ARTICLE TEMPLATE

Automatic computation of bending sequences for wire bending machines

Andrea Baraldo[a], Luca Bascetta[b], Fabrizio Caprotti[a], Sumit Chourasiya[b], Gianni Ferretti[b], Angelo Ponti[a] and Basak Sakcak[c]

[a]BLM SPA - BLM GROUP, via Selvaregina 30, 22063 Cantù (CO), Italia
[b]Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo da Vinci 32, 20133 Milano, Italia
[c]Center of Ubiquitous Computing, Faculty of Information Technology and Electrical Engineering, University of Oulu, Finland

ARTICLE HISTORY
Compiled September 26, 2022

ABSTRACT
Determining a feasible bending sequence, i.e., ensuring absence of wire self-collisions and wire-machine collisions, or even an optimal bending sequence, i.e., minimising time or energy required to perform it, can be a difficult and time consuming task for complex workpieces, even for an expert operator. The introduction of algorithms for the computation of wire bending sequences is thus crucial to increase productivity and production flexibility, and to decrease production costs. To this aim, this work proposes an algorithm to automatically determine an appropriate bending sequence for a given workpiece, bending tool, and machine 3D CAD model, that leverages on a representation of the wire as a robotic manipulator and of a bending sequence as a tree, and on the adoption of A* as graph search algorithm. A cost and a heuristic function, suitable for the wire bending problem, and an approach to parallelise the execution of A* are introduced, as well. In this way, a computationally simple and efficient wire bending sequence computation algorithm can be devised, able to determine a solution in an amount of time less or equal to the time used by an expert operator, without the need of high computational power. Finally, the effectiveness of this algorithm is assessed on two different test cases, relevant to industrial workpieces.

KEYWORDS
Bending sequence computation; bending sequence optimisation; wire bending sequence

1. Introduction

A bending machine is a forming machine tool that allows to manufacture a workpiece by using a bending tool and an ordered sequence of linear and rotational motions. The shape of the manufactured workpiece depends on the characteristics of each motion, i.e., position and velocity profiles, and on the sequence in which motions are executed. In the case of complex workpieces, an expert operator and a significant amount of time may be required to determine a sequence that allows to manufacture the workpiece, i.e., not all the sequences are always feasible, without causing self-collisions,

CONTACT Luca Bascetta. Email: luca.bascetta@polimi.it

or collisions with the machine or the bending tool, without introducing deformations in the workpiece shape due to material elasticity, and without generating too much vibrations, as they force to slow down the bending process.
For these reasons, devising an algorithm whose computational complexity is limited, so that it can be executed in a reasonable amount of time (i.e., an amount of time that should be less or equal to the time used by an expert operator to determine a solution) without the need of high computational power, that automatically determines an appropriate bending sequence for a given workpiece, bending tool, and machine 3D CAD model, allows to increase the productivity and the production flexibility, and to decrease the production costs.

The problem of automatic computation of feasible or optimal bending sequences has been already considered in the literature, in particular with reference to the sheet metal bending problem (Hoffmann, Geiler, and Geiger 1992; Shpitalni and Saddan 1994; De Vin et al. 1994).
In this context, different approaches based on a tree representation of the bending sequence and on the application of A*, or the travelling salesman problem algorithm, have been proposed (Faraz et al. 2017; Zhao, Zhang, and Shi 2014; Duflou, Kruth, and Van Oudheusden 1999; Duflou et al. 1999; Gupta et al. 1998). In particular, (Gupta et al. 1998) introduces a distributed planning architecture, composed of a central operation planner and three domain-specific planners; the operational planner is based on A* and leverages on domain-specific knowledges that are strictly related to the sheet metal bending problem. In (Zhao, Zhang, and Shi 2014), instead, the sequence planning is converted to a generalised shortest path problem, and an A* search is performed, whose distinguished element is a heuristic based on bending feasibility, dimensional accuracy and processing efficiency. Following a similar philosophy, in (Duflou, Kruth, and Van Oudheusden 1999; Duflou et al. 1999) and (Faraz et al. 2017) bending sequences are represented as a tree, consequently the best sequence computation is formulated as a travelling salesman problem and solved using a branch-and-bound technique based on manufacturing knowledge that is specific to sheet metal bending.
Two more approaches, based on a graph search algorithm, must be mentioned. The first one, presented in (Raj Prasanth and Shunmugam 2020), introduces a two-stage algorithm, in which the second stage is based on a best-first search algorithm. The second one, reported in (Markus, Váncza, and Kovács 2002), solves the bending sequence planning problem using a constraint-based planning approach.
Graph search is not the only technique considered in the literature relevant to sheet metal bending planning, but a number of different approaches based on stochastic search (Kannan and Shunmugam 2008) have been proposed, as well. In particular, in (Ong et al. 1997) an algorithm based on fuzzy set theory is presented, while in (Thanapandi, Walairacht, and Ohara 2001a,b) and (Kannan and Shunmugam 2008) optimal and near optimal bending solutions, based on the application of genetic algorithms, are proposed.
A further and last approach, presented in (Lin and Chen 2014; Lin and Sheu 2012; Rico et al. 2003), is based on the decomposition of the bending sequence into a series of basic predefined bending patterns, i.e., basic shapes like channels of spirals, each one with an associated operation rule.
To complete the state of the art on the automatic computation of bending sequences, it must be noticed that a simulation based bending sequence planning has been proposed in (Inui and Terakado 1998, 1999), while (Koguchi, Aomura, and Igoshi 2000; Koguchi and Aomura 2002; Aomura and Koguchi 2002) focused on the bending sequencing problem considering a robotised bending solution.

The analysis of the state of the art reveals that

- many works can be found in the literature focusing on the automatic computation of bending sequences for the specific application of sheet metal bending;
- a relevant number of these approaches leverages on a graph or tree representation of the bending sequence and on graph search algorithms, like A*, supported by a suitable heuristic function;
- almost all the approaches in the literature exploit domain specific knowledge, e.g., in the formulation of the heuristic, related to sheet metal bending.

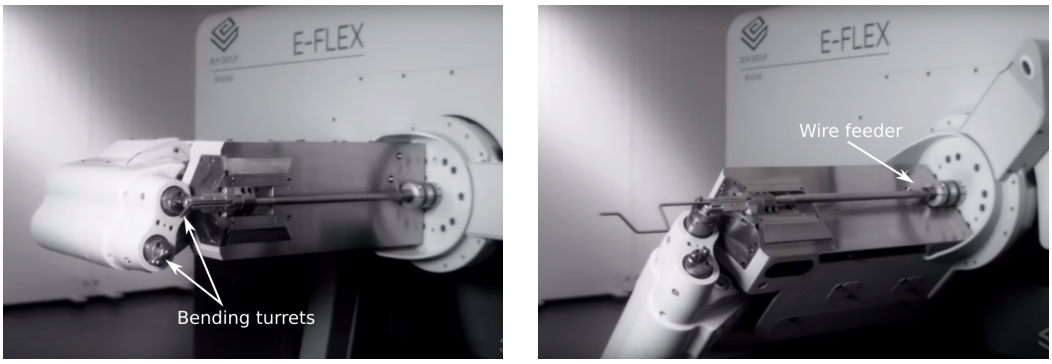
Furthermore, considering that sheet metal bending and wire bending are two different processes, each one characterised by its own peculiarities – e.g., sheet metal bending is mainly a 2D, while wire bending is a 3D process; wire is bent clockwise and counter-clockwise; wire has always one free and one constrained end, while metal sheet can be bent on both ends, etc. –, the approaches previously mentioned cannot be directly applied or easily adapted to the wire bending problem.

This paper introduces a novel approach for the automatic computation of bending sequences for wire bending machines, based on a tree representation of the bending operations and exploiting A* as graph search algorithm, whose main distinguished features are:

- wire is represented as a manipulator, where each bend is a rotational joint;
- bending sequences are represented in reverse order in a straightening tree;
- a simple cost and a heuristic function, that allow to efficiently search the tree using A*;
- a divide-et-impera approach to parallelise and speed up the execution of A*.

To appreciate the advantage of this approach, one has to consider that in industrial practice optimal bending sequences are determined case-by-case by expert machine operators, following a very time-consuming procedure. In fact, the operator starts from the bending sequence suggested by the CAD/CAM software, and follows an iterative trial-and-error process to determine the optimal bending sequence, using a suitable kinematic simulation environment to verify each attempt.
The proposed algorithm, instead, automatically determines an optimized solution, starting from a description of the workpiece geometry and a mathematical representation of the concept of "best bending sequence", in terms of a suitable cost function. It thus allows to avoid the case-by-case trial-and-error procedure, significantly reducing the time required to setup the machine for the production of a new workpiece (Baraldo et al. 2021) (consider that, according to industrial practice, the time required by an expert operator to determine a good bending sequence for a complex workpiece can vary from hours to half a day).

The paper is organised as follows. Section 2 introduces the bending process using a single-head bending machine. In Sections 3 and 4 the approach adopted in this work to model the wire and a bending sequence is presented. Section 5 proposes a methodology to compute a set of ordered feasible bending sequences, based on a novel heuristic function and an approach that allows to simplify and parallelise the computation, speeding up the process. In Section 6 two different case studies, based on industrial workpieces, are presented to demonstrate the effectiveness of the algorithm.



(a) (b)

Figure 1. An example of single-head bending machine.

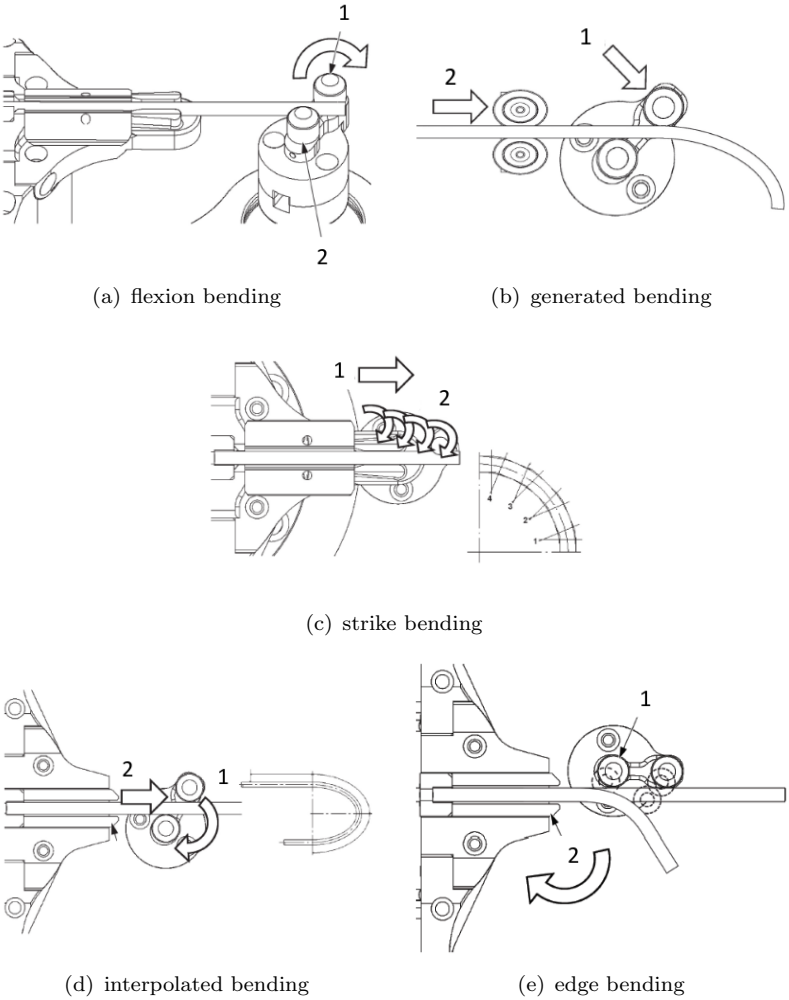
2. Single-head bending machine

A wire bending machine is a forming machine tool that allows to manufacture a workpiece by using a bending tool and an ordered sequence of linear and rotational motions. The shape of the manufactured workpiece depends on the characteristics of each motion, i.e., position and velocity profiles, and on the sequence in which motions are executed. The machine consists of many functional subsystems (Figure 1) that enable turret rotation, workpiece rotation, wire feeding, wire bending, and so on. In particular, bending is performed by rotating a bending arm that pushes the wire to the desired bend angle around a pivot pin.
Bending machines can be mainly classified into two types: single-head machines, that can execute only one bend at a time, even if they have more than one turret; double-head machines, that can execute two bends simultaneously, using two different turrets. This work considers only the case of single-head bending machines.

The single-head wire bending machine considered here as a case study is configured with two bending turrets (Figure 1(a)), which can rotate around an axis parallel to the bending axis. Moreover, the whole head can rotate around an axis parallel to the wire feeder axis (Figure 1(b)), thus allowing for 3D bends. Wires up to 10 mm in diameter can be processed.
Note that, though the case study concerns a wire bending machine, the methodology presented here is general, and can be applied either to wire or pipe bending.

Five different types of wire bending are considered:

- flexion bending, is realised by the folding roller (Figure 2(a), no. 1), rotating by a given angle with respect to the central peg (Figure 2(a), no. 2), with the wire fixed, folding roller and central peg form the folding arm;
- generated bending (variable radius), is obtained pushing the wire through the folding arm, the bending radius can be modified by simultaneously varying the angle of the folding arm (Figure 2(b), no. 1) and the displacement of the wire (Figure 2(b), no. 2);
- strike bending, allows to obtain a variable radius (Figure 2(c)) through a sequence of repeated flexion bendings, each one with an incremental displacement of the wire (Figure 2(c), no. 1) and a different folding arm angle (Figure 2(c), no. 2);
- interpolated bending, is similar to generated bending, but with a fixed folding arm angle (Figure 2(d));



(a) flexion bending

(b) generated bending

(c) strike bending

(d) interpolated bending

(e) edge bending

Figure 2. Types of bends.

- edge bending, is realised by pushing the wire against the central peg, while using the folding roller to curve the wire, without passing between them (Figure 2(e)).

The choice of the type of bend depends on many factors, like operator experience, mechanical strength, and geometry of the workpiece. To simplify the modelling operation, all different bend types are here approximated through one or a group of more than one flexion bends, as it will be clarified in Section 3.

3. Bending wire modelling

The first step to introduce a sequence planning algorithm concerns the technique adopted to represent the bending sequence. One possible indirect solution is to describe the configuration of the workpiece, as it were a manipulator with cylindrical links and rotational joints, using Denavit-Hartenberg (D-H) parameters (Figure 3 shows notation, joint variables and parameters, as they are introduced in the standard D-H convention (Siciliano et al. 2009) for a generic manipulator).
Note that joints are numbered in reverse order with respect to the classical D-H notation, each joint corresponds to a bend, and each bend is denoted by an integer b that represents the number associated to the bend in the sequence generated by a CAD/CAM software neglecting collisions and feasibility, parameters d_i are always set to zero, parameters a_i represent the displacement of the wire between bending b_{i-1} and b_i, and α_i denotes the rotation of the bending axis. Furthermore, as the wire can rotate about its longitudinal axis, a rotational joint, characterised by $d_i = a_i = \alpha_i = 0$, is added as first joint of the chain, directly connected to the fixed base. Therefore, any workpiece characterised by N bends is represented by a configuration vector of $N + 1$ elements[1]. Figure 4 exemplifies the procedure just described for the derivation of D-H parameters, in the case of a simple hook workpiece. The corresponding parameters are reported in Table 1. Note that, being the workpiece planar all α_i are equal to zero, i.e., all z-axis are perpendicular to the workpiece plane, and all d_i are equal to zero, i.e., all the frame origins lay on the workpiece plane. Finally, parameters a_i and θ_i represent the length of the wire between two consecutive bends and the bending angles, respectively.
Thanks to this reformulation of the problem, checking wire self-collisions and collisions between wire and machine components can easily be traced back to a planning problem in the presence of constraints, which can be tackled modelling the wire as a group of cylinders, and using algorithms consolidated in the literature, e.g., Gilber-Johnson-Keerthi (GJK) algorithm (Gilbert, Johnson, and Keerthi 1988) or octrees (Garcia and Le Corre 1989).

Some technological issues must be considered, regarding the elastic behaviour of the wire, and the cases of high values of the bending angle and variable radius bending.
First of all, bending angles θ_i, defining desired workpiece processing (Figure 4), may be different from those actually performed by the machine, i.e., $\hat{\theta}_i$. In fact, due to material elasticity, the bending performed by the machine has to compensate for the elastic motion of the metal wire after bending (spring back). This compensation is calculated upstream of the calculation of the possible sequences through specific corrections, accounting for material properties. For this reason, in the following only the values of angles $\hat{\theta}_i$ are reported.

[1]This additional joint, whose D-H parameters are straightforward and always constant, is not reported in D-H parameter tables.

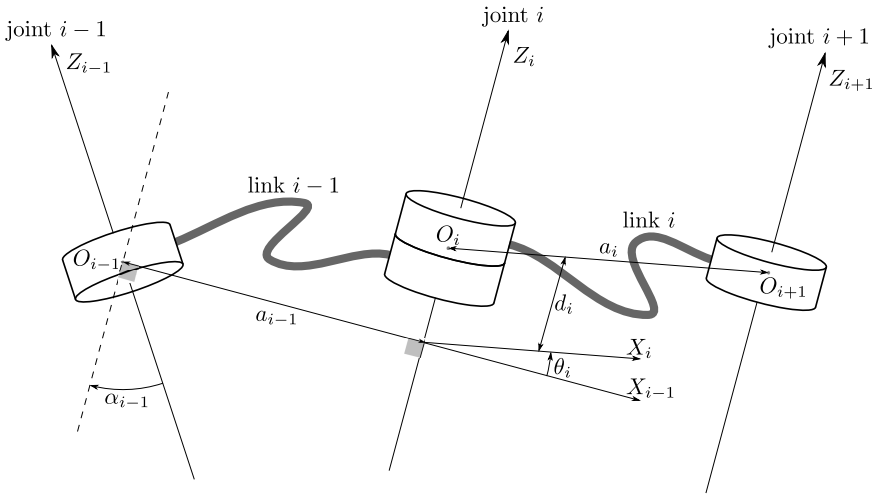


Figure 3. Notation, joint variables and parameters in D-H convention (Siciliano et al. 2009).

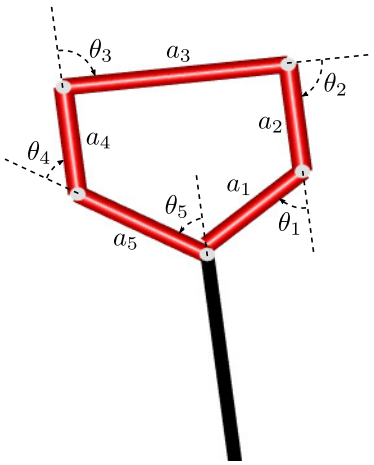


Figure 4. Hook workpiece configuration.

a_i	α_i	d_i	θ_i
25	0	0	−55
22	0	0	−90
40	0	0	−90
22	0	0	−55
26	0	0	55

Table 1. Hook workpiece D-H parameters (displacements are in mm, angles in degrees).

(a) (b)

Figure 5. High bending angle.

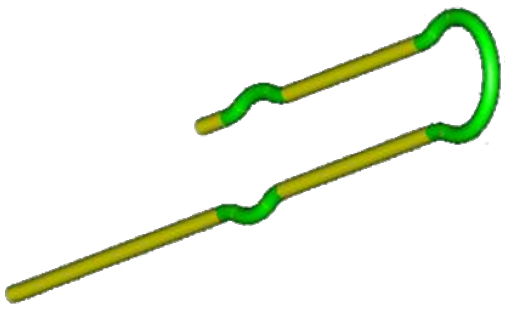


Figure 6. Clip example (green areas indicate bending operations to be performed).

	1	2	3	4	5	6	7	8	9	10
a_i	3	3	63	3	3	3	71	3	3	100
θ_i	28	−48	19	24	−48	−180	18	28	−48	19
$\hat{\theta}_i$	27	−54	27	27	−34	−200	27	27	−54	27

Table 2. D-H parameters of the clip (displacements are in mm, angles in degrees).

Secondly, in the model each bend is represented by a single rotational joint. However, for bending angles higher than a specified threshold, that here is assumed equal to 160 degrees, the diameter of the pivoting peg must be taken into account (Figure 5(a)). This can be done by splitting the bending into two consecutive operations, both with a value of the bending angle, e.g., θ_i and θ_{i+1}, equal to half of the overall angle, and with a_{i+1} equal to the diameter of the peg (Figure 5(b)). Clearly, this approximation does not reproduce the exact curvature, however, it is sufficient for collision detection, hence for verifying the feasibility of a bending sequence.
Finally, a variable radius bending can be included in this framework by modelling it as a strike bending, thus dividing the overall curve into a sequence of flexion bendings. The stroke division can be calculated upstream of the calculation of the sequences, such that the graph search algorithm treats it as a single bending, i.e., the sequence of approximating flexion bendings is respected.

As an example, Tables 2 and 3 report the D-H parameters for the clip in Figure 6 before (Table 2) and after (Table 3) the splitting operation previously mentioned, considering both final and executed bending angles, θ_i and $\hat{\theta}_i$, respectively (being the clip a planar workpiece, parameters α_i and d_i are all equal to zero and have been thus neglected). Displacements and angles are expressed in millimetres and degrees, respectively. Note that a_{11} in Table 2 (while it is a_{14} in Table 3) defines the last displacement of the wire before cutting.
Finally, Figure 7 shows a sketch of the manipulator used to represent the clip together with the corresponding D-H parameters reported in Table 3.

	1	2	3	4	5	6	7	8	9	10	11	12	13
a_i	3	3	63	3	3	3	3	3	3	71	3	3	100
$\hat{\theta}_i$	27	−54	27	27	−34	−60	−20	−60	−60	27	27	−54	27

Table 3. D-H parameters of the clip after splitting bending angles higher than 160 degrees (displacements are in mm, angles in degrees).

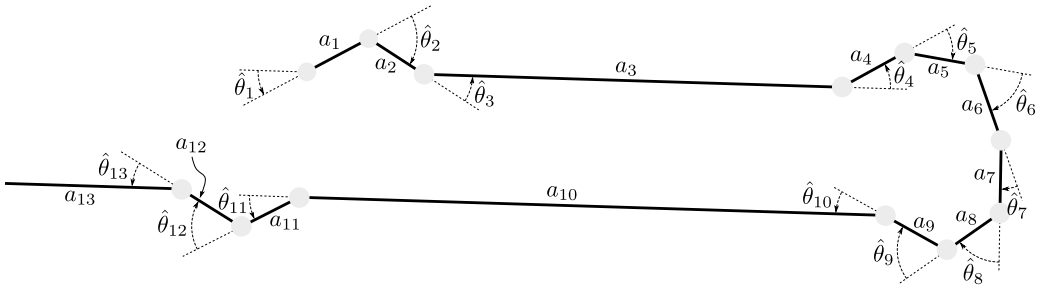


Figure 7. Sketch of the manipulator used to represent the clip (drawing proportions were modified to increase readability).

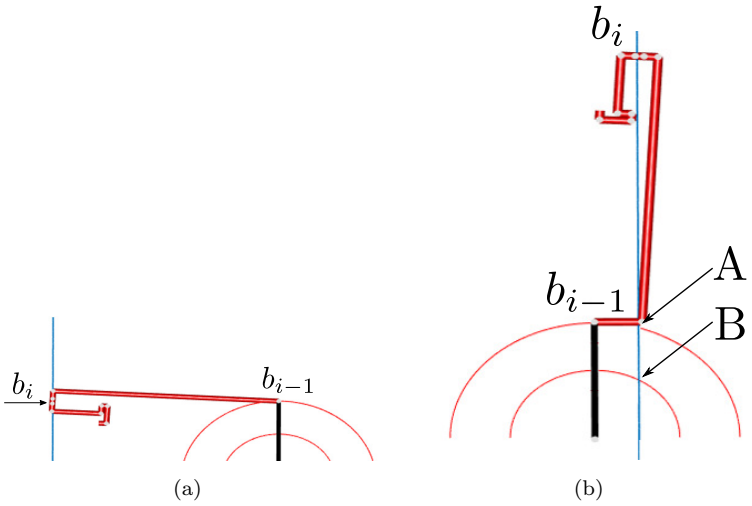


Figure 8. Examples of unreachable bends.

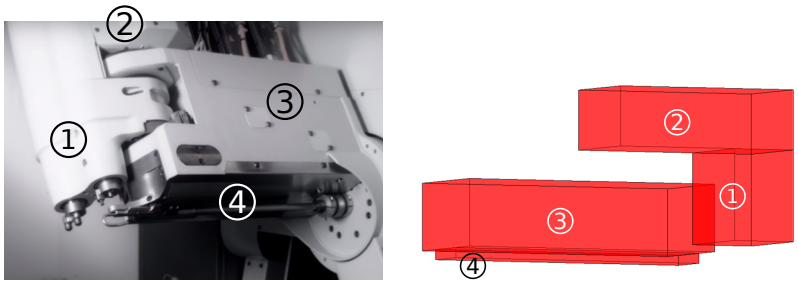
4. Bending sequence modelling

As already mentioned, the shape of a workpiece depends on the characteristics of each bend, and on the sequence with which bends are executed.
In the following, a sequence composed by N bends is denoted by

$$B^N = \{b_i, i = 1, \ldots, N\}$$

Note that, a sequence B^N is admissible if all the relevant bends b_i, $i = 1, \ldots, N$, are *reachable* and *collision-free* (see next section for a definition of these two properties).

Considering now a workpiece to be manufactured through a sequence of N bends, the $N_s \leq N!$ *admissible* sequences B_k^N, $k = 1, \ldots, N_s$, can be arranged in a tree structure, such that each path from the root of the tree to a leaf represents a possible sequence. The whole set of admissible bending sequences can be then determined by a depth-first-search over a tree. For example, one admissible bending sequence applicable for the workpiece in Figure 4 is $1, 2, 3, 4, 5$, obtained by cascading the bending angles $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$.



(a) (b)

Figure 9. Geometric model of the machine body.

4.1. *Reachability*

A bending b_i is reachable from a bending b_{i-1} if the Cartesian position of joint b_i, computable from the forward kinematics of the equivalent manipulator, can be made to coincide with the center of one of the two bending forks, by translating the wire and rotating the bending head. Considering the fact that the paths performed by the two bending forks are semicircular in the plane normal to the bending axis, the conditions for a bending to be reachable are that:

(1) there exists an intersection between a straight line, parallel to the translation axis of the wire and passing through the current position of joint b_i, and one of the two semicircular trajectories;
(2) the current distance between the position of joint b_i and this intersection is lower than the current length of the unworked wire coming out of the feeder.

Figure 8(a) shows an example in which bending $b_i = 6$ is not reachable from $b_{i-1} = 9$, because there is no intersection between the blue line, parallel to the wire translation axis, and both the trajectories of the bending tools. Figure 8(b), instead, shows an example in which the same bending is not reachable from $b_{i-1} = 10$, because the distance of the current position of joint b_i with respect to both intersections A and B is greater than the length of the unprocessed wire, denoted by the black line.

4.2. *Collision checking*

Another important property to guarantee that a bending sequence is admissible is related to the absence of collisions among links of the equivalent manipulator describing the wire configuration, as well as among the said links and the machine body.

The geometry of the wire can be approximated as a sequence of cylinders. The 3D pose of each cylinder is computed through the forward kinematics of the equivalent manipulator, whose joint positions evolve according to the sequence of bends.
The length of the workpieces prevents collisions with static parts of the bending machine. Therefore, it was decided to check collisions only between the wire and the moving parts of the machine. The geometric model of the machine is shown in Figure 9, and it is composed of four bodies, corresponding to:

(1) bending turrets;
(2) motor housing;
(3) rotating support;
(4) wire feeder.

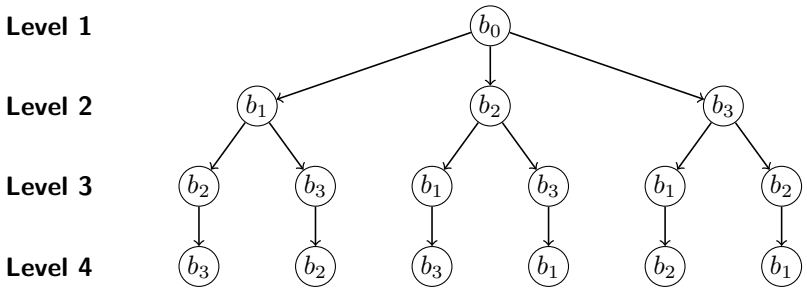


Figure 10. Bending sequences tree.

Each part of the geometric model is therefore defined as a box that envelopes the real component, so that it can be described using only the coordinates of the eight vertices, slightly overestimating the occupied space.

Collisions cannot be checked only for the start and end configurations associated to each bend, but along the entire trajectory covered by the wire. To this aim, each bending angle θ_i is divided into I_i intervals, so that $I_i + 1$ configurations are checked for collisions, each one corresponding to an increment of θ_i / I_i. In order to limit the number of collision checks, the number of intervals is decided on the basis of a heuristic: if the length of the portion of processed wire, i.e., the portion of wire that came out the feeder, is less than 100 mm, 10 intervals are considered, otherwise 20.

4.3. *Bending sequences tree*

In order to enumerate N_s admissible permutations of N bends, a tree representation is adopted[2] (Bhattacharya 1994) (Figure 10 shows an example for 4 bends), where:

- each node corresponds to a bend, and is represented by the number associated to the bend in the sequence generated by a CAD/CAM software neglecting collisions and feasibility;
- each node has a number of children equal to the difference between the number of bends in the sequence and the node level;
- all child nodes correspond to bends not already considered in parent nodes;
- each path from the root to one leaf must include all bends in the sequence.

Once the bending sequences tree has been computed, the set of admissible bending sequences could be determined through a depth-first-search algorithm, though, as it is explained in the following, the dimensionality of the problem advises against this approach. When exploring each node the admissibility checks for the corresponding bend are performed and, in the case they yield a positive result, the bend is added to the sequence, otherwise not only the current node but all its child nodes are pruned from the tree.
As proposed by Inui and Terkado (Inui and Terakado 1999), rejection of certain nodes and therefore paths are more likely to occur if bending sequence is traversed in a reverse manner. Reverse tree traversal means that the wire is assumed to be at its final shape and bends are straightened one-by-one, thus transforming the process into

[2]Note that, using a tree to represent all the permutations of a set of N bends is an intuitive and convenient way to convey the algorithm for the automatic computation of bending sequences. As far as the algorithm implementation is concerned, however, different techniques can be adopted to determine and store the permutations, e.g., (Fisher and Yates 1948; Sedgewick 1977; Heap 1963), aiming at reducing computational time or memory consumption.

	1	2	3	4	5	6	7	8	9
a_i	11	13	60	30	10	25	25	100	50
α_i	0	0	45	90	0	0	0	0	0
$\hat{\theta}_i$	-245	40	90	-45	45	90	90	90	90

Table 4. D-H parameters of the 3D eyelet (displacements are in mm, angles in degrees).

	1	2	3	4	5	6	7	8	9	10	11
a_i	8	8	10	13	60	30	10	25	25	100	50
α_i	0	0	0	0	45	90	0	0	0	0	0
$\hat{\theta}_i$	-65	-90	-90	40	90	-45	45	90	90	90	90

Table 5. D-H parameters of the 3D eyelet after splitting bending angles higher than 160 degrees (displacements are in mm, angles in degrees).

a straightening instead of a bending.

The dimensionality of the problem can be also largely reduced when rotations of the bending axis are required. In fact, assume that for bend b_i angle α_i is different from zero, in this case all bends b_j, with $j < i$, in the sequence must be performed before bend b_i, as the wire cannot be retracted after bend b_i.
For example, the eyelet in Figure 11, whose D-H parameters are reported in Tables 4 and 5 (before and after the splitting of bending angles higher than 160 degrees, respectively) and shown in the manipulator sketch of Figure 12, is a non-planar workpiece. In fact, after the first two bends (with reference to Table 4) the bending plane changes twice its orientation of 45 and 90 degrees, respectively, as reported by the values of parameters α_3 and α_4. As a consequence of the rotation of the bending axis required by bends 3 and 4, bends 1 and 2 must be executed first ($\{b_1, b_2\} = \{1, 2\}$), then bends 3 and 4 sequentially, and then a permutation of the remaining 5 bends must be explored. The total number of nodes is therefore $2 \cdot 5! = 240$, while the total number of sequences with 9 bends is $9! = 362880$.

As it is clear from the previous example, the tree is usually characterised by a very high number of possible sequences, and thus the computational effort required to evaluate the feasibility of each sequence is huge. However, if a cost can be associated to each edge of the tree, a graph search algorithm, e.g., A*, can be adopted to focus

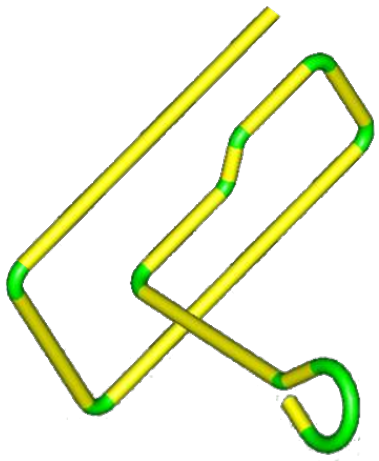
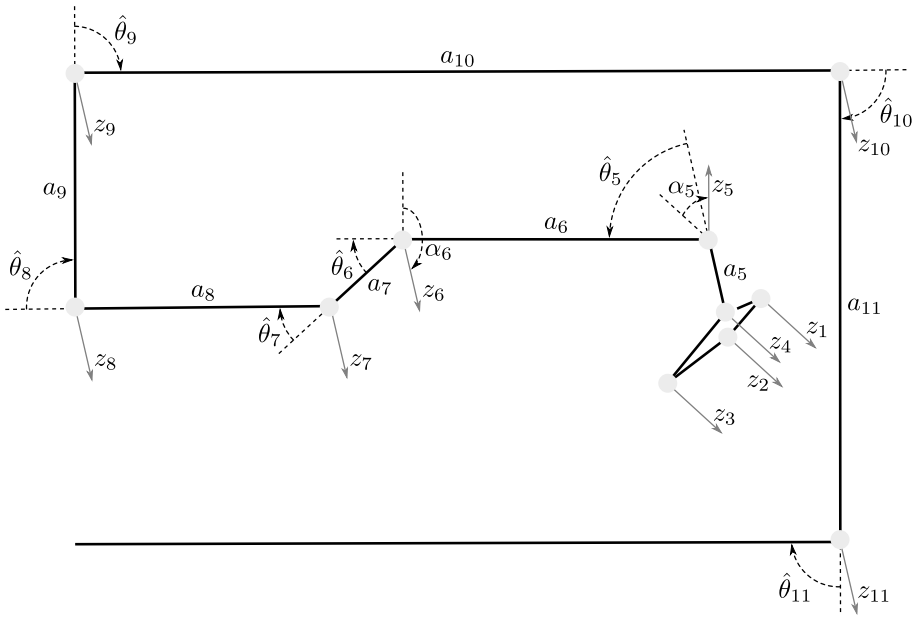
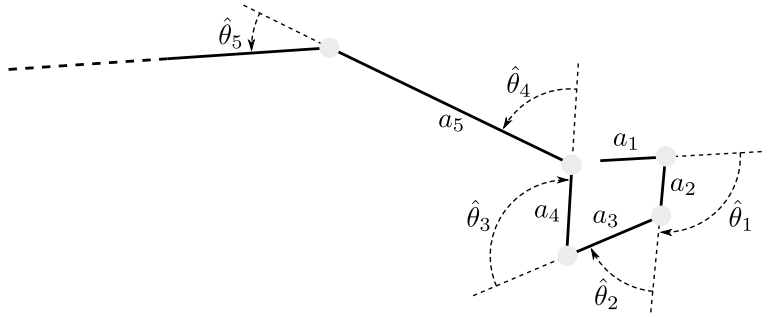


Figure 11. A 3D eyelet (green areas indicate bending operations to be performed).



(a) complete manipulator



(b) close-up of the first five joints

Figure 12. Sketch of the manipulator used to represent the 3D eyelet (drawing proportions were modified to increase readability)

the computational effort towards only a few good sequences in terms of the overall path cost. Collision checking is then performed on this small set of optimal sequences only, in order to further improve computational efficiency.

5. Bending sequence computation

As already mentioned, a graph search algorithm like A* can be adopted to determine a few good sequences in terms of the overall path cost.
A* is a search algorithm (Hart, Nilsson, and Raphael 1968) that determines the shortest path between a start and a goal node in a graph. It is one of the most popular technique used in graph traversals, due to its completeness, optimality, and efficiency. A* algorithm, unlike other traversal techniques, is a smart algorithm that determines the successive node based not only on the cost to reach the current node, but also anticipating the cost to reach the goal from thereon, by way of a problem-dependent heuristic function.

Though A* is a well-known algorithm, however, its application to the bending sequence computation problem requires the definition of an appropriate heuristic function and a way to estimate bending cost, as it will be clarified in the following.

5.1. *Bending cost estimation*

The cost of an edge in a bending tree can represent the energy or time required to execute a particular bend in a given wire configuration. In particular, between the two main operations required to perform a bend, i.e., motion of wire and turret to desired bending position (alignment) and bending execution (execution), only the first one is considered in the cost computation, as the execution effort is the same independently of the sequence.

The bending machine (see Figure 1) is composed of a base and a wire feeder that are fixed, and a turret sub-assembly that is mounted on a rotating support and can rotate about the wire.
The alignment motion for the execution of a generic bend b_i requires a translation t_i of wire outside the feeder, a wire rotation of an angle α_i, and a turret sub-assembly rotation of an angle β_i. As a consequence, assuming each bend has a minimum cost of one unit, the alignment change required to make bend b_{i+1} after bend b_i, i.e., the cost of the edge $e_{i,i+1}$, is given by

$$\text{Cost}\left(e_{b_i,b_{i+1}}\right) = 1 + w_1 \frac{t_{b_{i+1}} - t_{b_i}}{\Delta t_{max}} + w_2 \frac{\alpha_{b_{i+1}} - \alpha_{b_i}}{\Delta \alpha_{max}} + w_3 \frac{\beta_{b_{i+1}} - \beta_{b_i}}{\Delta \beta_{max}} \tag{1}$$

where w_1, w_2, and w_3 are suitable weights accounting for the fact that some movements, like wire rotation, may be faster or less energy consuming than others, Δt_{max}, $\Delta \alpha_{max}$, and $\Delta \beta_{max}$ are the maximum ranges of wire translation, turret sub-assembly rotation, and wire rotation, respectively.
Weights can be further scaled, based on relative time or energy consumption, and normalised such that $w_1 + w_2 + w_3 = 1$.

5.2. *Heuristic function*

Setting up an A* search requires to appropriately define the heuristic function $h(\cdot)$ and the actual cost $g(\cdot)$. The heuristic function estimates the cost of the shortest path from a node to the leaves, and must be admissible to guarantee optimality, i.e., it should never overestimate the cost to the target. Function $g(\cdot)$, instead, computes the actual cost of the shortest path from the root of the tree to a node, i.e., assuming an additive cost, the sum of the individual costs of each edge along the path.

From equation (1) and the cost additivity assumption, it follows that

$$g(b_{i+1}) = g(b_i) + e_{b_i,b_{i+1}}$$

where b_i is the parent node of b_{i+1}, and

$$h(b_i) = N - \text{depth}(b_i)$$

where N is the depth of the tree, i.e., the total number of bends in the workpiece, being thus constant for a given workpiece, and depth(b_i) computes the depth of node b_i in the tree, which is equal to the number of bends already straightened. In other words, $h(\cdot)$ simply computes the number of bends remaining to be straightened at any given instant.

The definitions of $h(\cdot)$ and $g(\cdot)$ guarantee that the heuristic function is admissible. In fact, it considers each bend taking only a unitary cost, while the actual cost is always greater than the unitary cost.

5.3. *A divide-et-impera approach*

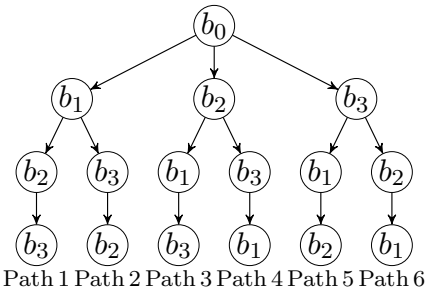
In the case of workpieces with high number of bends, i.e., more than 10 bends, A* can run into memory issues, as the allocated memory for the OPEN list grows factorially with the number of bends in the workpiece.

To overcome this issue, the problem can be broken down into smaller sub-problems, introducing an alternative implementation of A*, based on a path matrix approach, and splitting the bending tree into individual branches, each one corresponding to a possible bending sequence. Differently from the classical A* implementation, where a function allows to find successor nodes without storing the entire tree, with the path matrix approach each branch of the tree is stored as a row of a matrix (Figure 13), and the successor of any node can be found from the row corresponding to the current path.

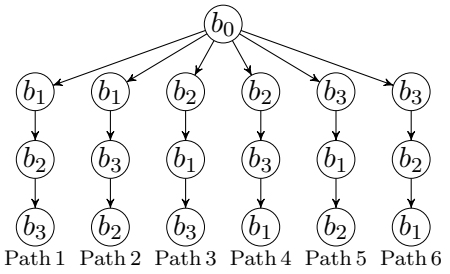
Note that for a workpiece with N bends, the path matrix approach needs to allocate memory for a matrix of $N!$ rows, and, in the worst case, the cardinality of the OPEN list set can be at most $N \cdot N!$. Consequently, the path matrix approach is definitely worse in terms of required memory, but its major advantage is the possibility of easily breaking a large problem into smaller ones.

The number of sub-problems is determined by the maximum memory that can be allocated for the OPEN list. Let r be the maximum number of rows allowed for the OPEN list, a workpiece characterised by N bends can be analysed braking down the problem into r/N sub-problems. Moreover, if a multi-processor system is available, A* search associated to each sub-problem can be run in parallel on different processors, thus significantly reducing the run time.

This divide-et-impera approach not only avoids memory allocation issues, but it also



(a) Tree Approach



Path 1	1	2	3
Path 2	1	3	2
Path 3	2	1	3
Path 4	2	3	1
Path 5	3	1	2
Path 6	3	2	1

Path matrix

(b) Path Matrix Approach

Figure 13. The path matrix approach.

allows rejecting more sequences during successive runs. In fact, considering that the f-value of a node, i.e., the sum of the values of the actual cost $g(\cdot)$ and the heuristic function $h(\cdot)$, increases going down in the tree, if m feasible sequences ordered in increasing cost are stored during the first sub-problem run, then any node (and its corresponding branch) with f-value greater than the cost of the m-th stored sequence can be closed.

6. Case studies

This section reports the results of two different test cases, constituted by two examples of industrial workpieces: the first one being more simple, as it is constituted by only 5 bends, the second one, characterised by 8 bends, more complex. The test cases were run on an Intel Core i5 processor with a frequency of 1.60 GHz, a total physical memory of 8 GB, an available physical memory of 2.4 GB, and the bending sequence computation algorithm has been implemented as a MATLAB™ script.

6.1. *A workpiece with 5 bends*

Figure 4 and Table 1 show the industrial workpiece considered as the first test case – a planar workpiece characterised by 5 bends, with all bend axes parallel to each other –, along with its D-H parameters.

For this workpiece, only $5! = 120$ possible bending sequences exist, and, in the worst case, the cardinality of the OPEN list set can be at most $5 \cdot 5! = 600$. No memory allocation issue is foreseen, and thus all paths can be evaluated in a single run.
Out of the 120 possible sequences, some of them may not be acceptable due to machine

Δt_{max}	135	w_1	0.4
$\Delta \alpha_{max}$	$\pi/2$	w_2	0.3
$\Delta \beta_{max}$	π	w_3	0.3

Table 6. 5-bend test case bending cost parameters (displacements are in mm, angles in degrees).

Sequence number	Bending sequence	Path cost
1	$\{5, 4, 3, 2, 1\}$	5.0529
2	$\{5, 4, 2, 3, 1\}$	5.0592
3	$\{4, 5, 3, 2, 1\}$	5.0606
4	$\{4, 5, 2, 3, 1\}$	5.0669
5	$\{4, 3, 2, 5, 1\}$	5.0719
6	$\{4, 2, 3, 5, 1\}$	5.0719
7	$\{2, 3, 4, 5, 1\}$	5.0783
⋮	⋮	⋮
22	$\{2, 4, 3, 5, 1\}$	5.2624
23	$\{1, 4, 5, 2, 3\}$	5.2636
24	$\{1, 4, 5, 3, 2\}$	5.2636
⋮	⋮	⋮

Table 7. 5-bend test case results.

limitations or collisions. By running an A* search with the bending cost characterised by the parameters reported in Table 6[3] on the straightening tree of this workpiece, 30 feasible sequences arranged in increasing cost order have been found in 1.1 seconds. Some of the feasible sequences determined by the algorithm are shown, in increasing cost order, in Table 7, where the first column stands for the position of the sequence in the cost ordered list.
Note that, thanks to the heuristic, only 497 nodes out of a maximum of 600 nodes have been evaluated in order to find the first 30 feasible sequences, thus avoiding the reachability check and cost computation for 103 nodes.

A collision check, using GJK algorithm, is then run on each of the sequences in Table 7, as described in Section 4.2. Note that, though including collision checking in the graph search algorithm guarantees that A* would visit the minimal number of nodes, this requires a tight heuristic. As the heuristic here considered is, instead, a coarse approximation of the cost of the shortest path from a node to the leaves, more nodes have to be visited. Therefore, number of collision checks required by finding the minimum cost admissible sequence is always less than or equal to the number of collision checks that would have been performed by A*. Furthermore, the computation time required by collision checking can be further reduced by distributing the operation over multiple processors.
For the 5-bend workpiece here considered, a GJK collision check on average requires 45

[3]Note that, cost function (1) is a translation into a mathematical form of the concept of "best bending sequence". Weight selection is thus application specific, and it represents an important task where the operator experience in the bending process plays a crucial role. In the case studies here reported, weight values have been selected, for the sake of example, to represent a situation in which the linear motion of the wire is faster, or less energy consuming, with respect to turret and wire rotations (see Tables 6 and 10).

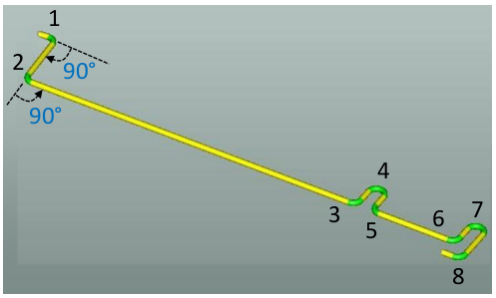


Figure 14. 8-bend test case.

a_i	α_i	d_i	$\hat{\theta}_i$
54	0	0	90
380	0	0	−87
16	0	0	−90
16	0	0	180
83	0	0	−90
16	0	0	−90
36	0	0	180
13	0	0	90

Table 8. D-H parameters of the 8-bend workpiece (displacements are in millimetres, angles in degrees).

	1	2	3	4	5	6	7	8	9	10
a_i	54	380	16	24	16	83	16	24	36	13
$\hat{\theta}_i$	90	−87	−90	90	90	−90	−90	90	90	90

Table 9. D-H parameters of the 8-bend workpiece after splitting bending angles higher than 160 degrees (displacements are in mm, angles in degrees).

seconds per sequence, as a consequence integrating A* with the collision check function would extend the time required to compute the first 30 sequences from approximately one second to 20 minutes.
As a result of the GJK algorithm, it was found that the 23rd sequence in Table 7, i.e., $\{1, 4, 5, 2, 3\}$, is the first that does not have collision issues, as almost all bends are made on the outer turret.

6.2. *A workpiece with 8 bends*

To assess the algorithm performance for workpieces with higher number of bends, a planar industrial workpiece with 8-bends (Figure 14) was considered as the second test case. The workpiece geometry is described by the D-H parameters, before and after the splitting of bending angles higher than 160 degrees, reported in Tables 8 and 9, respectively. Figure 15, instead, shows a sketch of the manipulator used to represent the workpiece together with the corresponding D-H parameters.
This workpiece was chosen as

- it allows to assess the algorithm in cases where there is a higher probability of wire self-collision and/or wire-machine collision;
- it has two 180 degree bends, each one split into two 90 deg bends that are regarded as grouped bends.

For this workpiece there are $8! = 40320$ possible bending sequences, and, in the worst case, the cardinality of the OPEN list set can be at most $8 \cdot 8! = 322560$. As MATLAB™ allows to allocate memory for so many rows, the code was run first without dividing the problem, and then by splitting it into smaller sub-problems. Table 11 shows the comparison of the average run time.
Clearly, as the number of sub-problems increases, the time required to set up each A*

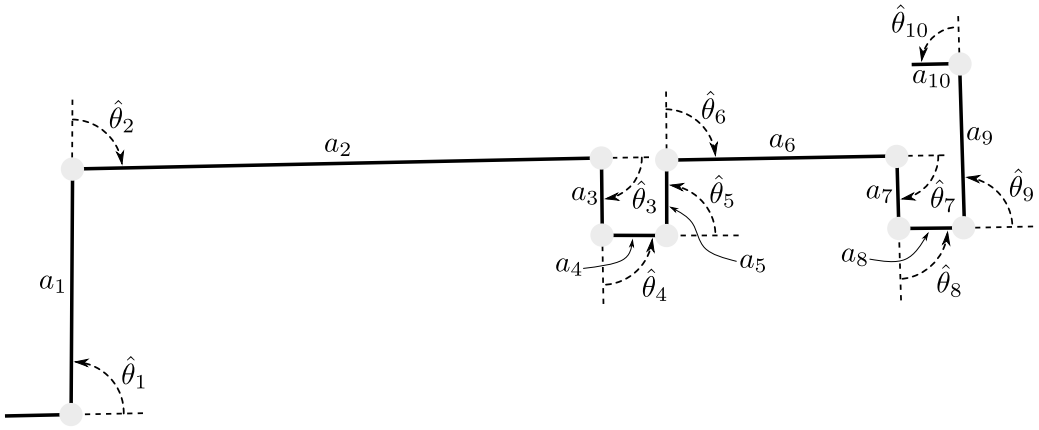


Figure 15. Sketch of the manipulator used to represent the 8-bend workpiece (drawing proportions were modified to increase readability).

Δt_{max}	614	w_1	0.4
$\Delta \alpha_{max}$	$\pi/2$	w_2	0.3
$\Delta \beta_{max}$	π	w_3	0.3

Table 10. 8-bend test case bending cost parameters (displacements are in mm, angles in degrees).

search increases as well, frustrating the time reduction gained by higher path rejection. However, the results do not depend on the way the problem is decomposed, but only in the number of sub-problems in which the problem is divided. Consequently, an optimal number of sub-problems can be easily derived.

By running an A* search with the bending cost characterised by the parameters reported in Table 10, some feasible sequences have been determined, and are reported, in increasing cost order, in Table 12. Note that, bends {5, 4} and {9, 8} are always consecutive bends in all bending sequences, as they are regarded as grouped bends. As in the first test case, GJK collision check algorithm has been run on the sequences reported in Table 12, showing that the execution of bends {5, 4} results in wire self-collision, as well as wire-machine collision, and that such collision occurs unless bend 6 is done before bends 5 and 4. At the end, the algorithm reveals that sequence 21 is the first that executes bend 6 before bends 5 and 4, and indeed it does not have any collision.

Number of sub-problems	Run time
1	6.7
4	5.4
33	20.8

Table 11. Run time comparison (time are expressed in seconds).

Sequence number	Bending sequence	Path cost
1	$\{10, 9, 8, 5, 4, 7, 6, 3, 2, 1\}$	10.0800
2	$\{9, 8, 10, 5, 4, 7, 6, 3, 2, 1\}$	10.0800
3	$\{10, 5, 4, 9, 8, 7, 6, 3, 2, 1\}$	10.0800
4	$\{5, 4, 10, 9, 8, 7, 6, 3, 2, 1\}$	10.0800
5	$\{9, 8, 5, 4, 10, 7, 6, 3, 2, 1\}$	10.0800
⋮	⋮	⋮
20	$\{5, 4, 9, 8, 10, 6, 7, 3, 2, 1\}$	10.1110
21	$\{7, 6, 5, 4, 9, 8, 10, 3, 2, 1\}$	10.1115
22	$\{7, 9, 8, 5, 4, 10, 6, 3, 2, 1\}$	10.1127
⋮	⋮	⋮

Table 12. 8-bend test case results.

7. Conclusions

As determining a bending sequence that allows to manufacture a complex workpiece, without causing self-collisions or collisions with the machine, requires a significant amount of time, even to an expert operator, this paper addresses the problem of the automatic computation of a set of feasible bending sequences, ordered according to a given cost function.

The algorithm here proposed leverages on a representation of the wire as a robotic manipulator, and of a bending sequence as a tree. It adopts A* as graph search algorithm, introducing a cost and a heuristic function that are particularly suitable for the wire bending problem, and an approach to parallelise the execution the graph search. The proposed algorithm is thus characterised by a limited computational complexity, so that it can be executed in an amount of time that is less or equal to the times needed by an expert operator to devise a solution, using a standard hardware.

Two test cases, considering a simple and a complex industrial workpiece, are presented to assess the effectiveness of the proposal. In particular, in the case of the workpiece with 5 bends, that is characterised by 120 possible bending sequences, the algorithm finds the best collision-free sequence after 15 minutes. Instead, for the workpiece with 8 bends, that is characterised by 40320 possible bending sequences, the algorithm finds the best collision-free sequence after 18 minutes. A comparison between the time required to determine the two solutions, considering that they correspond to the 23rd and 21st bending sequence found by the algorithm, reveals that the bottleneck is mainly in the collision checking procedure. In fact, the time required to perform 23 or 21 times collision checking dominates the time needed by the algorithm to find the best 23 or 21 sequences, even when the total number of possible sequences increases from 120 to 40320.

Nowadays, double head bending machines, that allow to machine more complex and longer workpieces, significantly increasing the flexibility and, thanks to a parallelisation of the bending operation, the throughput, are on the market. The presence of two independent heads, however, together with a moving gripper that can translate and rotate the wire independently from the bending heads, makes the problem of determining a good collision-free bending sequence rather hard, even for an expert operator.

Unfortunately, as in this case the number of possible bending sequences dramatically increases, the approach here described is no more able to find a suitable number of good sequences in a reasonable amount of time. Devising an algorithm for the automatic computation of bending sequences for double head wire bending machines is thus the most important topic of a future research in this field.

References

Aomura, S., and A. Koguchi. 2002. "Optimized bending sequences of sheet metal bending by robot." *Robotics and Computer-Integrated Manufacturing* 18 (1): 29–39.

Baraldo, Andrea, Luca Bascetta, Fabrizio Caprotti, Gianni Ferretti, and Angelo Ponti. 2021. "Procedimento di piegatura e macchina di piegatura per eseguire un procedimento di piegatura." Italian Patent.

Bhattacharya, P. 1994. "The representation of permutations by trees." *Computers & Mathematics with Applications* 28 (9): 67–71.

De Vin, L.J., J. De Vries, A.H. Streppel, E.J.W. Klaassen, and H.J.J. Kals. 1994. "The generation of bending sequences in a CAPP system for sheet-metal components." *Journal of Materials Processing Technology* 41 (3): 331–339.

Duflou, J., J. Kruth, and D. Van Oudheusden. 1999. "Algorithms for the design verification and automatic process planning for bent sheet metal parts." *CIRP Annals - Manufacturing Technology* 48 (1): 405–408.

Duflou, J.R., D. Van Oudheusden, J. Kruth, and D. Cattrysse. 1999. "Methods for the sequencing of sheet metal bending operations." *International Journal of Production Research* 37 (14): 3185–3202.

Faraz, Z., S. W. Ul Haq, L. Ali, K. Mahmood, W. A. Tarar, A. A. Baqai, M. Khan, and S. H. Imran. 2017. "Sheet-metal bend sequence planning subjected to process and material variations." *International Journal of Advanced Manufacturing Technology* 88 (1-4): 815–826.

Fisher, Ronald Aylmer, and Frank Yates. 1948. *Statistical tables for biological, agricultural and medical research*. London: Oliver and Boyd.

Garcia, G., and J. F. Le Corre. 1989. "A New Collision Detection Algorithm Using Octree Models." In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, 93–98.

Gilbert, E. G., D. W. Johnson, and S. S. Keerthi. 1988. "A fast procedure for computing the distance between complex objects in three-dimensional space." *IEEE Journal on Robotics and Automation* 4 (2): 193–203.

Gupta, S. K., D. A. Bourne, K. H. Kim, and S. S. Krishnan. 1998. "Automated Process Planning for Sheet Metal Bending Operations." *Journal of Manufacturing Systems* 17 (5): 338–360.

Hart, P.E., N.J. Nilsson, and B. Raphael. 1968. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." *IEEE Transactions on Systems Science and Cybernetics* 4 (2): 100–107.

Heap, B. R. 1963. "Permutations by Interchanges." *The Computer Journal* 6 (3): 293–298.

Hoffmann, M., U. Geiler, and M. Geiger. 1992. "Computer-aided generation of bending sequences for die-bending machines." *Journal of Materials Processing Technology* 30 (1): 1–12.

Inui, M., and H. Terakado. 1998. "Fast evaluation of geometric constraints for bending sequence planning." In *IEEE International Conference on Robotics and Automation*, Vol. 3, 2446–2451.

Inui, M., and H. Terakado. 1999. "Fast bending sequence planning for progressive press-working." In *IEEE International Symposium on Assembly and Task Planning*, 344–349.

Kannan, T. R., and M. S. Shunmugam. 2008. "Planner for sheet metal components to obtain optimal bend sequence using a genetic algorithm." *International Journal of Computer Integrated Manufacturing* 21 (7): 790–802.

Koguchi, A., and S. Aomura. 2002. "Automated process planning for sheet metal bending by handling robot process planning method by taking critical dimension into account." *Seimitsu Kogaku Kaishi/Journal of the Japan Society for Precision Engineering* 68 (4): 602–607.

Koguchi, A., S. Aomura, and M. Igoshi. 2000. "The Automated Process Planning for Sheet Metal Bending by Handling Robot." *Nihon Kikai Gakkai Ronbunshu, C Hen/Transactions of the Japan Society of Mechanical Engineers, Part C* 66 (646): 2060–2067.

Lin, A. C., and C. Chen. 2014. "Sequence planning and tool selection for bending processes of 2.5D sheet metals." *Advances in Mechanical Engineering* 2014.

Lin, A. C., and D. K. Sheu. 2012. "Sequence planning for bending operations in progressive dies." *International Journal of Production Research* 50 (24): 7493–7521.

Markus, A., J. Váncza, and A. Kovács. 2002. "Constraint-based process planning in sheet metal bending." *CIRP Annals - Manufacturing Technology* 51 (1): 425–428.

Ong, S.K., L.J. De Vin, A.Y.C. Nee, and H.J.J. Kals. 1997. "Fuzzy set theory applied to bend sequencing for sheet metal bending." *Journal of Materials Processing Technology* 69 (1-3): 29–36.

Raj Prasanth, D., and M. S. Shunmugam. 2020. "Geometry-based Bend Feasibility Matrix for bend sequence planning of sheet metal parts." *International Journal of Computer Integrated Manufacturing* 33 (5): 515–530.

Rico, J. C., J. M. González, S. Mateos, E. Cuesta, and G. Valiño. 2003. "Automatic determination of bending sequences for sheet metal parts with parallel bends." *International Journal of Production Research* 41 (14): 3273–3299.

Sedgewick, Robert. 1977. "Permutation Generation Methods." *ACM Computing Surveys* 9 (2): 137—-164.

Shpitalni, M., and D. Saddan. 1994. "Automatic Determination of Bending Sequence in Sheet Metal Products." *CIRP Annals* 43 (1): 23–26.

Siciliano, Bruno, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. 2009. *Robotics – Modelling, Planning and Control*. Springer.

Thanapandi, C. M., A. Walairacht, and S. Ohara. 2001a. "Genetic algorithm for bending process in sheet metal industry." In *Canadian Conference on Electrical and Computer Engineering*, Vol. 2, 957–962.

Thanapandi, C. M., A. Walairacht, and S. Ohara. 2001b. "Multi-component genetic algorithm for generating best bending sequence and tool selection in sheet metal parts." In *IEEE International Conference on Robotics and Automation*, Vol. 1, 830–835.

Zhao, Z. Y., L. C. Zhang, and Y. S. Shi. 2014. "A bending sequence planning algorithm based on multiple-constraint model." In *International Conference on Key Engineering Materials and Computer Science*, Vol. 1042 of *Advanced Materials Research*, 26–31.