A Family of Inequalities for the Generalized Assignment Polytope

I. R. de Farias, Jr.
Department of Industrial Engineering
State University of New York at Buffalo, 403 Bell Hall, Buffalo, NY 14260-2050

G. L. Nemhauser*
School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta, GA 30332-0205

June 26, 2000

Abstract

We present a family of inequalities that are valid for the generalized assignment polytope. Although the inequalities are not facet-defining in general, they define facets of a polytope of a relaxation. We report computational results on the use of the inequalities in a branch-and-cut scheme that demonstrate their effectiveness.

Keywords: integer programming, generalized assignment, branch-and-cut

1 Introduction

Given a set of \( m \) jobs and a set of \( n \) processors, the Generalized Assignment Problem (GAP) consists of finding the cheapest assignment of each job to a single processor that respects the capacity constraints of the processors. GAP has many applications and several algorithms have been proposed for solving it. Cattrysse and van Wassenhove [1] review solution approaches for solving GAP, and, more recently, Savelsbergh [10] proposed a branch-and-price algorithm.

Let \( M = \{1, \ldots, m\} \) and \( N = \{1, \ldots, n\} \). Given \( i \in M \) and \( j \in N \), we denote by \( b_j > 0 \) the capacity of processor \( j \), \( a_{ij} \geq 0 \) the capacity of processor \( j \) required by job \( i \), and \( c_{ij} \) the cost of assigning job \( i \) to processor \( j \). We assume that \( b_j, a_{ij} \in \mathbb{Z} \), that \( m, n \geq 2 \), and that GAP is feasible. GAP can be formulated as

\*Partially supported by NSF grant DMI-9700285 to the Georgia Institute of Technology.
\[
\begin{align*}
\min & \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} \\
\sum_{i \in M} a_{ij} x_{ij} & \leq b_j, \quad j \in N, \\
\sum_{j \in N} x_{ij} & = 1, \quad i \in M, \\
x_{ij} & \in \{0, 1\}, \quad i \in M, j \in N.
\end{align*}
\]

By modifying the objective function coefficients appropriately, (2) can be replaced by

\[
\sum_{j \in N} x_{ij} \leq 1, \; i \in M,
\]

see [7].

Let \( S = \{ x \in \{0, 1\}^{mn} : x \text{ satisfies (1)} \; \forall j \in N \text{ and (4)} \; \forall i \in M \} \) and \( PS = \text{conv}(S) \). Because \( PS \) is a full-dimensional polytope, we use the inequality formulation of GAP with (4) replacing (2). We call \( PS \) the GAP polytope. The LP relaxation of GAP is obtained by replacing (3) with \( x_{ij} \in [0, 1] \; \forall i, j \in M \times N \). The set of feasible solutions of the LP relaxation of GAP is denoted by \( IPS \).

In [5, 6], Gottlieb and Rao presented several inequalities that are valid for the GAP polytope. In particular, they proved that the inequalities defining facets of the 0-1 knapsack polytopes that correspond to the individual capacity constraints (1) are facet-defining for the GAP polytope as well. These inequalities - lifted cover inequalities (LCIs) - are automatically generated in several commercial and academic MIP solvers that implement branch-and-cut schemes.

In this paper we introduce a new family of inequalities that are valid for the GAP polytope. The inequalities, in general, are not facet-defining. However, they define facets of the convex hull of the set of feasible solutions of a relaxation of GAP obtained by allowing the jobs to be partially executed (or not executed at all), while keeping the constraint that a job is assigned to at most one processor. The inequalities have a simple structure, and it is possible to separate them efficiently with a heuristic that is fast in practice. We demonstrate through computational experimentation that the inequalities are effective when incorporated into a branch-and-cut scheme to solve GAP.

The paper is organized as follows. In section 2 we present a family of inequalities that define facets of the convex hull of the set of feasible solutions of the relaxation. In section 3 we report computational results on the use of the inequalities presented in section 2 in a branch-and-cut scheme to solve GAP.

2 A Family of Valid Inequalities

In this section we relax the constraint that the jobs must be fully executed, or even executed at all. We keep the capacity constraint and the constraint that a job is assigned to at most one processor. A formulation for the relaxed problem can be obtained by substituting \( x_{ij} \in [0, 1] \; \forall i, j \in M \times N \) for (3), and by adding the following multiple-choice constraint
at most one of $x_{i_1}, \ldots, x_{i_n}$ is positive $\forall i \in M$. \hspace{1cm} (5)

We call this relaxation the \textit{multiple-choice GAP} (MGAP).

\textbf{Theorem 1} MGAP is \textit{NP-hard}.

\textbf{Proof} We reduce the partition problem \cite{4} to MGAP.

\textbf{PARTITION}

\textbf{INSTANCE:} Two sets of positive integers $M = \{1, \ldots, m\}$ and $S = \{a_1, \ldots, a_m\}$.

\textbf{QUESTION:} Does $M$ have a subset $M'$ such that $\sum_{i \in M'} a_i = \sum_{i \in M - M'} a_i$?

Let $n = 2$, $a_{i_1} = a_{i_2} = a_i$ $\forall i \in M$, $b_1 = b_2 = \frac{1}{2} \sum_{i \in M} a_i$, and consider the MGAP

$$\max \sum_{i \in M} a_i (x_{i_1} + x_{i_2}) + \sum_{i \in M} (x_{i_1} + x_{i_2})$$

$$\sum_{i \in M} a_i x_{ij} \leq \frac{1}{2} \sum_{i \in M} a_i, \quad j \in \{1, 2\},$$

$$x_{i_1} + x_{i_2} \leq 1, \quad i \in M,$$

$$x_{ij} \in [0, 1], \quad i \in M, j \in \{1, 2\}$$

at most one of $x_{i_1}$ and $x_{i_2}$ is positive, $i \in M$.

Note that the optimal value of the above MGAP is at most $\sum_{i \in M} a_i + m$. Suppose that the optimal value of MGAP is $\sum_{i \in M} a_i + m$. Then, if $\tilde{x}$ is an optimal solution, $\tilde{x}_{i_1} + \tilde{x}_{i_2} = 1 \forall i \in M$, and thus $\tilde{x}_{i} \in \{0, 1\} \forall i \in M, j \in \{1, 2\}$. Also, $\sum_{i \in M} a_i \tilde{x}_{i_1} = \sum_{i \in M} a_i \tilde{x}_{i_2} = \frac{1}{2} \sum_{i \in M} a_i$. Therefore, by taking $M' = \{i \in M : x_{i_1} = 1\}$, we have that the solution to PARTITION is \textit{yes}.

Conversely, suppose that the solution to PARTITION is \textit{yes}. This means that $\sum_{i \in M'} a_i = \sum_{i \in M - M'} a_i = \frac{1}{2} \sum_{i \in M} a_i$. Let $\hat{x}$ be given by

$$\hat{x}_{i_1} = \begin{cases} 1 & \text{if } i \in M' \\ 0 & \text{if } i \in M - M', \end{cases}$$

and

$$\hat{x}_{i_2} = \begin{cases} 1 & \text{if } i \in M - M' \\ 0 & \text{if } i \in M'. \end{cases}$$

It follows that $\hat{x}$ is a feasible solution to MGAP with objective function value $\sum_{i \in M} a_i + m$, and therefore it is an optimal solution. Thus, the solution to PARTITION is \textit{yes} iff the optimal value to MGAP is $\sum_{i \in M} a_i + m$. \hfill \Box

We denote by $PS^1$ the convex hull of the set of feasible solutions to MGAP. Next, we present a family of inequalities that are valid and facet-defining for $PS^1$. Because $PS \subseteq PS^1$, the inequalities are also valid for $PS$. The inequalities are a strengthening of the inequalities
of a family that define facets of the convex hull of the set of feasible solutions of an SOS II relaxation of GAP \cite{3}. They can be obtained by fixing some of the variables at 0, and then lifting \eqref{1}; see \cite{2, 12} for a general definition and result about lifting. To motivate the family of inequalities, we present an example.

**Example 1** Let $m = 3$, $n = 3$, $a_{11} = a_{12} = a_{13} = 2$, $a_{21} = a_{22} = a_{23} = 4$, $a_{31} = a_{32} = a_{33} = 5$, and $b_1 = b_2 = b_3 = 5$. Fix all variables at 0, except for $x_{11}$ and $x_{21}$. The inequality

$$2x_{11} + 4x_{21} \leq 5$$

\eqref{6} is valid and facet-defining for the resulting polytope $PS^1 \cap \{ x \in \mathbb{R}^9 : x_{12} = x_{13} = x_{22} = x_{23} = x_{31} = x_{32} = x_{33} = 0 \}$. When $x_{12} > 0$, $x_{11} = 0$. Because $4x_{21} \leq 4$, and $x_{12} \leq 1$, the lifting coefficient of $x_{12}$ is 1. Likewise, the lifting coefficient of $x_{13}$ is 1. Thus, by lifting \eqref{6} with respect to $x_{12}$ and $x_{13}$, we obtain

$$2x_{11} + 4x_{21} + x_{12} + x_{13} \leq 5.$$ \eqref{7}

In the same way, by lifting \eqref{7} with respect to $x_{22}$ and $x_{23}$, we obtain

$$2x_{11} + 4x_{21} + x_{12} + x_{13} + 3x_{22} + 3x_{33} \leq 5.$$ \eqref{8}

We now lift \eqref{8} with respect to $x_{31}$. The lifting coefficient of $x_{31}$, $a_{31}$, is given by

$$2x_{11} + 4x_{21} + x_{12} + x_{13} + 3x_{22} + 3x_{33} + a_{31}x_{31} \leq 5.$$ \eqref{9}

By substituting the point $x_{11} = 1, x_{31} = \frac{2}{3}, x_{22} = 1, x_{12} = x_{13} = x_{21} = x_{23} = x_{32} = x_{33} = 0$, which is feasible for $PS^1 \cap \{ x \in \mathbb{R}^9 : x_{32} = x_{33} = 0 \}$, in \eqref{9}, we obtain $a_{31} \leq 0$. Since \eqref{8} is valid for $PS^1 \cap \{ x \in \mathbb{R}^9 : x_{32} = x_{33} = 0 \}$, $a_{31} = 0$. Likewise, the lifting coefficients of $x_{32}$ and $x_{33}$ are 0, and therefore \eqref{8} is facet-defining for $PS$. \hfill $\square$

**Theorem 2** Let $I \subseteq M$ and $t \in N$. Suppose that $\sum_{i \in I} a_{it} > b_t$ and $\sum_{i \in I \setminus \{r\}} a_{it} < b_t$ for some $r \in I$. Then,

$$\sum_{i \in I} a_{it}x_{it} + \sum_{i \in I} \left( \max \{ 0, b_t - \sum_{i \in I \setminus \{i\}} a_{it} \} \right) \sum_{j \in N \setminus \{t\}} x_{ij} \leq b_t$$ \eqref{10}

defines a facet of $PS^1$.

**Proof** We first prove that \eqref{10} is valid. Inequality \eqref{10} is clearly valid when $x_{ij} = 0 \forall i \in I$ such that $b_t - \sum_{i \in I \setminus \{i\}} a_{it} > 0$ and $j \in N \setminus \{t\}$. So, suppose that $x_{uv} > 0$ for some $u \in I$ with $b_t - \sum_{i \in I \setminus \{u\}} a_{it} > 0$ and $v \in N \setminus \{t\}$. It follows that

$$\sum_{i \in I} a_{it}x_{it} + \sum_{i \in I} \left( \max \{ 0, b_t - \sum_{i \in I \setminus \{i\}} a_{it} \} \right) \sum_{j \in N \setminus \{t\}} x_{ij} =$$
\[
\sum_{i \in I - \{u\}} a_{it} x_{it} + \sum_{i \in I - \{u\}} (\max\{0, b_t - \sum_{i \in I - \{t\}} a_{it}\}) \sum_{j \in N - \{t\}} x_{ij} + (b_t - \sum_{i \in I - \{u\}} a_{it}) x_{uv} \leq b_t
\]

\[
\sum_{i \in I - \{u\}} a_{it} x_{it} + \sum_{i \in I - \{u\}} a_{it} \sum_{j \in N - \{t\}} x_{ij} + (b_t - \sum_{i \in I - \{u\}} a_{it}) x_{uv} \leq \sum_{i \in I - \{u\}} a_{it} + (b_t - \sum_{i \in I - \{u\}} a_{it}) = b_t,
\]

where the first equality is implied by (5), the first inequality is implied by \(a_{it} > b_t - \sum_{i' \in I - \{t\}} a_{i't}\) and \(a_{it} \geq 0 \forall i \in I\), and the second inequality is implied by (4).

We now show that (10) is facet-defining. For every \(uv \in M \times N\), let \(e^{uv} \in \{0, 1\}^{nm}\) be given by

\[
e_{ij}^{uv} = \begin{cases} 
1 & \text{if } i = j = uv \\
0 & \text{otherwise.}
\end{cases}
\]

Because \(\sum_{i \in I} a_{it} > b_t\), \(PS^1\) has \(|I|\) linearly independent points with \(x_{uv} = 0 \forall uv \in (M \times N) - (I \times \{t\})\).

Let \(u \in I\). Suppose that \(\sum_{i \in I - \{u\}} a_{it} < b_t\). Then, \(\sum_{i \in I - \{u\}} e^{it} + e^{uv} \in PS^1\) and satisfies (10) at equality \(\forall v \in N - \{t\}\). Now, suppose that \(\sum_{i \in I - \{u\}} a_{it} \geq b_t\). Then, there exists \(\bar{x} \in PS^1\) such that \(\bar{x}_{ij} = 0 \forall ij \in (M \times N) - (I \times \{t\})\), \(\bar{x}_{ut} = 0\), and \(\bar{x}\) satisfies (10) at equality. But then, \(\bar{x} + e^{uv} \in PS^1\) and satisfies (10) at equality \(\forall v \in N - \{t\}\).

Let \(u \in M - I\). Because \(\sum_{i \in I - \{t\}} a_{it} < b_t\), for \(\epsilon > 0\) small enough, \(\sum_{i \in I - \{t\}} a_{it} + \epsilon a_{ut} \leq b_t\). Let \(v \in N - \{t\}\). Then, \(\sum_{i \in I - \{t\}} e^{it} + e^{rv} + \epsilon e^{uv} \in PS^1\) and satisfies (10) at equality. Finally, let \(\tilde{x} \in PS^1\) be such that \(\tilde{x}_{ij} = 0 \forall ij \in (M \times N) - (I \times \{t\})\), and \(\tilde{x}\) satisfies (10) at equality. Then, \(\tilde{x} + e^{uv} \in PS^1\) and satisfies (10) at equality. Thus, \(PS^1\) has \(mn\) linearly independent points that satisfy (10) at equality.

\[\square\]

3 Computational Experience

We tested the effectiveness of (10) when used as cuts in a branch-and-cut scheme. We used an IBM RS6000/590 to run our tests and MINTO 3.0 [8, 11] as branch-and-cut solver, with CPLEX 6.0 as LP solver. We performed preliminary tests to compare the following alternatives: use only (10); use only LCl\s; use (10) and LCl\s. The preliminary computation proved the first option to be significantly less practical than the other two. We then compared MINTO with LCl\s only against MINTO with LCl\s and (10). For each instance we limited the size of the branching tree to 50,000 nodes, and the computational time to 36,000 seconds.

We used the four test data generators A, B, C, and D suggested in [7] to obtain test instances. The A and B data are known to be the easiest, while the D are the hardest ones. For each data type we performed tests on several instance sizes, and for each instance size we performed 5 tests with different instances. The instances are denoted as X.n.m, where X denotes the data generator. For example, C.3.30 denotes instances of the C data type with \(n = 3\) and \(m = 30\).
3.1 Branch-and-Cut Options

We performed preliminary tests with MINTO’s pre-processing, node selection, and branching strategies. The alternatives that worked best in the preliminary tests were:

- perform pre-processing and limited probing
- use best bound for node selection
- use a pseudo-cost variable rule for variable branching selection.

We then used these options in our tests. We performed preliminary tests with different cut generation strategies, e.g. introduce (10) at all nodes, at the root node only, etc. As expected, the preliminary tests indicated that the more (10) is used, the smaller the enumeration tree becomes. However, the price paid in computational time grows substantially when (10) is used intensively. The preliminary tests indicated that using (10) in only part of the enumeration tree yields a good balance between tree size reduction and increased computational time at each node. Of all the alternatives we tested, the one that appeared to be the most efficient was:

- use (10) in the first 1,000 nodes
- use (10) between nodes 10,000 and 11,000
- use (10) in all nodes from 20,000 on.

We then used this option in our computations. We tested different options with respect to row management alternatives, i.e. perform no row management, delete constraints with positive slacks every 10 iterations, etc. The alternative that seemed to be best was to delete all LCI’s and (10) with positive slacks every 100 nodes, and that was the alternative we adopted in our computations.

3.2 Separation Heuristic

It is easy to show that the separation problem for (10) is NP-hard when we consider any point $x \in \mathbb{R}^{mn} - PS^1$. This can be done by reducing the subset sum problem to it [4]. On the other hand, it appears that it is more difficult to show that the restricted problem of separating points from $LPS - PS^1$ is in fact NP-hard, although we suspect that it is. Thus, we used a heuristic to solve the separation problem.

The heuristic is based on the following observation. Let $\bar{x}$ be a point that does not satisfy (5). Let $t \in N$,

$$I_t = \{ i \in M : \bar{x}_i > 0 \},$$

and suppose that

$$...$$
\[
\sum_{i \in I_t} a_{it} \bar{x}_{it} = b_t. \tag{11}
\]

Let \( u \in I_t \) be such that
\[
b_t - \sum_{i \in I_t - u} a_{it} > 0, \bar{x}_{ut} > 0, \text{ and } \bar{x}_{uv} > 0 \text{ for some } v \in N - \{t\}. \tag{12}
\]

Then \( \bar{x} \) is cut off by (10) with \( I = I_t \).

Suppose that the current node is among those for which (10) is used as cuts, and that the optimal solution \( \bar{x} \) of the current LP relaxation does not satisfy (5). The separation heuristic is as follows. We perform \( n \) iterations, from \( t = 1 \) through \( t = n \). At each iteration we first check whether (11) is satisfied. If (11) is satisfied, then we check whether (10) with \( I = I_t \) is not satisfied by \( \bar{x} \) (this is equivalent to Condition (12) being satisfied for some \( u \in I_t \)), in which case the inequality cuts off \( \bar{x} \), and we add it to the LP relaxation.

If at the end of the separation heuristic we have found at least one inequality that cuts off \( \bar{x} \), then we solve the new LP relaxation. If the new optimal solution satisfies (5), or the new optimal value is not greater than the objective function value of the best feasible solution found so far, we fathom the current node. Otherwise, we apply the separation heuristic to the optimal solution of the new LP relaxation. On the other hand, if at the end of the separation heuristic we do not find any inequality that cuts off \( \bar{x} \), we branch.

We illustrate the separation heuristic with the following example.

**Example 2** Consider the instance of Example 1. Let \( \bar{x} \) be given by \( \bar{x}_{11} = \bar{x}_{21} = \bar{x}_{31} = \bar{x}_{12} = \bar{x}_{32} = 0, \bar{x}_{22} = \frac{1}{4}, \bar{x}_{33} = 1, \) and \( \bar{x}_{23} = \frac{3}{4} \). For \( t = 1 \), \( 2\bar{x}_{11} + 4\bar{x}_{21} + 5\bar{x}_{31} = 0 < 5 \).

For \( t = 2 \), \( 2\bar{x}_{12} + 4\bar{x}_{22} + 5\bar{x}_{32} = 5 \). \( I_2 = \{2, 3\} \), and the resulting inequality (10) is \( 4\bar{x}_{22} + 5\bar{x}_{32} + \bar{x}_{31} + \bar{x}_{33} \leq 5 \). However, this inequality does not cut off \( \bar{x} \).

Finally, for \( t = 3 \), \( 2\bar{x}_{13} + 4\bar{x}_{23} + 5\bar{x}_{33} = 5 \). \( I_3 = \{1, 2\} \), and the resulting inequality (10) is \( 2\bar{x}_{13} + 4\bar{x}_{23} + \bar{x}_{11} + \bar{x}_{12} + 3\bar{x}_{21} + 3\bar{x}_{22} \leq 5 \). This inequality cuts \( \bar{x} \), and we add it to the LP relaxation. \( \square \)

### 3.3 Computational Results

We tested instances A and B of various sizes, including very large instances with over 10,000 variables. Most instances, however, were solved at the root node, the majority of time spent with pre-processing and probing. In most cases, MINTO generated at most 3 lifted cover inequalities and inequalities (10). The C and D instances were more interesting. We were able to solve C instances with up to 9,000 variables. MINTO with (10) added was able to solve to proven optimality all C instances tested. MINTO without (10) added was unable to solve to proven optimality 8 C instances among the ones with 4,000 variables or more. The D instances, as expected, were considerably harder. We were able to solve D instances with up to 1,575 variables. MINTO with (10) added was able to solve to proven optimality.
all D instances. MINTO without (10) added was unable to solve to proven optimality 3 D instances.

Table 1 gives, for each instance type and size, the average number of nodes and computational time, over the 5 instances tested, of MINTO with LCI s only and with LCI s and (10). Our tests indicate that using (10) can be particularly useful when solving large C instances and the D instances. Table 1 also gives the average number of cuts generated when MINTO generates LCI s only and when MINTO generates LCI s and (10). Because, when (10) is added, the number of nodes is smaller for the largest C instances and the D instances, the number of cuts generated when (10) is added is also smaller.

Overall, with the addition of (10), the total number of nodes was reduced by 53%, and the total CPU time was reduced by 66%. To the best of our knowledge the largest instances we tested are considerably larger than the ones reported in the current literature. For example, Savelsbergh [10] reports computational experience with C and D instances that have up to 1,000 variables. Note also that for the 4 instance dimensions reported on in [10] and by us, our average computational time is smaller.

Acknowledgment We are grateful to an anonymous referee and to the editors for their valuable suggestions.
Table 1: Number of nodes, computational time, and number of cuts

<table>
<thead>
<tr>
<th>Inst. type</th>
<th>Nodes</th>
<th>LCI only</th>
<th>LCI &amp; (10)</th>
<th>% Red.</th>
<th>LCI only</th>
<th>LCI &amp; (10)</th>
<th>% Red.</th>
<th>LCI only</th>
<th>LCI &amp; (10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.3.30</td>
<td>62</td>
<td>50</td>
<td>19</td>
<td>19</td>
<td>0.98</td>
<td>0.94</td>
<td>4</td>
<td>27</td>
<td>21</td>
</tr>
<tr>
<td>C.3.50</td>
<td>48</td>
<td>37</td>
<td>23</td>
<td>23</td>
<td>1.21</td>
<td>1.09</td>
<td>10</td>
<td>25</td>
<td>21</td>
</tr>
<tr>
<td>C.6.30</td>
<td>162</td>
<td>23</td>
<td>86</td>
<td>86</td>
<td>2.64</td>
<td>1.47</td>
<td>44</td>
<td>21</td>
<td>21</td>
</tr>
<tr>
<td>C.25.40</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3.27</td>
<td>3.44</td>
<td>-5</td>
<td>35</td>
<td>37</td>
</tr>
<tr>
<td>C.30.40</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4.20</td>
<td>4.20</td>
<td>0</td>
<td>25</td>
<td>20</td>
</tr>
<tr>
<td>C.35.45</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4.98</td>
<td>4.95</td>
<td>0</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>C.20.200</td>
<td>7106</td>
<td>2660</td>
<td>63</td>
<td>63</td>
<td>1,414</td>
<td>562</td>
<td>60</td>
<td>806</td>
<td>492</td>
</tr>
<tr>
<td>C.20.250</td>
<td>39935</td>
<td>5323</td>
<td>87</td>
<td>87</td>
<td>15,357</td>
<td>2,159</td>
<td>86</td>
<td>2,096</td>
<td>901</td>
</tr>
<tr>
<td>C.20.300</td>
<td>28740</td>
<td>5278</td>
<td>82</td>
<td>82</td>
<td>14,072</td>
<td>2,791</td>
<td>80</td>
<td>1,956</td>
<td>766</td>
</tr>
<tr>
<td>C.20.350</td>
<td>29087</td>
<td>2380</td>
<td>92</td>
<td>92</td>
<td>17,738</td>
<td>1,502</td>
<td>92</td>
<td>3,850</td>
<td>789</td>
</tr>
<tr>
<td>C.20.400</td>
<td>19715</td>
<td>9780</td>
<td>50</td>
<td>50</td>
<td>19,782</td>
<td>8,045</td>
<td>59</td>
<td>3,780</td>
<td>1,797</td>
</tr>
<tr>
<td>C.20.450</td>
<td>18183</td>
<td>8325</td>
<td>54</td>
<td>54</td>
<td>18,346</td>
<td>8,735</td>
<td>52</td>
<td>2,664</td>
<td>2,069</td>
</tr>
<tr>
<td>D.3.30</td>
<td>4521</td>
<td>2519</td>
<td>44</td>
<td>44</td>
<td>39</td>
<td>23</td>
<td>41</td>
<td>234</td>
<td>170</td>
</tr>
<tr>
<td>D.3.50</td>
<td>11077</td>
<td>6771</td>
<td>39</td>
<td>39</td>
<td>148</td>
<td>96</td>
<td>35</td>
<td>827</td>
<td>545</td>
</tr>
<tr>
<td>D.6.30</td>
<td>30114</td>
<td>16041</td>
<td>47</td>
<td>47</td>
<td>838</td>
<td>438</td>
<td>48</td>
<td>1,109</td>
<td>740</td>
</tr>
<tr>
<td>D.25.40</td>
<td>41200</td>
<td>28411</td>
<td>31</td>
<td>31</td>
<td>4,717</td>
<td>3,290</td>
<td>30</td>
<td>730</td>
<td>671</td>
</tr>
<tr>
<td>D.30.40</td>
<td>30964</td>
<td>26389</td>
<td>15</td>
<td>15</td>
<td>3,434</td>
<td>2,852</td>
<td>17</td>
<td>616</td>
<td>608</td>
</tr>
<tr>
<td>D.35.45</td>
<td>25871</td>
<td>20969</td>
<td>19</td>
<td>19</td>
<td>3,966</td>
<td>3,466</td>
<td>13</td>
<td>685</td>
<td>640</td>
</tr>
<tr>
<td>TOTAL</td>
<td>286788</td>
<td>134939</td>
<td>53</td>
<td>53</td>
<td>99,868</td>
<td>33,975</td>
<td>66</td>
<td>19,506</td>
<td>10,328</td>
</tr>
</tbody>
</table>

References


