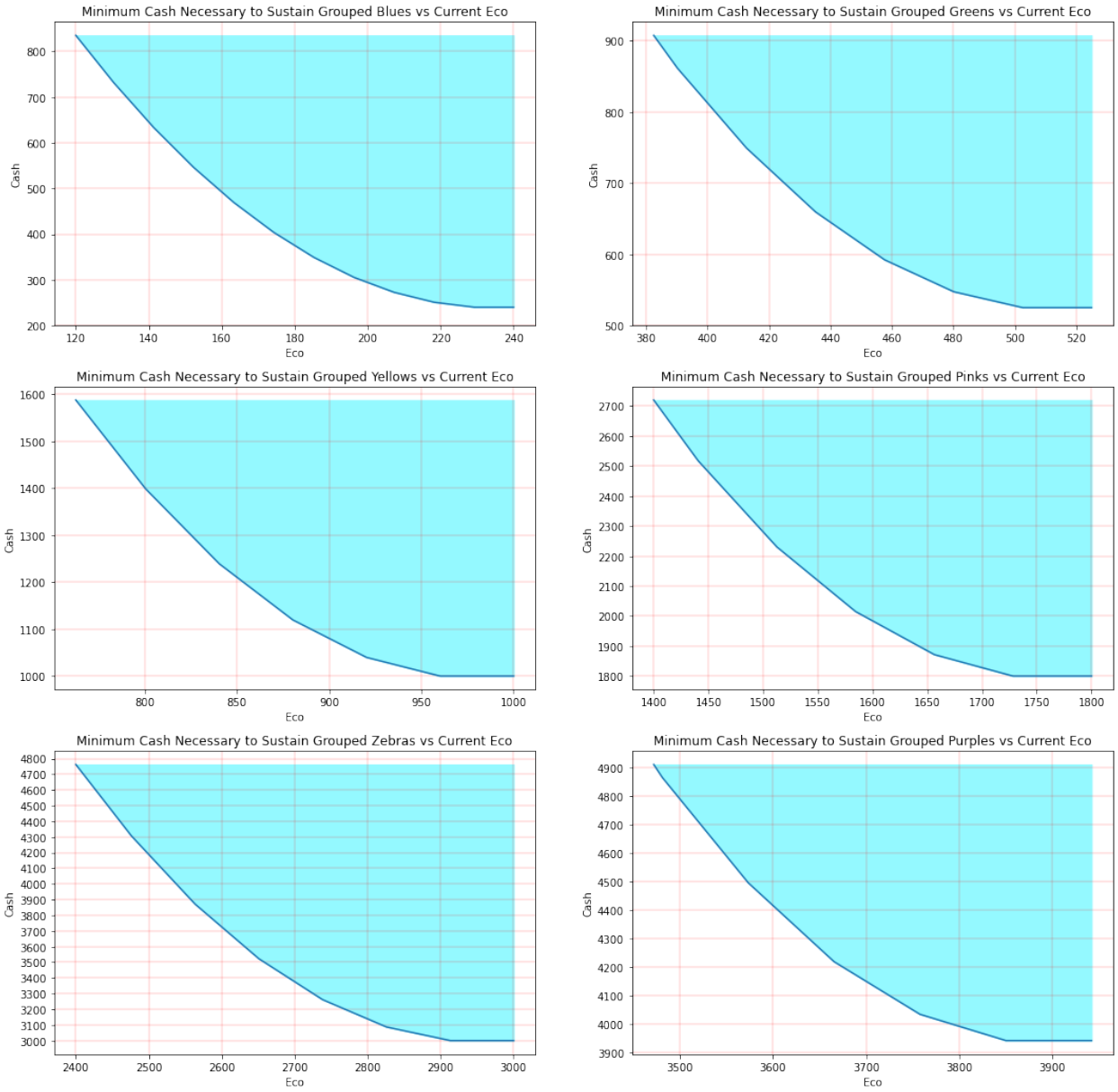

Figure 1: Sustainability region for some eco sends

2.3 Unsustainable Eco

In this section we compute a formula for the player's wealth after eco'ing with S in the time interval $[t_0, t)$ while now accounting for tuples of (C, E) that are unable to sustain S.

2.3.1 Derivation of a Recursive Formula

Assume to begin that $[t_0, t)$ is a large enough interval to encompass at least one eco tick. The first of these eco ticks then occurs at the time $\lceil t_0/6 \rceil$. Note here that $\lceil t_0/6 \rceil - t_0 < 6$, meaning the time that passes between t_0 and the first eco tick is not a full eco tick. If (C_0, E_0) is the player's initial cash and eco, then the player's cash and eco at time $\lceil t_0/6 \rceil$ (when the eco tick is paid out) is

$$C_1 = C_0 + E_0 + \min\Big(\lceil t_0/6 \rceil - t_0/6, \frac{C}{S_{cost}}\Big)(S_{eco} - S_{cost}) \tag{8}$$

$$E_1 = E_0 + \min\Big(\lceil t_0/6 \rceil - t_0/6, \frac{C}{S_{cost}}\Big)S_{eco} \tag{9}$$

If $i \geq 1$, then the cash and eco after i ticks have elapsed in the time interval is

$$C_{i+1} = C_i + E_i + \min\Big(1, \frac{C_i}{S_{cost}}\Big)(S_{eco} - S_{cost}) \tag{10}$$

$$E_{i+1} = E_i + \min(1, \frac{C_i}{S_{cost}})S_{eco} \tag{11}$$

In the time interval $[t_0, t)$, a total of $N(t_0, t) = \lceil t/6 \rceil - \lceil t_0/6 \rceil$ ticks will elapse, with the last tick occurring at time $t^* = 6(\lceil t/6 \rceil - 1)$. Now, in the time interval (t^*, t), the player will not collect any money from eco ticks but he will continue to send eco, which results in one last reduction in cash and increase in eco that must be measured. In this case:

$$C_{t^-} = C_{N(t,t_0)} - \min\Big(\frac{t - t^*}{6}, \frac{C_{N(t,t_0)}}{S_{cost}}\Big)S_{cost} \tag{12}$$

$$E_{t^-} = E_{N(t,t_0)} + \min\Big(\frac{t - t^*}{6}, \frac{C_{N(t,t_0)}}{S_{cost}}\Big)S_{eco} \tag{13}$$

Though we assumed implicitly that $t^* \geq t_0$, the formulas also work when this condition is not satisfied (See that $N(t, t_0) = 0$ and use the formulas 12) and thus we have

Lemma 2.5. *Assume the player at time t_0 has cash and eco (C_0, E_0) and attempts to eco S nonstop in the time interval $[t_0, t)$. Then, the player's cash and eco at time t^- is given by the formulas*

$$C_{t^-} = C_{N(t,t_0)} - \min\Big(\frac{t - t^*}{6}, \frac{C_{N(t,t_0)}}{S_{cost}}\Big)S_{cost} \tag{14}$$

$$E_{t^-} = E_{N(t,t_0)} + \min\Big(\frac{t - t^*}{6}, \frac{C_{N(t,t_0)}}{S_{cost}}\Big)S_{eco} \tag{15}$$

where

$$C_{i+1} = \begin{cases} C_i + E_i + \min\left(1, \frac{C_i}{S_{cost}}\right)(S_{eco} - S_{cost}) & 1 \leq i < N(t, t_0) \\ C_0 + E_0 + \min\left(\lceil t_0/6 \rceil - t_0/6, \frac{C_0}{S_{cost}}\right)(S_{eco} - S_{cost}) & i = 0 \end{cases} \tag{16}$$

$$E_{i+1} = \begin{cases} E_i + \min(1, \frac{C_i}{S_{cost}})S_{eco} & 1 \leq i < N(t, t_0) \\ E_0 + \min\left(\lceil t_0/6 \rceil - t_0/6, \frac{C}{S_{cost}}\right)S_{eco} & i = 0 \end{cases} \tag{17}$$

$$N(t_0, t) = \lceil t/6 \rceil - \lceil t_0/6 \rceil \tag{18}$$

2.3.2 Elimination of Unnecessary Recursion Steps

At this point we have achieved our goal of an eco formula for the arbitrary time interval $[t_0, t)$, but the results are somewhat unsatisfactory. For one, since the formulas are recursive, the computation time will increase in complexity with respect to time. This did not occur with the formula for sustainable eco, and while we will probably not be able to eliminate the recursion entirely, we can still discard some of it. One way to reduce the amount of recursion necessary is to revert to the formulas for sustainable eco once S becomes sustainable given (C, E). To start, let M be the smallest integer such that (C_M, E_M) is sustainable given S. That is, from the statement of Proposition 2.4, M is the smallest integer such that

$$C_M \geq S_{cost} - (E_M - S_{cost})N^* - S_{eco}\frac{N^*(N^* + 1)}{2} \tag{19}$$

$$N^* = \max\left(\mathcal{R}\Big(\frac{S_{cost} - E_M}{S_{eco}} - \frac{1}{2}\Big), 0\right) \tag{20}$$

Then, if $M \leq N(t, t_0)$ we can write

$$C_{N(t,t_0)} = C_M + (E_M - S_{cost})(N(t, t_0) - M) + S_{eco}\frac{(N(t, t_0) - M)(N(t, t_0) - M + 1)}{2} \tag{21}$$

$$E_{N(t,t_0)} = E_M + (N(t, t_0) - M)S_{eco} \tag{22}$$

and the set of formulas for the final cash and eco simplify slightly to

$$C_{t^-} = C_{N(t,t_0)} - \Big(\frac{t - t^*}{6}\Big)S_{cost} \tag{23}$$

$$E_{t^-} = E_{N(t,t_0)} + \Big(\frac{t - t^*}{6}\Big)S_{eco} \tag{24}$$

A second way to decrease the work necessary is to observe, even if S is not sustainble given (C, E), we may nonetheless be able to sustain S for sometime before we are unable to full eco. To this end, let $L \geq 1$ be the smallest integer such that $C_L < S_{cost}$. Assume that such an L is found. Then, we can compute

$$C_L = C_1 + (E_1 - S_{cost})(L - 1) + \frac{(L - 1)L}{2}S_{eco} \tag{25}$$

$$E_L = E_1 + (L - 1)S_{eco} \tag{26}$$

We leave it as an exercise to the reader to determine that

$$L = 1 + \lceil \frac{-(E_1 - S_{cost} - \frac{1}{2}S_{eco}) - \sqrt{(E_1 - S_{cost} - \frac{1}{2}S_{eco})^2 - 2S_{eco}(C_1 - S_{cost})}}{S_{eco}} \rceil \tag{27}$$

As a summary of the work above, the general strategy is to compute (C_1, E_1), then compute (C_L, E_L), compute (C_M, E_M) (using the recursive formulas), compute $(C_{N(t_0,t)}, E_{N(t_0,t)})$, and then finally compute (C_t, E_t). Other cases require the general approach. We are now ready to state the main theorem of this subsection:

Theorem 2.6. *Assume the player at time t_0 has cash and eco (C_0, E_0) and attempts to eco S nonstop in the time interval $[t_0, t)$. Let (C_i, E_i) denote the wealth immediately after the passing of the ith eco tick contained in $[t_0, t)$, let M be the smallest natural number such that S is sustainable given (C_M, E_M) and let L be the smallest natural number such that $C_L < S_{cost}$ Then, the player's cash and eco $\big(C(t_0, t, C_0, E_0, S), E(t_0, t, C_0, E_0, S)\big)$ at time t^- is given by the formulas*

$$C_{t^-} = C_{N(t,t_0)} - \min\Big(\frac{t - t^*}{6}, \frac{C_{N(t,t_0)}}{S_{cost}}\Big) S_{cost} \tag{28}$$

$$E_{t^-} = E_{N(t,t_0)} + \min\Big(\frac{t - t^*}{6}, \frac{C_{N(t,t_0)}}{S_{cost}}\Big) S_{eco} \tag{29}$$

where $t^* = \max(6\lceil t/6 \rceil - 6, t_0)$, $N(t_0, t) = \lceil t/6 \rceil - \lceil t_0/6 \rceil$ and

$$C_1 = C_0 + E_0 + \min\Big(\lceil t_0/6 \rceil - t_0/6, \frac{C_0}{S_{cost}}\Big)(S_{eco} - S_{cost}) \tag{30}$$

$$E_1 = E_0 + \min\Big(\lceil t_0/6 \rceil - t_0/6, \frac{C}{S_{cost}}\Big) S_{eco} \tag{31}$$

and also

$$C_L = C_1 + (E_1 - S_{cost})(L - 1) + \frac{(L-1)L}{2} S_{eco} \tag{32}$$

$$E_L = E_1 + (L - 1)S_{eco} \tag{33}$$

where L is given by the formula

$$L = 1 + \lceil \frac{-(E_1 - S_{cost} - \frac{1}{2}S_{eco}) - \sqrt{(E_1 - S_{cost} - \frac{1}{2}S_{eco})^2 - 2S_{eco}(C_1 - S_{cost})}}{S_{eco}} \rceil \tag{34}$$

and also

$$C_{i+1} = C_i + \frac{C_i}{S_{cost}} S_{eco} \tag{35}$$

$$E_{i+1} = E_i + \frac{C_i}{S_{cost}} S_{eco} \tag{36}$$

for $L \leq i < M$. Finally,

$$C_{N(t,t_0)} = C_M + (E_M - S_{cost})(N(t, t_0) - M) + S_{eco} \frac{(N(t, t_0) - M)(N(t, t_0) - M + 1)}{2} \tag{37}$$

$$E_{N(t,t_0)} = E_M + (N(t, t_0) - M) S_{eco} \tag{38}$$

2.4 The Efficient Eco Frontier

To gain eco faster, players rather than using grouped yellows can use grouped pinks, grouped zebras, or grouped purples. However, if the faster sends — faster in the sense that they permit eco to be gained faster — are not sustainable given the player's current cash and eco (C, E), then they will eventually run out of money. Intuitively, if the player wants to gain eco as fast as possible and cannot sustain an eco send for an entire tick, they should try to use that send for as long as possible but then switch to a cheaper send before they run out of money to sustain said cheaper send for the remainder of the eco tick duration.

In this section we detail and completely describe the only sending strategies that should be used to maximize monetary gain from eco (with pure eco strats).

Definition 2.7. *A **pure eco send** is an eco send S. A **mixed eco send** is an eco send $S = \sum_{i=1}^{N} a_i S_i$, where $\sum_{i=1}^{N} a_i = 1$, $0 \leq a_i \leq 1$ for all i and the S_i's are pure eco sends. In this case, we say that S is a **mixture** of the S_i's.*

Definition 2.8. *An eco send S_j is said to be **dominated** by the eco sends S_i and S_k if there exists a mixture S of S_i and S_k such that $S^{cost} = S_j^{cost}$ and $S^{eco} > S_j^{cost}$. In this case, we write $\{S_i, S_k\} > S_j$. S_i is dominated by $\{S_j\}_{j \in I}$ if S_i is dominated by S_{j_1} and S_{j_2} for all $j_1, j_2 \in J$ such that $j_1 \neq j_2$. This is written as $\{S_j\}_{j \in I} > S_i$.*

In lay mans' terms, S_i is dominated if we can use a combination of other eco sends to generate more eco than S_i^{eco} at the same cost as S_i^{cost}. An example of a mixed eco send is $\frac{1}{2}(S_{yellow} + S_{pinks})$, which is equivalent to sending grouped yellows for 3 seconds and grouped pinks for 3 seconds.

Proposition 2.9. *The eco send S_j is dominated by the eco sends S_i and S_k if and only if the inequality*

$$aS_i^{eco} + (1-a)S_k^{eco} > S_j^{eco} \tag{39}$$

$$a = \frac{S_j^{cost} - S_k^{cost}}{S_i^{cost} - S_k^{cost}} \tag{40}$$

is satisfied and $0 \leq a \leq 1$.

Proof. Verify that

$$aS_i^{cost} + (1-a)S_k^{cost} = S_j^{cost} \tag{41}$$

□

If our goal is simply to build eco as fast as possible, then theoretically we should never use dominated sends for this purpose. Note that as the game proceeds, certain eco sends will become available and unavailable, hence a send that is dominated before may become nondominated and vice versa. To determine the appropriate send to use at a given time, we introduce the notion of the efficient eco frontier:

Definition 2.10. *The **efficient eco frontier** corresponding to a set of eco sends $\{S_i\}_{i\in I}$ is the collection of all eco sends S (pure and mixed) such that S is not dominated.*

Lemma 2.11. *Suppose that $\{S_i\}_{i=0}^N$ is a collection of eco sends such that $S_{i+1}^{cost} > S_i^{cost}$ for all i and $S_0 = (0,0)$ is the zero eco send. Then, if none of the S_i's are dominated sends, then all sends on the efficient eco frontier take the form:*

$$(\alpha - \lfloor\alpha\rfloor)S_{\lfloor\alpha\rfloor+1} + \big(1 - (\alpha - \lfloor\alpha\rfloor)\big)S_{\lfloor\alpha\rfloor} \tag{42}$$

where $\alpha \in [0, N]$

Thus, the best send at any given time will always be a mixture of two nondominated pure sends. Below, we compute the the efficient eco frontier for all the important rounds of the game.

2.5 Exercises

Exercise 2.12. *Prove that the minimum of f as given in proposition 2.1 is indeed given by equation 5.*

Exercise 2.13. *List the times of all eco ticks that occur in the following intervals. If no eco ticks occur, then say so:*

1. $[0, 6]$
2. $[0, 6)$
3. $(6, 12)$
4. $(0, 12]$
5. $[200, 250)$

Exercise 2.14. *Prove the formula for L in formula 34. Hint: Consider the inequality*

$$f(N) - S_{cost} \leq 0 \tag{43}$$

Solve this inequality by first determining the zeros of $f(N) - S_{cost}$. Only of those zeros is positive and hence could possibly be the candidate value for L. But then you still need to explain where the $\lceil\cdot\rceil$ and $+1$ terms come from...

Exercise 2.15. *Suppose that the statement of Theorem 2.6 is modified to assert that*

$$E_{L+1} = E_L + \frac{C_L}{S_{cost}} S_{eco} \tag{44}$$

$$E_{i+1} = E_i + \frac{E_i}{S_{cost}} S_{eco} \tag{45}$$

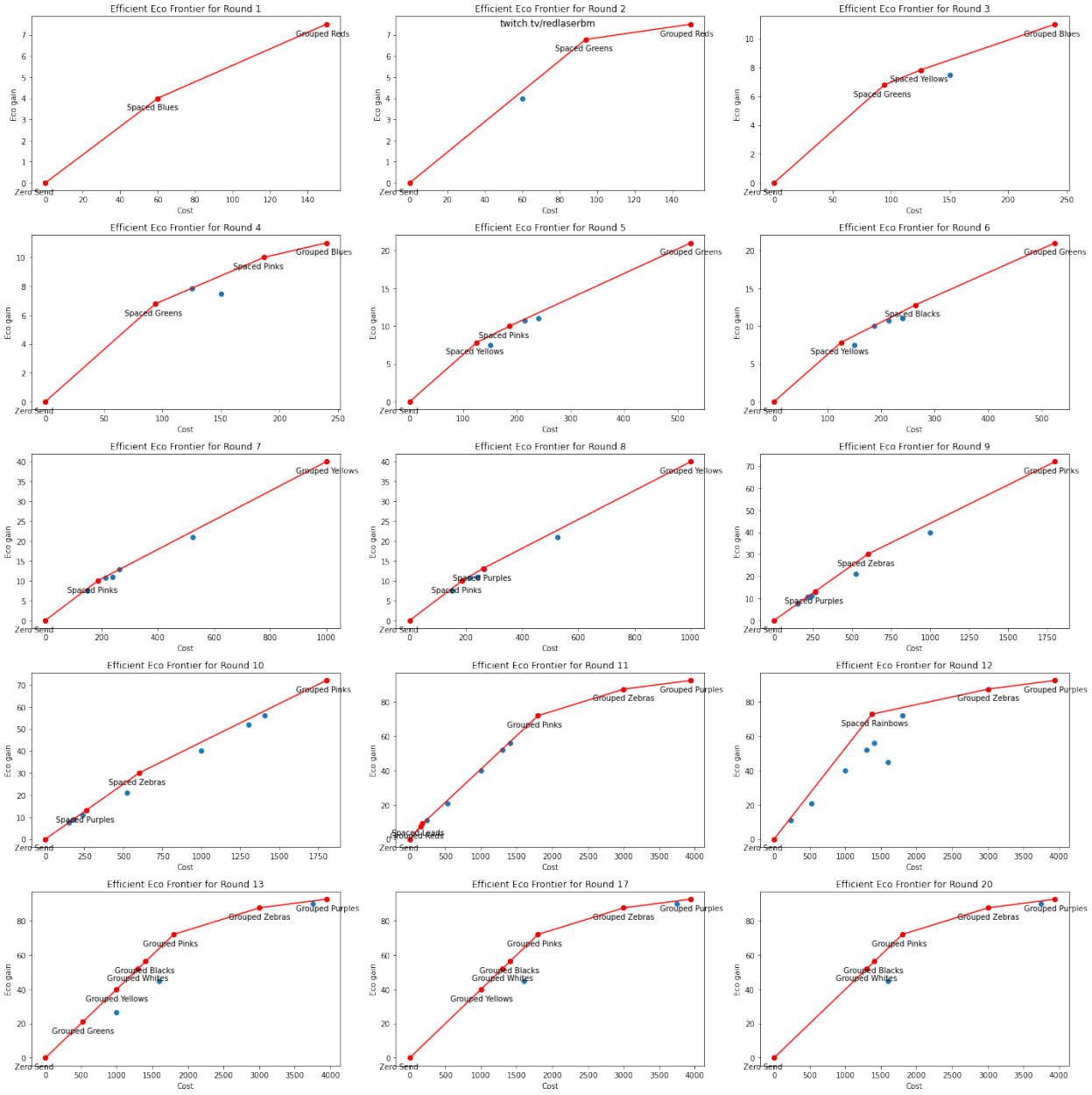

Figure 2: Efficient Eco Frontier for the most recent update

for $L+1 \leq i < M$. and also

$$C_{N(t,t_0)} = E_M + (E_M - S_{cost})(N(t,t_0) - M) + S_{eco}\frac{(N(t,t_0) - M)(N(t,t_0) - M + 1)}{2} \quad (46)$$

$$E_{N(t,t_0)} = E_M + (N(t,t_0) - M)S_{eco} \quad (47)$$

Most notably, $C_{L+1}, \ldots, C_M$ are not determined by the code, and the computation of $C_{N(t,t_0)}$ uses E_M instead of C_M. Answer the following questions

1. *Explain why the above modification to the theorem is does not yield errors.*
2. *Although the computation of $C_{L+1}, \ldots, C_M$ can be skipped, the computation of C_L cannot. Explain why this is so.*

Exercise 2.16. *Suppose that computer code is written using Theorem 2.6 to determine the player's cash and eco at time t^-. Assume that $N(t_0, t) = 10$, $L = 3$, and $M = 7$. Additionally, assume that the variables C and E in the code track the player's cash and eco respectively (and are hence overwritten upon each update). How many updates will be made to E in the code before it returns the player's final cash and eco?*

3 Banana Farms

We now turn our attention to the farm tower. In this section, we derive an explicit formula for the money paid out by a farm during an arbitrary time interval.

3.1 Farm Payout Over One Round

To begin, we derive a formula for the payout of a farm from t_0 seconds into the round to t seconds into the round (including t_0 but not t). Let s denote the amount of time it takes for all natural bloons to spawn in the round and let r denote the length of the round. A round in Battles 2 ends 4 seconds after one player pops all of the natural bloons for that round on their side. Hence, $r \geq s + 4$ and the round length depends on the intensity of stalling or anti-stalling that may occur during the game.

The timing of a farm's payouts during a round depends on when the farm upgrade was bought. First consider the case where the farm has been prebrought prior to the start of the round. Assume that the farm will pay out k times per round. Then, the ith payment of the farm will occur at a time

$$\frac{i-1}{k}s \tag{48}$$

seconds into the round. Observe that as a consequence, if a farm is bought/upgraded in the time interval $\left[\frac{k-1}{k}s_n, r_n\right)$, the upgraded farm will not pay out anything in this time interval. Thus, we have

Proposition 3.1. *The payout of a pre-brought farm that pays out p dollars k times a round in the time interval $[0, t)$ seconds of a round is*

$$p\lceil k \min(t/s, 1)\rceil \tag{49}$$

It may seem unusual to the reader that if $t = 0$, then the above formula equals 0 even though there is a farm payment that occurs right at the start of the round. The justification for this is that $[0, 0)$ is the empty set and in order for any payment to be received, some amount of time has to pass first! Now, suppose we buy a farm upgrade $t_p \geq 0$ seconds into the round and that the farm normally gives k payments of p dollars each over the course of a round. Then, the farm will pay out every s/k seconds, starting at the time $t_p + s/k$ and continuing until natural bloons finish spawning. Under the assumption that a farm cannot pay out exactly at the time s, this means the payout of the farm in the time interval $[t_p, t)$ is given by

$$p\big(\lceil k\frac{\min(t, s) - t_p}{s}\rceil - 1_{\{t>t_p\}}\big) \tag{50}$$

A convention we have implicitly adopted as a consequence of the above formula is that a farm upgrade must be bought strictly before the start of the round to receive the start-of-round

payout, and that buying exactly at the start of the round will not give the payout. This assumption is consistent with observations that heroes and farms must be placed explicitly before the start of the next round to receive start-of-round benefits. The reason for the $1_{\{t>t_p\}}$ term in the above expression is because, if we deleted that term, then the formula would imply that we always receive a payment immediately after purchase, which is not true. Naturally then, we have

Lemma 3.2. *Consider a farm with payout (p,k) bought t_p seconds into a round. If $0 \le t_p \le t_0 \le t < r$, the payout of the farm from $[t_0,t)$ is given by*

$$p\big(\lceil k\frac{\min(t,s)-t_p}{s}\rceil - \lceil k\frac{\min(t_0,s)-t_p}{s}\rceil - (1_{\{t>t_p\}} - 1_{\{t_0>t_p\}})\big) \tag{51}$$

Proof. (Hint) Write the formula for the payout from $[t_p,t)$ and subtract from this value the payout from $[t_p,t_0)$. □

3.2 Farm Payout Over Multiple Rounds

We now work to derive a formula which gives the payout of a farm in the time interval $[t_0,t)$, where now, unlike in the previous section, t_0 and t measure the amount of game time passed instead of the amount of round time passed. In general, the payout will depend on (in addition to the time interval we are measuring) the time t_P that the farm was bought/last upgraded, the number of payouts k the farm gives, and the payout amounts p. Expanding on the notation of the last section, we let $\{s_i\}_{i=0}^{50}$ and $\{r_i\}_{i=0}^{50}$ be sequences whose ith element denotes the amount of time it takes for the natural bloons to spawn in round i and the amount of time round i takes respectively. Now, define the functions

$$R(t) = \min\Big\{N : t - \sum_{i=0}^{N} r_i < 0\Big\} \tag{52}$$

$$T(t) = t - \sum_{i=0}^{R(t)-1} r_i \tag{53}$$

$R(t)$ is a function that outputs the current round of the game after t seconds. $T(t)$ is a function that outputs the amount of time elapsed in that round. Note that as a consequence of defining $R(t)$ in the way that we have, we have adopted the convention of round n spanning the time interval $[\sum_{i=0}^{n-1} r_i, \sum_{i=0}^{n} r_i)$ and thus if $t = \sum_{i=0}^{N} r_i$, we are on the start of round $N+1$.

To begin the discussion, assume that the farm was available to use since before the start of the game. That is, $t_0 = 0$ and $t_P = 0^-$. Then, $R(t)$ complete rounds have elapsed — here I'm counting "round 0", the time you have to place towers before the game starts, as a round — and the payout from those rounds is $pkR(t)$. On the other hand, during $R(t)$ itself, $T(t)$ seconds have elapsed in that round and in the time interval $[\sum_{i=0}^{R(t)-1} r_i, T(t))$ the farm has paid out

$$p\lceil k \min\Big(\frac{T(t)}{s_{R(t)}}, 1\Big)\rceil \tag{54}$$

dollars. Thus, the payout of the farm from $[0,t)$ is given by

$$pkR(t) + p\lceil k \min\Big(\frac{T(t)}{s_{R(t)}}, 1\Big)\rceil \tag{55}$$

To get the payout for $[t_0,t)$ we subtract the payout for $[0,t_0)$ from the payout for $[0,t)$ and get

Lemma 3.3. *A farm available from the start of the game that pays out k times a round with pay outs of p dollars will make*

$$f(t_0,t,p,k) = pk(R(t)-R(t_0)) + p\lceil k \min\Big(\frac{T(t)}{s_{R(t)}}, 1\Big)\rceil - p\lceil k \min\Big(\frac{T(t_0)}{s_{R(t_0)}}, 1\Big)\rceil \tag{56}$$

dollars in the time interval $[t_0,t)$.

The above formula remains correct so long as t_P is not in the same round of t_0. If it is, then we must adjust the formula. If $R(t) = R(t_0) = R(t_P)$ then the formula

$$p\big(\lceil k\frac{\min(T(t),s)-T(t_P)}{s}\rceil - \lceil k\frac{\min(T(t_0),s)-T(t_P)}{s}\rceil\big) - p(1_{\{t>t_P\}} - 1_{\{t_0>t_P\}}) \tag{57}$$

works — see formula 51. If $R(t) > R(t_0)$, then the payout of the farm on $\big[t_0, \sum_{i=1}^{R(t_0)} r_i\big)$ is

$$p\big(\lceil k\frac{s-T(t_P)}{s}\rceil - \lceil k\frac{\min(T(t_0),s)-T(t_P)}{s}\rceil - 1 + 1_{\{t_0>t_P\}}\big) \tag{58}$$

and the payout of the farm on the (possibly empty) interval $\big[\sum_{i=1}^{R(t)-1} r_i, t\big)$ is

$$p\lceil k \min\Big(\frac{T(t)}{s_{R(t)}}, 1\Big)\rceil \tag{59}$$

If there are any rounds in between, the payouts of those rounds are

$$(R(t) - R(t_0) - 1)pk \tag{60}$$

We are now ready to state the main theorem of this section:

Theorem 3.4. *The payout of a farm purchased at time t_P that pays out k times a round with pay outs of p dollars in the time interval $[t_0,t)$ is*

$$f(t_0,t,t_P,p,k) = p\big(\lceil k\frac{\min(t-\sum_{i=0}^{R(t_0)-1} r_i, s_{R(t_P)})-T(t_P)}{s_{R(t_P)}}\rceil - \lceil k\frac{\min(T(t_0),s_{R(t_P)})-T(t_P)}{s_{R(t_P)}}\rceil\big) \tag{61}$$

$$-p(1_{\{t>t_P\}} - 1_{\{t_0>t_P\}}) + p\lceil k \min\Big(\frac{T(t)}{s_{R(t)}}, 1\Big)\rceil 1_{\{R_t>R_0\}} + \max(R(t)-R(t_0)-1,0)pk \tag{62}$$

if $R(t_0) = R(t_P)$ and

$$f(t_0,t,t_P,p,k) = pk(R(t)-R(t_0)) + p\lceil k \min\Big(\frac{T(t)}{s_{R(t)}}, 1\Big)\rceil - p\lceil k \min\Big(\frac{T(t_0)}{s_{R(t_0)}}, 1\Big)\rceil \tag{63}$$

otherwise.

Corollary 3.5. *A farm that pays out k times a round with pay outs of p dollars will make $pk(N-M)$ dollars from the start of round M to the end of round $N-1$ assuming that it is purchased before the start of round M.*

We finish off the section with a updated version of the definition of sustainability which accounts for farms that the player might have on the playing field:

Definition 3.6. *The eco send S is **sustainable** given $(C, E, \{f_{ijk}\}, t_0, \{r_i\}_{i=0}^{50})$ if*

$$C + G(t_0, t, E, S, \{F_i\}_{i=1}^{N}, \{r_i\}_{i=0}^{50}) \geq 0 \tag{64}$$

for all $t > t_0$.

4 Monetary Gain From Arbitrary Eco-Farm Strats

In this section, our goal is to produce an algorithm from which the wealth over time of a player who adopts an arbitrary flowchart of eco'ing and farming can be computed.

Definition 4.1. *A **farm** f is a a tuple (p, k, t_P). A **farm build** is a collection of farms $F = \{f_i : i \in I\}$. A **farm flowchart** is a finite sequence of farm builds $\{F_i\}_{i=1}^{N}$.*

Let

$$\phi = \Phi(t_0, t_{eco}, C_0, E_0, S, \{F_i\}_{i=1}^{N}, \{r_i\}_{i=0}^{50}) \tag{65}$$

denote the time we obtain the farm build F_N starting from time t_0 assuming that we eco using the eco send S in the time period $[t_0, t_{eco})$ and assuming that at time t_0 we have (C_0, E_0) cash and eco with the farm build F_1. Our goal is to optimize Φ over the variable t_{eco} fixing the other variables $t_0, C, E, \{F_i\}_{i=1}^{8}, \{r_i\}_{i=0}^{50}$ as constants. Before we can optimize Φ, however, we must determine how to compute it. To compute Φ, let t_i denote the time we switch from F_i to F_{i+1}. That is,

$$t_i = \Phi(t_{i-1}, t_{eco}, C_{i-1}, E_{i-1}, S, \{F_k\}_{k=i}^{i+1}, \{r_k\}_{k=0}^{50}) \tag{66}$$

Then, we can compute $\phi = t_{N-1}$ by computing the values $t_1, t_2, \ldots, t_{N-2}, t_{N-1}$. The computation of t_i is split into two cases. First, assume that $t_i < t_{eco}$. To compute t_i, we compute

$$t_i^* = \inf\{t : W(t_{i-1}, t, C_{i-1}, E_{i-1}, S, F_i) \geq M(F_i, F_{i+1}) + S_{cost}\} \tag{67}$$

where W represents the player's wealth at time t^- and $M(F_i, F_{i+1})$ denotes the net cost of switching from farm build F_i to farm build F_{i+1}. The value t_i^* represents the time we can afford the farm assuming we constantly eco throughout $[t_0, t^*)$. If $t_i^* < t_{eco}$, then put $t_i = t_i^*$. Otherwise, let

$$C^* = W(t_{i-1}, t_{eco}, C_{i-1}, E_{i-1}, S, F_i) \tag{68}$$

$$E^* = E(t_{i-1}, t_{eco}, C_{i-1}, E_{i-1}, S) \tag{69}$$

where E can be computed by the formulas in Theorem 2.6. Then, put t_i equal to

$$t_i = \inf\{t : W(t_{eco}, t, C^*, E^*, 0, F_i) \geq M(F_i, F_{i+1})\} \tag{70}$$

Thus, the computation of t_i uses either formula 67 or 70 depending on how soon the next set of farm upgrades can be afforded. Simply knowing t_i is not sufficient to compute t_{i+1}, however. In order to compute t_{i+1}, we must also know the amount of cash and eco held at t_i. The updates to C and E are given by

$$C_i = \begin{cases} W(t_{i-1}, t_i, C_{i-1}, E_{i-1}, S, F_i) - M(F_i, F_{i+1}) & t_i < t_{eco} \\ W(t_{i-1}, t_{eco}, C_{i-1}, E_{i-1}, S, F_i) + W(t_{eco}, t_i, C^*, E^*, 0, F_i) - M(F_i, F_{i+1}) & t_1 \geq t_{eco} \end{cases} \tag{71}$$

$$E_i = E(t_{i-1}, \min(t_i, t_{eco}), C_{i-1}, E_{i-1}, S) \tag{72}$$

The equations 67, 70 and 71 form recursive equations that can be used to compute t_i if $t_{i-1} < t_{eco}$. Let M be the smallest integer so that $t_M \geq t_{eco}$. Then, t_i for $i > M$ can be determined by the recursive equations

$$t_i = \inf\{t : W(t_{i-1}, t, C_{i-1}, E_M, 0, F_i) \geq M(F_i, F_{i+1})\} \tag{73}$$

$$C_i = W(t_{i-1}, t_i, C_{i-1}, E_M, 0, F_i) - M(F_i, F_{i+1}) \tag{74}$$

The recursive equations are easier because we no longer to need track eco nor do we need to check whether or not the player has stopped eco'ing.

4.1 Nonlinear Optimization

Optimization over Φ stated as is may be difficult because it is not clear whether Φ has any nice exploitable properties or not. If the function was a continuous convex function, convex optimization techniques could be applied to find the minimum. We do that, *roughly* speaking, more time spent eco'ing should decrease the time spent to obtain the farm up to certain point, after which more time spent eco'ing will delay the time we get F_N.

5 Open Problems

The following are current open problems not answered in this document:

1. Given an arbitrary amount of cash and eco (C, E), how do I eco in such a way to maximize the eco I gain over the next N eco ticks of play?
2. How much money is gained by the engineer trap in the time interval $[t_0, t)$, assuming the trap was purchased at time $t_P \leq t_0$ and is being fed bloons from the eco send S?
3. What is the optimal time to stop eco'ing yellows when farming according to the Monkey Wall Street flowchart?