Can’t See the Wood for the Trees? Using Matrix Approximation for Solving Large-Scale Resource Allocation Problems in Data Centers

(Authors’ names blinded for peer review)

We consider the assignment of enterprise applications in virtual machines to physical servers, also known as server consolidation problem. IT service managers try to minimize the number of servers, but at the same time provide sufficient computing resources at each point in time. While historical workload data would allow for accurate workload forecasting and optimal allocation of enterprise applications to servers, the volume of data renders this task impossible for any but small instances. We look at the general problem of modeling large volumes of workload data in order to extract significant features and use these features to allocate VMs efficiently to physical servers using optimization. The geometric interpretation of significant features derived from singular value decomposition allows us to transform the original allocation problem in a low-dimensional integer program. We evaluate the approach using workload data from a large IT service provider and show that it leads to high solution quality, but at the same time allows for solving considerably larger problem instances than what would be possible without data reduction. The overall approach can also be applied to other large integer problems, as they can be found in applications of multiple or multi-dimensional knapsack or bin packing problems.

Key words: Matrix Approximation, Multi-Dimensional Packing, Vector Packing, Server Consolidation, Dimensionality Reduction, Truncated Singular Value Decomposition

1. Introduction

Energy usage has become a key cost drivers in data centers already accounting for 30 to 50% of the total operation costs (Filani et al. 2008). Because each piece of hardware consumes energy, a key concern for IT service managers is capacity management, a planning task to ensure that sufficient IT infrastructure resources are provided to maintain service level agreements (SLA) cost and energy efficiently (OGC 2000). Recent surveys of IT managers reveal that capacity management remains the top operational concern in data centers (Haight et al. 2009, Forrester Research, Inc. 2010).
Today’s larger data centers are typically hosting thousands of enterprise applications from various customers such as Enterprise Resource Planning (ERP) modules or database applications with different workload characteristics. The underlying IT infrastructures may contain hundreds of physical servers and various other hardware components. There is even a trend to increase the size of data centers to leverage economies of scale in their operation (Digital Realty Trust 2010).

To reduce the number of required servers, server virtualization has been increasingly adopted during the past few years (Hill et al. 2009). Server virtualization refers to the abstraction of hardware resources and allows the hosting of multiple virtual machines (VM) including business applications plus underlying operating system on a single physical server. The capacity of a physical server is then shared among the VMs, leading to increased utilization of physical servers.

Capacity planning and resource allocation have long been central research issues in Computer Science and Information Systems (Menasce et al. 2004, Hellerstein et al. 2004). While traditional planning approaches rely on queuing theory or simulation to predict the performance of a single dedicated system given available resources and resource demands (Markl et al. 2010), new challenges arise in the capacity planning for virtualized data centers.

We consider the problem of assigning enterprise applications with time-varying demands for multiple hardware resources such as CPU, memory, or I/O bandwidth efficiently to servers with finite capacities. In this domain, economic efficiency means the use of computing resources so as to maximize the number of IT services provided. These IT services must get enough resources, in order to conform to pre-defined SLAs. This problem has been described as the server consolidation problem (Speitkamp and Bichler 2010). This problem is important, because data centers regularly need to assign or reassign a set of VMs to a new hardware infrastructure.

Nowadays, VM managers such as VMWare or Xen allow for a dynamic reallocation of VMs during runtime (Clark et al. 2005). As of now, there is little understanding of the dynamics and the stability of the overall system that might emerge as a result of automated reallocations of VMs. Such reallocations require a significant amount of CPU and memory resources on the physical servers as well as network bandwidth. Repeated daily demand peaks might lead to many reallocations for
short time periods when dynamic reallocations are used. Therefore, IT managers prefer to search for stable allocations, which minimize the number of servers but require no regular reassignments.

1.1. Server Consolidation

Data centers usually measure resource utilization for each VM and for each physical server. For example CPU utilization, allocated memory, and utilized network bandwidth of a server are typically logged every minute or in five-minute intervals. Such log data can then be used for characterizing and predicting future resource demands and determining efficient VM allocations.

Industry practice usually follows simple approaches to consolidate servers based on peak workloads observed in the past. For example, administrators collect maximum VM demands for various resources over several weeks and derive a respective VM demand vector. VMs are then assigned to a physical server at a time until the demand exceeds the capacity.

Many enterprise applications exhibit predictable workloads with daily or weekly seasonalities (Rolia et al. 2005, Speitkamp and Bichler 2010). Enterprise applications with predictable workloads are actually some of the prime candidates for server consolidation projects as these allow for reliable resource planning and high potential server savings. For example, email servers typically face high loads in the morning and after the lunch break when most employees download their emails, payroll accounting is often performed at the end of the week, while workload of a data warehouse server has a daily peak very early in the morning when managers typically access their reports. A number of heuristics have been suggested in the academic literature to allocate VMs to physical machines more efficiently by leveraging complementarities among the workload patterns of different VMs and further reduce the server count. Similarly, it makes sense to combine VMs with higher CPU workload with data-intensive workloads with lower CPU but higher I/O utilization.

Cherkasova and Rolia (2006) describe an approach to minimize the number of servers based on genetic algorithms that penalize low server utilizations and server overload. Seltzam et al. (2006) also determine characteristic workload profiles and use a heuristic to determine an initial allocation of VMs to servers. Their heuristic is trying to balance the workload across servers.
We will introduce the multi-dimensional bin-packing (MBP) problem formulation as a succinct description of the server consolidation problem, as it will provide a foundation for the contribution in this paper. Suppose that we are given $J$ VMs $j = 1, \ldots, J$ to be hosted by $I$ servers $i = 1, \ldots, I$ or fewer. Different types of resources $k = 1, \ldots, K$ such as CPU, I/O, or main memory may be considered and each server has a certain capacity $s_{ik}$ of each resource $k$. $y_i$ is a binary decision variable indicating if server $i$ is used, $c_i$ describes the cost of a server (for example, energy costs per hour), and the binary decision variable $x_{ij}$ indicates which VM $j$ is allocated to which server $i$. The planning period is divided into time intervals indexed by $t = 1, \ldots, T$. These intervals might be minutes or hours per day if we have daily periodicity. Let $r_{jkt}$ be the capacity that VM $j$ requires of resource $k$ in time period $t$. The server consolidation problem can now be formulated as the multi-dimensional bin packing problem (MBP) in (1).

$$\begin{align*}
\min & \sum_i c_i \cdot y_i \\
\text{s.t.} & \sum_j x_{ij} = 1 & \forall j \leq J \\
& \sum_j r_{jkt} \cdot x_{ij} \leq s_{ik} \cdot y_i & \forall i \leq I, \forall k \leq K, \forall t \leq \tau \\
& y_i, x_{ij} \in \{0, 1\} & \forall i \leq I, \forall j \leq J 
\end{align*}$$

The objective function minimizes total server costs. The first set of constraints ensures that each VM is allocated to one of the servers, and the second set of constraints ensures that the aggregated resource demand of multiple VMs does not exceed a server’s capacity per host server, time interval, and resource type. As we will show in Section 4, $r_{jkt}$ can typically be estimated with sufficient quality from workload data.

Speitkamp and Bichler (2010) have shown that the consideration of daily workload cycles can lead to 31% savings in physical servers based on smaller instances of up to 50 ERP applications or 200-300 Web applications, but considering only CPU and aggregating the workload data to 15 minute time intervals. Considering memory and I/O is essential in most projects, but would not be tractable with this approach except for small problem sizes with ten or twenty VMs. Taking only larger time intervals into account can lead to a low solution quality either, as we lose valuable information about the demand patterns of the VMs. When we assume daily workload cycles and
Consider resource requirements for all five-minute time intervals per day, resource demand behavior is described by 288 distinct time intervals ($T = 288$). So the number of resource constraints in (1) grows $T$ times the number of resources $K$ and the number of servers $I$. In summary, with a larger number of relevant resource types $K$ and time intervals $T$ considered the applicability of such an optimization approach can be limited to only a few dozen virtual machines. Linear programming-based heuristics, which allow for the consideration of important technical allocation constraints have been evaluated, but did not lead to a much better scaleability compared to exact solutions. Although the approach has shown to yield considerable savings in server cost, this curse of dimensionality is a barrier to any practical application in data centers, where often hundreds of VMs need to be considered. In this paper, we will suggest a new approach which allows to solve significantly larger problem instances, but at the same time maintaining a high solution quality.

1.2. Solving Large-Scale Resource Allocation Problems

The sheer volume of workload data of hundreds of VMs is a challenge when applying mathematical optimization to solve the resource allocation problem. There is a significant literature on solving large-scale integer programs with many variables which consist of sparse or specially structured constraint matrices using techniques such as column generation. An overview of such techniques can be found in Barnhart et al. (1996). In our problem, however, the constraint matrix has many constraints rather than many variables. Furthermore, the constraint matrix is not sparse and it does not exhibit a special block diagonal structure as is required for Danzig-Wolfe composition algorithms and branch-and-price approaches.

Polynomial time approximation schemes (PTAS) have been developed for multi-dimensional bin packing as well, but the gaps between the upper and lower bounds on the solution quality increase with the number of dimensions, which limits their applicability to problems with a low number of dimensions. For $d$-dimensional vector packing, for instance, with $d$ as an arbitrary positive integer value, no polynomial-time approximations are at hand with performance guarantees better than $\ln(d) + 1$, as achieved by Bansal et al. (2006) using a polynomial-time randomized algorithm.
For practical problems with larger values of $d$ – as in our problem under consideration – this performance guarantee might be unacceptable. In addition, in contrast to mixed-integer programs where important technical allocation constraints are considered by adding additional constraints to the program, these approximation algorithms need to be adapted. No performance guarantees are known for such adaptations.

If the problem size considering all resource types and time intervals is intractable, IT managers can try to reduce the dimensionality of the problem by aggregating or even ignoring dimensions. Reducing the number of resource types by ignoring for instance I/O or memory can lead to capacity violations on these resources and might not be an option. However, the number of time intervals can be reduced by aggregating sets of consecutive intervals and taking the maximum over these intervals. So, instead of five-minute intervals, one can only consider hourly intervals. At certain times, when VM resource demands do not vary significantly in an aggregated time interval, such a reduction would not impact the solution quality in terms of the number of servers required. If there are short, time-delayed demand peaks for VMs in a one-hour interval, the demand for the entire hour will be overestimated. This will lead to a higher number of servers than necessary.

In this paper, we will propose an approach to reduce the matrix of workload data by losing as little relevant information as possible for our central resource allocation problem, at the same time being able to solve much larger problem sizes.

We focus on the server consolidation problem with time varying, but predictable workloads. In our data set, most workloads exhibited significant daily seasonalities and high regularity across all working days of a week.

We will ignore applications with erratic demand patterns, as can be observed for some online shopping systems of products with uncertain demand. Optimization can also be applied in these cases, but the demand estimates need to be much more conservative.

1.3. Contributions

Overall, we propose a new approach to solving high-dimensional packing and covering problems with a very large number of constraints based on low-rank matrix approximation. Although we
focus on server consolidation, the method is more general and can be applied to high-dimensional knapsack or bin packing problems as can be found in logistics, manufacturing, or inventory management. This approach is based on the following contributions:

First, we propose a concise description of data center workloads using truncated singular value decomposition (SVD). SVD is used to learn an optimal orthonormal transform from the high-dimensional workload data of hundreds of VMs. The transform allows for a compact representation of the patterns in the data with a smaller, more manageable set of new dimensions derived by the dominant singular vectors of the SVD. The approach exploits the fact that resource demands in proximate time intervals as well as for different resource types are correlated.

Second, we provide a geometric interpretation of significant features extracted from the low-dimensional space spanned by the dominant singular vectors. These features reduce the data to a more manageable set while preserving the relevant information within the data. The extracted patterns or features can be used for solving the server consolidation problem (see below), but it can also be used to automatically identify and visualize significant trends and longer-term developments in workload behavior of large numbers of VMs and servers. We will propose simple two-dimensional visualizations of the workload behavior of many VMs and servers. To our knowledge this is the first paper to use SVD for the analysis of workload data in IT service management.

Third, our main contribution will be an optimization formulation based on geometric interpretation of the features to determine resource allocation in virtualized data centers. We do not know of any other paper, which uses features extracted from SVD for solving resource allocation problems.

Fourth, we provide the results of extensive experimental analyses of the solution quality measured by the number of servers required. We will show that, in contrast to exact approaches, the SVD-based optimization computes feasible solutions for larger problems with hundreds of servers. In addition we will show that the shape of the singular value spectrum is an indicator of the suitability of the approach for a particular problem instance.

Several researchers in the IS design science community have used optimization successfully to solve problems in computer systems such as databases (Bala and Martin 1997, Gopal et al. 2001,
1995), for caching (Kaya et al. 2009, Dutta et al. 2006), for efficient resource allocation in grids (Kumar et al. 2009) and content delivery sites (Liu et al. 2010), and for networking (Kennington and Whitler 1999, Laguna 1998). Our paper is a form of prescriptive work in IS research as outlined by March and Smith (1994), aiming at improving IT performance building and evaluating IT artifacts. It follows the guidelines set out by Hevner et al. (2004) including research contribution, problem relevance, evaluation, and research rigor among other criteria.

From an optimization perspective, the problem is relevant as it provides a new approach to solving large scale bin packing and knapsack problems with many constraints (see also Section 1.2). Kellerer et al. (2004, p. 236) mention that multidimensional knapsack is a particularly difficult instance of integer programming because the constraint matrix is unusually dense whereas most relevant cases of integer programs are defined by sparse constraint matrices. The same holds for multidimensional bin packing. Server consolidation is a good example of such problem types.

The remainder of this paper is structured as follows. In Section 2 we review dimensionality reduction of multivariate data using SVD. In Section 3 we develop a mathematical model to solve the consolidation problem based on the features derived using SVD, and show how SVD coordinates can be used to track workload behavior. In Section 4 we evaluate the model using data from a professional data center. Finally, in Section 5 we conclude and discuss the managerial impact of the proposed solution for IT service management and alternative domains.

2. **Workload Analysis and Feature Extraction**

Feature extraction is an important area in statistics, machine learning, and pattern recognition used to reduce the dimensionality of data. For the problem considered in this work, for each VM the description of demand over time (time dimensionality) for multiple resource (resource dimensionality) must be reduced. Feature reduction techniques can then be used to transform the input data into a set of features, which extract the relevant information from the input data. There is a large body of literature on feature extraction and also multivariate time series analysis, which can be used for such purposes (see Hair et al. (2009) for an overview).
We will focus on singular value decomposition (SVD) for the following reasons. Golub and van-Loan (1990) have shown that SVD derives the best approximation of a matrix with a matrix of a given rank. In other words, given a defined number of target dimensions to approximate high-dimensional data, the application of SVD will result in the lowest possible aggregated approximation error. The decomposition can be computed in quadratic time and fast SVD approximations exist (P. Drineas et al. 2003, Frieze et al. 1998). In addition, SVD is a model-free approach and no parameters need to be defined or adjusted. Furthermore, it is applicable to arbitrary data matrices including non-square and not fully ranked workload matrices. In contrast to similar approaches such as Principle Component Analysis (PCA), SVD preserves the absolute scales of workload levels, which is important if the resulting features are then to be used in a resource allocation task.

We will briefly introduce the mathematical concepts relevant to this paper.

2.1. Truncated Singular Value Decomposition

Again, we will assume a matrix with workload data about a set of $J$ VMs (named items) $j$ ($j \in \{1,\ldots,J\}$), with varying resource demand (or workload) $r_{jkt}$ for different resource types $k$ ($k \in \{1,\ldots,K\}$) such as CPU or memory demand over a period of observation. The period of observation is divided into discrete time intervals indexed by $t$ ($t \in \{1,\ldots,T\}$). These might be five-minute intervals based on the monitoring frequency in data centers. In our context, we refer to a data dimension as the utilization of a particular resource in a particular time interval, i.e., a unique tuple $(k,t)$. The resulting $K$ time $T$ tuples describing an item’s workload time series for $K$ resources can be represented as a point in an $L$-dimensional Hilbert space, with $L = K \cdot T$.

To reduce dimensionality, we want to project these points from an $L$-dimensional to an $E$-dimensional space so that $E << L$. Let $R$ be the $J \times L$ matrix of $J$ items, where $l$ indicates a unique tuple $(t,k)$, such that $r_{jl}$ describes how much capacity an item $j$ requires in dimension $l$, i.e., of resource $k$ in time period $t$. Hence, each row vector in $R$ corresponds to workload time series for one or more resources of a single VM. For our purposes, we can restrict the attention to matrices of positive real numbers as resource demands are generally non-negative.
Golub and van Loan (1990) have proven that a matrix $R$ can be factorized by SVD to $U\Sigma V^T$, where $R$’s singular values $\sigma_e$ in the diagonal matrix $\Sigma$ (with $e \in \{1, \ldots, rank(R)\}$) are ordered in non-increasing fashion, $U$ contains the left singular vectors (the Eigenvectors of $R^T R$), and $V^T$ contains the right singular vectors (the Eigenvectors of $RR^T$). Here, $V^T$ denotes the conjugate transpose of $V$. All left and right singular vectors are mutually orthogonal and normalized by their ‘length’, defined in terms of the $L^2$ norm (i.e., an extension of the Euclidean distance to $d = \sqrt{(a^2 + b^2 + c^2)}$ in a three-dimensional space).

An interpretation of this factorization is that the workload behavior of an item $j$ (the $j$th row vector in $R$) can then be expressed by a linear combination of the right singular vectors as these define new axes, which form a basis of $R$’s row space. Sometimes in the literature, the columns of $V$ are called the principle components, since they form the basis for our data, i.e., for the row space of the data matrix. The $e^{th}$ singular value is a scaling factor for the $e^{th}$ axis (as it is associated with the $e^{th}$ right singular vector), and the $e^{th}$ element in the $j$th row in $U$ indicates the coordinate of $j$’s workload along the new dimension $e$. These coordinates are the features we extract from $R$.

As an illustration how SVD works, consider workload $r_{jl}$ of items $j$ in $L = 2$ dimensions (e. g., the maximum workload demand for one resource $K = 1$ observed during daytimes ($l := 1$) and nighttimes ($l := 1$)). Resulting data points of item workloads are shown in Figure 1.

For each $j$, coordinates (or features) $u_{j1}$ in the first subspace, i.e., along $v_1$, the first right singular vector in $V^T$, are calculated by perpendicular projection of the original points onto $v_1$ as shown for item 1. These coordinates show the best one-dimensional approximation of the data because $v_1$
captures as much of data variation as possible in one direction. In other words, \( v_1 \) is a regression line running through the points in the direction where the data is most widely spread.

Item coordinates \( u_{j2} \) regarding the second right singular vector \( v_2 \) \((v_2 \perp v_1)\) then capture the maximum variance after removing the projections along \( v_1 \). In this two-dimensional example, \( v_2 \) captures all of the remaining variance. In general, the number of singular vectors equals \( R \)'s rank.

Truncated SVD (tSVD) only considers the \( E \) dominant subspaces, i.e., the \( E \) row vectors of \( V^T \) and the \( E \) column vectors of \( U \) corresponding to the \( E \) largest singular values in \( \Sigma \). Hence the rest of the matrices is discarded and the original matrix \( R \) is approximated by \( \sum_{e=1}^{E} u_e \sigma_e (v_e)^T \). The idea is that variation below a particular threshold \( E \) can be ignored, if \( R \)'s effective rank does not exceed \( E \) (i.e., the rank corresponding to the \( E \) dominant singular values which are 'significantly' greater than zero), as the singular values associated with the right singular vectors sort them in the order of the explained variation from most to least significant. Depending on the decay of singular values, very large matrices can be approximated with high accuracy using only a very small set of singular values. Workload time series of enterprise applications usually exhibit daily or weekly cycles and can be approximated well with only a small set of eigentriples. The resulting approximation errors are analyzed in the next subsection.

2.2. Matrix Norms and Approximation Errors

Let \( R_E := \sum_{e=1}^{E} u_e \sigma_e (v_e)^T \) be \( R \)'s approximation derived with the first \( E \) subspaces. Let then \( R_E^\epsilon := R - R_E \) be \( R_E \)'s approximation error matrix (or residual matrix) with elements \( r_{ij}^\epsilon \). Although the tSVD is no longer an exact factorization of \( R \), Eckart and Young (1936) have proven that \( R_E \) is – in a useful sense for our purposes – the closest approximation to \( R \) possible with a matrix of rank \( E \) taking the Frobenius norm \( ||\cdot||_F \) or the spectral norm \( ||\cdot||_2 \) (also known as matrix 2-norm) as a measure. Such matrix norms extend the concept of vector norms or distances to matrices. They have shown that it is impossible to find a matrix \( A \) with rank \( E \) that satisfies (2).

\[
||R_E^\epsilon||_F > ||R - A||_F \quad OR \quad ||R_E^\epsilon||_2 > ||R - A||_2
\]  

(2)
Consider the scenario depicted on the left-hand side of Figure 2. The graphs describe one-hour aggregates of diurnal CPU workloads of 16 arbitrarily chosen enterprise applications taken from a data center log file from a large European IT service provider. The underlying workload matrix $R$ is of rank 16. Workload levels are described in percent relative to the overall CPU performance of a server.

The graph on the right-hand side of the figure shows the reconstruction of the VM workload curves with $E := 3$, i.e., using the first three SVD subspaces. The graphs in Figure 3 show the data reconstruction when using four and five subspaces, respectively. While the data approximation with three SVD subspaces already yields the principal workload behavior, the approximation with five subspaces captures the workload behavior, in particular for VM with higher workload that are of particular interest for our purposes, with high accuracy.

The approximation works well due to the significant initial decay in $R$’s singular value spectrum shown in Figure 4. The spectrum exhibits a sharp drop at the beginning and then continues with a rather moderate decay at a relatively low level of singular values, and higher-order singular values above five have negligible impact on the workload description. Hence, the aggregated impact or weight of the first five singular values is much higher than the sum of weights of the remaining singular values.
The Ky Fan $E$ norm of a matrix $R$ indicates the aggregated weight of the first $E$ subspaces formally, as the norm is defined to be the sum of the $E$ largest singular values of a matrix $R$ as defined in (3), given a positive integer $E$ such that $E \leq \text{rank}(R)$.

$$||R||_{K_E} := \sum_{e=1}^{E} \sigma_e$$  \hspace{1cm} (3)

The Ky Fan $F$-norm of $R_{E}^F$ (with a positive integer $F \leq \text{rank}(R) - E$) equals the sum of $R$’s singular values $\sigma_e$ from $e = E+1, \ldots, E+F$ as shown in (4).

$$||R_{E}^F||_{K_F} := \sum_{e=E+1}^{E+F} \sigma_e$$  \hspace{1cm} (4)

For $F := 1$ it follows that $||R_{E}^1||_{K_1} = \sigma_{E+1}$, where $\sigma_{E+1}$ is the largest singular value of $R_{E}^F$. $R_E$’s maximum approximation error equals $R_{E}$’s max-norm $\max_{j,l} |r_{jl}|$ ($R_{E}$’s largest absolute entry).
Since the max-norm of a matrix is upper-bound by its largest singular value, the shape of $R$’s singular value spectrum reflects the maximum approximation error when using a particular number of subspaces. We will use this information later to consider the approximation error adequately.

The histogram of entries in the approximation error matrices $R_E$ is depicted in Figure 5.

While the plot on the left-hand side shows the histogram for $E = 3$, the plot on the right-hand side shows the histogram for $E = 5$. Per definition, the mean value in $R_E$ is zero, and the sum of squared negative values equals the sum of squared positive values.

Increasing $E$ will decrease the maximum absolute value (absolute error) in this empirical distribution. Furthermore, increasing $E$ will also decrease the variance in this empirical distribution.

The Frobenius norm is defined as the square root of the sum of the values squared over all the elements of a matrix as defined in (5). The norm is typically used as a measure to quantify total matrix approximation accuracy. SVD-based matrix approximation also minimizes the Frobenius norm of the residual matrix given any particular number of dimensions (refeqn:frobenius).

$$||R||_F := \sqrt{\sum_{j=1}^{J} \sum_{l=1}^{L} (r_{jl})^2} = \sqrt{\sum_{e=1}^{\text{rank}(R)} (\sigma_e)^2}$$ (5)

Increasing $E$ will lead to a closer approximation to $R$ and smaller approximation errors. This should lead to better solution quality of the resulting resource allocation at the cost of a larger number of constraints to be included in the optimization. This will be discussed in more detail in the next section.
In most workloads of our data set, the singular value spectrum begins with a sharp decay of the first few singular values and then decreases slowly. This is, because the workload of applications typically follows a clear seasonal pattern and exhibits high autocorrelation.

2.3. Complementarities in Workload

From the fact that all entries in $R$ are non-negative it follows that all entries in $v_1$ and all VM coordinates $u_{j1}$ will be non-negative. Hence, $v_1$ not only captures the maximum variance in the data but can also be interpreted as a regression line, which describes the direction with the highest aggregated (squared) workload. We will refer to $v_1$ as the major workload direction. Consider the scenario depicted on the left-hand side of Figure 6. For this illustration, we only consider two dimensions or time intervals. The coordinate $u_{j1}$ of an item $j$ along $v_1$ fully describes $j$’s workload as $\sigma_e = 0$ for all $e > 1$.

However, as described in the previous subsection, typically $\sigma_2, \sigma_3, \ldots$ are non-zero and a workload estimate based on $v_1$ might lead to a significant approximation error. In the scenario depicted on the right-hand side of Figure 6, additional items $a$-$d$ with equal $u_{j1}$ but different $u_{j2}$ coordinates are considered. $u_{j2}$, the feature or coordinate of a VM $j$ along $v_1$, captures the Euclidean distance of $j$’s workload to the new axis $v_1$, which means, the approximation errors when considering $v_1$ only.

Workload in $t=1$ ($t=2$) is overestimated (underestimated) by $u_{j1}$ for items with positive $u_{j2}$ ($a$ and $b$); the opposite is true for items with negative $u_{j2}$ ($c$ and $d$). Hence, when combining VMs with positive and negative $u_{j2}$, for example $a$ and $d$ on the same physical server, the aggregated errors on $v_1$ compensate each other. Furthermore, as $v_1$ exhibits the average direction of workload over
all virtual machines – their most common resource demand pattern – for items with positive $v_2$ coordinates like $a$ and $b$, workload is below average in $t=1$, and above average in $t=2$. The same holds vice versa for items with negative $v_2$ coordinates. Therefore, the $v_2$ coordinates explicitly indicate workload complementarities. A key to efficient server consolidation is to leverage such complementarities in the workloads. The information will be used in the optimization formulation in the next section.

3. **SVD-Based Resource Allocation**

Based on the geometric interpretation of singular vectors in the previous section we will now propose a mixed integer program for the server consolidation problem to find an efficient allocation of VMs to servers based on the new coordinate system of the first SVD subspaces.

3.1. **Problem Formulation Based on the Dominant Subspace**

For introductory purposes, we will first assume that the coordinates $u_{j1}$ represent the resource demands of individual items well. Higher-order singular values are zero or very small and can be ignored. The problem can now be solved as a one-dimensional bin packing problem, where $u_{j1}$ is item $j$’s size, and the capacity limits of physical servers are the bin sizes.

Bin sizes are determined as follows. For each of the $K \cdot T$ original dimensions the capacity constraint for resource $k$ of a server or bin $i$ is $s_{ik}$ (for all $t$). Hence, for each server we obtain hyperplanes which form a convex polyhedron indicating its capacity limits on the various resources. The dashed lines in Figure 6 describe the capacity constraints for server $i=1$. In direction $v_1$, the capacity limit is the point $P_i$, where $v_1$ and a hyperplane intersect. Hence, $i$’s bin size equals $||P_i||$, the Euclidian norm of the vector from origin to $P_i$.

With $y_i$ as a binary decision variable indicating if a target host server $i$ is used, $c_i$ as cost of a target bin (e.g., energy costs of a server over a planning period), and with $x_{ij}$ as a binary decision variable indicating which item $j$ is allocated to which bin $i$, the problem can be formulated as a binary program in (6).
The objective is to minimize total costs and the first constraint ensures that each item is allocated exactly once. The second constraint ensures that the aggregated $v_1$-based workload estimate of items assigned to a bin does not exceed the bin’s capacity limit.

### 3.2. Consideration of Higher-Order Subspaces

Typically, higher-order singular values of $R$ are non-zero and one needs to take into account the deviations from the workload estimates in dimension $v_1$. Let us first focus on $e = 2$ as $v_2$ is the major direction of the deviation from $v_1$. Higher dimensions of $e > 2$ can then be handled analogously.

In order to avoid overload situations, $u_{j2}$, the deviations from $v_1$ need to be added to the $v_1$ coordinates of each item. As we have seen in Section 2.3, positive and negative coordinates $u_{j2}$ describe complementarities in the workload. Combining only items with positive $u_{j2}$ in a bin increases the approximation error from $v_1$, while the opposite is the case if we combine VMs with positive and negative coordinates $u_{j2}$. In the example in Figure 7 item $g$ has a lower $v_1$-based workload estimation than $a$ and $d$.

When dimension $v_2$ is ignored, co-locating $a$ and $g$ would lead to a lower aggregated workload estimation than a co-location of $a$ and $d$. By considering $v_2$ coordinates, the opposite is true and the estimated workload demand of bundle $(a, g)$ exceeds $(a, d)$’s workload, because the aggregated
deviation on \( v_2 \) is added to the \( v_1 \) workload estimation. Combining \( a \) and \( d \) would reduce the total approximation error and leverage the complementarities in the workloads, which needs to be considered in the optimization.

Let \( z_{i2} \) be the absolute sum of \( u_{j2} \) coordinates of items assigned to server \( i \). To avoid server overload, \( z_{i2} \) must be added to the aggregated \( u_{j1} \) coordinates of items assigned to \( i \) to ensure sufficient capacity in all original dimensions, i.e., for all time intervals and for all resources. The resulting program considering deviations from \( v_1 \) is shown in (7), where we assign variables for positive and negative \( z_{i2} \).

\[
\begin{align*}
\min & \sum_i c_i \cdot y_i \\
\text{s.t.} & \\
\sum_{j \leq J} x_{ij} & = 1 \quad \forall j \leq J \\
\sum_{j \leq J} (u_{j2} \cdot x_{ij}) - z_{i2}^+ + z_{i2}^- & = 0 \quad \forall i \leq I \\
\sum_{j \leq J} (u_{j1} \cdot x_{ij}) + z_{i2}^+ + z_{i2}^- & \leq ||P_i|| y_i \quad \forall i \leq I \\
y_i, x_{ij} & \in \{0, 1\} \quad \forall i \leq I, \forall j \leq J \\
z_{i2}^+, z_{i2}^- & \geq 0 \quad \forall i \leq I
\end{align*}
\]

(7)

We will refer to this problem formulation in (7) as SVD-based packing or SVPack. Again, the objective is to minimize server costs and the first constraint ensures that each VM is allocated exactly once. The second constraint determines \( z_{i2}^+ \) and \( z_{i2}^- \), which are either positive or negative deviations on \( v_2 \) required in the third constraint. The third constraint ensures that the aggregated \( v_1 \)-based workload estimate of VMs assigned to a target plus the deviations \( z_{i2}^+ \) and \( z_{i2}^- \) do not exceed the target’s capacity limit. Variations along \( v_3, v_4, \ldots \) can be considered similar to those in \( v_2 \), if the respective singular values are significant. For each \( e \{2 < e < E\} \) and each server \( i \) we introduce a constraint to determine \( z_{i2}^e \) and \( z_{i2}^-e \) and add those variables to the left-hand side of the third constraint type in (7).

### 3.3 Workload Analysis and Visualization

SVD provides a powerful method to extract relevant features from large data sets. While the main focus of this paper is on server consolidation, we also want to discuss applications of SVD to derive metrics that can support management decisions. Here, we will focus on the monitoring of relevant workload characteristics over time.
Server consolidation aims at finding an efficient allocation of VMs to servers, such that reallocations are unnecessary, if workload patterns do not change. In virtualized data centers, some VMs are withdrawn over time, while new ones need to be deployed. Also, workload patterns can change over time. For example, a VM hosting an e-commerce application might face increased workload around Christmas. In order to avoid overload situations, a controller needs to monitor workload developments and migrate VMs before the overall workload on a physical server hits the server’s capacity bounds.

However, migrations cause significant CPU and network bandwidth overheads and must be triggered early, before a system’s workload becomes too high. During the duration of a migration, additional CPU cycles up to 70% of the actual CPU demand of a VM are required, and the network bandwidth usage can saturate even a 1 GB/s network link for minutes to transfer the memory of a VM from one server to another. The CPU overhead is mainly due to additional I/O operations for main memory transfer from the source to the target server and the memory management via shadow pages.

Any type of manual or automated control needs to monitor workload data from thousands of VMs in a data center and detect relevant deviations in the workload behavior, which might cause a server to be overloaded. This is a challenging task considering the sheer volume of data. Feature extraction via SVD can be used to derive metrics to visualize relevant information about workload status and resource allocation decisions. Just looking at the first two dimensions of the singular vectors can provide a concise summary of a server’s state without looking at large volumes of time series data.

Consider a scatterplot per server, in which the coordinates \( u_{j1} \) and \( u_{j2} \) of the VMs, which are deployed on a common server, are depicted as shown in Figure 8. In the left-hand diagram in Figure 8, different VMs assigned to server Host 1 are shown as single dots, representing their workload estimates per day. The sum of the coordinates provides the total workload on Host 1 for that day. The dashed lines show the capacity limit of Host 1. The capacity limit decreases with increasing \( u_{i2} \) coordinate of a server \( i \) as described in the previous section. The point representing
the host’s workload is within a grey area of acceptable server utilization. The status of Host 2 in the middle of the figure can be considered as critical as the capacity is almost exhausted, and Host 3 would be considered underutilized as the expected degree of utilization is low and more VMs could be assigned to this server. By monitoring server coordinates over multiple days, relevant developments in overall workload per server or VM can be analyzed and a potential overload situation anticipated.

4. Evaluation

In this section, we will analyze the solution quality and computational effort of different approaches to solving the server consolidation problem. We consider consolidation scenarios with different sets of VM workload traces to be assigned to physical servers with multiple scarce resources such as CPU or main memory. First, we describe the available data set and the research design, before discussing the main results.

4.1. Data and Component Day Extraction

The workload data provided by a large European data center operator describes the CPU, I/O, and main memory usage monitored in intervals of five minutes over ten consecutive weeks for 458 enterprise applications. The applications stem from a diverse set of customers in various industries and include Web servers, application servers, database servers, and ERP applications.

Similar to Parent (2005), we found that a majority of the business applications hosted in data centers exhibit quite regular and cyclic workload patterns. Figure 9 provides a typical example,
which illustrates the maximum hourly CPU utilization of a VM hosting an SAP supply-chain-management application over five subsequent days, each line describing a day. Utilization levels are relative (in percent) to the server’s CPU capacity. Seasonal or cyclic components of time series can be detected with standard techniques, such as autocorrelation functions or spectral analysis of the workload time series (Hamilton 1994). The dataset has also been used by Speitkamp and Bichler (2010), who provide a detailed statistical description of the workload data in the appendix of the paper. They found that daily cycles are the dominant pattern in the 458 enterprise workloads. We found only a few time series with a significant trend component and will only focus on applications without significant trend in this paper.

As mentioned above, periodic patterns in workload data can be leveraged in server consolidation, where the selection of VMs with complementary workload traces allows the multiplexing of resources over time and the utilizing of resources more efficiently. We will extract the daily cycles in the data using SVD and use the resulting pattern in order to determine an allocation of VMs to physical servers. Note that we could use any other forecasting technique to extract a similar model of the daily workload pattern of a VM $j$. We will refer to this pattern as $j$’s component day, a term that is also used in the related literature on SVD to characterize time series (see Li (2000) and Kanjilal and Palit (1995) for applications in other domains).

Let $\beta_{jkt}$ be the underlying diurnal component day pattern in workload time series for a resource $k$ of a VM $j$ over $T$ time intervals $t$. Consider a representation of such a component day profile
in $\tau \subset T$ discrete time intervals (e.g., 288 five-minute-intervals over a day), while one observes
$\epsilon_{jkt} = \beta_{jkt} + \epsilon_{jkt}$, where $\epsilon_{jkt}$ is the residual workload behavior not explained by the component
day, such as the influence of unobserved variables or white noise. Observations from $D$ days are
assumed to be available, with $T = D \cdot \tau$. The corresponding $D \times \tau$ matrix $S$ of a VM $j$’s demand
for resource $k$ is formed by partitioning the time series into periods of length $\tau$ and rewriting each
period as a new row vector in a matrix $S$.

SVD can now be used to extract a component day as the typical daily pattern of an application.
After decomposing $S$ into the matrices $W\Lambda Y^T$ using SVD, the component day profile for a resource
$k$ is represented by the first right singular vector $y_1$ multiplied by its associated singular value $\lambda_1$.
The scale (amplitude) of this pattern over different days is described by the elements in $w_1$, the
first column vector in $W$.

If $\epsilon_{jkt}$ was 0 for a resource of type $k \forall t \leq \tau$, i.e., if there was no noise around the daily periodicity
described by the component day, $\lambda_2$ and subsequent singular values will be zero. The relative
Ky Fan 1-norm $\lambda_1 / \sum_{e=1}^{\text{rank}(S)} \lambda_e$ would be 1 and the component day profile could be characterized
perfectly with only one coordinate along $y_1$. With positive $\epsilon_{jkt}$ of $S$ $\lambda_1 / \sum_{e=1}^{\text{rank}(S)} \lambda_e$ will decrease,
and the workload approximation using the first dimension degrades. Hence, the proportion of $\lambda_1$
indicates the significance of the profile, which needs to be taken into account to consider for the
uncertainty in the component day profile. Figure 10 shows the extracted component day profile of
the workload example in figure 9. As in Figure 9, CPU demand values are shown in percentage of
a server’s total CPU capacity.

The main deviation of the component day profile is captured by the second right singular value as
shown in Figure 11. The figure shows that most uncertainty or empirical variance in the workload of this application is found in time interval 19, i.e., around 7 p.m. (3%) and around noon (1%). For the remaining time intervals, the major deviation is below 0.5% of the workload. Across all the available 458 VMs workload time series, component days describe 85% of the workload on average. In order to avoid server overload, we have added $y_2$-profiles to the $y_1$-component day profile before the optimization. 18 of the 458 applications have been excluded from our evaluation, as their component day profiles describe less than 75% of workload behavior. In other words, their workload traces exhibit sporadic demand and no regular pattern.

### 4.2. Research Design

We will mainly focus on server count as a metric for solution quality of the consolidation. This means, we compare the number of servers required to host a set of VMs based on different approaches to assign VMs to servers. We will explore the impact of different covariates such as 8 different set sizes, different numbers of resources to consider, different server capacities, and finally, different resource allocation models on server count. Table 1 provides an overview of the treatment variables used in our analysis.

<table>
<thead>
<tr>
<th>Treatment</th>
<th>VM Set Size</th>
<th>No. of Resources</th>
<th>Server Capacity</th>
<th>Consolidation Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td>[10, 400]</td>
<td>{1, 2, 3}</td>
<td>{5, 10, 15}</td>
<td>MBP(5min</td>
</tr>
<tr>
<td>No. of Values</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>9</td>
</tr>
</tbody>
</table>
models. For the purpose of statistical validation, we repeated each scenario six times with different VMs selected arbitrarily. We refer to one experiment as a run. In total, we conducted 3,888 runs.

Let us now explain the treatment variables in more detail. We use eight set sizes of workload traces with 10, 20, 40, 60, 80, 100, 200, and 400 VMs. We will consider servers with one, two, or three scarce resources such as CPU, main memory, and I/O capacity. The relationship of the resource demands of VMs to server capacities can play a role. If more servers are needed, more binary variables are used in the discrete optimization, and the problem instance takes longer to solve. Therefore, we explore scenarios, where 5, 10, and 15 VMs fit on a single server on average using a first fit allocation heuristic. We will refer to these as scenarios with small, medium-sized, and large servers, respectively. The “consolidation model” refers to the optimization formulations MBP (6) and SVPack (7) used to solve the server consolidation problem. MBP is solved with 288 five-minute, 24 hourly, two half-day, or only a single time interval for the day. The five-minute time intervals are aggregated to hourly or daily intervals by taking the maximum demand in these larger intervals, in order to ensure sufficient resources at all times. SVPack is solved with features extracted from the first three, four, five, and six subspaces of $R$’s SVD.

We have also implemented a simple multidimensional first fit heuristic (FF), which stacks one VM on top of the other until the capacity in one dimension is violated. Such heuristics do not consider side constraints and are therefore not applicable in most practical settings, but the results help us evaluate the solution quality of branch-and-cut algorithms in our experiments, which could not be solved to optimality.

All calculations are performed on a 2.4 Ghz Intel Duo with 4GB RAM using R for SVD calculation and IBM ILOG CPLEX 12.2 as the branch-and-cut solver. MBP with five-minute time intervals MBP(5min) could serve as a meaningful lower bound for all other allocations, as this formulation uses all available data. However, even for MBP(5min) with only 40 or 60 VMs and two or three resources the computation time for many problem instances exceeds days. For larger problems with a few hundred VMs we could not even find a feasible solution that improves a FF
solution with MDP(5min) because of the huge main memory requirements to store intermediate node-sets, which finally caused the solver to abort processing.

We will therefore use the objective function value of the LP relaxation of the MBP(5min) formulation (6) as a lower bound. In other words, we will assume that we can assign also parts of a VM on different physical servers. This allocation is technically not feasible, but easy to calculate and serves as a convenient lower bound for any feasible allocation considering that an entire VM needs to be hosted on a physical server.

Consultants typically want to analyze consolidation problems with different side constraints and different selections of VMs when working with a customer. As a consequence, there are limits on how long a user is willing to wait for a solution, and it is important to understand which problem sizes can be solved in acceptable time frames. MIP solvers allow the termination of the processing after a certain time limit and return the best integer solution found so far. The difference between the best LP relaxation in the branch-and-bound tree and the best integer solution found can serve as a worst-case estimate for optimality of the allocation. We found that even for large problems with 200 or 400 VMs, typically a feasible solution found within 30 minutes could not or could only marginally be improved after a computation time of a day or even a week.

So, we have used a time limit of 30 minutes for each problem and used the best integer solution found. This was also considered an acceptable time limit by our industry partner. Note that the time to compute the SVD of $R$ never exceeded ten seconds, even for the largest instances.

4.3. Problem Sizes of Different Consolidation Problems

It is worthwhile taking a look at the size of the constraint matrix of the mixed integer program of MBP and SVD as more VMs and more resources are considered. In both MBP as formulated in (1) and SVPack as formulated in (7), the number of decision variables grow mainly with $J \cdot I + I$.

Figure 12 shows the average number of binary decision variables in different scenarios when using SVPack(3) and MBP(5min) as an example. From left to right, we have a decreasing number of VMs per host on average.
The number of constraints in MBP(5min) formulations grows with $288 \cdot J \cdot K$, while the constraints in SVPack($E$) only grows with $E \cdot J$. With MBP(5min), even in rather small consolidation scenarios with only 60 VMs and three resources the number of constraints exceeds 10,000 and 60,000 with 400 VMs to be consolidated. As we will see, the large numbers of constraints in MBP(5min) make it increasingly hard to solve. Besides its computational complexity, the allocated memory in scenarios with 200 VMs and three resources exceeded 80 Gigabyte main memory with MBP(5min). With four Gigabyte main memory available, the CPU was waiting more than 99% of the time for data swapped between main memory and harddisk before the solver finally aborted the computation.

Figure 13 describes the number of constraints in the MBP(5 min) problem formulation in scenarios with different numbers of resources, server sizes, and VM set sizes (on the vertical axis). It shows the strong increase in the number of constraints with an increasing VM set size, an increasing number of resources, and a decreasing server size. Although MBP(5min) should lead to the minimal server count, the branch-and-bound tree grows so large that often, we were not able to find even a feasible integer solution equal to the FF solution within 30 minutes.

In contrast, even in the largest scenario with 400 VMs and small host servers, the number of constraints in SVPack(3) does not exceed 1040, and in our experiments the solver could at least find an integer solution, which required fewer or equal servers than the solution derived with FF.
in all but four runs in 30 minutes. However, also in SVPack there is a trade-off between increasing the number of dominant subspaces to consider and the optimality of the solution achieved in time. A larger number allows to model the workload data in more detail, but it will also lead to a larger constraint matrix.

4.4. Results

Aggregated results of our experiments are shown in Figure 14. For each consolidation model, the first (left) two bars are associated with the primary (left) Y-axis of the diagram and show the average solution quality of the consolidation model and the standard deviation of this solution quality. Again, solution quality was measured in number of servers required in the allocation relative to the objective function value of the LP relaxation of MBP(5min). A solution quality of 110% would mean that on average the allocation found needed 10% physical servers more than what was determined by the LP relaxation.

For each model, the two bars on the right-hand side, which are associated with the secondary (right) Y axis, show the percentage of experimental runs where a model could be solved to optimality (Solved to Optimality) and the percentage of runs where at least an integer solution better
than the solution derived by FF was computed (\textit{FF Improvement}).

In experimental runs where a feasible integer solution which improved the FF solution was found with MBP(5min), it achieved the highest solution quality with 118\% on average, followed by the SVPack(3) - SVPack(6) models with an average solution quality of around 124\%. The MBP model with hourly time interval aggregation achieved 134\%, while 12 hour or 24 hour time aggregation led to an average solution quality of 197\% or 213\%, respectively.

SVPack(3), SVPack(4), MBP(d/2), and MBP(d) could find a feasible integer solutions better than FF’s solution in over 99\% of cases, SVPack(5), SVPack(6), and MBP(h) could find such a solution in 90\% of the runs, while MBP(5min) found a feasible integer solution which improved the FF solution in less than 70\% of runs. With none of the models we could prove the optimality of the result in more than 50\% of the cases. Note that these are aggregate results across all problem sizes. We compared the models pairwise in terms of solution quality and found all differences to be significant at a 1\% level using a Wilcoxon signed rank test, except the difference between the SVPack models, which was not significant. We will sometimes talk about the solution quality of SVPack, meaning SVPack(4) as our standard model, relying on the fact that there was no significant difference to other versions of SVPack. These statistics should provide an initial overview, but we will drill down and provide more details below.

\textbf{Result 1:} SVPack allows the solving of significantly larger problem sizes to optimality than MBP(5min) and MBP(h). Among those instances, which cannot be solved optimally, SVPack could at least improve an FF solution in a larger number of instances.
Table 2  Scalability with Different Problem Formulations

<table>
<thead>
<tr>
<th>Server Size</th>
<th>Solution</th>
<th>VM SET SIZE</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>80</td>
<td>100</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>Small</td>
<td>Optimal</td>
<td>18</td>
<td>12</td>
<td>18</td>
<td>16</td>
<td>1</td>
<td>2</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>MBP(5MIN)</td>
<td>Improvement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No Solution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Optim</td>
<td>18</td>
<td>18</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Improvement</td>
<td></td>
<td>6</td>
<td>18</td>
<td>15</td>
<td>9</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>No Solution</td>
<td></td>
<td>3</td>
<td>9</td>
<td>14</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>Optim</td>
<td>18</td>
<td>18</td>
<td>17</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>Improvement</td>
<td></td>
<td>1</td>
<td>11</td>
<td>16</td>
<td>8</td>
<td>11</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>No Solution</td>
<td></td>
<td>4</td>
<td>7</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>Optim</td>
<td>18</td>
<td>9</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MBP(h)</td>
<td>Improvement</td>
<td></td>
<td>9</td>
<td>15</td>
<td>18</td>
<td>18</td>
<td>13</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>MBP(h)</td>
<td>No Solution</td>
<td></td>
<td>5</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Optim</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>10</td>
<td>7</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Improvement</td>
<td></td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>11</td>
<td>17</td>
<td>18</td>
<td>9</td>
</tr>
<tr>
<td>Medium</td>
<td>No Solution</td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>Optim</td>
<td>18</td>
<td>18</td>
<td>16</td>
<td>9</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>Improvement</td>
<td></td>
<td>2</td>
<td>9</td>
<td>5</td>
<td>18</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>No Solution</td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>Optim</td>
<td>18</td>
<td>12</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MBP(d)</td>
<td>Improvement</td>
<td></td>
<td>6</td>
<td>17</td>
<td>16</td>
<td>18</td>
<td>17</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>MBP(d)</td>
<td>No Solution</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Optim</td>
<td>18</td>
<td>17</td>
<td>14</td>
<td>10</td>
<td>5</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Improvement</td>
<td></td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>13</td>
<td>12</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Medium</td>
<td>No Solution</td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>Optim</td>
<td>18</td>
<td>18</td>
<td>17</td>
<td>14</td>
<td>11</td>
<td>13</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>Improvement</td>
<td></td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>14</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>No Solution</td>
<td></td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>Optim</td>
<td>18</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVPACK(4)</td>
<td>Improvement</td>
<td></td>
<td>13</td>
<td>17</td>
<td>16</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>SVPACK(4)</td>
<td>No Solution</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Optim</td>
<td>18</td>
<td>18</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVPACK(4)</td>
<td>Improvement</td>
<td></td>
<td>15</td>
<td>10</td>
<td>15</td>
<td>16</td>
<td>18</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>SVPACK(4)</td>
<td>No Solution</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>Optim</td>
<td>18</td>
<td>18</td>
<td>15</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>Improvement</td>
<td></td>
<td>3</td>
<td>10</td>
<td>9</td>
<td>14</td>
<td>17</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Large</td>
<td>No Solution</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Support: Table 2 shows the number of experimental runs with different problem formulations, which could be solved to optimality (Optimal), where no optimal, but a feasible equal or better than the FF solution could be found (Feasible), and where no feasible solution could be found which was at least as good as an FF solution (No Solution) grouped by VM set size and server size, the two key drivers for the size of an underlying constraint matrix.

With MBP(5min), no scenario with more than 40 VMs could be solved to optimality. In scenarios with small host servers, we could not compute the optimal solution even in scenarios with only 20 VMs and three resources within 30 minutes in two-thirds of the runs. Furthermore, in such
scenarios with small servers and VM set sizes of 100 or more, no feasible integer solution equal or better than FF could be found at all with MBP(5min).

In MBP(h) problems with up to 100 VMs could be solved to optimality, and for problems with fewer than 200 servers at least feasible integer solutions which improved the FF solutions were found. In scenarios with small host servers, solutions are found for VM set sizes up to 100. With MBP(d) and SVPack(4) we found feasible solutions for all scenarios. Only in two runs with small servers and 400 VMs could MBP(d) not find a feasible solution. Also, in two out of six runs we could find no feasible integer solution with SVPack(4), in scenarios with 400 VMs and small or medium-sized servers.

We will now describe the size of the constraint matrix of an optimization problem in terms of the number of constraints times the number of decision variables. This helps us compare the results of different scenarios. For each experimental run, Figure 15 shows the result of the CPLEX 12.2 branch-and-cut solver and plots size on a logarithmic scale. Problems with a size of less than one million could be solved to optimality in most models. For problems sizes of up to 100 million we could find a feasible integer solution better than the FF solution. For larger problems, we could typically not even find such a feasible solution in 30 minutes across all scenarios.

As an illustration, a problem with 80 VMs to be consolidated on small-sized host servers (with 5 VMs/server on average), with two resource types under consideration, leads to the following
Table 3 Problem Size of a scenario with I:=80, K:=2, Small Host Servers

<table>
<thead>
<tr>
<th></th>
<th>MBP(5min)</th>
<th>MBP(h)</th>
<th>MBP(d)</th>
<th>SVPack(3)</th>
<th>SVPack(6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cols</td>
<td>9296</td>
<td>1296</td>
<td>112</td>
<td>208</td>
<td>352</td>
</tr>
<tr>
<td>Rows</td>
<td>1296</td>
<td>1296</td>
<td>1296</td>
<td>1344</td>
<td>1392</td>
</tr>
<tr>
<td>Size</td>
<td>1.2E+7</td>
<td>1.1E+6</td>
<td>1.5E5</td>
<td>2.8E+5</td>
<td>4.9E+5</td>
</tr>
</tbody>
</table>

Figure 16 Difference between SVD(4) and MBP(5min) Solution Quality with Increasing Dimensionality

Result 2: The solution quality of the exact MBP(5min) formulation declines with increasing problem size compared to SV Pack with a given time limit due to the increase in the problem formulation of MBP(5min).

Support: We ordered all scenarios where MBP(5min) found an optimal or feasible integer solution by the size of an MBP(5min) model formulation. The scatterplot in Figure 16 shows the ratio between the solution quality of SV Pack(4) and MBP(5min) on the y-axis, and the MBP(5min) problem size on the x-axis. Positive values indicate the percentage of servers which is saved by using MPB(5min) instead of SV Pack(4).

As MBP(5min) could be solved to optimality in scenarios with small VM sets of 10 or 20 VMs, and in scenarios with 40 VMs and large servers, MBP(5min) never required more host servers than any other model. Once the size of MBP(5min)’s constraint matrix exceeds one million, MBP(5min) could not be solved exactly and the solution quality decreased. The reason for the decreasing
solution quality of MBP(5min) is the increasing branch-and-bound tree, which leads to the fact that the best integer solution found after 30 minutes is typically much worse than the optimal solution. Note that with MBP(5min), even problems with 60 VMs, two resource, and mid-size host servers, or problems with 40 VMs, small servers and three resources led to a constraint matrix with more than 1.3 million entries. It is also important to note that we ignored the major portion of runs with larger problems where no feasible integer solution could be found with MBP(5min) in Figure 16 (these instances are marked red).

Even in small scenarios with less than one million entries in the constraint matrix of MBP(5min), SVPack achieved the same solution quality as MBP(5min) in the majority of runs. On average, in small scenarios with a constraint matrix of less than four million entries in MBP(5min) the solution quality of SVPack(4) was 7.8% worse. In scenarios with MBP(5min) problem sizes between four million and 15 million entries, there was no significant difference in the solution quality (Wilcoxon rank test, $p = 0.7292$). In scenarios exceeding 15 million entries, MBP(5min) only found a feasible solution in less than 40% of the runs.

Figure 17 - Figure 20 provide detailed experimental results of scenarios with medium-sized servers and two resources for different VM set sizes to provide further evidence for result 2.

The results in scenarios with 40 VMs in Figure 17 show that MBP(5min) achieved the best solution quality, followed by the SVPack models (while SVPack solution quality increased with the number of considered subspaces). MBP models with time-aggregation of workload information required more servers, increasing with the number of aggregated time intervals. With all models,
a lower server count was achieved compared to an FF heuristic solution.

Figure 18 displays the results in scenarios with 80 VMs. With an average solution quality of 126%, MBP(5min) led to a slightly higher server count than the SVPack-models, which had an average solution quality between 118% and 123%. In addition, in 17% of runs MBP(5min) could not find a feasible solution which improved the FF solution. SVPack(4) and SVPack(3) models resulted in the lowest average number of required servers.

In scenarios with 100 VMs as shown in Figure 19, SVPack (3) and SVPack(4) models led to the best solution quality of 121% and 118%. MBP(h) achieved a better average solution quality (134%) than MBP(5min), which achieved 140%. In addition MBP(5min) only found feasible integer solution which improved the FF heuristic in 56% of runs. Figure 20 shows aggregated results for scenarios with 400 VMs. In these scenarios, none of the consolidation models could find a feasible solution improving an FF solution in all scenarios, and none of the consolidation models could be solved to optimality in any experimental run. The best solution quality was obtained with
SVPack(3) and SVPack(4) models, with an average solution quality of 129-130%. Again, MBP(h) led to a lower server count than MBP(5min), which only derived solutions better than FF in 17% of runs (compared to 50% with MBP(h), 67% with SVPack(5), and 33% with SVPack(6). **Result 3:** The solution quality of SVPack(4) is significantly higher than that of MBP(h).

**Support:** So far, we mainly compared SVPack and MBP(5min). MBP(d) and MBP(d/2) allow for larger numbers of VMs in the server consolidation problem, but at the expense of solution quality. Although the overall solution quality of SVPack models outperformed MBP(h), one might wonder whether an aggregation of five-minute time intervals to hourly time intervals in MBP(h) provides a good solution quality in particular in scenarios where MBP(h) can be solved optimally.

Again, in Figure 21 we order all scenarios where MBP(h) and SVPack(4) derived feasible integer solutions by the size of MBP(h)’s constraint matrix and show the difference in server count to SVPack(4). Positive values describe the percentage of servers saved when using MBP(h) compared to SVPack(4). SVPack typically provides better solutions or the same solution quality as MBP(h).

We now restrict our observation to the 185 experimental runs, where MBP(h) could be solved to optimality. Note, that in 95% of runs where optimal solutions could be derived the constraint matrix of MBP(h) had less than 500,000 entries. This can be translated into scenarios with 80 VMs, two resources, and mid-size servers. It is worthwhile noting that MBP(h) still considers 24 times the number of resources workload dimensions per VM, while SVPack considers only three to six workload dimensions per VM. On average, SVPack(4) had 10.1% higher solution quality than MBP(h), which is due to the fact that the hourly aggregation is too conservative at some times,
even though MBP(h) was solved optimal.

Among all runs with feasible solutions in MBP(h), SVPack(4) led to 12.3% higher solution quality on average. In addition to the higher solution quality, SVPack has a lower number of dimensions (or constraints) as shown in Figure 14. Hence, larger problems exceeding 8 million entries in the constraint matrix could not be solved reliably and for most of them we could not even find a feasible integer solution improving the FF solution within 30 minutes. An example of such a problem size would consist of 200 VMs, small servers, and two resources. The results show that SVD serves as an excellent approach to aggregating workload data.

5. Discussion and Managerial Implications

Today’s data centers and IT departments still offer many IT services on dedicated physical servers. Server virtualization provides a technical possibility for server consolidation, which has become very popular in the recent years. Multiple virtual servers – including operating systems and applications – can be reliably hosted on a single physical server sharing the same resources. Server consolidation describes the process of combining the workloads of several different servers on a set of target servers, using virtualization technology.

Cost considerations, specifically energy savings, are among the key drivers for such projects. Typically, workloads are volatile and have peaks at different times of the day or week. A central
question for IT service managers is, how virtual servers should be allocated to physical servers, in order to minimize the number of physical servers needed, given a certain quality-of-service level. The motivation for this is the reduction of investment costs, and, more importantly, energy costs in the data center. Server consolidation is seen as a central approach to saving energy in the data center (King 2007). Server consolidation has been one of the top priorities of CIOs in recent years. According to the National Association of State Chief Information Officers (NASCIO) the State CIO's Number 1 Policy Priority was "Consolidation" the Number 1 Technology Priority for 2009 was "Virtualization" (http://www.nascio.org/publications/).

IT service managers typically rely on manual planning and there is little tool support and no established methods for the planning of large server consolidation and capacity planning projects involving hundreds or even thousands of servers despite this being a regular occurrence, for example when a new data center is set up or a set of servers is migrated to a new type of server. Manual server consolidation is not only error prone, it is also a very time-consuming process, and it is almost impossible to consider multiple time dimensions, multiple resources (RAM, CPU), and technical allocation constraints without tool support. Existing work based on combinatorial optimization suffers from the large number of time dimensions, which need to be considered to characterize workloads appropriately.

In this paper, we have presented an approach based on singular value decomposition and integer programming to solve large, high-dimensional server consolidation problems. We propose SVD as a way to extract the relevant features from the large volumes of workload data available in corporate data centers. Such volumes of data would typically render the server consolidation problem intractable for all but small instances. To our knowledge, this is the first approach to use singular value decomposition in IT service or workload management and, above all, is the first paper to use the extracted features of the data to solve resource allocation problems.

The experimental evaluation based on real-world workload data shows that large instances with hundreds of servers and multiple resources of the server consolidation problem can be solved in 30 minutes. The approach does not require any additional data preprocessing or any additional
parameters and is purely based on the workload data, which is a significant advantage in any practical application. Even for smaller instances, where the server consolidation problem could be solved on the original data, the solution quality in terms of number of servers required, was not significantly worse.

Whether the approach is suitable or not depends on the regularities in the workload data. If the workload exhibits daily or weekly cycles and predictability, as is the case for many enterprise applications, then the workload can be described well with a few dominant SVD subspaces, and the approximation error of a reconstructed workload matrix based on these few subspaces is very low. This was actually the case with the workload data used in our experimental analysis, which is based on server consolidation projects from a European IT service provider. It is important for IT managers to understand if the workload data for a new server consolidation project satisfies the necessary requirements. As we have shown, this can easily be evaluated by an analysis of the singular value spectrum and the decay in the singular values. The Ky Fan norm provides a simple scalar description of the accuracy of a matrix approximation using a certain number of singular values, and it can serve as a metric, which is easy to interpret.

In general, the approach is applicable to multi-dimensional packing problems beyond server consolidation, where items to be packed are high-dimensional vectors and bins are high-dimensional objects. Unlike traditional problems in geometric combinatorics such as cutting stock, trim loss, or container loading problems, which are concerned with objects defined by one, two or three spatial dimensions of the Euclidean space (Dyckhoff 1990), we consider packing and covering problems where the sum of the high-dimensional vectors placed in a bin or knapsack must not exceed the capacity in any dimensions.

Multidimensional knapsack problems arise also in other areas of IT service management. An example is the allocation of business applications to physical servers, when a part of the server park has different quality attributes such as higher bandwidth connection, or enhanced security options. Here, the objective is to maximize the total business value of selected applications on a subset of
servers, while ensuring that the aggregated workload estimates do not exceed the capacity limits of these servers (Chen et al. 2002).

Another example is just-in-time manufacturing, different parts are produced according to a weekly or monthly production plan on a set of machines. Such a plan may contain the forecasted volumes of parts to be produced on each day during a planning period. A machine with requires a time and cost intensive setup to produce certain parts. Consequently, managers have to decide, which parts to assign to which machine over a planning period in order to minimize setup costs (Denizel and Erengc 1997). Overall, the method is applicable, whenever many resources or many periods need to be considered and demand estimates are available for these dimensions, which exhibit some regularity.

References


Seltzam, S., D. Gnach, S. Krompass, A. Kemper. 2006. An automatic administration concept for service-oriented database applications. *Int. Conference on Data Engineering (ICDE 2006)*. Atlanta, USA.