Efficient Lower Bounding Procedures with Application in the Allocation of Virtual Machines to Data Centers

JÜRGEN RIETZ, RITA MACEDO, CLÁUDIO ALVES, JOSÉ VALÉRIO DE CARVALHO
Departamento de Produção e Sistemas
Universidade do Minho
Campus de Gualtar, 4710-057 Braga
PORTUGAL
juergen_rietz@gmx.de; {rita, claudio, vc}@dps.uminho.pt

Abstract: -
Cloud computing is becoming an alternative model for delivering computing resources and services to end-users and companies. The configuration of the clouds raises many issues that come from the need to manage efficiently the available resources in the data centers and from the agreements on the quality of the service that must be delivered to the clients. One of the key issues in the operation of the clouds consists in determining how the workload should be distributed among the physical machines such that the utilization of the computing resources in the cloud computing data centers is maximized. In this paper, we address this latter problem. We describe in particular a set of new and fast procedures for computing lower bounds on the number of physical machines that are required by a cloud provider to execute efficiently a set of user applications (virtual machines).

To compute the bounds, we formulate this virtual machine allocation problem as a bin-packing problem and we address some of its variants. All our lower bounding procedures are polynomial-time algorithms that rely on the use of maximal dual-feasible functions. These functions are parameter dependent. We describe the best set of parameters when the 1-dimensional variant of the problem is considered, and we discuss the complexity of the lower bounding procedures that are proposed. We report also on extensive computational experiments conducted on benchmark instances of the literature. The results of these experiments show the strength of the lower bounds described in this paper.

Key-Words: - Combinatorial optimization; Lower bounds; Maximal dual-feasible functions; Polynomial complexity; Virtual machine allocation problem; cloud computing.

1 Introduction

Cloud computing is a recent computing paradigm based on making available a set of shared computing resources and services to end-users and companies. The resources and services of the clouds are accessed by the clients through a network and on an on-demand basis. The business models underlying the deployment and operation of the clouds are diverse, as are the types of clouds that are currently available and which range from public to virtual private clouds [20]. However, in essence, all these variants share a common fundamental objective which is to offer a service to meet the computing needs of their clients.

A cloud is characterized by its ability to serve simultaneously different end-users from a single data center [20]. Among the various characteristics that distinguish the cloud computing model, the possibility that the end-users have to use dynamically the computing resources of the clouds is clearly one of the most important. This feature of the clouds allows the clients to reduce their operational costs by acquiring only the resources which are really needed at a given point in time. It is also the expression of the pay-as-you-go model on which the cloud computing paradigm is based.

One of the reasons for the advent of cloud computing is the significant increase in the computing needs brought by the explosion of Internet services. Furthermore, all the advantages that the cloud computing paradigm brings to the companies such as a more efficient maintenance of the infrastructures and a better management of the costs (investment and utilization) justify the current shift to cloud computing [2].

The configuration and management of the clouds raise many challenges which have been summarized recently in [20]. In this paper, we consider in particular the problem related to the consolidation of servers in a cloud computing data center. This problem began to be addressed only very recently [20]. One of the key objectives of this consolidation

is to ensure that the available resources are used efficiently so that the energy consumption is minimized.

The increase of the demand for computing resources has led to the deployment of cloud computing data centers and to a corresponding and significant increase of the total energy consumed by these infrastructures. In [8,10], the authors report on the magnitude of the costs involved and on the impact for the environment caused by these levels of energy consumption. The expected rate of growth of these indicators is also impressive [8, 10]. Meanwhile, the average utilization of the resources in the data centers remains typically low (15% to 20% on average [19]). A solution to this issue consists in performing the aforementioned server consolidation through an efficient balancing of the workload across the physical machines of the data centers.

Server consolidation is achieved through the virtualization technology which is one of the foundations of cloud computing. In this context, the elementary processing component is the virtual machine which encapsulates the user applications. The data centers relying on virtualization are also referred to as virtualized data centers [11]. The virtualization technology allows for a flexible management of the computing resources available in the cloud computing data centers through the allocation of many virtual machines to the same physical server. This allocation aims at balancing the workloads assigned to each server to an optimal energy point avoiding idle energy wastage.

A virtual machine can be characterized through a set of hardware metrics representing resource consumptions at the physical server hosts. Examples of these resources are the amount of allocated RAM, the number of CPU cores, the network bandwidth or the disk size. The number of different resources to consider when allocating virtual machines to physical servers can be treated as a different dimension of a bin-packing problem. It is therefore possible to address the virtual machine allocation problem as a multi-dimensional bin-packing problem, where the number of dimensions depends on the considered resource constraints.

As mentioned above, the virtual machine allocation problem has been pointed out by many authors [11, 18, 20] as an important issue in the management of cloud computing data centers. The research on this field is very recent. In [18], the authors describe a set of integer programming based algorithms for the placement of virtual machines under several constraints defined by the user. These procedures are embedded in a general approach for cloud brokering. A different approach for the management of virtualized data centers was proposed in [11]. In the latter, the problem of scheduling tasks in a data center (and thus allocating them to physical machines) is formulated as a multi-objective optimization problem. To forecast the workload, the authors resort to predictions made through a fuzzy method.Complexity results are provided together with a set of experimental results.

The virtual machine allocation problem has been addressed by many authors who formulated it as a bin-packing problem [1, 8, 12, 16, 17]. Given the inherent complexity of these combinatorial optimization problems, and the requirement for a fast allocation of the virtual machines in real data centers, it is not surprising that all the contributions described in the literature focus exclusively on heuristic approaches. In this paper, we provide several lower bounding procedures to compute the minimum number of physical machines required to serve a given set of virtual machines. These procedures can be used within these heuristics to improve the quality of the generated solutions or simply to evaluate how far these solutions are from an optimal solution.

In [17], a private cloud management system is presented. The authors describe a system that integrates different mechanisms for allocating virtual machines to physical hosts, for managing congruent configurations and for the bootstrapping of blank physical machines and virtual machines. The allocation of the virtual machines is performed by considering three resources: the RAM, the number of CPU cores and the type of architecture. In fact, the latter is considered as a constraint on the subset of physical machines that can effectively handle some of the virtual machines. Given that each virtual and physical machine has usually a single type of architecture, the problem is decomposed into *n* different subproblems, one for each different architecture. Each subproblem only considers the machines of compatible architecture types. Hence, the problem reduces to a 2-dimensional bin-packing problem. The authors describe a heuristic for this problem based on the well-known first-fit decreasing algorithm.

In [8], the authors propose an algorithm for the virtual machine allocation problem based on the ant colony optimization meta-heuristic. Again, their approach is based on a formulation of the problem as a multi-dimensional bin-packing problem. The focus of their contribution is on the generation of solutions that minimizes the number of physical machines used in order to decrease the energy consumed by the data centers. Their algorithm was

compared to the standard first-fit decreasing heuristic. The results reported in their paper show that their swarm intelligence based approach outperforms this greedy heuristic.

In [1], the authors address the virtual machine allocation problem by focusing on a single type of resources, namely the memory of the physical machines in the data centers. Hence, the problem reduces to a classical 1-dimensional bin-packing problem. They describe online algorithms for this problem that are extensions of other approaches described in the literature, and they show that the approximation ratio of their approach remains similar to those of the original approaches.

Li et al. describe in [12] the EnaCloud approach for allocating virtual machines to physical servers with a focus on energy saving. They propose a so-called energy-aware heuristic that indicates dynamically how the applications encapsulated into virtual machine should be placed within the physical servers. Their approach is further improved by considering an over-provisioning method that addresses the variability related to the resource requirements of the applications.

In [16], Mastroianni et al. present an approach for the adaptive allocation of virtual machines to physical servers based on statistical procedures. A physical machine is selected to receive a virtual machine through Bernouilli trials by giving preference to those servers that are already highly loaded. The authors further consider the possibility of live migration of the virtual machines both to avoid the occurrence of servers with small workloads and to avoid the possible overloading of the servers which may result in a violation of the service agreements. The authors report on computational experiments from which it transpires that their method tends to be decentralized and self-organizing in part due to the adaptive nature of their migration approach.

In this paper, we contribute to the resolution of the virtual machine allocation problem by describing fast lower bounding procedures that allows to determine a bound on the minimum number of physical machines that are required to handle a given set of virtual machines. The remainder of the paper is organized as follows. In Section 2, we formalize the problem and we discuss its relations with the family of cutting and packing problems. In Section 3, maximal dual-feasible functions are introduced, and some important families are reviewed. In Section 4, their application to the 1-dimensional cutting stock problem (a problem that is equivalent to the bin-packing problem as shown in the next section) is analysed. In Sections 5 and 6, the 2-dimensional guillotine cutting stock problem with 2 stages is discussed, in the exact and the inexact case, respectively, and we show how to derive new lower bounds for those cases. We address the computation of lower bounds for the non-guillotine case in Section 7, and then we present computational results in Section 8 for ten classes of test instances from the literature [15]. Some conclusions are finally drawn in Section 9.

2 The Virtual Machine Allocation Problem

Given a set of physical machines (or physical servers) and a set of virtual machines, the virtual machine allocation problem consists in determining the best assignment of the virtual machines to the physical ones so as to minimize a given criterion. In most of the cases, the criterion that is considered is the number of physical machines that are effectively used. This criterion allows for the minimization of the energy consumption of the data centers as mentioned in the previous section. In the literature, the virtual machine allocation problem is sometimes referred to as the virtual machine placement problem.

The virtual machine allocation problem can be formulated as a bin-packing problem (BPP). Bin-packing problems belong to the class of cutting and packing problems. They can be generally stated as the problem of assigning a set of items to a set of larger objects, usually called bins, such that the total number of bins that are used is minimized. In the virtual machine allocation problem, the set of virtual machines can be seen as the set of items to be allocated to the set of physical machines, which may in turn be considered as the bins. The assignment of the items to a bin must be such that the capacity of the bin is never exceeded in none of its dimensions. In the virtual machine allocation problem, these capacities correspond to the levels of the resources that are considered.

In its general form, the virtual machine allocation problem is a multi-dimensional vector bin-packing problem. In the 1-dimensional case, this problem reduces to a standard 1-dimensional bin-packing problem (1D-BPP) as illustrated in Figure 1. In the 2-dimensional case, the problem can be seen as 2-dimensional bin-packing problem (2D-BPP) with guillotine cuts and cardinality constraints (Figure 2). In this paper, we consider these two equivalent formulations of the problem, and we provide fast (polynomial-time) procedures to compute lower bounds for each one. Although we are addressing

the cases where one or two resources are considered simultaneously, our lower bounding procedures can be easily extended to higher dimensions by considering either each dimension or each pair of dimensions (resources) separately, and then by choosing the smallest value of the generated bounds.

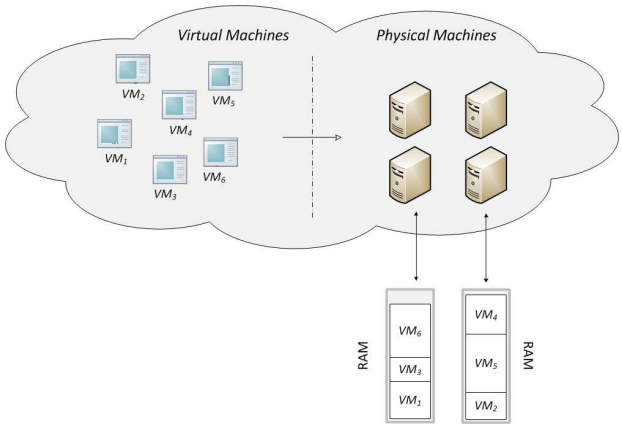

Figure 1: Virtual machine allocation problem as a 1D-BPP

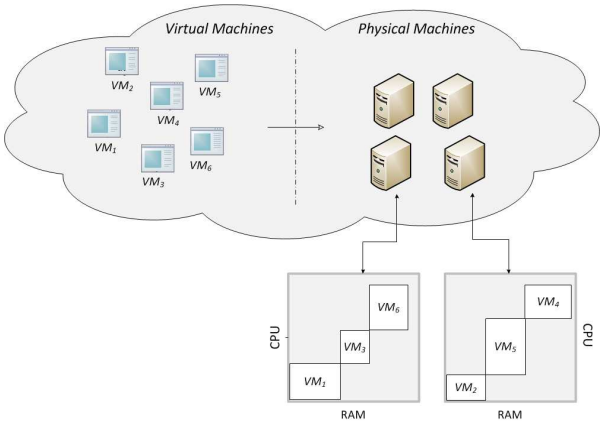

Figure 2: Virtual machine allocation problem as a 2D-BPP

In the cutting and packing terminology, the designation of bin-packing problem refers usually to those instances in which all the items are different in terms of their sizes (resource consumptions in the case of virtual machine allocation). If different items have the same sizes in all the dimensions, the problem is referred to as a cutting stock problem (CSP). The counterpart of a bin in the terminology of cutting stock problems is a stock sheet (or simply a stock), while the packing operation is seen equivalently as a cutting operation. For the sake of generality, from this point forward, we will refer to the cutting stock problem since we do not impose any special requirement on the resource consumptions of the virtual machines (*i.e.* we consider that different virtual machines may consume the same amounts of resources).

The 1-dimensional cutting stock problem (1D-CSP) is NP-hard in the strong sense [14]. Hence, its 2-dimensional counterpart (the 2D-CSP) is also NP-hard. Most of the approaches proposed in the literature to solve these combinatorial optimization problems rely on heuristics, see e.g. [7, 13]. In this paper, we resort to dual-feasible functions to derive efficient lower bounding procedures for different variants of the 2D-CSP. The bounds generated by these procedures are valid lower bounds for the corresponding virtual machine allocation problem. Our computational results reported at the end of this paper show that these procedures generate strong lower bounds very efficiently, *i.e.* with a low polynomial complexity.

An instance of the 2D-CSP can be described by the length L and width W of the stock sheets, the number m of (different) items to be cut, and their sizes $\ell_i \times w_i$ and order demands b_i, $i = 1,\ldots,m$. All input data are assumed to be positive and integer. The length L and width W correspond to the available amount of the resources in the virtual machine allocation problem, while the sizes l_i and w_i denote the corresponding consumption of each of these resources by the virtual machine i. We assume that the items cannot be rotated since that will imply exchanging the resource requirements of a virtual machine. This situation will certainly lead to solutions that are infeasible for the virtual machine allocation problem. Furthermore, we will assume without loss of generality that the first cut is horizontal and that $\ell_i \le L$ and $w_i \le W$, for all $i \in \{1,\ldots,m\}$. The so-called material bound is computed in the following way:

$$z_M = \sum_{i=1}^{m} \frac{b_i \times \ell_i \times w_i}{L \times W}, \qquad (1)$$

Clearly, $\lceil z_M \rceil$ is a lower bound for the 2D-CSP. Indeed, if $B \in IR$ is a lower bound, then $\lceil B \rceil$ is a lower bound too.

The majority of the authors addressed the computation of lower bounds for 2D non-guillotine CSP, as for instance [7]. A guillotine cut is a cut from one edge to the opposite edge of the stock

sheet that divides it into two parts. By definition, a lower bound for the 2D non-guillotine CSP is a valid lower bound for the variant with guillotine cuts. Similarly, both bounds are valid for the virtual allocation problem. One of the objectives of this paper is to show that better bounds can be obtained for the 2D guillotine CSP with 2 stages. Furthermore, we show how to choose good parameters for some families of maximal dual-feasible functions with low complexity.

3 Dual-Feasible Functions

Dual-feasible functions were introduced by Johnson in [9]. Since that time, many other functions were found and analysed. An exhaustive survey is given in [5].

Definition 1: A function $f:[0,1]\to IR_+$ is called a *dual-feasible function* (*DFF*), if for any finite set $\{x_i \in IR_+ : i \in I\}$ of nonnegative real numbers, the following holds:

$$\sum_{i\in I} x_i \le 1 \Rightarrow \sum_{i\in I} f(x_i) \le 1.$$

If f is a DFF, then it follows that $f(0)=0$, and $f(x)\le 1/n$ for all $IN\setminus\{0\}, x\in[0,1/n]$. Moreover, for an integer programming problem with a knapsack constraint of the form $\sum_i a_i x_i \le 1$, and all coefficients $a_i \in [0,1]$, we have that $\sum_i f(a_i)x_i \le 1$ is a valid inequality, because

$$\sum_{i=1}^{n} a_i x_i = \sum_{i=1}^{n}\sum_{k=1}^{x_i} a_i \Rightarrow \sum_{i=1}^{n}\sum_{k=1}^{x_i} f(a_i) = \sum_{i=1}^{n} f(a_i)x_i \le 1.$$

Dual-feasible functions became very important for deriving lower bounds for many integer linear optimization problems, and also to compute valid inequalities for these problems. Because the above definition allows a lot of non-interesting functions, stronger definitions were introduced to characterize the non-dominated DFFs. These functions are called *maximal* DFFs. The formal definition of maximality is given next.

Definition 2: A dual-feasible function $f:[0,1]\to[0,1]$ with $f(1)=1$ is called a *maximal dual-feasible function (MDFF)*, if there is no other dual-feasible function $g:[0,1]\to[0,1]$ with $g(x)\ge f(x), \forall x\in[0,1]$.

Note that any composition or convex combination of MDFFs is again a MDFF.

Our lower bounding procedures rely on a set of strong DFFs proposed in the literature, and surveyed recently in [5]. From this point forward, we will denote by *frac(.)* the non-integer part of the real argument, i.e. $frac(x) = x - \lfloor x \rfloor$. These functions are defined as follows.

• $f_{MT,0}$ (defined implicitly by Martello and Toth in [14]) for $\lambda \in (0,1/2]$:

$$f_{MT,0}(x) = \begin{cases} 0, & \text{if } x < \lambda, \\ x, & \text{if } \lambda \le x \le 1-\lambda, \\ 1, & \text{otherwise.} \end{cases}$$

• $f_{CCM,1}$ [4] for any $\lambda \ge 1$ with $\lambda \in IR$:

$$f_{CCM,1}(x) = \begin{cases} \lfloor \lambda x \rfloor / \lfloor \lambda \rfloor, & \text{if } 0 \le x < 1/2, \\ 1/2, & \text{for } x = 1/2, \\ 1 - f_{CCM,1}(1-x), & \text{if } 1/2 < x \le 1. \end{cases}$$

• $f_{BJ,1}$ [3] for $\lambda \ge 1$ and $\lambda \notin IN$:

$$f_{BJ,1}(x) = \left(\lfloor \lambda x \rfloor + \max\left\{0, \frac{frac(\lambda x) - frac(\lambda)}{1 - frac(\lambda)}\right\}\right) / \lfloor \lambda \rfloor,$$

• $f_{DG,1}$ [6] for $\lambda > 1$, $\lambda \notin IN$ and $k \in IN$, with $k \ge \lceil 1/(\lambda) \rceil$:

$$f_{DG,1}(x) = \frac{\lfloor \lambda x \rfloor}{\lfloor \lambda \rfloor} + \begin{cases} \dfrac{frac(\lambda x) - frac(\lambda)}{1 - frac(\lambda)} / \lfloor \lambda \rfloor, \\ \qquad \text{if } (k-1)\dfrac{frac(\lambda x) - frac(\lambda)}{1 - frac(\lambda)} \in IN, \\ \max\left\{0, \left\lceil (k-1)\dfrac{frac(\lambda x) - frac(\lambda)}{1 - frac(\lambda)} \right\rceil\right\} / (k \lfloor \lambda \rfloor), \\ \qquad \text{otherwise.} \end{cases}$$

- $f_{LL,2}$ [5] for $\lambda > 1, \lambda \notin IN$ and $\psi := \lceil 1/frac(\lambda) \rceil$:

$$f_{LL,2}(x) = \begin{cases} \left(\lfloor \lambda x \rfloor + \max\{0, \left\lceil \frac{frac(\lambda x) - frac(\lambda)}{1 - frac(\lambda)} (\psi - 1) \right\rceil / \psi \right\} \right) / \lfloor \lambda \rfloor, & \\ & if \ x < 1/2, \\ 1/2, & if \ x = 1/2, \\ 1 - f_{LL,2}(1-x), & if \ x > 1/2. \end{cases}$$

- $f_{FS,1}$ [7] for any $k \in IN \setminus \{0\}$:

$$f_{FS,1}(x) = \begin{cases} x, & if \ (k+1)x \in IN, \\ \lfloor (k+1)x \rfloor / k, & otherwise. \end{cases}$$

- $f_{VB,2}$ [5] for any $k \in IN \setminus \{0,1\}$:

$$f_{VB,2}(x) = \begin{cases} \max\{0, \lceil kx \rceil - 1\}/(k-1), & if \ x < 1/2, \\ 1/2, & if \ x = 1/2, \\ 1 - f_{VB,2}(1-x), & if \ x > 1/2. \end{cases}$$

3 1D-CSP and Best Parameter Choice

In this section, we review how the MDFFs should be used to generate lower bounds for the 1-dimensional case. For this purpose, let

$$f : [0,1] \to [0,1]$$

be a MDFF, and

$$E := (m; L; l; b)$$

an instance of the 1D-CSP with a stock length $L \in IR_+ \setminus \{0\}$, $m \in IN \setminus \{0\}$ items whose lengths and order demands are given in the vectors $l \in IR^m$ and $b \in IN^m$, respectively, with $0 < l_i \le L$ for $i = 1, \ldots, m$. The surrogate instance is the following instance E_f:

$$E_f := (m; 1; (f(\ell_1/L), \ldots, f(\ell_m/L))^{\mathrm{T}}; b).$$

Any lower bound for the optimal solution value of E_f is a lower bound for E too.

If f is chosen optimally, then $z_M(E_f)$ equals the continuous relaxation bound given by the column generation model of E. In fact, the optimal dual solution of the column generation model provides a dual-feasible function for *E*. The complexity of computing the column generation bound for *E* is well-known to be high, and so will be the complexity of finding an optimal MDFF.

In order to find a good MDFF with low complexity, we will resort only to the functions described in the previous section.

Finding the best parameter for $f_{MT,0}$ requires first that the items are sorted by their lengths. This step has a complexity of at least $O(m \ln m)$. After that, the complexity of the corresponding Martello and Toth lower bound is only linear in m. This bound can be computed by considering the pairs of items that do not fit together. The corresponding lower bounding algorithm is given below.

Algorithm MT_Original

Input: Instance $E = (m; L; l; b)$ of the 1D-CSP.

Output: Martello and Toth lower bound (B_{MT}) for the optimal solution value

1. Sort all items by their length such that $\ell_1 \ge \ldots \ge \ell_m$.
2. Calculate the material bound $z_M = \sum_{i=1}^{m} b_i \ell_i / L$ for E.
3. Set $B := z_M$, $B_{MT} := z_M$, $i := 1$, $j := m$.
4. **while** $i \le j$ **do**
   if $l_i + l_j < L$ **then**
   B:=B-b_j l_j/L; j:=j-1;
   end
   else
   B:=B+b_i (1-l_i/L); i:=i+1;
   if $B > B_{MT}$ **then** $B_{MT} := B$;
   end

This bound B_{MT} is never worse than the material bound z_M. Furthermore, it can yield very tight results if *E* has items greater than $L/2$. Hence, it is a good idea to combine this bound with the other functions, as long as the complexity remains low, e.g. $O(m)$ for trying a MDFF after the sorting. Since any MDFF is monotonous, the sorting can be done first.

In the sequel, we describe a new variant of the previous algorithm that avoids storing the modified item sizes given by the DFFs. These values may be used at the steps 2 and 4 of the algorithm MT_Original instead of the original values l_i. This variant is particularly useful when composing DFFs,

and it may speedup the computation of the lower bounds.

Algorithm MT_Online
Input: Instance $E = (m; L; l; b)$ of the 1D-CSP with $L \geq \ell_1 \geq \ldots \geq \ell_m > 0$ and a MDFF f.
Output: Martello and Toth lower bound for the corresponding instance E_f.

The auxiliary vector $d \in IR^6$ contains the largest remaining item, the smallest remaining one, the actual material bound, the difference between the Martello and Toth lower bound and the material bound, and in the 5th and 6th it contains auxiliary values. The order demands of the remaining largest and smallest item are stored in $r \in IN^2$. The indices i, j, k belong to the largest or smallest remaining item while $n \in \{1,2\}$ is used to decide if the largest or the smallest item is replaced.

Set ***d***:=***o*** *(the zero vector), i:=1, j:=m+1, k:=1, n:=1 and* ***r***:=$(0,0)^T$.
repeat
  $d_3 := d_3 + d_6$;
  $d_n := f(l_k/L)$; $r_n := b_k$;
  $d_6 := d_n \cdot r_n$;
  if $d_1 + d_2 \leq 1$ **then**
    $n := 2$; $d_5 := d_5 - d_2 \cdot r_2$;
    $j := j-1$; $k := j$;
  end
  else
    $n := 1$; $d_5 := d_5 + (1-d_1) \cdot r_1$;
    if $d_5 > d_4$ **then** $d_4 := d_5$;
    $i := i+1$; $k := i$;
  end
until $i > j$;
return $d_3 + d_4$.

If a lower bound $B \in IR$ was given, then the last line could be replaced by "if $d_3 + d_4 > B$ then $B := d_3 + d_4$", so as to update that bound.

To obtain strong lower bounds for the 1D-CSP, it is important to choose a good set of parameters. The decision on the parameter can be done regarding a single item in E. The aim is to let the rounding-down in the functions do as few as possible for the selected item. Trying all the m items yields therefore m calls of the above algorithm for each of the functions $f_{CCM,1}$, $f_{BJ,1}$, $f_{DG,1}$, and $f_{LL,2}$. The functions $f_{FS,1}$ and $f_{VB,2}$ are more complicated.

Choosing the parameters depending on the length ℓ_i of the i-th item can be done with a suitable small $\varepsilon > 0$ as follows:

- $f_{CCM,1}$: if $\ell_i < L/2$, then let $\lambda := L/\ell_i$ else $\lambda := L/(L - \ell_i + \varepsilon)$;

- $f_{BJ,1}$, $f_{DG,1}$ and $f_{LL,2}$: if the λ for $f_{CCM,1}$ is not integer, then this λ is suitable for $f_{BJ,1}$, $f_{DG,1}$, and $f_{LL,2}$ too.

- $f_{FS,1}$: Fekete and Schepers [7] found that using only this function yields a worst case performance ratio of $3/4$, *i.e.* that there is a sequence of bad instances such that the ratio of the obtained bound and the optimal solution value tends to $3/4$ even if all possible parameter values $k \in \backslash\{0\}$ are tried. Nevertheless, some values k can be tried such that the fractional part of $(k+1)\ell_i/L$ becomes very small. Such suitable values k can be computed with continued fractions, where L/ℓ_i is approximated by rational numbers with smaller denominators as follows:

Let $a := 0$; $b := 1$; $y := l_i / L$;
repeat
  $y := 1/frac(y)$;
  $c := \lfloor y \rfloor \cdot b + a$;
  $a := b$; $b := c$;
if $frac(b \cdot l_i/L) < 0.5$ and $b > 2$ **then**
  Use $k := b-1$ as parameter for $f_{FS,1}$;
until $frac(y) = 0$.

The value $k = 1$ yields the identity function, and hence it is clearly not an interesting value for the parameter. In the loop, the number of parameter values that are tried is $O(\ln L)$. Therefore, the complexity of the loop is $O(m \ln L)$, and hence it is linear in m (note that applying one of the functions to all the items requires clearly linear complexity).

In order to prevent to apply one parameter k arising from different lengths ℓ_i more than once for the function $f_{FS,1}$, all obtained values k can be stored in a binary search tree, for instance in a red-black tree. These trees guarantee that finding and/or inserting a node requires a complexity of $O(\ln n)$, if the tree contains $n > 0$ nodes. If the value k is not found in the tree, then it is inserted in the tree, and

the bound is computed using $f_{FS,1}$ with parameter k. Otherwise, the calculation can be omitted. Because $O(m \ln L)$ values of the parameter k can be in the search tree, a trivial searching structure cannot be recommended, since looking for every found parameter value in the structure would increase that complexity to $O(m \ln L)^2$. That would be more expensive than not using any check, if that value k was already tried or not, which would cause a total complexity $O(m^2 \ln L)$. If all $O(m \ln L)$ possible values for k are stored in a red-black tree, then the total complexity for searching these nodes is around $O(m \ln L \ln(m \ln L))$, and the entire effort for using $f_{FS,1}$ becomes $O(m^2 \ln L)$. The latter dominates the first expression.

- $f_{VB,2}$: here, the same idea as for $f_{FS,1}$ can be used.

In conclusion, a lower bound for the 1D-CSP can be computed with quadratic complexity in the number m of items as follows:

1. Sort all items by their size such that $L \geq \ell_1 \geq \ldots \geq \ell_m$;
2. Compute the Martello and Toth lower bound for the instance, i.e. using $f_{MT,0} \circ f_{id}$, with $f_{id}(x) = x$ being the identity function;
3. For all $i \in \{1,\ldots,m\}$, choose suitable parameters as described above and apply the functions together with $f_{MT,0}$.

4 2D Guillotine CSP with 2 Stages: the Exact Case

In the exact case, horizontal strips with widths belonging to $\{w_1,\ldots,w_m\}$ are cut in the first stage, while in the second stage, these strips are cut to the desired items. No additional cuts are allowed. In this paper, we will assume that neither the stock sheets nor the items can be rotated. The lower bound obtained for this 2D-CSP is valid for the virtual machine allocation problem introduced above.

In this problem, the gap between the optimal solution value of the integer problem and its linear relaxation can increase affine-linearly with m. Therefore, the difference between the obtained lower bound and an upper bound, found e.g. by a heuristic like the worst-fit-decreasing heuristic (WFD), can rise beyond any constant. A lower bound for the problem can be obtained using the following procedure:

1. Sort all the items such that $w_1 \geq \ldots \geq w_m$. If $w_i = w_j$ and $i < j$, then sort items *i* and *j* such that $\ell_i \geq \ell_j$;
2. For each item width, compute the number of necessary strips by applying MDFFs as described in the previous section. Take the maximum value for the bound and round it up to the next integer;
3. Compute the number of required stock sheets regarding the obtained numbers of strips analogously.

These calculations are a simple sequence of calculations of lower bounds for 1D-CSP instances. Therefore, the complexity to get the lower bound will be nearly $O(m^2)$.

The WFD heuristic can be implemented efficiently using a priority queue. For each width, a sufficient number of strips is calculated, and then the strips are combined into stock sheets. Since these are again only 1D instances, this can be done in detail as follows:

1. Initialize a priority queue q as empty. The head of q shall always contain the largest stored value, if q is not empty;
2. For each width $w \in \{w_1,\ldots,w_m\}$: if q is not empty, and the value at the head of q is greater than or equal to the length of the item with width w, then use the represented strip of width w, otherwise use a new strip with size $L \times w$. Put as much of the items into the strip as possible, regarding their order demands. Store the new or the changed strip in q. After putting all the items of the selected width w into the strips, the number of elements in q equals the number of needed strips. Store this number for the next step and empty q.
3. Obtain the number of needed stock sheets depending on the number of necessary strips of the different widths in the same way.

The complexity for the entire WFD is then only $O(m \ln m)$.

5 2D Guillotine CSP with 2 Stages: the Inexact Case

This problem differs from the previous in the fact that, for each item, one additional cut to remove waste is allowed. In practice, this variant allows the placement of items with different heights in the

same strip. This characteristic makes the calculation of good lower bounds a bit more complicated than in the exact case. The resulting bound remains feasible for the virtual machine allocation problem.

Here, we propose the following lower bounding algorithm for this problem with a complexity $O(m^3)$. Let $s \in IN^m$, our algorithm states as follows:

1. Sort all the items like in the exact case. Let $i := m$;
2. Compute the number of needed horizontal strips for all the items of $\{1,\ldots,i\}$, i.e. apply several MDFFs, take the maximum of the lower bounds and round it up to the next integer. Store this number in s;
3. If $i < m$, then subtract the last obtained number in s from the previous one;
4. Repeat $i := i-1$ until $i = 0$ or $w_i \neq w_{i+1}$;
5. If $i > 0$, then goto Step 2.
6. Compute the number of needed stock sheets according to the widths of the strips and their numbers stored in s.

Since the bound for the 1D problem needs sorted items, it is necessary to sort a copy of the items of $\{1,\ldots,i\}$ in Step 2. The complexity $O(m^3)$ can be argued as follows: if all the items have different widths then for $i = m, m-1,\ldots,1$, lower bounds for the number of strips with i items must be calculated. The complexity for that is each time $O(i^2)$, and together $O(m^3)$.

The following example shows that the lower bound for the 2-stage guillotine CSP can dominate the best lower bound for the non-guillotine counterpart.

Example 1: Let $L=W=m=3$, $l_1=w_2=b_1=b_2=2$, and $\ell_2 = w_1 = \ell_3 = w_3 = b_3 = 1$. In the first step, items 1 and 2 are exchanged during the sorting. That means that the items have now the sizes 1×2, 2×1 and 1×1. At the beginning, a lower bound for the number of all strips of widths greater than or equal to 1 is computed (Step 2). Since all the items are involved, the copied lengths (and order demands) have to be exchanged again such that $\ell_1 = 2$, $\ell_2 = \ell_3 = 1$. The simple material bound yields $(2\times 2 + 2\times 1 + 1\times 1)/3$, and hence three strips are necessary. In Step 4, the index variable i decreases to 1 because in the sorted list of items after Step 1, the relations $w_1 = 2 > w_2 = 1$ hold.

Now, the strips with the next larger width 2 are analyzed. The only item is 1×2, which is demanded twice. Therefore, one strip of width 2 can be enough. Since the total number of strips is (at least) 3, the lower bound for the number of stock sheets arises from one strip of width 2 and two strips of width 1. If strips of width 1 are changed to width 2, then neither the whole number of strips nor the total number of stock sheets can decrease. Therefore, the subtraction in step 3 is correct. In Step 6 the material bound yields that at least $(1\times 2 + 2\times 1)/3$, and hence two stock sheets are necessary.

Because there is a non-guillotine arrangement of all the items on one stock, this is an example where the obtained lower bound for the inexact case of guillotine cutting in two stages is higher than any valid lower bound for the non-guillotine packing.

6 The Non-Guillotine 2D-CSP

The strategy for obtaining lower bounds for this ordinary orthogonal cutting stock problem is the following. We apply a set of MDFFs to all the items in one direction and a set of the same or other MDFFs to all the items in the second direction. Then, the material bound z_M for the changed instance is a lower bound for the original one. Because it is necessary to explore both directions at the same time, and not sequentially as in the previous sections, the complexity increases.

Since in the 1D case the Martello and Toth lower bound was very successful when it was combined with the other MDFFs (except with $f_{LL,2}$ and $f_{VB,2}$), we resort again to the function $f_{MT,0}$ together with f_{id}, $f_{CCM,1}$, $f_{BJ,1}$, $f_{DG,1}$ and $f_{FS,1}$. Note that the total area of an item (with its original size or after applying a MDFF) is very often less than half of the stock area, causing $f_{MT,0}$ to yield a function value 0. Therefore, it is not useful to compute the Martello and Toth lower bound for the areas of all the items. Using the other MDFFs for the areas would also increase the complexity further without strong effect on the bounds.

The details of the lower bounding procedure are described next.

1. Construct vectors to sort all the items indirectly by their lengths and by their widths. Let $B := 0$;
2. Construct a surrogate instance E_f with stock length $L' = 1$ and item lengths $f(\ell_i/L)$, where f is

the composition of f_{id}, $f_{CCM,1}$, $f_{BJ,1}$, $f_{DG,1}$ or $f_{FS,1}$ and $f_{MT,0}$ according to the 1D case, and call an analogous procedure applied now to the item widths;
3. For each combination of MDFFs, compute the material bound. If the returned value is larger than B, then update B accordingly.

The value B is the desired lower bound. The complexity of this procedure is $O(m^4 \ln L \ln W)$, since for each different item length a set of MDFFs is applied to all item lengths, and for each such surrogate instance, the widths must be handled analogously.

Composing $f_{MT,0}$ with one of the other MDFFs does not increase the complexity. This can be confirmed as follows. Choosing the parameter for the MDFF brings a factor m and eventually additionally $\ln L$ or $\ln W$. Applying the chosen function with the chosen parameter yields another factor m. After that, $f_{MT,0}$ is applied to the data which needs no additional factor m. Since the computation of the material bound can be done before using $f_{MT,0}$ in the second direction, and then changing the parameter in $f_{MT,0}$ can be regarded each time with constant complexity, the total effort $O(m^4 \ln L \ln W)$ can be achieved. The factors $\ln L$ and $\ln W$ come from the parameter search for $f_{FS,1}$ (and $f_{VB,2}$).

7 Computational Results

The algorithms described in the previous sections were tested on instances from the literature, namely on ten test classes by Berkey and Wang and Martello and Vigo described in [15]. Each class is composed by 5 groups of 10 instances, one group for each value of *m* from 20 to 100. The instances can be obtained from www.or.deis.unibo.it/research_pages/ORinstances/2BP.html.

Our results are reported on Table 1 to 5. The entries in these tables have the following meaning: m stands for the number of items, z_M denotes the material bound for the original instances, the lower bound for the non-guillotine problem is denoted by B_{NG}, B_{in} and B_{ex} are the bounds inexact and exact case of the guillotine CSP, and WFD stands for the upper bound obtained with the worst-fit-decreasing heuristic for the exact 2D guillotine CSP in two stages.

To derive the value of each of the bounds B_{in} and B_{ex}, we used all the MDFFs described in Section 2, i.e. $f \circ f_{MT,0}$ with $f \in \{f_{id}, f_{CCM,1}, f_{DG,1}, f_{BJ,1}, f_{LL,2}, f_{FS,1}, f_{VB,2}\}$, and then we took the largest value found. B_{NG} was calculated analogously except that the functions $f_{LL,2}$ and $f_{VB,2}$ were omitted. Indeed, our preliminary experiments showed that these functions dominated the others only very rarely.

Given the low complexity of our lower bounding procedures, the time required to compute the bounds is typically a few miliseconds for all the instances that were used.

Table 4 and 5 report the exhaustive list of lower bounds obtained for each case. In 457 of the 500 instances, our lower bounding procedure for the exact case of the 2D guillotine CSP was able to find the value of the optimal solution (B_{ex} is equal to the upper bound provided by WFD heuristic). For 347 instances, the bound for the inexact case improved the material bound computed from the original instance. The average increase of the lower bound in the sense $\sum \lceil B_{NG} \rceil / \sum \lceil z_M \rceil$ is given in Table 1 for the ten classes.

The comparison with the results of Martello and Vigo [15] shows that the lower bounds with dual-feasible functions can yield better results. This can even happen if the number of large items is small as in class 10. In that class Martello and Vigo did not get better bounds than $\lceil z_M \rceil$.

Table 2 presents the average increase of the lower bounds $\sum B_{in} / \sum \lceil B_{NG} \rceil$, when comparing the non-guillotine 2D-CSP and the inexact 2 stages guillotine CSP. These results shows that the latter bounds can be sharpened for the inexact case too while the computational effort decreases.

Table 3 contains an analogous comparison of the inexact and the exact case, i.e. $\sum B_{ex} / \sum B_{in}$. It is not surprising that the lower bounds for the exact guillotine CSP with 2 stages rise very strongly compared with the inexact case, especially in Class 6. Indeed, in the exact case, since each strip can be used only for the items of the same width, it can easily happen that the strips contain only one item but much waste. In contrast, the inexact case allows the combination of different items in one strip.

CLASS	*m*=20	*m*=40	*m*=60	*m*=80	*m*=100
1	69/64	131/120	200/185	275/253	317/305
2	1	1	1	1	1
3	47/44	92/82	136/125	188/173	221/205
4	1	1	1	1	1
5	60/54	116/101	177/157	243/215	279/259
6	1	1	1	1	1
7	53/47	109/97	156/140	224/197	269/238
8	55/48	112/96	159/141	223/195	274/241
9	143/94	275/180	435/276	574/371	693/450
10	41/38	72/69	98/94	124/122	153/153

Table 3: Comparing B_{NG} with z_M

$$\sum \lceil B_{NG} \rceil / \sum \lceil z_M \rceil$$

CLASS	*m*=20	*m*=40	*m*=60	*m*=80	*m*=100
1	71/69	136/131	201/200	275/275	321/317
2	10/10	20/19	27/25	33/31	40/39
3	54/47	95/92	140/136	192/188	224/221
4	10/10	20/19	26/23	33/30	39/37
5	66/60	120/116	179/177	246/243	282/279
6	10/10	19/15	22/21	30/30	34/32
7	56/53	115/109	159/156	231/224	271/269
8	59/55	113/112	162/159	225/223	279/274
9	143/143	278/275	437/435	577/574	695/693
10	45/41	75/72	103/98	129/124	159/153

Table 4: Comparing B_{in} with B_{NG}: $\sum B_{in} / \sum \lceil B_{NG} \rceil$

CLASS	*m*=20	*m*=40	*m*=60	*m*=80	*m*=100
1	86/71	157/136	230/201	304/275	348/321
2	20/10	28/20	33/27	39/33	47/40
3	85/54	147/95	203/140	267/192	301/224
4	32/10	48/20	59/26	66/33	70/39
5	102/66	193/120	286/179	387/246	432/282
6	34/10	59/19	85/22	101/30	113/34
7	135/56	241/115	324/159	406/231	450/271
8	73/59	151/113	211/162	284/225	344/279
9	158/143	316/278	474/437	623/577	746/695
10	86/71	157/136	230/201	304/275	348/321

Table 5: Comparing B_{ex} with B_{in}: $\sum B_{ex} / \sum B_{in}$

8 Conclusion

In this paper, we proposed new and efficient procedures to compute good lower bounds for 2-dimensional cutting stock problems. All the lower bounds described in this paper are feasible for the virtual machine allocation problem in data centers. Hence, they can be used to determine the minimum number of physical machines that are required for handling a set of virtual machines. Our procedures are based on maximal dual-feasible functions. In the first part of the paper, we described strategies for choosing the best set of parameters for these functions.

We report on computational experiments conducted on instances from the literature. For many cases, it is important to emphasize that our procedures provided the optimal solution value. For many instances, we improved the results reported in the literature.

References:

[1] D. Bein, W. Bein, S. Phoha, Efficient Data Centers, Cloud Computing in the Future of Distributed Computing, *Seventh International Conference on Information Technology*, 2010, pp. 70-75.

[2] J. Blanchette, A Material History of Bits, *Journal of the American Society for Information Science and Technology*, Vol. 62, 2011, pp. 1042-1057.

[3] C. Burdett, E. Johnson, A Subadditive Approach to Solve Linear Integer Programs, *Annals of Discrete Mathematics*, Vol. 1, 1977, pp. 117-144.

[4] J. Carlier, E. Néron, Computing redundant resources for the resource constrained project scheduling problem, *European Journal of Operational Research*, Vol. 176, No. 3, 2007, pp. 1452-1463.

[5] F. Clautiaux, C. Alves, J. V. de Carvalho, A survey of dual-feasible functions for bin packing problems, *Annals of Operations Research*, Vol. 179, 2010, pp. 317-342.

[6] S. Dash, O. Gunluk, Valid Inequalities Based on Simple Mixed-Integer Sets, *Mathematical Programming*, Vol. 105, 2006, pp. 29-53.

[7] S. Fekete, J. Schepers, New Classes of Fast Lower Bounds for Bin Packing Problems, *Mathematical Programming*, Vol. 91, 2001, pp. 11-31.

[8] E. Feller, L. Rilling, C. Morin, Energy-Aware Ant Colony Based Workload Placement in Clouds, Technical Report, INRIA, 2011.

[9] D. Johnson, *Near Optimal Bin Packing Algorithms*, Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, 1973.

[10] J. Kaplan, W. Forrest, N. Kindler, Revolutionizing data center energy efficiency, Technical Report, McKinsey and Company, 2008.

[11] X. Kong, C. Lin, Y. Jiang, W. Yan, X. Chu. Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction, *Journal of Network and Computer Applications*, Vol. 34, 2011, pp. 1068-1077.

[12] B. Li, J. Li, J. Huai, T. Wo, Q. Li, L. Zhong, EnaCloud: An Energy Saving Application Live Placement Approach for Cloud Computing Environments, *IEEE International Conference on Cloud Computing*, 2009, pp. 17-24.

[13] A. Lodi, S. Martello, M. Monaci, Two-Dimensional Packing Problems: A Survey,

European Journal of Operational Research, Vol. 141, No. 2, 2022, pp. 241-252.

[14] S. Martello, P. Toth, *Knapsack problems - Algorithms and Computer Implementation*, Wiley, Chichester, 1990.

[15] S. Martello, D. Vigo, Exact Solution of the Two-Dimensional Finite Bin Packing Problem, *Management Science*, Vol. 44, No. 3, 1998, pp. 388-399.

[16] C. Mastroianni, M. Meo, G. Papuzzo, Self Economy in Cloud Data Centers: Satistical Assignmet and Migration of Virtual Machines, *Lecture Notes in Computer Science*, Vol.6852, 2011, pp. 405-416.

[17] R. Nielsen, C. Iversen, P. Bonnet, Private Cloud Configuration with MetaConfig, *Proceedings for IEEE 4th International Conference on Cloud Computing (CLOUD 2011)*, 2011.

[18] J. Tordsson, R. Montero, R. Moreno-Vozmediano, I. Llorente, Cloud Brokering Mechanisms for Optimized Placement of Virtual Machines across Multiple Providers, *Future Generation of Computer Systems*, Vol. 28, 2012, pp. 358-367.

[19] W. Vogels, Beyond server consolidation, *Queue*, Vol. 6, 2008, pp. 20-26.

[20] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, *Journal of Internet Services and Applications*, Vol. 1, 2010, pp. 7-18.

	CLASS 1					CLASS 2					CLASS 3					CLASS 4					CLASS 5				
m	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**
20	6,5	7,2	8	8	8	0,7	0,7	1	2	2	4,3	5,0	6	9	9	0,7	0,7	1	3	3	5,4	6,8	8	11	11
	4,3	4,5	5	8	8	0,5	0,5	1	2	2	2,6	2,8	4	7	7	0,4	0,4	1	3	3	3,2	3,6	5	9	9
	6,5	7,9	9	10	10	0,7	0,7	1	2	2	4,4	5,0	6	10	10	0,7	0,7	1	4	4	5,5	6,4	7	13	13
	4,7	4,8	6	7	7	0,5	0,5	1	2	2	3,1	3,3	4	9	9	0,5	0,5	1	3	3	3,9	4,2	5	9	9
	5,3	5,4	6	8	8	0,6	0,6	1	2	2	3,5	3,7	5	7	7	0,6	0,6	1	3	3	4,4	4,6	6	8	9
	7,4	8,5	9	11	11	0,8	0,8	1	2	2	4,8	5,9	7	9	9	0,8	0,8	1	3	4	6,0	8,0	9	11	11
	5,3	5,5	6	8	8	0,6	0,6	1	2	2	3,4	4,0	5	8	8	0,5	0,5	1	3	3	4,3	4,8	6	10	10
	5,1	5,9	6	7	7	0,6	0,6	1	2	2	3,2	3,7	5	8	8	0,5	0,5	1	3	3	4,0	4,7	5	9	9
	6,2	7,3	8	10	10	0,7	0,7	1	2	2	4,2	4,6	5	9	9	0,7	0,7	1	4	4	5,3	6,8	7	12	12
	6,6	7,7	8	9	9	0,7	0,7	1	2	2	4,6	6,3	7	9	9	0,7	0,7	1	3	3	5,8	7,0	8	10	10
40	9,0	9,1	10	13	13	1,0	1,0	2	3	3	5,6	5,7	7	12	12	0,9	0,9	2	4	4	7,0	7,2	8	16	16
	10,7	11,0	12	14	14	1,2	1,2	2	2	2	7,1	7,5	8	15	15	1,1	1,1	2	5	5	8,9	9,7	10	19	19
	13,9	16,0	16	19	19	1,5	1,5	2	3	3	9,1	10,5	11	14	15	1,5	1,5	2	5	5	11,5	14,3	15	19	20
	12,5	14,0	15	16	16	1,4	1,4	2	3	3	8,2	9,4	10	15	15	1,3	1,3	2	5	5	10,4	11,8	13	18	18
	13,8	14,5	15	17	17	1,5	1,5	2	3	3	9,4	11,5	12	17	17	1,5	1,5	2	5	5	11,9	13,0	14	20	20
	10,6	13,3	14	15	15	1,2	1,2	2	3	3	7,0	9,5	10	15	15	1,1	1,1	2	5	5	8,8	12,0	12	22	22
	10,7	10,7	12	14	14	1,2	1,2	2	3	3	7,3	8,0	8	14	14	1,2	1,2	2	5	5	9,1	10,0	10	20	20
	14,5	18,0	19	21	21	1,6	1,6	2	3	3	9,9	12,8	13	19	19	1,6	1,6	2	5	5	12,5	17,0	17	25	25
	10,1	10,2	11	14	14	1,1	1,1	2	2	2	6,7	6,7	8	14	14	1,1	1,1	2	5	5	8,4	8,8	10	16	16
	10,3	10,6	12	14	14	1,1	1,1	2	3	3	6,7	6,8	8	12	12	1,1	1,1	2	4	4	8,5	9,6	11	18	18
60	20,1	22,1	23	26	26	2,2	2,2	3	4	4	13,5	15,8	16	23	23	2,2	2,2	3	6	6	17,1	19,5	20	31	31
	17,7	18,1	19	23	23	2,0	2,0	3	4	4	11,4	12,0	13	18	19	1,8	1,8	2	6	6	14,4	15,8	17	27	28
	18,2	21,0	21	24	24	2,0	2,0	3	3	3	12,3	12,8	14	20	20	2,0	2,0	3	6	6	15,5	18,5	19	30	30
	18,2	21,8	22	25	25	2,0	2,0	3	3	3	12,3	14,2	15	22	22	2,0	2,0	3	6	6	15,5	19,3	20	33	33
	16,5	18,1	19	21	21	1,8	1,8	2	3	3	10,9	11,1	12	18	18	1,7	1,7	2	5	5	13,7	14,6	15	25	25
	16,6	16,8	18	21	21	1,8	1,8	2	3	3	11,1	11,3	12	18	18	1,8	1,8	2	6	6	13,9	14,9	16	24	24
	14,8	15,0	16	18	18	1,6	1,6	2	3	3	10,0	10,8	12	18	18	1,6	1,6	2	6	6	12,5	13,8	14	23	24
	18,8	20,7	21	24	24	2,1	2,1	3	3	3	12,8	14,3	15	20	20	2,0	2,0	3	6	6	16,1	18,3	19	29	29
	17,4	17,4	18	21	21	1,9	1,9	3	3	3	11,6	12,5	13	18	19	1,9	1,9	3	6	6	14,5	15,8	16	29	30
	21,1	23,5	24	27	27	2,3	2,3	3	4	4	14,6	17,0	18	28	28	2,3	2,3	3	6	6	18,4	23,0	23	35	35
80	22,7	24,1	25	27	27	2,5	2,5	3	4	4	15,2	16,7	17	27	27	2,4	2,4	3	7	7	19,0	22,0	22	39	39
	24,1	25,4	26	28	28	2,7	2,7	3	4	4	15,9	17,4	19	26	26	2,5	2,5	3	7	7	20,1	22,1	23	39	39
	23,0	26,8	27	30	30	2,6	2,6	3	3	3	15,5	17,3	18	26	26	2,5	2,5	3	7	7	19,5	23,8	25	40	40
	24,4	26,8	27	30	30	2,7	2,7	3	4	4	16,3	17,6	18	25	25	2,6	2,6	3	6	6	20,4	24,3	25	38	39
	23,5	25,3	26	29	29	2,6	2,6	3	4	4	15,6	15,8	17	26	26	2,5	2,5	3	7	7	19,6	22,0	23	37	37
	25,2	27,6	28	31	31	2,8	2,8	3	4	4	17,2	19,8	20	27	27	2,7	2,7	3	6	6	21,6	25,0	25	39	39
	26,6	30,8	31	35	35	3,0	3,0	4	4	4	18,1	19,8	21	29	29	2,9	2,9	4	7	7	22,8	26,8	27	39	39
	26,4	28,6	29	33	33	2,9	2,9	4	4	4	18,0	21,5	22	30	30	2,9	2,9	4	6	6	22,9	25,9	26	40	40
	27,9	29,6	30	33	33	3,1	3,1	4	4	4	18,6	21,0	22	27	27	3,0	3,0	4	7	7	23,5	25,7	27	42	42
	24,3	25,3	26	28	29	2,7	2,7	3	4	4	16,2	18,0	18	24	25	2,6	2,6	3	6	6	20,4	22,2	23	34	35
100	27,3	27,5	28	30	31	3,0	3,0	4	4	4	18,0	18,1	19	26	26	2,9	2,9	4	7	7	22,6	23,0	24	40	41
	30,4	30,6	32	35	35	3,4	3,4	4	5	5	20,8	21,6	23	30	30	3,3	3,3	4	7	7	26,2	27,7	29	43	43
	26,7	28,5	29	32	32	3,0	3,0	4	4	4	17,6	18,0	19	28	28	2,8	2,8	3	7	7	22,2	23,1	24	39	40
	28,8	29,9	31	34	34	3,2	3,2	4	5	5	19,1	19,1	20	27	28	3,1	3,1	4	7	7	24,0	25,5	26	42	42
	30,2	31,5	32	35	35	3,4	3,4	4	5	5	20,2	21,5	22	30	30	3,2	3,2	4	7	7	25,4	27,3	28	43	44
	33,8	36,3	37	40	40	3,8	3,8	4	5	5	23,0	26,0	27	37	37	3,7	3,7	4	7	7	29,0	33,3	34	50	50
	27,4	27,6	29	30	30	3,0	3,0	4	4	4	18,4	19,3	20	28	28	2,9	2,9	4	7	7	23,1	24,4	25	43	43
	31,2	32,7	34	37	37	3,5	3,5	4	5	5	20,9	22,3	23	30	30	3,3	3,3	4	7	7	26,3	28,7	30	42	42
	29,9	30,7	31	34	34	3,3	3,3	4	5	5	19,7	21,3	22	30	30	3,2	3,2	4	7	7	24,9	26,6	27	41	42
	34,2	37,6	38	41	41	3,8	3,8	4	5	5	23,2	28,5	29	35	35	3,7	3,7	4	7	7	29,3	34,8	35	49	49

Table 1: Comparison of the bounds for the different cases (part 1)

	CLASS 6					CLASS 7					CLASS 8					CLASS 9					CLASS 10				
m	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**	z_M	B_{NG}	B_{in}	B_{ex}	**WFD**
20	0,6	0,6	1	4	4	4,4	4,7	5	13	13	4,8	5,8	6	6	6	11,1	18,5	19	19	19	5,0	6,0	7	8	8
	0,4	0,4	1	3	3	3,8	4,6	5	15	15	4,6	6,0	7	11	11	7,6	12,5	13	17	17	2,4	2,5	4	9	9
	0,6	0,6	1	4	4	3,9	4,1	5	12	12	4,4	4,7	6	7	7	9,2	13,5	14	16	16	3,5	3,8	5	8	8
	0,4	0,4	1	3	3	5,1	6,4	7	15	15	5,0	6,2	7	9	9	8,8	16,0	16	18	18	3,4	4,0	5	8	8
	0,5	0,5	1	3	3	4,6	5,4	6	14	14	4,2	5,0	6	6	6	9,9	16,0	16	16	16	3,2	3,6	4	6	7
	0,7	0,7	1	4	4	4,4	4,8	6	13	13	4,7	5,4	6	7	8	9,8	14,0	14	16	16	3,0	3,1	4	7	7
	0,5	0,5	1	3	3	3,3	3,6	4	13	13	3,4	3,9	5	6	7	6,5	9,0	9	11	11	3,9	4,4	5	9	9
	0,4	0,4	1	3	3	4,7	5,7	7	16	16	4,2	4,8	5	7	7	8,4	14,0	14	15	15	1,9	2,0	3	6	6
	0,6	0,6	1	4	4	4,8	5,5	6	12	12	4,9	5,4	7	9	9	10,2	15,0	15	17	17	4,4	4,7	5	9	9
	0,6	0,6	1	3	3	3,4	3,8	5	12	12	3,1	3,6	4	5	5	7,7	12,5	13	13	13	2,0	2,0	3	6	6
40	0,8	0,8	1	5	5	8,4	9,3	10	23	23	9,5	10,5	12	19	19	16,2	24,0	25	31	31	7,0	7,5	8	18	18
	1,0	1,0	2	6	6	10,3	11,9	13	27	27	10,6	12,9	13	16	16	18,5	31,5	32	33	33	6,6	6,8	8	14	14
	1,3	1,3	2	6	6	8,3	9,0	10	23	23	8,9	10,1	11	14	14	18,5	28,0	29	32	32	8,0	8,6	9	16	16
	1,2	1,2	2	6	6	11,2	13,4	14	26	26	10,0	11,7	12	14	14	18,7	31,0	31	32	32	5,5	5,5	7	11	11
	1,3	1,3	2	6	6	7,9	9,0	10	20	20	7,3	8,6	9	11	11	16,6	26,5	27	29	29	5,1	5,3	6	12	12
	1,0	1,0	2	6	6	9,6	10,6	12	24	24	9,5	11,4	12	16	16	18,4	29,0	29	34	34	5,0	5,0	6	13	13
	1,0	1,0	2	6	6	9,1	10,8	12	25	25	8,9	10,3	11	15	15	15,8	23,5	24	28	28	6,1	6,6	7	13	14
	1,4	1,4	2	6	6	9,6	10,8	12	22	22	9,2	10,2	11	16	17	18,0	25,5	26	33	33	6,1	6,4	7	16	16
	0,9	0,9	2	6	6	6,9	7,9	9	23	23	7,7	8,1	9	15	15	14,9	21,0	21	27	27	6,8	7,5	8	17	17
	0,9	0,9	2	6	6	10,2	12,5	13	28	28	10,5	12,1	13	15	15	20,1	33,0	34	37	37	7,2	8,3	9	15	15
60	1,9	1,9	3	9	9	14,7	16,5	17	36	36	14,6	16,5	17	20	20	29,3	45,5	46	48	48	10,3	10,6	12	19	19
	1,6	1,6	2	8	8	12,4	13,2	14	30	30	14,4	16,2	17	21	21	29,3	44,0	45	49	49	10,9	11,6	12	21	21
	1,7	1,7	2	9	9	14,1	16,4	17	33	33	14,0	15,5	17	21	22	29,7	46,0	46	50	50	10,3	10,4	12	19	19
	1,7	1,7	2	9	9	12,8	14,1	15	30	30	12,5	14,7	15	21	21	26,2	44,0	44	51	51	6,9	6,9	8	18	19
	1,5	1,5	2	7	7	12,1	13,8	15	33	33	12,2	13,5	15	22	22	24,7	40,5	41	46	46	7,2	7,2	8	17	18
	1,5	1,5	2	8	8	12,1	14,6	15	32	32	12,9	14,3	15	21	22	24,2	37,0	37	41	41	11,1	12,5	13	24	24
	1,4	1,4	2	7	7	12,4	14,4	15	30	30	12,1	13,2	14	20	21	24,7	40,0	41	47	47	9,0	9,0	10	19	20
	1,8	1,8	2	9	9	14,5	16,6	17	32	32	14,7	17,0	17	24	24	27,9	46,5	47	49	49	9,0	9,2	10	21	21
	1,6	1,6	2	9	9	12,7	13,8	15	32	32	13,7	15,7	17	20	20	27,0	44,5	45	47	47	7,9	7,9	9	17	17
	2,0	2,0	3	10	10	16,2	17,8	19	36	36	15,0	17,6	18	21	21	28,9	45,0	45	46	46	7,3	7,5	9	15	15
80	2,1	2,1	3	10	10	17,9	20,0	21	38	38	18,4	21,2	22	27	28	35,1	58,0	59	61	61	11,7	11,8	13	23	24
	2,2	2,2	3	9	10	20,9	24,1	25	42	42	19,4	23,1	24	27	27	36,0	58,0	58	62	62	9,8	9,8	11	19	19
	2,2	2,2	3	10	10	17,7	20,0	21	38	38	17,3	19,7	20	27	27	35,1	57,0	57	65	65	10,1	10,1	11	22	22
	2,3	2,3	3	10	10	18,4	20,9	22	37	37	17,2	19,2	20	28	29	35,6	52,0	53	60	60	12,9	13,2	14	25	25
	2,2	2,2	3	11	11	19,7	22,7	24	42	42	21,9	25,5	26	31	31	38,7	61,0	62	67	67	12,7	13,1	14	25	26
	2,4	2,4	3	11	11	19,2	22,0	23	42	42	18,7	21,6	22	28	28	37,3	61,5	62	63	63	12,2	12,4	13	25	25
	2,5	2,5	3	10	10	20,6	23,4	24	44	44	18,9	21,4	22	28	28	38,5	59,0	59	63	63	13,4	13,6	15	26	26
	2,5	2,5	3	10	10	18,9	21,7	23	38	38	19,4	21,3	23	29	29	36,9	57,5	58	61	61	9,4	9,4	10	22	22
	2,6	2,6	3	11	11	19,7	22,9	24	42	42	18,8	20,5	22	29	29	33,2	48,5	49	56	56	11,6	11,6	13	23	23
	2,3	2,3	3	9	9	20,4	23,5	24	43	43	20,8	23,6	24	30	30	39,0	59,5	60	65	65	13,4	13,9	15	26	26
100	2,5	2,5	3	11	11	23,6	26,6	27	44	44	22,7	25,9	27	32	32	45,4	71,0	71	77	77	13,6	13,6	15	24	24
	2,9	2,9	4	11	11	23,2	26,3	27	46	46	22,8	26,4	27	33	34	43,1	63,5	64	69	69	14,6	14,6	16	28	29
	2,5	2,5	3	10	11	20,8	23,9	25	44	44	20,6	23,4	24	31	31	41,8	68,0	68	72	72	14,9	14,9	16	28	29
	2,7	2,7	3	12	12	22,6	25,7	27	46	46	25,5	29,9	31	35	36	47,5	77,5	78	82	82	16,2	16,6	18	30	31
	2,8	2,8	3	11	11	21,5	24,2	25	44	44	24,1	28,5	29	37	37	41,3	64,5	65	72	72	16,5	16,7	18	32	32
	3,2	3,2	4	13	13	24,8	27,6	28	45	45	22,6	26,0	27	32	32	45,9	70,5	71	74	74	12,2	12,2	13	23	23
	2,6	2,6	3	12	12	22,8	26,3	27	43	43	22,6	25,0	26	36	36	41,6	65,5	66	71	71	12,9	12,9	14	28	28
	2,9	2,9	4	10	10	24,4	28,1	29	45	45	24,5	27,6	29	35	35	46,6	73,0	74	80	80	17,0	17,5	18	31	32
	2,8	2,8	3	11	11	22,6	24,4	25	43	43	23,4	26,2	27	35	35	43,5	65,0	66	71	71	15,1	15,1	16	30	30
	3,3	3,3	4	12	12	27,2	30,2	31	50	50	27,4	31,6	32	38	39	48,6	71,5	72	78	78	14,1	14,1	15	25	25

Table 2: Comparison of the bounds for the different cases (part 2)