PATENT APPLICATION

RULE RELAXATION AND SUBSET OPTIMIZATION SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

This application relates to co-pending and concurrently filed Application No. 09/999,078 (Attorney Docket No. DT.0106) filed November 30, 2001, entitled "Selective Merchandise Price Optimization Mechanism", by Michael Neal, Krishna Venkatraman, Rob Parkin, Suzanne Valentine, Phil Delurgio, Hau Lee, and John Close, which is incorporated by reference herein for all purposes.

BACKGROUND OF THE INVENTION

The present invention relates to providing optimized pricing for a subset of a plurality of products for a plurality of stores.

In businesses, prices of various products must be set. Such prices may be set with the goal of maximizing profit or demand or for a variety of other objectives. Profit is the difference between total revenue and costs. Total sales revenue is a function of demand and price, where demand is a function of price. Demand may also depend on the day of the week, the time of the year, the price of related products, the location of a store, and various other factors. As a result, the function for forecasting demand may be very complex. Costs may be fixed or variable and may be dependent on demand. As a result, the function for forecasting costs may be very complex. For a chain of stores with tens of thousands of different products, forecasting costs and determining a function for forecasting demand are difficult. The enormous amounts of data that must be processed for such determinations are too cumbersome even when done by computer.

It is desirable to provide an efficient process and methodology for determining the prices of individual products such that profit (or whatever alternative objective) is optimized.

During optimization, it may be found that one of the plurality of rules is infeasible. Such a finding of infeasibility may stop the optimization process. It would

be desirable to provide a method of handling optimization when a rule is found to be infeasible.

It is desirable to update prices as new information about the products and competitive environment is received. Examples of this type of information would include product cost updates, product addition or discontinuance, and new competitive price or base price data. Ideally, when a user receives new information, they would run a secondary optimization over a whole product category. This would allow the optimization to fully capture all of the complementary and substitution effects in selecting the new set of optimal prices. However, if the cost of changing prices is not negligible, optimization and changing prices of all products in a whole product category may not be desirable.

During a price optimization, the prices of hundreds of items may be set for hundreds of stores. It may be difficult to understand the effect of various price changes.

SUMMARY OF THE INVENTION

To achieve the foregoing and other objects and in accordance with the purpose of the present invention, a method for computing a preferred set of prices for a subset of a plurality of products is provided. Generally, initial prices for a plurality of products are stored. A subset of products of the plurality of products is designated, where the number of products in the subset of products is less than the number of products in the plurality of products. Prices for the subset of products are optimized, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

In another embodiment of the invention, an apparatus for computing a preferred set of prices for a subset of a plurality of products, comprising computer readable media, is provided. Generally, the computer readable media comprises computer readable code for storing initial prices for a plurality of products, computer readable code for designating a subset of products of the plurality of products, wherein the number of products in the subset of products is less than the number of products in the plurality of products, and computer readable code for optimizing prices for the subset of product, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

In another embodiment of the invention, a method for setting prices for a subset of products of a plurality of products is provided. Optimized prices for a product category are received. Every item in the product category is priced according to the received optimized prices. New data is then provided. New prices for the subset of products of the product category are then received, where the subset is smaller than the product category, and where the received new prices are generated by storing initial prices for a plurality of products, designating a subset of products of the plurality of products, and where the number of products in the subset of products is less than the number of products in the plurality of products, and optimizing prices for products in the subset of products, while freezing the initial prices of products of the plurality of products in the product category that are not in the subset of products. Prices for the subset of products are then set according to the received new prices.

In another embodiment of the invention, computer data signal embodied in a carrier wave and representing sequences of instructions which when executed by a processor, causes the processor to compute a preferred set of prices for a subset of a plurality of products is provided. Initial prices for a plurality of products are stored. A subset of products of the plurality of products is designated, where the number of products in the subset of products is less than the number of products in the plurality of products. Prices for products in the subset of products are optimized, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

In another embodiment of the invention a price database is provided. Initial prices for a plurality of products are stored. A subset of products of the plurality of products is designated, where the number of products in the subset of products is less than the number of products in the plurality of products. Prices for products in the subset of products are optimized, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

In another embodiment of the invention, a method for obtaining optimized price data on a client system is provided. Sales data is sent to a server system for a plurality of products. Optimization preferences are selected. The optimization preferences are sent to the server system. Optimization prices for all of the plurality of products are received from the server system. Additional sales data is sent to the

server system. A subset constraint is selected. The subset constraint is sent to the server system. A new set of optimization prices for a subset of the plurality of products is received, where the number of products in the subset is less than the number of the plurality of products.

In another embodiment of the invention, a method for forming a plurality of stores into a plurality of clusters is provided. An optimization is performed for the plurality of stores to obtain individual store prices. The individual store prices are used to create the plurality of clusters. An optimization for the plurality of clusters is performed to obtain cluster prices.

In another embodiment of the invention, an apparatus for forming a plurality of stores into a plurality of clusters is provided. Computer readable code for collecting store specific information is provided. Computer readable code for optimizing combinations for each individual store based on the store specific information is provided. Computer readable code for creating clusters based on the closeness of the optimal combinations is provided.

In another embodiment a method for providing prices for items is provided. At least one iteration in a price optimization is performed and a plurality of values is generated. The determined plurality of values is used to generate an indicator of significance of price change. At least one iteration in an indicator of significance of price using the price optimization is performed to obtain optimized prices for the items.

These and other features of the present invention will be described in more detail below in the detailed description of the invention and in conjunction with the following figures.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

FIG. 1 is a high level schematic view of a price optimizing system.

FIG. 2 is high level flow chart of a price optimization process.

FIG. 3 is a more detailed schematic view of an econometric engine.

FIG. 4 is a more detailed schematic view of an optimization engine and support tool.

FIG. 5 is a block diagram to illustrate some of the transaction costs that occur in retail businesses of a chain of stores.

FIG. 6 is a flow chart of a preferred embodiment of a rule relaxation process.

FIG. 7 is an overall flow chart of a process that uses subset optimization.

FIG. 8 is a more detailed flow chart of an embodiment of a step of optimizing product category within subset limits.

FIG.'S 9A and 9B illustrate a computer system, which forms part of a network and is suitable for implementing embodiments of the present invention.

FIG. 10 is a schematic illustration of an embodiment of the invention that functions over a network.

FIG. 11 is high level flow chart of how a clustering process might be used.

FIG. 12 is a more detailed flow chart of a clustering process.

FIG. 13 is graph illustrating item prices by store.

FIG. 14 is a flow chart of another embodiment of the invention.

FIG. 15 is a flow chart of another embodiment of the invention.

FIG. 16 is a generic flow chart of an embodiment of the invention.

FIG. 17 is high level flow chart of an optimization process.

FIG. 18 is a flow chart of a generation of a significance of price change indicator process.

FIG. 19 is a more detailed flow chart of the reoptimization decision process.

FIG. 20 is a flow chart of an image metric generation process.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will now be described in detail with reference to a few preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps and/or structures have not been described in detail in order to not unnecessarily obscure the present invention.

To facilitate discussion, FIG. 1 is a schematic view of a price optimizing system 100, which may be used in rule relaxation and subset optimization. The price optimizing system 100 comprises an econometric engine 104, a financial model engine 108, an optimization engine 112, and a support tool 116. The econometric engine 104 is connected to the optimization engine 112, so that the output of the econometric engine 104 is an input of the optimization engine 112. The financial model engine 108 is connected to the optimization engine 112, so that the output of the financial model engine 108 is an input of the optimization engine 112. The optimization engine 112 is connected to the support tool 116 so that output of the optimization engine 112 is provided as input to the support tool 116 and output from the support tool 116 may be provided as input to the optimization engine 112. The econometric engine 104 may also exchange data with the financial model engine 108.

To facilitate understanding, FIG. 7 is an overall flow chart of a process that uses subset optimization 700. First, a product category is optimized (step 701). A demand group is defined as a set of products that are substitutes or near substitutes for each other. A product can belong to only one demand group. A product category consists of one or more demand groups. FIG. 2 is a more detailed flow chart of a preferred embodiment of a process that utilizes the price optimizing system 100 to optimize prices for a product category (step 701). Data 120 is provided from the store

computer systems 124 to the econometric engine 104 (step 204). Generally, the data 120 provided to the econometric engine 104 may be point-of-sale information, product information, and store information. The econometric engine 104 processes the data 120 to provide demand coefficients 128 (step 208) for a set of algebraic equations that may be used to estimate demand (volume sold) given certain marketing conditions (i.e. a particular store in the chain), including a price point. The demand coefficients 128 are provided to the optimization engine 112. Additional processed data from the econometric engine 104 may also be provided to the optimization engine 112.

The financial model engine 108 may receive data 132 from the store computer systems 124 (step 216) and processed data from the econometric engine 104. The data 132 received from the stores is generally cost related data, such as average store labor rates, average distribution center labor rates, cost of capital, the average time it takes a cashier to scan an item (or unit) of product, how long it takes to stock a received unit of product and fixed cost data. The financial model engine 108 may process the data to provide a variable cost and fixed cost for each unit of product in a store. The processing by the econometric engine 104 and the processing by the financial model engine 108 may be done in parallel. Cost data 136 is provided from the financial model engine 108 to the optimization engine 112 (step 224).

The optimization engine 112 utilizes the demand coefficients 128 to create a demand equation. The optimization engine is able to forecast demand and cost for a set of prices and promotion conditions such as store displays and ads to calculate net profit. The stores 124 may use the support tool 116 to provide optimization rules to the optimization engine 112 (step 228). The optimization engine 112 may use the demand equation, the variable and fixed costs, and the rules to compute an optimal set of prices that meet the rules (step 232). For example, if a rule specifies the maximization of profit, the optimization engine would find a set of prices that cause the largest difference between the total sales revenue and the total cost of all products being measured.

If a rule providing a promotion of one of the products by specifying a discounted price is provided, the optimization engine may provide a set of prices that allow for the promotion of the one product and the maximization of profit under that

condition. The rules normally include an objective, such as optimizing profit or optimizing volume of sales of a product and constraints such as a limit in the variation of prices

When profit is maximized, it may be maximized for a sum of all measured products. Such a maximization, may not maximize profit for each individual product, but may instead have an ultimate objective of maximizing total profit. The optimization provides a plurality of values such as optimized prices. The demand equation may be used to provide the optimization and to provide a plurality of values such as elasticities, margins, cost, and revenues.

Other embodiments may use other types of optimization processes and other demand models and other methods to generate a plurality of values.

The optimal (preferred) set of prices may be sent from the optimization engine 112 to the support tool 116 so that the stores 124 may use the user interface of the support tool 116 to obtain the optimal set of prices. Other methods may be used to provide the optimal set of prices to the stores 124. The price of the products in the stores 124 is set to the optimal set of prices (step 236), so that a maximization of profit or another objective is achieved.

Each component of the price optimizing system 100 will be discussed separately in more detail below.

FIG. 3 is a more detailed view of the econometric engine 104. The econometric engine comprises an imputed variable generator 304 and a coefficient estimator 308. The data 120 from the stores 124 is provided to the imputed variable generator 304. The data 120 may be raw data generated from cash register data, which may be generated by scanners used at the cash registers. The raw data may be processed to create imputed variables. The imputed variables may be used to estimate parameters for a demand equation D(x), which may be used to estimate the amount of sales using variables such as price.

In a preferred embodiment of the invention, sales for a demand group (S) is calculated and a market share (F) for a particular product is calculated, so that demand (D) for a particular product is estimated by D=S•F. A demand group is defined as a

collection of highly substitutable products. More preferably the creation of the demand model relies on a mixed-model framework, simultaneously utilizing information across all stores and products in a client category, where a category is defined as a collection of substitutable or complementary products. The mixed model methodology is also referred to as "Bayesian Shrinkage" Modeling, because by combining data from various stores and/or products, one can "shrink" individual parameter estimates towards the average estimate, dampening the extreme values that would result if traditional regression were used.

In developing product-level volume models for each store within a chain, one may be presented with a wide range of historical data in terms of modeling sufficiency. For some stores and/or products, the history will be quite rich, with many price changes and a variety of promotion patterns. For other stores and/or products, the history will be quite sparse, with very few price changes and little promotional activity. To maximize the stability of estimated model parameters, one might consider developing a single regression model across stores and/or products. This model might have stable parameter estimates; however, it would not fully leverage the store and/or product variability, and the resulting model would likely not predict well for a particular store and/or product. On the other hand, one might consider developing individual regression models for each store and/or product, to utilize the specific information contained in the data history. While these models might fit and predict well for the stores and/or products with substantial price variability, models would not be estimable for stores and/or products without a rich data history.

A mixed-model framework addresses the need for both highly predictive models and the existence of an estimable model for each store and product. In a mixed-effect model, information (in the form of data history) is leveraged across all stores and products, and a single cohesive model is built. Stores and products with little or no information in their data history default to an "average" (fixed-effect) model. Likewise, stores and products with a wealth of information in their data history will end up with unique parameter estimates tailored to their response pattern, via estimation of non-zero store and/or product-specific adjustment factors (random effects) which are added to the fixed-effect portion of the model.

The financial model engine 108 receives data 132 from the stores 124 and may receive imputed variables and data from the econometric engine 104 to calculate fixed and variable costs for the sale of each item. To facilitate understanding, FIG. 5 is a block diagram to illustrate some of the transaction costs that occur in retail businesses of a chain of stores. The chain of stores may have a headquarters 504, distribution centers 508, and stores 512. The headquarters 504 may place an order 516 to a manufacturer 520 for goods supplied by the manufacturer 520, which generates an order placement cost. The manufacturer 520 may ship the goods to one of the distribution centers 508. The receiving of the goods by the distribution center 508 generates a receiving cost 524, a cost for stocking the goods 528, and a cost for shipping the goods 532 to one of the stores 512. The store 512 receives the goods from one of the distribution centers 508 or from the manufacturer 520, which generates a receiving cost 536 and a cost for stocking the goods 540. When a customer purchases the item, the stores 512 incur a check-out cost 544. With the large number of retail chains, different purchasing and delivery processes may be used. Even within a single chain, different manufacturers may provide different invoicing and delivery procedures and costing system, which may be different than the processes illustrated in FIG. 5.

The financial model engine 108 should be flexible enough to provide a cost model for these different procedures. These different costs may have variable cost components where the cost of an item is a function of the amount of sales of the item and fixed cost components where the cost of an item is not a function of the amount of sales of the item. The financial model engine 108 may use these fixed and variable costs to determine $C_{s,i,k,t}$, where $C_{s,i,k,t}$, is a cost for a particular product (k) given a store (s), demand group (i), and a day (t). The financial model engine 108 may use industry data to provide standard estimates. For example, instead of measuring how long it takes to stock a box of an item, an industry data may be used to estimate this time. The standard estimates helps to reduce the amount of data that must be collected. In a preferred embodiment of the invention, the stores may only need to supply labor costs of the stores and distribution centers, cost of capital, size of an item, and number of items in a case to allow a cost modeling. By using these estimations, costs may be more easily calculated on a store level, instead of being averaged over all of the stores. This is because if large amounts of data were

measured by hand, measuring such data for each store would be difficult. The tailoring of costs per store allows the maximization of profits for each store.

FIG. 4 is a more detailed schematic view of the optimization engine 112 and the support tool 116. The optimization engine 112 comprises a rule tool 404 and a price calculator 408. The support tool 116 comprises a rule editor 412 and an output display 416.

In operation, the client (stores 124) may access the rule editor 412 of the support tool 116 and provides client defined rule parameters (step 228). If a client does not set a parameter for a particular rule, a default value is used. Some of the rule parameters set by the client may be constraints to the overall weighted price advance or decline, branding price rules, size pricing rules, unit pricing rules, line pricing rules, and cluster pricing rules. These rules will be discussed in more detail regarding a preferred embodiment of the invention. The client defined parameters for these rules are provided to the rule tool 404 of the optimization engine 112 from the rule editor 412 of the support tool 116. Within the rule tool 404, there may be other rules, which are not client defined, such as a group sales equation rule. The rule parameters are outputted from the rule tool 404 to the price calculator 408. The demand coefficients 128 and cost data 136 are also inputted into the price calculator 408. The client may also provide to the price calculator 408 through the support tool 116 a desired optimization scenario rules. Some examples of scenarios may be to optimize prices to provide the optimum profit, set one promotional price and the optimization of all remaining prices to optimize profit, or optimized prices to provide a specified volume of sales for a designated product and to optimize price. The price calculator 408 then calculates optimized prices. The price calculator 408 outputs the optimized prices to the output display 416 of the support tool 116, which allows the stores 124 to receive the optimized pricing (step 232).

The system can optimize for profit, revenue, and sales. For each type of objective, users can impose constraints on the values of the other two. For instance a user can optimize profit while ensuring that total sales and revenue do not fall below their original values, or he/she can optimize sales while requiring profit to increase by 1%.

A preferred embodiment of the optimization engine uses the group sales equation S(x) and the market share equation F(x), previously described, to predict group sales and product market share, respectively. These two are then combined to predict product sales at the store level D(x)=S(x)•F(x).

If the objective is to maximize profit then the following equation sum is maximized:

$$\sum_{i \in G} \sum_{k \in Dem_i} \hat{D}_{s,i,k,t}(P_{s,i,k,t} - C_{s,i,k,t})$$
$$= \sum_{i \in G} \sum_{k \in Dem_i} \hat{F}_{s,i,k,t} \hat{S}_{s,i,t}(P_{s,i,k,t} - C_{s,i,k,t})$$

The above mentioned rules may be constraints forming boundaries. These constraints are rules that provide limitations to price or other changes. In a preferred embodiment such constraints may be used as follows:

1. Group price advance or decline.

Since demand groups are made up of like or substitutable products, managers often wish to constrain the overall weighted price advance or decline in price for them, where the weights are the market shares of the products that constitute the demand groups. The constraints are:

$$\forall s,i\ ,\ PMIN_{s,i} \le \sum_{k} F_{s,i,k} P_{s,i,k} \le PMAX_{s,i}$$

2. Brand Pricing Rules:

Products are described and categorized by attributes such as brand, size and flavor. Therefore, each product may be associated with a set of attributes and values. These attributes are useful to us for several reasons. The attributes may be used in the regression modeling in order to create a parsimonious set of regression coefficients. They may also be used in setting rules. For instance, category managers might wish to offer store brands at lower prices compared to the competing national brands. They might also wish to constrain some product brands to be less expensive than others, either for strategic considerations or to meet their contractual obligations. Specifically, a manager can create a set

$\text{Brand}_{s,i} \equiv \{(p_{s,i,l_i}, p_{s,i,k_2}): p_{s,i,k_1}$ must cost less than $p_{s,i,k_2}\}$, which leads to the following constraints:

$$\forall s,i \text{ and } (p_{s,i,k_1}, p_{s,i,k_2}) \in Brand_{s,i}\ p_{s,i,k_1} \le p_{s,i,k_2}$$

3. Size Pricing Rules:

Managers might also wish to create rules that relate the price of one product versus another based on their sizes. For instance, they might wish for products belonging to the same brand and sharing other attributes, but with different sizes to be priced such that the larger sized product costs less per equivalent unit of measure than a smaller one. Then $\text{c Size}_{s,i} \equiv \{(p_{s,i,l_i}, p_{s,i,k_2}): p_{s,i,k_1}$ must cost less than $p_{s,i,k_2}\}$, and create a set , which leads to the following constraints:

$$\forall s,i \text{ and } (p_{s,i,k_1}, p_{s,i,k_2}) \in Size_{s,i}\ p_{s,i,k_1} \le p_{s,i,k_2}$$

4. Unit Pricing Rules:

Continuing in the same vein, managers might wish to ensure that two products that are identical in every respect but size should be priced such that the larger product costs more than the smaller one. This rule is closely related to the size pricing rule. To implement this rule, managers can create a set $\text{Unit}_{s,i} \equiv \{(p_{s,i,l_i}, p_{s,i,k_2}): e_{s,i,k_1} p_{s,i,k_1}$ must cost less than $e_{s,i,k_2} p_{s,i,k_2}\}$, where $e_{s,i,k}$ is the multiplicative factor to convert equivalent units into whole product units. This leads to the following constraints:

$$\forall s,i \text{ and } (p_{s,i,k_1}, p_{s,i,k_2}) \in Unit_{s,i}\ e_{s,i,k_1} p_{s,i,k_1} \le e_{s,i,k_2} p_{s,i,k_2}$$

5. Line Pricing Rules:

Retail customers expect common prices for certain groups of products such as, for example, cough lozenges of the same brand but different flavors. These rules may be classified as line pricing rules and implement them as follows. Product groups called line price groups may be defined as $L_l\, l = 1,..\|L\|$, where every product within a line group must have the same price in a store; i.e.,

$\forall L_l, l = 1,..\|L\|$, and $\forall k_1, k_2 \in L_l \; P_{s,i,k_1} = P_{s,i,k_2}$

6. Cluster pricing:

Retailers define geographic and other regions within which they maintain the same price for a given product. This may be translated to the notion of store clusters. A store cluster $Cluster_c$ is a set of stores such that the price of every product is invariant within the cluster. In other words, every product has the same price in every store within the cluster although each product may have a different price. In order to implement these constraints the set of stores may be partitioned into store clusters $Cluster_1,...,Cluster_{\|c\|}$ where $C \equiv \{\text{Cluster}_1,...,\text{Cluster}_{\|C\|}\}$, and a new price variable $P_{c,i,k}$ which represents the common price for product k in cluster $Cluster_c$ may be defined. This variable may be used in place of the original price variable $P_{s,i,k}$ whenever $s \in Cluster_c$.

Other rules may also be used. An example of such additional rules may be, but is not limited to, a gross margin rule, a store volume rule, or a competition rule.

A gross margin rule provides a constraint on the change of a mark up. A gross margin rule may provide a constraint on the change of the mark up of each individual item in a group or on the average mark up of the group or both. An example of a gross margin rule may be that the percentage change of the average gross margin of a group is not to exceed 5%.

A volume rule provides a constraint on the change or the volume of sales. The volume rule may provide a constraint on the change of volume of sales for each individual store or on the change of total volume of sales for a plurality of stores for an individual product or for a plurality of products. An example of a volume rule may be that for each store, the change of demand for a product is not to exceed 7%.

A competition rule provides a constraint on the difference between a competing store's prices and their prices. A competition rule may allow constraints for one or more competitors. An example of a multiple competition rule is that the price of each item must not exceed the price for the same item sold by competitor store X by more than 10% of the price and must not exceed the price of the same item

sold by competitor store Y by more than 9% of the price. To implement these constraints a store may use monitors to check prices of items in competitor store X and competitor store Y.

The objective (such as profit) is then maximized. Such an objective would provide a maximized profit given the demand data and the constraints.

RULE RELAXATION

Given the objective and the plurality of constraints the solution may be infeasible due to a conflict between the objective and the plurality of constraints. The objective may be expressed as an equation. Each constraint may also be expressed as an equation. A feasible solution may not exist that satisfies all equations representing the objective and the constraints. A preferred embodiment of the invention uses a rule relaxation process to provide a feasible solution when a feasible solution does not exit which satisfies all constraints and objectives.

During an optimization, it may be found that a rule is infeasible. A rule is determined to be infeasible, if either it cannot be satisfied when all other rules are satisfied or if an optimization cannot be performed without violating the rule. Although, one method of dealing with an infeasible rule is to drop the rule found to be infeasible, the preferred embodiment of the optimization step uses the rule relaxation process described below:

FIG. 6 is a flow chart of a preferred embodiment of the rule relaxation process. The rules are prioritized (step 604). A default prioritization may be provided, with an interface, which may allow a user to change the prioritization from the default. A check is made to see if a rule is infeasible (step 608). A rule is deemed to be infeasible if the relationship expressed by the rule is not able to be satisfied. For instance a rule $X+Y \leq 10$, is violated for when X is 7 and Y is 7 since then $X+Y=14$ which is greater then 10. If no rule is found to be infeasible, the rule relaxation process is stopped (step 628). If at least one rule is found to be infeasible, the lowest priority infeasible (LPI) rule is found (step 612). A determination is made whether rules with lower priorities than the priority of the LPI rule may be relaxed to allow the LPI rule to become feasible (step 616). In the preferred embodiment the lower priority rules are checked before higher priority rules. If it is found that rules with

lower priorities than that priority of the LPI rule may be relaxed to a point that allows the LPI rule to become feasible, then these rules with lower priorities are relaxed incrementally so that the LPI rule becomes feasible (step 620). In the preferred embodiment, lower priority rules are relaxed before higher priority rules. If it is found that rules with lower priorities than the priority of the LPI rule cannot be relaxed to allow the LPI rule to become feasible, then the LPI rule is relaxed until it becomes feasible (step 624). The rules are then rechecked to see if there are any remaining rules that are infeasible (step 608). This process is continued until all rules are feasible.

Example

To provide an example of rule relaxation, six categories of rules are provided. According to the method illustrated in the flow chart of FIG. 6, the categories of rules are prioritized (step 604) in the following order, with a ranking of 1 designating the category of rules with the highest priority and 6 designating the category of rules with the lowest priority.

1. Group price advance or decline rules. The user sets a maximum weighted group price advance or decline to 10%.

2. Size pricing rules. The user goes with the default that larger items cost less per equivalent unit than smaller identical items.

3. Brand pricing rules. For soft drinks, the user designates that the price of Brand A is never less than the price of Brand B. For juices the user designates that Brand C is always greater than Brand D.

4. Unit pricing rules. The user goes with the default that the overall price of larger items is greater than the overall price of smaller identical items.

5. Competition rules. The user designates that all prices must be at least 10% less than the prices of the same items sold by competitor X and are within 2% of the prices of the same items sold by competitor Y.

6. Line price rules. The user designates that different flavors of the same item are priced the same.

In this example, Brand A is sold by competitor Y for $8, while Brand B is sold by competitor Y for $10. In addition competitor Y sells one flavor of item C for $9 and another flavor of the same item C for $10.

A determination is made if a rule is infeasible (step 608). It is found that one of the price branding rules is infeasible, since Brand A is supposed to always have a price greater than the price of Brand B and yet competitor Y sells Brand A for $8 and Brand B for $10, and since the competition rules designate that the same items must be within 2% of the prices of competitor Y. In addition, it is found that one of the line price rules is infeasible. This is because the line price rule constrains different flavors of the same item to be the same, but competitor Y is selling one flavor for $9 and another flavor for $10.

Since the branding rules have a priority of 3 and the line price rules have a priority of 6, it is found the line price rules are the lowest priority infeasible (LPI) rules (step 612). A check is made to see if rules with a priority lower than 6 may be relaxed so that the line price rules become feasible (step 616). Since the line price rules are the lowest priority rules, no rule with a lower priority may be relaxed so that the line price rule becomes feasible. As a result, the line price rule is relaxed allowing the different flavors to have different prices by about 10% (step 624).

The process then returns to step 608 to see if any rules are still infeasible. It is found that one of the price branding rules is infeasible, since Brand A is supposed to always have a price greater than the price of Brand B and yet competitor Y sells Brand A for $8 and Brand B for $10, and since the competition rules designate that the same items must be within 2% of the prices of competitor Y. Since the price branding rules are the only rules found to be infeasible, they are found to be the lowest priority infeasible (LPI) rules with a priority of 3 (step 612). A check is made to see if rules with priorities lower than the LPI rules may be relaxed to allow the LPI to be feasible (step 616). Preferably, the lower priority rules are checked first. So first the line price rules with a priority of 6 are checked to see if they may be relaxed to allow the branding price rules to be feasible. If not, then the next lowest priority rules, the competition rules are checked to see if they may be relaxed to allow the branding rules to be feasible. In this example, it is found by relaxing a competition rule with a priority of 5 the price branding rule becomes feasible. Therefore the

competition rule is relaxed so that prices are within about 20% of the prices of competitor Y, which allows Brand A and Brand B to have the same price (step 620).

The process again returns to step 608 to see if any rules are still infeasible. It is found that all rules are now feasible, and the rule relaxation process is stopped (step 628).

In the preferred implementations, the specific infeasible solution at which the optimization process stops will depend on the optimization algorithm and the initial starting point. This means that specific rules that are relaxed in order to get to feasibility depend on both the optimization algorithm and the starting values These examples are merely illustrative and are not meant to limit the scope of the invention. In addition, rule priority was assigned by rule category. In other embodiments the rule priority may be assigned to individual rules instead of to rule categories.

Other specific methods of rule relaxation may be used. In general, rule relaxation methods provide for the prioritization of rules and cause lower priority rules to be relaxed before relaxing higher priority rules in order to make all resulting rules feasible. Another specific embodiment of this general rule relaxation method may allow different combinations of lower priority rules to be relaxed. These lower priority rules may be relaxed at the same rate or at different rates until the higher priority rule becomes feasible. Different combinations of the relaxation of lower priority rules may be specified in different embodiments.

Once the optimization (step 701) is completed, the prices of the products are sent to the stores. The prices of the products are then set to the optimized prices that are provided by the optimization.

SUBSET OPTIMIZATION

After an optimization, such as the optimization described above has been performed, new data may then be provided to the system (step 703). Such data may be provided through the Econometric Engine 104 (FIG. 1) or the Financial Model Engine 108, or through another source. If the entire product category was optimized using the new data, it is possible that all items in the product category would change because of the new data. To avoid an optimization that may cause a repricing of all

products in the product category (since the cost of repricing is not negligible), the user is allowed to specify subset limits (step 705). An example of such subset limits would be a maximum number N of products that may be repriced. A new optimization is performed using the new data, where the optimization is constrained to comply with the subset limits (step 707). For example, if a maximum number N of products may be repriced, the optimization provides an optimization with price changes of no greater than N products. Since the price of no more than N products is changed, instead of possibly the price of all products of the category, only a subset of the product category is optimized. The subset optimization may use the results of the previous optimization as initial values.

The subset optimization may choose the products that comprise this subset in a way that has the largest impact on the client's objective function. If, for example, the client's objective is to maximize profit, it is desirable to populate the subset of products whose prices are allowed to change with those products that are most likely to have the largest impact on profit. In one way of doing this, all of the available products may be ranked by their marginal contribution to the objective and then the subset of products whose prices are allowed to change is selected via mixed integer program.. The prices of all of the products not in the "optimal" subset may be frozen to provide a variant of the original scenario to obtain a new set of optimized prices.

A restatement of the process in FIG. 7, is as follows: The optimization of a product category may performed as discussed in the previous sections (step 701) to provide optimal prices, demand functions and constraints for all of the products considered in the product category. All of the input data used in the initial optimization of the product category and $\bar{p}^{*}_{Cluster_c}$, the set of optimal prices derived from that data, may be used to provide initial conditions and parameters for the subset optimization.

Changes to the state of information that the user wishes to incorporate into the new optimization are also provided (step 703). For changes in costs, base price, or competitive prices, the change in the data from the original scenario measured as a percentage of base price is used. For discontinued or new products, the input data required to construct their demand functions and constraints as they would have been defined in the original optimization, as well as an initial starting base price, are used.

The subset limits, such as the maximum number of products that the user wishes to allow in the subset optimization, are specified (step 705). One possible assumption for an algorithm may be that the maximum number is at least as large as the number of products with information changes. A new optimization is then performed within the subset limits (step 707).

FIG. 8 is a more detailed flow chart of an embodiment of the step of optimizing product category within subset limits (step 707). A first step constructs a set of products that will be considered as candidates for the subset to be optimized (step 801). A second step chooses which of the candidates will be members of the subset (step 803), for example, by solving a mixed integer problem. A third step finds the new optimal prices for the products in the subset while holding the prices of the other goods at their prior optimal level (step 805).

Construction of the Set of Candidates for Subset Optimization

The purpose of this step is to define a set of products, *C*, which will be candidates for subset optimization (step 805). In defining this set, priority is given to two distinct types of products: those products that have had changes in their state of information, and those products that have constraints that would allow their price to move if they are selected to be members of *C*.

Products with changes in state information are automatically included in *C* and in the subset of products to be optimized. The second group of products is chosen based on characteristics of their constraints and the direction of the gradient of the price variables with respect to the objective function at the incumbent optimal price. It is desirable to only choose products whose prices will actually be able to move in the subset optimization. If a product's price is at a bound in the original scenario and the gradient of the price variable indicates that the price will continue in the direction of the bound if the bound is relaxed, then the price will not move in the subset optimization since it will still hit that bound. If, however, the product's price was initially not at a bound or if it is at a bound but the gradient indicates that the price would move away from the bound if it were relaxed, then it would be expected that this price would change, if it is included in the subset optimization. In order to discover the direction of the gradient of prices at their previous optimal price, a simple model is formulated that evaluates the original objective function while

allowing prices to vary slightly from their previous optimal level, $\overline{p}^{*}_{Cluster_c}$. This model consists of the fundamental model equations that define demand group sales, market share, the user-defined objective function, and simple bounds holding prices within a small neighborhood of their previous optimal values. For a profit-maximizing problem, the model might be formulated as:

$$\text{Max} \sum_{i \in G} \sum_{k \in Dem_i} \hat{D}_{s,i,k,t} (P_{s,i,k,t} - C_{s,i,k,t})$$

$$\overline{p}^{*}_{Cluster_c} - \delta \le P_{s,i,k} \le \overline{p}^{*}_{Cluster_c} + \delta, \qquad \forall s,i,k$$

where δ is a small number.

λ^{u}_{k} and λ^{l}_{k} are defined as the marginal values on the upper and lower bounds defined in the final inequality constraints. These marginal values will be used as a measure of the price gradient as the price moves away from the incumbent optimal price. They, in concert with information about which bounds were active in the original problem, will determine which products are allowed to have price changes in the subset optimization.

Products for the second group should have the freedom to change prices during the subset optimization. To this end, all of the constraints in the initial model are evaluated and three operations are performed. First, a check is performed to see if that constraint was tight for the optimal price in the original scenario. Essentially, this just means that those store-product combinations where the right and left-hand sides of the constraint equations are the same are found. For example, suppose that there is an upper bound on optimal price:

$$P_{c,i,k} \le PMAX_{c,i,k}, \forall i,k$$

where c is the store cluster, i is the demand group, and k is the product.

For this rule, all of the store-product combinations where

$$P_{c,i,k} = PMAX_{c,i,k}, \forall i,k$$

would be found.

Rule relaxation, discussed above, may have been utilized in the original optimization. As discussed above, rule relaxation allows the optimization to automatically relax constraints subject to a hierarchy defined by the user. Since the subset optimization is based on the original constraints of the initial scenario, it is possible to have optimal prices from the initial scenario that are infeasible with

respect to some of the original constraints because those constraints were relaxed in the first scenario optimization. The second operation relaxes the bound on any store-product combinations for which the incoming optimal price is infeasible. Thus, for an upper price bound, we would set $PMAX_{c,i,k}$ equal to $P_{c,i,k}$.

The third procedure changes the initial bounds on those products which have had cost, base price or competitive price changes. If the cost, base price, or competitive price has increased, it is desirable to increase the upper bound constraints so that the optimal price is allowed to move. The upper price bound, for example, would become

$$P_{c,i,k} \le PMAX_{c,i,k}\ (1+\xi_{c,k}), \forall i,k$$

where $\xi_{c,k}$ is percentage change in cost or price with respect to base price. If the cost has increased, it is also desirable to raise the lower bound to prevent the price from flipping from the upper to the lower bound. The corresponding lower price bound would be

$$P_{c,i,k} \ge Max\left(PMAX_{c,i,k}\ (1-\lambda_{c,k}), \overline{p}^{*}_{Cluster_c}\right) \forall i,k$$

where $\lambda_{c,k}$ is a scaling factor for the allowable price region and $\overline{p}^{*}_{Cluster_c}$ is the optimal price for that product from the original scenario. If the cost or price has declined, then the lower price bound would become

$$P_{c,i,k} \ge PMIN_{c,i,k}\ (1-\xi_{c,k}), \forall i,k$$

and the upper bound becomes

$$P_{c,i,k} \ge Min\left(PMIN_{c,i,k}\ (1+\lambda_{c,k}), \overline{p}^{*}_{Cluster_c}\right) \forall i,k$$

For constraints on the average price of a demand group, the bound is changed by the market share weighted average change in the cost or price relative to initial base price.

The set of candidates for subset optimization, *C*, may now be constructed as follows:

For each store, product and constraint:

1. Initialize *C* to include all available products.
2. If $\lambda^{l}_{k} > 0$ or $\lambda^{u}_{k} < 0$, then $k \notin C$
3. If constraint is an active upper bound on price and $\lambda^{u}_{k} > 0$, then $k \notin C$

4. If constraint is an active lower bound on price and $\lambda_k^l < 0$, then $k \notin C$
5. If product has change in the state of information, then $k \in C$

Selection of the Set of Products for Subset Optimization

In selecting the N products that will comprise the set of products, *R*, which are allowed to have price changes in the subset optimization, a simple mixed integer problem is solved. A binary variable, r_k, is created that indicates whether or not a given product is chosen to become a member of *R*. An objective function is the sum of these binary variables weighted by the maximum marginal value on each product's price obtained in the step of constructing the set of candidates for subset optimization. The only constraint is that the number of products chosen must be less than *n*, the user-defined maximum number of products allowed in the subset optimization minus the number of products with changes in the state of information. This problem can be written as

$$\underset{r}{Max}\left[\sum_{k\in C} r_k \cdot Max\left(\left|\lambda_k^l\right|,\left|\lambda_k^u\right|\right)\right]$$

subject to

$$\sum_{k\in C} r_k \le n.$$

The set of products, *R*, may be defined to be considered in subset optimization as $k \in R : \{(k \in C \mid r_k = 1) \cup (k \in C \mid k \text{ has change in state})\}$.

Optimize Prices over the Product Subset R

The price variables of products that are not members of *R* are fixed and this new problem is solved with a complete optimization, such as the initial optimization described above, except that prices not in the subset are held fixed. Regression models and the predictive equations from the econometrics models are used to construct the optimization model. For example, the objective may be to maximize profit:

$$\sum_{i\in G}\sum_{k\in Dem_i} \hat{D}_{s,i,k,t}\left(P_{s,i,k,t} - C_{s,i,k,t}\right)$$

subject to constraining price changes to be within a given range of current prices.

$$PMIN_{s,i,k,t} \le P_{s,i,k,t} \le PMAX_{s,i,k,t}$$

To simplify notation, the time index may be removed from the equations. So the objective becomes:

$$\text{Maximize: } \sum_{i \in G} \sum_{k \in Dem_i} \hat{D}_{s,i,k}(P_{s,i,k} - C_{s,i,k}).$$

When the subset optimization is completed and prices are optimized over the subset (step 805) a unique subset optimized data package is provided. The data package of the new prices for the subset of products is provided to the stores. The prices for the products in the subset, which is a smaller set than all of the products, are changed (set) according to the optimized subset prices.

The subset optimization may be repeated several times between a complete optimization allowing all prices to change. For example, after subset optimization, new data may be provided. A user may specify new subset limits or use the previous subset limits. A new subset optimization may then be performed, using the specified subset limits and holding any frozen prices to the prices provided by the previous subset optimization.

In addition to the constraints listed above, the preferred embodiment may model several other business rules via constraints. These include limits on group price advance or decline, brand pricing rules, size pricing rules, and line pricing rules, as discussed above.

INTELLIGENT CLUSTERING SYSTEM

In examining purchase data, it is found that geographical price zones (clustering by geography alone) may reduce profits by as much as 60%.

To facilitate understanding FIG. 16 is a broad flow chart of an embodiment of the invention. Store specific information is collected (step 1604). Such store specific information may be individual store point-of-sales data, cost data, or customer survey data. The store specific information is used and processed to provide optimal combinations fore each individual store (step 1608). In the specification and claims, the use of the word "optimal" may mean a value or combination or selection that is

determined through the use of an algorithm as the best result found by the algorithm. Since optimization may be a complex process, the optimal result may not be the absolute best result but may be a localized best result. Clusters of stores are then formed based on the closeness of the optimal combinations for each store (step 1612). New cluster based combinations are then provided (step 1616).

To facilitate discussion, FIG. 11 is a flow chart of a process intelligently clustering stores, which is an example of an embodiment of the invention. In this example store specific point-of-sales and cost data has already been collected. First, an optimization is performed using the point-of-sales and cost data(step 1104). Preferably, the same optimization is performed for all stores. In this preferred embodiment, the prices are provided on a store-by-store basis, instead of by groups of stores. In addition, in this example the prices are also provided on an item-by-item basis. The result of this optimization provides an optimal product and price combination for each store.

FIG. 12 is a more detailed flow chart of a preferred embodiment of step 1108 for creating clusters (groups) based on the closeness of the optimized product and price combinations for each store. Constraints may be provided (step 1204). Examples of such constraints are prohibiting certain stores from being within the same cluster, forcing certain stores to be in the same cluster, setting a maximum number of clusters, and setting a minimum cluster size. The forcing certain stores to be in the same cluster may be used to place geographically close stores in the same cluster or to place stores with similar traits, such as being in a downtown area or near a freeway exit, in the same cluster.

Stores with the closest prices and which meet the constraints are then grouped together in clusters (step 1208). There may be many different approaches to clustering stores with the closest prices. Such clustering may attempt to minimize the sum of differences or distances in prices between stores in a cluster.

FIG. 13 is a chart of an example of four stores, which are formed into two clusters, with each store carrying item X and item Y. Table 1 below provides a list of the prices for items X and Y per store.

TABLE 1

STORE\ITEM	ITEM X	ITEM Y
STORE 1	$6.00	$10.00
STORE 2	$6.00	$9.00
STORE 3	$9.00	$7.00
STORE 4	$8.00	$6.00

The prices are plotted in FIG. 13 for store 1 (1304), store 2 (1308), store 3 (1312), and store 4 (1316) with the price of item X plotted along the horizontal axis and the price of item Y plotted along the vertical axis. A distance may be calculated by: $\text{Distance} = \sqrt{(\text{Price}_{S,X} - \text{Price}_{S,X})^2 + (\text{Price}_{S,Y} - \text{Price}_{S,Y})^2}$, where s is for one of the stores. This distance may be calculated between all possible store pair combinations. Table 2 illustrates the distances for all pairings.

TABLE 2

	DISTANCE
STORE 1 AND STORE 2	1
STORE 1 AND STORE 3	4.24
STORE 1 AND STORE 4	4.47
STORE 2 AND STORE 3	3.61
STORE 2 AND STORE 4	3.61
STORE 3 AND STORE 4	1.41

Therefore, the total distances for each possible cluster pairings are shown in Table 3.

TABLE 3

CLUSTER	TOTAL DISTANCE
1 WITH 2 AND 3 WITH 4	2.41
1 WITH 3 AND 2 WITH 4	7.85
1 WITH 4 AND 2 WITH 3	8.08

The store pair combination with the lowest distance would be chosen as the best cluster pairing. From Table 3, it can be seen that by putting stores 1 and 2 in one cluster and stores 3 and 4 in another cluster, the total distances between stores within the same cluster are minimized, and therefore the stores are clustered accordingly.

These distances may be weighted according to volume sales by multiplying the distances by revenue.

Various algorithms may be used to actually form clusters. One example of such clustering is the k-means clustering method. The FASTCLUS procedure available in SAS, provided by the SAS Institute, Inc of Cary, North Carolina is one procedure that may be used to create the clusters.

If a store does not sell a particular product, an average price for the product may be calculated and used as the price of the product for the store that does not sell the product, so that the absence of the product does not affect the clustering of the stores. If a constraint requires that two or more stores be placed in the same cluster, their prices may be averaged together and the average price may be used as individual store prices for these stores, so that the clustering algorithm automatically places these stores in the same cluster, since their prices are identical.

After the clusters are created (step 1108), prices for items in the store are set by cluster so that stores within the same clusters have items with the same prices (step 1112). Setting the product and price combinations by cluster is an example of providing cluster based combinations (step 1616). Preferably, the new prices may be obtained by performing a new optimization, where prices are calculated by cluster instead of by store. Such an optimization might be weighted by volume of sales for each store. Table 4 is an example of such an optimization by cluster.

TABLE 4

STORE\ITEM	ITEM X	ITEM Y
STORE 1	$6.00	$9.60
STORE 2	$6.00	$9.60
STORE 3	$8.50	$6.50
STORE 4	$8.50	$6.50

The prices for the items in store 1 are the same as the prices for store 2, since they are in the same cluster. Similarly, the prices for the items in store 3 are the same as the prices for store 4, since they are in the same cluster. In this example, the prices for the cluster are close to and in between the prices for the individual stores in the cluster derived when the optimization was done on a store-by-store basis. By providing clusters so the prices for each cluster are as close as possible to the prices for an optimization done on a store-by-store basis, the optimization by cluster is kept as close to the optimization on a store-by-store basis, which minimizes the loss of optimization opportunity.

Over time, as more sales data is collected, the new data may be added to iteratively change the clusters over time. Such iteration may limit the number of stores that can change clusters or may have a similar constraint.

In an embodiment of the invention, each store may have a plurality of product categories. Clustering may be according to product categories, so that a store may be in one cluster for one product category and in another cluster for another product category. This clustering may allow more constraints. For example, a product category with perishable items may have a constraint so that stores within a cluster are relatively close together, while a product category with non-perishable items may not have such a constraint.

In another embodiment of the invention, it may be found that for some product categories there may be more products available than what may be offered in each store. For example, there may be more than 500 different soap products, although the stores may only have room to sale 200 products. An optimization engine as described

above may be used to determine the most profitable combination of products for each store. A chain of 100 stores may have a hundred product combinations. It may be desirable to provide five clusters of product combinations. This embodiment of the invention would provide clusters of stores according to product combinations (assortment zones) for an entire store or a product category. FIG. 14 is a flow chart on a broader embodiment of the invention. Store specific point-of-sales or survey data may be collected. Using the above example of soap products, an optimization may be performed using the store specific data to determine a combination of 200 products of the 500 soap products that would optimize sales on a store by store bases for each of the 100 stores (step 1404). The stores are then clustered into five clusters so that each cluster that a store is placed has the most similar combination of products (step 1408). Stores are then provided an assortment of products (product combinations) according to the cluster that the store is in (step 1412). As mentioned before, such clustering would have lower sales and/or profit than providing individual product combinations for each individual store, but would be designed to maximize sales and/or revenue for different combinations of five clusters. In one embodiment, the prices are set or optimized on a store by store basis. In another embodiment, an optimization may be done where stores in the same assortment zone form price clusters. Stores may be placed in an assortment cluster so that all stores in the same assortment cluster receive the same assortment and also in a price cluster so that all stores in the same price cluster have the same prices.

FIG. 15 is a flow chart of another embodiment of the invention. In this example store specific information may be collected by a survey (step 1554). The survey data is used to provide a store by store promotion combination optimization (step 1558). In one example, 200 products may discussed in the survey. A promotion combination may be a combination of 20 of the 200 products at different discounts and displays. It may be found that each store of 100 stores in a chain has a different promotion combination optimization. Promotion clusters are created to cluster the stores according to how close their promotion combinations are (step 1562). Promotion combinations for each store are then determined by the cluster that the store is in (step 1566)

Other embodiments may be generated according to margins or by metric or according to direction of movement.

SIGNIFICANCE OF PRICE CHANGE INDICATOR

To facilitate discussion, FIG. 17 is a flow chart of a process providing a price optimization that uses an indicator of significance of a price change. First, an optimization for an objective is performed (step 1704). The optimization may determine optimal prices to obtain an objective, such as maximizing profit, maximizing sales volume, or providing a promotion. The plurality of values is used to generate an indicator of significance of price change (step 1708). In one embodiment, the indicator is a price image metric, which provides a measure of an image impact of a change in price. Preferably the image metric is based on price change, price elasticity and demand (or revenue as its proxy), so that the values provided by optimization and demand equation are optimized prices, elasticities, and the initial revenues of each item to be used in the image metric. In a preferred embodiment, an image metric equation is:

$$\text{Image Metric} = \%\Delta EqP_{i,s}\left(\left(\left(\frac{|\varepsilon_{i,s}|}{|\bar{\varepsilon}_s|}\right)w_\varepsilon\right)+\left(\left(\frac{R_{i,s}}{\bar{R}_s}\right)w_R\right)\right),$$

where EqP = equivalent price, ε = price elasticity, i = item, s = store, w = weight, and R = revenue (initial). The equivalent price of a product is defined as the price of a standardized unit of measure, which may be calculated based on the product description and the spread of sizes/counts that apply to that description. The metric uses the change in equivalent price (%ΔEqP) for a particular item (i) for a particular store (s), which is multiplied by the sum of two weighted index values. The first term of the sum is a normalized elasticity value, which in this example is the absolute value of the elasticity for an item in a store divided by the average elasticity for a range of items in the store multiplied by an elasticity weight value. The second term of the sum is a normalized revenue value, which in this example is the revenue for an item in a store divided by the average revenue for a range of items in the store multiplied by a revenue weight value.

In an example of the working of this equation, an item A may have an original price of $1.00. As a result of an optimization, a new price of $1.10 is provided. This would cause %ΔEqP=10%. Preferably, the elasticity of the item, the average elasticity, revenue of the item, and average revenue may be initial revenue determined from point-of-sale information. In this example, the magnitude of the elasticity of the

item is 15% higher than the magnitude of the average elasticity, so that $\frac{|\varepsilon_{i,s}|}{|\bar{\varepsilon}_s|}=1.15$. If the weight of the elasticity w_ε is 0.4, then $\left(\frac{|\varepsilon_{i,s}|}{|\bar{\varepsilon}_s|}\right)w_\varepsilon=(1.15)(0.4)=0.46$. In this example, the revenue for the item at the store increases by 20% over the average revenue, so that $\left(\frac{R_{i,s}}{\bar{R}_s}\right)=1.20$. The weight of the revenue w_R is 0.6, so that $w_\varepsilon+w_R=1$. Therefore $\left(\frac{R_{i,s}}{\bar{R}_s}\right)w_R=(1.20)(0.6)=0.72$. Therefore,

$$\%\Delta EqP_{i,s}\left(\left(\left(\frac{|\varepsilon_{i,s}|}{|\bar{\varepsilon}_s|}\right)w_\varepsilon\right)+\left(\left(\frac{R_{i,s}}{\bar{R}_s}\right)w_R\right)\right)=10\%(0.46+0.72)=11.8\%.$$

This metric would then indicate that the increase in price would penalize the image metric of the store by 11.8%.

A large positive percentage change in price would more significantly increase the image metric than a smaller percentage, which indicates that the image would be hurt more. A negative percentage change in price would indicate an improved price image. In addition, a more highly elastic or higher revenue item also would provide a greater effect on the price image.

This metric may be averaged over all items in a range and over a store range to provide an average image metric. This average may be weighted by volume or revenue contribution. The range of items and stores may be the range of items and stores over which the optimization was performed. In the alternative, a select group of items and/or stores may be selected as the range used to provide the image metric.

A reoptimization decision step is then made (step 1712). FIG. 19 is a more detailed flow chart of the reoptimization decision step in an embodiment of the invention. In this embodiment, the indicator of significance of price change and the optimized prices are provided to a client, such as the stores (step 1904). The prices and indicator of significance of price change may be provided to the stores through the support tool 116. Feedback is then received from the client regarding the significance of price change (step 1908). Such feedback may be a request for a lower

price image metric or that certain price changes be constrained. The feedback may be received through the support tool 116. A decision is made concerning whether or not the feedback requires reoptimization (step 1912). Generally, feedback requiring a reduction of the image metric in this example or a lowering of prices would require a reoptimization, where no change in the image metric and optimized prices would not require a reoptimization. Such feedback may also be used for the initial optimization.

If a reoptimization is desired, an optimization is performed using the significance of price change (step 1716), which includes using feedback regarding the significance of price change and price change in the reoptimization. The feedback may be used to provide a constraint for the reoptimization. A new indicator of significance of price change may be provided from the reoptimization (step 1708). The new indicator of significance of price change and prices may then be provided to the client through the support tool 116, until the client is satisfied with the significance of price changes and the optimized prices. The client may then set prices to the optimized prices (step 1720) and the process is completed.

In another embodiment, unit price may be used instead of the equivalent price to generate an image price metric. In another embodiment, category price may be used.

In another embodiment, a competitive image metric may be calculated as the indicator. The competitive metric provides a measure of a competitive impact of a change in price. Preferably, the competitive metric is based on price elasticity and revenue. In a preferred embodiment, a competitive metric equation is:

$$\text{Competitive Image Metric} = \%\Delta EqP_{i,s}\left(\left(\left(\frac{|\varepsilon_{i,s}|}{|\bar{\varepsilon}_s|}\right)w_\varepsilon\right)+\left(\left(\frac{R_{i,s}}{\bar{R}_s}\right)w_R\right)\right),$$

where *EqP* = equivalent price, ε = price elasticity, *i* = item, *s* = store, *w* = weight, and *R* = revenue (initial). The metric uses the difference between the equivalent price for a particular item (i) for a particular store (s) and for the same item for a competitor (%ΔEqP), which is multiplied by the sum of two weighted index values. The first term of the sum is a normalized elasticity value, which in this example is the absolute value of the elasticity for an item in a store divided by the average elasticity for a range of items in the store multiplied by an elasticity weight value. The second term of the sum is a normalized revenue value, which in this

example is the revenue for an item in a store divided by the average revenue for a range of items in the store multiplied by a revenue weight value.

In another embodiment of the invention, a risk assessment attribute is used as an indicator of significance of price change. In this embodiment, the risk assessment attribute is an indicator of the price change of an item. Such an indicator of price change may use actual price change, percent price change, equivalent price change, percent equivalent price change, or some other similar price change measure.

FIG. 18 is a flow chart of a process that may be used to provide a risk assessment attribute. Attribute ranges are set (step 1804). In one example, five ranges may be provided and ranked from 1 to 5. If a percent price change is used, a user may place price changes of greater than 50% in range 5, percent price changes from 20% to 50% in range 4, percent price changes from 0% to 20% in range 3, percent price changes from -20% to 0% in range 2, and percent price changes for -50% to -20% in range 1. After the optimization (step 1704), in this embodiment the indicator of the significance of the price change (step 1708) is provided by calculating the percentage price changes of each item and placing the percent price changes in the set ranges (step 1808). A graphical indicator is then used to indicate the range that an item is placed. Such a graphical indicator may be a color coding. An example of such a use of color may be by providing a table of optimized prices according to item. The prices, item, or background may be colored according to the range that the percentage price change for the item is placed. The use of color allows a user scanning a table of optimized prices to quickly see which items had a percent price change greater than 50% (range 5) or between 25% to 50% (range 4). In another embodiment, a bar graph or an icon may be used to graphically indicate ranges.

In another embodiment of the invention, the risk assessment attribute may use a volume weighted price change. If the image metric were applied to a group of products, such as a product that comes in different sizes, a volume weighted average may be used to generate an image metric for the product using sales of all the different sizes.

In another embodiment of the invention, the indicator, such as an image metric, may be used as a constraint in the optimization using the indicator of significance of price change (step 1716). A rule used in the optimization may specify a range for the indicator. The optimization using the indicator of significance of price change may then be performed keeping the indicator within the specified range. If the

optimization is iterated over several cycles, the indicator of significance of price change may be changed over each cycle as new values for price change and elasticities are calculated, causing the constraint to change every iteration.

FIG. 20 is a flow chart of another a process of obtaining an image metric. A price optimization is performed (step 2004). Such price optimization may be a price optimization as described above. The optimization and demand model are used to generate price differences and demand elasticities (step 2008). Other values such as revenues, margins, and sales volumes may also be generated. The price differences and elasticities are used to generate an image metric (step 2012). The generation of the image metric may use the above equations for the image metric or the competitive image metric. Such equations may use revenues to generate the image metric.

FIG'S. 9A and 9B illustrate a computer system 900, which is suitable for implementing embodiments of the present invention. FIG. 9A shows one possible physical form of the computer system. Of course, the computer system may have many physical forms ranging from an integrated circuit, a printed circuit board, and a small handheld device up to a huge super computer. Computer system 900 includes a monitor 902, a display 904, a housing 906, a disk drive 908, a keyboard 910, and a mouse 912. Disk 914 is a computer-readable medium used to transfer data to and from computer system 900.

FIG. 9B is an example of a block diagram for computer system 900. Attached to system bus 920 are a wide variety of subsystems. Processor(s) 922 (also referred to as central processing units, or CPUs) are coupled to storage devices, including memory 924. Memory 924 includes random access memory (RAM) and read-only memory (ROM). As is well known in the art, ROM acts to transfer data and instructions uni-directionally to the CPU and RAM is used typically to transfer data and instructions in a bi-directional manner. Both of these types of memories may include any suitable of the computer-readable media described below. A fixed disk 926 is also coupled bi-directionally to CPU 922; it provides additional data storage capacity and may also include any of the computer-readable media described below. Fixed disk 926 may be used to store programs, data, and the like and is typically a secondary storage medium (such as a hard disk) that is slower than primary storage. It will be appreciated that the information retained within fixed disk 926 may, in

appropriate cases, be incorporated in standard fashion as virtual memory in memory 924. Removable disk 914 may take the form of any of the computer-readable media described below.

CPU 922 is also coupled to a variety of input/output devices, such as display 904, keyboard 910, mouse 912 and speakers 930. In general, an input/output device may be any of: video displays, track balls, mice, keyboards, microphones, touch-sensitive displays, transducer card readers, magnetic or paper tape readers, tablets, styluses, voice or handwriting recognizers, biometrics readers, or other computers. CPU 922 optionally may be coupled to another computer or telecommunications network using network interface 940. With such a network interface, it is contemplated that the CPU might receive information from the network, or might output information to the network in the course of performing the above-described method steps. Furthermore, method embodiments of the present invention may execute solely upon CPU 922 or may execute over a network such as the Internet in conjunction with a remote CPU that shares a portion of the processing.

In addition, embodiments of the present invention further relate to computer storage products with a computer-readable medium that have computer code thereon for performing various computer-implemented operations. The media and computer code may be those specially designed and constructed for the purposes of the present invention, or they may be of the kind well known and available to those having skill in the computer software arts. Examples of computer-readable media include, but are not limited to: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD-ROMs and holographic devices; magneto-optical media such as floptical disks; and hardware devices that are specially configured to store and execute program code, such as application-specific integrated circuits (ASICs), programmable logic devices (PLDs) and ROM and RAM devices. Examples of computer code include machine code, such as produced by a compiler, and files containing higher level code that are executed by a computer using an interpreter. Computer readable media may also be computer code transmitted by a computer data signal embodied in a carrier wave and representing a sequence of instructions that are executable by a processor.

FIG. 10 is a schematic illustration of an embodiment of the invention that functions over a computer network 800. The network 800 may be a local area network (LAN) or a wide area network (WAN). An example of a LAN is a private network used by a mid-sized company with a building complex. Publicly accessible WANs include the Internet, cellular telephone network, satellite systems and plain-old-telephone systems (POTS). Examples of private WANs include those used by multi-national corporations for their internal information system needs. The network 800 may also be a combination of private and/or public LANs and/or WANs. In such an embodiment the price optimizing system 100 is connected to the network 800. Computer systems used by the stores 124 are also connected to the network 800. The computer systems for the stores 124 are able to bi-directionally communicate with the price optimizing system 100 over the network 800.

While this invention has been described in terms of several preferred embodiments, there are alterations, modifications, permutations, and substitute equivalents, which fall within the scope of this invention. It should also be noted that there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, modifications, permutations, and substitute equivalents as fall within the true spirit and scope of the present invention.

CLAIMS

What is claimed is:

1. An apparatus for computing a preferred set of prices for a subset of a plurality of products, comprising computer readable media, comprising:

computer readable code for storing initial prices for a plurality of products;

computer readable code for designating a subset of products of the plurality of products, wherein the number of products in the subset of products is less than the number of products in the plurality of products; and

computer readable code for optimizing prices for products in the subset of products, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

2. The apparatus, as recited in claim 1, wherein the computer readable code for designating a subset, comprises:

computer readable code for allowing a number N to be designated; and

computer readable code for selecting no more than N products of the plurality of products to form the subset of products.

3. The apparatus, as recited in claim 2, wherein the computer readable code for selecting no more than N products selects products that provide the greatest optimization for any set of N products of the plurality of products.

4. The apparatus, as recited in any of claims 1-3, further comprising computer readable code for providing initial prices by optimizing prices for all of the plurality of products.

5. The apparatus, as recited in claim 4, further comprising computer readable code for providing new data subsequent to providing initial prices by optimizing prices.

6. The apparatus, as recited in claim 5, wherein the computer readable code for providing new data comprises computer readable code for providing new price data and computer readable code for providing new bound data.

7. The apparatus, as recited in any of claims 1-6, further comprising computer readable code for providing rule relaxation.

8. The apparatus, as recited in claim 7, wherein the computer readable code for providing rule relaxation comprises:

computer readable code for allowing the prioritization of a plurality of rules; and

computer readable code for relaxing at least one lower priority rule to allow a higher priority rule to become feasible.

9. A method for computing a preferred set of prices for a subset of products of a plurality of products, comprising:

storing initial prices for a plurality of products;

designating a subset of products of the plurality of products, wherein the number of products in the subset of products is less than the number of products in the plurality of products; and

optimizing prices for products in the subset of products, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

10. The method, as recited in claim 9, wherein the designating a subset comprises:

allowing a number N to be designated; and

selecting no more than N products of the plurality of products to form the subset of products.

11. The method, as recited in claim 10, wherein the selecting no more than N products selects products that provide the greatest optimization for any set of N products of the plurality of products.

12. The method, as recited in any of claims 9-11, further comprising providing initial prices by optimizing prices for all of the plurality of products.

13. The method, as recited in claim 12, further comprising providing new data subsequent to providing initial prices by optimizing prices.

14. The method, as recited in claim 13, wherein the new data comprises new price data and bound data.

15. The method, as recited in any of claims 9-14, further comprising providing rule relaxation.

16. A method for setting prices for a subset of products of a plurality of products, comprising:

receiving optimized prices for a product category;

pricing every item in the product category according to the received optimized prices;

providing new data

receiving new prices for the subset of products of the product category, wherein the subset is smaller than the product category, wherein the received new prices are generated by storing initial prices for a plurality of products, designating a subset of products of the plurality of products, wherein the number of products in the subset of products is less than the number of products in the plurality of products, and optimizing prices for products in the subset of products, while freezing the initial prices of products of the plurality of products in the product category that are not in the subset of products; and

setting prices for the subset of products according to the received new prices.

17. A computer data signal embodied in a carrier wave and representing sequences of instructions which when executed by a processor, causes the processor to compute a preferred set of prices for a subset of a plurality of products, by performing the steps comprising:

storing initial prices for a plurality of products;

designating a subset of products of the plurality of products, wherein the number of products in the subset of products is less than the number of products in the plurality of products; and

optimizing prices for products in the subset of products, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

18. A price database generated by the method comprising:

storing initial prices for a plurality of products;

designating a subset of products of the plurality of products, wherein the number of products in the subset of products is less than the number of products in the plurality of products; and

optimizing prices for products in the subset of products, while maintaining the initial prices of products of the plurality of products that are not in the subset of products.

19. A method for obtaining optimized price data on a client system, comprising the steps of:

sending sales data to a server system for a plurality of products;

selecting optimization preferences;

transmitting said optimization preferences to said server system;

receiving from said server system optimization prices for all of the plurality of products;

sending additional sales data to the server system;

selecting a subset constraint;

sending the subset constraint to the server system; and

receiving from the server system a new set of optimization prices for a subset of the plurality of products which is less than the plurality of products.

20. A method for forming a plurality of stores into a plurality of clusters, comprising:

collecting store specific information;

providing optimized combinations for each individual store based on the store specific information; and

creating a plurality of clusters based on the closeness of the optimal combinations.

21. The method, as recited in claim 20, further comprising providing cluster based combinations.

22. The method, as recited in any of claims 20-21, wherein the store specific information is selected from a group comprising point-of-sales data, customer survey data, and cost data.

23. The method, as recited in any of claims 20-22, wherein the combinations is selected from a group comprising item and price, assortment, and promotion combinations.

24. The method, as recited in any of claims 20-23, wherein the creating the plurality of clusters, comprises:

providing at least one constraint; and

placing stores that meet the constraints and with the closest optimal combinations in the same cluster of the plurality of clusters.

25. The method, as recited in any of claims 24, wherein the at least one constraint places two stores in the same cluster, by making each store of the two stores have the same optimal combination.

26. The method, as recited in any of claims 24-25, wherein the at least one constraint specifies a maximum number of clusters.

27. An apparatus for forming a plurality of stores into a plurality of clusters, comprising computer readable media, comprising:

computer code for collecting store specific information;

computer code for providing optimized combinations for each individual store based on the store specific information; and

computer code for creating a plurality of clusters based on the closeness of the optimal combinations.

28. The apparatus, as recited in claim 27, further comprising computer code for providing cluster based combinations.

29. A method for providing prices for items, comprising:

performing at least one iteration in a price optimization and generating a plurality of values;

using the generated plurality of values to generate an indicator of significance of price change; and

performing at least one iteration in an indicator of significance of price change using price optimization to obtain optimized prices for the items.

30. The method, as recited in claim 29, wherein a first set of the plurality of values comprises price differences, wherein the price differences are used to generate the indicator of significance of price change.

31. The method, as recited in claim 30, wherein a second set of the plurality of values comprises elasticities, wherein the elasticities are used to generate the indicator of significance of price change.

32. The method, as recited in claim 31, wherein a third set of the plurality of values comprises revenues for each item, wherein the revenues are used to generate the indicator of significance of price change.

33. The method, as recited in any of claims 29-32, wherein the performing the at least one iteration in an indicator of significance of price change using price optimization uses the indicator of significance of price change as a constraint.

34. The method, as recited in any of claims 30-33, wherein the price differences are the differences between optimized prices and initial prices.

35. The method, as recited in any of claims 30-33, wherein the price differences are the differences between the optimized prices and competitor prices.

36. The method, as recited in any of claims 29-35, wherein the using the determined plurality of values to generate an indicator of significance of price change comprises providing a graphical indicator to indicate a range in which a price indicator is placed.

37. The method, as recited in any of claims 29-36, further comprising the step of allowing a user to set a plurality of price indicator ranges.

38. The method, as recited in any of claim 36, wherein the graphical indicator is provided by using a different color for each range.

39. The method, as recited in any of claims 29-38, wherein the performing the at least one iteration in the price optimization comprises,

collecting point-of-sales data;

modeling the point-of-sales data;

modeling cost data; and

providing an optimization based on the modeled point of sales data and cost data.