Interval Disaggregate: A New Operator for Business Planning

Sang K. Cha[†§], Kunsoo Park[†§], Changbin Song[†], Kihong Kim[†], Cheol Ryu[†§], Sunho Lee[†§]

[†]SAP Labs Korea
Seoul, Korea

[§]Seoul National University
Seoul, Korea

{sang.k.cha, kunsoo.park01, chang.bin.song, ki.kim, cheol.yoo, sunho.lee}@sap.com

ABSTRACT

Business planning as well as analytics on top of large-scale database systems is valuable to decision makers, but planning operations known and implemented so far are very basic. In this paper we propose a new planning operation called *interval disaggregate*, which goes as follows. Suppose that the planner, typically the management of a company, plans sales revenues of its products in the current year. An interval of the expected revenue for each product in the current year is computed from historical data in the database as the prediction interval of linear regression on the data. A total target revenue for the current year is given by the planner. The goal of the interval disaggregate operation is to find an appropriate disaggregation of the target revenue, considering the intervals.

We formulate the problem of interval disaggregation more precisely and give solutions for the problem. Multidimensional geometry plays a crucial role in the problem formulation and the solutions. We implemented interval disaggregation into the planning engine of SAP HANA and did experiments on real-world data. Our experiments show that interval disaggregation gives more appropriate solutions with respect to historical data than the known basic disaggregation called referential disaggregation. We also show that interval disaggregation can be combined with the deseasonalization technique when the dataset shows seasonal fluctuations.

Keywords

Business planning, interval disaggregation, referential disaggregation, deseasonalization

1. INTRODUCTION

On-Line Analytic Processing (OLAP) [27] and Data Cubes [9] are common in large-scale data warehouses in many companies. However, *business planning* as well as analytics on top of large-scale database systems is a valuable tool to decision makers [31, 2], and only recently research on planning operations has started [12,13].

Business planning is an important task of companies in which business targets are defined for future periods in order to get specific guidelines for current operations which can also serve as a means to check whether the targets have been reached or not [12]. Jaecksch and Lehner [12] extended existing OLAP operations with planning operations such as copy, delete, revalue, disaggregate and forecast by using the Extended Multidimensional Data Model [21]. A main planning operation in [12] as well as in Oracle Hyperion [20] and IBM Cognos Express [11] is disaggregation, also called *referential disaggregation*. An example of referential disaggregation is shown in Table 1. A company had the sales revenues 40, 30, and 30 in 2013 for products A, B, and C, respectively, and it wants the total revenue of 110 for 2014 (i.e., 10% increase). Referential disaggregation is to disaggregate the total amount to each product by some reference values, which in this case are the revenues of 2013. The result of disaggregation is shown in Table 1, which is 10% increase in each product. As we can see in this example, however, referential disaggregation is too simple and obvious to decision makers, and it doesn't consider, for instance, sales trends of the products.

Table 1. Referential disaggregation

Product	2013	2014
A	40	44
B	30	33
C	30	33
Total	100	110

Business forecasting typically uses time series (e.g., monthly, quarterly, or yearly) data, and it adopts various methods to calculate forecasts [15, 4, 5, 7, 25]. The most popular method to identify a trend in historical data is linear regression with least squares fitting [15, 7]. (We describe a more realistic model for the trend of historical data in Section 5.) An advantage of linear regression is that it also provides prediction intervals for future points [18]. For example, Figure 1 shows a linear regression with sales data from 2005 to 2011 and the 90% prediction interval for 2012 (i.e., a future point of 2012 will fall in this interval 90% of time). This prediction interval reflects the characteristics of data, i.e., not only the slope of linear regression, but also variations of data points. The smaller the variations are, the narrower is the interval (Figure 2). When monthly or quarterly forecasting is being done, the data may show seasonal fluctuations, in which case a technique called deseasonalization [15] can be applied to the data before forecasting and then the result is adjusted with seasonal effects after forecasting (a detailed example of deseasonalization will be given in Section 3.3).

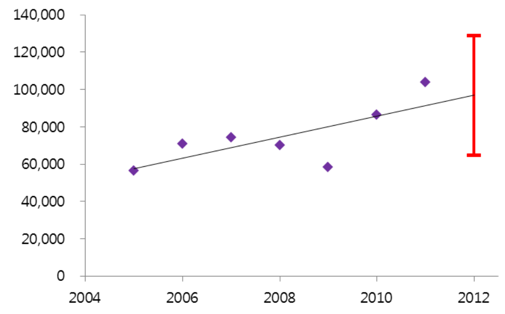

Figure 1. Linear regression and 90% prediction interval

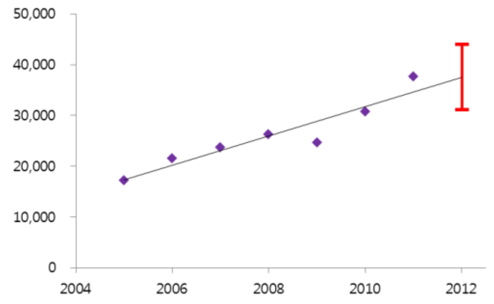

Figure 2. Narrower prediction interval

We propose a new planning operation called *interval disaggregate*. A typical example of this operation goes as follows (see Table 2). Suppose that the planner, typically the management of a company, plans sales revenues of its products in the current year. The revenues of products in the previous year are drawn from the database. An interval of the expected revenue for each product in the current year is computed from historical data in the database (e.g., as the 90% prediction interval). A total target revenue for the current year is given by the planner. The goal of the interval disaggregate operation is to find an appropriate disaggregation of the target revenue, considering the intervals. (The interval for a product, say product A, may be represented by actual values like 40-48 rather than percentages. But as we will see in real-world data where actual values are large, percentages are easier to see their meanings than actual values.)

In Section 2, we formulate the problem of interval disaggregation more precisely, and give solutions for interval disaggregation. Multidimensional geometry plays a crucial role in the problem formulation and the solutions. In Section 3, we describe our implementation of interval disaggregation on SAP HANA, and show experimental results of interval disaggregation on real-world data. Our experiments show that interval disaggregation gives more appropriate and advanced solutions than referential disaggregation, when historical data are taken into consideration. In Section 4 we describe related work, and we conclude with future research directions on business planning in Section 5.

Table 2. Interval disaggregation

Product	2013	Interval	2014
A	40	100-120%	
B	30	90-110%	
C	30	100-110%	
Total	100		110

2. INTERVAL DISAGGREGATION

In this section we formulate the problem of interval disaggregation, and present solutions for the problem.

2.1 Example Scenario

Consider the following business planning scenario. The planner wants to plan the revenue of the company for year 2014 based on historical data. The sales revenues of products and product groups in 2013 are shown in Table 3. The planner typically works out disaggregation of the target revenue top-to-bottom. Initially, he works on the level of product groups. For product groups, the intervals of expected sales for 2014 are computed from historical data as prediction intervals. The planner gives a total target revenue of 2014 and wants to find an appropriate disaggregation of the target revenue for product groups. Suppose that the target revenues of product groups in Table 4 are computed by the interval disaggregate operation, which we will describe below. The planner may adjust the disaggregated target revenues by some other managerial considerations as in Table 5.

Table 3. Example scenario

Product	2013	Interval	2014
A	40	100-120%	
A1	20		
A2	10		
A3	10		
B	30	90-110%	
B1	30		
C	30	100-110%	
C1	20		
C2	10		
Total	100		110

Table 4. Interval disaggregation on product groups

Product	2013	Interval	2014
A	40	100-120%	46.1
A1	20		
A2	10		
A3	10		
B	30	90-110%	31.6
B1	30		
C	30	100-110%	32.3
C1	20		
C2	10		
Total	100		110

Table 5. User adjustment

Product	2013	Interval	2014
A	40	100-120%	47
A1	20		
A2	10		
A3	10		
B	30	90-110%	31
B1	30		
C	30	100-110%	32
C1	20		
C2	10		
Total	100		110

Now the planner goes one level down and works on the level of products for product group A. The system again shows the planner the intervals of expected sales for the products in group A, and it also computes the disaggregation of group A's target revenue which is also an interval disaggregate operation (Table 6). Finally, the planner may adjust the disaggregated values.

Table 6. Interval disaggregation on products

Product	2013	Interval	2014
A	40	100-120%	47
A1	20	100-120%	23.5
A2	10	110-140%	13.6
A3	10	90-100%	9.9
B	30	90-110%	31
B1	30		
C	30	100-110%	32
C1	20		
C2	10		
Total	100		110

In this example scenario, we have seen two occurrences of the interval disaggregate operation, which we will formulate in the next section. Hierarchical disaggregation in this scenario can be applied not only to the product dimension but also to the time dimension (monthly or quarterly from yearly) and the location dimension, plus any combinations of these dimensions.

2.2 Problem Formulation

We define the interval disaggregate operation as follows. The input of the operation is:

- Interval of expected sales for each product in the current year
- Total target revenue for the current year

The output of the operation is an appropriate disaggregation of the target revenue.

A main question in this definition is: what is an appropriate disaggregation? We answer this question by the example in Table 2 because three-dimensional geometry is easier to comprehend than other dimensions. The intervals of products A, B, and C define a rectilinear polyhedron, which will be called a *rectilinear box*. When a, b, and c are disaggregations of 110 for products A, B, and C, respectively, $a + b + c = 110$ is a hyperplane as shown in Figure 3. The intersection of the rectilinear box and the hyperplane is the set of feasible solutions for the disaggregation. Therefore, an appropriate disaggregation would be some midpoint in the intersection. In general, the set of feasible solutions is a (d-1)-dimensional polytope (where d is the number of products), which will be called the *feasible polytope*.

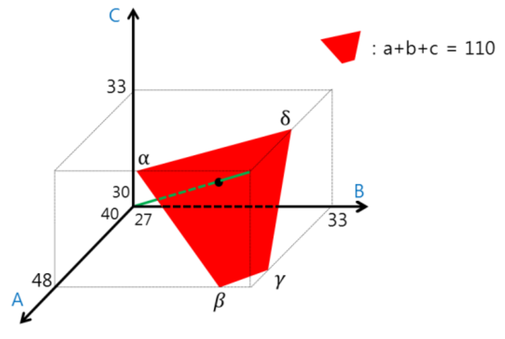

Figure 3. Feasible polytope and intersecting point

There can be many candidates for the midpoint, but we mainly consider the following four points.

1. Point where the min-max line and the feasible polytope intersect
2. Center of mass of the feasible polytope
3. Centroid of the feasible polytope vertices
4. Point whose probability is maximized in the linear regression model

The min (max) point is the point of the rectilinear box defined by the intervals which has the minimum (maximum) value in each axis. The min-max line is the line between the min point and the max point. The point where the min-max line and the feasible polytope intersect will be called the *intersecting point*. For the example in Table 2, the intersecting point is shown in Figure 3. The *center of mass* is the center point of the distribution of mass in the feasible polytope [29]. The *centroid* of the feasible polytope vertices is simply the average of the vertices [22]. The point whose probability is maximized is called the *mode* [23].

Let $\mathrm{Min} = (p_1, \dots, p_d)$ and $\mathrm{Max} = (q_1, \dots, q_d)$ be the min point and the max point of the rectilinear box, respectively. Note that all intervals $(p_1, q_1), \dots, (p_d, q_d)$ are specified by Min and Max. Let T be the target value of interval disaggregation. In Figure 3, $\mathrm{Min} = (40,27,30)$, $\mathrm{Max} = (48,33,33)$, and $T = 110$. In the following sections, we describe how to find the four points.

2.3 Finding Intersecting Point

Given the rectilinear box and the hyperplane, we describe how to find the intersecting point. The min-max line, which passes through Min and Max, is denoted by a vector $X = \mathrm{Min} + t(\mathrm{Max} - \mathrm{Min})$ for a real number t, i.e., $X = (p_1 + t(q_1 - p_1), \dots, p_d + t(q_d - p_d))$. See Figure 4 in the case of two dimensions. The equation of the hyperplane is $a_1 + \cdots + a_d = T$ for a d-dimensional point $(a_1, \dots, a_d)$. To find the intersecting point, we plug the coordinates of X into the equation of the hyperplane, which results in $(p_1 + \cdots + p_d) + t\big((q_1 + \cdots + q_d) - (p_1 + \cdots + p_d)\big) = T$. Hence,

$$t = \frac{T-(p_1+\cdots+p_d)}{(q_1+\cdots+q_d)-(p_1+\cdots+p_d)}.$$

If we put the value of t into X, we get the intersecting point. For the example in Table 2, $X = (40 + 8t,\ 27 + 6t,\ 30 + 3t)$, $t = \frac{13}{17}$, and the intersecting point is $(46.1,\ 31.6,\ 32.3)$.

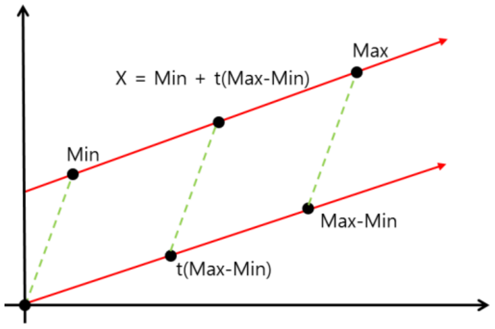

Figure 4. Line passing through Min and Max

Another way to view the above computation is as follows. Given a vector $(a_1, \dots, a_d)$ in d dimensions, its Manhattan distance (also known as L_1 distance) [22] is $a_1 + \cdots + a_d$, while its Euclidean distance is $\sqrt{a_1{}^2 + \cdots + a_d{}^2}$. The line segment between Min and Max (i.e., vector $\mathrm{Max} - \mathrm{Min}$) has Manhattan distance $(q_1 + \cdots + q_d) - (p_1 + \cdots + p_d)$, and the line segment between Min and the intersecting point has Manhattan distance $T - (p_1 + \cdots + p_d)$ because the Manhattan distance of the intersecting point is T and that of Min is $p_1 + \cdots + p_d$. Hence, the intersecting point is the point in the line segment between Min and Max whose relative distance from Min is $t = \frac{T-(p_1+\cdots+p_d)}{(q_1+\cdots+q_d)-(p_1+\cdots+p_d)}$, i.e., it is $\mathrm{Min} + t(\mathrm{Max} - \mathrm{Min})$.

2.4 Finding Feasible Polytope (Center of Mass and Centroid)

To find the center of mass and the centroid, we need to find the feasible polytope, i.e., the vertices of the feasible polytope. The vertices of the feasible polytope are the intersections of the hyperplane and the edges (1-faces by the terminology of [17]) of the rectilinear box. In Figure 3, the intersection of the hyperplane and the edge between (48,27,33) and (48,33,33) is $\alpha = (48,29,33)$, and the other intersections are $\beta = (48,32,30)$, $\gamma = (47,33,30)$, and $\delta = (44,33,33)$.

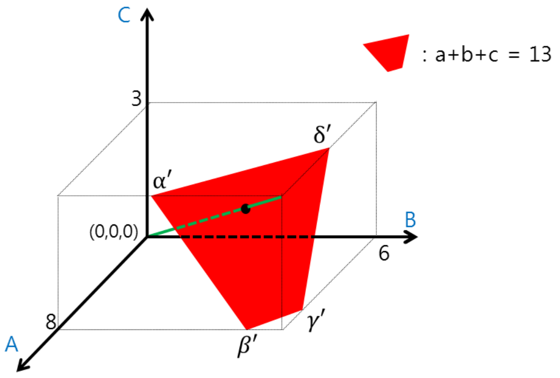

Figure 5. Rectilinear box when Min = (0,0,0)

To find the intersections of the hyperplane and the edges of the rectilinear box, we assume that the min point of the rectilinear box is (0,0,0). See Figure 5. After finding intersections with this assumption, we can get the correct intersections by adding the min point to the intersections. Consider the edges parallel to the A-axis in Figure 5. There are four edges parallel to the A-axis, and the value of a in these edges satisfies $0 \le a \le 8$. Since the hyperplane is $a + b + c = 13$, the intersections with these edges satisfy $5 \le b + c \le 13$. Since the value of b is either 0 or 6 and the value of c either 0 or 3 in these edges, we need to find the combinations of these values that satisfy $5 \le b + c \le 13$. They are $b = 6$, $c = 0$, which results in intersection $\gamma' = (7,6,0)$, which is $\gamma = (47,33,30)$ after adding the min point, and $b = 6$, $c = 3$, which results in $\delta' = (4,6,3)$, which is $\delta = (44,33,33)$.

In general, let $(r_1, r_2, \ldots, r_d)$ be the max point of the rectilinear box, when the min point is $(0, \ldots, 0)$, i.e., $r_i = q_i - p_i$ for $1 \le i \le d$. To find the intersections on the edges parallel to the A-axis, we need to find all subsets of $\{r_2, \ldots, r_d\}$ whose sum is between $T' - r_1$ and T', where $T' = T - (p_1 + \cdots + p_d)$. This is a variant of the subset sum problem [3,16,28,32]. Similarly, we can find the intersections on the edges parallel to the B-axis, etc. In the following, however, we present two algorithms to find the intersections for *all* axes at the same time, i.e., all vertices of the feasible polytope.

Table 7. First algorithm when $r_1 = 8, r_2 = 6, r_3 = 3, T' = 13$

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Sum	0	8	6	14	3	11	9	17

The first algorithm computes all possible subsets of $\{r_1, \ldots, r_d\}$ and checks if each subset can produce intersections. Let $\mathrm{Sum}[0..2^d - 1]$ be an array such that $\mathrm{Sum}[i]$ is the sum of the subset represented by the binary notation of i, i.e., r_k is in the subset if the k-th rightmost bit of i is 1. For example, if $i = 110_2$ in Figure 5, $\mathrm{Sum}[6] = 9$ because 110_2 represents $\{r_2, r_3\}$ where $r_2 = 6$ and $r_3 = 3$. Array Sum can be computed as follows.

```
$\mathrm{Sum}[0] = 0$
$\mathrm{for}(k = 1 \text{ to } d)$
    $\mathrm{for}(j = 0 \text{ to } 2^{k-1} - 1)$
        $\mathrm{Sum}[2^{k-1} + j] = \mathrm{Sum}[j] + r_k$
```

From each entry $\mathrm{Sum}[i]$, we find intersections by the following cases. Let $\mathrm{maxr} = \max_{1 \le k \le d}\{r_k\}$.

1. $\mathrm{Sum}[i] = T'$: the subset represented by i is an intersection.
2. $T' - \mathrm{maxr} < \mathrm{Sum}[i] < T'$: for each dimension $1 \le k \le d$, we do the following: if the k-th rightmost bit of i is 0 and $\mathrm{Sum}[i] > T' - r_k$, output intersection $(b_1, \ldots, b_d)$, where $b_j = r_j$ if $j \ne k$ and the j-th rightmost bit of i is 1; $b_j = 0$ if $j \ne k$ and the j-th rightmost bit of i is 0; $b_j = T' - \mathrm{Sum}[i]$ if $j = k$. (If $\mathrm{Sum}[i] = T' - r_k$, there is an intersection, but this intersection will be found in Case 1 of some other entry.)
3. $\mathrm{Sum}[i] \le T' - \mathrm{maxr}$: output no intersections. (If $\mathrm{Sum}[i] = T' - \mathrm{maxr}$, again this intersection will be found in Case 1 of some other entry.)

For the example in Figure 5, $r_1 = 8, r_2 = 6, r_3 = 3$, and $T' = 13$. Array Sum for this example is shown in Table 7. Since $\mathrm{maxr} = 8$, $\mathrm{Sum}[2]$ is in Case 2, and it produces intersection $\gamma' = (7,6,0)$.

Let S be the number of intersections, and P the number of entries such that $T' - \mathrm{maxr} < \mathrm{Sum}[i] \le T'$. Computing Sum takes $O(2^d)$ time. For each entry such that $T' - \mathrm{maxr} < \mathrm{Sum}[i] \le T'$, the three cases above take at least $O(d)$ time, and if there are many intersections from the entry then Case 2 takes $O(d)$ time for each intersection. Therefore, the time complexity of the first algorithm is $O(2^d + d \cdot P + d \cdot S)$.

Table 8. Second algorithm when $r_1 = 8, r_2 = 6, r_3 = 3, T' = 13$

A	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	8	8	8	8	8	8
2	0	0	0	0	0	0	6	6	8	8	8	8	8	8
3	0	0	0	3	3	3	6	6	8	9	9	11	11	11

The second algorithm uses dynamic programming. Let $A(i, w)$ be the maximum value $\le w$ that can be obtained with $r_1, \ldots, r_i$. Then a dynamic programming recurrence for $A(i, w)$ is:

$$A(i,w) = \begin{cases} 0 & \text{if } i = 0 \text{ or } w = 0 \\ A(i-1, w) & \text{if } i > 0, w > 0, \text{and } r_i > w \\ \max\begin{cases} A(i-1, w) \\ A(i-1, w - r_i) + r_i \end{cases} & \begin{array}{l} \text{if } i > 0, w > 0, \\ \text{and } r_i \le w. \end{array} \end{cases}$$

We first compute all entries of table $A(0..d, 0..T')$ by the recurrence above, and then find all paths from the entries $A(d, T' - \mathrm{maxr} + 1..T')$ to $A(0,0)$ by backtracking [30]. Each path corresponds to a subset of $\{r_1, \ldots, r_d\}$ whose sum is $> T' - \mathrm{maxr}$. Hence, the number of distinct paths is exactly P. For each path, we perform Cases 1 and 2 of the first algorithm, where $\mathrm{Sum}[i]$ is now the sum of the subset corresponding to the path. The time complexity of the second algorithm is $O(d \cdot T' + d \cdot P + d \cdot S)$, since computing table A takes $O(d \cdot T')$ time and backtracking $O(d \cdot P)$.

For Figure 5, dynamic programming table A is shown in Table 8, where there are four backtracking paths, each of which produces an intersection (e.g., the backtracking path from A(3,6) produces intersection $\gamma' = (7,6,0)$). If we backtrack from A(3,7), we arrive at (0,1). Thus, if there are identical values in $A(d, T' - \mathrm{maxr} + 1..T')$, we backtrack only from the leftmost one, which leads to A(0,0).

Comparing the two algorithms, the first one is preferable when the dimension d is small whereas the second is a good choice when the target value T' is moderate. Note that the second is also an exponential time algorithm because the value of T' is exponential with respect to the input size representing T'. When we run the two algorithms on a PC with a 4GB memory, the maximum value of 2^d for the first algorithm and that of dT' for the second algorithm is 2^{30} due to the memory limit. The running times of the two algorithms are shown in Table 9. When either d or T' is moderate, the algorithms are very fast. Even in the extreme case, they finish within 5 seconds or so.

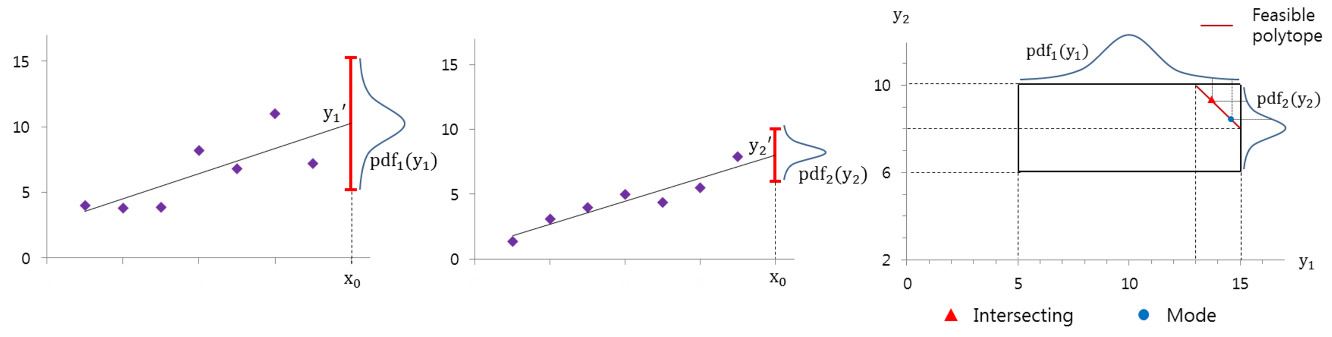

Figure 7. Two-dimensional interval disaggregation with pdf_1 and pdf_2 (t-distribution with 5 degrees of freedom)

Table 9. Running times (sec) of two algorithms on various values of d and T′

(d, T')	$(2^4, 2^{11})$	$(2^4, 2^{26})$	$(30, 2^{11})$	$(30, 2^{25})$
First algorithm	0.001	0.001	5.009	5.007
Second algorithm	0.001	4.140	0.002	3.720

Once all vertices of the feasible polytope are found by one of the two algorithms above, the centroid, which is the average of the vertices, can be easily computed. For instance, the centroid is $\frac{\alpha+\beta+\gamma+\delta}{4} = (46.75, 31.75, 31.5)$ in Figure 3.

To find the center of mass, we first divide the feasible polytope into simplexes (e.g., a simplex in two dimensions is a triangle, and it is a tetrahedron in three dimensions) by using Delaunay triangulation [8]. The center of mass of a simplex is the centroid of its vertices, and the center of mass of the feasible polytope is the weighted sum of the centers of mass of the simplexes where the weights are the volumes of the simplexes [8]. We implemented this computation by incorporating the code for Delaunay triangulation from CGAL, Computational Geometry Algorithms Library [1]. For the example in Figure 3, the center of mass is $(46.6, 31.6, 31.8)$.

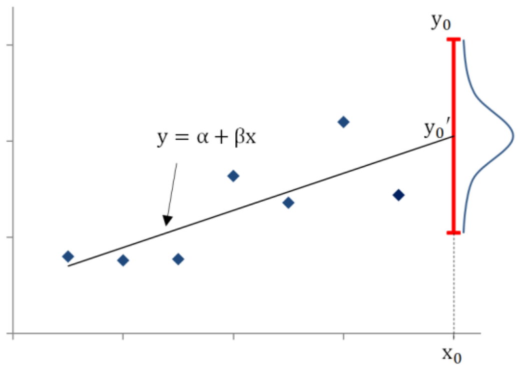

Figure 6. Linear regression and probability density function

2.5 Finding Mode

When there are d dimensions in interval disaggregation, we apply linear regression to each dimension. Let $y = \alpha + \beta x$ be the linear equation obtained by linear regression on a sample of n points in a dimension. We are interested in the value of the response variable y_0 at a future value x_0. Let $y_0{}' = \alpha + \beta x_0$. The value of y_0 at x_0 follows a t-distribution whose mean is $y_0{}'$. See Figure 6, where n=7. If $pdf(\cdot)$ is the probability density function of the t-distribution with n-2 degrees of freedom, the probability density function for y_0 is

$$pdf\left(\frac{y_0 - y_0{}'}{\sqrt{MS_{Res}\left(1+\frac{1}{n}+\frac{(x_0-\bar{x})^2}{S_{xx}}\right)}}\right),$$

where MS_{Res}, $\bar{x}$, S_{xx} are values computed from the n sample points [18].

Let y_i, $1 \le i \le d$, be the response variable (y_0 in the previous paragraph) in the i-th dimension at the future value x_0. Let $pdf_i(y_i)$ be the probability density function for y_i. If we assume for simplicity that the d dimensions of interval disaggregation are independent, the mode is a point $(a_1, \dots, a_d)$ such that $pdf_1(a_1) \times \dots \times pdf_d(a_d)$ is maximized. Consider a two-dimensional interval disaggregation in Figure 7, where Min=(5,6), Max=(15,10), and T=23. The mode is the point (a_1, a_2) such that $a_1 + a_2 = 23$ and $pdf_1(a_1) \times pdf_2(a_2)$ is maximized. For the example in Figure 3, the four points are shown in Figure 8.

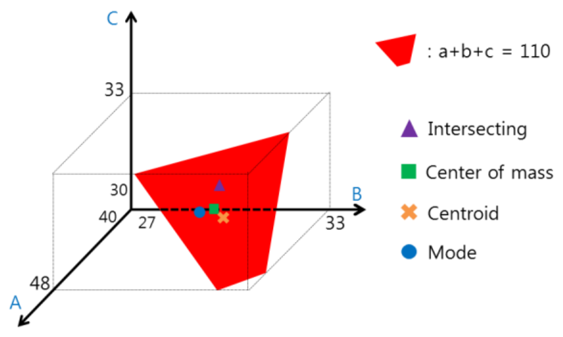

Figure 8. Intersecting (46.1, 31.6, 32.3), center of mass (46.6, 31.6, 31.8), centroid (46.75, 31.75, 31.5), mode (47.01, 31.22, 31.7)

To find the mode in the search space of the feasible polytope, we use three methods: hill climbing of our own implementation, pattern search in MATLAB [19], and Interalg of Openopt [14]. In every experiment in Section 3, the three methods found the same point, which is the mode.

Table 10. Comparison of intersecting point, center of mass, and mode. Inside parentheses are values of probability density functions.

Product	Intersecting	Center of mass	Mode
A	13.57 (0.134)	14.00 (0.108)	14.55 (0.081)
B	9.43 (0.134)	9.00 (0.218)	8.45 (0.336)
Prob.	0.0180	0.0236	0.0273

Table 10 shows the intersecting point, the center of mass, and the mode for the example in Figure 7. The intersecting point is the point $(13.57, 9.43)$ where $\mathrm{pdf}_1(13.57) = \mathrm{pdf}_2(9.43) = 0.134$. The center of mass (also centroid) is (14,9), which is the center point of the feasible polytope (which is a line segment in this case). The mode is $(14.55, 8.45)$ where $\mathrm{pdf}_1(14.55) \times \mathrm{pdf}_2(8.45) = 0.0273$ is the maximum.

2.6 Cell Locking

As described in the scenario of Section 2.1, the planner may adjust disaggregated values after interval disaggregation. During the process of adjusting, he may fix some values and want to see the remaining values determined by interval disaggregation. For the disaggregated values in Table 4, suppose that the planner fixes the value of product group A to 47 and wants the values of product groups B and C to be computed by interval disaggregation (Table 5). This is a case of cell locking, and the locking of k cells reduces a d-dimensional problem to a (d-k)-dimensional one.

In Table 5 where the disaggregated value of product group A is fixed to 47, it becomes a two-dimensional interval disaggregation in which Min = (27,30), Max = (33,33), and the target value is 110 – 47 = 63. For each of the four points as the midpoint, the two-dimensional interval disaggregation can be solved. For instance, we can compute the intersecting point as follows. Max – Min = (6,3), which has Manhattan distance 9. Since the Manhattan distance of the intersecting point is 63 and that of Min is 57, the intersecting point has relative distance 6/9 from Min, and it is $\mathrm{Min} + (6,3) \cdot \frac{6}{9} = (31,32)$, as shown in Table 5.

2.7 Discussions

We now make a comparison between the four points. Consider the two-dimensional interval disaggregation in Figure 7. Now we change the value of T from 13 to 23 as shown in Figure 9. Note that the feasible polytope is a line segment, e.g., the line segment between (5,8) and (7,6) when T=13.

The intersecting point is the intersection of the min-max line and the feasible polytope. Hence, it moves from (6.43, 6.57) to (13.57, 9.43) as T goes from 13 to 23 in Figure 9. In d dimensions, the intersecting point is $\mathrm{Min} + \mathrm{t}(\mathrm{Max} - \mathrm{Min})$, and thus it has relative distance t from Min in each dimension. Hence, it is a point $(\mathrm{a}_1, \dots, \mathrm{a}_\mathrm{d})$ such that $\mathrm{pdf}_1(\mathrm{a}_1) = \cdots = \mathrm{pdf}_\mathrm{d}(\mathrm{a}_\mathrm{d})$, i.e., it balances the probabilities in all dimensions.

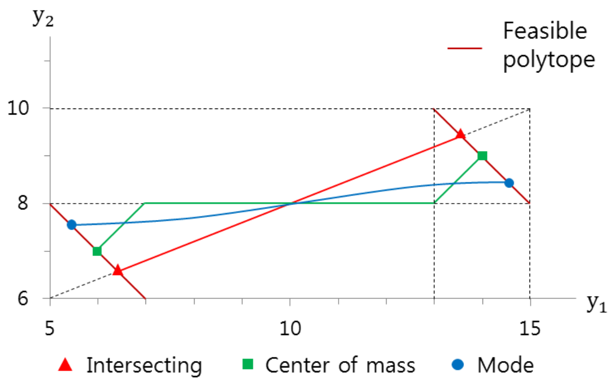

Figure 9. Intersecting point, center of mass, and mode when T changes from 13 to 23

Since the center of mass is the center point of the feasible polytope, it moves from (6,7) to (7,8), then to (13,8), and finally to (14,9) as T goes from 13 to 23 in Figure 9. The centroid is the same as the center of mass in two dimensions, but it may be different in higher dimensions. In Figure 3, if we change T from 112 to 110, the feasible polytope is a triangle from T=112 to 111, but just after 111 it becomes a quadrilateral. Thus the centroid does not make a continuous line when T changes from 112 to 110, while the center of mass always makes a continuous line. Hence, the centroid may be less appropriate as the midpoint than the center of mass in three or higher dimensions.

An advantage of finding the feasible polytope (on the way of computing the center of mass or the centroid) is that we can tighten the intervals so that they don't have ranges where there exist no feasible solutions. In Table 2, input intervals are 40-48 for A, 27-33 for B, and 30-33 for C, but the tightened intervals are 44-48 for A, 29-33 for B, and 30-33 for C as computed in Section 2.4.

The mode is a point $(\mathrm{a}_1, \dots, \mathrm{a}_\mathrm{d})$ such that $\mathrm{pdf}_1(\mathrm{a}_1) \times \dots \times \mathrm{pdf}_\mathrm{d}(\mathrm{a}_\mathrm{d})$ is maximized, and it moves from (5.45, 7.55) to (14.55, 8.45) in Figure 9. Let us compare the mode and the intersecting point in Table 10. The intersecting point (13.57, 9.43) satisfies $\mathrm{pdf}_1(13.57) = \mathrm{pdf}_2(9.43)$. In the dimension where the prediction interval is larger, i.e., 1st dimension, the mode point moves to the direction of decreasing pdf_1, and in the other dimension it moves to the direction of increasing pdf_2, because the increased amount of pdf_2 is larger than the decreased amount of pdf_1 as shown in Table 10 and Figure 7.

Our algorithms find the intersecting point, the center of mass, and the centroid (in addition to finding the feasible polytope) exactly as they are defined. For the mode, however, we presented heuristics (hill climbing, pattern search of MATLAB, and Interalg of Openopt) to find a point which is close enough to the mode.

All the programs that compute the four points finish instantaneously in every experiment of Section 3, except pattern search of MATLAB and Interalg of Openopt which take 1-3 seconds depending on function tolerances.

3. EXPERIMENTS

3.1 Planning in SAP HANA

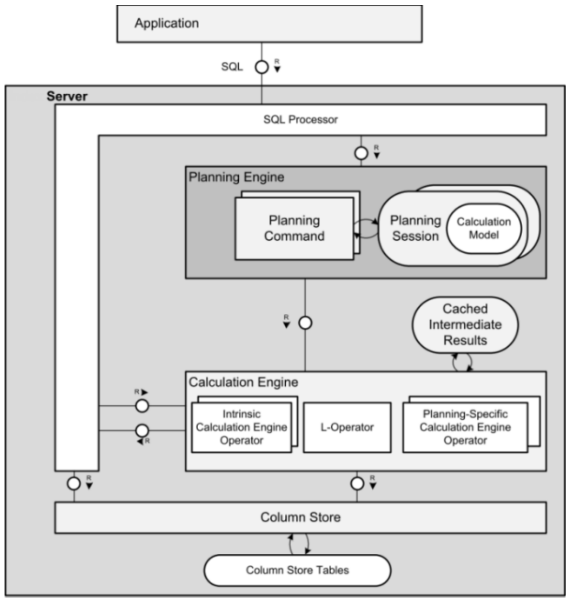

Figure 10. Planning engine of SAP HANA

Figure 10 shows a diagram of SAP HANA, a commercial in-memory column-store database system. Planning users build an initial plan and refine it interactively by changing queries, target values and the distribution of target values. The planning engine supports this use case via planning commands to build a calculation model for the initial plan and to interactively change the calculation model. A calculation model is essentially a data flow graph, where each node represents a relational operator, a composition of relational operators in the form of a SQL query against its input nodes, or a custom operator written in the L language. Calculation models are optimized and evaluated by the calculation engine.

SAP HANA supports disaggregation as relational operators. Disaggregate operators can be placed in any proper location of relational algebra trees. The result of any relational operator such as aggregation can be fed into a disaggregate operator. And the result of a disaggregate operator can be fed into other relational operators such as a join operator. Disaggregation is provided as two variants, a unary disaggregate operator and a binary disaggregate operator. Each of them supports two different disaggregation algorithms, referential disaggregation and interval disaggregation.

Product Group	Interval Min	Interval Max	2014
A	40	48	46.1
B	27	33	31.6
C	30	33	32.3

Interval Disaggregate (Target = 110)

R

Product Group	Interval Min	Interval Max
A	40	48
B	27	33
C	30	33

Figure 11. Unary interval disaggregate operator

Figure 11 illustrates a unary interval disaggregate operator. It receives relation R as an input, distributes the given target value, 110, to the three tuples of R. Its output is the relation R augmented with a new column 2014 storing the disaggregated values, where the intersecting point is used as the midpoint.

Product Group	Product	Interval Min	Interval Max	2014
A	A1	20	24	23.5
A	A2	11	14	13.6
A	A3	9	10	9.9
B	B1	27	33	31
C	C1	20	22	21.3
C	C2	10	11	10.7

Interval Disaggregate

R

Product Group	Product	Interval Min	Interval Max
A	A1	20	24
A	A2	11	14
A	A3	9	10
B	B1	27	33
C	C1	20	22
C	C2	10	11

S

Product Group	Target
A	47
B	31
C	32

Figure 12. Binary interval disaggregate operator

Figure 12 illustrates a binary interval disaggregate operator. It receives two input relations, a disaggregation element relation and a target relation storing the target values to distribute. It works as follows. First, the element relation is partitioned by the grouping columns. In the example, the target relation has three partitions by the Product Group column. Second, for each partition, the corresponding target value is retrieved from the target relation. This lookup is similar to equi-join processing. Third, the target value is distributed to the tuples of the corresponding partition in the element relation. The element relation plus a new column storing disaggregated values is emitted as the operator output.

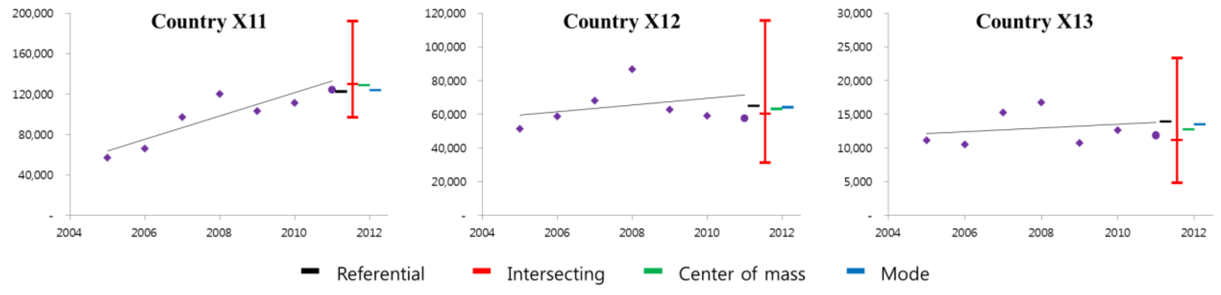

Figure 14. Experiment with group X1 and target value SR×1.1

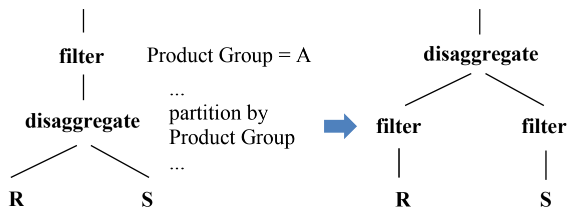

Figure 13. Filter pushdown across disaggregate operator

For performance optimization, a filter or selection operator on the partition columns or a subset of the partition columns are pushed down across binary disaggregation operators, as shown in Figure 13. The binary disaggregation works partition by partition. Thus, it is safe to skip disaggregation for a partition if the partition is filtered out later. In this regard, the disaggregate operator can be compared to the SQL window functions, across which filters on partition-by columns are pushed down.

In real-world planning scenarios, disaggregate operations usually take place in a hierarchy as described in Section 2.1, and fixing values (cell locking) may be interleaved with disaggregate operations as in Section 2.6. Such a situation requires disaggregation as relational operators, because SQL implementations may not be able to handle it.

3.2 Experimental Data and Results

In this section we show experimental results of interval disaggregation on real-world data. The dataset used in our experiments is a nine-dimensional data cube represented as a star schema with a fact table of 67.2 million rows and nine dimension tables such as product, region, time, etc. Each dimension has one or more hierarchies defined in it. For instance, the time dimension hierarchy consists of three levels: year, quarter, and month. The cube contains seven years (2005~2011) of general ledger data for financial accounting in one of the leading industrial companies, which manufactures industrial tools and equipment. Note that the original line-item table has tens of billions of rows and the 67.2 million fact table rows are aggregated ones at the lowest hierarchy levels. When loaded into HANA's in-memory column-store tables, the total table size is about 2.9GB, including indexes.

In our experiments we used the revenues of the company in various countries. We selected 9 countries from region X, 6 countries from region Y, and 3 countries from region Z which have all the data in 2005-2011, and divided them into groups of 3 countries each, i.e., the groups are X1-X3, Y1-Y2, and Z1. We applied referential disaggregation and interval disaggregation to the data of six years 2005-2010 and compared the results of disaggregation against the actual revenues of 2011. In interval disaggregation we used the 90% prediction intervals which are computed from the t-distribution with 4 degrees of freedom. The target values of disaggregation were based on the sum, SR, of the revenues in 2010 (last data points for business planning) or the center, CP, of the prediction interval in 2011 to reflect the trend of historical data. The target values used in experiments were SR, SR×1.1, SR×1.2, CP, CP×1.15, and CP×0.85. Since there were 6 groups and 6 target values in each group, we had 36 experiments.

Table 11. Experimental result of group X1 and target value SR×1.1

	Mean percentage error	Rank
Referential	10.5%	4
Intersecting	5.0%	1
Center of Mass	9.5%	3
Centroid	10.5%	5
Mode	8.4%	2

Figure 14 illustrates one of the 36 experiments, where the group is X1 and the target value is SR×1.1. In Figure 14, the 90% prediction intervals and the results of disaggregation (4 bars) are shown. The leftmost bar is the result of referential disaggregation, the next bar (on the prediction interval) the intersecting point, the next the center of mass, and the last the mode. The results of referential disaggregation are simply 110% of the data points of 2010. But the intersecting point, the center of mass, and the mode reflect the trend in the linear regression model. Thus, in country X11 where the trend is upward, they are higher than the referential point, and in country X13 where the trend is almost flat, they are lower than the referential point. The intersecting points are in the same positions in the prediction intervals relative to the lengths of

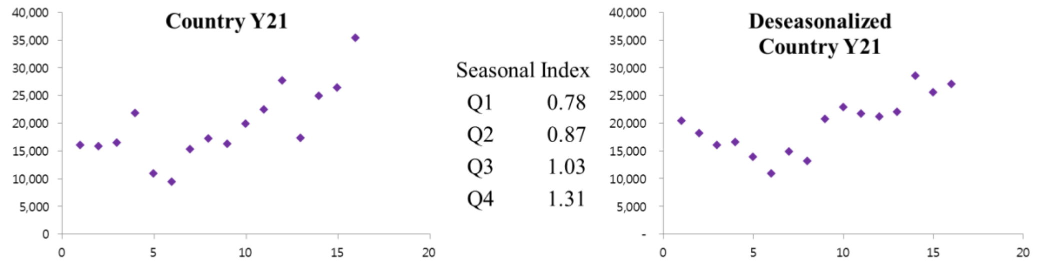

Figure 15. Deseasonalized data points

the intervals, and in country X13 where the prediction interval is small, the mode is closer to the center of the interval than the intersecting point. The results of disaggregation are compared against the actual revenues of 2011. For each point, say the mode, we take the absolute value of the percentage error between the mode and the actual revenue of 2011 in a country, and calculate the mean of the absolute values for the three countries. The mean values are shown in Table 11, where the rightmost column shows the ranks of the points based on the mean values.

Table 12. Summary of experimental results

Group	X1	X2	X3	Y1	Y2	Z1	Average
Referential	3.33	2.67	3.67	3.83	3.50	3.83	3.47
Intersecting	2.83	2.00	2.50	2.17	1.67	3.17	2.39
Center of Mass	3.00	3.50	3.67	2.67	3.17	2.50	3.09
Centroid	3.83	3.50	2.83	3.50	3.83	2.67	3.36
Mode	2.00	3.33	2.33	2.83	2.83	2.83	2.69

The result of the 36 experiments are shown in Table 12, where the value of an entry, say the mode in group X1, is the average of the ranks of the mode in the 6 experiments with 6 target values. A value in the rightmost column is the average in the 36 experiments. In general, the referential point which depends only on the last data point does not perform well, and the four points in the linear regression model are better, which shows that it is helpful to use historical data in disaggregation. Among the points in the linear regression model, the intersecting point and the mode are better than other points.

We remark on how to select one among the five points. If the trends of all dimensions are more or less the same, referential disaggregation works fine. But, if the trends differ from each other, interval disaggregation should be used. Among the four points of interval disaggregation, the intersecting point performs the best in general, and the mode may be used if the planner wishes to put more bias on the dimension of a larger interval (i.e., when the target value is larger than the center point of the rectilinear box, the dimension of a larger interval gets a bigger portion than that of the intersecting point in disaggregation; it gets a smaller portion, otherwise, as shown in Figure 9).

Table 13. Disaggregation for Q1 2011.

Country	Deseasonalized intersecting point	2nd column × Q1 seasonal index	Actual revenue
Y21	26080.6	20465.1 (5.46%)	19405.0
Y22	9416.2	8530.7 (3.10%)	8803.8
Y23	6302.7	5583.2 (8.11%)	6076.0

3.3 Deseasonalization

So far, interval disaggregation was applied to yearly planning. Suppose that the planner wants quarterly planning, but the quarterly dataset shows seasonal fluctuations. An easy solution is that if the next quarter to plan is Q1, then we pick only Q1 data points and apply interval disaggregation as in yearly planning. But, if sales started to increase significantly from Q2 of last year, this recent trend cannot be reflected in this easy solution. That is, we want to use all data points of the quarterly dataset, in which case a technique called deseasonalization [15] should be incorporated. In this section we describe how our interval disaggregation can be combined with deseasonalization in quarterly or monthly planning.

When we are going to use all data points for quarterly planning, we need to remove the effect of seasonal variations. In deseasonalization, we first compute a seasonal index for each quarter [15], and then the sales revenue of each quarter is divided by its seasonal index. Figure 15 shows (1) quarterly revenues of country Y21 in 2007-2010 where Q4 revenues are higher than those of other quarters, (2) seasonal indexes in the four quarters, and (3) the deseasonalized quarterly data points which show the overall trend that is increasing.

We apply interval disaggregation (with intersecting points and target values CP) to the deseasonalized quarterly data of group Y2 in 2007-2010 as in yearly planning, which is shown in Figure 16 and Table 13. Finally, we multiply the seasonal index of Q1 to the intersecting points computed by interval disaggregation (2nd column of Table 13) to get the final disaggregation (3rd column of Table 13) for Q1 2011. A value inside parentheses in Table 13 is the absolute value of the percentage error between final disaggregation (3rd column) and the actual revenue of Q1 2011 (4th column).

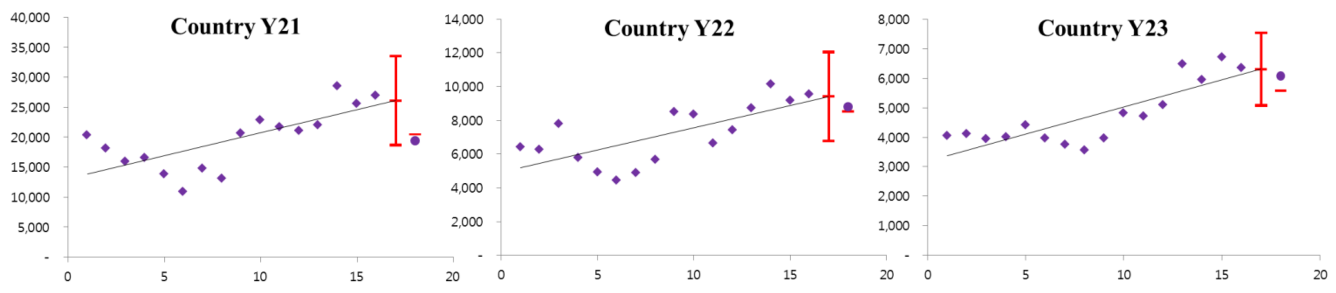

Figure 16. Interval disaggregation on deseasonalized quarterly revenues. Bars on prediction intervals are the intersecting points, bars on the right are intersecting points multiplied by the seasonal index of Q1, and circles are actual revenues of Q1 2011.

4. RELATED WORK

The needs of tools for business planning were identified as early as 2000 [31], but research on planning operations has been done only recently [12,13]. A main planning operation in the literature and in practice [11,20] is referential disaggregation, and the problem formulation of interval disaggregation proposed in this paper is new. So is the solution approach for interval disaggregation, including the feasible polytope, etc.

Referential disaggregation frequently appears not only in planning applications but also in business accounting applications. Disaggregation has been implemented as application logic or as a function that application servers provide. For instance, SAP accounting applications heavily use disaggregation functions to distribute shared costs, such as office maintenance costs, to relevant departments by a certain criteria such as the number of employees per department. It is a kind of referential disaggregation and is implemented as business functions that SAP application servers provide [24]. However, we are unaware of any effort to provide disaggregation as a relational operator in database engines.

The subset sum problem has been studied extensively [3,16,28,32], but to our knowledge the problem of finding the feasible polytope in Section 2.4, i.e., solving *d* subset sum problems simultaneously is new in the field of algorithms. So are the two algorithms in Section 2.4 that solve the problem.

Deseasonalization has been studied in the context of business forecasts [15], and Section 3.3 is a straightforward adaptation of it to interval disaggregation.

5. CONCLUDING REMARKS

We have proposed a new planning operation called *interval disaggregate* and presented solutions for interval disaggregation. Our experiments on real-world data show that interval disaggregation gives more appropriate and advanced solutions than the known basic disaggregation called referential disaggregation.

However, this is just a beginning of new research directions on business planning. More advanced disaggregate operations can be developed by incorporating various factors. For example, the relationship between revenues and costs as in the cost-volume-profit analysis [6, 10] can be considered in disaggregation, and more sophisticated constraints such as factory capacities may be taken into account. Also product life-cycles [26] rather than linear regression can be used as a trend model of historical data. Developing new kinds of business planning operations other than disaggregation will be interesting as well.

Acknowledgments. The work of Park, Ryu, and Lee was supported in part by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2011-0029924).

6. REFERENCES

[1] CGAL, Computational Geometry Algorithms Library, http://www.cgal.org

[2] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton. MAD skills: new analysis practices for big data. *Proceedings of the VLDB Endowment*, 2(2):1481-1492, 2009.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT press, 3th Edition, 2009.

[4] L. Dannecker, M. Böehm, W. Lehner, and G. Hackenbroich. Partitioning and multi-core parallelization of multi-equation forecast models. In *Scientific and Statistical Database Management,* pages 106-123. Springer Berlin Heidelberg, 2012.

[5] L. Dannecker, G. Hackenbroich, M. Boehm, U. Fischer, F. Rosenthal, and W. Lehner. A survey of forecast models for energy demand and supply. *Journal of the ACM*, 2(3):1-35, 2001.

[6] C. Drury. *Cost and Management Accounting*. CengageBrain.com, 7th Edition, 2007.

[7] JD Edwards World Forecasting Guide, Release A9.3, 2013, from Oracle: http://docs.oracle.com/cd/E26228_01/doc.93/e20706.pdf

[8] J. E. Goodman, J. O'Rourke. *Handbook of Discrete and Computational Geometry*. CRC press, Second Edition, 2004.

[9] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*. 1(1):29-53, 1997.

[10] C. T. Horngren, S. M. Datar, and M. Rajan. *Cost Accounting: A Managerial Emphasis.* Prentice Hall, 14th Edition, 2011.

[11] IBM Cognos Express Advisor User Guide, Version 9.0.0, 2009, from IBM Corp: http://download.boulder.ibm.com/ibmdl/pub/software/data/cognos/documentation/docs/en/9.0.0/ea_ug.pdf

[12] B. Jaecksch, and W. Lehner. The planning OLAP model – A multidimensional model with planning support. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems VIII*, pages 32-52. Springer Berlin Heidelberg, 2013.

[13] B. Jaecksch, W. Lehner, and F. Faerber. A plan for OLAP. In *Proceedings of the 13th International Conference on Extending Database Technology,* pages 681-686. ACM, 2010.

[14] D. L. Kroshko. Technical Report, 2014, from OpenOpt: http://openopt.org/interalg

[15] D. A. Lind, W. G. Marchal, and S. A. Wathen. *Basic Statistics For Business and Economics*. Boston: McGraw-Hill/Irwin, 2006.

[16] S. Martello, and P. Toth. A mixture of dynamic programming and branch-and-bound for the subset-sum problem. *Management Science*, 30(6):765-771, 1984.

[17] J. Matoušek. *Lectures on Discrete Geometry*. Springer, Vol. 212, 2002.

[18] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to Linear Regression Analysis*. Wiley, 5th Edition, Vol. 821, 2012.

[19] Optimization Toolbox™ User's Guide, Release 2014a, 2014, from MathWorks, Inc: http://www.mathworks.co.kr/help/releases/R2014a/pdf_doc/optim/optim_tb.pdf

[20] Oracle Hyperion Planning – System 9, Release 9.3.1, 2007, from Oracle: http://docs.oracle.com/cd/E10530_01/doc/epm.931/hp_user.pdf

[21] T. B. Pedersen, C. S. Jensen, and C. E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Information Systems*, 26(5):383-423, 2001.

[22] F. P. Preparata, and M. I. Shamos, *Computational Geometry: An Introduction*. Springer-Verlag New York, 1985.

[23] S. Ross. *A First Course in Probability*. Pearson Education India, 9th Edition, 2012.

[24] SAP ERP, Business Functions, SAP EHP1 for SAP CRM 7.0, 2013: http://help.sap.com/saphelp_crm700_ehp01/helpdata/en/11/53929b94c84bc89334719b1606c4de/content.htm?frameset=/en/d5/b9e2574bbd4373b59fce413325f731/frameset.htm

[25] SAP HANA Predictive Analysis Library, ver. 1.0, 2013, from SAP: http://help.sap.com/hana/SAP_HANA_Predictive_Analysis_Library_PAL_en.pdf

[26] J. Stark. *Product Lifecycle Management.* Springer London, 2011.

[27] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts*. Hightstown: McGraw-Hill, 6th Edition, 2009.

[28] N. Y. Soma, and P. Toth. An exact algorithm for the subset sum problem. *European Journal of Operational Research*, 136(1):57-66, 2002.

[29] G. B. Thomas, R. L. Finney, and M. D. Weir. *Calculus and Analytic Geometry*. Reading, Massachusetts: Addison-Wesley, Vol. 9, 1996.

[30] M. S. Waterman, and T. H. Byers. A dynamic programming algorithm to find all solutions in a neighborhood of the optimum. *Mathematical Biosciences*, *77*(1):179-188, 1985.

[31] G. Wiederhold. Information systems that really support decision-making. *Journal of Intelligent Information Systems*, 14(2-3):85-94, 2000.

[32] G. J. Woeginger. Exact algorithms for NP-hard problems: a survey. *Combinatorial Optimization—Eureka, You Shrink!*, pages 185-207, Springer Berlin Heidelberg, 2003.