Two Dimensional Knapsack problem using industrial Robots

Guillermo Camacho, David Álvarez Martínez, Paula Velasco
La Salle University – Bogotá Colombia, gcacamacho@unisalle.edu.co, pvelasco15@unisalle.edu.co
Universitat Politècnica de València – Valencia, Espanha, dalvarez@iti.es,

1. Abstract

We have developed an automatic cutting system to resolve the two-dimensional non guillotine single knapsack problem (2D-SKP). The system is composed by four blocks: optimization stage, code-generator, manipulator and plasma cutter. This paper is the first of two companion papers that describe the system and its use. In this paper we concentrate on development and validation of the stages associated with the optimization and code-generator. At the optimization stage we use a Greedy-Randomized-Adaptive-Search-Procedure (GRASP); an iterative procedure that resolves the location of each piece in the container. At the code generation stage we use the forward kinematic model (FKM) of the manipulator; this model allows computing the position and orientation of the end-effector from specified values for the joint variables. The GRASP procedure was validated by using a benchmark and comparing: computation time, used area of the solution. The results allow concluding the correctness of the procedure and a superior performance in computation time. Finally, we propose a post-processing stage to improve the cutting performance. The post-processing stage increases the performance for the case-study with a reduction of 37.65% for cutting time and a reduction of 0.84 units in accuracy, in terms of relative error.

2. Keywords: Denavit Hartenberg parameters, Forward Kinematic Model (FKM), Greedy-Randomized-Adaptive-Search-Procedure (GRASP), Industrial Robots, Two-Dimensional Knapsack Problem (2D-SKP)

3. Introduction

Two-dimensional non-guillotine single knapsack problem (2D-SKP) consists of orthogonally cutting a subset of pieces within a single rectangular container such that the sum of the values of the cut pieces is maximized (Dyckhoff 1990; Toro-ocampo 2010). In order to execute the cutting task, the solution of the problem must be communicated to a mechanical system. In this work, the solution is communicated to a 6-DOF Robot through a special machine-code and using the architecture in Fig. 1.

![Fig. 1. Conceptual view of the blocks that conforms the cutting system](image-url)
post-processing stage to improve the cutting performance by the elimination of repeated cuttings-paths. The post-processing stage improves the performance for a benchmark with a reduction of 37.65% for cutting time and a reduction of 0.84 units in accuracy, in terms of relative error. These improvements must be examined with new benchmarks before generalization.

4. Background

4.1 GRASP algorithm (Greedy Randomized Adaptive Search Procedures)

GRASP algorithm was developed by Feo and Resende (Feo & Resende 1989) to resolve hard combinatorial optimization problems. Publications in (Parreño et al. 2008a), (Alonso et al. 2014), (Alvarez-Martínez et al. 2015)(Escobar-Falcón et al. 2015) demonstrate its quality and robustness to resolve packing and cutting problems. For an introduction we suggest to consult (Resende & Ribeiro 2003). The GRASP algorithm is an iterative procedure that combines a constructive algorithm with an improvement strategy. In the constructive algorithm a solution is computed by a sequential forward selection; i.e. adding elements to the solution at each iteration. The version used in this work adequate the constructive phase presented in (Parreño et al. 2008b) by representing solutions with maximal spaces. Maximal spaces are the largest empty rectangles available to be filled with pieces. Each selected piece is cut in a new space, updating the list of maximal spaces as indicated in Fig. 2.

![Management procedure of maximal spaces: creation, update and delete. Source: (Parreño et al. 2008b)](image)

The pseudo-code for the constructive component is described in Fig. 3. This algorithm uses a list $S$ of the maximal spaces and a list $P (P=P_1, P_2,...,P_n)$ with the pieces that remain to be cut.

Primarily a list $S$ with the available maximal-spaces is created and an empty list solution is created.

At the line 3 the algorithm applies a selection mechanism to choose a maximal-space. This mechanism uses two criteria: minimal area, minimal distance to the origin of the plate. Initially, the first criterion is applied; just in case of match the second criterion is applied. The first criterion represents a strategy to use the smallest available spaces, the second one tries to consume the spaces from the extreme to the center of the plate.

At the line 4 the algorithm selects the piece to be cut. Again, we have two criteria to conduct the selection: best-fit, max-volume. In the first one criterion, the algorithm computes the distances between a border of each piece $P_i$ and the extreme of the plate. Then, it sorts the list $P$ in non-decrement order of distance. Finally, it selects the piece with the minimal distance (best-fit). The second criterion represents a greedy strategy where the selected piece is the one that generates the best incremental cost in fitness (Resende & Ribeiro 2003). In case of multiple instances of the piece $P_i$, the algorithm creates rows of pieces combining the horizontal and vertical axes.

```
procedure GRASP_Constructive(l List:P; O List: solution)
1    initialize S, solution
2    while P ≠ Ø or S≠ Ø do
3        s ← selectMaximalSpace(S, criteria)
4        p ← selectItemP(criteria)
5        solution← addElementToSolution(s,p)
6        update(P,S,p,s)
7    end
8    return solution
end
```

Fig. 3. Pseudocode of the constructive component in GRASP Algorithm
At the line 5, the selected piece is added to the solution in the selected maximal space. Then at the line 6 the \( P \) and \( S \) lists are updated. Note that unless the selected piece \( p \) fits exactly in the selected maximal space \( s \), the selection will generate new available maximal-spaces. These new spaces appear in addition to the existsents and all of them compose the updated list \( S \). Once the list \( S \) has been updated, the \( P \) list is updated and the maximal spaces that cannot accommodate any of the pieces are deleted from \( S \). The process repeats until \( S = \emptyset \) or \( P = \emptyset \).

### 4.1. Randomization

For each type of pieces and each allowed orientation, the selectItem method builds a layer, based on the selection criteria (best-fit, max-volume). Each of these layers is called candidate and the set of all possible candidates are grouped into the Restricted Candidate List (RCL). Then, the selectItem method chooses a candidate by random based on the classic parameter \( \delta (\delta \in [0, 1]) \).

#### 4.1.1. Improve movement

In this work we eliminate the last \( k\% \) of the cut pieces from the solution, i.e. 60\%. The \( k \) value is selected by random between [30, 90]. Then, the removed pieces and the remain-to-be-cut pieces are processed by the constructive deterministic algorithm (\( \delta = 1 \)). In this movement the constructive algorithm uses both fitness functions: max-area, best-fit. The improvement movement is invoked only if the solution of the constructive algorithm is promissory, as represented in the line 3 of Fig. 4. The solutions are restricted to those that are greater than a dynamic threshold. In the beginning, the threshold takes the value of the first solution computed by the constructive algorithm. The procedure is detailed in (Alvarez-Martinez et al. 2015).

```plaintext
procedure GRASP_Improve(l List; P; O List: solution)
    for i in 0, 1, 2, ... Imax do
        solution ← GRASP_Constructive()
        if FilteringSolution(solution) then
            solution ← improveMovement(solution);
        end
    end
    return solution
```

Fig. 4. Pseudocode of the improve component in GRASP Algorithm. Imax is the maximal number of iterations.

### 4.2. Denavit Hartenberg Convention

The Denavit–Hartenberg parameters are four parameters that allow describing the spatial relationship between successive link coordinate frames of a manipulator. With these parameters it is possible to express the position and orientation of the manipulator’s relative to its base and for a set of joint variables. This representation is knowledge as the forward kinematic model and is given by equation (1)

\[
T_0^i = T_0^0 T_1^1 T_2^2 \cdots T_{n-1}^{n-1}
\]  

(1)

Where,

\[
T_{j-1}^{j} = \begin{bmatrix}
R_{j-1}^{j} & p_{j-1}^{j} \\
0 & 1
\end{bmatrix}
\]  

(2)

The transformation matrix \( T_{j-1}^{j} \) is a square matrix of size 4x4. This matrix describes the position \((p_{j-1}^{j})\) and orientation \((R_{j-1}^{j})\) of the \(i\)-th link frame relative to \((i-1)\). Both, the position and orientation are expressed in terms of the Denavit-Hartenberg parameters. There are two different forms of DH parameters (Reddy 2014): classical convention, presented by Denavit and Hartenberg (Denavit & Hartenber 1955) and used in textbooks as (Paul 1981; Fu et al. 1987; Spong et al. 1989; Siciliano et al. 2009), modified convention, presented by Khalil-Kleinfinger (Khalil & Kleinfinger 1986) and used in textbooks as (Craig 2001; Ollero 2001; Vivas 2010). Both notations represent a joint \( i \) as four DH parameters: translations \((a_i, d_i)\) and two angles \((\theta_i, \alpha_i)\). In this work we use the classical convention as expressed in equation (3).

\[
T_{i-1}^{i} = \begin{bmatrix}
c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\
s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\
0 & s\alpha_i & c\alpha_i & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\]  

(3)
4.3 Orientation Representation.
In this article we use a representation of the frame orientation knowledge as X-Y-Z fixed angles (Craig 2001). This selection was motivated for the convention used in the programming pendant of the HP20D manipulator (Motoman a Yaskawa Company 2015). Under this convention, the orientation of the end-effector frame (frame 6 in Fig. 4b) is computed as indicated in equations 4 to 6.

$$\beta = A \tan 2\left(- r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right) \quad (4)$$

$$\alpha = A \tan 2\left(r_{21}, r_1 / \cos(R_i)\right) \quad (5)$$

$$\gamma = A \tan 2\left(r_{32}, r_3 / \cos(R_i)\right) \quad (6)$$

Where, $Atan2(y,x)$ is a two-argument arc tangent function; $\gamma$, $\beta$, $\alpha$ are degrees of rotation in the fixed reference frame $O_x$, around the $x$, $y$ and $z$ axis, respectively.

5. Methodology
5.1 Development of the GRASP procedure
The described procedure in Figures 3 and 4 were implemented in C++. Two modules were added: (i) a graphical module to generate an image of the computed result, (ii) a module to compute the performance of the solution in terms of computing time and used area. To validate these procedures, we execute experiments with a case of study from literature: gcut13 (Beasley 1985). This case was selected because the container specified is square, as the available material for cutting in the laboratory. The results are presented in the section 6 of the paper.

5.2 Development of the Forward Kinematic Model (FKM)
This procedure is decomposed in three phases: (1) defining the morphology of the robot and assigning its frames, (2) computing the DH parameters, (3) computing the forward kinematic model.

5.2.1 Defining the morphology of the robot and assigning its frames
The Yaskawa Motoman HP-20D consists of a base link, which is fixed, and other six links which are movable. The sixth link will be called an end-effector. The tool is attached to this link. The neighboring links are connected by revolute joints. Fig. 5a illustrates the structural schema of the manipulator with its six degrees of freedom. According to the indications in (Spong et al. 1989) we can assign the coordinate system illustrated in Fig. 5b. This figure represents the home position of the robot; the $y$ axis is not illustrated for easy of reading, this axis is established to complete a right hand frame between $x$ and $z$.

5.2.2 Computing the DH parameters
Applying definitions in (Spong et al. 1989) to the Fig. 5 we obtain the parameters in Table 1, where the sub-index $i$ represents the parameters associated with the link $i$. Table 3 presents the values of parameters for the model HP20D from Yaskawa (Motoman a Yaskawa Company 2015).

5.2.3 Computing the forward kinematic model
Applying the DH parameters to equations (1) and (3) it is possible to obtain expressions to describe position and orientation of each $i$ link frame relative to $(i-1)$. The set of transformation matrix obtained is presented in equations (7) to (12). With those expressions, it is possible to compute the matrix $T^{p}_{i}$ as expressed in equation (13). This equation represents the forward kinematic model.

5.3. Post-optimization algorithm
This algorithm improves the performance of the cutting process through the elimination of repeated cuttings-paths. The performance is measured as a function of process-time and accuracy in the cut piece. The algorithm is based in the iterative seek of cutting-paths that have intersections and present the same orientation. For those cutting-paths, the algorithm makes a join in a single cutting-path. For this purpose, the rectangles are decomposed in two cutting-path lists (vertical and horizontal); each one representing a tow lines. Those lists are created with a sorting criteria for reduce the computational effort during the search. Then a join process is applied for the cutting-paths where the intersection exists. The pseudo-code of the procedure for the horizontal-case is illustrated in Fig. 7. This procedure receives as input the cutting-pattern computed for the GRASP algorithm. A cutting-pattern is composed by a list of pieces arranged in a Cartesian plane, i.e. a list of rectangles. Each rectangle is decomposed in four cutting paths (two horizontal and two vertical). Then, the algorithm seeks the duplicity in cutting paths and for those cases performs a join operation. This algorithm returns a minimal list of cutting-paths for the cutting-pattern at the input.
computeAndSortCpaths. Receive two arguments: the list of rectangles with the cutting-pattern computed by the GRASP algorithm and the split-criteria. This method splits the input list in horizontal or vertical cutting-paths, based on its second input argument. For the case of horizontal split-criteria, the method maps each rectangle into two horizontal cutting-paths, and then stores those paths into a non-ascending ordered list. The order criteria is the value of hCut.y

isThereIntersection. This method identifies the existence of intersection between two cutting-paths (A, B) with the same orientation. Returns a true value when exists intersection. There are five cases of intersection presented in Fig. 6.

joinCpaths. This method is designed to join two cutting-paths that intersect, forming a single cutting-path. If the first cutting-path is contained in the second cutting-path, then the first cutting-path disappears as illustrated in cases three and four of Fig. 6.

Table 1. Denavit-Hartenberg parameters for the Motoman HP20D. Classical Convention

<table>
<thead>
<tr>
<th>Link</th>
<th>( \alpha_i )</th>
<th>( a_i )</th>
<th>( \theta_i )</th>
<th>( d_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \pi/2 )</td>
<td>( a_1 )</td>
<td>( \theta_1 )</td>
<td>( d_1 )</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>( a_2 )</td>
<td>( \theta_2 + \pi/2 )</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>( \pi/2 )</td>
<td>( a_3 )</td>
<td>( \theta_3 )</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>(-\pi/2)</td>
<td>0</td>
<td>( \theta_4 )</td>
<td>( d_4 )</td>
</tr>
<tr>
<td>5</td>
<td>( \pi/2 )</td>
<td>0</td>
<td>( \theta_5 - \pi/2 )</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>( \theta_6 )</td>
<td>( d_6 )</td>
</tr>
</tbody>
</table>

Table 2. Values of DH parameters different of zero. Where \( d \) is an extra length associated with the end-effector (Motoman a Yaskawa Company 2015).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>150</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>760</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>140</td>
</tr>
<tr>
<td>( d_1 )</td>
<td>505</td>
</tr>
<tr>
<td>( d_4 )</td>
<td>795</td>
</tr>
<tr>
<td>( d_6 )</td>
<td>105+d</td>
</tr>
</tbody>
</table>

\[
T_{1,\text{classical}}^0 = \begin{bmatrix}
    c_1 & 0 & s_1 & a_1c_1 \\
    s_1 & 0 & -c_1 & a_1s_1 \\
    0 & 1 & 0 & d_1 \\
    0 & 0 & 0 & 1
\end{bmatrix}

(7) \quad T_{2,\text{classical}}^1 = \begin{bmatrix}
    -s_2 & -c_2 & 0 & -a_2s_2 \\
    c_2 & -s_2 & 0 & a_2c_2 \\
    0 & 0 & 1 & 0 \\
    0 & 0 & 0 & 1
\end{bmatrix}

(8)
\[ T_{5}^{\text{classical}} = \begin{bmatrix} c_4 & 0 & s_4 & a_s c_3 \\ s_4 & 0 & -c_3 & a_s s_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

\[ T_{6}^{\text{classical}} = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \]

with,

\[
\begin{align*}
T_{1} & = \left( s_1 s_4 - c_1 c_2 s_3 \right) k_2 - c_1 c_4 s_2 s_3 k_6 + (s_1 c_4 + c_1 c_4 s_2 s_3) k_6 \\
T_{2} & = \left( -s_1 s_4 - c_1 c_2 c_3 \right) s_3 - c_1 c_3 s_2 s_3 k_6 + (s_1 c_4 + c_1 c_3 s_2 s_3) k_6 \\
T_{3} & = \left( c_1 c_2 s_3 - s_1 s_3 \right) j_2 + c_1 s_4 c_3 \\
T_{4} & = \left( c_1 c_2 s_3 + s_1 s_3 \right) j_3 - s_4 s_6 c_23 \\
T_{5} & = \left( -c_1 s_3 s_4 - c_1 s_2 c_3 \right) k_6 - s_4 s_6 c_23 \\
T_{6} & = s_3 s_23 - c_4 c_23
\end{align*}
\]

\[ r_{1} = \left( s_1 s_4 - c_1 c_2 s_3 \right) k_2 - c_1 c_4 s_2 s_3 k_6 + (s_1 c_4 + c_1 c_4 s_2 s_3) k_6 \\
\]

\[ r_{2} = \left( -s_1 s_4 - c_1 c_2 c_3 \right) s_3 - c_1 c_3 s_2 s_3 k_6 + (s_1 c_4 + c_1 c_3 s_2 s_3) k_6 \\
\]

\[ r_{3} = \left( c_1 c_2 s_3 - s_1 s_3 \right) j_2 + c_1 s_4 c_3 \\
\]

\[ r_{4} = \left( c_1 c_2 s_3 + s_1 s_3 \right) j_3 - s_4 s_6 c_23 \\
\]

\[ r_{5} = \left( -c_1 s_3 s_4 - c_1 s_2 c_3 \right) k_6 - s_4 s_6 c_23 \\
\]

\[ r_{6} = s_3 s_23 - c_4 c_23
\]

Data List that describes the cutting-pattern computed for the GRASP algorithm.

Result Minimal list of cutting-paths that describes the cutting-pattern at the input rectangle. A rectangle is described for two corners (lower left, upper right). Each corner is described for two coordinates.

Structures A cut is described for three paths; this is a consequence of the repetition of one of its paths.

\[
\begin{array}{c}
\text{case 1} \\
\text{case 2} \\
\text{case 3} \\
\text{case 4} \\
\text{case 5}
\end{array}
\]

Fig. 6. Cases of intersection of cutting-paths
procedure joinHorizontalCuts (I List: rectangleList; O List: horizontalCutList)

start

horizontalCutList ← computeAndSortCpaths(rectanglesList, "horizontal")

for ih in 0,1,2…length(horizontalCutList) do

for ihaux in 0,1,2…length(horizontalCutList) do

if ih=ihaux then

continue

end if

if horizontalCutList[ih].y= horizontalCutList[ihaux].y then

if isThereIntersection(horizontalCutList[ih], horizontalCutList[ihaux]) then

joinCpaths(horizontalCutList[ih], horizontalCutList[ihaux])

end if

else

break

end if

end for

end for

return horizontalCutList
end

6. Results

The results were computed in the cutting system using the architecture in Fig. 1. The architecture is composed of five blocks, these blocks are connected to one another through software or hardware interfaces. The system developed in this project involves the first four blocks; the last block is supplied by the Robot vendor. The input is composed of descriptors of the rectangular container and rectangular pieces to be cut. This information is processed before deliver to the GRASP algorithm. The pre-processing is necessary to eliminate repetitions, configure clearances and validate data format. Then, the GRASP algorithm computes the optimal distribution of pieces in the rectangular container; the result is expressed as a set of rectangles located at the space. This result is processed before deliver to the Code generator. In this post-processing we can alter the order of cut of the rectangles and eliminate duplicate cutting-paths with the post-optimization algorithm. Then the Code Generator computes the kinematic models for each position and orientation involved in the cutting task and assembles a code using the native language of the robot Controller. This code is loaded into the Robot controller and the cutting task is performed. The element for cutting is a plasma cutter. The developed system is built up in C++ language.

6.1. Validation of the Forward Kinematic Model (FKM)

Table 3. Test values for articular variables. The forward kinematic model implemented in the controller uses alternative variables for $\theta_2$ and $\theta_5$. The relation between alternative variables and ordinary variables was concluded during experimentation in the Laboratory and is expressed in the two last columns of Table 5. We assume that this relation is a consequence of the value of parameter $\theta$ in the Table 1 for links two and five.

<table>
<thead>
<tr>
<th>Test Value</th>
<th>$\theta_1$</th>
<th>$\theta_2$</th>
<th>$\theta_3$</th>
<th>$\theta_4$</th>
<th>$\theta_5$</th>
<th>$\theta_{2alt} = \theta_2 - \pi/2$</th>
<th>$\theta_{5alt} = \theta_5 + \pi/2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>$\pi/2$</td>
<td>0</td>
<td>0</td>
<td>$-\pi/2$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>$\pi/2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$\pi/2$</td>
<td>$\pi/2$</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>$\pi/2$</td>
<td>$\pi/2$</td>
<td>$\pi/2$</td>
<td>$\pi/2$</td>
<td>0</td>
<td>$\pi$</td>
</tr>
<tr>
<td>4</td>
<td>$\pi/2$</td>
<td>0</td>
<td>$\pi/2$</td>
<td>0</td>
<td>$\pi/2$</td>
<td>0</td>
<td>$\pi/2$</td>
</tr>
</tbody>
</table>

This validation consist in feed the available FKM models with a set of test values and compare the resulted position and orientation. The available models are: (i) computed model as presented in equation (13) and simulated in the toolbox developed by Peter Corke (Corke 2011), (ii) MotoSim model as defined in the simulation software of Yaskawa with version EG 5.2 (Motoman a Yaskawa Company 2016), (iii) ) controller model as implemented in the DX100 Controller (Motoman a Yaskawa Company 2015).

We selected four test values as summarized in Table 3. For each value and each model we measured position an orientation. Then we compute the relative error of the computed model with respect to: (i) MotoSim model, (ii) Model of the DX100 Controller. This error was computed with equation 14 and is presented in Tables 4 and 5. These data allow concluding that the computed model coincides with the model implemented in the software MotoSim but differs with the model implemented in the DX100 controller.

\[ e_y = \frac{100 \cdot \text{abs}(y' - y)}{y_{\text{max}} - y_{\text{min}}} \]

(14)
Where, $y^*$ is the reference value, $y$ is the measured value, $y_{\text{max}}$ and $y_{\text{min}}$ are the maximal and minimal value of the variable, respectively.

Table 4. Positions Values obtained

<table>
<thead>
<tr>
<th>Test Value</th>
<th>Position (mm)</th>
<th>Simulation Scene</th>
<th>Relative Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Computed Model</td>
<td>MotoSim [17] DX100 Controller</td>
</tr>
<tr>
<td>1</td>
<td>X 945,000</td>
<td>945,000</td>
<td>956,703</td>
</tr>
<tr>
<td></td>
<td>Y 0,000</td>
<td>0,000</td>
<td>-5,673</td>
</tr>
<tr>
<td></td>
<td>Z 795</td>
<td>794,990</td>
<td>685,730</td>
</tr>
<tr>
<td>2</td>
<td>X 1050</td>
<td>1050,002</td>
<td>1159,270</td>
</tr>
<tr>
<td></td>
<td>Y 0</td>
<td>0,000</td>
<td>-5,673</td>
</tr>
<tr>
<td></td>
<td>Z 900</td>
<td>899,987</td>
<td>911,703</td>
</tr>
<tr>
<td>3</td>
<td>X 10</td>
<td>10,010</td>
<td>-5,670</td>
</tr>
<tr>
<td></td>
<td>Y 1050</td>
<td>1050,000</td>
<td>1159,270</td>
</tr>
<tr>
<td></td>
<td>Z 900</td>
<td>900,000</td>
<td>888,297</td>
</tr>
</tbody>
</table>

Table 5. Orientation values obtained

<table>
<thead>
<tr>
<th>Test Value</th>
<th>Orientation (degrees)</th>
<th>Simulation Scene</th>
<th>Relative Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Computed Model</td>
<td>MotoSim [17] DX100 Controller</td>
</tr>
<tr>
<td>1</td>
<td>$\gamma$ 180,00</td>
<td>180,00</td>
<td>180,00</td>
</tr>
<tr>
<td></td>
<td>$\beta$ 0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td></td>
<td>$\alpha$ 0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>2</td>
<td>$\gamma$ 180,00</td>
<td>180,00</td>
<td>180,00</td>
</tr>
<tr>
<td></td>
<td>$\beta$ -90,00</td>
<td>-90,00</td>
<td>-90,00</td>
</tr>
<tr>
<td></td>
<td>$\alpha$ 0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>3</td>
<td>$\gamma$ 90,00</td>
<td>90,00</td>
<td>90,00</td>
</tr>
<tr>
<td></td>
<td>$\beta$ 0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td></td>
<td>$\alpha$ 0,00</td>
<td>0,00</td>
<td>0,00</td>
</tr>
<tr>
<td>4</td>
<td>$\gamma$ -90,00</td>
<td>-90,00</td>
<td>-90,00</td>
</tr>
<tr>
<td></td>
<td>$\beta$ 90,00</td>
<td>90,00</td>
<td>89,999</td>
</tr>
<tr>
<td></td>
<td>$\alpha$ 0,00</td>
<td>0,00</td>
<td>180</td>
</tr>
</tbody>
</table>

6.2 Validation of the GRASP algorithm.

Figure 8 presents the cutting pattern computed by the GRASP algorithm and Table 6 presents the performance in contrast with two results reported in literature. These results are obtained processing the benchmark gcut13 available at (Beasley 1985). This benchmark were used in other publications as indicated in Table 6 (Cintra et al. 2008).

Table 6. Comparison between solution computed by the GRASP algorithm and two reported solutions to the case gcut13

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Optimal Solution</th>
<th>Area (%)</th>
<th>Time (sec)</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRASP</td>
<td>8971544</td>
<td>99,68</td>
<td>3,17</td>
<td>GRASP proposed in this work</td>
</tr>
<tr>
<td>DP 2-Staged</td>
<td>8906216</td>
<td>98,958</td>
<td>21,82</td>
<td>(Cintra et al. 2008)</td>
</tr>
<tr>
<td>DP 4-Staged</td>
<td>8997780</td>
<td>99,975</td>
<td>43,72</td>
<td>(Cintra et al. 2008)</td>
</tr>
</tbody>
</table>

To perform the cutting path, we used a plasma cutter: Cebora Plasma Prof 163 (Cebora S. P. A. 2015). The dimensions of the benchmark were scaled to fit a container of 700x700mm. The material to be cut is cold-rolled steel with Gauge 18. The plasma cutter was configured with the parameters indicated in the Table 7. The velocity of cutting was set to 100mm/s. At the first experiment, the
computed pattern was implemented without post-processing. At the second experiment we apply the post-optimization processing before cut the material. The length and width of the obtained pieces were measured and compared with the references values by means of relative error. The results of comparisons are illustrated in Figure 9 and Table 8.

Table 7. Configuration Parameters for the plasma cutter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>38</td>
<td>Amps</td>
</tr>
<tr>
<td>Pressure</td>
<td>45</td>
<td>psi</td>
</tr>
<tr>
<td>Pierce Height</td>
<td>3</td>
<td>Mm</td>
</tr>
<tr>
<td>Shield</td>
<td>Art 1997</td>
<td>/</td>
</tr>
<tr>
<td>Torch type</td>
<td>CP161-40A</td>
<td>/</td>
</tr>
</tbody>
</table>

Table 8. Comparison of indicators of cutting

<table>
<thead>
<tr>
<th>Version</th>
<th>Cutting time (seg)</th>
<th>Relative Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No post-optimized</td>
<td>579</td>
<td>3.8897</td>
</tr>
<tr>
<td>Post-optimized</td>
<td>218</td>
<td>3.0452</td>
</tr>
</tbody>
</table>

Fig. 9. Comparison of relative errors for length (L) and width (W). The tag Classic indicates that the results were obtained without post-processing stage, the tag postOpt indicates that the results were obtained with post-optimization algorithm

7. Conclusions

In this paper we developed and validate two stages of an automatic cutting system to resolve the two-dimensional non guillotine single knapsack problem (2D-SKP): optimization stage, code-generator. At the optimization stage we use a Greedy-Randomized-Adaptive-Search-Procedure (GRASP); an iterative procedure that resolves the location of each piece in the container. This procedure was validated by using a benchmark and comparing: computation time, used area of the solution. The results allow concluding the correctness of the procedure and a superior performance in computation time. Future works will explore this superiority with new benchmarks before generalization.

At the code generation stage we use the forward kinematic model (FKM) of the manipulator; this model allows computing the position and orientation of the end-effector from specified values for the joint variables; we used a six DOF robot: HP20D from Motoman. The computed model presented in equation (13) was compared with two references models: model used in the MotoSim software simulator, model implemented in the robot Controller DX100. The results allow concluding the correctness of the computed model with respect to the model implemented in the MotoSim software simulator, but a difference with respect to the model implemented in the robot Controller DX100. The differences between the computed model and the model implemented in the robot Controller DX100 are due to variations in the orientation selected for the frames during the development (see Fig. 5b). Future works for parametrization consist in modify the orientations of the frames used in this work, until get coincidence with the model in the DX100 controller

The developed stages were evaluated in integration with an automatic cutting system. The overall system is composed by a Greedy-Randomized-Adaptive-Search-Procedure (GRASP) code-generator, an industrial Robot and a plasma cutter. The computed solution for a benchmark allows concluding the correctness of the overall system. Future works consist in evaluate the performance of the automatic cutting system. We proposed a post-processing stage to improve the cutting performance. The post-processing stage increases the performance for the case-study with a reduction of 37.65% for cutting time and a reduction of 0.84 units in accuracy, in terms of relative error. These indicators are valid for the benchmark obtained and must be examined with detail in future works.
7. References
Motoman a Yaskawa Company, 2015. General purpose and handling with the HP-series.
Sicialiano, B. et al., 2009. Robotics Modelling, Planning and Control Springer, ed.,