

Optimal play in *Cookie Clicker*

Claude (Anthropic)

for jehlp.net and the Cookie Monster+ project

October 2026

Abstract

Cookie Clicker is an idle game. The player buys buildings whose prices grow geometrically, collects random golden cookies, and can reset the game for a permanent production multiplier. We treat play as a growth process with discrete purchases and resets. Within explicit models we prove four rules.

Purchases follow a pairwise exchange index, which the community payback period implements exactly when no candidate is affordable. Cookies held for the “Lucky!” golden cookie beat a purchase exactly when the purchase pays back more slowly than a threshold of about two hours at base frequency. Heavenly chips go to multiplicative upgrades by a Smith-type ratio. Late in the game, each run should multiply all-time cookies by a factor that depends only on the growth exponent of the run. This corrects the marginal value theorem by up to a factor of six. A dynamic program covers the early game.

Further rules cover achievements, Godzamok, wrinklers, the minigames and seasons. In a simulator built from the game’s formulas, the stopping rule and the planner reach 98.6% and 97.6% of the best outcome in hindsight. The Cookie Monster+ mod implements every rule.

MSC 2020: 90C39 (primary); 60G40, 90B35 (secondary).

Keywords: incremental games, optimal stopping, dynamic programming, exchange arguments, scheduling.

Contents

1	Introduction	3
1.1	The four rules	3
1.2	Scope and conventions	3
2	Model	4
2.1	Mechanics	4
2.2	Notation	4
2.3	Assumptions	4
3	Purchase order	5
4	Golden cookies and the bank	6
4.1	When to hold a “Lucky!” bank	6
4.2	Values under automation	7
5	Heavenly chips	8

6	When to ascend	9
6.1	Time rescaling	9
6.2	Formulation	10
6.3	Late game: a closed form	10
6.4	Early game: why early resets fail	12
6.5	The general planner	12
7	The rest of the game	13
7.1	Achievements	14
7.2	Godzamok	14
7.3	Wrinklers and wrath cookies	16
7.4	Minigames	17
7.5	The dragon, seasons, sugar lumps and permanent slots	19
8	Numerical validation	20
9	Implementation	21
10	Limitations	22
A	Algorithms	22
	References	23

1 Introduction

Cookie Clicker [8] is an incremental game. Clicks and buildings produce cookies. Each copy of a building costs 1.15 times the previous copy. Upgrades multiply production, and random *golden cookies* give temporary bonuses.

At any time the player can *ascend*. Ascension abandons the current run. In return the player gains *prestige*, a permanent production multiplier that grows like the cube root of all cookies ever baked. Prestige also gives *heavenly chips*, which buy permanent upgrades.

Four decisions recur in play: what to buy next, whether to keep cookies for golden cookies, how to spend heavenly chips, and when to ascend. This paper proves a rule for each decision within an explicit model. It then tests the rules in a simulator.

1.1 The four rules

Let c be the income rate. Item i costs p_i and adds Δ_i to the income.

1. *Purchase order* (Theorem 3.1). Buy the item with the smallest index

$$E_i = \frac{p_i}{c} + \frac{p_i}{\Delta_i}.$$

2. *The “Lucky!” bank* (Proposition 4.1). Spend a cookie on item i if and only if

$$\frac{p_i}{\Delta_i} < \theta = \frac{1}{0.15 g \lambda_L}.$$

Here $\bar{\Delta}_i$ is the full income gain of the item, g multiplies golden-cookie gains and λ_L is the rate of “Lucky!” cookies.

3. *Heavenly chips* (Theorem 5.1). Upgrade j costs p_j chips and multiplies speed by m_j . Buy the upgrades in decreasing order of

$$\rho_j = \frac{1 - 1/m_j}{p_j}.$$

4. *Ascension* (Corollary 6.3). Late in the game, ascend when all-time cookies have grown by the root v^* of

$$\frac{\beta}{3} \left(1 - \frac{1}{v}\right) = \frac{1 - v^{-\gamma/3}}{\gamma}, \quad \gamma = \frac{3}{\beta} - 1,$$

where β is the growth exponent of the current run.

Section 7 adds rules for the other systems: achievements, Godzamok, wrinklers, the minigames, the dragon and seasons. Section 8 tests the rules in a simulator built from the game’s building formulas.

1.2 Scope and conventions

“Optimal” always means optimal within the stated model. The proofs are exact under their assumptions. Numerical tests, not proofs, measure how well the assumptions fit the game. Cookie Monster+ [3], a fork of Cookie Monster [4], implements every rule (Section 9).

Times are in seconds unless a unit is given. All logarithms are natural. The game runs at 30 frames per second. A *run* is the period between two ascensions, and *CpS* means cookies per second from buildings.

2 Model

2.1 Mechanics

A run starts with no buildings. Building k with n copies costs

$$p_k(n) = p_k(0) 1.15^n. \quad (2.1)$$

Let A be the cookies baked in all previous runs, and C the cookies baked in the current run. Ascension converts all-time cookies into the prestige

$$P(A + C) = \left(\frac{A + C}{10^{12}} \right)^{1/3}. \quad (2.2)$$

Each prestige level adds h percent to production. The *prestige power* h becomes 1 after the heavenly upgrade *Legacy* (one chip) and five cheap store upgrades, *Heavenly chip secret* through *Heavenly key*. A run that starts at prestige P thus has the production multiplier

$$M(P) = 1 + 0.01 h \lfloor P \rfloor. \quad (2.3)$$

The player receives $\lfloor P \rfloor$ heavenly chips over the whole game. Each heavenly upgrade has a price in chips and a set of parent upgrades, which must be owned first.

2.2 Notation

Symbol	Meaning
c	income rate (cookies per second)
b	cookies in the bank
p_i, Δ_i	price of purchase i and the income it adds
A, C	cookies baked before the current run, and in it
P, h	prestige (2.2) and prestige power
S	speed of a run: the factor by which permanent bonuses multiply all income
$F(x)$	run curve: cookies baked after x units of speed-one time
β	local growth exponent of a run, $\beta = cT/C$
τ	time lost per ascension

2.3 Assumptions

Assumption A1. Between purchases the income rate is constant. Purchases are instantaneous.

Assumption A2. The income that a purchase adds does not depend on the order of a given pair of purchases.

Assumption A3. Permanent bonuses multiply every income source (buildings, clicks and golden-cookie rewards) by the same factor S .

Assumption A4. Purchase decisions depend on the bank and the holdings only through comparisons that do not change when all incomes are multiplied by a constant.

Assumption A5. (Section 6.3 only.) The run curve is a power law, $F(x) = \alpha x^\beta$ with $\beta > 1$.

3 Purchase order

Theorem 3.1 (Pairwise exchange). *Assume A1 and A2. Two purchases A and B occur one after the other, from a bank $b \leq \min(p_A, p_B)$ at income c . Buying A first completes both purchases no later than buying B first if and only if*

$$E_A \leq E_B, \quad E_i = \frac{p_i}{c} + \frac{p_i}{\Delta_i}. \quad (3.1)$$

Both orders end in the same state.

Proof. With A first, the bank reaches p_A after $(p_A - b)/c$ seconds and is then empty. Purchase B follows after a further $p_B/(c + \Delta_A)$ seconds. Thus

$$T_{AB} = \frac{p_A - b}{c} + \frac{p_B}{c + \Delta_A}, \quad T_{BA} = \frac{p_B - b}{c} + \frac{p_A}{c + \Delta_B}.$$

The bank b cancels in the difference, and

$$\begin{aligned} T_{AB} \leq T_{BA} &\iff p_A \left(\frac{1}{c} - \frac{1}{c + \Delta_B} \right) \leq p_B \left(\frac{1}{c} - \frac{1}{c + \Delta_A} \right) \\ &\iff \frac{p_A \Delta_B}{c + \Delta_B} \leq \frac{p_B \Delta_A}{c + \Delta_A} \\ &\iff \frac{p_A}{c} + \frac{p_A}{\Delta_A} \leq \frac{p_B}{c} + \frac{p_B}{\Delta_B}. \end{aligned}$$

The last step multiplies by $(c + \Delta_A)(c + \Delta_B)/(c \Delta_A \Delta_B) > 0$. After both purchases, the holdings, the empty bank and the income $c + \Delta_A + \Delta_B$ agree by A2. $\square$

Remark 3.2. Both orders end in the same state. Thus every objective of the form “reach a given state as early as possible” prefers the order with the smaller index. The rule “buy the candidate with the smallest index at the current income” is *locally optimal*: no swap of two consecutive purchases improves it. A single global sort is not possible, because E_i depends on c , which grows.

Corollary 3.3 (Payback period). *Cookie Monster ranks purchases by the payback period*

$$PP_i = \frac{\max(p_i - b, 0)}{c} + \frac{p_i}{\Delta_i}. \quad (3.2)$$

When no candidate is affordable, $PP_i = E_i - b/c$. The payback period and the exchange index then give the same ranking.

Proposition 3.4 (Affordable pairs). *If $\max(p_A, p_B) \leq b < p_A + p_B$, buying A first is better if and only if $\Delta_A > \Delta_B$.*

Proof. Buy A at once. The bank keeps $b - p_A < p_B$, and the remaining $p_A + p_B - b$ cookies arrive at the rate $c + \Delta_A$:

$$T_{AB} = \frac{p_A + p_B - b}{c + \Delta_A}.$$

The expression for T_{BA} is symmetric, with the same numerator. $\square$

With a large bank the exchange index is therefore not exact. Section 8 shows that the effect is small. Over nine scenarios, the bank-independent index or a look-ahead to tier upgrades changes the time to a target by at most 0.35% in continuous play. When purchases occur only once an hour, the change is at most 1.7%.

4 Golden cookies and the bank

4.1 When to hold a “Lucky!” bank

A “Lucky!” golden cookie pays

$$\min(0.15 b, 900 c) g + 13 \quad (4.1)$$

cookies, where g collects the golden-cookie gain multipliers. Below the cap, $b < 6000 c$, the reward is linear in the bank.

Proposition 4.1 (Marginal value of banked cookies). *Let “Lucky!” cookies occur at the rate λ_L , and let $b < 6000 c$. A marginal banked cookie earns $0.15 g \lambda_L$ cookies per second in expectation. A cookie spent on item i earns $\bar{\Delta}_i/p_i$, where $\bar{\Delta}_i$ is the expected increase in total income (Definition 4.4). Spending is better if and only if*

$$\frac{p_i}{\bar{\Delta}_i} < \theta, \quad \theta = \frac{1}{0.15 g \lambda_L}. \quad (4.2)$$

Proof. Below the cap, each “Lucky!” cookie pays $0.15 g$ per banked cookie. These cookies arrive at the rate λ_L . By linearity of expectation, a banked cookie returns $0.15 g \lambda_L$ per second. A cookie spent on item i returns $\bar{\Delta}_i/p_i$ per second. Compare the two marginal returns. $\square$

Without golden cookies or clicks, $\bar{\Delta}_i$ is the CpS gain Δ_i . With both automated, Δ_i undervalues every purchase: golden cookies and clicks multiply a CpS gain about eightfold (Example 4.3). With Δ_i alone, purchases whose payback lies between θ and several times θ would wait, although a purchase is better.

The rate is $\lambda_L = \pi_L/\mathbb{E}[I]$. Here $\pi_L \approx 0.42$ is the probability that a golden cookie is “Lucky!”, and I is the interval between golden cookies.

Lemma 4.2 (Golden-cookie interval). *Golden cookies spawn between $t_{\min}$ and $t_{\max}$. In each frame of the window, the spawn probability is $((t - t_{\min})/W)^5$, where t is the current frame and the window lasts W frames. Write $I = t_{\min} + x(t_{\max} - t_{\min})$. Then*

$$\mathbb{E}[x] \approx \Gamma(\frac{7}{6}) \left(\frac{6}{W}\right)^{1/6}. \quad (4.3)$$

Proof. The probability of surviving to the fraction x of the window is

$$\prod_{k \leq xW} (1 - (k/W)^5) \approx \exp\left(-W \int_0^x u^5 du\right) = \exp(-W x^6/6).$$

With $\int_0^\infty e^{-ax^n} dx = a^{-1/n} \Gamma(1 + 1/n)$,

$$\mathbb{E}[x] = \int_0^1 e^{-W x^6/6} dx \approx \int_0^\infty e^{-W x^6/6} dx = \left(\frac{6}{W}\right)^{1/6} \Gamma(\frac{7}{6}). \quad \square$$

Example 4.3. Table 1 evaluates the lemma with $\pi_L = 0.42$ and $g = 1$. The base window runs from 5 to 15 minutes. *Lucky day* and *Serendipity* each halve both limits.

Frequency	W (frames)	$\mathbb{E}[x]$	$\mathbb{E}[I]$	θ
Base	18000	0.244	447 s	7.1×10^3 s
Both upgrades	4500	0.308	121 s	1.9×10^3 s

Table 1: Mean golden-cookie interval and the bank threshold θ of (4.2).

Outcome	Share	Effect
Frenzy	$\pi_F \approx 0.42$	production $\times 7$ for 77 ds
“Lucky!”	$\pi_L \approx 0.42$	$\min(0.15b, 900c)g$
Building special	$\pi_B \approx 0.10$	CpS $\times (1 + n/10)$ for 30 ds ; n counts a random type with at least ten copies
Click frenzy	$\pi_C \approx 0.04$	clicks $\times 777$ for 13 ds
Cookie storm	$\pi_S \approx 0.0085$	about 105 d drops, each worth one to seven minutes of CpS

Table 2: Golden-cookie outcomes when every golden cookie is clicked at once ($d = 2$ under *Get lucky*).

In the second case, take building specials on types that average $\bar{n} = 60$ copies (Definition 4.4). Then $1 + \sigma_c \approx 4.5$ and $1 + \sigma_k \approx 6.1$. Three mouse upgrades each add 1% of CpS per click. At $r = 20$ clicks per second, $r\delta_i = 0.6\Delta_i$, so $\bar{\Delta}_i \approx 8.2\Delta_i$. A purchase then waits only if its CpS payback exceeds

$$8.2\theta \approx 1.6 \times 10^4 \text{ s},$$

four and a half hours. Early in a run, payback times are seconds to minutes, so spend every cookie. Late in a run, payback times are hours to days, so keep the “Lucky!” bank.

4.2 Values under automation

Cookie Monster’s simulator values upgrades by the CpS that they add. Click upgrades and golden-cookie upgrades therefore show a value of zero. Under automation they have a definite value.

Definition 4.4 (Automation values). Let c be CpS and k the income of an auto-clicker at r clicks per second. A purchase adds Δ_i to c and $r\delta_i$ to k , where δ_i is its gain in cookies per click, including click power that is a share of CpS. Table 2 lists the golden-cookie outcomes. Per unit of CpS and of click income, they add on average

$$\sigma_c = \frac{(462\pi_F + 3\bar{n}\pi_B + 25200g\pi_S)d}{\mathbb{E}[I]}, \quad \sigma_k = \frac{(462\pi_F + 3\bar{n}\pi_B + 10088\pi_C)d}{\mathbb{E}[I]}, \quad (4.4)$$

where $\bar{n}$ is the mean count over the building types with at least ten copies. A purchase raises the expected income by

$$\bar{\Delta}_i = (1 + \sigma_c)\Delta_i + (1 + \sigma_k)r\delta_i. \quad (4.5)$$

Golden cookies bring in

$$G = \sigma_c c + \sigma_k k + \lambda_L g \min(0.15b, 900c) \quad (4.6)$$

in all. The expected income, which fills the bank, is $c + k + G$.

Frenzies and building specials multiply CpS, and so also the click power that comes from it. Below the cap, “Lucky!” pays a share of the bank, not of production, so it is absent from (4.5).

The golden-cookie upgrades follow from (4.6). *Lucky day* and *Serendipity* double the frequency, so each is worth G . *Get lucky* doubles every duration and is worth $\sigma_c c + \sigma_k k$. *Golden goose egg* adds 5% frequency and is worth $0.05G$.

With incomes and gains measured this way, Theorem 3.1 and Proposition 4.1 apply without change. Both use the income without current buffs. Otherwise a Frenzy would make every

purchase look seven times better for its 77 seconds. The heavenly-upgrade values of Table 3 use the simpler shares

$$\sigma_F = \frac{462 \pi_F d}{\mathbb{E}[I]}, \quad \sigma = \sigma_F + \frac{\pi_L g \min(900, 0.15 b/c)}{\mathbb{E}[I]}, \quad (4.7)$$

which count Frenzies and “Lucky!” only.

5 Heavenly chips

Chips arrive over the course of the game. Earning the chips between the budget levels B and $B + dB$ takes the time

$$\frac{q(B)}{S(B)} dB,$$

where q is the number of cookies that each chip costs and $S(B)$ is the speed. The cost q does not depend on the purchases. Heavenly upgrade j costs p_j chips and multiplies S by $m_j > 1$. The prestige factor (2.3) is common to all orders, so q absorbs it.

Theorem 5.1 (Ratio rule for multiplicative speed-ups). *Suppose that there are no parent constraints and that q is constant over the budget range. Then the order of purchases that minimises the total time sorts the items by*

$$\rho_j = \frac{1 - 1/m_j}{p_j} \quad (5.1)$$

in decreasing order. A swap of an adjacent pair strictly improves any order that is not sorted.

Proof. Let Π be the product of the multipliers bought before two adjacent items i and j . A swap leaves every other term unchanged, because after both items the product is $\Pi m_i m_j$ in either order. Let S_0 be the initial speed. The contribution of the pair is

$$\frac{q}{S_0 \Pi} \left(p_i + \frac{p_j}{m_i} \right) \quad \text{with } i \text{ first}, \quad \frac{q}{S_0 \Pi} \left(p_j + \frac{p_i}{m_j} \right) \quad \text{with } j \text{ first}.$$

Hence i first is strictly better if and only if

$$p_i + \frac{p_j}{m_i} < p_j + \frac{p_i}{m_j} \iff p_i \left(1 - \frac{1}{m_j} \right) < p_j \left(1 - \frac{1}{m_i} \right) \iff \rho_i > \rho_j.$$

An unsorted order contains an adjacent inverted pair, and its swap strictly reduces the total time. There are finitely many orders, so a sorted order is optimal. $\square$

Remark 5.2.

1. For small effects $1 - 1/m \approx \ln m$, so the rule reduces to the log-gain per chip. For large effects the two differ: *Heralds* can be worth $m = 2$.
2. The rule implies *saving up*. A cheaper item with a lower ratio, bought first, delays an item with a higher ratio, and the interchange argument shows that this is worse. Cookie Monster+ therefore spends chips along the sorted order and stops at the first unaffordable item.

Heavenly upgrade	Chips	Production multiplier m
Heavenly cookies	3	1.10
Heralds	100	$1 + 0.01 \cdot \text{heralds}$ (heralds ≤ 100)
Heavenly luck	77	$1 + 0.05 \sigma$
Lasting fortune	777	$1 + 0.10 \sigma_F$
Lucky digit	777	$1.01 (1 + 0.01 \sigma_F)$
Lucky number	77,777	$1.01 (1 + 0.01 \sigma_F)$
Lucky payout	77,777,777	$1.01 (1 + 0.01 \sigma_F)$
Kitten angels	9,000	$1 + 0.1 \cdot \text{milk}$
Halo gloves	55,555	$1 + 0.1 s_{\text{click}}$
Wrinkly cookies	6,666,666	1.10
Distilled essence of redoubled luck	7,777,777	$1 + 0.01 \sigma$
Sugar baking	2×10^8	$1 + 0.01 \min(\text{unspent lumps}, 100)$
Sugar crystal cookies	10^9	1.05
Divine discount	99,999	1.01 (estimate)
Divine sales	99,999	1.005 (estimate)
Divine bakeries	399,999	1.03 (estimate)
Synergies Vol. I / Vol. II	222,222 / 2,222,222	1.05 each (estimate)
Cookie boxes, four early / three late	25 / 3.3×10^{11}	1.02 each (estimate)
Persistent memory	500	1.01 (estimate)
Starter kit / Starter kitchen	50 / 5,000	1.005 each (estimate)

Table 3: Heavenly upgrades valued by the ratio rule; s_{click} is the share of income from clicks. Chips are list prices; parents are added when blocks are formed. Cosmetics, switches, offline production, the dragon, wrinklers, seasons, fortunes and permanent upgrade slots are bought only as parents of valued upgrades, unless Section 7 values them.

3. Parent constraints are handled by blocks. An upgrade and its unowned ancestors form one block, with multiplier $\prod m$ and cost $\sum p$. The interchange argument applies to adjacent blocks without change. This is the idea of Sidney’s decomposition for scheduling with precedence constraints [6, 5]. We restrict blocks to the ancestor closures of single upgrades. This is exact without constraints (a unit test checks it against brute force on random instances) and a heuristic in general.

Table 3 lists the multipliers. They follow the in-game descriptions where the effect on production is direct. Estimates are marked where the effect works through cheaper or newly unlocked store upgrades.

6 When to ascend

6.1 Time rescaling

Lemma 6.1 (Time rescaling). *Assume A1, A3 and A4. If a run at speed 1 bakes $F(x)$ cookies in the time x , then the same run at speed S bakes $F(St)$ cookies in the time t .*

Proof. Let $q(t)$ be the holdings and $r(q)$ the base income. At speed S the bank obeys $\dot{b} = S r(q)$ between purchases. By A4, purchase decisions depend only on (b, q) . In the time variable $\tilde{t} = St$ the dynamics are $db/d\tilde{t} = r(q)$, with the same decision rule, independent of S . Hence the path $(b, q)(\tilde{t})$ is the same for every S , and the cookies baked, $\int r d\tilde{t}$, equal $F(\tilde{t})$. $\square$

Ranking purchases by payback period satisfies A4, since every PP_i scales by $1/S$. Golden cookies satisfy it only approximately, because their spawn clock runs in real time. So does the “Lucky!” reserve, which scales with S .

6.2 Formulation

Run k starts with the all-time cookies A_k and the speed $S_k = S(P(A_k))$. If it lasts T_k , it bakes $C_k = F(S_k T_k)$ cookies by Lemma 6.1, and $A_{k+1} = A_k + C_k$. The player chooses the run lengths to minimise the total time $\sum_k (T_k + \tau)$ needed to reach a distant goal A^* . This is an optimal stopping problem with the Bellman equation [1]

$$V(A) = \min_{A' > A} [T(A \rightarrow A') + \tau + V(A')], \quad V(A^*) = 0. \quad (6.1)$$

6.3 Late game: a closed form

In the late game $M(P) \approx 0.01 hP$, so $S(P) = \kappa P$. Take $\tau = 0$ and assume A5. Let $y = \ln P$. A run whose log-prestige grows by Δ must bake $C = 10^{12} P^3 (e^{3\Delta} - 1)$ cookies. At speed κP this takes

$$T(y, \Delta) = \frac{(C/\alpha)^{1/\beta}}{\kappa P} = K e^{\gamma y} \varphi(\Delta), \quad (6.2)$$

where

$$\varphi(\Delta) = (e^{3\Delta} - 1)^{1/\beta}, \quad \gamma = \frac{3}{\beta} - 1, \quad K = \frac{(10^{12}/\alpha)^{1/\beta}}{\kappa}. \quad (6.3)$$

When $\gamma > 0$ ($\beta < 3$), runs become longer as prestige grows. When $\gamma < 0$, they become shorter.

Theorem 6.2 (Lower bound and optimal step). *Let $G(\Delta) = (e^{\gamma\Delta} - 1)/\gamma$, with $G(\Delta) = \Delta$ if $\gamma = 0$, and let*

$$w = \inf_{\Delta > 0} \frac{K\varphi(\Delta)}{G(\Delta)}.$$

Every ascension schedule from y_0 to y^* satisfies

$$\sum_k T_k \geq \frac{w}{\gamma} (e^{\gamma y^*} - e^{\gamma y_0}) \quad (\text{or } w(y^* - y_0) \text{ if } \gamma = 0), \quad (6.4)$$

with equality if and only if every step Δ_k minimises φ/G . For $\beta > 1$ a minimiser Δ^* exists.

Proof. Bound. For $\gamma \neq 0$ and $\Delta > 0$, $G(\Delta) > 0$ and

$$e^{\gamma y} G(\Delta) = \frac{e^{\gamma(y+\Delta)} - e^{\gamma y}}{\gamma}.$$

By the definition of w , $K\varphi(\Delta) \geq w G(\Delta)$ for every $\Delta > 0$. With $y_{k+1} = y_k + \Delta_k$ the sum telescopes:

$$\sum_k T_k = \sum_k K e^{\gamma y_k} \varphi(\Delta_k) \geq w \sum_k e^{\gamma y_k} G(\Delta_k) = \frac{w}{\gamma} \sum_k (e^{\gamma y_{k+1}} - e^{\gamma y_k}) = \frac{w}{\gamma} (e^{\gamma y^*} - e^{\gamma y_0}).$$

The only inequality is an equality exactly when $K\varphi(\Delta_k) = w G(\Delta_k)$ for every k . The case $\gamma = 0$ is the same with $G(\Delta) = \Delta$.

Existence. As $\Delta \rightarrow 0^+$,

$$\frac{\varphi(\Delta)}{G(\Delta)} \sim 3^{1/\beta} \Delta^{1/\beta-1} \rightarrow \infty,$$

because $\beta > 1$. As $\Delta \rightarrow \infty$, $\varphi(\Delta) \sim e^{3\Delta/\beta}$, while G grows at most like $e^{\gamma\Delta}/\gamma$ with $\gamma < 3/\beta$. The ratio again tends to infinity. A continuous function on $(0, \infty)$ that tends to infinity at both ends attains its minimum. $\square$

β	γ	v^* (6.6)	prestige $\times$ per run	naive MVT v	brute force	dynamic program
1.5	1.000	4.24	1.62	2.40	4.42	4.39
2.0	0.500	7.87	1.99	4.92	7.95	7.18
2.5	0.200	12.03	2.29	9.31	11.49	11.50
3.0	0.000	16.80	2.56	16.80	15.95	16.69
4.0	-0.250	28.39	3.05	50.44	31.69	28.95
6.0	-0.500	60.91	3.93	397.38	63.59	60.13

Table 4: Optimal factor v^* by which each run multiplies all-time cookies in the late game. “Brute force” scans fixed factors to grow prestige a millionfold; the last, partial run causes the scatter. “Dynamic program” is the planner of Section 6.5, averaged over runs that have baked 0.1, 0.5 and 2 times the previous all-time cookies.

Repeating the step Δ^* achieves the bound, up to the rounding of the last step before the goal. For $\beta \leq 1$ the ratio φ/G has no interior minimum, and resets should be as frequent as the overhead allows. This degenerate case motivates a safeguard in the implementation (Section 6.5).

Corollary 6.3 (Stopping rule). *At an interior minimiser, $\varphi'/\varphi = G'/G$. The current run’s length is proportional to $\varphi(\Delta)$, so the optimal moment to ascend is when the marginal log-prestige rate of the run falls to*

$$\frac{d\Delta}{dT} = \frac{1 - e^{-\gamma\Delta}}{\gamma T}. \quad (6.5)$$

For a run whose cookies grow like T^β , $\Delta = \frac{1}{3} \ln v$ with $v = (A + C)/A$. Ascension is optimal when all-time cookies have grown by the factor v^* that solves

$$\frac{\beta}{3} \left(1 - \frac{1}{v}\right) = \frac{1 - v^{-\gamma/3}}{\gamma}. \quad (6.6)$$

Proof. From $T'(\Delta)/T(\Delta) = \varphi'/\varphi = G'/G$,

$$\frac{d\Delta}{dT} = \frac{G}{G'T} = \frac{e^{\gamma\Delta} - 1}{\gamma e^{\gamma\Delta} T}.$$

With $C \propto T^\beta$, $dv/dT = \beta(v - 1)/T$, so $d\Delta/dT = \frac{1}{3v} dv/dT = \frac{\beta}{3}(1 - 1/v)/T$. Equating the two expressions gives (6.6). $\square$

Remark 6.4 (The marginal value theorem). For $\gamma = 0$ ($\beta = 3$) the rule reads $d\Delta/dT = \Delta/T$: ascend when the marginal rate equals the average rate. This is the marginal value theorem of optimal foraging [2]. Its use at $\beta \neq 3$ ignores that later runs become longer or shorter. It then solves

$$\beta(v - 1) = v \ln v$$

instead of (6.6). The error is a factor of 1.6 at $\beta = 2$ and 6.5 at $\beta = 6$ (Table 4).

Remark 6.5 (Overhead). With $\tau > 0$ the problem is no longer self-similar. Corollary 6.3 then holds approximately, with T replaced by $T + \tau$. Dynamic programming solves the general problem exactly (Section 6.5).

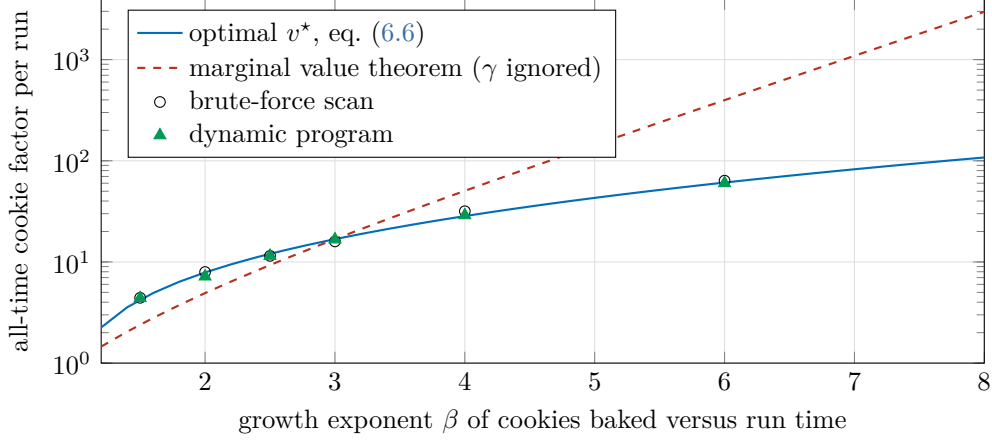


Figure 1: The late-game optimum and the uncorrected marginal value theorem. The two agree only at $\beta = 3$.

6.4 Early game: why early resets fail

Proposition 6.6 (No prestige effect, no reset). *Suppose that every run has the same speed, $S(P) \equiv S$, and that $F(x) = \alpha x^\beta$ with $\beta > 1$. Then for every $\tau \geq 0$, a single run is the fastest way to bake a given number of cookies.*

Proof. Split D cookies into runs of $D_1, \dots, D_n$. The total time is

$$\sum_k \frac{(D_k/\alpha)^{1/\beta}}{S} + (n-1)\tau.$$

Since $q = 1/\beta \in (0, 1)$, the map $x \mapsto x^q$ is strictly subadditive on $[0, \infty)$: $(\sum_k D_k)^q < \sum_k D_k^q$ for $n \geq 2$ positive parts. Hence one run is strictly fastest. $\square$

Early in the game, ascension raises the speed only slightly, so the problem is close to this case and early resets are costly. An earlier version of the advisor applied the late-game rule to a first run of 4.89×10^{12} cookies: prestige 1.70, one level, +1% production. It recommended an ascension. The planner below says to keep baking. It puts the cost of an ascension at that point at 11 to 16 hours of total progress (Table 5).

6.5 The general planner

Speed model. A run that starts at prestige P has the speed

$$S(P) = \left(\prod_{j \in J(\lfloor P \rfloor)} m_j \right) (1 + 0.01 h \lfloor P \rfloor). \quad (6.7)$$

Legacy is bought with the first chip. $J(\lfloor P \rfloor)$ is the longest prefix of the order of Theorem 5.1, parents included, that the chips available at P pay for.

Run-time model. The current run has baked C cookies in the time T , and $\beta = cT/C$. A fresh run rebuilds to C convexly and then grows like the current run:

$$t(X) = \begin{cases} T (X/C)^{1/\beta_r}, & X \leq C, \quad \beta_r = \max(\beta, 1.5), \\ T (X/C)^{1/\beta}, & X > C. \end{cases} \quad (6.8)$$

A run at speed S needs $t(X) S_{\text{now}}/S$ to bake X cookies (Lemma 6.1). The floor $\beta_r \geq 1.5$ states that every run starts from zero production, even when the current run has stalled.

β	ascend now?	optimal target	forecast (growth $\pm 25\%$) / cost of ascending now
1.5	no	prestige 4 (+14.4% speed)	50.6–129.4 h / 11.4 h
2.0	no	prestige 4 (+14.4% speed)	30.9–68.6 h / 14.2 h
2.5	no	prestige 4 (+14.4% speed)	22.0–45.0 h / 15.2 h
3.0	no	prestige 4 (+14.4% speed)	16.9–33.0 h / 15.8 h
4.0	no	prestige 4 (+14.4% speed)	11.6–21.2 h / 16.3 h

Table 5: The planner on a first run of 4.89×10^{12} cookies in 17.2 hours without Legacy. Prestige 4 pays for Legacy and *Heavenly cookies* together, so the first ascension buys several heavenly upgrades at once.

Proposition 6.7 (Exactness of the dynamic program). *Restrict ascensions to a finite grid of all-time cookie levels $a_0 < a_1 < \dots < a_n = A^*$, and let $t(i, j)$ be the duration of a run from a_i to a_j . Then*

$$V(i) = \min_{j > i} [t(i, j) + \tau \mathbf{1}[j < n] + V(j)], \quad V(n) = 0, \quad (6.9)$$

computed for $i = n - 1, \dots, 0$ in $O(n^2)$ time, is the minimum total time from a_i . The minimising j is the optimal next ascension point. For the current run, the optimal target minimises

$$(t(a_j - A) - T) + \tau + V(j).$$

Proof. All-time cookies only increase, so schedules are paths in a directed acyclic graph on the grid, with the edge weights $t(i, j) + \tau$. The recursion is Bellman’s equation (6.1) on this graph. By induction on $n - i$, $V(i)$ is the shortest path length from i to n . $\square$

Four implementation choices keep the planner robust.

- *Horizon.* The goal is the level that the current growth would reach in 10^4 run lengths. A more distant goal would let floating-point error swamp the hour-scale differences of the current decision.
- *End effects.* The target is the median over five goals spaced over one decade of all-time cookies. This cancels the dependence on how the last run fits before the goal.
- *Minimum gain.* An ascension must raise the speed by at least 10%. This guards against a stalled run, for example while nothing is bought. Such a run pushes β toward 1, where Theorem 6.2 favours constant resets.
- *Grid and age.* The grid has 400 log-spaced levels plus every whole prestige level below 400. No plan is made before a run is ten minutes old.

7 The rest of the game

The rules above settle purchases, golden cookies and ascension. The game has further systems, and most of them reset at every ascension. Each system has a rule that uses the long-run income model of Definition 4.4, extended below to wrath cookies and wrinklers. This section states the facts that the rules rest on.

7.1 Achievements

Proposition 7.1 (Value of an achievement). *Each achievement, shadow achievements aside, raises the milk μ by 0.04. Let the kitten upgrades have the factors k_i , and let m be the milk multiplier. CpS carries the factor $\prod_i (1 + k_i m \mu)$, so one more achievement multiplies CpS by*

$$\prod_i \frac{1 + k_i m (\mu + 0.04)}{1 + k_i m \mu} \approx \exp\left(0.04 \sum_i \frac{k_i m}{1 + k_i m \mu}\right). \quad (7.1)$$

Proof. The game multiplies CpS by $1 + k_i m \mu$ for each kitten owned. Substitute $\mu + 0.04$ and take the ratio. The approximation is $\ln(1 + x) \approx x$ for each factor. $\square$

On an observed save with eight kittens and $\mu = 9.52$, one achievement adds 2.14% of CpS. Achievements that only take an action are therefore taken at once. Examples are 50 clicks on the news ticker, a golden cookie clicked in its last second, a wrinkler poked 50 times, one of Santa’s helper grandmas, a ticker click in a window shrunk below 900 pixels, and a gift of one cookie sent to oneself. Actions with a cost are priced in time.

Proposition 7.2 (What an achievement is worth in time). *Suppose, as in Lemma 6.1, that every future run follows its path at a speed proportional to income. A permanent gain of a share x of income is then worth*

$$\frac{xH}{1+x} \approx xH$$

seconds of progress over the remaining play time H . A pause of D seconds costs D , and so do C cookies spent at income I , with $D = C/I$. An action that earns achievements worth $x_1, \dots, x_k$ (Proposition 7.1) at the delay D pays if and only if

$$D < H \sum_i x_i. \quad (7.2)$$

Proof. With every production rate multiplied by $1 + x$, the state reached at the wall time t is the one that the unboosted game reaches at $(1 + x)t$. At the horizon, the boosted game is xH seconds ahead. Equivalently, the remaining time to any fixed goal shrinks from H to $H/(1 + x)$. A pause shifts the whole path by D . $\square$

The horizon is unknown. With no other information, a scale-free prior on the total length of play gives a median remaining time equal to the time played so far. Autoplay therefore takes H to be the age of the save. On the observed save, after 77 hours, one achievement is worth 1.66 hours. Table 6 applies (7.2) to three kinds of action.

A Born again run starts from nothing: no prestige, heavenly upgrades, sugar lumps or minigames. A simulation of such a run (buildings only, the 15 permitted clicks spent on the first cursor, golden cookies clicked) gives the times in Table 6. One run earns both achievements. In the run, autoplay clicks the big cookie exactly 15 times, buys only buildings and ascends back into a normal run as soon as both achievements are won.

Cheap spells would roughly double the number of casts, because magic regenerates like $\sqrt{m}$ (Proposition 7.7). They would reach 99 casts in about 22 hours instead of 41.

7.2 Godzamok

Godzamok, one of the Pantheon’s gods, rewards the sale of buildings. Each building sold adds $b = 1\%$ to the click power of a ten-second “Devastation” buff (0.5% in the second slot, 0.25% in the third). Sales while the buff runs add to it without limit, but they do not extend it.

Action	Achievements	Cost	Decision
Born again run	Neverclick, Hardcore	a new run of about 4 h: 10^6 cookies in 41 min, 10^9 in 3.8 h	start when $2xH > 4$ h, about 93 h of play
Bursting spree	Wrinklesquisher, Moistburster	50 or 200 bursts in one run, $N\tau/n_{\max}$: 15 or 62 min at reduced income	burst when the loss is less than xH and the run lasts long enough
Cheap spells	spell counts (9, 99)	golden cookies worth about 1.5 h	keep casting the Hand of Fate: the gain is about 0.5 h

Table 6: Actions priced by (7.2). In the spree, each of the $n_{\max} = 12$ wrinkler slots refills after $\tau = 222$ s on average (stage 3 with Unholy bait), and income falls to that of wrath without wrinklers.

Proposition 7.3 (Godzamok round trips). *Sell all n copies of a building whose next copy costs p , and buy them back in the same frame. Production does not change, and the round trip costs*

$$(1 - 1.15\sigma) \frac{p(1 - 1.15^{-n})}{0.15}, \quad \sigma = 0.25, \quad (7.3)$$

which is 0.7125 times the cost to buy the copies back. The round trip adds bn to the multiplier of the buff, which is worth $bnKt$ cookies. Here K is the click income without the buff and t is the remaining buff time. If U buildings are cycled in each of the $F = 300$ frames of a buff, the click multiplier averages

$$1 + bU \frac{F+1}{2} \approx 1 + 150bU. \quad (7.4)$$

Proof. A sale at the amount j refunds σ times the price of copy $j+1$, which is 1.15 times the price of copy j . Summed over the top n copies, the refund is 1.15σ times the buy-back cost, a geometric sum. Click power is linear in the multiplier of the buff, so a sale adds bnK per second for the remaining t seconds. Cycling U buildings per frame makes the multiplier $1 + bUk$ in frame k . Average over $k = 1, \dots, F$. $\square$

Autoplay cycles, in every frame and within three milliseconds, each building type whose round trip returns at least twice its cost. It never sells grandmas, because the sale of the last grandma ends the Elder Pledge. Late in a run this covers every type except the newest tiers, whose copies cost more than they add.

The mouse upgrades make a click worth 15% of CpS, so at 20 clicks a second $K \approx 3c$ before the buff. On the observed save, about 2,200 buildings per frame kept the multiplier at 3,329 on average. Income rose to about 22,000 times CpS, from a few times CpS before. The game kept 30 frames a second.

The economy model includes the average multiplier (7.4) in click income, so the Pantheon slots Godzamok as soon as a worship swap allows. Each ascension restores three swaps. Once they are spent, the next swap takes 16 hours. When a much better arrangement waits for swaps, autoplay keeps the next sugar lump for the refill that restores all three.

Stage	Frenzy	“Lucky!”	Special	Click fr.	Storm	Chain	Elder fr.	Clot	Ruin	Cursed
0	.422	.422	.100	.040	.008	.008	—	—	—	—
1	.294	.341	.076	.032	.024	.023	.017	.090	.090	.008
2	.157	.280	.056	.027	.039	.039	.034	.171	.171	.015
3	—	.245	.041	.023	.055	.055	.051	.245	.244	.023

Table 7: Golden-cookie outcome shares by Grandmapocalypse stage, with at least ten buildings and no dragon auras. Building debuffs, 1.8% at stage 3, are omitted.

Outcome	Value	Outcome	Value
Frenzy	$462(Wc + k)$	Click frenzy	$10088k$
Building special	$3\bar{n}(Wc + k)$	Cookie storm	$25200c$
Elder frenzy	$3990(Wc + k)$	“Lucky!”	$900c$
Clot	$-33(Wc + k)$	Cookie chain	$1290c$
Cursed finger	$2000c - 10(Wc + k)$	Ruin	$-300c$

Table 8: Value of each outcome in cookies, per unit of effect duration, with a full “Lucky!” bank. Buffs count W times because wrinklers digest buffed CpS; payouts count once.

7.3 Wrinklers and wrath cookies

Proposition 7.4 (Wrinkler factor). *With n wrinklers digesting, a share $s = \min(1, n/20)$ of the CpS c withers, and every wrinkler accumulates sc per second. A popped wrinkler returns $r \geq 1.1$ times its contents. Income from CpS is therefore $cW(n)$, with*

$$W(n) = 1 - s + snr, \quad W(10) = 6, \quad W(12) = 8.32 \quad (r = 1.1). \quad (7.5)$$

Proof. In the game’s update, each digesting wrinkler adds cs per second to its stomach: the whole withered share, not $1/n$ of it. The bank receives $(1 - s)c$. Sum over the n wrinklers and apply the return factor. $\square$

Wrinklers appear only during the Grandmapocalypse. At stage $\ell \in \{1, 2, 3\}$, each golden cookie is a wrath cookie with probability $\ell/3$. Wrath cookies replace Frenzies with clots, which halve production for 66s, and ruin, which takes $\min(0.05b, 600c)$. In return they bring Elder frenzies (production $\times 666$ for 6s), cursed fingers (each click worth 10s of production for 10s) and cookie storms: about 105 golden cookies in 7s, each worth one to seven minutes of CpS. Table 7 gives the outcome shares, simulated from the game’s choice rule.

Proposition 7.5 (Wrath cookies win under automation). *Let the click income be $k = 0.6c$, and let building specials be worth 588s. With a full “Lucky!” bank, an average golden cookie is worth 1248s of CpS at stage 0 and 2141s at stage 3 without wrinklers. With ten wrinklers, the stage-3 cookie is worth 3236s. Total income at stage 3 is about 2.8 times that of stage 0.*

Proof. Multiply the shares of Table 7 by the values of Table 8 and sum. $\square$

Autoplay therefore researches up to the Elder Pact. It buys the research that the game guards with a confirmation prompt. Godzamok changes this verdict.

Proposition 7.6 (Calm wins once clicks dominate). *With the Elder Pact, a pledge costs at most $8^{14} \approx 4.4 \times 10^{12}$ cookies and returns by itself when it runs out. The stage is thus a free choice between calm (stage 0, no wrinklers) and wrath (stage 3, $W(n_{\max})$). Take $d = 2$, a*

mean of $\bar{n} = 200$ copies per building type, $\mathbb{E}[I] = 121$ s and twelve wrinklers. Golden cookies then multiply click income by

$$1 + \frac{(595 + 0.30\bar{n})d}{\mathbb{E}[I]} = 11.8 \text{ (calm)}, \quad 1 + \frac{(428 + 0.12\bar{n})d}{\mathbb{E}[I]} = 8.5 \text{ (wrath)}, \quad (7.6)$$

and CpS income by 8.7 (calm) and 62.8 (wrath). Calm earns more if and only if

$$\frac{k}{c} > \frac{62.8 - 8.7}{11.8 - 8.5} \approx 16. \quad (7.7)$$

Proof. For clicks, Frenzies, building specials and Click frenzies bring $(595 + 0.30\bar{n})d$ buff-seconds per golden cookie in calm. In wrath, Elder frenzies and Click frenzies, less clots, cursed fingers and debuffs, bring $(428 + 0.12\bar{n})d$. The shares are those of Table 7. Divide by $\mathbb{E}[I]$ and add one. For CpS, the same buffs count W times, and storms and cursed fingers pay directly. Compare $8.7c + 11.8k$ with $62.8c + 8.5k$. $\square$

Without Godzamok, clicks bring about $3c$ and wrath wins. With him, clicks bring thousands of times c , and calm earns about 40% more. Autoplay compares the two incomes with the economy model and renews the pledge whenever calm wins. It lets the wrath reach stage 3 once, for the Grandmapocalypse achievement. It then runs the elder achievements below and revokes the covenant that they leave, which costs 5% of CpS.

The bank threshold also changes with the stage, because ruin takes 5% of the bank. Per banked cookie,

$$\frac{1}{\theta} = \frac{0.15\pi_L g + 0.215\pi_{\text{chain}} g - 0.05\pi_{\text{ruin}}}{\mathbb{E}[I]}, \quad (7.8)$$

so θ rises from about 1.9×10^3 s at stage 0 to 3.3×10^3 s at stage 3.

When to pop. Wrinklers lose nothing by waiting, but their contents earn nothing until popped. Pop a wrinkler that holds X cookies now, T seconds before the ascension. Its cookies then grow to $X e^{\rho T}$ by the end, where $\rho = (\beta - 1)/t$ is the income growth rate of the run. Its slot stays empty for the respawn time τ , and meanwhile the other wrinklers each digest a twentieth less. The loss is

$$L = (0.05(2n - 1)r - 0.05)c\tau, \quad (7.9)$$

and it compounds to $L e^{\rho T}$. Autoplay pops the fattest normal wrinkler when

$$X > \frac{L e^{\rho T}}{e^{\rho T} - 1}. \quad (7.10)$$

Elder achievements. With the Elder Pact owned, the pledge returns as soon as a signed covenant is revoked. Pledge, covenant and revoke, repeated five times, earn all three elder achievements in seconds for a negligible amount of cookies, instead of two and a half hours of peace.

7.4 Minigames

Proposition 7.7 (Cast at full magic). *Below its cap M , magic regenerates at the rate $\kappa\sqrt{m}$. A spell that costs $C \leq M$ is cast most often when it is cast whenever magic reaches M .*

Proof. A refill from a to $a + C$ takes

$$\int_a^{a+C} \frac{dm}{\kappa\sqrt{m}} = \frac{2}{\kappa}(\sqrt{a+C} - \sqrt{a}),$$

which decreases in a . Every cycle of a periodic policy refills some interval of length C , and $a = M - C$ is the largest possible start. $\square$

With $M \approx 72$ and Force the Hand of Fate at $C \approx 53$, a cast at full magic gives one cast every 23 minutes instead of every 40.

Pantheon. Three slots take three of eleven gods. Autoplay evaluates every ordered triple of the useful gods with the game's CpS formula and the income model. The golden-cookie penalties of Mokalsium and Jeremy enter through the interval. Autoplay moves gods only when income rises by more than 1% and worship swaps are available.

Garden. New seeds come only from mutations, which grow in empty tiles next to mature plants of the right kinds.

Lemma 7.8 (Checkerboard breeding). *On a rectangular plot of at least 2×2 tiles, plant the parent kind A on the tiles with $x + y$ even and x even, and the kind B on the tiles with $x + y$ even and x odd. Leave the tiles with $x + y$ odd empty. Then every empty tile touches both kinds (two parents of kind A when $A = B$), so every empty tile can mutate into a seed bred from A and B .*

Proof. An empty tile (x, y) has $x + y$ odd. Its horizontal neighbours $(x \pm 1, y)$ are parent tiles with the parity of $x + 1$, and its vertical neighbours $(x, y \pm 1)$ are parent tiles with the parity of x . In a plot at least two tiles wide and tall, at least one tile of each pair lies inside the plot. $\square$

Autoplay breeds along the chains whose parents can be planted: Thumbcorn from wheat, then Cronerice, Gildmillet, Clover, Shimmerlily, Elderwort, Golden clover and Bakeberry. It uproots whatever else sprouts in the empty tiles. It lets each new kind grow up and picks it when mature, which unlocks its seed. Wood chips (three mutation rolls per tick, from 300 farms), or else fertilizer (a tick every three minutes instead of five), speed this up.

Thumbcorn, at +2% click power, is worth 2% of nearly all income once Godzamok multiplies clicks. Golden clover, at +3% golden-cookie frequency, is worth 3% of the golden-cookie income. The plot is cleared at every ascension, so breeding advances only in runs long enough for the parents to mature.

With nothing left to breed, autoplay plants a seed j on clay (effects $\times 1.25$). Let the seed add v_j to long-run income over the mature lifetime T_j , for the price p_j . It is planted when

$$v_j T_j > p_j \quad \text{and} \quad \frac{p_j}{v_j} < \text{payback of the next purchase.} \quad (7.11)$$

Plants are harvested while mature, just before the growth tick that could kill them, which counts toward the harvest achievements. One tile stays empty for mutations.

Stock Market. Prices revert toward the resting value $R_i = 10 + 10i + (\ell_{\text{bank}} - 1)$. A purchase costs at most 20% extra and a sale costs nothing. A purchase below $0.6 R_i$ and a sale at R_i therefore return at least

$$\frac{1}{0.6 \cdot 1.2} - 1 = 39\% \quad (7.12)$$

per completed round trip. The rule uses only the prices that the market shows.

Offices and loans. Office upgrades sacrifice 100 to 700 cursors and need cursor levels 2, 4, 8, 10 and 12. They open the loans of Table 9. Loan i multiplies CpS by m_i for the time t_i , and then by $m'_i < 1$ for t'_i , against a down payment of a share f_i of the bank.

A run that ends before t_i never pays the interest, and the ascension also earns Debt evasion. Autoplay takes a loan once the planner's slow estimate of the time left is at most $0.9 t_i$. It ends the run early when a loan is about to run out and

$$\text{cost of ascending now} < (1 - m'_i) \min(t'_i, \text{time left}). \quad (7.13)$$

Loan	m_i	t_i	m'_i	t'_i	f_i
1	1.5	2 h	0.25	4 h	20%
2	2	40 s	0.1	40 min	40%
3	1.2	48 h	0.8	120 h	50%

Table 9: Stock Market loans: the boost, its length, the interest that follows, its length and the down payment.

Right before an ascension, the bank is about to be lost, so autoplay also takes any open loan then.

7.5 The dragon, seasons, sugar lumps and permanent slots

Dragon. Sacrificing n copies of a building whose next copy costs p costs

$$p \frac{1 - 1.15^{-n}}{0.15} \approx 6.67 p \quad (n = 100) \quad (7.14)$$

to buy back. The dragon starts over every run. Radiant Appetite doubles production and needs the sacrifice of 100 copies of each of the first 14 buildings. Training toward it goes ahead once all those buildings are at hand and buying them back costs at most an hour of income. The same rule holds for Dragon Guts, which adds two wrinklers that digest and return 20% more, so that W reaches about 11.7. Auras are chosen by evaluation, and a change costs one top building.

Seasons. Christmas brings the 14 levels of Santa, which cost

$$\sum_{j=1}^{14} j^j \approx 1.2 \times 10^{16} \quad (7.15)$$

cookies in total, with a gift at each level (Santa's legacy gives +3% per level). It also brings reindeer, each worth one to two minutes of CpS. Valentine's day brings seven heart biscuits, +2% each, and Easter brings eggs. Each run visits Christmas until Santa is complete, Valentine's day once the hearts cost a minute of production, and Easter until seven eggs while that achievement is missing. It otherwise stays in Christmas. The auto-buyer keeps up to a minute of production for the next seasonal action.

Sugar lumps. Lumps go first to the minigames and then to the Sugar frenzy of each run, once Sugar craving is owned: production $\times 3$ for an hour, two hours of income for one lump. Next come cursor levels until the first loan opens, 11 lumps counting the bank's own level, for +50% CpS in the last two hours of every run. The rest go to building levels. A level of building b adds 1% of its CpS, a share

$$\frac{0.01 s_b}{1 + 0.01 \ell_b} \quad (7.16)$$

of CpS, for $\ell_b + 1$ lumps, plus the value (7.1) at level 10. Levels are bought by gain per lump, saving up for the best one as in Remark 5.2. The gamble on a mature but unripe lump (Hand-picked), which yields nothing half the time, is taken once the minigames are open.

Permanent slots. Milk carries over, so a kitten kept in a slot multiplies CpS by $1 + km\mu$ from the first second of each run, and a cookie upgrade by 1 plus its power. The slots take last run's upgrades in decreasing order of the logarithm of that multiplier. These systems also enter

Speed, target	play	payback period	exchange index	with look-ahead
$S = 1, 10^{15}$	continuous	68.73 h	0.00%	+0.03%
	every 10 min	70.33 h	-0.17%	-0.09%
	every 60 min	77.39 h	-0.06%	-0.08%
$S = 20, 10^{20}$	continuous	36.58 h	0.00%	+0.09%
	every 10 min	38.62 h	-0.02%	+0.01%
	every 60 min	46.65 h	-0.97%	-1.68%
$S = 200, 10^{25}$	continuous	627.81 h	0.00%	+0.35%
	every 10 min	629.95 h	0.00%	+0.35%
	every 60 min	638.06 h	-0.12%	+0.19%

Table 10: Time to bake a target number of cookies. The last two columns give the speed-up relative to the payback period; negative means slower.

Scenario	planner (DP)	stopping rule	naive MVT	best fixed v	worst fixed v
$K = 2500, A_0 = 10^{15}$	93.5	98.2	93.9	79.3 ($v = 60$)	13.4 ($v = 1.5$)
$K = 2500, A_0 = 10^{24}$	98.5	98.7	93.9	98.7 ($v = 8$)	79.1 ($v = 1.5$)
$K = 500, A_0 = 10^{15}$	98.8	98.0	90.8	84.5 ($v = 12$)	20.5 ($v = 1.5$)
$K = 500, A_0 = 10^{24}$	98.2	98.7	94.0	97.9 ($v = 12$)	68.2 ($v = 500$)
$K = 100, A_0 = 10^{15}$	98.1	98.9	91.1	92.8 ($v = 12$)	34.1 ($v = 1.5$)
$K = 100, A_0 = 10^{24}$	98.3	99.3	91.6	98.9 ($v = 8$)	45.6 ($v = 500$)
Mean	97.6	98.6	92.6	92.0	—

Table 11: Prestige reached as a percentage of the best rule at each horizon, averaged over five horizons from 60 to 960 days. K is the milk parameter and A_0 the all-time cookies at the start. The “best fixed” rule is chosen in hindsight per scenario; no single v is best everywhere.

the heavenly-upgrade values, as estimates: How to bake your dragon 1.8, Season switcher 1.3, permanent slots 1.12 down to 1.03, Elder spice 1.25, Unholy bait 1.05, Sacrilegious corruption 1.04 and Wrapping paper 1.02.

8 Numerical validation

Simulator. The simulator uses the game’s formulas for the base price and production of all 21 buildings. It adds tiered upgrades that double a building at 1, 5, 25, 50, 100, ... copies, global +10% upgrades whose prices grow twentyfold, a milk proxy $1 + n/K$ for n buildings, and clicks at 5 cookies per second. Purchases are greedy by the exchange index. Ascension rules are compared by the prestige reached after a fixed play time, as a percentage of the best rule for that horizon, averaged over horizons of 60 to 960 days.

Purchase order. Table 10 compares the payback period with the bank-independent index of Theorem 3.1, and with a look-ahead that values a building together with the tier upgrade that it unlocks. The differences are negligible in continuous play. In infrequent play, where large banks build up, the payback period is slightly better.

Ascension timing. Table 11 compares the dynamic-programming planner, the stopping rule of Corollary 6.3 with β measured live, the uncorrected marginal value theorem, and fixed rules that ascend when all-time cookies have grown v -fold, for v from 1.5 to 500.

Both the planner and the stopping rule beat the uncorrected marginal value theorem in every scenario, and the hindsight-tuned fixed rule on average. In these late-game scenarios the simpler stopping rule is slightly better: 98.6% against 97.6%.

Rule	30 days	120 days	prestige at each ascension (120 days)
Planner (DP)	658.3	18,512.5	4, 206, 815, 1759, 3789, 7321, 13092, 18512
$P_{\min} = 150, v = 20$ (best fixed)	421.2	17,939.4	150, 407, 1105, 3000, 8143, 17939
$P_{\min} = 150, v = 8$	359.7	17,686.5	150, 300, 600, ..., 17686
$P_{\min} = 50, v = 20$	368.8	17,137.6	50, 135, 368, 1000, ...
$P_{\min} = 4, v = 3$	68.1	6,812.0	4, 5, 8, 12, 17, ...

Table 12: Final prestige from a fresh game. Twelve fixed rules were tested; the best and three others are shown.

The gap comes from extrapolation. The planner projects the current growth exponent forward, but simulated runs keep decelerating, so a continuation looks slightly cheaper than it is. In the first scenario, the planner’s first two runs last 1033 and 917 hours, against 879 and 833 for the stopping rule. This costs about five points. A run-curve model that anticipates deceleration is the natural fix. The stopping rule, however, assumes $S \propto P$. The planner is the policy that remains valid from a fresh game.

From a fresh game. Table 12 starts with no prestige, so Legacy, *Heavenly cookies* and the prestige power all enter through the speed model. Fixed rules ascend once prestige is at least $P_{\min}$ and all-time cookies have grown v -fold.

9 Implementation

The rules run in Cookie Monster+ [3], a fork of the Cookie Monster mod [4] for the Steam version of the game (2.053). Every second the mod records the ascension inputs. Every five seconds it solves the dynamic program again, which takes 10 to 60 ms. In testing, no main-thread task exceeded 50 ms. With automation on, the mod also does the following.

- It clicks the big cookie 20 times per second. The game ignores clicks less than 20 ms apart.
- It clicks every golden cookie and reindeer as soon as it appears.
- In every frame with Godzamok slotted, it sells and buys back the building types of Proposition 7.3 within three milliseconds. The counts are restored before the frame ends, so only click power is computed again.
- Every nine frames, it buys the item with the smallest payback period, computed with the automation values of Section 4 and the “Lucky!” reserve of Proposition 4.1. It skips switches, sugar-lump costs and the Chocolate egg.
- On the frames between, it runs the systems of Section 7: the Grandmapocalypse (stage choice, sprees and wrinklers), the dragon, seasons, sugar lumps, the Pantheon, Grimoire, Garden and Stock Market, and the achievements that only take an action.
- Once the planner has said “ascend now” for ten seconds, it pops normal wrinklers, buys the Chocolate egg and ascends. It spends chips along the order of Theorem 5.1, parents first and saving up for the next item. It fills the permanent slots and reincarnates, into a Born again run when Proposition 7.2 says that the challenge pays. It ascends early when a loan is about to reach its interest.

The mathematical components have 124 unit tests. They include a reproduction of the closed form of Table 4 by the dynamic program, the refusal to ascend in the first-run case, and brute-force optimality of the chip order.

10 Limitations

- The run curve is extrapolated from one number, the current growth exponent. Real runs accelerate when new tiers of upgrades unlock, and slow down when they run out. In simulation, the extrapolation of a decelerating run made the planner too patient by about 15% in its first runs (Section 8).
- The heavenly multipliers marked as estimates in Table 3 do not come from the game’s code paths. Blocks restricted to ancestor closures are a heuristic for general precedence graphs.
- The golden-cookie probabilities π_F and π_L are approximate. Assumption A4 holds only approximately for golden cookies and the “Lucky!” reserve.
- The analysis is risk-neutral. The rules of Section 7 rest on the income model and on simulated outcome shares. Several heavenly values, the Garden’s seed values and the Stock Market thresholds are estimates.
- Garden breeding covers the chains whose parents can be planted. Fungi enter only after weeds leave their spores. Skruuia, the Dragonflight aura, brokers and spell counts are not pursued. The play horizon of Proposition 7.2 is a prior, not a measurement.
- The simulator is a stylised economy. Its agreement with the theory supports the internal consistency of the rules, not their exact optimality in the real game.

A Algorithms

Algorithm 1 Ascension planner (every five seconds)

- 1: measure A , C , the run time T and the income c ; set $\beta \leftarrow \min(\max(cT/C, 0.3), 12)$
 - 2: build the chip order of Theorem 5.1; define $S(P)$ by (6.7)
 - 3: **for** each of five goals a_k^* spaced over one decade beyond the 10^4 -run-length horizon **do**
 - 4: grid $\leftarrow$ 400 log-spaced levels in $[A + C, a_k^*] \cup$ whole prestige levels < 400
 - 5: solve V backwards by (6.9); choose the target j_k that minimises $\text{remaining}(j) + \tau + V(j)$
 - 6: consider only targets with a speed gain of at least 10%
 - 7: **end for**
 - 8: target $\leftarrow$ median of $j_1, \dots, j_5$; **ascend now** if and only if the target is the current level
-

Algorithm 2 Automation tick (every frame; clicks on a separate 50 ms timer)

```

1: click every golden cookie and reindeer on screen
2: if frame mod 9 = 0 and the store is in buy mode then
3:   for each candidate  $i$ : value  $\Delta_i$  by (4.5); reserve  $R_i \leftarrow$  the “Lucky!” bank if  $p_i/\bar{\Delta}_i > \theta$ ,
   else 0
4:    $i^* \leftarrow \arg \min_i \max(p_i - (b - R_i), 0)/(c + k + G) + p_i/\bar{\Delta}_i$ 
5:   buy  $i^*$  if  $p_{i^*} \leq b - R_{i^*}$ 
6: end if
7: if the planner has said “ascend now” for 10 s then
8:   pop normal wrinklers; buy the Chocolate egg; ascend; spend chips along the order;
   reincarnate
9: end if

```

References

- [1] R. Bellman, *Dynamic Programming*, Princeton University Press, 1957.
- [2] E. L. Charnov, *Optimal foraging, the marginal value theorem*, Theoret. Population Biol. **9** (1976), 129–136. doi:10.1016/0040-5809(76)90040-X.
- [3] Cookie Monster+, <https://github.com/JWKNT/cookie-clicker>, guide at <https://jehlp.net/cookie-clicker/>.
- [4] Cookie Monster Team, *Cookie Monster*, a Cookie Clicker add-on, <https://github.com/CookieMonsterTeam/CookieMonster>.
- [5] E. L. Lawler, *Sequencing jobs to minimize total weighted completion time subject to precedence constraints*, Ann. Discrete Math. **2** (1978), 75–90. doi:10.1016/S0167-5060(08)70323-6.
- [6] J. B. Sidney, *Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs*, Oper. Res. **23** (1975), 283–298. doi:10.1287/opre.23.2.283.
- [7] W. E. Smith, *Various optimizers for single-stage production*, Naval Res. Logist. Quart. **3** (1956), 59–66. doi:10.1002/nav.3800030106.
- [8] J. Thiennot (Orteil), *Cookie Clicker*, version 2.053, DashNet, 2013–2025. <https://orteil.dashnet.org/cookieclicker/>.