The purpose of this paper is to provide efficient methods for solving a large spectrum of industrial cutting stock problems. We propose two methods. Both are based on dynamic programming. In both methods, we reduce the computation burden by keeping, at each stage of the dynamic programming process, only the states which seem to be the most promising in terms of cost. One of the methods favours the computation time at the expense of the quality of the solution; this method is used in the sale department, where the goal is to propose a "good" solution in less than one minute. The second method favours the quality of the solution. Industrial applications are proposed. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Dynamic programming; Industrial application; Multicriteria cutting stock problem

1. Introduction

Cutting problems are NP-hard (see [5,7,8]). Thus, only small size problems can be solved optimally. These problems are solved using either integer linear programming (see [6,9]) or dynamic programming, or branch-and-bound, depending on the type of problem.

But most of the cutting problems use heuristic algorithms. The next fit (NF) heuristic (see [1,2]) consists in selecting the stored bars in a random order and of assigning the ordered bars also in a random order. If the next ordered bar cannot be assigned to the stored bar under consideration, a new ordered bar is selected and the unused part of the previous ordered bar is considered as scrap. The first fit (FF) heuristic (see [1–3]) is similar to NF, except that the unused part of a stored bar is considered as a stored bar and can be used later. The first fit decreasing (FFD) algorithm (see [1,2]) is the heuristic NF in which the ordered bars are selected in the decreasing order of their length. Numerous other heuristic algorithms are available as, for instance, best fit (BF) (see [1–3]), worst fit (WF) (see [1–3]), almost worst fit (AWF) (see [1,2])
to quote only a few. All these approaches are similar, except in the choice of the stored and ordered bars, and in the way the unused parts of the stored bars are processed.

The heuristic algorithms presented hereafter are based on a dynamic programming approach in which, at each step of the process, only some of the states are kept for further computations. Each of these states is supposed to be among the most promising (in terms of cost). The smaller the number of states kept for further consideration, the faster the computation, but the more likely is the solution obtained far from the optimal one.

Two algorithms are proposed. The first one, called Heuristic Constraining the Utilisation of the Stored Bars (HCUS) aims at reducing as much as possible the computation time while reaching a reasonable cost. It is used by the salesmen to respond in “real time” to the customers’ demands. The second one, called Heuristic Constraining the Utilisation of Stored and Ordered bars (HCUSO) aims at reaching a near-optimal solution: reducing the computation burden is no more the main goal in this case, since the computation is performed during the night to prepare the production of the next day.

In Section 2, we present the dynamic programming formulation and introduce the notations. Section 3 is devoted to HCUS and Section 4 to HCUSO. Section 5 proposes comparative results extracted from an industrial application. Section 6 is the conclusion.

2. Problem formulation and dynamic programming approach

The problem at hand consists in cutting bars to smaller bars in order to meet the customers’ requirements. We denote by stored bars (SBs for short) the bars kept in stock from which the required bars are cut, and by ordered bars (OBs for short) the bars required by the customers.

Let \( L_1, L_2, \ldots, L_M \) be the lengths of the SBs and \( N_i \) the number of available SBs of length \( L_i \) \((i = 1, 2, \ldots, M)\). We denote by \( S \) the \( M \)-components row vector whose \( i \)-th component is \( N_i \).

Similarly, \( l_i \) and \( n_i \) are respectively the length and the number of OBs of length \( l_i \), for \( i = 1, 2, \ldots, m \).

We represent these customers’ requirements by a row vector \( O \) with \( m \) elements, the \( i \)-th element being \( n_i \), number of OBs of length \( l_i \).

The objective function to be minimised is a cost which is the sum of the costs resulting from scrapes, set-up activities, cutting activities, etc. In the problem considered in this paper, these costs are known.

To reduce the costs resulting from set-up and cutting activities, it is possible to group SBs into faggots if their lengths are the same and if they have to be cut out from the same pattern. The size of a faggot \( f \) is denoted by \( s(f) \): it is the number of identical SBs grouped in faggot \( f \).

In the remainder of this paper, notation \( V \prec W \), where \( V \) and \( W \) are two \( m \)-vectors, means that any component of \( V \) is strictly less than or equal to the corresponding component of \( W \) and that at least one component of \( V \) is strictly less than the corresponding component of \( W \). Notation \( V \preceq W \) means that any component of \( V \) is less than or equal to the corresponding component of \( W \); in this case, all the components of \( V \) may be equal to the corresponding components of \( W \). Finally, notation \( V < W \) means that any component of \( V \) is strictly less than the corresponding component of \( W \). Indeed, symbols \( >, \geq \) and \( > \) have a meaning which is the opposite to \( <, \leq \) and \( <, \) respectively.

Let us denote by \( \Phi_{S,0}(p) \) the cost incurred if we carry out all the customers’ requirements represented by the row vector \( O \) in the SBs represented by vector \( S \) by cutting \( p \) faggots. If it is impossible to meet all the requirements with \( p \) faggots, we set the cost to infinity.

Let:
- \( S_0 \) be the vector representing the bars in stock, i.e. the SBs.
- \( O_0 \) be the vector representing customers’ requirements, i.e. the OBs.

Then, for all \( S \preceq S_0 \), \( O \preceq O_0 \) and \( p \),

\[
\Phi_{S,0}(p) = \min_{s' < S, o' < O} \left\{ \Phi_{s',0}(p-1) + \Phi_{S-s',O-o'}(1) \right\}.
\] (1)
In this formulation, the number of $p$ faggots belongs to $1, 2, \ldots, P$, where $P = \min \{l_M, S', l_m, O'\}$.

$l_M$ (respectively, $l_m$) is a row vector the $M$ (respectively $m$) elements of which equal $l$, and $t$ denotes the transpose.

Relation (1) can be applied recursively, starting by computing $\Phi_{S, O}(1)$ for all $S \leq S_0$ and $O \leq O_0$, and then computing successively, for $p = 2, 3, \ldots, P$, $\Phi_{S, O}(p)$ for all $S \leq S_0$ and $O \leq O_0$. Doing so, the components of the right hand side of relation (1) are always computed before they are needed. Thus, only the values of the $\Phi_{S, O}(1)$ functions have to be computed at the beginning of the computation process.

The drawback with this dynamic programming approach is its complexity to compute $\Phi_{S, O}(p)$, we have to consider all the $S' < S$ and $O' < O$; furthermore, relation (1) should be applied to all pairs $(S, O)$ such that $S \leq S_0$ and $O \leq O_0$. As a consequence, this approach cannot be used for industrial problems. In the following sections, we propose two approaches derived from the previous one, but which take into account only the partial solutions which lead the most likely to an optimal (or to a near-optimal) solution.

3. Heuristic constraining the utilisation of stored bars

3.1. HCUS formulation

We denote by $(p, x, S, O)$ the set of cutting activities which consists in cutting ordered bars belonging to the set of bars represented by vector $O$ using all the bars of a subset of $x$ bars of $S$ grouped in $p$ faggots. Note that a necessary (but not sufficient) condition to have a solution to a cutting activity belonging to this set is that $p$ be greater than or equal to the number of different lengths of bars in the subset of $x$ bars under consideration, and lower than or equal to $x$. If a cutting activity belonging to $(p, x, S, O)$ has no solution, the corresponding cost is set to infinity. Moreover, $\Phi_{S, O}(p, x)$ is the lowest cost among the costs associated to the cutting activities belonging to $(p, x, S, O)$, i.e. the costs associated with the best choice of ordered bars taken from the set represented by $O$ and cut in a set of $x$ stored bars taken from $S$ and grouped into $p$ faggots.

The recursive equation used in HCUS is the following

$$\Phi_{S, O}(p, x) = \min_{p-1 \leq x' < x} \left\{ \Phi_{S, O}(p - 1, x') + \Phi_{S - S', O - O}(1, x - x') \right\},$$

(2)

where:

- $S'$ is the subset of $x'$ stored bars taken from $S$ which leads to the minimal of $\Phi_{S, O}(p - 1, x')$.
- $O'$ is the set of ordered bars obtained as the result of the computation of $\Phi_{S, O}(p - 1, x')$.

In this formulation, we consider only the best solution (in terms of cost) of the set of cutting activities $(p - 1, x', S, O)$ instead of all the possible solutions, assuming that this solution is more likely to lead to the global optimal solution. Indeed, this may be false, but the results presented in Section 5 prove that this assumption is realistic in our case and for commercial purpose.

Note that formulation (2) reduces considerably the computational burden by applying relation (2) successively for $p = 2, 3, \ldots, P$. The only drawback is the computation of $\Phi_{S - S', O - O}(1, x - x')$ which consists in cutting $(x - x')$ stored bars belonging to $S - S'$ grouped in one faggot to obtain a subset of the ordered bars represented by $O - O'$. Since the cutting operation should be made in one faggot, it is impossible if:

(i) the vector representing the $(x - x')$ bars belonging to $S - S'$ has more than one strictly positive component,

(ii) it is physically impossible to group the $(x - x')$ bars into one faggot because of the size or of the stability of such a faggot. This does not mean that $x - x'$ is constant: the number of SBs which can be cut in one faggot depends on the section of the bars and on the possibility to fix these bars in a way which guarantees that they will not move during the cutting activity.

In these cases, $\Phi_{S - S', O - O}(1, x - x') = +\infty$. If the vector representing the $(x - x')$ bars has only one strictly positive component and if it is possible to group the corresponding SBs into one faggot, we will show hereafter how to select the ordered
bars included in the ordered bars represented by $O - O'$ in order to perform a cut in the faggot which is as inexpensive as possible. Since this problem is NP-hard, we will use a heuristic approach FFD. Indeed, by doing so we introduce another systematic error but, as the numerical examples presented in the remainder of this paper show, this approach leads to solutions which are satisfactory.

3.2. First fit decreasing

In this section, the goal is to evaluate $\Phi_{S-S', O-O'}(1, x - x')$, the minimal cost among the costs of the cutting activities which consist in cutting ordered bars from the set represented by $O - O'$ in a set of $x - x'$ bars belonging to the set represented by $S - S'$ and grouped in one faggot.

We define the vector $O$ whose components are the truncated values of the corresponding components of $O - O'$ divided by $(x - x')$. In other words, $O'$ represents the OBs to cut in the same faggot (and thus following the same pattern).

The problem to be solved is to find the optimal cut of a subset of the set of OBs represented by $O'$ in a SB the length $L$ of which is the length of the bars corresponding to the unique strictly positive component of the vector representing the $x - x'$ bars. The FFD algorithm is applied at this point.

Let $(l_i, n_i)$ be respectively the length and the number of the artificial OBs represented by $O'$ $(i = 1, 2, \ldots, n)$. These pairs are classified in the decreasing order of the lengths $l_i$, i.e. $l_1 > l_2 > \cdots > l_n$.

Applying the FFD algorithm consists in laying the artificial OBs on the unique artificial SB. This algorithm can be summarised as follows:

1. Set $\lambda = L$
2. For $i = 1$ to $n$ do
2.1. $r_i = \min \left\{ \left\lfloor \frac{\lambda}{l_i} \right\rfloor, n_i \right\}$
2.2. $n_i = n_i - r_i$
2.3. $\lambda = \lambda - r_i l_i$

The last value of $\lambda$ multiplied by the number of SBs in the faggot is the total length of scrap material. Since we also know the cost of the cutting activity (which depends on the size of the faggot) and the set-ups, it is possible to compute the cost $\Phi_{S-S', O-O'}(1, x - x')$ when the corresponding problem has a solution.

3.3. The HCUS algorithm

The HCUS algorithm starts by computing the optimal cost $\Phi_{S,O}(1, x)$ for all the $x \leq X$. $X$ being the total number of SBs represented by $S$. This cost is equal to infinity when it is impossible to form a unique faggot with $x$ stored bars extracted from the set represented by $S$, either because the size of the faggot would be too big for the cutting resource available, or because the faggot would be unstable on the machine. Otherwise, we apply the FFD algorithm or, if the size of the problem is reasonable, its extension (i.e. the same algorithm starting from various orders of the artificial OBs).

In both cases, the value obtained for $\Phi_{S,O}(1, x)$ is not optimal but, as shown through the numerical applications, is very good from a practical point of view.

Then, for $p = 2, 3, \ldots, P$, where $P$ is the minimum between the total number of SBs and the total number of OBs, and for all $x \leq X$, we compute $\Phi_{S,O}(p, x)$ using relation (2). Note that $\Phi_{S,O}(p - 1, x')$, $x' < x$, has been previously computed. Furthermore, $\Phi_{S-S', O-O'}(1, x - x')$ is computed using FFD as shown in Section 3.2.

The algorithm stops when $p = P$ and $x = X$.

4. Heuristic constraining the utilisation of stored and ordered bars

4.1. HCUSO formulation

We denote by $(p, x, y, S, O)$ the set of cutting activities which consist in cutting $y$ ordered bars belonging to the set of ordered bars represented by $O$ out of $x$ stored bars belonging to $S$ grouped in $p$ faggots. Note that, in this problem, all the $y$ OBs represented must be cut out of all the $x$ SBs
grouped into $p$ faggots. The difference between a cutting activity belonging to $(p, x, S, O)$ and a cutting activity belonging to $(p, x, y, S, O)$ introduced in Section 3.1 is that the number of OBs to be cut is fixed in the latter case while it is not specified in the first one.

Some of the cutting activities of $(p, x, y, S, O)$ are unfeasible, either because it is impossible to cut the $y$ OBs selected out of the $x$ SBs selected grouped in $p$ faggots, or because the faggots obtained cannot be handled by the cutting machines. We set the cost associated with these unfeasible cutting activities to $+\infty$. The cost $\Phi_{S, O}(p, x, y)$ used in the remainder of this section is the lowest cost among the costs of all the cutting activities belonging to the set $(p, x, y, S, O)$.

Eq. (1) leads to

$$
\Phi_{S, O}(p, x, y) = \min_{(x', y') \prec (x, y)} \left\{ \Phi_{S, O}(p - 1, x', y') + \Phi_{S - S', O - O'}(1, x - x', y - y') \right\}
$$

(3)

with $x \geq p - 1$ and $x' = y'$.

In relation (3), the sets $S'$ and $O'$ are respectively the set of the $x'$ stored bars of $S$ and the set of the $y'$ ordered bars of $O$ which lead to the optimal solution, i.e. which lead to the minimal cost $\Phi_{S, O}(p - 1, x', y')$.

Formulation (3) leads to a higher computation burden than formulation (2) since we keep only one solution for each set of $x$ stored bars in Eq. (2) instead of all the pairs of $x$ stored bars and $y$ ordered bars in Eq. (3).

When applying Eq. (3), we start by computing the cost for all possible pairs $(x, y)$, where the $x$ stored bars are grouped in one unique faggot. This leads to the minimal cost $\Phi_{S, O}(1, x, y)$. This computation is performed using a heuristic based on the permutation of the $y$ OBs. This heuristic will be presented in Section 4.2.

We then apply relation (3) for $p = 2$ and for all the possible pairs $(x, y)$. This requires the computation of $\Phi_{S, O}(1, x', y')$, which has been done previously, and the computation of $\Phi_{S - S', O - O'}(1, x - x', y - y')$. This last computation concerns the specific set $S - S'$ of SBs, and the specific set $O - O'$ of OBs: This computation is different from the computation of $\Phi_{S, O}(1, x - x', y - y')$ which has been previously done based on $O$ and $S$. The computation of $\Phi_{S - S', O - O'}(1, x - x', y - y')$ will be performed using the permutation heuristic algorithm presented in Section 4.2. As explained in Section 4.3, the application of relation (3) needs:

- the use of a previous result for $\Phi_{S, O}(p - 1, x', y')$,
- the computation of $\Phi_{S - S', O - O'}(1, x - x', y - y')$.

### 4.2. The permutation heuristic

The permutation heuristic (PH) algorithm consists in assigning a given number (say $y$) of OBs to one SB bar. In other words, we are looking for $y$ bars taken from the set represented by vector $O$ which, if cut out from the given SB bar, leads to a minimal cost.

The PH works as follows. The bars represented by $O$ are ordered in the increasing order of their length. Taking the bars in this order, the $y$ first bars represented by $O$ are put on the SB bar. If no solution exists (i.e. if the SB bar is too short), the cost is set to $+\infty$. If one solution exists, we try to replace the shortest OB belonging to the solution by the longest possible OB bar which does not belong to the solution, and so on until no further change can be done.

**Algorithm PH**

1. Let $L$ be the length of the SB.
2. Let $l_1, l_2, \ldots, l_n$ be the lengths of the OBs, after ordering these bars in the increasing order of their lengths, i.e. $l_1 \leq l_2 \leq \cdots \leq l_n$. We denote by $i$ the bar of lengths $l_i$.
3. If $l_1 + l_2 + \cdots + l_y > L$, the problem does not have a solution, and the corresponding cost is set to $+\infty$. Otherwise, compute the corresponding cost.

**Note:** In the software designed for the client company, the cutting-lines and the parts of the SB cut out to obtain a proper end are taken into account to check the feasibility of a cut.

4. For $i = 1$ to $y$, do:
   4.1. For $j = n$ to $y + 1$, step 1
   If $l_1 + l_2 + \cdots + l_{i-1} + l_j + l_{i+1} + \cdots + l_y \leq L$, (i) compute the corresponding cost,
(ii) if this cost is less than the lowest cost previously obtained, replace \(l_i\) by \(l_j\) and rename \(i\) as \(j\) and \(j\) as \(i\).

4.2. End of loop \(j\).

5. End.

The best solution is taken as an evaluation of \(\Phi_{S,O}(1, x, y)\).

In Section 4.3, we show how to use PH in the algorithm HCUSO.

4.3. Algorithm HCUSO

The HCUSO algorithm was already presented in Section 4.1. Hereafter, we formalise this algorithm.

1. For each pair \((x', y')\), where \(x'\) is less than or equal to the maximum number of SBs of the same length, and \(y'\) is greater than or equal to \(x'\) and less than or equal to the total number of OBs.

  Compute \(\Phi_{S,O}(1, x', y')\).

   This computation requires the comparison of the costs of each pair \((x', y')\) taken from \((S, O)\).

   \(\Phi_{S,O}(1, x', y')\) is the minimum of these values when the computation is completed. We store in \((S', O')\) the vectors which lead to this minimum.

   This computation is made using the PH.

2. Set \(p = 2\).

3. For each pair \((x, y)\), where \((x, y)\) verifies the same constraints as \((x', y')\) in 1, we apply relation (3) for \(p\). The first part of the second member of Eq. (3) was computed in step 1. The second part of the second member of Eq. (3), that is \(\Phi_{S-S', O-O'}(1, x - x', y - y')\), is computed using algorithm PH.

   \(\Phi_{S,O}(2, x, y)\) is the smallest value among all the pairs \((x, y)\), and we keep the pair which leads to this minimal value. This pair is \((S', O')\).

4. If \(p\) is less than the number of SBs, then set \(p = p + 1\) and go to 3.

Remark. When computing \(\Phi_{S,O}(1, x', y')\) or \(\Phi_{S-S', O-O'}(1, x - x', y - y')\) using PH, we have to cut all the \(y'\) bars (respectively the \(y - y'\) bars) in the \(x'\) (respectively the \(x - x'\)) in one faggot. This implies that the \(x'\) (respectively the \(x - x'\)) are bars of the same length to be cut in one faggot to obtain the \(y'\) (respectively the \(y - y'\)) OBs.

A consequence of this is that:

(i) the number of OBs of the same length is a multiple of the number of SBs of the same length,

(ii) if \(r_1, \ldots, r_s\) are these multiples for the OBs bars of \(s\) different lengths, say \(l_1, l_2, \ldots, l_s\), then we should be able to cut \(r_1, r_2, \ldots, r_s\) OBs of lengths \(l_1, l_2, \ldots, l_s\) respectively in one SB bar.

If one of these conditions does not hold, then the cost is infinite. Otherwise, the PH is used to perform (ii).

5. Industrial application

Algorithms HCUS and HCUSO have been used for about three years in a partner company which belongs to a French subcontractor of a Belgian steel consortium. In Table 1, we present 24 examples which are considered as the most difficult ones by our partner.

Column “No. SB” gives the number of stored bars, while column “No. OB” displays the number of ordered bars which have to be cut. All the costs included in this table are expressed in terms of their equivalent working times expressed in minutes. These costs include the costs resulting from scrape, set-up activities and cutting activities.

The column “Industrial Optimax Time” is the cost obtained using Optimax, a software tool previously available in the company. This software tool is based on a branch-and-bound approach and stops when it reaches a solution whose scrap is less than a percentage given by the user. The drawback with this software was that it required between 30 min and several hours to provide a result; sometimes, computations were to be cancelled after several hours without result. The next three columns provide respectively the cost, the computation time, and the difference in percentage with the results provided by Optimax when using HCUS. The last three columns provide the same information when HCUSO is used.

As previously outlined, HCUS is more efficient than HCUSO in terms of computation time, while the reverse holds when the criterion is the cost.
HCUS is used by the salesmen to answer the customers' requirements on the phone and to tell them, in real time, if they can satisfy their demand or not, taking into account the inventories. At this level, the time-to-answer is the most important, and a reasonable cost is acceptable. The final cutting plan is computed during the night using HCUSO and leads to solutions which are, on the average, much better than the ones obtained using HCUS. A global measure of the quality of HCUSO has been made at the end of the first year this algorithm was used. This measure was the ratio of the number of tons delivered to the customers by the number of tons consumed. The result was slightly greater than 0.95. In other words, scrap was less than 5%; this includes the cutting-lines and the parts of the stored bars cut out to obtain proper ends. Indeed, it is impossible to provide a global comparison with Optimax for two reasons: (i) Optimax is sometimes unable to provide a solution; in this case, no comparison is possible, and (ii) the company stopped the use of Optimax less than one month after the previous algorithms were implemented since they were rapidly convinced that these algorithms are much better than Optimax.

6. Conclusions

When a dynamic programming approach leads to an excessive computation burden, it is desirable to remove some states at each stage of the process and to pursue the computation with the remaining states. The set of remaining states is supposed to contain at least a state belonging to the critical path. The process used to define this set depends on the problem at hand. For instance, it is often

Table 1
Numerical results

<table>
<thead>
<tr>
<th>Files</th>
<th>Industrial optimax cost (min)</th>
<th>HCUS</th>
<th>HCUSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>SB</td>
<td>OB</td>
<td>Cost (min)</td>
</tr>
<tr>
<td>1</td>
<td>796</td>
<td>3124</td>
<td>8549.78</td>
</tr>
<tr>
<td>2</td>
<td>95</td>
<td>126</td>
<td>3636.71</td>
</tr>
<tr>
<td>3</td>
<td>112</td>
<td>137</td>
<td>4360.68</td>
</tr>
<tr>
<td>4</td>
<td>35</td>
<td>157</td>
<td>747.58</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>34</td>
<td>2374.05</td>
</tr>
<tr>
<td>6</td>
<td>106</td>
<td>145</td>
<td>4523.02</td>
</tr>
<tr>
<td>7</td>
<td>108</td>
<td>300</td>
<td>6352.77</td>
</tr>
<tr>
<td>8</td>
<td>207</td>
<td>338</td>
<td>3740.58</td>
</tr>
<tr>
<td>9</td>
<td>53</td>
<td>146</td>
<td>2039.63</td>
</tr>
<tr>
<td>10</td>
<td>55</td>
<td>60</td>
<td>6157.66</td>
</tr>
<tr>
<td>11</td>
<td>204</td>
<td>1031</td>
<td>3109.16</td>
</tr>
<tr>
<td>12</td>
<td>32</td>
<td>285</td>
<td>2278.53</td>
</tr>
<tr>
<td>13</td>
<td>772</td>
<td>1491</td>
<td>9972.5</td>
</tr>
<tr>
<td>14</td>
<td>59</td>
<td>1821</td>
<td>659.61</td>
</tr>
<tr>
<td>15</td>
<td>167</td>
<td>417</td>
<td>1520.66</td>
</tr>
<tr>
<td>16</td>
<td>70</td>
<td>257</td>
<td>1136.66</td>
</tr>
<tr>
<td>17</td>
<td>1253</td>
<td>2637</td>
<td>774.49</td>
</tr>
<tr>
<td>18</td>
<td>246</td>
<td>767</td>
<td>5866.26</td>
</tr>
<tr>
<td>19</td>
<td>88</td>
<td>593</td>
<td>1631.66</td>
</tr>
<tr>
<td>20</td>
<td>352</td>
<td>1831</td>
<td>9143.98</td>
</tr>
<tr>
<td>21</td>
<td>207</td>
<td>437</td>
<td>2206.93</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>2</td>
<td>316.06</td>
</tr>
<tr>
<td>23</td>
<td>128</td>
<td>274</td>
<td>4258.83</td>
</tr>
<tr>
<td>24</td>
<td>60</td>
<td>60</td>
<td>1209.09</td>
</tr>
<tr>
<td>Totals</td>
<td>5229</td>
<td>14570</td>
<td>87728.88</td>
</tr>
<tr>
<td>Average</td>
<td>27.81</td>
<td>78.46</td>
<td>466.64</td>
</tr>
</tbody>
</table>
possible to simulate the system starting from each of the state to evaluate their potentialities, and to keep the states having the highest potentialities.

In the algorithms presented in this paper, the approach consists in defining a priori sets of solutions, and to keep the optimal state of each set. The quality of the solution, as well as the computation burden, depend on the definitions of these sets. In the contract which was performed for the company mentioned before, several sets of solutions were defined, and we kept the two algorithms presented in this paper. HCUS was selected first for its rapidity, which is the most important aspect for the sales department, while HCUSO was selected first for the quality of the solutions. The company accepted both algorithms because they exposed an appreciable improvement compared to the existing situation.

7. For further reading

[4]

Acknowledgements

The authors wish to thank the reviewers for their valuable comments. The help provided by one of the reviewers led to fundamental changes in the notations and, consequently, in the presentation of the paper.

References