Cookie Clicker

Erik Demaine, Hiro Ito, Stefan Langerman,
Jason Lynch, Mikhail Rudoy, Kai Xiao
kaix@mit.edu

December 12, 2016

Abstract

Cookie Clicker is a popular online incremental game. The goal of the game is to bake as many cookies as possible by clicking on cookies or by using cookies as currency to buy items and upgrades that increase cookie generation rate. In this paper, we analyze strategies for playing Cookie Clicker optimally. While simple to state, the game gives rise to interesting analysis involving ideas from NP-hardness, approximation algorithms, and dynamic programming. You can try playing Cookie Clicker yourself here: http://orteil.dashnet.org/cookieclicker/.

Contents

Kai Xiao
kaix@mit.edu

1 Introduction

In Cookie Clicker, your goal is to bake as many cookies as possible. You can click on a big cookie icon to bake a cookie, and you can also obtain items that automatically bakes cookies for you over time. You can use the cookies you have baked to purchase items, which generate a certain number of cookies per second for you. Items can be purchased multiple times, but after each item purchase, the item's cost will increase at an exponential rate, given by $C_n = C_1 \cdot \alpha^{n-1}$, where C_1 is the cost of the first item and C_n is the cost of item n. In the actual game, $\alpha = 1.15$. Because of the exponential cost growth of these items, item costs will grow faster than cookie generation rates can catch up, so cookie generation rates will not blow up exponentially over time. In the actual game, there is no end condition.

2 Models

In this paper, we will generally assume that you start with 0 cookies and that the rate of cookies baked per second from clicking on the big cookie icon is 1. We will describe each item by a tuple (x, y, α), where x denotes how much the item will increase your generation rate, y denotes the initial cost of the item, and α denotes the increase in item cost after each purchase. The case where $\alpha = 1$ for every item is a special case called the *fixed-cost* case.

We will analyze various versions of Cookie Clicker, which give rise to interesting and varied results. We will analyze the game when there is exactly 1 item available, the most basic case. Then, we will analyze cases with 2 items and n items respectively. For each possibility for the number of items, we will analyze both the fixed-cost case and the regular case.

We will also define a generalized version of Cookie Clicker that allows you to start with some amount of cookies. Finally, we will analyze a discretized version of Cookie Clicker where decisions regarding whether or not to buy an item happen in time steps.

There are multiple possible objective functions that we could want to optimize for. The most obvious ones are

1. Reaching M cookies in as little time as possible
2. Reaching a generation rate of G in as little time as possible

We will primarily analyze the first variant, and I will use "Cookie Clicker" to refer to this specific variant. I will use "Generation-Rate Cookie Clicker" to refer to the second variant.

3 Positive Results

3.1 1 Item Cookie Clicker

Suppose you start with 0 cookies and your goal is to reach M cookies as quickly as possible. The only item available has initial cost y, increases your cookies per second by x (we can call this quantity the rate increase), and has cost gain $\alpha \geq 1$. You now seek to find the optimal sequence and timing of item purchases to achieve your goal. Given these constants, we define a game state as a tuple (c, n) where c is the number of cookies you have and n is the number of items you have. Note that your current status in the game is entirely described by the game state.

3.1.1 Useful tools

Claim 3.1. *If the next step of the optimal strategy involves buying an item at some point in the future, you should buy the item as soon as you can afford it.*

Proof. Suppose that from a given game state, a strategy involves buying the item t seconds after you can afford it. Let G denote the cookie generation rate at the current game state. The net change in game state after these t seconds is that you gained 1 item, and your cookie amount changed by $Gt - y'$.

Then consider the strategy that buys the item as soon as you can afford it and waits t seconds afterwards. At that time, the net change in game state after these t seconds is that you gained 1 item, and your cookie amount changed by $(G + x)t - y'$. Thus, this new strategy results in the exact same thing as the original strategy, except that it gains an extra tx cookies, which is strictly better. Thus, an optimal strategy that intends to buy an item must buy it as soon as it can be afforded. □

This claim tells us that the optimal strategy will always decide to wait until it can purchase an item and purchase it immediately, or it will wait until the target number of cookies M is reached. Thus, the problem really just involves jumping between game states in which you have 0 cookies and need to make a decision between waiting until the end or buying an item. This means that the only thing we need to keep track of to determine our game state is n, the number of items we have purchased. Thus, we can define $f(n)$ to be the minimum amount of time needed to reach M cookies from the game state $(0, n)$.

From this claim, we can deduce that the optimal solution will have 2 phases. We will call them the "**Buying Phase**," where the solution tries to buy items, and the "**Waiting Phase**," where the solutions has bought all the items it needs and just waits until the items generate M cookies. Thus, every optimal solution can be represented by the sequence of items that should be bought in the "Buying Phase."

Next, we determine conditions for when buying an item is optimal.

Claim 3.2. *If your current cookie generation rate is G, you should buy an item with cost y and rate increase x if and only if*

$$\frac{M}{y} \geq 1 + \frac{G}{x} \tag{1}$$

Proof. Suppose we are at a state where we have purchased n_0 items. Then, the optimal decision is either to purchase another item or to enter the "Waiting Phase."

In the first case, the time taken is equal to

$$\frac{y}{G} + f(n_0 + 1) \leq \frac{y}{G} + \frac{M}{G + x}$$

because a valid (but possibly not optimal) strategy from the state $(0, n_0 + 1)$ is to wait.

In the second case, the time taken is equal to $\frac{M}{G}$.

Thus, if it is the case that

$$\frac{y}{G} + \frac{M}{G + x} \leq \frac{M}{G} \tag{2}$$

then we should go with the first strategy and purchase an item. Rearranging (2) gives (1). This means that if (1) is satisfied, purchasing the item is better.

Now we show that if (1) is not satisfied, then waiting is better. Suppose that (1) is not satisfied, so $\frac{M}{y} < 1 + \frac{G}{x}$. Then for any rate $G' > G$ and $y' \geq y$, the inequality $\frac{M}{y'} < 1 + \frac{G'}{x}$ still holds. Written in the form of (2), this inequality becomes $\frac{y}{G} + \frac{M}{G+x} > \frac{M}{G}$. Now, suppose that the optimal strategy from this point forward is to purchase k items for some $k > 0$ and then wait. Let y_i and G_i denote the price and cookie generation rate after i item purchases from this point forward, and note that $G_m = G_{m-1} + x$. Then the time taken to achieve this equals

$$\begin{aligned}
\frac{y}{G} + \frac{y_1}{G_1} + \cdots + \frac{y_{k-1}}{G_{k-1}} + \frac{y_k}{G_k} + \frac{M}{G_k + x} &> \frac{y}{G} + \frac{y_1}{G_1} + \cdots + \frac{y_{k-1}}{G_{k-1}} + \frac{M}{G_k} \\
&= \frac{y}{G} + \frac{y_1}{G_1} + \cdots + \frac{y_{k-1}}{G_{k-1}} + \frac{M}{G_{k-1} + x} \\
&> \frac{y}{G} + \frac{y_1}{G_1} + \cdots + \frac{y_{k-2}}{G_{k-2}} + \frac{M}{G_{k-2} + x} \\
&\cdots \\
&> \frac{y}{G} + \frac{M}{G + x} \\
&> \frac{M}{G}
\end{aligned}$$

Thus, if (1) is not satisfied, then the optimal strategy is to wait. This completes the if and only if statement's proof. □

Isolating G from **Claim 3.2** and reversing the statement, the following claim holds.

Claim 3.3. *If your current cookie generation rate is G, you should stop buying items with cost y and rate increase x if and only if*

$$G > \frac{Mx}{y} - x \tag{3}$$

3.1.2 Solution

Armed with the tools we developed in the previous section, we solve the 1 Item Cookie Clicker problem. Based on the results of the previous section, the optimal strategy is to purchase k items for some $k \geq 0$ as soon as each item becomes affordable and then wait until we reach M cookies. The total time that this takes is

$$\sum_{n=0}^{k-1} \frac{y \cdot \alpha^n}{1+nx} + \frac{M}{1+kx} \tag{4}$$

Using the results of **Claim 3.2**, we know that we should stop buying items when $\frac{M}{y} < 1 + \frac{G}{x}$. Here, $G = 1 + kx$ and $y = y \cdot \alpha^k$.

In the special case of $\alpha = 1$, which we call the *fixed-cost* case, the inequality becomes

$$\begin{aligned} \frac{M}{y} &< 1 + \frac{1+kx}{x} = 1 + k + \frac{1}{x} \\ \iff \frac{M}{y} - 1 - \frac{1}{x} &< k \end{aligned}$$

so k is the smallest integer larger than $\frac{M}{y} - 1 - \frac{1}{x}$. In this case, the total time is equal to

$$\begin{aligned} \sum_{n=0}^{k-1} \frac{y}{1+nx} + \frac{M}{1+kx} &= \frac{y}{x} \sum_{n=0}^{k-1} \frac{1}{1/x+n} + \frac{M}{1+kx} \\ &\approx \frac{y}{x} \sum_{n=0}^{k-1} \frac{1}{n} + \frac{M}{1+kx} \\ &\approx \frac{y}{x} \ln k + \frac{M}{1+kx} \\ &\approx \frac{y}{x} \ln \frac{M}{y} + \frac{M}{\frac{Mx}{y}} \\ &= \frac{y}{x} (\ln \frac{M}{y} + 1) \end{aligned}$$

If $\alpha > 1$, the inequality then becomes

$$\begin{aligned} \frac{M}{y \cdot \alpha^k} &< 1 + \frac{1+kx}{x} = 1 + k + \frac{1}{x} \\ &\iff \frac{M}{y(1+k+\frac{1}{x})} < \alpha^k \\ &\iff \log_\alpha \frac{M}{y} - \log_\alpha (1 + k + \frac{1}{x}) < k \end{aligned}$$

In most reasonable cases, the log term on the left hand side of the inequality is fairly small, so $k \approx \log_\alpha \frac{M}{y}$.

3.2 Fixed Cost Cookie Clicker for 2 Items

In this section, we analyze the cases where all the α's are equal to 1, which we call *Fixed Cost Cookie Clicker*. This is a natural starting point, as it corresponds to the economic situation in which items are fixed in price due to enough supply existing.

In the 2 Item Cookie Clicker problem, our goal is to reach M cookies as quickly as we can, and the 2 items available are described by the tuples (x_1, y_1) and (x_2, y_2). These are defined analogously to the 1 item case. Without loss of generality, we can assume that $y_2 > y_1$. In this problem, we will also make the assumption that $\frac{x_2}{y_2} > \frac{x_1}{y_1}$. This is because if the reverse inequality held, then buying $\frac{y_2}{y_1}$ copies of item 1 gives a higher rate increase than buying a single instance of item 2, which means that it should never be optimal to buy item 2 if M is large enough (M must be large enough so that the effect of $\frac{y_2}{y_1}$ not being an integer is irrelevant in the long run).

Given these constants, we define a game state as a tuple (c, n_1, n_2) where c is the number of cookies you have and n_1 and n_2 are the number of item 1 and item 2 you have obtained respectively. We can apply the two claims from the 1 item case here too, so we know from **Claim 3.1** that the optimal strategy will jump between states where we have 0 cookies, and that there is a "Buying Phase" and a "Waiting Phase." As before, we can represent every optimal solution by the sequence of items that should be bought in the "Buying Phase."

Additionally, we can determine an upper bound on the generation rate G beyond which it will not be worth it to purchase any more items. The modified version of **Claim 3.3** is as follows:

Claim 3.4. *If your current cookie generation rate is G and the items have rate increases and costs (x_i, y_i), you should stop buying items if and only if*

$$G > \max_i \frac{Mx_i}{y_i} - x_i \tag{5}$$

We now solve this problem. I will show that the sequence of items in the "Buying Phase" must be $[1, 1, \ldots, 1, 2, 2, \ldots, 2]$ when M is large enough. Then,

finding the optimal solution simply involves figuring out when to stop buying item 1 and when to start buying item 2, which can be determined in polynomial time.

To help us solve our problem, we will define the following.

Definition 3.5. *The efficiency score of an item of cost y and rate increase x when you have generation rate G is $\frac{y}{x} + \frac{y}{G}$.*

Claim 3.6. *If you plan to buy both items, you should always buy the item with the lower efficiency score. In particular, let $T = (y_2 - y_1)/(\frac{y_1}{x_1} - \frac{y_2}{x_2})$. Then, if $G < T$, you should never purchase item 2 followed by item 1, and if $G > T$, you should never purchase item 1 followed by item 2.*

Proof. The efficiency score of an item dictates whether or not buying item 1 then item 2 is better than buying item 2 then item 1.

Suppose we have generation rate G. Then the cost of buying item 1 then item 2 is equal to $\frac{y_1}{G} + \frac{y_2}{G+x_1}$ and the cost of buying item 2 then item 1 is equal to $\frac{y_2}{G} + \frac{y_1}{G+x_2}$. If $G < T$, we can rearrange the inequality to get that

$$\begin{aligned}
&\frac{y_1}{G} + \frac{y_1}{x_1} < \frac{y_2}{G} + \frac{y_2}{x_2} \\
\iff\ & y_1\Big(\frac{G+x_1}{x_1}\Big) < y_2\Big(\frac{G+x_2}{x_2}\Big) \\
\iff\ & y_1\Big(\frac{x_2}{G(G+x_2)}\Big) < y_2\Big(\frac{x_1}{G(G+x_1)}\Big) \\
\iff\ & y_1\Big(\frac{1}{G} - \frac{1}{G+x_2}\Big) < y_2\Big(\frac{1}{G} - \frac{1}{G+x_1}\Big) \\
\iff\ & \frac{y_1}{G} + \frac{y_2}{G+x_1} < \frac{y_2}{G} + \frac{y_1}{G+x_2}
\end{aligned}$$

On the other hand, if $G > T$, then the reverse is true.

□

Now, suppose that we have some optimal solution represented as a sequence of item 1's and item 2's. Now, we know that until the rate G reaches T, we will never have a 2 followed by a 1. Similarly, after the rate G passes T, we will never have a 1 followed by a 2. Thus, the final sequence must be of the following form.

$$[1, 1, \ldots 1, 1, 2, 2, \ldots, 2, 2, 1, 1, \ldots, 1, 1]$$

Somewhere in the middle of the list of 2's, the generation rate reaches T.

Now, I will show that for large enough M, there will be no list of 1's at the end of the optimal solution.

Claim 3.7. *Let $f(x_1, x_2, y_1, y_2) = \max\Big(2, \frac{2}{x_1} \times \frac{y_1+y_2}{\frac{y_1}{x_1} - \frac{y_2}{x_2}}\Big)$ If $M \geq (f+2)y_1$, then the optimal solution will have no 1's at the end.*

Proof. Suppose for the sake of contradiction that there are k 1's at the end of the list representing the optimal solution for some $k > 0$. I will show that replacing the final 1 with a 2 results in a better solution, which disproves the optimality of the original solution.

Denote that the rate before purchasing the final 1 in the optimal solution as R.

The time it takes to buy the final 1 and then wait until the goal M is reached is equal to $\frac{y_1}{R} + \frac{M}{R+x_1}$. The time it takes to buy a 2 instead of the final 1 and then wait until the goal M is equal to $\frac{y_2}{R} + \frac{M}{R+x_2}$. We want to prove that

$$\frac{y_2}{R} + \frac{M}{R+x_2} < \frac{y_1}{R} + \frac{M}{R+x_1} \tag{6}$$

$$\iff \frac{M}{R+x_2} - \frac{M}{R+x_1} < \frac{y_1}{R} - \frac{y_2}{R} \tag{7}$$

Now, we know from **Claim 3.2** and the fact that the optimal solution bought the final 1 that

$$1 + \frac{R}{x_1} < \frac{M}{y_1} \iff \frac{M}{R+x_1} > \frac{y_1}{x_1}$$

Similarly, we know that since the optimal solution can not buy another 2 after the final 1 that

$$1 + \frac{R+x_1}{x_2} \geq \frac{M}{y_2} \iff \frac{M}{R+x_1+x_2} \leq \frac{y_2}{x_2}$$

Combining the above two equations, we end up with

$$\begin{aligned}
&\frac{M}{R+x_1+x_2} - \frac{M}{R+x_1} < \frac{y_2}{x_2} - \frac{y_1}{x_1} \\
\iff &\frac{M}{R+x_2} - \frac{M}{R+x_1} < \frac{y_2}{x_2} - \frac{y_1}{x_1} + \frac{M}{R+x_2} - \frac{M}{R+x_1+x_2} \\
&= \frac{y_2}{x_2} - \frac{y_1}{x_1} + \frac{Mx_1}{(R+x_2)(R+x_1+x_2)} \\
&< \frac{y_2}{x_2} - \frac{y_1}{x_1} + \frac{Mx_1}{R^2}
\end{aligned}$$

Thus, to prove (5), we just have to prove that

$$\begin{aligned}
&\frac{y_2}{x_2} - \frac{y_1}{x_1} + \frac{Mx_1}{R^2} < \frac{y_1}{R} - \frac{y_2}{R} \\
\iff &\frac{Mx_1}{R^2} + \frac{y_2 - y_1}{R} < \frac{y_1}{x_1} - \frac{y_2}{x_2}
\end{aligned}$$

Now note that because the optimal solution can not buy another 1 after the final 1,

$$1 + \frac{R+x_1}{x_1} \geq \frac{M}{y_1} \iff R \geq \Big(\frac{M}{y_1} - 2\Big)x_1$$

Since $M \geq (f+2)y_1$ and $f \geq 2$, we can deduce that

$$R \geq \Big(\frac{M}{y_1} - 2\Big)x_1 = fx_1 \geq \frac{f+2}{2}x_1 = \frac{Mx_1}{2y_1}$$

Thus, $\frac{Mx_1}{R^2} = \frac{M}{R}\frac{x_1}{R} \leq \frac{2y_1}{x_1}\frac{x_1}{R} = \frac{2y_1}{R}$. Using this, all we have to prove now is that

$$\begin{aligned} \frac{2y_1}{R} + \frac{y_2 - y_1}{R} = \frac{y_1 + y_2}{R} &< \frac{y_1}{x_1} - \frac{y_2}{x_2} \\ \iff \frac{y_1 + y_2}{\frac{y_1}{x_1} - \frac{y_2}{x_2}} &< R \end{aligned}$$

But this is true because we know

$$R \geq \frac{Mx_1}{2y_1} \geq \frac{(f+2)x_1}{2} > \frac{fx_1}{2} > \frac{y_1 + y_2}{\frac{y_1}{x_1} - \frac{y_2}{x_2}}$$

□

Thus, we have shown that for large enough M, the optimal solution will be of the form

$$[1, 1, \ldots 1, 1, 2, 2, \ldots, 2, 2]$$

where the 1's only appear if the total generation rate at that point is less than the threshold T. We can experimentally verify that the point at which the optimal solution transitions from 1's to 2's is not exactly T, but is usually close to $\frac{T}{2}$.

Thus, to solve Fixed Cost Cookie Clicker for 2 Items, we just have to find the optimal number of 1's to buy before transitioning to 2's and subsequently solving the 1-item cookie clicker game. Finding this optimal number involves maximizing a function of a single variable (the number of 1's to buy before transitioning to 2's), which can be done in polynomial time.

3.3 Fixed Cost Cookie Clicker for n Items

A natural follow-up is to extend this problem from 2 items to n items. So far, no progress has been made on a strongly polynomial time solution, though a weakly polynomial dynamic programming solution exists. Experimentally, it appears that all such solutions involve buying more efficient items as generation rate increases. In particular, we have not found solutions where an item I is bought, then different item(s) are bought, then item I is bought again.

3.3.1 Dynamic Programming Solution

In the fixed cost case, the items do not change in price over time. Thus, the only thing that determines the game state is the current generation rate. Using **Claim 3.4**, the final generation rate is bounded above by $\max_i \frac{Mx_i}{y_i}$. Let DP[n] denote

the minimal amount of time needed to reach a goal of M from a state where you have 0 cookies and generation rate n. This corresponds to entering the "Waiting Phase" after achieving a generation rate of n. The dynamic program can be solved by the following recurrence.

$$DP[n] = \min\left(\frac{M}{n}, \min_i \frac{y_i}{n} + DP[n + x_i]\right) \tag{8}$$

If there are k items and $R = \max_i \frac{x_i}{y_i}$, then solving the original problem, which is equivalent to finding the value of DP[1], takes $O(MRk)$.

3.3.2 Local Optimizations

One idea for a faster algorithm is to use "local optimizations" on a given solution sequence to try to obtain a better one.

As we saw from Section 3.2, one example of a local optimization is a "swap" that involves swapping consecutive elements in a solution if doing so improves the solution. Another natural local optimization, which we saw in the proof of **Claim 3.7**, was the "replacement" of one item in a solution sequence with another. These local optimizations are easy to analyze because they does not affect other parts of the solution.

Thus, to try to solve this problem, I tried using random local optimizations on a randomly generated initial solution until it reached a point where local optimizations could no longer improve the solution. The local optimizations we used included.

1. Adding an item to the solution at a specific index
2. Deleting an item from the solution at a specific index
3. Replacing one item with a different item at a specific index
4. Moving an item from one index to another. If an item is moved from index $i + 1$ to i, this is equivalent to a swapping of consecutive elements.
5. Sorting the solution so that the cheapest items come first

In general, these local optimizations would improve the initial solution sequences. In some cases these local optimizations would result in a final solution that matched the globally optimal solution computed using dynamic programming. However, in other cases, these local optimizations get stuck at a local optimum - these are points which are worse than the globally optimal solution, but can not be improved any further using any of the local optimization operations. Thus, this leads us to the conclusion that our list of local optimizations is insufficient for finding a global solution, and any proof that relies solely on these operations will not be able to show global optimality of a solution.

3.4 Increasing Cost Cookie Clicker for n Items

From this point forward, we assume that the α's, the cost increase rates, all satisfy $\alpha > 1$. This is how the original Cookie Clicker game works, and is also a reasonable assumption. It corresponds to the economic situation in which items increase in price due to limited supply.

3.4.1 Dynamic Programming Solution

In this section, I present a dynamic programming solution which finds optimal solutions in $O(k \log_\alpha^k(\frac{M}{y}))$ time.

For simplicity, let us solve the problem for the case with 2 items first. Recall that your state in the game is completely described by the tuple (n_1, n_2), where n_i denotes the number of item i you have purchased. Note that it will never be worth it to buy an item if the item costs more than the goal M. This gives us upper bounds on n_1 and n_2, namely, $n_i < \log_{\alpha_i} \frac{M}{y_i} + 1$. This motivates us to define

$$DP[a][b] := \text{the minimum time it takes to reach } M \text{ from the state (a,b).}$$

Let the cookie generation rate at state (a, b) be represented by $g_{ab} = 1 + ax_1 + bx_2$. From the state (a, b), the optimal strategy is one of the following three choices - waiting, buying item 1, or buying item 2. We can then derive the recurrence

$$DP[a][b] = \min\Big(\frac{M}{g_{ab}}, \frac{y_1 \cdot \alpha_1^a}{g_{ab}} + DP[a+1, b], \frac{y_2 \cdot \alpha_2^b}{g_{ab}} + DP[a, b+1]\Big) \tag{9}$$

corresponding to each of those three choices. If A and B are the upper bounds for n_1 and n_2 respectively, then we can revise the recursive formulas for $DP[A][b]$ and $DP[a][B]$ for any a and b in the proper range to only correspond to two choices (e.g. for $DP[A][b]$, you can either buy item 2 or wait). We only need to initialize the value $DP[A][B] = \frac{M}{g_{AB}}$ and then use the recurrence to fill out the rest of the dynamic programming table. Finally, our answer is $DP[0][0]$.

Initializing the single boundary value takes $O(1)$ time. Then, filling out the rest of the AxB table takes $O(AB)$ time, so the total runtime is $O(AB)$.

This dynamic programming approach can easily be extended to the k item problem. As before, one can note that the game state of the k item problem is described entirely by the k-tuple $(n_1, n_2, \ldots, n_k)$, where n_i is the number of item i that you have purchased. We can use the same upper bound $n_i < \log_{\alpha_i}(\frac{M}{y_i}) + 1$. Let $N_i = \log_{\alpha_i}(\frac{M}{y_i}) + 1$ denote the upper bounds for each n_i.

We can similarly define $DP[(n_1, n_2, \ldots, n_k)]$ to be the minimum time it takes to reach M from the state $(n_1, n_2, \ldots, n_k)$. Then, filling in any square in the grid involves checking the solutions the adjacent squares and doing an $O(1)$ computation for each adjacent square. In total, this takes $O(k)$ time. The only square we need

to initialize is the corner $DP[(N_1, N_2, \ldots, N_k)]$. Thus, the total time complexity of this program is $O(k\prod_{i=1}^{k} N_i) = O(kN_{max}^k) = \boxed{O\Big(k\log_\alpha^k\Big(\frac{M}{y}\Big)\Big)}$.

3.4.2 Greedy Solutions

Natural Greedy Solution: One greedy solution that arises naturally in normal gameplay involves buying the item that has the highest rate increase to cost ratio $\frac{x_i}{y_i}$ if it is better to buy that item than to wait. This is the calculation that most human players do when playing the game unaided. For the 2 item case, given most reasonable setting of the parameters, this approach actually performs fairly well. However, for certain settings of the parameters, this approach can be quite bad. For example, take $M = 10000, (x_1, y_1, \alpha_1) = (1, 10, 2), (x_2, y_2, \alpha_2) = (10000, 9999, 2)$.

Efficiency Score Greedy Solution: As we saw in the analysis of the fixed cost case, the efficiency score was a helpful metric to determine which item to buy. Another idea for a greedy algorithm is to compute the efficiency score of each item and always choose the item with the lowest efficiency score. This approach is generally very close to optimal. In fact, we can prove approximation guarantees for this greedy solution.

3.4.3 Approximation Ratio

Once again, we will begin with the 2 Item case. We derive an approximation ratio of $1+\epsilon$ for the **Efficiency Score Greedy Solution** for sufficiently large M. The main idea is to use two facts:

1. Before you're anywhere close to reaching your goal M, you'll want to purchase another copy of item 1 and another copy of item 2
2. When G is big enough, α is big enough such that the most efficient item is locally and globally optimal

Suppose for now that the first fact is true. I will show that the second fact is then true given certain constraints on G.

Suppose WLOG that we are currently at a state where buying items in the order $1, 2$ is better than buying them in the order $2, 1$. We will say that this state has the E_1 property, meaning that item 1 is currently more efficient, and thus the locally optimal decision would be to buy item 1.

Due to fact 1, we know that any optimal solution from this current state will purchase k copies of item 2 followed by a copy of item 1. We want to show that given a solution where $k > 0$, we can produce a better solution by choosing to purchase item 1 first before purchasing any of the k copies of item 2. That is, given that $[2, 1]$ is worse than $[1, 2]$, then

$$[2, 2, \ldots 2, 1] \text{ is worse than } [1, 2, 2, \ldots 2]$$

which can be argued by sequentially showing that

$$\begin{aligned}[2,2,\dots 2,2,1] \text{ is worse than } &[2,2\dots 2,1,2]\\ \text{is worse than } &[2,2\dots 1,2,2]\\ &\dots\\ \text{is worse than } &[1,2,2,\dots 2,2]\end{aligned}$$

This would be true if the E_1 property still held after each purchase of item 2. Intuitively, it should, because purchasing item 2 actually makes future purchases of item 2 more expensive. However, the E_1 property doesn't necessarily hold after some number of purchases of item 2 because when the generation rate goes up, the more expensive item, which could be item 2, could become the more efficient item. Thus, we will rely on the following claim.

Claim 3.8. *Let $q_2 = \frac{G}{x_2}$. Suppose that a state with generation rate G satisfies the E_1 property. Then the next item that should be purchased is item 1 if $q_2^2 + 2q_2 \geq \frac{1}{\alpha_2 - 1}$.*

Proof. Let y_1 and y_2 denote the current costs of item 1 and item 2 after factoring in the cost increases. If item 1 has a lower efficiency score than item 2 at generation rate G, then

$$\frac{y_1}{x_1} + \frac{y_1}{G} \leq \frac{y_2}{x_2} + \frac{y_2}{G}$$

First, let us define some notation. Suppose that L_1 and L_2 are two lists that contain the same number of 1's and 2's, but in different orders. Then, we will say that $L_1 > L_2$ if the sequence of purchases in L_1 takes less time than the sequence of purchases in L_2. Using this notation, we are currently in a state where $[1,2] > [2,1]$.

First, we will find conditions such that $[1,2] > [2,1] \implies [2,1,2] > [2,2,1]$.

To show that $[2,1,2] > [2,2,1]$, we need to prove that

$$\frac{y_1}{x_1} + \frac{y_1}{G + x_2} \leq \alpha_2\Big(\frac{y_2}{x_2} + \frac{y_2}{G + x_2}\Big)$$

We know that

$$\frac{y_1}{x_1} + \frac{y_1}{G + x_2} < \frac{y_1}{x_1} + \frac{y_1}{G} \leq \frac{y_2}{x_2} + \frac{y_2}{G}$$

so we just have to show that

$$\begin{aligned}
&\frac{y_2}{x_2}+\frac{y_2}{G} && \leq \alpha_2\Big(\frac{y_2}{x_2}+\frac{y_2}{G+x_2}\Big)\\
\iff &\frac{y_2}{G} && \leq (\alpha_2-1)\frac{y_2}{x_2}+\frac{\alpha_2 y_2}{G+x_2}\\
\iff &\frac{1}{x_2 q_2} && \leq (\alpha_2-1)\frac{1}{x_2}+\frac{\alpha_2}{x_2 q_2+x_2}\\
\iff &q_2+1 && \leq (\alpha_2-1)q_2(q_2+1)+\alpha_2 q_2\\
\iff &1 && \leq (\alpha_2-1)(q_2^2+2q_2)\\
\iff &\frac{1}{\alpha_2-1} && \leq q_2^2+2q_2
\end{aligned}$$

This is the original assumption in **Claim 3.8**. Thus, we know that $[2,1,2] > [2,2,1]$ if the original condition holds. We also know that $[1,2,2] > [2,1,2]$, so the optimal solution from our current state can not start with a $[2,1]$ sequence or a $[2,2,1]$ sequence.

Now, I claim that this holds for any string of 2's in the beginning - that is, $[2,2,\dots 2,1,2] > [2,2,\dots 2,2,1]$. This is true because at the point where $[1,2]$ needs to be compared to $[2,1]$, the generation rate G' satisfies $G' > G$, so $q_2' > q_2$. Then, the same argument holds because

$$\frac{1}{\alpha_2-1} \leq q_2^2+2q_2 < q_2'^2+2q_2'$$

Thus, if G is large enough and we are in a state that satisfies the E_1 property, then the next item that should be purchased must be item 1. □

Next, I will prove one more claim that helps us analyze approximation ratios.

Claim 3.9. *Let $OPT(n_1,n_2)$ denote the minimum amount of time needed to reach the goal M from a state where you have 0 cookies, $n_1 \geq 0$ copies of item 1 have been purchased, and $n_2 \geq 0$ copies of item 2 have been purchased. If $n_1+n_2>0$, then $OPT(n_1,n_2) < OPT(0,0)$.*

Proof. Consider the strategy $S(n_1,n_2)$ that basically mirrors the strategy of $OPT(0,0)$ except that it doesn't purchase items when it doesn't need to. Specifically, when $OPT(0,0)$ purchases its n-th copy of item 1, S will also choose to purchase the same item unless $n < n_1$, in which case S will do nothing. When $OPT(0,0)$ enters the "Waiting Phase," S will as well. Due to how S is defined, when S gets to the "Waiting Phase," the generation rate of S will be at least that of $OPT(0,0)$. On the other hand, anything that happens before that will take S less time than $OPT(0,0)$ takes because S starts with more items and this gives S a higher generation rate and/or lets S save time because S may not have to purchase some items that $OPT(0,0)$ does. Thus, it is clear that, $OPT(n_1,n_2) \leq S(n_1,n_2) < OPT(0,0)$. □

From **Claim 3.8** and **Claim 3.9**, we can show that the greedy algorithm that uses the efficiency score is an approximation algorithm with approximation ratio arbitrarily close to 1.

Suppose we are given an instance of Cookie Clicker for 2 Items with sufficiently large M. WLOG, suppose that $x_2 > x_1$. Let Q_2 be the larger positive solution to $x^2 + 2x = \frac{1}{\alpha_2 - 1}$.

Consider the greedy algorithm on this instance of Cookie Clicker. We will let $G(0,0)$ be the amount of time it takes to reach the goal M. It will reach a rate of $G \geq Q_2 x_2$ after some amount of time, which we will denote by T_1. The amount of time it takes to reach this point is bounded above by the amount of time it takes to purchase $\lceil Q_2 \rceil$ copies of item 2 while ignoring item 1, which is a function of the inputs x_2, y_2, α_2. Thus, $T_1 < d$, where $d = f(x_2, y_2, \alpha_2)$.

Next, suppose that when the greedy algorithm passes that rate, it has n_1 copies of item 1 and n_2 copies of item 2. Let $G(n_1, n_2)$ denote the amount of time it takes the greedy solution to reach the goal M from here. The greedy algorithm will continue to make locally optimal decisions from that point forward, which, as **Claim 3.8** shows, are globally optimal decisions. Thus $G(n_1, n_2) = OPT(n_1, n_2)$. Then, using **Claim 3.9**, we have that

$$\begin{aligned} G(0,0) &= T_1 + G(n_1, n_2) \\ &= T_1 + OPT(n_1, n_2) \\ &< d + OPT(n_1, n_2) \\ &< d + OPT(0,0) \end{aligned}$$

Finally, note that in the optimal solution, the maximum possible generation rate that the solution will have before going into the "Waiting Phase" is $x_1 \log_{\alpha_1} \frac{M}{y_1} + x_2 \log_{\alpha_2} \frac{M}{y_2}$, which grows with $O(\log M)$. Thus, the "Waiting Phase" will take at least $\frac{M}{O(\log M)}$ time, so $OPT(0,0) > c \cdot \frac{M}{\log M}$, where c is a function of $(x_1, x_2, y_1, y_2, \alpha_1, \alpha_2)$ but is independent of M.

The desired approximation ratio of our greedy algorithm is $\frac{G(0,0)}{OPT(0,0)}$, and we know that

$$\frac{G(0,0)}{OPT(0,0)} < \frac{d}{OPT(0,0)} + 1 < \frac{d}{c} \cdot \frac{\log M}{M} + 1$$

As $M \to \infty$, the quantity $\frac{d}{c} \cdot \frac{\log M}{M}$ goes to 0. Thus, the approximation ratio is arbitrarily close to 1 for sufficiently large M.

Indeed, experimental results confirm that using the efficiency score in a greedy algorithm performs exceptionally well relative to the optimal solution, and almost always matches the optimal solution, except possibly at the very beginning of the solution. Even then, at most a few decisions are ordered differently, and after the initial differences the two strategies match up exactly.

4 Negative Results

4.1 General Cookie Clicker

We will now define the General Cookie Clicker problem. The inputs to this problem are

- k, the number of cookies you start out with
- Vectors X, Y, and α, where each triple (x_i, y_i, α_i) represents the (generation rate gain, initial cost, cost gain) of each item
- r, the initial generation rate
- M, the target number of cookies

The goal of this game is to find the optimal order of items to purchase to reach the goal M as quickly as possible. This game is the original cookie clicker game with the added flexibility of starting with any number of cookies in the bank and a variable initial generation rate.

4.1.1 Dynamic Programming Solution

The dynamic programming solution from section **3.4.1** can be modified slightly to solve this generalized problem. We will use the 2 Item case to illustrate our example. Suppose that we start with k cookies and a generation rate of r.

Just as before, we know that the optimal solutions is to buy items whenever they are affordable or to enter the "Waiting Phase." Thus, if we have cookies left over and an item is affordable and beneficial, we will choose to buy that item right away. This means that right when the game starts, the strategy will be to buy some set of items all at once, until items are either not affordable or not beneficial, and then to wait until items become affordable again. This implies that each game state in the optimal solution can still be described by 2 numbers (n_1, n_2), corresponding to the number of item 1 that has been purchased and the number of item 2 that has been purchased.

Let C_{n_1,n_2} be the cost of purchasing n_1 copies of item 1 and n_2 copies of item 2. It seems like the number of left over cookies could be a third parameter, but the number of left over cookies is determined entirely by (n_1, n_2) based on this strategy. If $C_{n_1,n_2} < k$, the number of left over cookies is just $k - C_{n_1,n_2}$, and if $C_{n_1,n_2} > k$ the number of left over cookies is exactly 0. Let $L_{n_1,n_2} = \max\left(k - C_{n_1,n_2}, 0\right)$ denote the number of leftover cookies.

Thus, as before, each state can be described by 2 numbers, and we can derive the recurrence.

$$DP[a][b] = \min\Big(\frac{M - L_{a,b}}{g_{ab}}, \frac{\max(y_1 \cdot \alpha_1^a - L_{a,b}, 0)}{g_{ab}} + DP[a+1, b], \frac{\max(y_2 \cdot \alpha_2^b - L_{a,b}, 0)}{g_{ab}} + DP[a, b+1]\Big)$$

Here, $g_{ab} = r + ax_1 + bx_2$.

We can generalize this formula to n Items and it will still hold. Thus, dynamic programming provides a weakly-polynomial time solution to the General Cookie Clicker problem as well.

4.1.2 NP-hardness of the General Cookie Clicker problem

We will now prove that General Cookie Clicker problem is NP-hard.

Theorem 4.1. *General Cookie Clicker is NP-Hard.*

Proof. Our strategy will be to use a reduction from 2-Partition. Suppose we are given an instance of 2-Partition in the form of a set of integers $(a_1, a_2 \dots a_n)$ such that

$$\sum_{i=1}^{n} a_i = 2B$$

We will construct an instance of General Cookie Clicker such that solving General Cookie Clicker will solve 2-Partition.

To do so, we can construct an instance of General Cookie Clicker with the inputs set as follows.

- $k = nA + B$
- $(x_i, y_i, \alpha_i) = (a_i + A, a_i + A, \infty)$ for $1 \le i \le n$
- $(x_i, y_i, \alpha_i) = (A, A, \infty)$ for $n + 1 \le i \le 2n$
- $r = 0$
- $M = nA + B + 1$

Here, A is just some big number (for example, $A = 1000B$). The fact that $\alpha_i = \infty$ means that we can buy at most 1 of each item. I will now show the following equivalence.

Claim 4.2. *A solution to 2-Partition exists if and only if the minimum amount of time it takes to reach M in the corresponding General Cookie Clicker problem is $\frac{M}{nA+B}$.*

Proof. First, suppose that a 2-Partition solution exists. That means we can choose some set of the integers $(a_1, a_2 \dots a_n)$ such that they sum to B. Equivalently, this means we can use our initial $k = nA + B$ cookies to buy n total items for a price of $nA + B$ at the very beginning of the game. We then wait until we have M cookies. The total amount of time it takes to reach M using this strategy is $\frac{M}{nA+B}$.

Next, I will show that there is no faster way to reach M cookies. Recall again the optimal strategy has a "Buying Phase" and a "Waiting Phase." We will analyze what rate the strategy ends up with at the end of the "Buying Phase."

Case 1: The strategy ends at a rate less than $nA + B$

If the strategy ends at a rate of $nA + B - j$ for $j > 0$, then it must have spent $nA + B - j$ purchasing items at $t = 0$ and then waited from that point forward. The total time that this strategy takes is $\frac{M-j}{nA+B-j}$. It's easy enough to verify that

$$\begin{aligned} \frac{M}{nA+B} < \frac{M-j}{nA+B-j} &\iff -jm < -j(nA+B) \\ &\iff nA + B < M \end{aligned}$$

Case 2: The strategy ends at a rate greater than $nA + B$.

First, I claim that the strategy can buy at most n items. This is because after buying n items, there will not be enough cookies left over to purchase the next item right away. The General Cookie Clicker problem then reduces to the original Cookie Clicker problem where you have no cookies at the start. We can then recall from **Claim 3.3** that if you have generation rate G, it is only worth it to buy an item with rate increase x and cost y if

$$\frac{Mx}{y} - x \geq G$$

Since $x_i = y_i$ for all i, this becomes

$$M - x_i \geq G \tag{10}$$

After purchasing n items, $G \geq nA$. Then $G + x_i \geq G + A \geq (n+1)A > M$, so it is no longer worth it to buy any items after purchasing n items.

Furthermore, suppose that we have purchased $n-1$ items. Then equation (10) tells us that item i is only worth purchasing if $M \geq G + x_i$. Thus, the final rate after purchasing the n^{th} item, $G + x_i$, is upper bounded by $M = nA + B + 1$. Since we are currently considering the case where the final rate is greater than $nA + B$, the only possible final rate for this case is then $nA + B + 1$.

Now, we have reduced this case to the specific scenario where $n - 1$ items are purchased at time $t = 0$, giving a rate of $(n-1)A + c$ for some $0 \leq c \leq 2B$. Then, the strategy purchases another item as soon as it can, ending up at a rate of

$nA+B+1$. After purchasing the first $n-1$ items, the strategy will have $A+B-c$ cookies left. The cost of the last item, which will push the rate up to $nA+B+1$, will be $nA+B+1-(n-1)A-c = A+B-c+1$. Thus, this strategy needs exactly 1 more cookie to purchase this last item.

The amount of time this strategy takes is

$$\frac{1}{(n-1)A+c} + \frac{M}{nA+B+1} \geq \frac{1}{(n-1)A+2B} + \frac{M}{nA+B+1}$$

We want to check that the quantity on the RHS is greater than $\frac{M}{nA+B}$. Indeed

$$\begin{aligned}
\frac{1}{(n-1)A+2B} + \frac{M}{nA+B+1} &> \frac{M}{nA+B} \\
\iff \frac{1}{(n-1)A+2B} &> \frac{M}{(nA+B)(nA+B+1)} \\
\iff \frac{1}{(n-1)A+2B} &> \frac{1}{nA+B} \\
\iff nA+B &> (n-1)A+2B \\
\iff A &> B
\end{aligned}$$

which is true. Thus, any strategy that ends at a rate greater than $nA+B$ will also get to M slower than the proposed optimal method using the partition solution. The "if" direction of **Claim 4.2** is proven.

The "only if" direction of **Claim 4.2** is easy to prove using the same results. If we are given a solution to General Cookie Clicker that reaches the goal in $\frac{M}{nA+B}$ time, then our analysis for the "if" direction's proof shows that the final rate upon reaching the goal must have been $nA+B$. Since the cost and the rate of all items are the same, this means that we spent $nA+B$ on items, which means all of the money was spent at $t=0$. The only way we could have spent exactly $nA+B$ at time $t=0$ is if there existed a solution to 2-Partition, as desired.

□

We have shown that given an instance of 2-Partition, which is NP-complete, we can construct an instance of General Cookie Clicker in polynomial time such that being able to solve General Cookie Clicker means being able to solve 2-Partition. Thus, **Theorem 4.1** is proven and General Cookie Clicker itself is NP-hard. □

4.2 Discrete Cookie Clicker

This is a variant of Cookie Clicker in discrete time, whereas we have been analyzing it in continuous time. This model has also been proven to be NP-hard via a reduction from 3-Partition.

The problem statement is as follows: Given an initial amount of cookies k, an initial income r, and n items described by tuples (x_i, y_i, α_i), is there a strategy that can be made so that you obtain M cookies by target time T?

Theorem 4.3. *Discrete Cookie Clicker is NP-hard*

Proof. We reduce from 3-Partition. Suppose we are given an instance of 3-Partition $(a_1, a_2, \ldots, a_n)$, such that $\sum_{i=1}^{n} a_i = A$.

We will encode 3-Partition as Discrete Cookie Clicker as follows. Choose a number $B > \frac{An}{3}$.

1. For each number a_i, make an item described by the tuple $(x_i, y_i, \alpha_i) = (a_i, B \cdot a_i, \infty)$.
2. $M =$ Temporary Placeholder. Computing this value is not super instructive
3. $k = 0$ and $r = \frac{3BA}{n}$.
4. $T = \frac{n}{3} + 2B$

As before, we want to constrain the problem so that the optimal solution must proceed in two distinct phases - the "Buying Phase" and the "Waiting Phase." We will call these Phase 1 and Phase 2.

First, note that after time step $\frac{n}{3}$, ignoring any extra income we get from buying items in those time steps, we will have produced BA cookies just from our initial generation rate. Thus, it will be possible to purchase every single item by the end of time step $\frac{n}{3}$. In particular, Phase 1 will last at most $\frac{n}{3}$ turns.

Next, note that buying every single item results in a final generation rate of $\frac{3BA}{n} + A$. Thus, the generation rate is always upper bounded by $\frac{3BA}{n} + A$. At the end of time step $\frac{n}{3}$, the total amount of cookies generated will be at most $BA + \frac{An}{3}$. Because $\frac{An}{3} < B$ and all item costs are multiples of B (and thus at least B), we know that any cookies generated from our items and not from the original income will not increase our ability to purchase buildings in Phase 1. In other words, the cookies that our items generate will not improve our buying power, and we essentially get $\frac{3BA}{n}$ to spend every turn. We can also conclude that Phase 1 will last exactly $\frac{n}{3}$ turns.

Next, note that because the cost of every item scales linearly with the increase in generation rate, spending cookies on items will always produce the same increase in generation rate per cookie. Thus, the best way to spend cookies is to spend it earlier, because this maximizes the amount of extra generation rate every cookie will give you.

Thus, the optimal strategy is to spend as many of your cookies as possible at every time step in Phase 1, then wait in Phase 2. The fastest way to reach M will be to spend all $\frac{3BA}{n}$ of your generated cookies on every timestep in Phase 1. This is only possible if the numbers $(a_1, a_2, \ldots, a_n)$ can be partitioned into subsets such that for each subset, the total sum is $\frac{3A}{n}$, which is exactly the 3-Partition problem. □

5 Code

For python implementations of the dynamic programming solution and the greedy solutions to the General Cookie Clicker problem, the dynamic programming solution to the Fixed Cost Cookie Clicker problem, and various other scraps of code, see here: https://github.com/kaixiao/Cookie-Clicker

6 Future Work

I will analyze Generation-Rate Cookie Clicker, which was discussed as a possible model of the game but has not been analyzed. Further analysis of this variant can also be used to improve analysis of various parts of Cookie Clicker.

7 Conclusion

Cookie Clicker, while a seemingly simple game, gives rise to many interesting optimization problems. We were able to analyze these problems through the context of dynamic programming, approximation algorithms, and NP-hardness. For specific variants of Cookie Clicker, we were able to classify the structure of optimal solutions, thereby limiting our search space for such solutions. This allowed us to devise polynomial time algorithms for solving the problem. For more general variants of Cookie Clicker, NP-hardness results were proved via reductions from 2-Partition and 3-Partition. Although these problems are NP-hard, their solutions can be approximated very well with a specific greedy algorithm based on a specific efficiency metric, and we can prove an approximation ratio guarantee of $1 + \epsilon$.

This paper is currently a work in progress, and will be edited over time.