Invention Disclosure for Optimization of Markdown Scheduling and Allocation of Distribution Center Inventory

Adam N. Rosenberg       Jeffrey D. Moore

2004 December 7
## Contents

1 INTRODUCTION  
2 THE PROBLEM  
3 THE BUSINESS RULES  
4 MATHEMATICAL MODELS  
  4.1 Demand Model  
  4.2 One Product and Store  
  4.3 Linked Products and Stores  
5 MARKDOWN OPTIMIZATION  
  5.1 Combinatorial Nested Search Algorithm  
  5.2 Price Points Search  
  5.3 Step Dates Search  
  5.4 Distribution Center (DC) Algorithm  
6 FORECASTING  
7 CONCLUSION


1 INTRODUCTION

Product life-cycle management includes price optimization during introduction, discontinuation, regular, and promotion periods. We describe a new invention for optimizing pricing during product discontinuation. We call this markdown pricing, and the objective is to schedule discounts to clear inventory for retail. This invention includes several new methods for markdown optimization.

- Inventory exhaustion or profit maximization strategies.
- Business rules and constraints.
- Efficient combinatorial search algorithm.
- Stochastic hedging.
- Joint optimization schedules for linked products and stores.
- Optimized allocation of distribution center inventory.
- Incremental model updates during a markdown.
- Inventory and unit-sales forecasts.
- Integration with regular price optimization and promotions.

A markdown is a retail activity where remaining inventory of a product or collection of products is sold by a store or a collection of stores for a designated period of time. A markdown is also called a clearance or, more casually, a blowout. In contrast to usual retail sales where inventory is continually replenished as needed, markdown inventory is considered fixed and perishable. Once the end of the markdown time interval is reached, the remaining inventory has a salvage value which may be zero.

2 THE PROBLEM

The optimal markdown strategy can be determined by one of two objectives. First, a margin optimization attempts to realize the maximum profit, price minus cost times quantity sold plus salvage value minus cost for inventory remaining. Second, an inventory optimization attempts to reduce inventory to minimum levels where ties are broken by the margin objective.

Shopper demand for markdown products in markdown stores is time dependent over the markdown interval, both in quantity of shoppers considering buying marked-down inventory and how much they buy as a function of price. The discounting activity of the markdown process creates its own additional demand factor called promotional lift in addition to the increase in sales from reduced price.

A markdown is described by a markdown schedule, the sequence of price cuts from regular price as shown in Figure 1. While price is not required to change over time, any price change
Figure 1: Markdown price cuts with inventory depletion
must be downward in discrete steps. Product-store combinations in a markdown are divided into schedule groups where all products and stores in each group are linked in their markdown schedules by their percents off regular price. A schedule group may contain products with different prices as the discounts are linked by discounting common fractions of price.

The markdown decision is dynamic. By that we mean that the markdown decision may be revised throughout the markdown interval so long as the history-plus-proposed-markdown behavior follows the rules for acceptable markdown pricing. For example, if we find that some products are selling fewer than expected, then we could decrease their future markdown prices for the rest of the markdown interval.

Another part of the markdown problem is allocated inventory from a distribution center (DC) to stores for each product in the markdown. While some inventory is specific to a store and may not be moved to another store, some inventory may be at a DC at the beginning of the markdown. Each DC has a list of products and stores and we can allocate its inventory of markdown products to the stores that maximize our objective.

The data flow in the markdown optimization and forecasting is summarized in Figure 2.

### 3 THE BUSINESS RULES

In addition to the above description of a markdown, clients impose their own specific business rules. These rules define a solution space for markdowns.

- **Minimum and maximum number of markdowns.** The markdown solution for each schedule group is a sequence of discrete steps in price. Zero markdowns means maintaining regular price for the entire interval. This rule can require some minimum number of steps and forbid more than some maximum number.

- **Minimum and maximum percent off.** Once there is a markdown from regular price, there may be limits on the minimum and maximum discounted prices in terms of their percentage off regular price.

- **Minimum and maximum change in percent off at each markdown.** Each step in the markdown involves a price reduction, some percentage of regular price. This rule limits the range of price drop at each step.

- **Maximum initial percent off.** The initial drop may have a more severe limit on its drop or there may be a single specified percent off for it.

- **Minimum and maximum days between markdown steps.** When there are two or more markdown steps, the time between them can be limited to a specified range.

- **Minimum and maximum days for the last markdown.** The last markdown may have limits on its duration, perhaps different than the time intervals between markdown steps.

- **Allowed percent-off or ending-number values.** A list of allowed percents off may be used or, alternatively, rules for each digit of the price may be specified for price ranges. An example of ending number rules would be ending in .49 or .99 up to five dollars, ending in .99 up through twenty dollars, and ending in 4.95 or 9.95 through forty dollars, and ending in 9.00 above that. There are additional rules that exclude product-stores on markdown as candidates for promotions or regular pricing.
Figure 2: Data flow for markdown optimization and forecasting
4 MATHEMATICAL MODELS

We start with the demand model used in Khimetrics regular retail price optimization and extend that model to markdown for a single product in a single store. We extend the discussion to multiple product-store combinations linked by a common percent-off markdown schedule.

4.1 Demand Model

Shopper demand for a product in a store is modeled by the following formula.

\[
D(t) = TDD(t) e^{q_0 - \beta p(t) + v}
\]

where

- \(D(t)\) = expected demand at time \(t\),
- \(p(t)\) = price as a function of time,
- \(TDD(t)\) = time dependent demand factor,
- \(q_0\) = scale factor of demand curve,
- \(\beta(t)\) = demand elasticity parameter, and
- \(v\) = promotional lift, 0 for regular price.

If \(\beta(t)\) is a constant \(\beta\) over the markdown interval, then we can model total demand \(D\) as a function of a price \(p\) constant over time.

\[
D = Q e^{-\beta p + v}
\]

where \(Q = q_0 \sum_t TDD(t)\).

Our knowledge of the markdown process is incomplete. We represent this incomplete knowledge with statistical uncertainty in the markdown models. For this reason, the markdown optimization process consists of an initial optimization at the beginning followed by reoptimization calculations during the markdown interval as shown in Figure 3.

4.2 One Product and Store

For a schedule group with a single product and store, a \(\beta\) value that does not change over time, deterministic demand, and no rules-restrictions on percent off, the best solution is a single optimal price \(p^*\). There are two potentially optimal solutions, inventory exhaustion and treating salvage value as cost.

The inventory exhaustion solution is reached by setting total demand \(D\) to inventory \(I\) so

\[
I = Q e^{-\beta p + v}
\]

which gives us

\[
p = \frac{\log Q - \log I}{\beta}.
\]

(All logarithms here are base \(e\), natural logarithms.)
Figure 3: Optimization and reoptimization to improve markdown

When inventory is not constrained, maximum profit $\pi$ is reached when price $p$ is $1/\beta$ higher than cost $c$.

\[
q = \text{constant } e^{-\beta p} \quad \text{(demand curve)} \quad (6)
\]

\[
\frac{dq}{dp} = -\beta q \quad \text{(calculus chain rule)} \quad (7)
\]

\[
\pi = (p - c)q \quad \text{(profit is price minus cost times quantity)} \quad (8)
\]

\[
\frac{d\pi}{dp} = q + (p - c)\frac{dq}{dp} \quad \text{(calculus product rule)} \quad (9)
\]

\[
\frac{d\pi}{dp} = q - \beta(p - c)q \quad \text{(substitution)} \quad (10)
\]

\[
0 = q^* \left(1 - \beta(p^* - c)\right) \quad \text{(set derivative to zero for maximum $\pi$)} \quad (11)
\]

\[
p^* = c + \frac{1}{\beta} \quad \text{(algebra)} \quad (12)
\]

The salvage-cost solution follows the same pattern except that the cost is the salvage cost $s$ instead of the invoice cost $c$.

\[
p^* = s + \frac{1}{\beta} \quad (13)
\]

We choose the higher of these two prices for a margin optimization and use the price in Equation 5 for an inventory optimization.
There is one wrinkle in the single-product-and-store case having to do with the stochastic nature of shopper demand. The markdown rules allow prices only to go down, so it makes sense to start with a higher price with the expectation of lowering the price later in the markdown interval as shown in Figure 4. The price can be lowered as planned if product sales reach or exceed expected levels and the higher price can stay if sales are at the low end of their probability distribution.

The stochastic distribution of markdown behavior can come from variation in the statistical calculation of model parameters \( q_0, \beta, \) et cetera \( \) or from the statistical variation from the mean in shopper behavior, mostly when inventory levels are small.

If the \( \beta(t) \) value is allowed to decline over time to reflect shoppers’ expectation of lower price, then we expect the optimal price over time \( p^*(t) \) to decline over time.

### 4.3 Linked Products and Stores

When multiple product-store combinations are linked in the same schedule group, their percent-off discounts over time are required to be the same during the markdown. We have been calling product-store combinations squares as shown in Figure 5. The optimal price trajectory \( p^*(t) \) is no longer constant even if \( \beta \) is constant for each product and store. The optimal price over time changes when one of the products at one of the stores runs out of inventory.

Once a square runs out of inventory during the markdown, there is a downward change in the optimal price. Any square running out of inventory before markdown end would be more profitable with a higher price that would sell the same inventory more slowly. Once that square sells out, there is less pressure to keep the price up and \( p^*(t) \) takes a jump downward at square exhaustion time.

Calculating the optimal price trajectory \( p^*(t) \) for a schedule group with a large number of squares is difficult. There is a mathematical closed form solution if all the squares have the same \( \beta \) value even if they have different \( q_0, TDD(t), I, \) and \( s \) values, but it is difficult to calculate and it requires determining the order of square exhaustion amid the varying parameters in the model.

If the \( \beta \) values are different for different squares in the schedule group, then any mathematical closed form is going to be difficult to formulate and even more difficult to solve numerically.

### 5 MARKDOWN OPTIMIZATION

The markdown optimization is explained below and summarized in Figure 6. As described in the subsections below, the algorithm is DC allocation followed by three nested loops shown as three shaded boxes in the figure. Each schedule group is independently optimized in the outer loop. Selections of markdown price points form the middle loop and evaluation of varying dates of their realization form the innermost loop of the markdown optimization.
Figure 4: Starting markdown high as hedge against stochastic variation
Figure 5: Product-store combinations form markdown squares
Figure 6: Flowchart diagram of markdown optimization
5.1 Combinatorial Nested Search Algorithm

Even when we have a pleasant, mathematically-closed solution for markdown price, business rules create their own constraints on markdown pricing. The problem is multi-dimensionally complex, some number of price points depending on some rules are selected from another number of choices with transition dates that conform to other rules. An analytic solution method might seek the price trajectory \( p(t) \) closest to the mathematically optimal pricing \( p^*(t) \) somewhere in all these rules-determined choices.

Instead of finding the closest rules-based solution to some mathematically-determined trajectory, we use a combinatorial search over the rules-based choices and evaluate the objective function to find the best one. As a complete search of all the possible choices can be computationally enormous, we divide the search into two phases, price points and step dates, and then select a subset of the choices to create a targeted search of the markdown price choices.

5.2 Price Points Search

The valid price points are determined as a list of percents off or ending numbers depending on the rules. The zero-percent-off option of regular price is added to the list. The choices of percent off or ending number are tested against the minimum and maximum amounts in the rules. We enumerate the complete list of choices and sort them in price order from high to low.

For each price on the list, we use the minimum and maximum percent change rules to determine which prices can come next to form a network shown in Figure 7. Starting with zero-discount regular price, we can enumerate the paths through this network. After the zero-length path of regular price with no markdowns, we step through the paths of length one, then the paths of length two, and so on.

If there are many price points, common with ending numbers where hundreds of prices end in nine, then we trim the search tree using spaced samples. Small changes in price create small changes in revenue, so this computer-time-saving heuristic costs little in objective value. If there are \( N \) choices and we want to sample only \( K \) branches, then we use every \( M^{th} \) choice on the list, where \( M = N/K \), or all of them if there are no more than \( K \) choices. The value of \( K \) is reduced as the number of markdown steps increases to keep the total search from growing faster than two to the number of markdown steps. Otherwise, a three-markdown search with 1000 ending-number price points and typical rules involves about 100 million combinations to try.

We use a price-point-path counting algorithm that increments the last non-maximal markdown until there are no more and then increases the number of markdowns by one.

5.3 Step Dates Search

Once we have a sequence of prices, we need to find the dates for price changes, the step dates. We can think of moving the dates later as increasing average price through the markdown, but solving this in a mathematically closed form has been elusive for multi-square schedule groups, so we do a step-dates search for each price-points path.
Figure 7: Price points and network of allowed markdown changes

For a single markdown the only decision is when the markdown starts with earliest and latest possible dates determined by the rules. For multiple markdown steps we have a complex space of date choices. The step-date space (try saying that three times quickly) is convex. By that we mean that if dates $d_1, d_2, \ldots, d_n$ are rules-valid start dates for $n$ markdown steps and dates $e_1, e_2, \ldots, e_n$ are also rules-valid, then the convex-combination dates are also valid.

$$\lambda d_1 + (1 - \lambda)e_1, \lambda d_2 + (1 - \lambda)e_2, \ldots, \lambda d_n + (1 - \lambda)e_n$$

(14)

This holds true even if fractional dates are rounded to the nearest whole day after this calculation.

In the single-markdown case, we find the earliest and latest start date for $d_1$ and $e_1$ and we vary $\lambda$ from zero to one. The search divides the zero-to-one search space into five even spacings with six samples, $\lambda = 0.0, \lambda = 0.2, \lambda = 0.4, \lambda = 0.6, \lambda = 0.8,$ and $\lambda = 1.0$. The $\lambda$ values on either side of the best of these (not less than zero or greater than one) are used as boundaries for another five-point search.

For the multiple-markdown case we expand the search space to two dimensions. The dates $d_1, d_2, \ldots, d_n$ are the earliest dates the rules allow and $e_1, e_2, \ldots, e_n$ are the latest. We determine the earliest-latest window for each step date by working backward with the shortest duration allowed by the rules for each price point and, if there is a maximum duration for the last price point, with the longest duration. We work forward with the shortest duration for price points and, if there is already a markdown, with the longest duration. The combination of these constraints defines a minimum and maximum date for each step date.
The typical solution for $d_j$ and $e_j$ leaves all the middle steps at their shortest possible length as $d_j$ packs markdown steps at the early dates and $e_j$ packs markdown steps at the late dates. To allow the middle steps some flexibility in their duration, we create another pair of endpoints $f_j$ and $g_j$. For one of these we set the first markdown as short as possible and all the others as long as possible. For the other we set the last markdown as short as possible and all the others as short as possible. Unless the two are identical, one of these will be consistently earlier than the other and that one is $f_1, f_2, ..., f_n$ while the other is $g_1, g_2, ..., g_n$.

We create a two-dimensional square of solution search space with search parameters $\lambda$ and $\mu$ as shown in Figure 8.

$$\mu(\lambda d_1 + (1 - \lambda)e_1) + (1 - \mu)(\lambda f_1 + (1 - \lambda)g_1),$$  
$$\mu(\lambda d_2 + (1 - \lambda)e_2) + (1 - \mu)(\lambda f_2 + (1 - \lambda)g_2), ...,$$  
$$\mu(\lambda d_n + (1 - \lambda)e_n) + (1 - \mu)(\lambda f_n + (1 - \lambda)g_n)$$

We search over a four-by-four grid of sixteen possibilities in $\lambda$ and $\mu$ going from zero to one by thirds. Whichever of these is best defines the next, smaller search area. This shrinking of search rectangles goes on until no improvement is seen in the objective function.

We can think of $\lambda$ adjusting the average price and $\mu$ adjusting the variability of price, the two parameters together approximating a two-moment search of price distribution over the mark-
down interval.

These calculations are facilitated by keeping \( TDD(t) \) in the form of a cumulative demand for each date, \( TDD_{total}(t) \) and relying on \( \beta \) being constant, so the \( t_0 \)-to-\( t_1 \) demand for price \( p \) is

\[
(TDD_{total}(t_1) - TDD_{total}(t_0)) e^{\beta p v}. 
\] (18)

5.4 Distribution Center (DC) Algorithm

The inventory available for a markdown square can come from the store itself and can be augmented by inventory from a distribution center (DC). Each product has zero or more DCs each of which has inventory and a list of stores in the markdown.

In the DC-allocation stage, we represent the post-DC-allocation markdown optimization with the single-price margin optimization for maximum revenue \( r \). The assumed pricing strategy is to sell all inventory so long as the incremental value of inventory \( dr/dI \) exceeds the salvage value \( s \) and so long as the price \( p \) does not exceed regular price \( p_{reg} \). The optimal inventory allocation is when every store receiving inventory from the DC is operating at the same incremental revenue per unit inventory \( dr/dI \).

Define the total quantity term \( Q \) as in Equation 3.

\[
Q = q_0 \sum_t TDD(t) 
\] (19)

The incremental revenue per unit inventory is

\[
p_{reg} \quad \text{when} \quad I < Q e^{-\beta p_{reg}} 
\] (20)

\[
s \quad \text{when} \quad I > Q e^{-\beta s - 1} 
\] (21)

\[
\frac{\log Q - \log I}{\beta} \quad \text{otherwise} 
\] (22)

There is an asymmetry in these equations, the \(-1\) term in the exponent in Equation 21, as the barrier to raising markdown price \( p \) is regular price \( p_{reg} \) while the barrier to lowering price is the incremental value of inventory falling below salvage value \( s \). This causes a \( 1/\beta \) discontinuity at \( I = Q e^{-\beta p_{reg}} \) in the incremental value of inventory.

The DC strategy is to allocate inventory initially and then to move that inventory from lower to higher incremental value. This is a stable process because the incremental value of inventory decreases for each store as inventory is added.

6 FORECASTING

In the Khimetrics pricing model for retail, products are aggregated at three levels and stores at two. Through the product-linking process, products are aggregated into price families where prices, brand, and seasonal trends are linked, demand groups where seasonal trends are linked, and categories where relative store weights are linked. Through the store-linking process, stores are aggregated into superstores where model parameters are linked and zones where optimal
price assignments are linked. The model parameters described in Section 4 are computed for aggregated products and stores in the regular (non-markdown) Khimetrics price-optimization product.

The markdown optimization requires model parameters by product and store for individual markdown squares. So we use a maximum-likelihood statistical-model calculation to refine the aggregated parameters down to the individual product-store level. For each square, the Bayesian prior weights of the aggregated parameters depend on the number of sales observations available.

The individual-square model parameters is used to calculate the initial inventory for the markdown process as the input inventory is typically for an earlier time.

During the markdown interval, we forecast product sales in each store using the model parameters and using a markdown-wide $v$ parameter for the promotional lift of having a markdown. These forecasts based on individual-square model parameters are used in the markdown optimization to calculate the margin and inventory levels for the optimization in Section 5. These forecasts are aggregated and reported to the user. The forecast accounts for regular pricing, promotions, and markdown. The forecast for unit sales are subtracted from inventory to produce an inventory forecast.

7 CONCLUSION

The algorithms described here are executed in the Markdown Optimization Engine (MOE) which is currently part of the Khimetrics Markdown product. It has produced markdown plans for ending-number price lists with hundreds of price points and for markdowns with thousands of product-store combinations. We expect it will handle enterprise-scale optimizations within reasonable computer-time limits.