These suggestions may improve the reliability and business value of A/B tests - even without adding complex new algorithms.

1. **Batch Simulations and Robust Reporting**
2. **Stochastic (Noisy) Rewards to Mimic Real-World Data**

BATCH SIMULATIONS AND ROBUST REPORTING

Motivation:
Bandit algorithms are often evaluated on a single run, but in reality, random chance can create big swings in performance. Businesses and researchers need to know not just “who won,” but how reliable those results are across multiple runs.

Implementation:
Run each algorithm (e.g., Epsilon-Greedy, Thompson Sampling) across 50-100 independent simulation batches.
Compute and report average reward, standard deviation, min/max, and confidence intervals for all key metrics (cumulative reward, regret, optimal arm selection).
Plot “uncertainty bands” around curves to make visual comparisons more honest.

Benefits:
Shows whether an algorithm’s victory is lucky or robust
Helps tune algorithms for consistent, not just best-case, performance
Matches best practices in both academic research and industry reporting

STOCHASTIC REWARDS FOR REALISTIC SIMULATION

The problem:
Most student bandit assignments use fixed rewards (e.g., Arm 1 always gives “1”) - but in actual A/B testing, outcomes (like clicks, purchases) are always noisy.

Implementation:
Make the reward for each arm a random draw from a distribution (e.g., Normal, Bernoulli, or Poisson) centered at that arm’s “true mean.”
Optionally, test performance as the variability of rewards increases.

Benefits:
Experiments will better match real-world marketing, recommendation, and website experimentation
Reveals which algorithms are truly robust to noise, not just “lucky”
Surfaces new insights, like which bandit is less sensitive to bad runs when customer behavior is unpredictable

UPGRADE TO EXISTING ALGORITHMS: ADAPTIVE “COOL-OFF” STRATEGY
Motivation:
In real A/B tests, top-performing arms can change due to trends, seasonality, or new customers. Instead of “locking in” a winner, periodically allow a brief reset window for additional exploration.

Implementation:
Every set number of trials (e.g., every 1000 pulls), temporarily increase exploration (reset epsilon, or increase Thompson variance).
Monitor if a different arm starts to outperform.
Return to normal settings if no switches happen.

Benefits:
Helps adapt to “shifting winners” and maximizes long-term value
Simulates real continuous testing at companies, especially for live web or marketing systems