The Bin Packing Problem with Precedence Constraints

Mauro Dell’Amico  José Carlos Díaz Díaz  Manuel Iori

Department of Sciences and Methods for Engineering  
University of Modena and Reggio Emilia
Outline

1 Introduction
2 The Maths’ Contribution
   • Simple Lower Bounds
   • Better Lower Bounds
   • Further Improvements on the Lower Bounds
3 The Heuristic Contribution
   • Variable Neighborhood Search
4 The Integration
   • Branch-and-Bound
5 Computational Results
   • Lower and Upper Bounds
   • Branch-and-Bound
6 Conclusions
"Bin Packing Problem" (BPP)

Given \( n \) items of weight \( w_j \) \((j = 1, 2, \ldots, n)\) and \( m \) identical bins of capacity \( C \), find the minimum number of bins that can contain all the items.
The Problem

**Bin Packing Problem (BPP)**
Given $n$ items of weight $w_j$ ($j = 1, 2, \ldots, n$) and $m$ identical bins of capacity $C$, find the minimum number of bins that can contain all the items.

**Bin Packing Problem with Precedence Constraints (BPP-P)**
Given an additional set of precedences among items, pack the items in the minimum number of bins, with the additional constraint that bins can be ordered in such a way that all the precedences be satisfied.
A Simple Example

Precedence constraints impose that only 1-2, 2-4 or 3-4 can be packed together.

The size of the items impose that 1 and 2 are packed in separate bins.
Seminal paper: Garey, Graham, Johnson and Yao (1976) considered a generalization of the BPP-P where several kind of resources are considered (i.e., the bin is multidimensional). They studied the performance of greedy heuristics.

A related problem is the Simple Assembly Line Balancing Problem (SALBP) which asks to assign the tasks of an assembly line to workstations of given capacity in order to minimize the number of workstations used. The SALBP can be modeled as a BPP with loose precedences: $i$ precedes $j$ implies that $i$ must be packed before $j$ or in the same bin with $j$ (see Becker and Scholl 2006). Our strict precedence constraints imply that the SALBP methods cannot be used.
Seminal paper: Garey, Graham, Johnson and Yao (1976) considered a generalization of the BPP-P where several kind of resources are considered (i.e., the bin is multidimensional). They studied the performance of greedy heuristics.

A related problem is the *Simple Assembly Line Balancing Problem* (SALBP) which asks to assign the tasks of an assembly line to workstations of given capacity in order to minimize the number of workstations used.
Seminal paper: Garey, Graham, Johnson and Yao (1976) considered a generalization of the BPP-P where several kind of resources are considered (i.e., the bin is multidimensional). They studied the performance of greedy heuristics.

A related problem is the *Simple Assembly Line Balancing Problem* (SALBP) which asks to assign the tasks of an assembly line to workstations of given capacity in order to minimize the number of workstations used.

The SALBP can be modeled as a BPP with loose precedences: \( i \) precedes \( j \) implies that \( i \) must be packed *before* \( j \) or *in the same bin* with \( j \) (see Becker and Scholl 2006). Our *strict* precedence constraints imply that the SALBP methods cannot be used.
We start with an input graph $G_0 = (V_0, A_0)$
We start with an input graph $G_0 = (V_0, A_0)$

We work on a precedence graph $G = (V, A)$ in which we:
- add vertex 0 and connect it to vertices with 0 indegree
- add vertex $n + 1$ and connect vertices with 0 outdegree to it
- apply pre-processing to remove items or at least lift their weights
**Notation**

\[ c_{ij} \quad \text{number of arcs in the longest path from } i \text{ to } j \text{ in } G \]
(note that \( G \) is **acyclic**)

\[ \text{head}_j = c_{0j} \quad \text{max number of arcs in a path terminating in node } j \]

\[ \text{tail}_j = c_{j,n+1} \quad \text{max number of arcs in a path starting in } j \]

\[ \text{head}_j = 2 : \text{ item } j \text{ can be packed in bins } 2, 3, ... \]

\[ \text{tail}_j = 4 : \text{ item } j \text{ can be packed in bins } m - 3, m - 4, ... \]

\((m = \text{last bin used in a feasible solution})\)
Lower Bounds from the BPP

Immediate bound: remove the precedence constraints, solve the resulting BPP (or compute a lower bound for it)
Lower Bounds from the BPP

Immediate bound: remove the precedence constraints, solve the resulting BPP (or compute a lower bound for it)

**Proposition:** The optimal BPP solution value is arbitrarily smaller than the corresponding optimal BPP-P value

Enough to consider a BPP-P instance with \( n \) items of small weight forming a single chain from item 1 to item \( n \)
Lower Bounds from the BPP

Immediate bound: remove the precedence constraints, solve the resulting BPP (or compute a lower bound for it)

**Proposition:** The optimal BPP solution value is arbitrarily smaller than the corresponding optimal BPP-P value

Enough to consider a BPP-P instance with $n$ items of small weight forming a single chain from item 1 to item $n$

Nevertheless bounds from the BPP may be useful in practice, we use

**Lower bound 1:**

$$L_1 = \left\lceil \frac{\sum_{j=1}^{n} w_j}{C} \right\rceil$$
Lower bounds from CPM

Further bounds can be obtained by taking into account only the precedence constraints

**Lower bound 2:** Compute, through the *Critical Path Method* (CPM) algorithm, the *longest path* from node 0 to node $n + 1$ in $G$. We obtain

$$L_2 = c_{0,n+1} - 1$$
Lower bounds from CPM

Further bounds can be obtained by taking into account only the precedence constraints

**Lower bound 2:** Compute, through the *Critical Path Method* (CPM) algorithm, the *longest path* from node 0 to node \( n + 1 \) in \( G \). We obtain

\[
L_2 = c_{0,n+1} - 1
\]

**Proposition:** \( L_2 \) has an arbitrarily bad worst case performance

Enough to consider a BPP-P instance with no precedences at all
Lower bound 3:

\[ L_3 = \max\{L_1, L_2\} \]
Lower bound 3:

\[ L_3 = \max\{L_1, L_2\} \]

Proposition: The worst case performance ratio of \( L_3 \) is \( \frac{1}{3} \) and is tight

Sketch of the proof:

- Build a heuristic solution \( H \) assuming infinite bin capacity
- Solution \( H \) uses exactly \( L_2 \) bins, each bin containing items with no precedences among them
- Consider in turn each bin \( i \in H \) and replace it with a number of bins sufficient for a feasible packing of all its items (pure BPP)
- The number of bins added to \( H \) is \( \ell \leq 2L_1 \)
- The resulting feasible solution uses at most \( L_2 + 2L_1 \leq 3L_3 \) bins
Lower bound 3

To see that the $\frac{1}{3}$ performance is tight, let $n = 3k$ and

$$w_j = \begin{cases} \epsilon & j = 1, \ldots, k \\ \frac{C}{2} + \epsilon & j = k + 1, \ldots, n \end{cases}$$

Precedences

Optimal solution: 3$k$ bins

\[
L_1 = \left\lceil \frac{k \epsilon + 2k(\frac{C}{2} + \epsilon)}{C} \right\rceil = k + \left\lceil \frac{3k \epsilon}{C} \right\rceil = k + 1
\]

\[
L_2 = k + 1 \Rightarrow L_3 = k + 1
\]
Lower bound 4

Derived from (Gendreau, Laporte and Semet, 2004) for BPP with conflicts. Pack each item in a longest path in a separate bin. Fill these bins with (fractions of) other items. Compute a lower bound on what is left.
Lower bound 4

Derived from (Gendreau, Laporte and Semet, 2004) for BPP with conflicts. Pack each item in a longest path in a separate bin. Fill these bins with (fractions of) other items. Compute a lower bound on what is left

- Let $P$ denote the items in a longest path of $G$
- Define a bipartite graph $\bar{G} = (Q, P, \bar{A})$ with $Q = V_0 \setminus P$
- $\bar{A} = \{(i, j) : i \in Q, j \in P, i$ and $j$ can be packed in the same bin$\}$
- Compute a max-flow from $Q$ to $P$ and get a value $f$
Lower bound 4

Derived from (Gendreau, Laporte and Semet, 2004) for BPP with conflicts. Pack each item in a longest path in a separate bin. Fill these bins with (fractions of) other items. Compute a lower bound on what is left

- Let $P$ denote the items in a longest path of $G$
- Define a bipartite graph $\bar{G} = (Q, P, \bar{A})$ with $Q = V_0 \setminus P$
- $\bar{A} = \{(i,j): i \in Q, j \in P, i$ and $j$ can be packed in the same bin$\}$
- Compute a max-flow from $Q$ to $P$ and get a value $f$

**Lower bound 4:**

$$L_4 = L_2 + \left\lceil \sum_{j \in Q} w_j - f \right\rceil$$

**Proposition:** Lower bound $L_4$ dominates $L_3$
Lower bound 4

**Proposition:** The worst case performance ratio of $L_4$ is $\frac{1}{2}$ and is tight.
Proposition: The worst case performance ratio of $L_4$ is $\frac{1}{2}$ and is tight.

Sketch of the proof:

1. Let $F \subseteq Q$ denote the set of items used by flow $f$.
2. *Items in $P \cup F$ can be packed in a solution $H$ using at most $2L_2$ bins.*
3. Let $R = Q \setminus F$ denote the set of remaining items, pack them into $\ell$ separate bins.
   - $\ell \leq \frac{2}{C} \sum_{j \in R} w_j$
   - $\sum_{j \in R} w_j \leq \sum_{j \in Q} w_j - f$
4. Hence
   $$z_H = 2L_2 + \ell \leq 2 \left( L_2 + \frac{\sum_{j \in R} w_j}{C} \right) \leq 2 \left( L_2 + \frac{(\sum_{j \in Q} w_j - f)}{C} \right) \leq 2L_4$$
Proposition: The worst case performance ratio of $L_4$ is $\frac{1}{2}$ and is tight

Sketch of the proof:

- Let $F \subseteq Q$ denote the set of items used by flow $f$
- *Items in $P \cup F$ can be packed in a solution $H$ using at most $2L_2$ bins*
- Let $R = Q \setminus F$ denote the set of remaining items, pack them into $\ell$ separate bins
  - $\ell \leq \frac{2}{C} \sum_{j \in R} w_j$
  - $\sum_{j \in R} w_j \leq \sum_{j \in Q} w_j - f$
- Hence
  $$z_H = 2L_2 + \ell \leq 2 \left( L_2 + \frac{\sum_{j \in R} w_j}{C} \right) \leq 2 \left( L_2 + \frac{(\sum_{j \in Q} w_j - f)}{C} \right) \leq 2L_4$$

To prove the tightness: consider an instance with no precedence and $n$ items, each of weight $\frac{C}{2} + \varepsilon$: $L_4 = L_1 = \frac{n}{2} + 1$, $z_{opt} = n$
Lower bound 5

Let $S^q_r \subseteq V_0$ be a set of items such that, for each $j \in S^q_r$:

- $\text{head}_j \geq r$
- $\text{tail}_j \geq q$

$$L_5 = \max \{ r, q : r + q \leq c_0, n + 1 \}$$

where $L$ is any valid lower bound.
Lower bound 5

Let $S_r^q \subseteq V_0$ be a set of items such that, for each $j \in S_r^q$:

- $\text{head}_j \geq r$
- $\text{tail}_j \geq q$

Lower bound 5:

$$L_5 = \max_{r,q:r+q \leq c_0.n+1} \{(r - 1) + L(S_r^q) + (q - 1)\}$$

where $L$ is any valid lower bound
Any lower bound for the BPP can be improved using dual feasible functions (DFFs). In practice, DFFs are (heuristic) attempts to modify items weights so as to lift the lower bound while ensuring optimality.
Improvements based on the BPP Component

Any lower bound for the BPP can be improved using *dual feasible functions* (DFFs). In practice DFFs are (heuristic) attempts to modify items weights so as to lift the lower bound while ensuring optimality.

We improve $L_1$ using

- $L_{FS}$: (Fekete and Schepers, 2001)
- $L_{diff}$: new combination of classical DFFs that we obtained on the basis of considerations by
  - (Fekete and Schepers, 2001)
  - (Boschetti and Mingozzi, 2003)
  - (Haouari and Gharbi, 2005)
  - (Crainic, Perboli, Pezzuto and Tadei, 2007)
Improvements based on the Precedence Constraints

Precedence constraints can be tightened as follows:

\begin{itemize}
\item \textbf{for each} pair of items \( j \) and \( k \) \textbf{do}
  \begin{itemize}
  \item compute \( \text{Sep}(j, k) \), set of items in all paths from \( j \) to \( k \)
  \item modify \( G \) by adding arc \((i, j)\) with weight \( \omega_{ij} = L(\text{Sep}(i, k)) + 1 \), where \( L \) is a valid lower bound
  \end{itemize}
\end{itemize}

\textbf{endfor}

Compute the new longest paths between each pair of vertices. Store the paths in matrix \( c' \) (\textit{enhanced graph})
Heuristic and Local Search

**Greedy** (Garey, Graham, Johnson and Yao, 1976)
- order items $j$ by increasing $head_j$ breaking ties by decreasing $w_j$
- apply First Fit
Heuristic and Local Search

Greedy (Garey, Graham, Johnson and Yao, 1976)
- order items $j$ by increasing $\text{head}_j$; breaking ties by decreasing $w_j$
- apply First Fit

Local Search
- MOVE: item $j$ packed in bin $a$ is moved to bin $b$
- SWAP: item $j$ packed in $a$ and item $k$ packed in $b$ are interchanged
Perform moves or swaps if this leads to more unbalanced loads
Variable Neighborhood Search (VNS)

Given $k \in \{1, \ldots, n\}$ let $\mathcal{N}_k$ be a neighborhood defined as follows:

- randomly select $k$ items and remove them from the solution
- reinsert one at a time the items in a feasible bin different from the original one (if no such bin exists, initialize a new one)
Variable Neighborhood Search (VNS)

Given \( k \in \{1, \ldots, n\} \) let \( \mathcal{N}_k \) be a neighborhood defined as follows:

- randomly select \( k \) items and remove them from the solution
- reinsert one at a time the items in a feasible bin different from the original one (if no such bin exists, initialize a new one)

**VNS**

generate a solution \( S^* \) by Greedy and Local Search, set \( k := 1 \);

for \( i := 1 \) to \( n_{iter} \) do

randomly select \( S' \in \mathcal{N}_k(S^*) \) and apply Local Search;

if \( z(S') < z(S^*) \) then set \( S^* := S' \), \( k := 1 \);

else set \( k := k \mod k_{max} + 1 \);

end for
Branch-and-Bound (B&B)

1. Compute the best lower bound $LB$ using all the previous bounds
2. Compute the upper bound $UB$ by means of procedure VNS
3. If $LB=UB$ then stop
   else
      - explore a branch-decision-tree in depth-first
      - at each node:
         - compute a new heuristic solution
         - compute a valid lower bound
Branching rule

We scan the heuristic solution associated to the node and select two items \( j \) and \( k \) and branch as follows:

1. **First branch**: we impose that \( j \) and \( k \) are packed in the same bin by replacing them with a unique item of weight \( w_j + w_k \) and having all precedences of \( j \) and \( k \)

2. **Second branch**: we impose a precedence from \( j \) to \( k \)

3. **Third branch**: we impose a precedence from \( k \) to \( j \)
Algorithms tested on the 269 established benchmark for the *Simple Assembly Line Balancing Problem* considering strict precedences (www.assembly-line-balancing.de)

- Very heterogeneous set: $n$ varies from 7 to 297 and $C$ from 6 to 17067
- Procedures coded in C++ and run them on an INTEL® Core 2 Duo at 3.00 Ghz
- VNS halted after 400 CPU seconds, B&B after 2 hours
Scholl Precedence Graph
### Lower Bounds

- $22.60$
- $7.17$
- $0.95$
- $1.05$
- $2.33$
- $0.95$
- $1.05$
- $1.34$
- $1.67$
- $0.65$
- $0.67$
- $0.89$
- $0.63$
- $0.64$
- $0.89$
- $0.63$
- $0.64$

Dell'Amico, Díaz Díaz, Iori (DISMI)
VNS and LBs provide 235 optimal solutions out of 269 instances, the remaining 34 ones are given to the B&B
Best Settings

We computed a series of preliminary tests running the B&B for 1200 CPU seconds under different configuration.

Lower bounds

- Tested in details:
  1. $L_4$
  2. $L_5(L_1)$
  3. $L_5^{\text{diff}}(L_1)$

- $L_5(L_1)$ slightly better than $L_4$ and much better than $L_5^{\text{diff}}(L_1)$
Best Settings

We computed a series of preliminary tests running the B&B for 1200 CPU seconds under different configuration.

Upper bounds
- Tested in details:
  1. Greedy
  2. Greedy + Local Search
  3. Greedy + Local Search + VNS (0.01 CPU seconds)
  4. Greedy + Local Search + VNS (larger times)
- Option 2 much better than 1 and 4 and slightly better than 3
Best Settings

We computed a series of preliminary tests running the B&B for 1200 CPU seconds under different configuration.

Choice of the couple of items

- We choose two items $j$ and $k$ packed in the same bin in the current heuristic solution
  1. select the first bin containing at least two items, then choose the first two items in it
  2. choose the pair of items having the largest total weight
  3. choose the item $j$ having largest degree in the current graph, and then the item $k$ having the smallest degree
- Option 3 much better than 2 and slightly better than 1
- Effect of the heuristic component is important
Summary of the B&B Computational Results

On the set of 34 challenging instances, the B&B

- Improves lower bound value 11 times
- Improves upper bound value 7 times
- Proves the optimality of 18 instances
- Achieves an average gap of 1.74%
- Leads to a largest difference of two bins between $L$ and $U$
Conclusions

- The two BPP-P components (BPP and precedence) make the problem very difficult to be solved.
Conclusions

- The two BPP-P components (BPP and precedence) make the problem very difficult to be solved.
- Our combination of exact and heuristic algorithms solve 253 instances out of 269 ones.
- The heuristics play a crucial role in:
  - proving the optimality for the majority of the instances at the root node
  - fathoming nodes in the branch-and-bound
  - allowing an effective branching rule in the branch-and-bound

Instances, precedence graphs, detailed computational results are available at www.or.unimore.it
Conclusions

- The two BPP-P components (BPP and precedence) make the problem very difficult to be solved.
- Our combination of exact and heuristic algorithms solve 253 instances out of 269 ones.
- The heuristics play a crucial role in proving the optimality for the majority of the instances at the root node.
  - fathoming nodes in the branch-and-bound
  - allowing an effective branching rule in the branch-and-bound
- Instances, precedence graphs, detailed computational results are available at www.or.unimore.it.
THANK YOU
## Conclusions

<table>
<thead>
<tr>
<th>Name</th>
<th>n</th>
<th>m</th>
<th>C</th>
<th>$L_{\text{root}}$</th>
<th>$U_{\text{root}}$</th>
<th>sec</th>
<th>L</th>
<th>U</th>
<th>t. lim.</th>
<th>sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arcus2</td>
<td>111</td>
<td>176</td>
<td>6267</td>
<td>29</td>
<td>30</td>
<td>422.0</td>
<td></td>
<td></td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6837</td>
<td>28</td>
<td>427.9</td>
<td></td>
<td></td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7916</td>
<td>26</td>
<td>420.5</td>
<td></td>
<td></td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Barthol2</td>
<td>148</td>
<td>175</td>
<td>85</td>
<td>50</td>
<td>51</td>
<td>432.5</td>
<td></td>
<td></td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>106</td>
<td>40</td>
<td>433.4</td>
<td></td>
<td></td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Buxey</td>
<td>29</td>
<td>36</td>
<td>27</td>
<td>13</td>
<td>14</td>
<td>411.4</td>
<td>14</td>
<td>–</td>
<td>5.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td>12</td>
<td>408.8</td>
<td>13</td>
<td>–</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>33</td>
<td>11</td>
<td>408.3</td>
<td>12</td>
<td>–</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Gunther</td>
<td>35</td>
<td>45</td>
<td>44</td>
<td>15</td>
<td>16</td>
<td>409.4</td>
<td>16</td>
<td>–</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>49</td>
<td>15</td>
<td>411.6</td>
<td>16</td>
<td>–</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>54</td>
<td>15</td>
<td>411.0</td>
<td>16</td>
<td>–</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>Mukherje</td>
<td>94</td>
<td>181</td>
<td>201</td>
<td>26</td>
<td>27</td>
<td>420.6</td>
<td></td>
<td>26</td>
<td>284.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>248</td>
<td>23</td>
<td>419.4</td>
<td></td>
<td>23</td>
<td>3333.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>301</td>
<td>21</td>
<td>424.6</td>
<td></td>
<td></td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>324</td>
<td>21</td>
<td>423.5</td>
<td></td>
<td></td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Sawyer</td>
<td>30</td>
<td>32</td>
<td>25</td>
<td>14</td>
<td>15</td>
<td>408.3</td>
<td>15</td>
<td>–</td>
<td>80.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>30</td>
<td>12</td>
<td>409.7</td>
<td>13</td>
<td>–</td>
<td>10.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>33</td>
<td>11</td>
<td>408.2</td>
<td>12</td>
<td>–</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>n</td>
<td>m</td>
<td>C</td>
<td>$L_{\text{root}}$</td>
<td>$U_{\text{root}}$</td>
<td>sec</td>
<td>L</td>
<td>U</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------------------</td>
<td>-------------------</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>Scholl</td>
<td>297</td>
<td>423</td>
<td>1515</td>
<td>80</td>
<td>82</td>
<td>744.8</td>
<td>–</td>
<td>80</td>
<td>7131.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1548</td>
<td></td>
<td></td>
<td>80</td>
<td>81</td>
<td>538.1</td>
<td>–</td>
<td>80</td>
<td>2622.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1584</td>
<td></td>
<td></td>
<td>80</td>
<td>83</td>
<td>542.8</td>
<td>–</td>
<td>80</td>
<td>6954.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1620</td>
<td></td>
<td></td>
<td>80</td>
<td>81</td>
<td>575.7</td>
<td>–</td>
<td>80</td>
<td>1264.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1742</td>
<td></td>
<td></td>
<td>80</td>
<td>81</td>
<td>579.1</td>
<td>–</td>
<td>80</td>
<td>2596.6</td>
<td></td>
</tr>
<tr>
<td>Tonge</td>
<td>70</td>
<td>86</td>
<td>160</td>
<td>24</td>
<td>25</td>
<td>417.8</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>168</td>
<td></td>
<td></td>
<td>24</td>
<td>25</td>
<td>415.9</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>176</td>
<td></td>
<td></td>
<td>24</td>
<td>25</td>
<td>420.7</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>185</td>
<td></td>
<td></td>
<td>23</td>
<td>24</td>
<td>414.7</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td>Warnecke</td>
<td>58</td>
<td>70</td>
<td>54</td>
<td>34</td>
<td>35</td>
<td>413.2</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>56</td>
<td></td>
<td></td>
<td>32</td>
<td>33</td>
<td>416.5</td>
<td>33</td>
<td>–</td>
<td>98.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>58</td>
<td></td>
<td></td>
<td>32</td>
<td>33</td>
<td>412.3</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td></td>
<td></td>
<td>30</td>
<td>31</td>
<td>416.3</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>62</td>
<td></td>
<td></td>
<td>29</td>
<td>31</td>
<td>412.1</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>65</td>
<td></td>
<td></td>
<td>28</td>
<td>29</td>
<td>416.8</td>
<td>29</td>
<td>–</td>
<td>1768.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>68</td>
<td></td>
<td></td>
<td>27</td>
<td>28</td>
<td>414.0</td>
<td>–</td>
<td>–</td>
<td>t. lim.</td>
<td></td>
</tr>
</tbody>
</table>