Cookie Clicker

Erik D. Demaine[*] Hiro Ito[†] Stefan Langerman[‡] Jayson Lynch[*]
Mikhail Rudoy[§] Kai Xiao[*]

Cookie Clicker[1] is a popular online incremental game where the goal of the game is to generate as many cookies as possible. In the game, you can click on a big cookie icon to bake a cookie, which we model as an initial cookie generation rate. You can also use the cookies you have generated as currency to purchase various items that increase your 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.

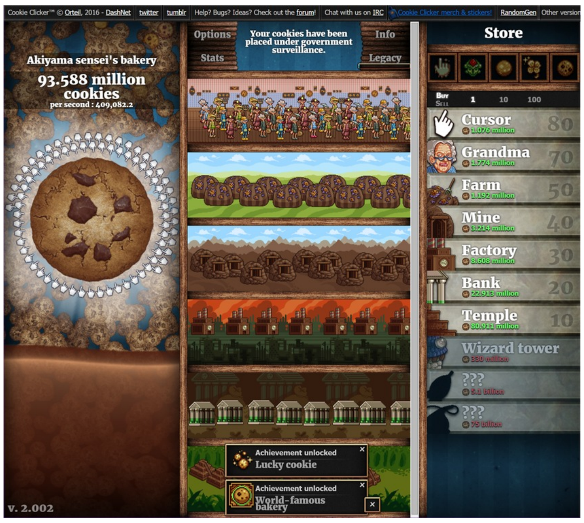

Figure 1: Screenshot of Cookie Clicker.

Each cookie-generating item in this game 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 the nth item. In the actual game, $\alpha = 1.15$. There is no real end condition in the game, but in this paper we have two possible end conditions: reaching a certain number M of cookies or reaching a certain cookie generation rate R.

Cookie Clicker falls into a broader class of popular online games called "incremental" games, in which the primary mechanic of the game is acquiring income and spending that income on income generators in order to acquire even more income. Some of the other well-known games in this genre include Adventure Capitalist, Cow Clicker, Clicker Heros, Shark Souls, Kittens Game, Egg Inc., and Sandcastle Builder (Based around the xkcd comic Time, number 1190).

Models. In most of this paper, we will assume that you start with 0 cookies and that the initial cookie generation rate 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 cookie generation rate, y denotes the initial cost of the item, and α denotes the multiplicative increase in item cost after each purchase. The case where $\alpha = 1$ for every item is a special case called the *fixed-cost* case. The goals of the game is to find the optimal sequence and timing of item purchases that minimizes some objective function.

There are multiple possible objective functions that we could want to optimize for, but we will focus on the following two:

1. Reaching M cookies in as little time as possible

[*]CSAIL, Massachusetts Institute of Technology
[†]University of Electro-Communications
[‡]Directeur de Recherches du F.R.S.-FNRS, Université Libre de Bruxelles
[§]Google Inc. Work completed at MIT CSAIL.
[1]`http://orteil.dashnet.org/cookieclicker/`

2. Reaching a generation rate of R in as little time as possible

Our analysis of various versions of Cookie Clicker gives rise to interesting and varied results; refer to Table 1. First, we present some general results, such as the fact that the optimal strategy involves a "Buying Phase" where items are purchased in some sequence as quickly as possible, and then a "Waiting Phase" where no items are purchased.

We begin our version-by-version analysis by examining the case where exactly 1 item is available for purchase, and we present formulas describing how many copies of the item should be purchased in both the fixed-cost case and the increasing-cost case.

Next, we analyze cases involving 2 items. In the 2 item fixed-cost case, we prove that the optimal solution always involves buying some number of copies of one item, followed by some number of copies of the other item.

Then, we analyze the case involving k items. In the k items fixed-cost case, a weakly polynomial time dynamic programming solution can be used to find the optimal sequence of items to buy, and in the increasing-cost case, a strongly polynomial time dynamic programming solution can be used. Additionally, a greedy algorithm can be devised with an approximation ratio that approaches 1 for sufficiently large values of M.

Afterwards, we present negative results, including proofs of Weak NP-hardness of the decision version of the problem of reaching a generation rate of R as quickly as possible, as well as for a version of Cookie Clicker that allows you to start with a nonzero number of cookies. Finally, we define a discretized version of Cookie Clicker where decisions regarding whether or not to buy an item happen in discrete time steps and prove Strong NP-hardness for that version.

Table 1: Summary of Results

Problem Variant	**Result for M version**	**Result for R version**
1 Item Fixed-Cost with item $(x, y, 1)$	Final answer is $\approx \frac{y}{x} \ln \frac{M}{y}$ $O(1)$ to solve	Final answer is $\approx \frac{y}{x} \ln \frac{R}{x}$ $O(1)$ to solve
1 Item Increasing-Cost with item (x, y, α)	Stop "Buying Phase" after $\log_\alpha \frac{M}{y}$ items $O(1)$ to solve	Stop "Buying Phase" after $\frac{R}{x}$ items $O(1)$ to solve
2 Items Fixed-Cost with items $(x_i, y_i, 1)$ where $y_2 > y_1$	Solutions of the form $[1, 1, \ldots, 1, 2, \ldots, 2]$ for large enough M $u_1 \log_\phi u_2 + O(u_1)$ to solve, where $u_i \approx \frac{y_i}{x_i} \log \frac{M}{y_i}$	Solutions of the form $[1, 1, \ldots, 1, 2, \ldots, 2, 1, 1]$ for a small number of 1's at the end for large enough R.
k Items Fixed-Cost with items $(x_i, y_i, 1)$	Dynamic Programming solution, runtime $O(\max_i(\frac{Mx_ik}{y_i}))$	Dynamic Programming solution, runtime $O(kR)$
k Items Increasing-Cost with items (x_i, y_i, α_i)	$O(\max_i(k \log^k_{\alpha_i} \frac{M}{y_i}))$ using Dynamic Programming Greedy Algorithm has Approximation Ratio of $1 + O(\frac{1}{\log M})$ for $k = 2$	$O(\max_i(k(\frac{R}{x_i})^k))$ using Dynamic Programming Weakly NP-hard by reduction from PARTITION
k Items Increasing-Cost with items (x_i, y_i, α_i) with Initial Cookies	Weakly NP-hard by reduction from PARTITION	Weakly NP-hard by reduction from M version
Discrete k Items Increasing-Cost with items (x_i, y_i, α_i) with Initial Cookies	Strongly NP-hard by reduction from 3-PARTITION	Strongly NP-hard by reduction from M version