A Linear–Markov Model for Quality Upcycling in Factorio

Trevor Squires

December 17, 2025

1 Introduction

The date is 12-15-2025. I've just completed a 27 hour 'speedrun' of a game called Factorio bringing my total hours played to near 650 - 200 of which occuring in the past month. Having committed such a large percentage of recent my brain activity towards this game, I feel it is fitting to share some of those analytical inspirations that make the game so interesting for me. In this document, I will advance the community's resources by providing simple to understand relations that govern one of the game's more analytically challenging mechanics.

For those of you who aren't Factorio mega-fans, Factorio is a popular factory building game that is ripe with mathematical potential. The goal is to produce different materials, progress through a technology tree, and ultimately build an ever-growing factory of material automation. In my opinion, it is a must-have for engineers (particularly those in the software space). Much of the core principles of Factorio are shared with engineering such as: abstraction, system design, modularity, latency, debugging, and dozens of others. The main draw for me in particular is its system design aspects which themselves are often represented by mathematical systems. I've done countless (and often unnecessarily complicated) analyses of Factorio systems to optimize the tiniest amount of efficiency. And while it's true that the majority of that work is largely overkill, it often comes in the form of neat applications of deeper mathematical theory that you won't get elsewhere. For our discussion today, we will focus on one such area in particular that I haven't had a chance to do (or write about) from other games: a Markovian-like dynamical system defined by Factorio's upcycling routines.

2 Quality Mechanics in Factorio

The Desire for Quality

To understand the problems faced in the upcycling routine, we must first understand the quality mechanic in Factorio. As mentioned previously, Factorio is a game of building items. In a recent update, the developers added different quality versions of items. These higher quality items are better than the base version in different ways, but usually represent a significant - even necessary - improvement over the base item. For example, for items that have a "speed" mechanic, i.e. they do something in a certain amount of time, a top tier quality item will do this thing 2.5 times faster than the base version. This means if you were to utilize these higher quality items instead of regular ones, your base would be significantly smaller. Obtaining quality versions of every item is a challenge every large-factory-building Factorio veteran will tackle at some point.

Quality Upgrading

There are two ways of obtaining a quality version of an item. The first is simple: use quality ingredients whenever you craft the item. However simple, this is not usually the way most quality items are obtained. After all, we've only reduced the problem by one degree: we still have to get the quality ingredients. The more common (and complicated) way is to craft said item with a certain "quality percentage" for the craft. This quality percentage can be affected through the use of modules, but for now let's just assume that we don't have any control over this value that we call q. If an item of base quality is crafted with a quality percentage of q, then the likelihood that the output is 1 quality level higher is $\frac{9q}{10}$, 2 levels higher is $\frac{9q}{100}$ and in general for k levels higher it is $\frac{9q}{10^k}$. Let's see this in an example using 5 quality levels that exist in the game:

1. Normal
2. Uncommon
3. Rare
4. Epic
5. Legendary

If I craft an item using normal ingredient with $q = 30\%$ quality, then I will have a $1-q = 70\%$ chance for the resulting item to be of normal quality, but a 27% chance for the item to be uncommon, 2.7% chance for rare, and so forth. The sum of the probabilities must equal 1, so the remaining tail of the probability distribution gets tucked into the legendary percentage, but it is typically trivial for analysis sake. Had I crafted with rare materials instead, the probabilities remain the same, except with every quality level being raised by two stages.

Quality Upcycling

As described above, one can obtain quality goods from non-quality ones simply by crafting with an innate quality percentage. However, obtaining quality goods in this way can be quite obnoxious for a few reasons:

1. The quality percentage cannot go very high and is often a small number, leading to many crafts necessary to obtain a high quality item
2. All the failed attempts of quality crafting essentially go to waste. For expensive items, it is not easy to craft and waste so many of them

Thankfully, Factorio introduced a way to alleviate both of these concerns: recycling. Recycling is a mechanic that allows you to recycling a crafted item for 25% of the ingredients. So all of these 'failed upgrades' during the quality process can now be recycled and attempted again, though at a reduced efficiency rate. However, there is an additional benefit to recycling - items can be recycled with an innate quality percentage just like assembling/crafting. In this regard, the recycled ingredients of an item with quality recycling follow the same $1 - q$ probabilities described above, though they still suffer the 75% loss of materials.

These two mechanics together (assembling and recycling) can be combined in an iterative process usually referred to as upcycling. The process looks like the following:

1. Craft an item with a certain quality percentage with a target quality in mind

2. If the result is not the desired quality, recycle it down to ingredients again using quality percentage

3. If you have enough raw ingredients to craft a new item, craft said new item. If not, go back to step 1 and start again

In this upcycling process, resources are fully utilized to obtain the desired quality without any waste. The challenge, of course, is understanding how this upcycling system is going to behave. Because of the tricky quality upgrading mechanics and the fact that there is are two steps in the iteration make it non-trivial to analyze. Even worse, there are additional mechanics such as productivity that complicate the process. Though there are simple cases, I hope to do a full analysis of the upcycling dynamics and provide relations to completely understand the long-term system behavior of the non-trivial upcycling loop.

3 Modeling Components

The purpose of this section is to define a minimal mathematical language that mirrors the mechanics of quality upcycling in Factorio, while remaining flexible enough to support analysis and optimization. Let us index one iteration loop by a counter $t = 0, 1, 2, \ldots$, where each iteration corresponds to one complete pass through recycling followed by assembly. Quality is discretized into $n = 5$ ordered classes indexed by $i \in \{0, 1, 2, 3, 4\}$, with $i = 0$ representing base quality and $i = 4$ representing legendary quality. Legendary items are terminal outputs of the system and are removed from circulation immediately after assembly.

Decomposition of a Loop

Each loop consists of two conceptually distinct stages.

1. *Recycling*: assembled items are recycled into ingredient-equivalents, returning a fraction of their material input and potentially increasing quality.

2. *Assembly*: all recycled ingredients are assembled back into items, possibly increasing quality further and applying a productivity bonus. Legendary items are extracted at this stage.

This ordering reflects the physical behavior of a typical upcycling setup: assembled items circulate through recycling, ingredients are reassembled, and only completed legendary items are removed from the loop.

State Variables

Let $X_{i,t}$ denote the expected number of *assembled items* of quality i entering the recycling stage at the beginning of iteration t. Legendary items are not part of the circulating state, so $X_{4,t} = 0$ for all t. Similarly, at the beginning of the process, there are no items in the loop, so $X_{i,1} = 0$ for all i. Further, let $M_{i,t}$ denote the amount of recycled ingredient-equivalents of quality i produced during iteration t and $C_{i,t}$ the number of assembled items of quality i produced during iteration t. For brevity, we collect these quantities into vectors:

$$X_t = (X_{0,t}, X_{1,t}, X_{2,t}, X_{3,t}, X_{4,t})^\top, \quad M_t = (M_{0,t}, M_{1,t}, M_{2,t}, M_{3,t}, M_{4,t})^\top, \quad C_t = (C_{0,t}, C_{1,t}, C_{2,t}, C_{3,t}, C_{4,t})^\top.$$

Assembly and Recycling Operators

Assembly quality behavior is encoded by a matrix $A \in \mathbb{R}^{5\times5}$, where A_{ij} is the probability that an ingredient of quality i produces an assembled item of quality j. Recycling quality behavior is encoded by a matrix $R \in \mathbb{R}^{5\times5}$, defined analogously for the recycling process. Let $p > 0$ denote the additive productivity applied during assembly, so that each assembly produces a total of $1 + p$ items of identical quality. Let $r \in (0, 1)$ denote the fraction of ingredient-equivalents returned by recycling. Define q_a and q_r as the quality percentages applied during assembly and recycling, respectively. Using Factorio's quality mechanics, both A and R share the same upper-triangular structure:

$$R_{ij} = \begin{cases} 0, & j < i, \\ 1 - q_r, & j = i, \\ \dfrac{9q_r}{10^{j-i}}, & i < j < 4, \\ \dfrac{q_r}{10^{3-i}}, & j = 4. \end{cases}$$

The matrix A is defined analogously with q_a in place of q_r. Both matrices are row-stochastic and encode the expected quality transition during each sub-process.

4 Derivation of Key Relations

With the modeling framework in place, we now translate the mechanics of upcycling into algebraic recurrence relations.

Recurrence Relations

At the beginning of iteration t, assembled items X_t enter recycling together with an external inflow S of assembled items. Recycling produces ingredient-equivalents according to

$$M_t = r\, R^\top (X_t + S).$$

All recycled ingredients are then assembled, yielding

$$C_t = (1 + p)\, A^\top M_t.$$

Legendary items are extracted after assembly, and only non-legendary items re-enter the loop. Let Π denote the projection matrix that zeros out the legendary component. The circulating state therefore evolves as

$$X_{t+1} = \Pi C_t.$$

Combining these relations yields the affine recurrence

$$X_{t+1} = r(1 + p)\, \Pi A^\top R^\top (X_t + S)$$

We define the loop operator

$$L = r(1 + p)\, \Pi A^\top R^\top,$$

so that

$$X_{t+1} = L(X_t + S).$$

Mass Decay

Since both A and R are row-stochastic and Π is a contraction, the induced 1-norm of L satisfies

$$\begin{aligned}\|L\|_1 &\leq r(1+p)\,\|\Pi\|_1\|A^\top\|_1\|R^\top\|_1 \\ &< r(1+p)\|A^\top\|_1\|R^\top\|_1 \\ &= r(1+p).\end{aligned}$$

Whenever $r(1+p) \leq 1$, it follows that $\rho(L) < 1$, ensuring that circulating mass decays geometrically in expectation. This condition guarantees that the upcycling process produces a finite amount of output from any finite input. In fact, this condition is guaranteed to hold. Indeed, the developers of Factorio recognized the need for a control on item expansion and have capped the additional productivity p at 300%. Since Factorio uses a fixed $r = 0.25$, the game ensures a stable spectral radius $\rho(L)$.

Legendary Production

Legendary items produced during iteration t are given by the legendary component of C_t:

$$y_t = e_4^\top C_t = e_4^\top (1+p)A^\top r R^\top (X_t + S).$$

For a one-time injection S with no further inflow, the total expected legendary output is

$$\sum_{t=0}^{\infty} y_t = e_4^\top (1+p)A^\top r R^\top \sum_{t=0}^{\infty} L^t S.$$

Since $\rho(L) < 1$, the Neumann series converges and

$$\sum_{t=0}^{\infty} y_t = e_4^\top (1+p)A^\top r R^\top (I-L)^{-1} S.$$

For $S = e_0$, this expression gives the expected number of legendary items ultimately produced per base-quality input item.

Steady State Behavior

If a constant inflow S of assembled items is added each iteration, the system converges to a steady state satisfying

$$X^* = L(X^* + S),$$

or equivalently

$$X^* = (I-L)^{-1} L S.$$

The steady-state legendary production per iteration is then

$$y^* = e_4^\top (1+p)A^\top r R^\top (X^* + S).$$

This quantity characterizes the long-run operating behavior of the upcycler and directly links factory scale to sustained legendary output[1].

[1] Ask yourself: why does this look so similar to the legendary items produced in the previous subsection?

5 Results Summary and Examples

If the last section was a bit math-heavy, this section should be a reprieve as it is primarily focused on using the results above. In fact, the above results are so simply stated that they can be employed in a short 30 line script. For the sake of an example, let us choose the following parameters

- $r = 0.25$
- $p = 1 + 0.5$
- $q_r = 20\%$
- $q_a = 25\%$

For the Factorio fans out there, these parameters are derived from assembling red circuits using EM plants with legendary quality 3 modules in every step of the process. Let us assume that we begin with 32 red circuit-equivalent materials to start out with - meaning we could craft a red circuit 32 times using these materials. Let us also assume that during each loop of our upcycler system, we add 32 more red circuit-equivalent materials to keep the process healthy. By constructing the associated matrices and relevant parameters, we can quite quickly computed that

1. After 1 full iteration of assembly + recycling, we are left with $[7.2, 3.78, 0.864, 0.135, 0.018]$ red circuit-equivalents of normal, uncommon, rare, epic, and legendary quality, respectively (before adding our additional 32)
2. If we let our upcycling run for an indefinite amount of time without adding any additional inputs, the original 32 materials produce roughly 0.65 legendary circuits. Meaning we need to produce about 50 normal quality red circuits to convert it into one legendary quality circuit.
3. Similarly, if we continuously supply our system with 32 circuit-equivalents each iteration, our steady state system produces roughly 0.65 legendary circuits per loop. Additionally, there will be $[46.45, 21.01, 6.21, 2.07, 0.65]$ circuits of each respective quality in circulation. This helps designing the necessary size of the upcycler.

6 Choosing Productivity vs. Quality Modules in Assemblers

One of the most practical decisions an engineer must make when designing an upcycling system is how to allocate module slots in assemblers between productivity and quality. In Factorio, this choice typically appears as a tradeoff between an additional 25% productivity or an additional 5% quality per module. While this decision is often discussed heuristically in the community, the linear–Markov framework developed above allows us to reason about it in a more principled way. At a high level, productivity and quality influence the upcycling system in fundamentally different manners. Productivity acts as a scalar multiplier on the total amount of material flowing through the system, whereas quality reshapes the transition dynamics governing how quickly material escapes the recycling loop and is converted into legendary output. Understanding this distinction is key to making informed decisions.

Productivity as Mass Amplification

Recall that productivity enters the model through the scalar factor $(1 + p)$ in the loop operator

$$L = r(1 + p)\,\Pi A^\top R^\top.$$

Increasing productivity uniformly scales the amount of material produced during assembly and, consequently, the amount of material that re-enters the recycling stage. In isolation, higher productivity appears desirable: more items are produced per unit of input. However, this amplification is indiscriminate. Additional productivity increases not only the flow of high-quality items toward absorption, but also the circulation of low-quality items that must pass through the recycler multiple times and suffer material loss at each iteration. From the perspective of the loop operator, increasing productivity pushes the spectral radius $\rho(L)$ upward, lengthening the expected lifetime of material within the recycling loop and increasing cumulative recycling losses. In short, productivity increases throughput, but it also increases exposure to loss.

Quality as Escape Acceleration

Quality, by contrast, alters the structure of the transition matrices A and R. Increasing quality shifts probability mass upward in quality space, increasing the likelihood that an item reaches legendary quality during assembly and is extracted from the system. In terms of the linear dynamics, higher quality decreases the expected number of loop iterations before absorption. This effect is nonlinear: a modest increase in quality can substantially reduce the expected number of recycling passes, especially when starting from base or intermediate quality levels. As a result, quality improvements tend to reduce total material loss even though they do not increase raw throughput. From the perspective of the fundamental matrix $(I - L)^{-1}$, increasing quality reduces the total mass accumulated in the transient subsystem before absorption.

Interpreting the Tradeoff

The tension between productivity and quality can therefore be summarized as follows:

- Productivity increases how much material is generated during each assembly step.
- Quality decreases how long material remains in the lossy recycling loop.

An effective upcycling system must balance these two effects. Excessive productivity without sufficient quality leads to large volumes of low-quality material circulating for many iterations, incurring repeated recycling losses. Excessive quality without sufficient productivity, on the other hand, produces clean but anemic throughput. Within the mathematical model, this balance is reflected in the behavior of the loop operator L and its associated fundamental matrix $(I - L)^{-1}$. Productivity increases the magnitude of L, while quality reshapes L so that probability mass is directed more quickly toward absorption.

Rules of Thumb

Several practical heuristics emerge naturally from this analysis. First, quality is most valuable when the expected number of recycling passes is large. Early in an upcycling chain, or when starting from base-quality inputs, increasing quality tends to pay for itself by sharply reducing downstream losses. Second, productivity becomes more attractive once the system reliably upgrades items within one or two loops. When most material reaches epic or legendary quality quickly, additional productivity primarily scales useful output rather than waste. Third, a useful mental model is to think in terms of expected loop count. If a typical item circulates through many recycling iterations before reaching legendary quality, prioritize quality. If most items are upgraded rapidly, prioritize productivity. Finally, base productivity matters. When assemblers already have substantial built-in productivity, the marginal benefit of additional productivity modules diminishes relative to quality

modules. In such settings, allocating early module slots to quality often produces better long-run results.

A Broader Perspective

While the canonical Factorio numbers of 25% productivity versus 5% quality motivate much of this discussion, the same reasoning applies more generally. Any change that increases throughput should be weighed against its effect on the expected lifetime of material within lossy subsystems. Conversely, any change that accelerates absorption reduces cumulative loss, even if it does not increase immediate output. Seen through this lens, the choice between productivity and quality modules is not merely a numerical comparison of percentages, but a question of system dynamics. The optimal choice depends less on local gains and more on how quickly the system converts raw inputs into terminal output.

Model-Based Comparison and Optimal Module Allocation

The discussion above can be made precise within the linear framework already developed. In particular, the choice between productivity and quality modules can be posed as an optimization problem over the expected legendary yield obtained from a single injection of assembled input material. Suppose an assembler has a fixed number of module slots, say $M = 5$. Let m_p denote the number of productivity modules and $m_q = M - m_p$ the number of quality modules installed. We assume that productivity modules increase additive productivity by a fixed increment Δ_p per module, while quality modules increase assembly quality by a fixed increment Δ_q per module. Let p_0 denote the assembler's base additive productivity. Then the effective parameters entering the model are

$$p = p_0 + m_p\Delta_p, \qquad q_a = m_q\Delta_q,$$

with q_r and r held fixed by the recycler configuration.

For a one-time injection of assembled items represented by a vector S, the total expected number of legendary items produced over the lifetime of the system is

$$e_4^\top (1+p)\, A^\top r R^\top (I - L)^{-1} S,$$

where

$$L = r(1+p)\,\Pi A^\top R^\top.$$

Normalizing by the total amount of injected material, we define the expected legendary yield per unit input as

$$\eta(p, q_a; S) = \frac{e_4^\top (1+p)\, A^\top r R^\top (I - L)^{-1} S}{\mathbf{1}^\top S}.$$

In the common case where one unit of base-quality assembled material is injected, $S = e_0$, this quantity has a direct interpretation: it is the expected number of legendary items ultimately produced per base item consumed.

This scalar objective provides a principled way to compare module allocations. For each integer choice $m_p \in \{0, 1, \ldots, M\}$, one constructs the corresponding matrices A and L using $q_a = (M - m_p)\Delta_q$ and $p = p_0 + m_p\Delta_p$, evaluates η, and selects the maximizing configuration. Because the feasible set is small and discrete, the optimum can be found by straightforward enumeration. Beyond identifying a single optimum, this formulation also clarifies the local tradeoff between

productivity and quality. Consider replacing one productivity module with one quality module, holding all else fixed. The sign of the discrete difference

$$\eta(p - \Delta_p, q_a + \Delta_q; S) - \eta(p, q_a; S)$$

determines whether quality or productivity is locally preferable at that operating point. This comparison captures, in a single quantity, both the throughput benefits of productivity and the loss-reduction benefits of quality. Viewed this way, the common intuition discussed earlier can be restated more sharply. Quality modules tend to dominate when material is expected to circulate through many recycling loops before reaching legendary quality, since increasing q_a substantially reduces the expected lifetime of material in the lossy subsystem. Productivity modules become more attractive once most material upgrades quickly, at which point additional throughput translates more directly into legendary output. The model provides a quantitative boundary between these regimes and allows the optimal balance to be computed rather than guessed.