RC23084 (W0401-127) January 27, 2004
Computer Science

IBM Research Report

Visualizing Costs of Customer Service Strategies

Donna Gresh
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Eugene Kelton
IBM Integrated Technology Services
2020 Technology Parkway
Mechanicsburg, PA 17050

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Visualizing Costs of Customer Service Strategies

Donna Gresh
IBM T.J. Watson Research Center

Eugene Kelton
IBM Integrated Technology Services

***Abstract*— We have created a visualization application to help in formulating business strategy in the customer service arena. Currently, stocking levels at hardware replacement part depots are optimized to find the lowest cost solution, given a particular set of customer service targets. The targets are treated as inputs to the optimization, as it is impossible as a practical matter to express the benefit of a particular service strategy mathematically. Visualization is used to allow an intuitive understanding of the relative costs of different service strategies, thus allowing cost-effective targets to be chosen for input to the optimization algorithms.**

***Index Terms*— visualization, information visualization, optimization, VisAD**

I. Introduction

OUR users have the responsibility for delivering computer hardware support to their customers. They need to do so cost-effectively, while simultaneously ensuring that customers receive the high level of service they expect. Because IBM services a wide range of hardware platforms (not even all of them made by IBM), the number of different parts to be stocked is huge (exceeding 20,000). Because the customer expects rapid resolution of their hardware failure (in a time frame on the order of hours for critical failures), the parts cannot simply be stocked in a central warehouse to be flown to the customer location when necessary. Thus a relatively large number of parts depots are necessary (currently approximately 150 in the United States alone), and an important problem to solve is how many of a given part to store at each parts depot. Because some of the parts are quite expensive, costing in excess of $10,000, it is important to choose the stocking levels well: sufficient to satisfy projected demand, but not so much that excessive inventory is maintained. We must also keep in mind that because of constantly evolving technology, parts become obsolete over time, so that an expensive part may become worthless in a matter of a year or two.

The stocking problem is typically considered in two separate phases. First, a set of target service levels that is likely to be satisfactory to customers is established. Service targets are a promise to deliver a customer service representative (CSR) and a piece of hardware to a customer site within a very short period of time. For example, for a given class of critical machine parts, a set of target service levels might be

- 85% of customers receive the CSR and part within 2 hours
- 90% of customers receive the CSR and part within 4 hours
- 95% of customers receive the CSR and part within 12 hours

(Given overnight delivery, essentially 100% of customers can be serviced in the 24 hour time frame).

Second, given this set of target service levels, an optimization algorithm determines how much hardware stock should be inventoried, and where. Depending on the resultant costs, there may need to be some iteration on the chosen target levels. However this can only be done in a limited way, because the time needed to run a complete optimization solution is several hours. NOTE: POSSIBLE SIDEBAR ON OPTIMIZATION

Solving the stocking problem requires the use of sophisticated optimization techniques that have been developed previously by IBM. One needs to take into account the transportation infrastructure of a particular geographic region, the interrelation of different machine classes which might share a given part, the cost of moving the part from one depot to another, inventory expense, and other constraints. IBM's solution entails executing a large optimization run each week, the output of which specifies which new parts should be inventoried, and where. It may also recommend transferring parts from one depot to another. The use of sophisticated optimization algorithms, coupled with a "neighborhood" stocking strategy (in which multiple depots within a given radius of a customer location can be called on to provide a part) has led to the saving of approximately $5 million dollars a year for IBM, while maintaining or improving levels of customer service.

While the neighborhood optimization approach has already proven valuable, it does not by itself solve the entire business problem. It minimizes costs for a given set of service target goals, but provides no guidance for how those goals should be set in the first place. Due to the complexity of the problem, two different sets of service level targets that may be roughly equivalent to customers (*e.g.* a 2-hour target of 80% and a 4-hour target of 87%, *vs.* a 2-hour target of 75% and a 4-hour target of 98%) may differ in cost by a factor of two or three (in either direction). Thus the service strategist has a need to understand the "landscape" of cost with respect to service targets, so that the targets can be set with an eye to the likely effect on final cost. To address this need, we have created a visualization tool that allows a service strategist to

gresh@us.ibm.com, IBM T.J. Watson Research Center, 35-210, 1101 Kitchawan Road, Yorktown Heights, NY 10598

kelton@us.ibm.com, IBM Integrated Technology Services, 2020 Technology Parkway, Mechanicsburg, PA, 17050

understand this landscape. Using this tool, one can set cost-effective target levels, and balance a limited budget across the various machine classes, service regions, and customer service goals.

Visualization is necessary in this application because we do not have a straight optimization problem: the user can't mathematically specify the entire objective function. There is some benefit to the customer for a given service level. However this benefit is hard to quantify mathematically, and the customer or service strategist certainly doesn't want to think about the value of *each* of the possible combinations of service levels. In order to solve the complete using optimization, (that is, determine both the optimal service targets *and* the optimal stocking strategy given those targets), it would be necessary to put a dollar value on each combination of customer service targets so that the "best" solution could be found (that which maximizes the quantity (benefit−cost). However the users would rather see the costs first, so they can focus their thought on the region of space that is likely to matter, *i.e.*, where the costs seem reasonable with respect to the service level. Thus visualization fills the gap between the user's overall goal of finding the best combination of service targets and the ability to mathematically formulate the problem.

An outline of this paper is as follows: Section II describes our design goals, and why other systems fall short of meeting our design criteria. Section III describes the application we have developed, discussing both the landscape views and the information visualization views included. Section IV describes some of our planned future work. Finally, Section V summarizes the contributions of this work.

II. Design Goals and Related Work

In terms of the interface design, an important goal was to have the visual representations lead naturally from the table views of the data with which the service strategists are familiar. A common and necessary interaction mechanism is to be able to investigate data at various levels of "roll-up." That is, the strategists have the desire to look at costs across parts of all machines of a particular type, across all of a particular geographic region, across all of a particular vitality (importance), or across some user-specified combination of these. The interface must provide these choices in a natural and intuitive way, and the visual representations must seem as natural and intuitive as the table views. Thus we use the familiar table view as the "control widget" to any resulting visual representations. SQL queries to a database of cost information are launched by interaction with the tables, which in turn populate the visual representations.

Second, it was necessary to design the application so that a strategist using it can not only see the landscape of cost, but also use the insight obtained to investigate the effect of new targets on the aggregate cost. In this way, the tool can become a part of the budgeting process, as tradeoffs may need to be made in one machine class *vs.* another. Interactive picking can be used to select new target points in the landscape to feed back into the table views of cost. Thus the strategist can interactively choose good candidate target levels using the visual presentations, and immediately see the effect of using those targets on the overall budget.

While various commercial OLAP packages are available to present visual representations of information in relational databases (*e.g.* DataDesk [1] and Alphablox [2]), by and large their visualization capabilities remain of the simple charting variety in order to support general relational data, the columns of which may have an arbitrary relationship to one another. For the work described here, we can exploit the fact that a *spatial* representation of the underlying data makes sense and is of value. That is, costs can be expected to have a smooth and continuous relationship to the variables of target service levels. Since our users want to understand the landscape of cost *vs.* (primarily) two variables, two-hour service target and four-hour service target, we decided to visually represent the functional relationship of "*xy*" to "data," as a surface in three dimensions. Thus we were led to using a scientific visualization toolkit, which we integrated with database access capability. We also needed to be able to customize the visual presentations, offering custom colormaps, gridlines, marker points, etc. In addition, we found that because our users are unfamiliar with three-dimensional interaction mechanisms, we needed to be able to automatically set particular viewpoints rather than require the user to use the mouse to reach those views. A scientific visualization toolkit was ideal for this purpose.

We note that the choice of a *particular* scientific visualization toolkit was made based on our own desire to use the Java programming language, and on general functionality and usability, but that other toolkits (*e.g.* AVS [3] or Data Explorer [4]) could certainly also have been used. In any case, regardless of toolkit, significant amounts of custom code would be necessary to build the various requirements into our application, even when starting with a commercial product.

Other systems have also been designed to visualize the data in relational databases. Some examples are Polaris [5], DEVise [6], and Rivet [7]. These systems define visual presentations which can be applied to general tabular and relational data. The Polaris system, in particular, develops an algebra of relationships which can be used to automatically create representations, which are tightly linked to the concepts of SQL queries. However, besides not being commercially available, Polaris would not provide the specific visual representations appropriate to our data, in particular the three-dimensional cost landscape which we will discuss.

An obvious question is how much of the landscape representation we require could be provided with other standard charting tools. While built-in charting features, such as those in Excel, can ostensibly render surfaces in three-dimensions, they are generally extremely limited in functionality, interactivity, and extensibility. For example, in this application, the surface to be realized is a triangulation of an unevenly sampled set of points in two-dimensions, which is not supported by the surface chart function in Excel. To create a rendering somewhat like the desired effect using Excel, it would be necessary to resample the actual data values to a regular grid; missing data values would not be rendered properly (two hour service target cannot realistically be greater than the four hour

service target), and there would be no interactivity in viewpoint, picking, or other custom features commonly provided by three-dimensional rendering and visualization libraries. Moreover, our users desired a fully customized application, not an Excel worksheet. The amount of data generated and to be considered is large, and, in order to allow the desired query and aggregation capabilities, must reside in a database. The users also want to be able to use the application to directly modify the target service levels in their production database, after visual investigation. Thus we were led to consider custom visualization tools which could be integrated into a complete application.

III. The SOS Application

The Service Offering Simulator (SOS) system consists of two important components. The first is the data generation component, which uses special techniques in order to create the large amount of data necessary to fully explore the landscape of costs and service. Since the full operational run of the entire model, consisting of a *single* set of input target levels, takes on the order of hours to compute, a simplistic brute-force approach to creating the thousands of data points necessary is simply infeasible. In our solution, the full model is broken down into separate subproblems which may be assumed to be at least roughly independent. Then an initial starting point for target service level is chosen and the smaller model is run for that set of targets. In contrast to the operational run, where integral solutions must be found (because, of course, it is necessary to determine an *integral* number of parts to move, or to obtain), in the simplified model, stocking values are allowed to be non-integral. This allows an incremental approach to be used to solve each new problem, where targets differ slightly from the previous solution. In addition to making the solution tractable, there are other justifications for using a non-integral approach here, including the fact that integral solutions would lead to discontinuities in cost (*i.e.*, a significantly "lumpy" surface) that would serve to distract rather than enlighten the user, since a particular part acquired this week may not need to be acquired for another 10 weeks, on average. Thus an "average" number of parts acquired is the more relevant value.

For the data set discussed here, there are approximately 800 smaller problems, covering the entire United States. For each of these we compute the cost for approximately 250 combinations of service targets. Based on current priorities for our users, we focus on the landscape of costs for the 2 hour and 4 hour service targets. A chart showing the coverage is shown in Figure 1. Note that the target four hour service is of course always greater than or equal to the target two hour service, and that we sample more finely in the regions of high service levels (which is where costs are likely to increase rapidly).

The second part of SOS is the visualization interface. The interface accesses the generated data through SQL queries to a IBM DB2 database containing the results of the optimization runs. The queries populate all of the views in the interface. The SOS application is built in Java, and incorporates visual presentations from two different visualization libraries. The

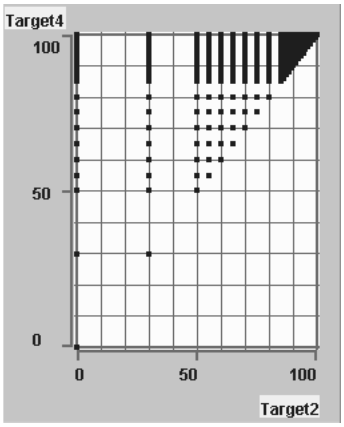

Fig. 1. Coverage of service level targets, for the combinations of target service within 4 hours and target service within two hours. More cost points are computed at the high percentage target service levels (above 85%), where costs increase rapidly.

first library used is the VisAD visualization library [8], an open source package with a sophisticated and well-developed data model. VisAD uses Java3D for interactive three-dimensional visualization on Windows, Linux, and SunOS operating systems. The second is the Opal information visualization library, developed internally within IBM research (and discussed in Gresh *et. al* [9]). Opal provides a variety of automatically linked views of tabular data.

The interface window which opens first for the user is shown in Figure 2. The goal was to open the interface with a representation of the data that would be familiar to the typical user. Thus interactive tables predominate, allowing the user to choose levels of aggregation and investigate sortable tables of dollar cost values. A given user might be interested in examining costs for all machines of a particular type, or for all machines in a particular region or for all machine parts of a particular "vitality," (*i.e.* importance), or for some combination of any of these. These choices can be made from the "Grouping" panel available directly from the initial window. In the case of Figure 2, the costs are aggregated over all machines of a particular type. One could, however, instead choose to aggregate over machine type and region (thus separately considering the costs for a particular machine type and the various geographic regions), or over only machine class, for example.

The lower table in Figure 2 shows the details for all of the items currently aggregated into the highlighted line of the upper table. Thus, we see that for machine=3184, there are actually many different classes being aggregated, for a variety of different service regions (SR) and vitalities (V). To delve further, the "Data Details" button allows the strategist to see all of the simulation results for all of the combinations of target service levels for the machine/vitality/region/etc. currently selected in the second table.

A simple two-dimensional plot shows the cost curve for the particular machine/region/vitality/etc. currently selected, and for a chosen service time frame. (One can also choose to see a curve for the "aggregate" selected line of the top table.) The

user can switch to other service time frames using the radio buttons shown. The curve distills the information which the "Simulation Details" button would provide, by showing the results attributable to only the currently selected target. For example, when showing the 2-hour data, the 4-hour and 12-hour targets are set equal to the 2-hour target (thus adding no additional constraints to the problem). Similarly, when showing the 4-hour curve, the 2-hour target is set equal to zero, and the 12-hour target is set equal to the 4-hour target.

Both the curve and the tables reflect the current state of the SLO values, or "Service Level Objectives." On initial loading, these values represent the current set of target objectives for a particular machine, region, vitality, etc., as used in production. The cost (aggregated over the relevant subset of points) represented by this set of SLOs is displayed in the top table, and the position of the SLO for the machine shown in the curve is indicated by a red line[1]. In the course of using the application, the user has the opportunity to modify the targets to see the resulting effect on costs (to be reflected in the "NEWCOST" column). SLOs in the top table are aggregated over all the subclasses contained therein, weighted by demand.

The initial interface window was designed to be familiar and to fit the paradigms that the users are used to; however it immediately points to the limitations of this simplistic view of the data. Generally, the user would like to set the SLO somewhere near the "knee" of the cost curve, just before costs start to rapidly increase. However the curve shows only one dimension of the cost landscape. It is impossible for the user to understand the interactions between different service target levels, such as various combinations of two hour and four hour targets. Also, other factors besides just the derivative of the curve or surface, such as customer expectations, marketing strategies, or product lifespan, need to be taken into account in setting the appropriate SLO. Thus we have a need for visualization and human judgement.

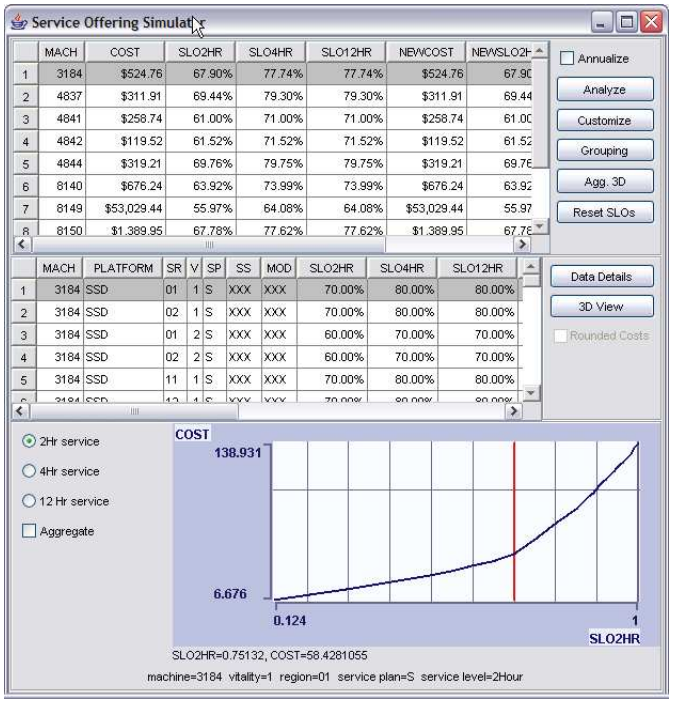

Fig. 2. Initial SOS Window. This "control window" allows the user to choose the level of aggregation and subsets to be viewed in other representations of the data. Cost is shown for the current target service levels (SLOs) in production. The red line on the curve represents the target service level for the machine/vitality/region/etc. shown. Simplistically, it is desirable to place the target level near the knee of the curve, before costs begin to rise rapidly. However actual "optimal" location will depend on many considerations of customer expectations, cost, etc., as well as the cost behavior of the 4 and 12-hour targets. Thus this simple representation of the data does not serve to show the user the implications of *combinations* of various target service levels. (Mach=Machine, SR=Service Region, V=Vitality, SP=Service Plan, Mod=Model)

A. Three-Dimensional Views

The primary motivation for using three-dimensional visual representations, such as those provided by VisAD, is to allow the user to better visualize the multidimensional landscape of cost *vs.* service targets, particularly the two-hour and four-hour targets. Thus a fundamental representation we produce is that shown in Figure 3. This view is available using the "3D View" button in the initial window. Here we provide two views of the landscape, one above the other: a three-dimensional view, with redundant encoding of cost using both height and color, and a flat view, colored by cost, and annotated with contour lines. We use two (redundant) views of the data because we have found that depending on the particular shape of the surface, one or the other of these views may be easier to decode. The views in this panel are completely interactive, with rotation, zooming, and translation provided by the VisAD infrastructure, which uses Java3D. For the three-dimensional view, grid lines are additionally added for ease in finding particular *xy* (2 hour/4 hour) points. Picking in either view allows the user to interactively query data values using the mouse, as well as use those values to be fed back into the cost profile, if desired. Shown in Figure 3 is a blue to yellow colormap, with luminance monotonically increasing; other colormaps are provided as an option. Options are also available to cull the data to a smaller subset. This is particularly useful when the data shows a high degree of variability, as in Figure 3. A user can choose to disregard, for example, two-hour target levels greater than 75%, and four-hour targets greater than 85%, given that they clearly are leading to rapidly escalating costs, and see the landscape without those points, as shown in Figure 4.

Figures 3 and 4 show the cost *vs.* service landscape for a single machine/region/vitality/etc. While useful, strategists typically deal on a much higher level of aggregation when setting target service levels. For example, they may be interested in setting a global strategy for all machines in the RISC System 6000 class, perhaps differentiated by region (as some regions, *e.g.* the northern plains, are more difficult or costly to service in a short time frame due to transportation infrastructure issues or low density of customers). Thus we also provide views of the landscape of aggregated costs, as

[1]For business confidentiality reasons, the particular SLOs, or target service objectives, shown in this paper are for example purposes only, and do not represent actual targets currently used. Similarly, machine identifiers have been altered.

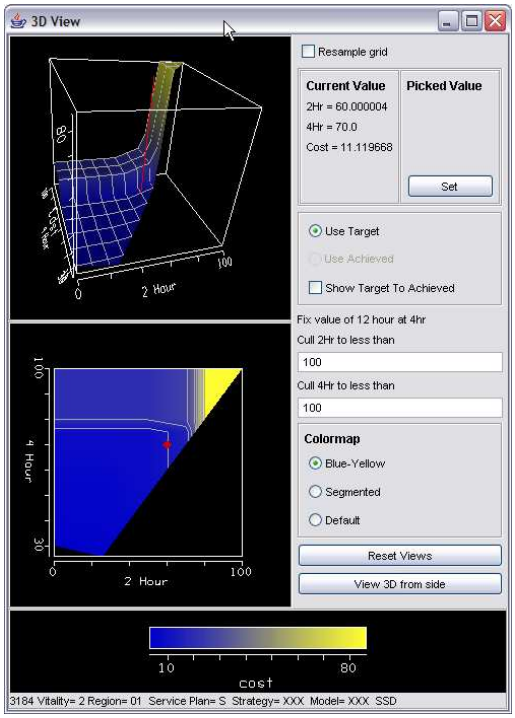

Fig. 3. Landscape of cost *vs.* two hour and four hour target service levels. The red mark/line identifies the current customer service targest.

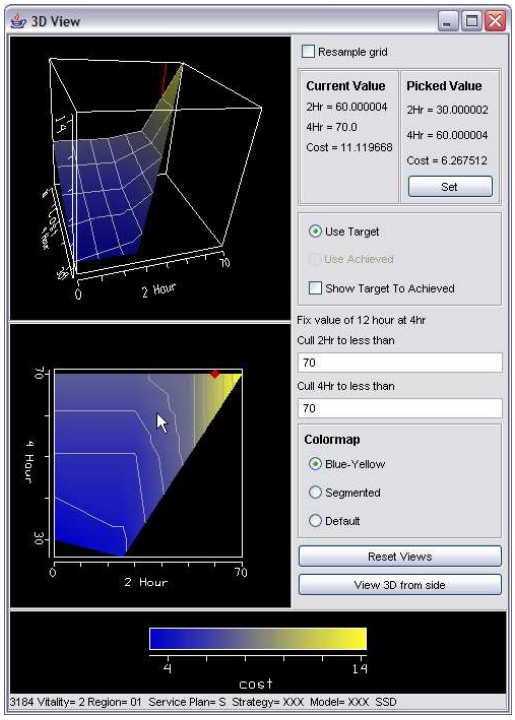

Fig. 4. Landscape of cost *vs.* two hour and four hour target service levels, culled to a desired subset of points. The data are otherwise the same as shown in Figure 3. The user has "picked" in the diagram using the mouse to investigate the costs associated with an alternate service strategy; the associated cost is shown in the upper right of the panel.

shown in Figure 5. This view is available from the "Agg. 3D" button of the initial window. The data comprising this view is exactly the same as that included in the top table of Figure 2; that is, all machines which have been aggregated into the selected row of the table would used to populate Figure 5. In this example, we are looking at all the machines of type 3184, over all of the various regions, vitalities, etc. The SLO displayed as a line and marker is a demand-weighted average of the SLOs in the aggregation.

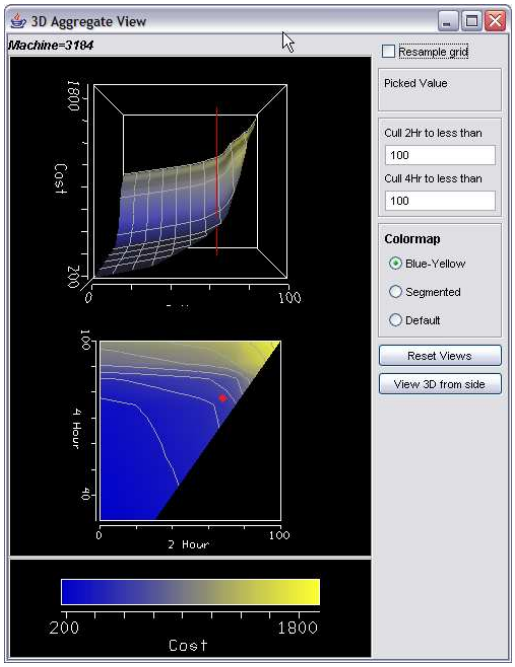

Fig. 5. Landscape of cost *vs.* two hour and four hour service target levels, aggregated over the subset requested from the initial interface window (In this example, all machines of type 3184). One could also aggregate over other variables, or combinations of variables.

To highlight the way in which the three-dimensional views can quickly present the fundamental relationships between cost and service targets, consider the images shown in Figures 6 and 7. Immediately we see two fundamentally different shapes of the cost landscape. For the case of the machine type shown in Figure 6, cost is primarily influenced by the two hour service target level, and is relatively insensitive to the four hour target level. In contrast, Figure 7 shows a completely different behavior, with cost almost entirely dependent on the four-hour target, and insensitive to the two-hour target. This may seem rather surprising, and in fact it is surprising. What is occurring in this latter case is that while two hour targets can be set to various values, due to particularities of the problem, the actual *achieved* two hour service is equal to whatever the four hour target is set to. Thus, whether we target 2 hour service at zero or at 50%, the actual achieved 2 hour service will simply be equal to whatever the four hour target is set to. In fact a choice in the panel allows the user to see the relationship between achieved and targeted values using arrows. Finding this simple yet important characteristic of the cost behavior becomes quite straightforward with the visual representation shown.

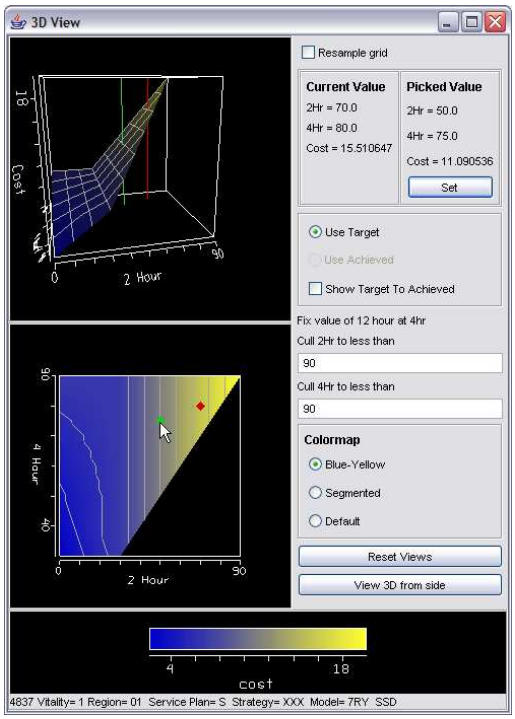

Fig. 6. Cost landscape for a particular machine, vitality, region, etc. combination which shows cost primarily dependent on two hour target service levels. Contrast with that shown in Figure 7. The red line/mark indicates the current SLO setting; the green line/mark indicates a proposed new point, chosen by a user pick in the image.

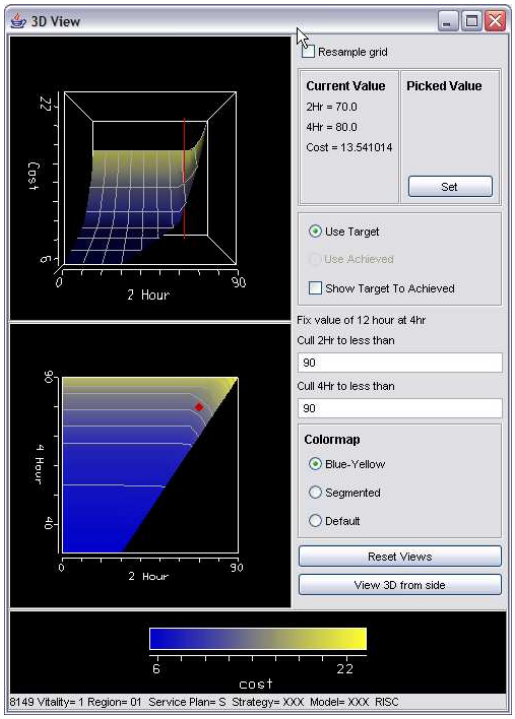

Fig. 7. Cost landscape for a particular machine, vitality, region, etc. combination which shows cost primarly dependent on four hour target service levels. Contrast with that shown in Figure 6.

B. Information Visualization to Understand the Current State

In addition to views of the cost landscape, the SOS application also provides more general visualization capabilities of our data, which we use highlight outliers and focus on the particular machines, classes, or regions which are contributing most to costs. The Analyze panel (available from the "Analyze" button on the main panel) displays the costs associated with the *current* set of service objectives for the various parts under consideration and is shown in Figure 8. The data displayed are created on the fly via SQL queries to the database. For these views we are not concerned with understanding the complete landscape of *possible* costs, but rather want to investigate the costs which result from the *current* set of target levels, allowing us to see outliers.

In Figure 8, we have chosen to slice the data by the variables model, strategy, vitality, region, machine and class. These choices determine the aggregation within the individual data points shown. In Figure 8 the data are finely resolved, down to the individual machine groups. However, if the user chose to slice only by "region" for example, then one data point per region (with cost aggregated) would be shown. The user can then decide which of the sliced variables to display and how to display them. In Figure 8 the user has chosen to use the scatterplot to display the cost *vs.* the vitality, the aggregate plot to display proportional measures of the machine, platform, and service region variables, and the category picture to display the machine types. All of these choices are made from the panel list widgets. The user has highlighted the most expensive non-vitality 5 points in the scatterplot red, and in addition, has marked relatively expensive points for vitality 5 (*i.e.* low importance parts) green. He can then quickly see that all of the highlighted parts appear in the PLATFORM=RISC category, and that machine type 8149 has many relatively high cost vitality 5 points. (Note that these high cost vitality 5 points occur mainly because the LSOs used for this demonstration, for business confidentiality reasons, are not the actual ones used in production). The information found in this way can lead the strategist to the cost landscapes that are most profitable to investigate further, as they are the ones contributing high overall costs.

From the Analyze panel it is also possible to "drill-down" to find out which machine groups comprise the current set of colored points. For example, Figure 9 shows the result of requesting the set of all red points in Figure 8. We see in this case that two of the highlighted points are for the Hawaii region.

We note that, in the SOS application, the database is not simply a convenient way to store data, but rather, is integrally important to the functionality of the application. The power of SQL queries offers an orthogonal, and equally important function, as the visualizations themselves. Visualization, while powerful, is often of limited use especially in business environments, if it is not enabled with the aggregation, roll-up, and slicing features of SQL.

IV. Future Work

We would like to implement refined picking and SLO setting in the aggregate presentations. Our users need a more nuanced

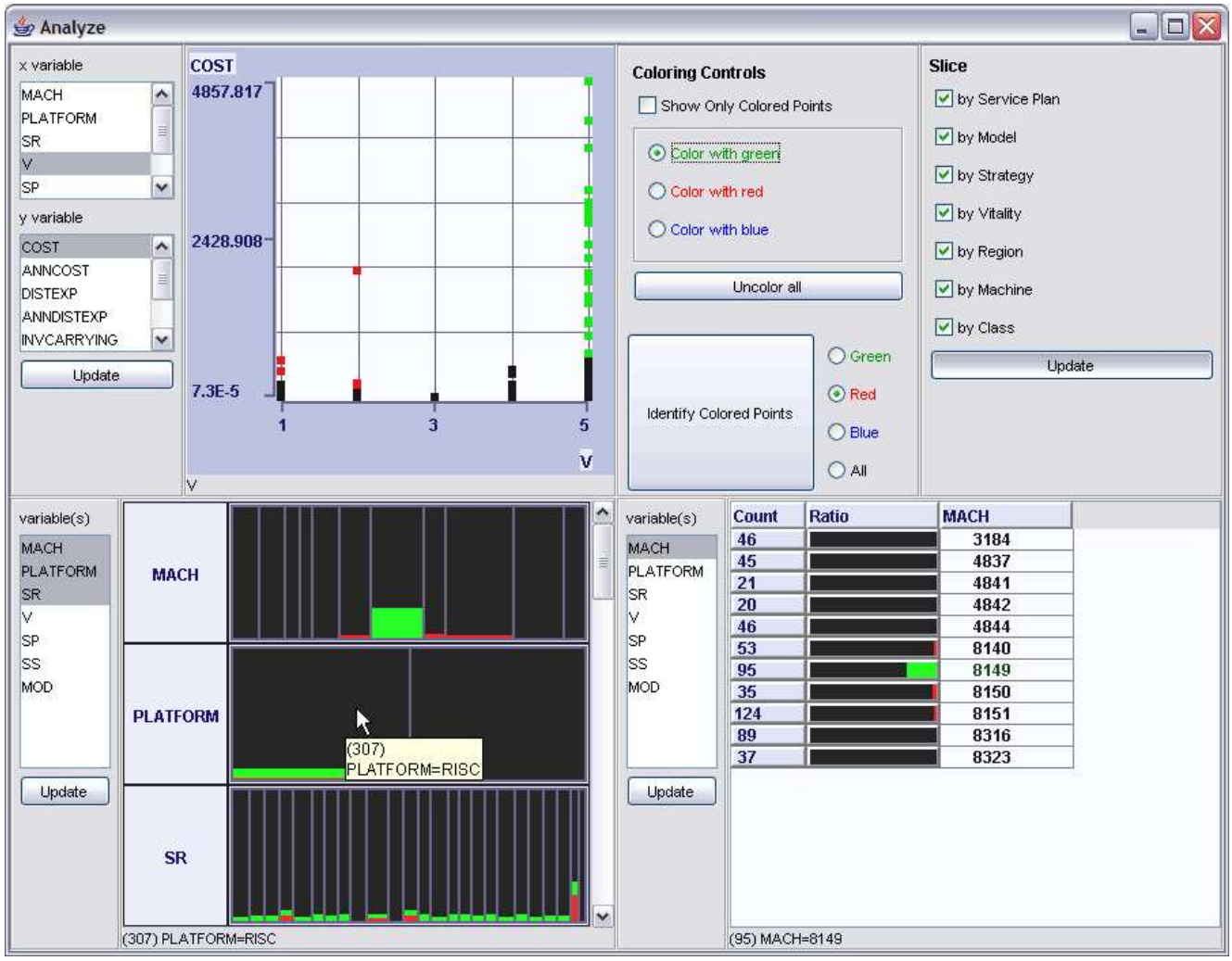

Fig. 8. This panel allows the user to visually query various aspects of the data. This panel is populated with all data represented by a SQL query defined by the "Slice" controls in this panel. The user has chosen to look at cost *vs.* vitality in a scatterplot, and has highlighted (in red) high cost, vitality 1-4 points, and has highlighted in green high cost vitality 5 points. These points are then displayed in the other views. We see for example a large proportion of machine type 8149 points are green (high cost, vitality 5).

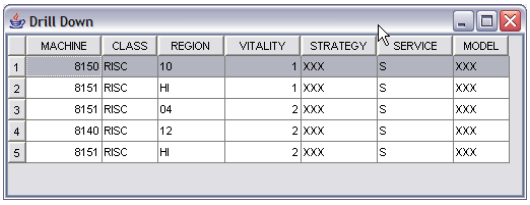

Fig. 9. Result of requesting the list of "red" points from Figure 8. We see that these high cost points occur for machines 8140, 8150, and 8151, and for a few geographic regions including Hawaii (HI).

approach than simply setting the SLO of *all* aggregated components to the picked value. Rather, they would wish to define heuristically a "pattern" of SLOs, such that, for example region 2 targets might always be set to, say, 80% of region 1 targets. This might be desirable, if, for example, the density of customers in region 2 is much lower than in region 1, or are otherwise more expensive to serve.

Depending on how users tend to interact with the application, further development may include, for example, tighter linkage of the information-visualization and spatial views of the data. For example, it may be useful to allow a user to choose interesting machine classes interactively in a scatterplot (*e.g.* by high cost or other characteristics) and immediately populate a spatial landscape view with the selected point(s).

In many of the application areas in which visualization is used, the ultimate goal is to eventually discard visualization as a way of obtaining the solution, once a greater understanding of the problem is reached. That is, just as optimization algorithms determine the optimal stocking locations in a "hands-free" manner, it would be wonderful to devise an algorithm to optimally determine the "best" placement of the service level targets. However, we do not see this as a possibility in the near future. Regional considerations, marketing strategy, intuition, product life-cycle issues and understanding of likely customer reactions all play a part in determining the best set-point for the customer service targets. However, as this application is used we hope to gain insight into how strategists choose optimal

points, and use that to heuristically suggest a customer service strategy, which would almost certainly still need to be verified by the users.

V. Contributions

The Service Offering Simulator quickly allows our users to visually see areas of opportunity. For example, it may be possible to market higher service levels if it can be determined through the three-dimensional representation that such higher levels of service will result in only a nominal increase in investment.

SOS effectively communicates the alternative support strategies in an intuitive way through the landscape views, and allows data to be displayed at a user-specified level of aggregation. Through the information-visualization views, SOS also allows our users to recognize strategies with deviant characteristics, in either service or cost.

The presentation of this information will allow support strategies to be reviewed throughout the product lifecycle to insure continued affordability in light of the shifting characteristics of the product base. These shifts might include such things as customer base shifts, parts failure projections updates, changes in transportation alternatives, network design changes, and various other cost and demographic changes.

Depending on how one defines "visualization," it is either used very little in the business arena, or used all the time. While sophisticated information visualization techniques, such as parallel coordinates, dendograms, and the like, have made little headway in production environments, charting tools are ubiquitous, and are extremely useful in exploring relational and spreadsheet data. In the work described here, we extend the paradigm of data analysis to include some views of the data, which while not terribly sophisticated from a visualization point of view, provide obvious value over what is typically available to strategists. Tools familiar to users of scientific visualization toolkits, such as shaded, interactive surfaces, colorbars, contour lines, and interactive picking are provided here to users who have not traditionally been exposed to such tools. Because we have been careful to integrate these tools with the common paradigms of interaction, and because we have started from familiar views and extended the application from there, we have made these tools accessible to our users. Finally, we have concentrated on the problem our users are trying to solve and have provided views which match the mental model that they are grappling with, along with views for quickly identifying and understanding outlier data points.

VI. Possible Sidebar on Optimization

Optimization refers to a broad range of techniques which seek to mathematically find the best solution to a problem, where "best" may have a variety of meanings. For example, in scheduling airplane crews, one would like to use the fewest number of employees, while needing to satisfy constraints of hours worked in a row, and hours worked per week, plus geographic realities of where the work is to take place. If one wishes to route delivery vehicles, then one seeks to minimize total miles traveled while ensuring that all packages are delivered within a particular time frame, and that the number of packages placed in a truck will actually fit. If one is cutting patterns from fixed widths of material, then the goal is to minimize the waste fabric while ensuring that any necessary pattern matches can be accommodated. In general, one wishes to either maximimize or minimize some "objective function" subject to a set of "constraints."

Our users' part stocking model is rather complex and includes the following input data:

- P = the set of parts
- G = the set of service groups (including, *e.g.,* the machine type, geographic region, etc.)
- Z = the set of zipcodes from which demand for parts will come
- $D(P,G,Z)$ = the per-period forcasted demand for part P in service group G in region Z
- L = the set of stocking locations
- $N(L)$ = the set of "neighbor" locations for location L (possible alternate parts depots from the primary depot)
- $Z(L)$ = the set of zipcodes with location L as its primary source
- $S(L,P)$ = the current stocking level of part P at location L
- $V(P)$ = the unit cost for part P
- $C(L,Z)$ = the per unit shipping cost from location L to zipcode Z
- $R(L,P)$ = the per period holding cost at location L for part P
- T = the set of service types (eg. 2 hour, 4 hour, 12 hour)
- $\alpha(G,T)$ = the target service level for service group G and type T

These are the quantities which are *given* at the beginning of the run. Then there are the variables: the quantities which are allowed to vary in order to find an optimal (lowest cost) solution:

- $S^+(L,P)$ = Increase of stock at location L for part P
- $S^-(L,P)$ = Decrease of stock at location L for part P
- $Y(L,P)$ = per period flow out of location L for part P
- $\lambda(P,G,L,L')$ = Average per period flow of part P in service group G from location L to zipcodes that have location L' as their primary location

Then we write constraint functions which express the requirement that the overall demand is satisfied: the total flow for a particular part and a particular service group to a given location must satisfy the forecast demand for that part, service group, and location. We must also ensure that the demand satisfied within a particular time frame, divided by the total demand, is at least equal to the input target service level objective for that time frame.

Finally we write the "objective function" which is an expression of the cost. This cost will consist of several terms, including inventory cost (which will depend on stocking levels, holding costs, and per-unit part costs), transportation costs to the customer, transportation costs between depots, and a cost to "bin" a newly-acquired part at a depot. The point of the optimization run is simply to minimize this total cost, subject to the constraints given. In practice this is accomplished by

solving a "mixed integer program," where "mixed integer" refers to the fact that some of the variables are required to be integers (for example the increase or decrease in stock must be an integral value).

If we wished to solve this problem *without* specifying the target service levels as an input to the problem, it would be necessary to make the service level $\alpha(P,G,T)$ a *variable* in the problem and then modify the objective function such that instead of it simply being a sum of the various costs, it would be a sum of the various "benefits" minus the sum of the costs. Then the goal instead of minimizing cost C would be instead to maximize $B-C$, where in this case benefit B would be a function of service level $\alpha(P,G,T)$. The obvious problem is that there is no known functional relationship for $B(\alpha)$, or even an approximation for it.

ACKNOWLEDGMENT

We would like to thank John Forrest for developing the algorithms which enable us to generate a large number of data samples quickly. The Service Offering Simulator would not be possible without this work. We would like to thank Jon Lee for the mathematical description of the stocking model given here.

Donna Gresh is a research staff member at the IBM T.J. Watson Research Center in Yorktown Heights, NY. Her current interests include the application of visualization to business problems. She earned a B.S. in Engineering from Swarthmore College in 1983, and an M.S. and Ph.D. in Electrical Engineering from Stanford University in 1985 and 1990 respectively. She is a member of IEEE and the IEEE Computer Society.

Eugene Kelton is an operations research staff member at the IBM Service Parts Solutions organization in Mechanicsburg, P.A. He is currently engaged in the development and global deployment of Network Neighborhood, a next generation inventory stocking model, both internally and externally. He earned a B.S. in Business Information Systems from Messiah College in 1991 and is currently pursing his M.B.A. with a Supply Chain Management concentration at Lehigh University.

REFERENCES

[1] datadesk, "*http://www.datadesk.com/datadesk*," 2003.
[2] alphablox, "*http://www.alphablox.com*," 2003.
[3] avs, "*http://www.avs.com*," 2003.
[4] opendx, "*http://www.opendx.org*," 2003.
[5] Chris Stolte and Pat Hanrahan, "Polaris: A system for query, analysis and visualization of multi-dimensional relational databases," in *Proceedings of IEEE Information Visualization*, 2000.
[6] M. Livny, R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lamwande, J. Myllymaki, and K. Wenger, "Devise: Integrated querying and visual exploration of larget datasets," in *Proceedings of ACM SIGMOD*, 1997.
[7] R. Bosch, C. Stolte, D. Tang, J. Gerth, M. Rosenblum, and P. Hanrahan, "Rivet: A flexible environment for computer systems visualization," *Computer Graphics*, pp. 68–73, Feb. 2000.
[8] W. Hibbard, "VisAD: Connecting people to computations and people to people," *Computer Graphics*, vol. 32, no. 3, pp. 10–12, 1998.
[9] D.L. Gresh, D.A. Rabenhorst, A. Shabo, and S. Slavin, "Prima: A case study of using information visualization techniques for patient record analysis," in *Proceedings of IEEE Visualization*, 2002.