Variable Cost and Size Cluster Vector Bin Packing - A Model and Heuristics for Cloud Capacity Planning

Laura Wolf[1], Sabrina Klos[2], and Stefan Nickel[1]

[1] Karlsruhe Institute of Technology, Institute for Operations Research, 76131 Karlsruhe, Germany
{laura.wolf, stefan.nickel}@kit.edu
[2] SAP SE, 69190 Walldorf, Germany

Abstract. Vector bin packing is a problem in combinatorial optimization that is particularly relevant in the area of cloud computing but also finds application in various areas of logistics. The problem deals with how to optimally place items into bins with constraints on multiple separate resource dimensions. We extend the problem to a cluster structure of bins, including variable bin sizes and cluster costs. The proposed extension of vector bin packing, which we term VCSCVBP, allows us to model a practical problem in the area of cloud computing, namely, the cloud capacity planning problem, where servers are organized in clusters. Optimizing data center capacity in terms of costs and fulfillment of customer demands in the form of virtual machines has become crucial due to the increasing demand for computing resources. We introduce several novel heuristics, called CS-P heuristics, consisting of a packing and a cluster selection step. The algorithms are evaluated with a benchmark based on practically relevant cloud computing data. Substantial runtime improvements are demonstrated by the computational experiments. For two out of three considered cost scenarios, only a slight deviation of the objective value obtained by the CS-P heuristics from the objective value obtained by the solver is observed. By exploiting cluster information and discarding certain cluster types through an additional procedure, this is also achieved for the third cost scenario.

Keywords: Real-World Vector Bin Packing · Capacity Planning · Cloud Computing

1 Introduction

In this paper, we extend the vector bin packing (VBP) problem to a cluster structure of bins and consider cluster costs and variable bin sizes. Each cluster consists of a set of bins and is associated with different costs. If at least one bin of the cluster is used, the overall cluster costs are taken into account. This problem applies to the area of cloud capacity planning for data centers, where virtual machines (VMs) need to be placed on hosts, which are organized in clusters.

Data center optimization has gained considerable relevance in recent years due to the increasing demand for computing resources. As McKinsey & Company [4] forecasts, the global spending on data center construction will be 49$ billion in 2030, corresponding to a compound annual growth rate of 5.4%. Moreover, since cost minimization is considered the main challenge in cloud computing, according to the Flexera 2024 State of the Cloud Report [9], planning and using cloud resources in the best possible way is of utmost importance.

Therefore, we formulate a novel model for cost minimization in tactical capacity planning in data centers. The model expecially allows for the definition of clusters of hardware, which is a typical way of organizing servers in data centers. The proposed model is based on a VBP formulation, which we extend by a cluster structure. In the following, we give an overview of the related work on VBP and cloud capacity planning.

First, we discuss the related literature in the field of VBP. In d-dimensional VBP, a finite set J of items is considered, where each item j has a defined size w_{jk} in dimension k with $1 \leq k \leq d$. The items need to be packed into a bin i of the set I of bins without exceeding the bin capacities W_{ik}. The item sizes in the different dimensions can be seen as d-dimensional vectors to be packed into the d-dimensional bin size vectors. In the literature, the VBP problem is also known as the multi-dimensional bin packing problem [14]. VBP is a generalization of bin packing (BP), where items are packed into one-dimensional bins. Due to its NP-hardness, several publications explore the development of heuristics for VBP. A comprehensive overview of the different algorithms for VBP can be found in Mommessin et al. [17]. The different approaches are divided into item-centric, bin-centric, and multi-bin approaches. In addition, Nagel et al. [18] provide an extensive analysis of different VBP heuristics. Various further work on the topic of different problems in VBP exist. Research on BP and VBP, which focuses on minimizing costs or considering variable bin sizes, is particularly relevant to cloud capacity planning. Examples of such research include the variable cost and size bin packing problem (VCSBPP) [6], vector bin packing with heterogeneous bins (VBPHB) [10], and the two-dimensional vector packing problem with piecewise linear cost function (2DVPP-PLC) [12]. With respect to clustering, existing studies address problems categorized as bin packing with clustering, but in this context clustering refers to assigning items to a cluster such that only items of the same cluster are packed together [3]. In contrast, our approach refers to clusters as a grouping of bins into higher-level structures and additionally includes costs associated with these clusters. This is where the novelty of our approach lies. Beyond its application in cloud capacity planning, the introduced problem is also relevant to the field of logistics, particularly for packing with predefined compartments. In this paper, we select and adapt a subset of the algorithms analyzed in [17] and [18] as a subroutine to the proposed heuristics for the novel model formulation of VBP with bin clusters and costs. Section 3 gives a detailed overview and justification of the selected algorithms.

Next, we continue with an overview of the related work on cloud capacity planning. Cloud computing offers access to on-demand computing resources [16]. A

customer's application is typically run in a VM that consumes resources in different dimensions (memory, CPU, storage, and others) of the underlying physical hardware (host/server) on which it is placed. In order to fulfill the customer demand and not over-provide computational resources associated with high costs, cloud providers have to estimate the required resources in their data centers. Capacity management describes the cloud provider's management process associated with the IT infrastructure and services [15]. Capacity planning, a subprocess of capacity management, focuses on defining the capacity of the required resources. Due to supply chain cycles of five months to one and a half years, the data center providers have to decide which hardware to use regularly [2]. We focus on the tactical task of cloud capacity planning from a cloud provider's perspective with a mid-term planning horizon and cost minimization as the objective. As tactical planning, we consider mid-term planning with a planning scope of 6 to 24 months to meet demand in the most effective and profitable manner [21]. Different publications cover the topic of cloud capacity planning. The ones using mathematical programming mainly combine purchasing decisions with long-term planning, such as location planning [8,19], or with short-term decisions, such as scheduling [5]. Numerous publications deal with the operational problem of VM placement on the hosts of the existing data center servers, which is often modeled as VBP with different objectives, such as the minimization of active hosts or energy consumption [22]. Other methods besides mathematical programming include time series analysis [13] or simulation [2].

Since hosts in data centers are typically organized in physical clusters and racks, sharing technical components, it is crucial to consider a higher level structure than host level as the planning size on a tactical level. A cluster structure of hosts is already considered in Andreadis et al. [2], who also include a survey of practitioners in their approach and whose problem scope comes closest to this paper. In contrast to this paper, Andreadis et al. use discrete-event simulation instead of mathematical programming to deal with cloud capacity planning. To the best of our knowledge, no other publication in cloud capacity planning uses mathematical programming and a cluster structure of hosts as purchasing and operating size for planning. Therefore, we expand the existing VBP formulation by a cluster structure of bins, shown in the following Section 2. A preliminary version of the model formulated more specifically on cloud capacity planning and not on VBP in general was introduced by us in [23].

The contributions of our paper with respect to the current state of research are as follows:

- We introduce a mathematical problem formulation for variable cost and size cluster vector bin packing (VCSCVBP).
- We present the novel CS-P heuristics with a focus on clusters with homogeneous bins that enable reduced runtime.
- We evaluate the CS-P heuristics with computational experiments based on a benchmark data set generated with data from the field of cloud computing.

Therefore, Section 2 introduces the mathematical formulation and explains its application to cloud capacity planning. The proposed CS-P heuristics to

reduce runtime significantly for large instances are described in Section 3 and numerically evaluated in Section 4. Section 5 summarizes the findings of this paper and provides an outlook on further topics.

2 Problem Definition and Model

In this section, we present the variable cost and size cluster vector bin packing (VCSCVBP) problem. This formulation builds upon the formulations of the VBP and the BP problem with variable bin sizes and costs. For VCSCVBP, we combine these problems and add a cluster structure of bins with cluster costs.

We consider a finite set I of bins. Each bin $i \in I$ is specified by its sizes W_{ik} in the different dimensions k and has a maximum fill level f_{ik}. Moreover, we have a finite set G of clusters. Each cluster $g \in G$ consists of a fixed number and configuration of bins I_g^G and has a fixed individual cost parameter c_g. Every bin belongs to a cluster. In addition, we consider a finite set J of items specified by their sizes w_{jk} in the d different dimensions. All sets and parameters are listed in Table 1, together with the decision variables. In the considered problem, the goal is to place each item into a bin in such a way that the costs for the used clusters are minimized. Therefore, for each bin and cluster, the decision has to be made whether or not to use it and, if yes, which items to place.

Table 1: Sets, parameters, and decision variables of the VCSCVBP

Sets	
G	Clusters
I	Bins
I_g^G	Bins in cluster $g \in G$
J	Items
K	Resource dimensions
Parameters	
c_g	Costs of cluster $g \in G$
f_{ik}	Max. fill level of resource dimension $k \in K$ of bin $i \in I$
w_{jk}	Size of item $j \in J$ in resource dimension $k \in K$
W_{ik}	Capacity of bin $i \in I$ in resource dimension $k \in K$
Variables	
$\mathrm{x}_g = \begin{cases} 1 & \text{If cluster } g \in G \text{ is chosen,} \\ 0 & \text{otherwise.} \end{cases}$	
$\mathrm{y}_i = \begin{cases} 1 & \text{If bin } i \in I \text{ is chosen,} \\ 0 & \text{otherwise.} \end{cases}$	
$\mathrm{z}_{ij} = \begin{cases} 1 & \text{If item } j \in J \text{ is placed into bin } i \in I, \\ 0 & \text{otherwise.} \end{cases}$	

VCSCVBP:

$$\min \quad Z = \sum_{g \in G} c_g x_g \tag{1}$$

$$\text{s.t.} \quad x_g \leq \sum_{i \in I_g^G} y_i \qquad g \in G \tag{2}$$

$$x_g \geq y_i \qquad g \in G, i \in I_g^G \tag{3}$$

$$\sum_{i \in I} z_{ij} = 1 \qquad j \in J \tag{4}$$

$$\sum_{j \in J} w_{jk} z_{ij} \leq f_{ik} W_{ik} y_i \qquad i \in I, k \in K \tag{5}$$

$$x_g, y_i, z_{ij} \in \{0, 1\} \qquad g \in G, i \in I, j \in J \tag{6}$$

The VCSCVBP problem can now be stated as follows: The objective (1) is to minimize the overall costs composed of the sum of individual cluster costs. Constraints (2) and (3) state that if one bin of a cluster is used, the overall cluster is considered. Constraints (4) ensure that each item must be placed in exactly one of the bins. Constraints (5) guarantee that the capacity of the bins is not exceeded in any dimension $k \in K$, taking into account a maximum fill level prescribed by f_{ik}. The cluster structure distinguishes the proposed model from the related work on VBP presented in Section 1.

Next, we show how the VCSCVBP problem can serve as a model for cloud capacity planning. Each bin represents a physical host in a data center. The hosts are grouped into clusters, racks or any other grouping of hosts that share the same infrastructure. VMs are used to place customer jobs on physical hosts. Hence, each VM can be seen as an item in the packing problem. The cost parameters c_g are used to model the individual costs of each cluster of hosts, which can consist of multiple cost components and typically vary depending on the specific underlying host configurations. For example, it might be much lower for already existing clusters than for new ones. The parameter f_{ik} serves to keep a safety buffer for each host or to allow over-provisioning in the planning. Note that we choose a BP formulation instead of a cutting stock formulation in order to be able to take placement restrictions of individual VMs into account. Such placement restrictions typically occur in terms of VM affinities to specific hosts or anti-affinities among VMs. The proposed model formulation is NP-hard. As high solver runtimes pose a challenge in the practical application, we propose heuristics for the VCSCVBP in Section 3.

3 Vector Bin Packing Heuristics for VCSCVBP

In this section, we present novel heuristics for solving the introduced mathematical formulation of the VCSCVBP problem. We differentiate two different cluster structures concerning the contained bin types: Clusters with homogeneous bins

and clusters with heterogeneous bins. A cluster with homogeneous bins consists of a set of bins of the same type, i.e., all contained bins share the same technical configuration. However, different cluster types can contain different bin types. A cluster with heterogeneous bins may contain bins of different types. In this paper, we present heuristics that can be applied to both clusters with homogeneous and heterogeneous bins. However, the presented heuristics are more tailored towards scenarios consisting of multiple types of clusters of homogeneous bins, i.e., where each cluster type contains only bins of the same type, but different cluster types can contain different bin types. For clusters consisting of heterogeneous bins with significantly different resource dimensions, alternate heuristics might be more advantageous, or the presented heuristic could create an initial solution to be improved with the help of further (meta)heuristics. Heuristics specifically tailored towards clusters with a heterogeneous structure of bins will be the subject of future work.

The presented heuristics consist of two parts, namely, (i) one of the several vector bin packing algorithms and (ii) a cluster selection algorithm. Therefore, the name of each introduced heuristic CS-P is made up of the general cluster selection CS and the respective packing method P adjusted to the cluster structure, for example, CS-FFD for the cluster selection in combination with a First Fit Decreasing algorithm. The two parts of the heuristics alternate. First, the cluster selection begins with selecting the first cluster to be used. This is followed by a loop of vector packing and cluster selection. Before we present the algorithms in detail in Sections 3.1 and 3.2, we first introduce the terminology used to define and update the status of bins and clusters. Therefore, we divide the available bins into the following categories: A bin is called *active* if it is currently considered in the packing algorithm. Moreover, a bin is called *closed* if it has already been packed and is not considered for packing anymore. In addition, a bin is called an *unused bin of a used cluster* if other bins in the same cluster are already *active* or *closed*. Finally, a bin is called a *bin of an unused cluster* if none of the bins in the cluster has been used yet. Opening a bin describes changing the status of an *unused bin in a used cluster* to an *active bin*. Clusters can be divided into two categories: A cluster is called a *used cluster* if one of the bins of this cluster is *active* or *closed*, and an *unused cluster* otherwise.

The packing algorithms presented in Section 3.1 take a given set of clusters as input or can work with a set of cluster types with an infinite set of clusters each. For the clusters to be used, the algorithms determine which items to place into which bin of which available cluster. This includes the decisions on when to open a new bin from the *unused bins of used clusters* and when to close an *active* bin. Only if a new *active bin* is needed and there are no *unused bins of used clusters*, the cluster selection process presented in Section 3.2 determines which additional cluster to use next. The bins of the selected cluster then change their status from *bins of an unused cluster* to *unused bins of a used cluster*. Next, the packing algorithm is called with the additional bins of the newly *used cluster* as input. Initially, we start with a cluster selection phase because no bins are in a *used cluster*. We set the status of the first chosen cluster to *used* and the

contained bins to *unused bins in a used cluster*. Then, the first packing process starts, and we continue the packing and cluster selection loop.

3.1 Vector Bin Packing Algorithms

We apply different state-of-the-art heuristics for the packing procedure, which we describe in the following, including all adjustments that we introduce due to the cluster structure of the bins. For the VBP algorithms, we orient ourselves mainly on the analysis of the heuristics of Nagel et al. [18] and complement it with the evaluation of Mommessin et al. [17]. Table 2 gives an overview of the heuristics used and adjusted. Below, we describe each in detail. Different methods to define weights to balance the different resource dimensions against each other can be used.

Table 2: Overview of the used VBP algorithms for the CS-P heuristics

Algorithm Class	Algorithm	Norm
Item-centric	First Fit Decreasing (FFD) [14]	L_2
Bin-centric	Norm-based Greedy (NBG) [20]	L_2
	Dot Product (DP) [20]	Dot product
	Local Search (LS) [18]	L_2
Multi-bin activation	Hybrid Heuristic (HyL_2, HyDP) [18]	L_2, dot product

For item-centric approaches, as Mommessin et al. [17] state, Best Fit Decreasing (BFD) and Worst Fit Decreasing (WFD) algorithms do not significantly outperform First Fit Decreasing (FFD) algorithms. We, therefore, use FFD introduced by Kou and Markowsky [14]. Items are sorted in a decreasing order and assigned to the bins using the first fit. Different norms, such as L_1, L_2, and L_∞, and the weights define the combined size v to sort the items. We iteratively add a new bin each time the item to be placed does not fit in the *active bins*.

Three different bin-centric approaches are evaluated for the VCSCVBP problem: The Norm-based Greedy (NBG) and the Dot Product (DP) by Panigraphy et al. [20] as well as the Local Search (LS) by Nagel et al. [18]. Only one bin is *active* at a time. The algorithms by Panigraphy et al. [20] follow the same procedure: In each iteration, the algorithm selects one item to be placed into the *active bin*. Panigraphy et al. [20] introduce different methods to define this item. NBG chooses the item that reduces the difference between the item size and the remaining capacity of the *active bin* the most, considering the L_2-norm. DP selects the item with the largest dot product of the remaining capacity in the *active bin* and the item size. If no unplaced item fits into the *active bin*, the bin is closed, and a new one is opened. In contrast, the Local Search (LS) algorithm follows a different procedure. First, LS [18] opens a bin, in which the item with the largest combined size v fits, places it into this *active bin*, assigns random items to fill the bin, and tries afterward to exchange placed items to reduce the remaining space of the *active bin* concerning the L_2-norm.

The third algorithm class considered is multi-bin activation. We apply the Hybrid Heuristic (Hy) proposed by Nagel et al. [18] to the VCSCVBP problem.

Here, several bins are *active* simultaneously. Then, the best bin and the best item to be placed are defined considering the remaining bin capacities of *active bins* and the sizes of the unplaced items. A new bin is opened if one of the non-assigned items cannot be placed into any of the *active bins*. To adapt the Hybrid Heuristic to the considered cluster structure of bins, we use an incremental bin activation strategy and add only one new bin at a time from the *unused bins of used clusters* to the *active bins*. As suggested by Nagel et al. [18], we use the L_2-norm (HyL_2) and the dot product (HyDP). The cluster selection algorithm complementing the packing algorithm is presented in Section 3.2.

3.2 Cluster Selection Algorithm

This section presents the cluster selection process. When a new *active bin* is needed to continue the packing algorithm, and there are still *unused bins of used clusters*, one of these is set to *active* and filled. However, if there are no *unused bins of used clusters* or the item to be placed does not fit into the bins of the *used clusters*, then the cluster selection algorithm starts. Besides the *used clusters*, the *unused clusters*, and the bins in the categories *active bins*, *closed bins*, and *bins of unused clusters*, the algorithm takes an item as input. This item depends on the packing algorithm.

In the following, the cluster selection algorithm is described, as displayed in Algorithm 1. First, all *unused clusters* are sorted in ascending order by a newly introduced cost/combined size ratio (c/v-ratio). To define the combined size v, we use the same norm as the one used by the respective vector packing algorithm, and we have the option to include weights. If the dot product is chosen, we apply the L_2-norm. Unit cost to volume ratios are also used in heuristics for the VCSBPP [6]. For each cluster, the combined size of each of its single bins is determined first and then summed up to define the cluster's combined size v. For clusters with the same c/v-ratio, the cluster with lower total costs c is prioritized. Next, it is checked into which bin of the clusters in the sorted list the item fits first. This cluster is set to *active*. The item considered depends on the packing algorithm. For the item-centric algorithm, we hand over only the current item we want to place to the cluster selection algorithm. For the bin-centric algorithms, we use the largest combined size unplaced item, and for the Hybrid Heuristic, we hand over the largest combined size item that was not placeable into the *active bins*. In the case of clusters with heterogeneous bins, the bins are sorted in a descending order considering v to fill larger bins first.

After all items are placed, we perform an additional post-processing step. We want to prevent having a last *used cluster* with a low c/v-ratio ratio but with high total costs, which is only partially filled. Therefore, we adjust the post-processing procedure mentioned by Crainic et al. [6] for the VCSCVBP problem to a cluster structure of bins. Suppose the items placed into the last cluster can be packed into another cluster with lower total costs, even though the c/v-ratio is higher than for the primarily chosen cluster. In that case, we repack the items to reduce the overall costs. In Section 4, we conduct computational experiments to evaluate the different heuristics.

Algorithm 1 Cluster Selection

```
sort clusters_unused by c/v-ratio in ascending order
for cluster in clusters_unused do
    if item fits into any bin of cluster then
        add cluster to clusters_used
        sort bins of cluster by v in descending order
        append bins of cluster to unused_bins_of_used_clusters
        delete cluster from clusters_unused
        break
    end if
end for
```

4 Computational Experiments

This section delineates the computational experiments to evaluate the introduced heuristics. First, we describe the data set and the test environment used in the experimental setup. Second, we investigate the results.

4.1 Experimental Setup

We create a data set based on publicly available data from cloud capacity planning and consider a two-dimensional problem ($d = 2$) with memory and CPU as resource dimensions. Nevertheless, the introduced algorithms are all designed for an arbitrary number of dimensions. As stated in Section 3, the CS-P heuristics focus on a problem setting with clusters of homogeneous bins. We consider four different cluster types. Due to the widespread use of Dell servers [11], we use these for defining the host sizes [7]. The numbers of hosts per cluster are generated arbitrarily. The cluster configurations are shown in Table 3. Note that we consider three different cost scenarios: In cost scenario (a), the c/v-ratio increases, and in cost scenario (b), the c/v-ratio decreases with increasing host size. In cost scenario (c), the c/v-ratio increases more than in cost scenario (a). Table 4 delineates the VM sizes. We use AWS EC2 instance sizes [1] and select VM types with different memory-to-CPU ratios as they are typical for different workloads: The VM types V1.x correspond to VMs for general purpose, whereas V2.x are compute-optimized, and V3.x are memory-optimized VMs.

Table 5 presents the instance classes we use for the numerical evaluation. For every instance class, we create ten different instances, where for each instance, the actual VM demands per VM size are sampled randomly. The demand number displayed in the table serves as the mean for a normal distribution with a standard deviation of 10% of the mean. The different VM sizes correspond to the VM sizes displayed in Table 4. For example, in Class A1, the demand of Vx.1 means that we independently sample for V1.1, V2.1, and V3.1 with a mean of $\mu = 5$. As stated in [2], current practice in industrial applications is to overprovision CPU resources while avoiding overprovisioning memory. Therefore, we

Table 3: Cluster and host configurations of the cloud benchmark

Cluster type	C1	C2	C3	C4
Logical cores per host	96	112	128	192
RAM [GB] per host	512	768	1024	2048
Hosts per cluster	15	15	10	10
c/v-ratio in cost scenario (a)	0.7	0.8	0.9	1
c/v-ratio in cost scenario (b)	1.3	1.2	1.1	1
c/v-ratio in cost scenario (c)	0.2	0.4	0.6	1

Table 4: VM configurations of the cloud benchmark

General-purpose VM	V1.1	V1.2	V1.3	V1.4	V1.5	V1.6	V1.7
vCPU	8	16	32	48	64	96	192
RAM [GiB]	32	64	128	192	256	384	768
Compute-optimized VM	V2.1	V2.2	V2.3	V2.4	V2.5	V2.6	V2.7
vCPU	8	16	32	48	64	96	192
RAM [GiB]	16	32	64	96	128	192	384
Memory-optimized VM	V3.1	V3.2	V3.3	V3.4	V3.5	V3.6	V3.7
vCPU	8	16	32	48	64	96	192
RAM [GiB]	64	128	256	384	512	768	1536

arbitrarily select $f_{CPU} = 2$ and $f_{memory} = 0.9$ as the maximum fill levels for all hosts. For a given instance class and cluster type, we determine the number of clusters needed if all items are placed only on this cluster type as an upper bound for the solver. With nine different instance classes, three cost scenarios, and a sample size of ten, we include 270 instances in our evaluation.

Table 5: Problem instances of the cloud benchmark

Class	No. of clusters				No. of demanded VMs						
	C1	C2	C3	C4	Vx.1	Vx.2	Vx.3	Vx.4	Vx.5	Vx.6	Vx.7
A1	10	7	7	5	5	5	5	5	5	5	5
A2	10	7	7	5	10	10	5	5	5	0	0
A3	10	7	7	5	0	0	5	5	5	10	10
A4	20	15	15	10	10	10	10	10	10	10	10
A5	20	15	15	10	20	20	10	10	10	0	0
A6	20	15	15	10	0	0	10	10	10	20	20
A7	30	25	25	15	20	20	20	20	20	20	20
A8	30	25	25	15	30	30	20	20	20	10	10
A9	30	25	25	15	10	10	20	20	20	30	30

The model is implemented in Python 3.10 and solved with Gurobi. All computations are conducted with a computational time limit of 900 seconds. The computational experiments run on an Intel Core i5 processor with 2.60 GHz and 16 GB of RAM running Windows 11.

4.2 Computational Results

This section presents the computational results considering the introduced cloud data set. First, we investigate the following three aspects: (i) runtimes of the CS-P heuristics, (ii) solution quality of the CS-P heuristics, and (iii) comparison of the best solution found by the CS-P heuristics and by the solver. Note that the displayed results are obtained using static reciprocal average weights to prioritize among dimensions. They are defined as the inverse of static average weights, which are based on the average resource demand in this dimension [10, 17].

Figure 1 displays the runtime of the different CS-P heuristics. Since the results of the two bin-centric approaches, CS-NBG and CS-DP, are similar in terms of runtime, we only show one of the two for clarity. The same applies to the two Hybrid Heuristics. For all instance sizes, the item-centric heuristic CS-FFD outperforms the bin-centric approaches CS-NBG, CS-DP, and CS-LS in terms of runtime. The multi-bin activation heuristics CS-HyL_2 and CS-HyDP have the highest runtimes.

Fig. 1: Runtime comparison of the CS-P heuristics on the cloud benchmark

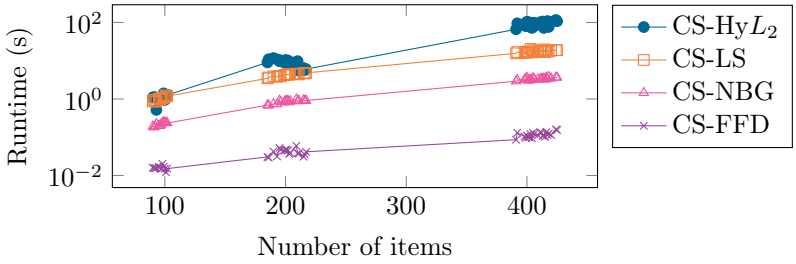

Table 6 compares the different CS-P heuristics in terms of the best solution found. It shows the percentage difference between the average minimal objective value obtained over all heuristics' solutions and the average objective value of the currently considered heuristic. CS-LS outperforms the other heuristics on average for most of the instance classes. This confirms the findings from Nagel et al. [18], who show that LS performs as one of the best among the heuristics shown for VBP. Nevertheless, CS-P heuristics other than CS-LS sometimes achieve better results for an individual instance of the corresponding class. In addition, particularly in small instance classes, the other CS-P heuristics often perform as well as CS-LS on average. This results from the cluster structure of bins and the fact that additional costs only occur for a new cluster and not for single bins.

Next, we compare the best solution obtained by the CS-P heuristics with those obtained by the solver, as displayed in Table 7 and Table 8. Note that we limit the comparison to the instance classes A1-A6. There exists a widening MIP gap already for the instance classes A4-A6; hence, the informative value of larger instances is reduced. Due to the different performances of the CS-P heuristics depending on the instance, we use a combined approach. For each instance, we calculate the objective value with each CS-P heuristic individually, more pre-

Table 6: Performance comparison of the CS-P heuristics in % compared to the best found solution with a CS-P heuristic on the cloud benchmark

Class	CS-						Class	CS-					
	FFD	NBG	DP	LS	Hy L_2	Hy DP		FFD	NBG	DP	LS	Hy L_2	Hy DP
A1_a	**0**	**0**	**0**	2.9	3.9	3.9	A5_c	**0**	**0**	**0**	**0**	**0**	**0**
A1_b	**0**	**0**	**0**	**0**	**0**	**0**	A6_a	1.0	1.0	1.0	**0**	1.0	1.0
A1_c	7.0	7.0	7.0	**0**	1.9	1.9	A6_b	2.9	2.9	2.9	**0**	2.9	2.9
A2_a	**0**	**0**	**0**	**0**	**0**	**0**	A6_c	3.8	3.8	3.8	**0**	3.8	3.8
A2_b	**0**	**0**	**0**	**0**	**0**	**0**	A7_a	**0**	**0**	**0**	0.7	0.4	1.6
A2_c	**0**	**0**	**0**	**0**	**0**	**0**	A7_b	8.8	8.8	8.8	**0**	8.8	8.8
A3_a	5.8	5.8	5.8	**0**	5.8	4.4	A7_c	5.0	5.0	5.0	**0**	4.1	4.6
A3_b	5.3	5.3	5.3	**0**	5.3	5.3	A8_a	**0**	**0**	**0**	0.3	0.5	1.3
A3_c	10.6	10.6	10.6	**0**	10.6	10.0	A8_b	**0**	**0**	**0**	**0**	**0**	**0**
A4_a	0.3	0.3	0.3	**0**	1.1	1.1	A8_c	4.0	4.0	4.0	**0**	3.9	4.3
A4_b	7.3	7.3	7.3	**0**	7.3	7.3	A9_a	0.6	0.6	0.6	**0**	0.9	0.5
A4_c	5.8	5.8	5.8	**0**	5.6	5.6	A9_b	8.5	8.5	8.5	**0**	8.5	8.5
A5_a	**0**	**0**	**0**	**0**	**0**	**0**	A9_c	3.2	3.2	3.2	**0**	2.6	2.4
A5_b	33.3	33.3	33.3	**0**	33.3	33.3							

cisely, with all six introduced heuristics. We then use the minimal objective value obtained over all heuristics' solutions and compare it as a percentage difference with the best solution found by the solver. In this case, the runtime of the combined CS-P heuristic is the sum of the runtimes of all six different CS-P heuristics applied to the problem instance. To improve the runtime in the future, it is also possible to combine only a selection of the six heuristics. Tables 7 and 8 display ΔZ and Δt as performance measures. ΔZ outlines the average deviation from the best solution found by the solver. For every instance, the difference between the objective value of the combined CS-P heuristic and the one achieved with the solver, in relation to the latter, is calculated. Subsequently, the value is averaged across all ten instances of the instance class. Δt is the average difference in the runtimes of the solver and the combined CS-P heuristic. In a similar fashion, Δt is calculated for each instance and averaged across the instance class. Significant performance differences in the heuristics concerning the different cost structures are evident. As Table 7 shows, the average ΔZ for cost structures (b) and (c) is 3.1% and 3.7%, respectively. Hence, the combined CS-P heuristic performs very well for these cost structures.

However, for cost structure (a), the average ΔZ is 15.4%, resulting from the second column of Table 8. The significant difference for cost structure (a) results from the fact that clusters with larger bins only have a slightly larger, i.e., worse c/v-ratio than clusters with smaller bins. However, larger bins enable more packing possibilities. This may outweigh the slightly higher costs, which are not considered so far in the CS-P heuristics. Based on this observation, we expand our approach by introducing additional iterations of the combined CS-P heuristic. In each iteration, an additional cluster type is excluded from consideration and, thus, cannot be used: In the second iteration, only cluster types C2-C4; in the

Table 7: Comparison of the combined CS-P heuristic with the solver on instances of the cloud benchmark for cost scenario (b) and (c)

Class	ΔZ (%)	Δt (s)	MIP gap (%)	Terminated	Class	ΔZ (%)	Δt (s)	MIP gap (%)	Terminated
A1_b	0.0	1.9	0.0	10/10	A1_c	8.0	29.1	0.0	10/10
A2_b	0.0	-0.9	0.0	10/10	A2_c	0.0	14.1	0.0	10/10
A3_b	5.7	717.3	14.1	2/10	A3_c	-0.1	462.6	1.2	8/10
A4_b	4.6	185.2	0.7	9/10	A4_c	1.5	866.5	5.9	0/10
A5_b	2.2	331.0	1.7	8/10	A5_c	11.2	541.1	3.7	6/10
A6_b	6.1	878.2	9.4	0/10	A6_c	1.7	874.4	5.7	0/10

third iteration, C3-C4; and in the fourth only C4 are considered. During the post-processing of the last filled cluster, all cluster types can be used again. This procedure increases the runtime but achieves a reduction of the deviation of the best solution found by the combined CS-P heuristics from 15.4% to 2.0% for cost scenario (a) as displayed in the fourth column of Table 8. This corresponds to an improvement of the best solution found for 55 out of 60 instances. As anticipated, cost structures (b) and (c) exhibit significantly less improvement. Consequently, we do not present detailed results for these structures in the table: Regarding cost structure (b), an improvement of the solution through the additional iterations is only observed for 14 out of 60 instances, which results in a decrease of the average ΔZ from 3.1% to 2.3%. For cost structure (c), only 4 out of 60 instances can be improved by the additional iterations, which corresponds to a decrease in the average ΔZ from 3.7% to 3.2%. It is important to note that the strategic exclusion of cluster types should be adapted to the respective cost structure, which is particularly beneficial when there are slight differences in the c/v-ratio. Runtime can be reduced by considering only certain cluster type combinations defined with the help of test runs.

Table 8: Comparison of the combined CS-P heuristic with the solver on instances of the cloud benchmark for cost scenario (a)

Class	Without Extension		With Extension		Solver	
	ΔZ (%)	Δt (s)	ΔZ (%)	Δt (s)	MIP gap (%)	Terminated
A1_a	9.4	29.7	0.0	18.0	0.0	10/10
A2_a	22.5	12.9	0.0	3.6	0.0	10/10
A3_a	10.5	633.8	5.0	620.7	9.5	3/10
A4_a	15.7	717.8	3.0	631.0	3.6	3/10
A5_a	21.8	782.0	0.9	728.9	5.4	3/10
A6_a	12.5	873.6	3.3	791.2	10.3	0/10

5 Conclusion and Outlook

In this paper, we introduced a new mathematical model called VCSCVBP, which describes a VBP problem with a cluster structure of bins, variable bin sizes, and

cluster costs. We motivated its application in the area of cloud capacity planning with clusters as relevant cost drivers. Due to the increasing demand for cloud computing resources, an optimization approach for tactical cloud capacity planning is of high importance to balance cost optimization and hardware utilization. We introduced the novel CS-P heuristics for the VCSCVBP problem with a focus on clusters of homogeneous bins in the same cluster type consisting of the two parts of vector packing and cluster selection. For the cluster selection, we defined a new metric called cost/combined size-ratio (c/v-ratio). By conducting computational experiments, we demonstrated the runtime benefits of the proposed CS-P heuristics on a self-generated data set based on realistic data from cloud computing. In two out of three cost scenarios, an average deviation of at most 3.7% from the objective value obtained by the solver was reached with a combined heuristic of the different CS-P heuristics. For the third cost structure, we introduced an extension discarding different cluster types, which resulted in an average deviation of 2.0% from the best solution found by the solver. While this paper focused on clusters of homogeneous bins in the same cluster type, we are currently extending our work to clusters of heterogeneous bins. Overall, this paper opens up a new VBP problem setting for further research with a real-world application.

Acknowledgments. The research was conducted as part of the strategic partnership of SAP SE and KIT. We would like to thank SAP SE for the support.

References

1. Amazon Web Services: Amazon EC2: Instance-typen (2024), https://aws.amazon.com/de/ec2/instance-types/, (accessed February 11, 2025)
2. Andreadis, G., Mastenbroek, F., van Beek, V., Iosup, A.: Capelin: Data-driven compute capacity procurement for cloud datacenters using portfolios of scenarios. IEEE Transactions on Parallel and Distributed Systems **33**(1), 26–39 (2022). https://doi.org/10.1109/TPDS.2021.3084816
3. Azar, Y., Emek, Y., van Stee, R., Vainstein, D.: The price of clustering in bin-packing with applications to bin-packing with delays. In: Scheideler, C., Berenbrink, P. (eds.) The 31st ACM Symposium on Parallelism in Algorithms and Architectures. pp. 1–10. ACM, New York, NY, USA (2019). https://doi.org/10.1145/3323165.3323180
4. Bangalore, S., Srivathsan, B., Bhan, A., Del Miglio, A., Sachdeva, P., Sarma, V., Sharma, R.: Investing in the rising data center economy (2023), https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/investing-in-the-rising-data-center-economy/, (accessed February 12, 2025)
5. Bichler, M., Setzer, T., Speitkamp, B.: Capacity planning for virtualized servers. In: 16th Workshop on Information Technologies and Systems (WITS 2006), Milwaukee, Wisconsin, United States (2006)
6. Crainic, T.G., Perboli, G., Rei, W., Tadei, R.: Efficient lower bounds and heuristics for the variable cost and size bin packing problem. Computers & Operations Research **38**(11), 1474–1482 (2011). https://doi.org/10.1016/j.cor.2011.01.001

7. Dell Technologies: Dell server (2024), https://www.dell.com/de-de/shop/scc/sc/servers, (accessed September 16, 2024)
8. Faiz, T.I., Noor-E-Alam, M.: Data center supply chain configuration design: A two-stage decision approach. Socio-Economic Planning Sciences **66**, 119–135 (2019). https://doi.org/10.1016/j.seps.2018.07.008
9. Flexera: 2024 state of the cloud report (2024), https://info.flexera.com/CM-REPORT-State-of-the-Cloud, (accessed February 12, 2025)
10. Gabay, M., Zaourar, S.: Vector bin packing with heterogeneous bins: application to the machine reassignment problem. Annals of Operations Research **242**(1), 161–194 (2016). https://doi.org/10.1007/s10479-015-1973-7
11. Gartner Inc.: Gartner forecasts worldwide public cloud end-user spending to surpass $675 billion in 2024 (2024), https://www.gartner.com/en/newsroom/press-releases/2024-05-20-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-surpass-675-billion-in-2024, (accessed February 12, 2025)
12. Hu, Q., Zhu, W., Qin, H., Lim, A.: A branch-and-price algorithm for the two-dimensional vector packing problem with piecewise linear cost function. European Journal of Operational Research **260**(1), 70–80 (2017). https://doi.org/10.1016/j.ejor.2016.12.021
13. Jiang, Y., Perng, C.S., Li, T., Chang, R.: Self-adaptive cloud capacity planning. In: 2012 IEEE Ninth International Conference on Services Computing. pp. 73–80. IEEE (2012). https://doi.org/10.1109/SCC.2012.8
14. Kou, L.T., Markowsky, G.: Multidimensional bin packing algorithms. IBM Journal of Research and Development **21**(5), 443–448 (1977). https://doi.org/10.1147/rd.215.0443
15. Kouki, Y., Alvares, F., Ledoux, T.: Cloud capacity planning and management. In: Murugesan, S., Bojanova, I. (eds.) Encyclopedia of Cloud Computing, pp. 275–290. Wiley - IEEE, John Wiley & Sons, Chichester, West Sussex (2016). https://doi.org/10.1002/9781118821930.ch23
16. Mell, P.M., Grance, T.: The NIST definition of cloud computing. Recommendations of the National Institute of Standards and Technology, Special Publication **800-145**, 1–3 (2011). https://doi.org/10.6028/NIST.SP.800-145
17. Mommessin, C., Erlebach, T., Shakhlevich, N.V.: Classification and evaluation of the algorithms for vector bin packing. Computers & Operations Research **173**, 106860 (2025). https://doi.org/10.1016/j.cor.2024.106860
18. Nagel, L., Popov, N., Süß, T., Wang, Z.: Analysis of heuristics for vector scheduling and vector bin packing. In: International Conference on Learning and Intelligent Optimization, vol. 14286, pp. 583–598. Springer (2023). https://doi.org/10.1007/978-3-031-44505-7_39
19. Ntaimo, L., Sen, S.: The million-variable "march" for stochastic combinatorial optimization. Journal of Global Optimization **32**(3), 385–400 (2005). https://doi.org/10.1007/s10898-004-5910-6
20. Panigrahy, R., Talwar, K., Uyeda, L., Wieder, U.: Heuristics for vector bin packing (2011), https://www.microsoft.com/en-us/research/publication/heuristics-for-vector-bin-packing/
21. Silver, E.A., Pyke, D.F., Peterson, R.: Inventory management and production planning and scheduling. John Wiley & Sons, New York and Weinheim, 3rd edn. (1998)
22. Varasteh, A., Goudarzi, M.: Server consolidation techniques in virtualized data centers: A survey. IEEE Systems Journal **11**(2), 772–783 (2017). https://doi.org/10.1109/JSYST.2015.2458273
23. Wolf, L., Klos, S., Nickel, S.: Cloud capacity planning under demand uncertainty - a two-stage stochastic approach (2024), (manuscript submitted for publication)