TGTG Fast Simulation Notebook: Model, Simulation Design, and Results

1 Scope and Reproducibility

This document *only* describes what happens in the notebook `tgtg_research_runner.ipynb`:

- It imports the compiled simulator from `tgtg_fast.py`.
- It trains a baker policy (production vector q and reserved share b) using a simple evolutionary optimizer.
- It compares (i) a baseline with $b = 0$ (no TGTG) against (ii) a regime where b is optimized (TGTG available).
- It evaluates trained policies out-of-sample (fresh randomness and fresh demand draws).
- It then defines a hyperparameter sweep over L, relative prices, and demand volatility, and produces summary plots from that sweep (this is described structurally here; results appear when the sweep is executed).

2 Economic Environment and State Variables (as implemented)

The environment is parameterized by:

- N: number of consumers per day.
- L: number of goods.
- $r \in [0, 1]$: walk-out probability when a consumer encounters a stock-out at a given preference rank.
- χ: unit cost of production.
- ρ: regular unit price.
- τ: TGTG unit price *per unit* in this notebook implementation.

2.1 Preferences

Consumers are represented by an integer matrix `prefs`$\in \{0, \dots, L-1\}^{N\times L}$. Row i is a permutation of goods, interpreted as a strict ranking of preferred goods for consumer i. The notebook uses `mode="correlated"` in its main experiments (preferences are not i.i.d. uniform permutations).

2.2 Daily demand: visit probability

On day t, each consumer visits independently with probability $\alpha_t \in (0,1)$. The notebook studies multiple stochastic specifications for the path $\{\alpha_t\}_{t=1}^{D}$:

1. **Constant:** $\alpha_t \equiv \alpha$.
2. **Beta shocks (i.i.d.):** $\alpha_t \sim \mathrm{Beta}(a,b)$ with mean fixed at $\mathbb{E}[\alpha_t] = \mu$ and "concentration" $\kappa = a + b$ controlling volatility.
3. **Logit-AR(1):** letting $z_t = \log(\alpha_t/(1-\alpha_t))$,

$$z_t = \phi z_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0,\sigma^2), \quad \alpha_t = \frac{1}{1+e^{-z_t}}.$$

3 Baker policy, timing, and sales mechanics (as implemented)

The baker chooses:

- $q \in \mathbb{Z}_{\geq 0}^{L}$: production quantities for each good (fixed within an epoch).
- $b \in [0,1]$: share of total inventory reserved for TGTG.

At the start of each day, inventory is set to q and total inventory is $Q = \sum_{\ell=1}^{L} q_\ell$. Reserved inventory is computed as reserved $= \lfloor bQ \rfloor$.

3.1 Regular sales process

Consumers arrive in a random order. A visiting consumer attempts to buy according to their ranking:

- If their top-ranked available good is in stock and selling one unit would *not* reduce total inventory below `reserved`, a sale occurs at price ρ.
- If the good is out of stock at a given rank, the consumer walks out with probability r; otherwise they check the next rank.
- If total inventory is already at or below `reserved`, regular sales are effectively closed and visiting consumers walk out.

3.2 TGTG sales and waste

After regular sales:

- TGTG sales equal `reserved` units (capped by remaining inventory), priced at τ per unit.
- Waste is leftover inventory after TGTG sales.

3.3 Profit and risk-adjusted fitness

Daily profit is:

$$\pi_t = (\text{regular\_sales})\rho + (\text{tgtg\_sales})\tau - \chi \sum_{\ell=1}^{L} q_\ell.$$

Over an epoch of D days, the notebook computes mean and standard deviation of π_t and defines fitness:

$$\text{fitness} = D\left(\bar{\pi} - \gamma\sigma_\pi\right).$$

4 Simulation acceleration and parallelization (as implemented)

The notebook relies on:

- **Common random numbers:** for each generation, all candidates share pre-generated arrival permutations and uniform draws, reducing selection noise.
- **Numba-compiled core:** the inner simulation loop over consumers and days is compiled.
- **Parallel evaluation:** candidates in a population are evaluated in parallel using a thread pool (Numba releases the GIL for compiled regions).

5 Main result block executed: Baseline vs TGTG across demand specifications

For each demand specification, the notebook runs two training problems:

- **Baseline:** force $b = 0$ (no TGTG).
- **TGTG available:** optimize both q and b.

Then it performs out-of-sample evaluation (fresh randomness and fresh demand draws) and compares production, waste, and profit.

5.1 Parameters used in this block

- Environment: $N = 600$, $L = 6$, $r = 0.35$, $\chi = 1.0$, $\rho = 2.5$, $\tau = 0.8$.
- Risk aversion: $\gamma = 0.8$.
- Epoch length: $D = 30$ days.
- Optimizer budget: population $P = 80$, generations $G = 30$.
- Out-of-sample replications: 25.

5.2 Out-of-sample summary table (executed output)

Demand spec	Prod_B	Prod_T	ΔProd	Waste_B	Waste_T	ΔWaste	Profit_B	Profit_T	ΔProfit	
constant $\alpha = 0.35$	5550	5790	240	73.12	65.28	-7.84	271.41	268.76	-2.65	0.
beta conc=5 (high vol)	5160	5340	180	1026.76	1106.40	79.64	172.44	174.80	2.36	0.
beta conc=20	5370	5250	-120	452.80	391.52	-61.28	230.77	229.87	-0.89	0.
logit-AR1 ($\phi = 0.8$, $\sigma = 0.6$)	6390	5850	-540	879.84	599.80	-280.04	246.18	242.52	-3.66	0.

Table 1: Baseline vs TGTG (optimized b) across demand specifications. Values are out-of-sample means from the notebook run.

5.3 Figures generated by the notebook block

prod_baseline_vs_tgtg.png

Figure 1: Production per day: baseline vs TGTG across demand specifications.

delta_prod_by_spec.png

Figure 2: Δ production per day (TGTG minus baseline) across demand specifications.

delta_waste_by_spec.png

Figure 3: Δ waste per day (TGTG minus baseline) across demand specifications.

delta_profit_by_spec.png

Figure 4: Δ mean profit (TGTG minus baseline) across demand specifications (out-of-sample).

5.4 Interpretation (aligned with the notebook's intended questions)

The notebook's robustness question is operationalized as the sign and magnitude of Δ production across demand models:

- In this run, Δ production is positive under constant demand and highly volatile i.i.d. Beta shocks, but negative under moderate Beta volatility and persistent logit-AR(1) demand.
- In several specifications, the optimizer chooses $b^{=0}$, meaning that even when "TGTG is available" the best-response found by the notebook is to not reserve inventory for TGTG.

6 Hyperparameter sweep block (what the notebook does)

The notebook then defines a sweep over:

- Number of goods $L \in \{3, 6, 10\}$.
- Margin via $\rho/\chi \in \{1.8, 2.5, 3.2\}$ (implemented by setting $\rho = \text{margin} \cdot \chi$).
- TGTG discount $\tau/\rho \in \{0.2, 0.35, 0.5\}$ (implemented by setting $\tau = \text{discount} \cdot \rho$).
- Demand volatility via Beta concentration $\kappa \in \{5, 20, 100\}$.

For each sweep point, it trains a baseline ($b = 0$) and a TGTG-available policy (optimize b), then evaluates both out-of-sample and stores:

$$\Delta\text{Production}, \ \Delta\text{Waste}, \ \Delta\text{Profit}, \ b^{\cdot}$$

The notebook produces:

- A plot of mean Δ production per day vs concentration (log scale), grouped by L.
- A plot of mean b vs discount τ/ρ.
- Grouped summary tables (means by L and concentration).

7 Note on the TGTG price interpretation

As stated in the notebook text, τ is treated as a *per-unit* TGTG price in this implementation. If you want a bag model (bag size k, price per bag), the simulator logic must be adjusted accordingly; the notebook explicitly flags this as a modeling choice.