5 THINGS TO CONSIDER WHEN DEVELOPING PRECISION-PRICING OPTIMIZATION TOOLS

The industry-wide scramble for deposits makes it all too clear that surgical pricing is the wave of the future. A successful strategy of building relationships and customer treatments can mean the difference between a customer who brings more money to the bank and one who walks away from the bank entirely. With this change, complexity has expanded exponentially to add much more granular pricing across product, geographic and customer segments. It is now common for one bank to have tens of thousands of price points within its retail franchise alone — and that number is only expected to grow as analysis and treatments are increasingly brought down to the customer level.

Such complexity demands a different approach to pricing for many institutions. To manage this detailed level of pricing, some sort of optimization is needed. But while optimization is easy conceptually, there are enormous practical limitations to deployment.

Novantas technology experts Kaushik Deka and Ted Gibson recently spoke at Strata Data Conference 2018, one of the world's largest big data and analytics conferences, about PriceTek's Next Gen Optimization engine. We asked Kaushik and Ted to share some lessons learned about taking the practical aspects of optimization and the opportunity of taking precision pricing to the next level. There are five key takeaways for creating a practical optimization environment:

1. Fast runtimes and granular price optimization at scale are both possible with the right architecture.

Pure supply-side pricing ("what is the value to the bank?") based on costs and mechanical funds transfer pricing have given way to demand-side pricing ("what is the value to the customer?"). Furthermore, these changes have occurred at a time of rising rates, new digital banking competitors, increased customer churn and shifting regulatory requirements. As a result, **the scale of pricing decisions today requires an easy-to-use granular solution while guaranteeing fast runtimes.**

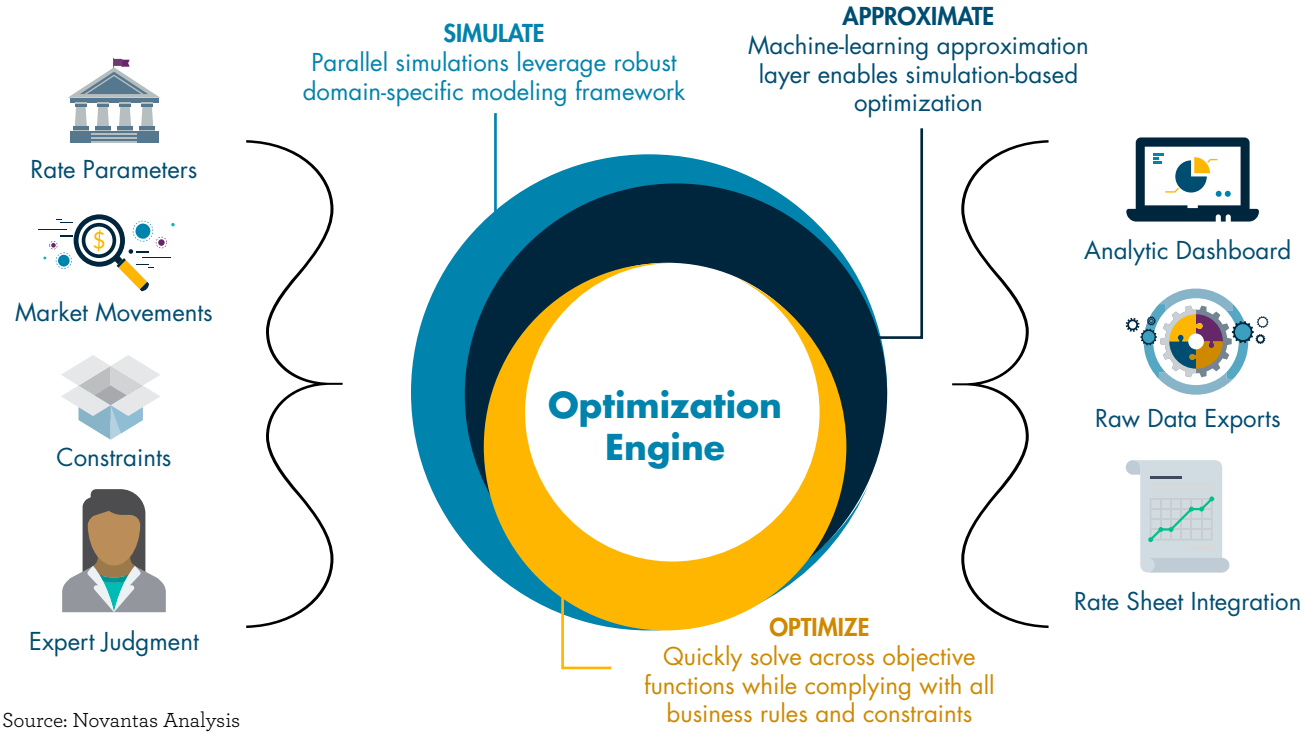
At the same time, it remains vitally important to retain modeling complexity for forward-looking projections to make sure pricing decisions are made based on the best possible information. **A solution architecture designed for simulation-based optimization and one that leverages cluster computing can balance performance, flexibility and complexity of objective functions.** Parallel simulations under various assumptions and constraints can be spawned in a big data cluster to improve overall runtime. A machine-learning model fitted upon the simulator output can further speed up optimization over the complex solution space.

2. Rate projections are as important as balance projections.

Typical optimization use cases include maximizing revenue or minimizing interest expense while meeting a predefined portfolio balance target. These often require adjustments to 'front book' (new acquisition) and 'back book' (existing customer) prices across all available products, geographies, customer segments and product attributes.

It is essential to capture intrabank

FIGURE 1: BEST PRACTICES FOR PRICE OPTIMIZATION



Source: Novantas Analysis

cannibalization, or switch between products, separately from other flows in order to accurately capture the rate paid for those balances. Modeling switch separately from other flows, along with incorporating price position at last renewal and teaser and term product schedules, ultimately introduces enough cross-product and time series complexity to require a sophisticated simulation-based optimization solution instead of a simpler approach.

3. The right data model is key.

To accurately project forward-looking deposit balances at a granular level, models should be based on the customer, account and competitive attributes that are most crucial to identifying differences in behavior. Siloed data sources, which are still all too common, make this cross-data fertilization impossible. For this reason, the best practice is to move towards a customer-centric data model that harmonizes data from multiple sources. Data pipelines can then transform this raw synthesized information into various derived metrics across the pricing grid, ultimately enriching the models that feed into the optimization engine.

4. Expert judgment is a key component of statistical models.

Pricing analysts and managers often use business knowledge to augment statistical models, such as knowledge of the sales force, modifying extrapolated projections, using models from adjacent cells with a wider range of historical experience (particularly after running a pricing pilot) and other factors outside the models.

When incorporating business judgment as part of an optimization routine, it is important to prioritize governance and easy reusability. Creating a framework to save these types of rules into a reusable library that interacts directly with the optimization engine can achieve this goal.

5. A team approach can help combat integration challenges.

Due to the number of moving parts, building an integrated system can be challenging without proper business foresight, seasoned product management and a deep engineering bench.

The first major need is a declarative, extensible, business-friendly language that can be interpreted both by the simulator and the optimizer. One potential solution is to adopt a "business rules" framework where all user inputs (such as constraints, bounds and targets) can be designed as rules to enable a flexible and intuitive expression of business goals.

A model abstraction framework is also needed to enable fast optimization of complex large-scale simulations. Dividing the simulation space into partitions that allow independent distributed computing can help alleviate this problem. Building approximate models on the simulated dataset allows the optimizer to explore the optimization space expediently, obviating expensive full simulation runs and thus further improving overall runtime.

Finally, a streamlined data pipeline architecture must integrate the distributed simulator and the optimizer within an application session. This roundtrip framework is needed to simulate and create training datasets on the cluster, build approximation models, optimize, re-simulate and ultimately send optimization results back to the front-end pricing application — all on demand and at the push of a button. ■