RWTH Aachen

Computational Engineering Science

Automatic Bend Sequence Planning for Press Brakes

CES Seminar

Summer Semester		2015
Presented by	:	Guo, Xuanyu
Matriculation No.	:	299344
Supervisor	:	Arndt Steffen
		Beckhoff Automation GmbH & Co. KG

Table of Contents

1 Introduction

Sheet metal forming is one of the oldest manufacturing processes known to mankind [Knoblach99]. Countless everyday objects, from switch boxes to car bodies, are produced in this method with tools such like press brake. The goal of this essay is to develop an automatic sequence planning algorithm for press brake with help of computational engineering techniques.

Bending sequence is the processing order of all bends on the product. A feasible sequence is the one who makes sure there is no collision during the whole machining. However, the planning of bending sequence is more complicated than discussion about feasibility. A better sequence will include less tools and tool sets to produce more complex work piece. It can also reduce the options of press brake operator, such as moving or flipping the sheet metal, and take collinear cases into consideration in order to reduce the time cost. In the past, planning of bending sequence relied on manual operation. However, the rapid improvement of hardware and software in last decades has enabled this type of work to be performed by computer-aided methods [Ong97].

By complex product, for instance with more than 30 bends, the major difficulty lies in the numerous combinatorial possibilities. The amount of calculation for a brute force evaluation of all possibilities would be the factorial of bends number. Exhaustive searches could lead to unacceptable response time. In job shop environments, process planning could easily become a more resource-consuming activity than the actual shop floor production. The most important objective for all the companies involved in this project was to minimize the makespan. [Cattrysse 06]

To reduce planning time, intelligent algorithms need to be applied to enhance this situation. The algorithm discussed in this essay, “Branch & Bound” with “depth first search” Method, and Pattern Method, are two of advanced algorithms used in the world by now. With help of certain geometric features called “pattern”, the Pattern Method looks for an intelligent pre-identification of potentially interesting sequences. On the other hand, Branch & Bound looks for all combinatorial possibilities but in a time-efficient way.

The sequence planning problem can include various other interesting research topics. First subject comes up is the coupling with tool selection. By change of either or both tool selecting and sequence planning, a more efficient process could be obtained. The priority of the two options is however controversial. If specific subjects are brought up, such as the co-linear bend issue (several bends manufactured with a single stroke at the same time), the setting and combing of stations (a set of punches and dies), and process planning (take other factors besides bending into consideration), there are specific modules. And all of these well-done but complicated research are based on the sequence planning method being applied.

In this essay important concepts and tools of sequence planning such as bending direction and collision detection will be introduced first. The description for “Branch & Bound” and “Pattern Method” follows with a simple comparison. At last pattern method algorithm programed in C++ will be tested and improved by visual warning and manual change.

2 Sequence Planning

A geometric model will be built for sequence planning. How to describe bending direction would be critical important for this model's illustration. Both Pattern method and Branch & Bound method are basically routines used to rule out non-optimum solutions by means of collision detection. As the tool of deciding if a planning algorithm is good or not, collision detection need to be introduced first. Once these two important concepts being explained, the sequence planning method can be discussed.

2.1 Bending Direction

Several articles about process planning have all invented their own representations of bending direction. To avoid ambiguous, positive or negative bending angles are used in this thesis to distinguish different bending directions.

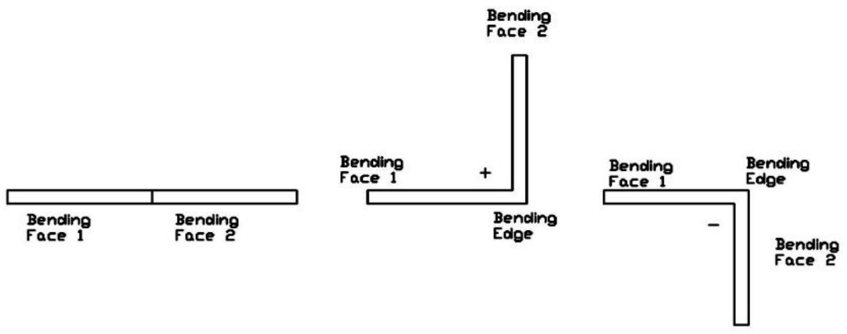


Figure 2.1 Bending Direction

Assuming that the sheet metal lies horizontal when it is not processed and first bending face stays horizontal after processing. If the second bending face turns clockwise after machining, this bending angle would be declared as positive with symbol "+". On the contrary, if the second bending face turns counterclockwise, the bending angle will be considered negative described by "-".

2.2 Collision Detection

Typically, the upper tool "punch" is mounted on the upper side (ram) of a press brake, while the lower tool, a V-shaped groove "die", is connected with the lower part (bed) of bending machine. The punch presses the sheet metal down into the V-shaped die causing the metal to bend.

Collision can occurs in the adjust state between die and sheet metal as shown in Figure 2.2 left or in a bend state between punch and sheet metal as shown on the right. Collision between tools, e.g. Back Gauge and punch, can also happen during the positioning in adjust state. All these kinds of collisions can be generalized as intersection of two line segments in geometry.

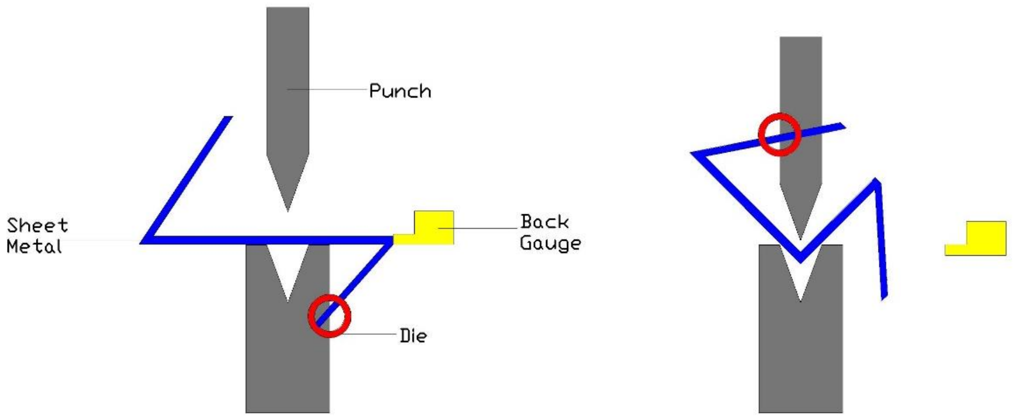
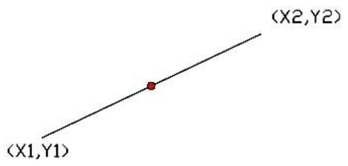


Figure 2.2 Collision between sheet metal and tools at different states

Generally spoken, given two points (x_1, y_1) and (x_2, y_2) a line can be defined. If a point A (x, y) is on the same line, it has to satisfy:

$$\begin{cases} x = (x_2 - x_1)u + x_1 \\ y = (y_2 - y_1)u + y_1 \end{cases}$$

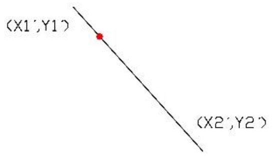


Further more, if point A is within the line segment with two end points (x_1, y_1) and (x_2, y_2) on this line, another condition need to be satisfied: $0 \le u \le 1$.

Likewise we can define another point B (x´, y´) located on another line segment with end points (x'_1, y'_1) and (x'_2, y'_2).

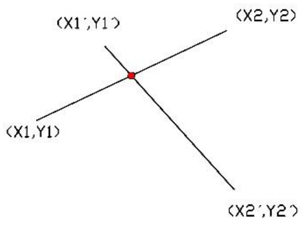
$$\begin{cases} x' = (x'_2 - x'_1)u' + x'_1 \\ y' = (y'_2 - y'_1)u' + y'_1 \end{cases}$$

when $0 \le u' \le 1$.



The problem of two line segments intersecting each other can be transformed into, whether a single point can locate on two line segments at the same time. In other words, point A is at certain moment also point B.

$$\begin{cases} x = x' \\ y = y' \end{cases}$$



1) If these two line segments are not parallel then the value u and u´ could be described as:

$$u = \frac{x'_2 y'_1 - y'_2 x'_1 + x_1(y'_2 - y'_1) - y_1(x'_2 - x'_1)}{(x'_2 - x'_1)(y_2 - y_1) - (y'_2 - y'_1)(x_2 - x_1)}$$

$$u' = \frac{x_1 y_2 - y_1 x_2 - x'_1(y_2 - y_1) + y'_1(x_2 - x_1)}{(x'_2 - x'_1)(y_2 - y_1) - (y'_2 - y'_1)(x_2 - x_1)}$$

$$when\ (x'_2 - x'_1)(y_2 - y_1) - (y'_2 - y'_1)(x_2 - x_1) \neq 0$$

For simplicity, the denominator could be represented by:

$$D= (x'_2-x'_1)(y_2-y_1)-(y'_2-y'_1)\,(x_2-x_1)$$

The two segments will not intersect (in our case of bending process, which normally means no collisions between tool contour and sheet metal), if and only if:

$$u < 0, \text{ or } u > 1, \text{ or } u' < 0, \text{ or } u' > 1.$$

2) In specific cases two line segments can be parallel, which leads to D=0.

$$(x'_2-x'_1)(y_2-y_1)-(y'_2-y'_1)\,(x_2-x_1)=0$$

The collision detecting problem becomes collinear problem. The aim is to check if any part of the first segments is located on the second one, i.e. to check whether any end point of the first line segment lies on the second line segment.

The parameter u_1 decides here, if first end point (x_1, y_1) of first line segment is on the second line segment. Parameter u_2 shows if the second end point (x_2, y_2) of the first line segment is on the second line segment.

When $x'_2 \neq x'_1$ and $y'_2 \neq y'_1$ we need satisfy both of following condition to obtain a collision free geometry.

$$\begin{cases} \dfrac{x_1-x'_1}{x'_2-x'_1} = \dfrac{y_1-y'_1}{y'_2-y'_1} \\ u_1 = \dfrac{x_1-x'_1}{x'_2-x'_1} < 0, or\ u_1 > 1 \end{cases} \quad ①$$

$$\begin{cases} \dfrac{x_2-x'_1}{x'_2-x'_1} = \dfrac{y_2-y'_1}{y'_2-y'_1} \\ u_2 = \dfrac{x_2-x'_1}{x'_2-x'_1} < 0, or\ u_2 > 1 \end{cases} \quad ②$$

On the other hand, once $y'_2=y'_1$ (second line segment horizontal), or $x'_2=x'_1$ (second line segment vertical), u_1 can only be expressed by $u_1 = \frac{x_1-x'_1}{x'_2-x'_1}$ or $u_1 = \frac{y_1-y'_1}{y'_2-y'_1}$, but the algorithm stays the same. As long as $u_1 < 0$ or $u_1 > 1$, no collision will happen.

2.3 Branch and Bound Method

2.3.1 Traveling Salesman Problem

Apparently, the sequence planning problem for press brake has a very specific structure. Each bend can be bent exactly only once and the goal is to minimize the total cost. A well-known model, the Traveling Salesperson Problem, can capture this structure. [Cattrysse 06]

The Traveling Salesperson Problem (TSP) is one of many combinatorial optimization problems in the set NP-complete [Verlinden 07]. A salesperson needs to visit n cities and has to start and finish at the same city. He must visit all the cities but each city only once. The cost of visiting is usually defined by the distances between cities, which is nonnegative, and is specified in a cost matrix. The wish of this tour is to find the sequence of visiting that make the overall cost minimum.

The formula (**1**) is the objective function, the goal of this optimization problem is to minimize the cost. The c_{ij} is the cost from city i to city j, and the value of x_{ij} is 1 or 0. 1 means the sales man choose the tour from city i to j and 0 means he doesn't. The condition (**2**) is that, every city is visited exactly once. (**3**) is Balancing constraints, a city entered has to be left as well. Introducing (4) shows the procedure starts at starting point. Integer constraints is described by (**5**).

$$\text{Cost} = \min \sum_{i \in I} \sum_{j \in I} c_{ij} x_{ij} \qquad (\mathbf{1})$$

$$\sum_{j \in \mathrm{I}} x_{ji} = 1 \qquad i \in I \qquad (\mathbf{2})$$

$$\sum_{j \in \mathrm{I}} x_{ij} = \sum_{j \in \mathrm{I}} x_{ji} \qquad i \in I \qquad (\mathbf{3})$$

$$\sum_{j \in \mathrm{I}_0} x_{0i} = 1 \qquad (\mathbf{4})$$

$$x_{ij} \in \{0,1\} \qquad i,j \in I \qquad (\mathbf{5})$$

There are several procedures to solve the TSP. In this essay the Branch & Bound procedure is discussed. Before getting into details of Branch & Bound, the subroutine which is used to search should be introduced first.

2.3.2 Depth First Search

Depth-first search (DFS) is one of the simplest algorithm to search a graph. It has been widely used since late 1950s. The strategy of depth-first search is, as its name implies, to search "deeper" in the graph whenever possible. Given a graph G = (V, E) with V vertex and E edges and the distinguished source vertex s, Depth-First Search explores edges out of the most recently discovered vertex v who still has unexplored edges leaving it. Once all of v's edges have been explored, the search "backtracks" to explore edges leaving the vertex from which v was discovered. This process continues until we have discovered all the vertices that are reachable from the original source vertex. The running time of DFS is therefore Θ (V+E) [CLRS], which is quite impressive.

2.3.3 Branch and Bound

For bending sequence planning, the cost matrices for TSP could be affected by couple of factors. Firstly the feasibility of this sequence must be verified. If there is a collision in a sequence, or in other words this sequence is not feasible, the cost should be relatively high. Other factors affect the efficiency of the process. For example, if replacement of tools is needed for certain sequence, the makespan will increase accordingly. Therefore the cost will be higher. Other factors need to be taken into consideration include the amount of movements between different tool stations and the amount of flip (or swap) for work piece. According to [Kannan08] the objective function should be defined as:

$$F = C_1 \sum \omega_i R_i + \mathrm{C_2 P}$$

It comprises two terms, first term representing cost for feasible sequences and second one for infeasible sequences. Although for each sequence only one term is valid it's handy to put both cases in one formula. R_i can be the amount of flip operation, movement from stations and so on, depending on how many and what kind of factors the algorithm want to focus on. And ω_i is the weight of each R, so the importance of each factors can be assigned and controlled. The weight of infeasible sequence should be rather larger to differ from sum of all terms of feasible cost. In following example it is settled as 100. To find out whether a collision happens, we need to call the collision detection function discussed in section 2.2 for all segments of tools and work pieces.

In context of bending sequence planning, visiting city 1 from city 2 means processing first bend first and second bend next, which should be opposite from visiting city 2 from city 1(processing first bend after the second bend). So the matrix is always asymmetric. A possible cost matrices for 5 bends could looks like this:

$$\begin{matrix} \infty & 3 & 100 & 4 & 2 \\ 5 & \infty & 2 & 3 & 100 \\ 5 & 100 & \infty & 1 & 2 \\ 5 & 1 & 7 & \infty & 3 \\ 5 & 2 & 100 & 3 & \infty \end{matrix}$$

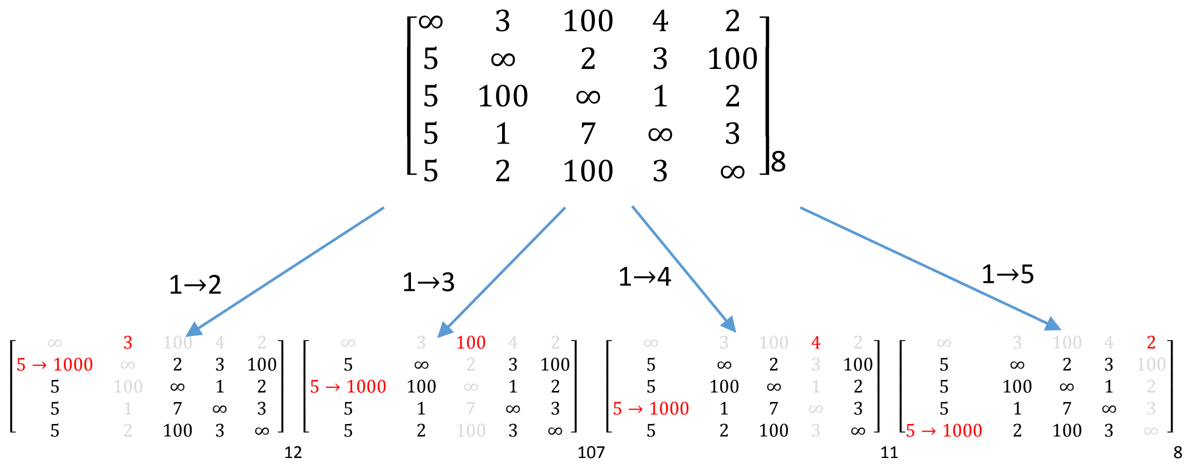
As a tour from a city can't visit itself, the elements on the diagonal should be infinite big. In practice, we can set this element with a relative big value, like 1000. So we start from city 1 for example. Next we are going to compute the lower bound of this matrix. For description we can name every element in this matrix C_{ij} for cost. The smallest value starts from city one is C_{15}=2 means from city 1 to city 5. In same meaning, the smallest value for other rows are: C_{23}=2, C_{34}=1, C_{42}=1, C_{52}=2. No allover tour can cost less than 2+2+1+1+2=8. We can use subscript 8 to show this lower bound.

$$\begin{bmatrix} \infty & 3 & 100 & 4 & 2 \\ 5 & \infty & 2 & 3 & 100 \\ 5 & 100 & \infty & 1 & 2 \\ 5 & 1 & 7 & \infty & 3 \\ 5 & 2 & 100 & 3 & \infty \end{bmatrix}_8$$

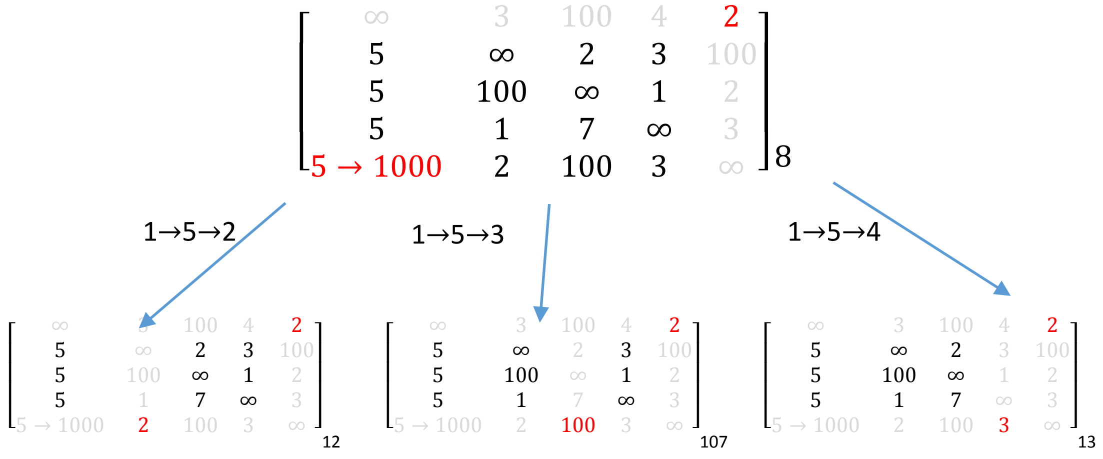
There are four possibilities in second level, to city 2, 3, 4, 5. From city 1 to 2, we take the value 3 and delete the first row. Because we need to make sure, no other solutions can visit city 2, we delete second column too. To avoid just return city 1 from city 2, we make C_{21} very big, like 1000. Now the lower bound is 3+2+1+3+3=12:

$$\begin{bmatrix} \infty & 3 & 100 & 4 & 2 \\ 5 \to 1000 & \infty & 2 & 3 & 100 \\ 5 & 100 & \infty & 1 & 2 \\ 5 & 1 & 7 & \infty & 3 \\ 5 & 2 & 100 & 3 & \infty \end{bmatrix}_{12}$$

In this way, the search begins and the tour is written on the branch. Lower bound for each branch is also calculated. 1→3=100+3+1+1+2=107, 1→4=4+2+2+1+2=11, 1→5=2+2+1+1+2=8. The first level is drawn.



By Depth-First Search, the node with the smallest lower bound is used to generate nodes at level 3, other nodes are not considered first, which means we take 1→5 here. This pattern continues until at the lowest level, nodes representing full tours are obtained. The third level can be 1→5→2, 1→5→3, 1→5→4. 1→5→2 means we take C_{52} for sure, and delete fifth row and second column. The lower bound is 2+2+2+1+5=12. The lower bound of other solution in third level are also calculated.



We take 1→5→2 further, and the two leaves are generated: 1→5→2→3→4(→1) or 1→5→2→4→3(→1). The lower bound are 12 and 19.

So a solution 1→5→2→3→4(→1) is found, and the branch is bounded from below by 12. The branch 1→5→2→4→3(→1) is pruned, because it is a more expensive tour. Before searching other candidate solutions of a branch, those branches are checked against lower bound on the current optimal solution 12, and those branches will be discarded if they can't produce a better solution than 12. This pruning has the effect of sparing the computational process of generating nodes below the pruned node. This could result in a significant saving if the pruned node were relatively near the top of the

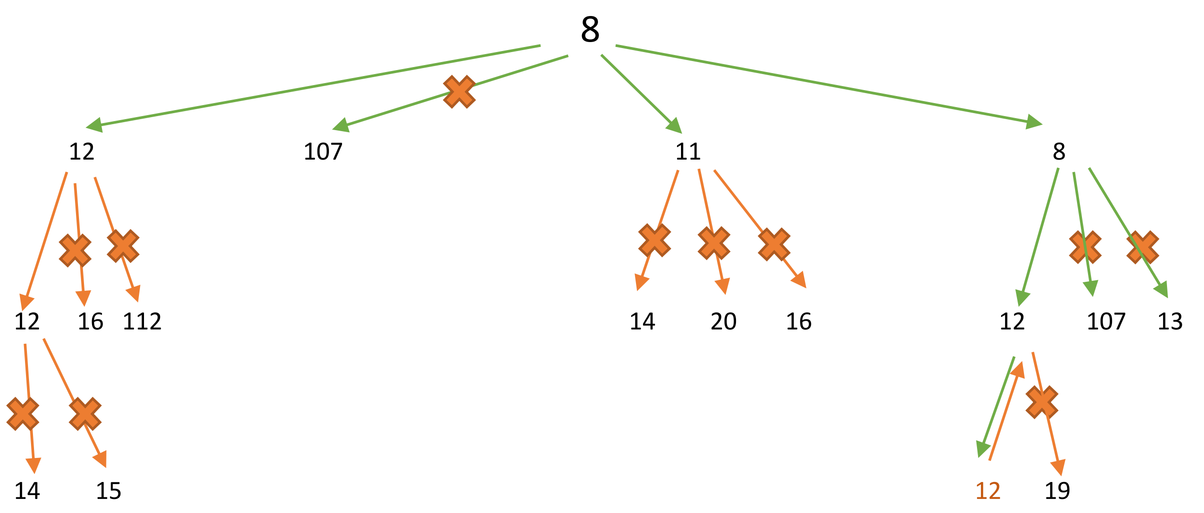


Figure 2.3 Searching process with Branch and Bound Method

tree.

If we have an event of a tie (two nodes with equal lower bound), we can calculate the sum of the city numbers in the partial tour (e.g. the sum of partial tour 1→5→2 is 1+5+2=8). Then the node with the smaller sum is searched first. It is obvious that these rules doesn't affect pruning at all, it just make sure two nodes don't have an equal priority, and define an absolute order of searching. [Wiener 03]

As shown in above graph, we get the first solution (green line) of 12, and go backtracks to prune other nodes (orange line). While the searching goes, we find no better solution (orange line). If we have a better solution, we will prune the original best solution 12, carry on the searching and compare further lower bound with new best solution. After all branch (instead nodes) are checked, 12 is chosen as least cost and sequence 1→5→2→3→4(→1) is chosen as the sequence. As we known, bend 1 is not necessary to bend twice, the first column is settled all as 5 (coming back to city 1 from any city is same), to make sure no problem happens. We can also solve the problem of 5 bends with 6×6 cost matrix by adding an invalid starting point.

Although Branch and Bound method still generate many nodes, it cost however very little time at each node. There are no constrains to compute for each node and no constraints to store in each node [Wiener 03]. Therefore the calculating time is actually not so much. Another thing to notice is: nodes at a deeper level have higher priority than nodes at a lower level, because the depth first algorithm.

In spite of the fact that this Branch and Bound method seems promising in sequence planning, it also contains a major difficulty in industrial application. First of all, a panel embedded with sequence planning algorithm is developed to assist operator finishing process design, therefore the procedure has to run online. In this context, heuristic procedures are preferred [Cattrysse 06]. While the makespan is reduced, a relative optimum instead of a perfect solution is obtained. Comparing with the Pattern Method, the complicated Branch and Bound method doesn’t seem so superior anymore. Secondly, for our concrete subject of bending sequence planning, the matrices are always dynamically updated, because every single decision of planning will influence the further sequence. The consequential calculation of matrix brings in problematic time consuming. Based on the above stated reasons, the pattern method with visualization alarm mechanism is our preference.

2.4 Pattern Method

2.4.1 6 Patterns

According to [Lin14], any desired sheet metal profile could be geometrically divided into series of fractions, which can all be generalized as six patterns. These 6 patterns to be introduced are Pattern Ω-Pattern, P-Pattern, C-Pattern, Z-Pattern, U-Pattern, and L-Pattern.

L-Pattern, named after its shape, is the simplest pattern. It is the only possible figure after bending a flat sheet metal.

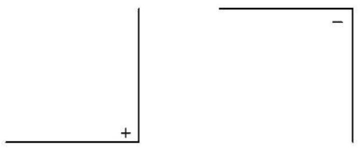


Figure 2.4 L-Pattern

The Pattern U-Pattern have two bending angles α1 and α2 with same sign, “++” or “--”. Comparing with U-Pattern, Z-Pattern is also bent twice, but with two bending direction, denoted as “+-” or “-+”.

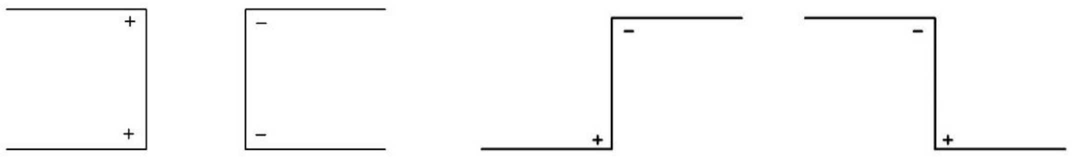


Figure 2.5 U-Pattern and Z-Pattern

Three bends with their own same direction are processed in Pattern C-Pattern, hence “+++” “---” can be used to describe this Pattern.

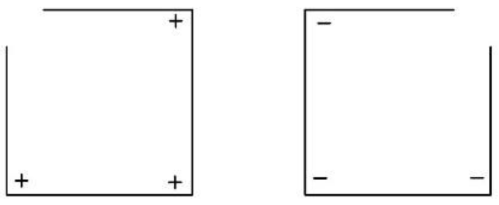


Figure 2.6 C-Pattern

Even though P-Pattern (also named “hat” pattern in [Aomura02]) obtains the same amount of bends as C-Pattern, the situation is more complicated. Not only a P-Pattern has all together four possibilities of set up (“++-”, “--+”. “+--”and “-++”), the analysis of the P-Pattern is also more tricky.

Because P-Pattern has bends with different directions (Z-Pattern and Ω-Pattern have the same issue), the collisions we need to look out don’t only happen between punch and work piece, they can also happen during the adjustment before bending between workpiece and die. Furthermore, the collision detection of a P-Pattern doesn’t only mean one detection of tool and sheet metal, the searching includes all intermediate state of P-Pattern machining process of the work piece and tool (during bend with punch, during adjusting with die). That causes certain amount of drawing and calculation for computer.

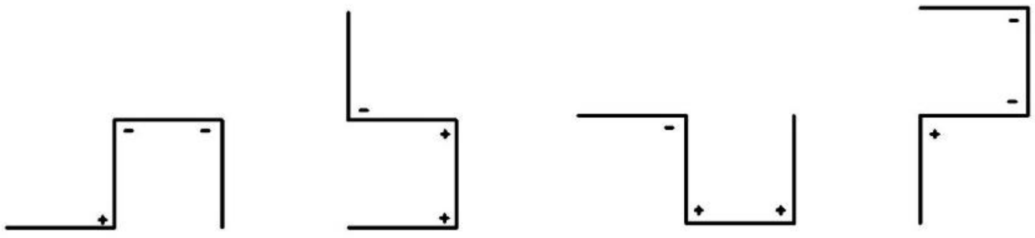


Figure 2.7 P-Pattern

The last pattern is Ω-Pattern (or “channel” pattern in [Aomura02]). Not surprisingly, it owns a shape similar to the symbol “Ω”. It contains four bending lines and five bending surfaces. It could be interpreted either with “+--+” or “-++-”.

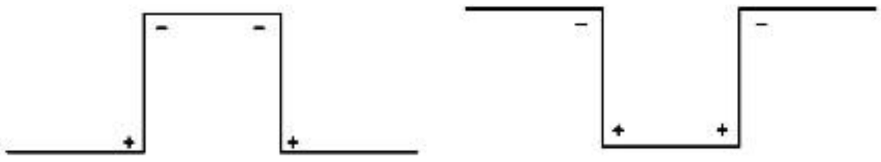


Figure 2.8 Ω-Pattern

2.4.2 Recognition of Product and Generation of Sequence

By practical sequence planning with pattern method, bending angle with symbol “+” or “-” will be firstly assigned to bending position, according to their direction. The created code would be compared with the codes of different patterns. The pattern will either match this code, then a certain part of the product will be recognized as this pattern, or will not match the code, so a next pattern or next several codes will keep the detecting going until all codes are recognized. After the whole detecting is finished, the desired profile of product would be totally recognized as combination of patterns.

However, the major problem is the order of recognition. What if a couple of codes can be recognized both as a part of Ω-Pattern or as a part of U-Pattern? This is relevant with the processing order of each pattern, after we get the recognition. In general, pattern with more bending steps requires likely more space during the manufacturing process. There will be also a higher risk of interference if they are arranged in the later stages of the bending operation. Therefore, based on the number of bending steps in each pattern, an order of bend processing is determined. Ω-Pattern contains four bending steps; it will be always bent at first. P-Pattern and C-Pattern both have three bends. However, because P-Pattern contains bends with different bending direction, which means the sheet metal will be inevitably flipped, the handling time of P-Pattern will be much longer than C-Pattern in practical manufacturing. Therefore, P-Pattern will be bent in prior to C-Pattern, right after Ω-Pattern.

For the same reason, Z-Pattern will be bent after C-Pattern. U-Pattern and L-Pattern will be bent at last.

Obviously, the first processed pattern should also have the higher priority by recognition, so the member of this pattern won't be detected and considered as a part of lower-priority pattern. Hence, the order of recognition can be described as: Ω-Pattern→P-Pattern→C-Pattern→Z-Pattern→U-Pattern→L-Pattern.

An example for the pattern recognition is explained in accordance with above mentioned method. Take the work piece in Figure 2.9 as an example. First, "+" and "-" are assigned to every bending position, the code obtained is shown on the right. The list of "+" and "-" is demonstrated as "+ + - - - + + - - - + +".

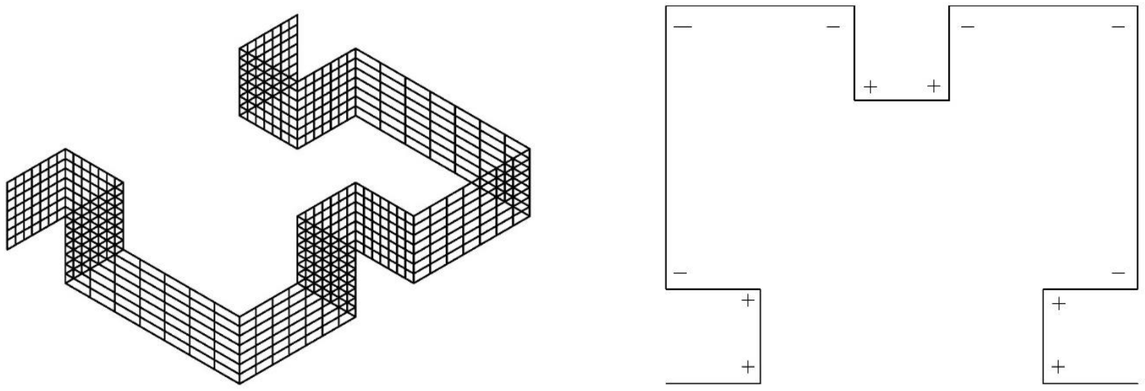


Figure 2.9 Example of product and its bending code

Subsequently, comparison between profile and patterns will be executed in accordance with predefined order Ω → P →C → Z → U → L. When all comparison with six patterns are finished, there won't be any digit of bending code not recognized, because combination of these 6 patterns exhausted all possibilities (actually even only U-Pattern and Z-Pattern can already exhausted all possibilities). In this manner, a combination of patterns is obtained. The bending sequence will be easily decided accordingly.

Step 1. Bending codes are compared with pattern Ω-Pattern first.

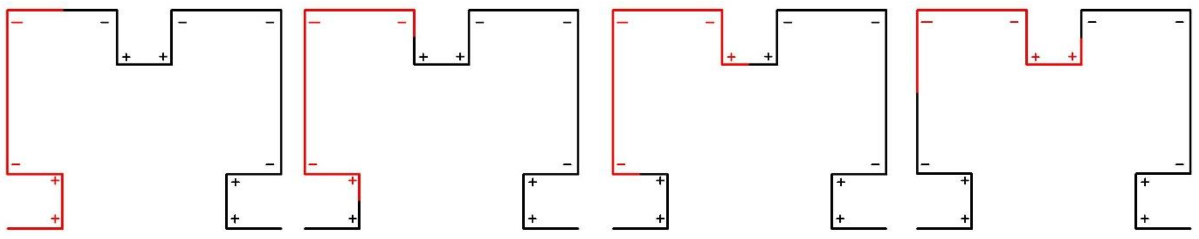


Figure 2.10 Recognition of Omega, from bend 1 to bend 4

Take the first four digits "+ + - -" of the bending code list and compare them with the codes of Ω-Pattern "- + + -" and "+ - - +". Apparently, we do not get a match here. Thus, the second to fifth digit "+ - - -" are sequentially taken and compared with Ω-Pattern. The result shows this is not a Ω-Pattern either. Next, the third to sixth digits "- - - +" are taken and compared, still not recognized as Ω-Pattern. The comparisons carry on, the fourth to seventh digits "- - + +" are also not a Ω-Pattern.

Then the fifth to eighth digits "- + + -"are picked, compared and recognized as a match. Therefore the fifth to eighth digits are settled as the first found Ω-Pattern. The next comparison starts right after the found pattern, from ninth digit. These means, the detecting jumps over the sixth, seventh, and the eighth digits (which is a noticeable behavior in implementation).

But the ninth to twelfth digits "- - + +" are not the match. For starting from next digit, the tenth, there won't be enough digits to compose a Ω-Pattern, the comparison with Ω-Pattern is thence finished.

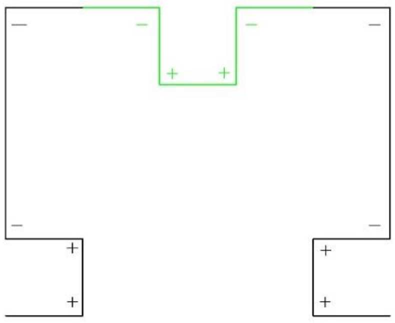


Figure 2.11 Recognition of Omega, bend 5

Step 2. Comparison with other patterns continue. According to the principle, the next pattern to be compared is P-Pattern.

The first three digits "+ + -"are picked to be compared. Because they are recognized as a match to P-Pattern, these three digits are settled. The next comparison starts from the next unsettled digit, the fourth. And from fifth to eighth digit are settled as Ω-Pattern, there are not enough digits to build up a P-Pattern (here again, for implementation the detecting will jump over fourth and fifth digits).

The comparison will start from next unsettled digit, which should be the ninth. From ninth to eleventh digit, the bending codes are "- - +". It is another match for P-Pattern, and there is only twelfth digit left.

Step 3. The unsettled digit of bending codes will be compared in order with C-Pattern, Z-Pattern, and U-Pattern, ending with L-Pattern. In the end all comparison are accomplished. Now the computer recognize the profile as a series of patterns: starting with a P-Pattern, followed by an L-Pattern, a Ω-Pattern, another P-Pattern and ending with another L-Pattern.

A bending sequence is generated after all bends are recognized. The work piece can be considered geometrically as P_1, L_1, Ω, P_2, L_2, and then the machining order should be $\Omega \rightarrow P_1 \rightarrow P_2 \rightarrow L_1 \rightarrow L_2$. Therefore the bending sequence of this product can be easily generated as:

$$5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 4 \rightarrow 12.$$

2.4.3 Drawback of Pattern Method

As shown in following figure, a Ω-Pattern under sequence $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$, the bending procedure carried out by press brake contains two "Flip" action against X-Axis. That means the operator should flip the sheet metal during the machining process, i.e. change the bending direction.

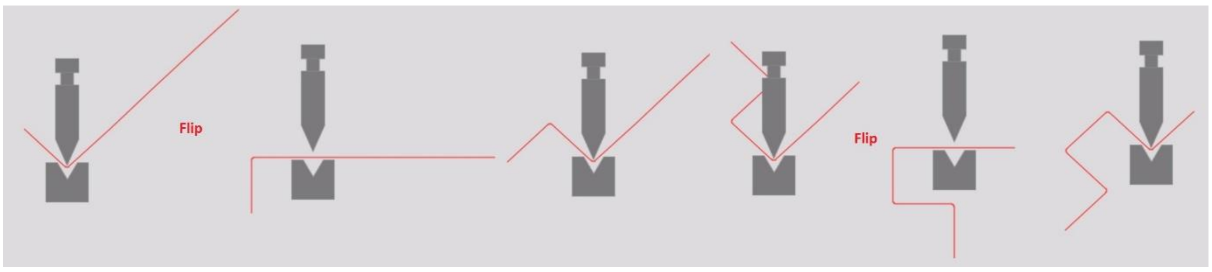


Figure 2.12 Original bending process of an Omega pattern with 2 Flip Operation

At the same time, a sequence of 1 → 4 → 2 → 3 only brings one “Flip” action.

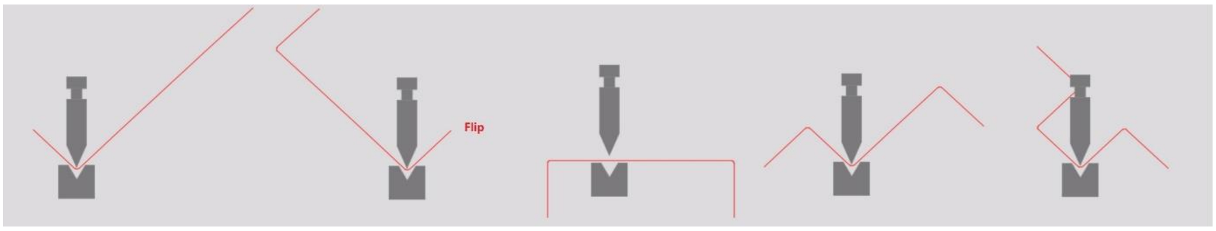


Figure 2.13 Improved bending sequence of an Omega Pattern with 1 Flip Operation

As explained in section 2.3.3, the amount of “Flip” is an important factor of sequence planning. Considering the operating skills of plant worker, the “Flip” action in real life can easily take more time than a stroke of press brake. This should be strictly taken into consideration, because by sequence planning, the algorithm should always be aiming at reducing time cost.

Another disadvantage of Pattern Method is the inside algorithm of sequence planning. Still take the above mentioned -Pattern as example, the sequence of 1 → 4 → 2 → 3 is not always feasible. Another sequence of 1 → 3 → 4 → 2 is capable of solving these situations. Even though the amount of flip operation is higher (3 instead of 1), the feasibility of the sequence has heavier weight in the object function. And this factor is not or can’t be easily implemented in pattern method, because the intension behind Pattern Method is to use a heuristic constrains of pattern to decrease the size of searching problem. A concrete calculation for each inside pattern sequence planning will hurt the strong point of this method. While this essay is written, an inside pattern sequence planning has been developed. Even though the time consuming is not of big problem because the small size of searching for each pattern (from 2 for U, and Z-Pattern to 4 for Ω-Pattern), and the inside algorithm is still not a silly enumerating, this solution is not quite optimized.

The biggest withdraw of Pattern Method lies in the unintelligent mechanism of recognition. Until now, this method still searching in the geometrical order of product, which makes little sense by being chosen as the best solution of searching.

The solution for the product discussed in section 2.4.2 could for example in certain conditions be improved to avoid collision between tools and sheet metal. As shown in following drafts, both plan are consisted with one Ω-Pattern, two P-Pattern, and two L-Pattern. But the assignment of P-Patterns can make a big difference. The one on the left side will generate a sequence $5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 4 \rightarrow 12$, while the one on the right side obtains sequence $5 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 10 \rightarrow 11 \rightarrow 12 \rightarrow 4 \rightarrow 9$. And the one on the right won’t cause a collision. This kind of decision will still relay on work experience of operator.

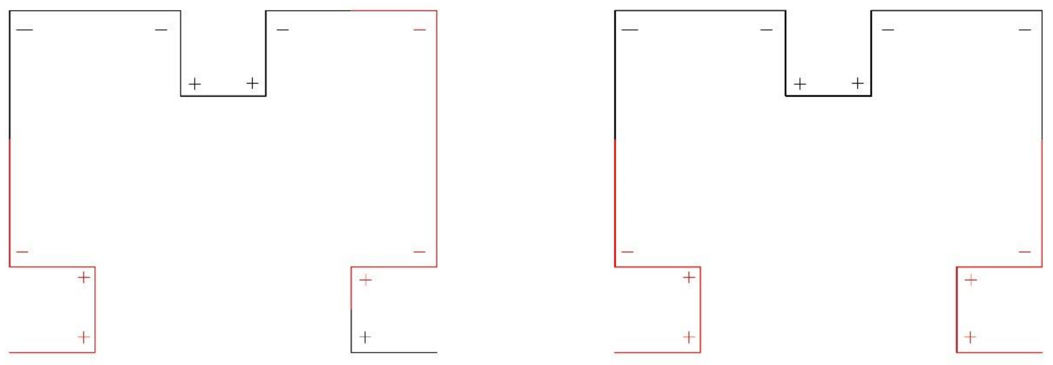


Figure 2.14 Two way to assign P-Patterns in this product

Even though this is an unpleasant disadvantage, but to improve this algorithm means to enumerate all possible assignment of pattern objects, which is no difference as a normal searching problem.

An alternative solution for pattern method, which is also applied in this essay is introducing a visualization of machining process, and alarm mechanism of collision. In this way, the operator can easily acknowledge the caused collision with any sequence, and manually modify each parameter or the whole sequence. As an essential element of computer-aided industrial tool, the visualization of manufacturing is for the press brake panel not an assistance of sequence planning, but an independent tool. Therefore the help from visualization won't actually cause more trouble during the implementation.

2.4.4 Visualization and Collision Error Alarm

In order to illustrate exact reality of machining reality, all kinds of parameter of press brake, tools, and gauge position need to be calculated, even the arc caused by bending should be interpreted. The operator not only can modify the bending sequence, but also improve the performance by adjusting the tools and press brake, for instance change the opening of lower tools to reach a better bend against spring off effect.

In contrast with Branch and Bound Method, there is no collision detection in sequence planning with Pattern Method at all. Instead, a collision detection function is called right after sequence is generated, before this process is visualized. Here the collision detection is rather a reminding mechanism than the decision to rule out unqualified solutions. But as the Pattern Method can satisfy almost most common production already, the manual fine turning will not be a too complicated mission. As by-product, position of back gauge could be generated as well.

2.4.5 Summary

As discussed in this section, the procedure of Pattern Method is rather simple and effective. The flaw of this method is mainly about the recognition of patterns. However the great things about it are also obvious. The most notable advantage of Pattern Method is of course its rapidity. One of solutions with Branch and Bound method could need about 40 seconds to finish the whole procedure (also including other steps besides sequencing) for a 10 bends product, and around 110 seconds for a 20 bends [Duflou 05]. On the other hand, by testing of the pattern algorithm developed in this essay, a

10 bends product could need only about 1 second. Even though the conditions of job could be quite different, and the branch and bound method can provide a better solution, the huge save of time consuming is a factor we can not ignore. Actually some leading company of press brake has already been using not perfect-solution method (could be like the Pattern Method) combined with collision alarm.

3 Implementation

The pattern method algorithm for sequence planning is implemented in C++ instead of PLC Programming, because in this way it’s easier to bring in manual impaction. The operator is always able to change the sequence according to local circumstances or even his or her preferences. As a well applied object oriented programming language, C++ is very suitable for this job.

A pseudo code of this procedure:

```
AutomaticSequencePlanning ()

    Initialize and load
    Get all the bends from user input into the list
    Get the angle of each bend
    SequencePlanningForPatternOmega ()
    SequencePlanningForPatternP ()
    SequencePlanningForPatternC ()
    SequencePlanningForPatternZ ()
    SequencePlanningForPatternU ()
    SequencePlanningForPatternL ()
    Sort all bends according to their bending sequence
    Draw punch, die, back gauge and press brake machine
    Draw the sheet metal
    For all the bends
       Draw the adjust mode, put sheet metal figure in right place
       IF Collision is detected
       Alarm
       EndIF
       Draw the bend mode, change the figure of sheet metal
       IF Collision is detected
       Alarm
       EndIF
    EndFor
    End
```

When the user draw a product, corresponding information are loaded in the bend list, and the recognition program carries on automatically. We take the recognition of Omega Pattern for example:

```
SequencePlanningForPatternOmega ()
    For all the bends available in the list
    If there is not suitable bends for Ω pattern
       End
    Else get bend direction of next 4 bends in a row
    If the bend directions satifisy "+--+" or "-++-"
       Recognize this Ω object and save the start point;
       For all Bends in this Omega object
       Assign bending sequence sucessively
       EndFor
       Start point of next search is end of this object
    EndIf
    EndFor
    End
```

Because the Omega Pattern is detected first, the searching for P-Pattern should jump over the Omega objects to avoid duplicate cognition.

```
SequencePlanningForPatternP ()
    For all the bends available in the list
        If any of next 3 bends in a row is member of known Ω object
             Jump this Ω object according to its start point
        Else If next 3 bends satifisy “+--”, “--+”, “++-” or “-++”
             Recognize this P object and save the start point;
             For all Bends in this P object
                 Assign bending sequence sucessively
             EndFor
             Start next search from end of current p object
        EndIf
    EndFor
    End
```

Conclusion and confirmation of the supervisor

This report includes the task I engaged at Beckhoff Automation GmbH & Co. KG for computational engineering science study at RWTH Aachen. Generally speaking, an algorithm of sequence planning for bending process executed by press brake is invented. The procedure is based on the pattern method.

The Pattern Method proposed by Aomura and Lin fit the objectives which need rapid response and compatibility with flexible manual control. It introduces patterns of L-Pattern, U-Pattern, Z-Pattern, C-Pattern, P-Pattern, and Ω-Pattern. Firstly computer assigns bending properties (angle, direction, and length of bending) to drawing of expected product. Next, with help of corresponding algorithm, product’s contour would be compared with patterns, and recognized as a permutation of different patterns. According to the manufacturing priority of each pattern, a feasible sequence is automatic generated.

To improve this method, so it suit the practical jobs better, several arrangement according to bumping bends and some inside-pattern sequencing are made. For the circumstance of not frequently replacing of tools and no collinear bends, pattern method demonstrates its great advantage of rapid response and accuracy, thanks to its simplicity. The algorithm is programmed in C++, and tested in product. Combining this method with visualization and collision detection, the drawback of Pattern Method can be avoid. In this case, an alarm would be launched when there is a collision, and the manual manipulation would be much easier and more intuitive.

More complicated algorithm can be discussed to improve sequencing performance for complex 3D products, or just to supply an optimum sequencing solution. By circumstance with more factors (for example taking amount of swap or flip operation or replacement of tools into consideration), the sequence planning problem turns usually into an optimization problem, Travelling Salesman Problem. The Branch and Bound method with Depth First Search mechanism bring in its power. Even though Branch and Bound method can bring a better solution, its massive time consuming is the reason of not being our preference.

List of Cited Literature

[Knoblach99] J. Fries-Knoblach, “Sheet metal working in the bronze age and iron age in southern central Europe”, *Proceedings of the 7th International Conference on Sheet Metal*, ISBN: 3-87525-110-5, pp.23-34, Erlangen, September 1999

[Ong97] S. K. Ong, L. J. De Vin, A. Y. C. Nee, and H. J. J. Kals, “Fuzzy set theory applied to bend sequencing for sheet metal bending”, *Journal of Materials Processing Technology*, vol. 69, no. 1–3, pp. 29–36, 1997

[Lin14] Alan C. Lin and Chao-Fan Chen “Sequence Planning and Tool Selection for Bending Processes of 2.5D Sheet Metals”, *Advances in Mechanical Engineering* vol. 2014

[Aomura02] S. Aomura and A. Koguchi, “Optimized bending sequences of sheet metal bending by robot,” *Robotics and Computer-Integrated Manufacturing*, vol. 18, no. 1, pp. 29–39, 2002

[Kannan08] T. R. Kannan and M. S. Shunmugam, “Planner for sheet metal components to obtain optimal bend sequence using a genetic algorithm”, *International Journal of Computer Integrated Manufacturing*, vol. 21, no. 7, pp. 790–802, 2008

[Cattrysse 06] D. Cattrysse, P. Beullens, P. Collin, J. Duflou, D. Van Oudheusden “Automatic production planning of press brakes for sheet metal bending“, *International Journal of Production Research,* vol. 44, No. 20, 15 October 2006, 4311–4327

[Verlinden 07] B. Verlinden, D. Cattrysse, H. Crauwels, J. Duflou and D. Van Oudheusden, “Sequencing and scheduling in the sheet metal shop”, *Multiprocessor Scheduling: Theory and Applications, Book edited by Eugene Levner,* ISBN 978-3-902613-02-8, pp.436, December 2007, Itech Education and Publishing, Vienna, Austria

[Wiener 03] Richard Wiener, “Branch and Bound Implementations for the Traveling Salesperson Problem - Part 2: Single threaded solution with many inexpensive nodes”, *Journal of Object Technology,* Vol. 2, No. 2, February-March 2003

[Duflou 05] Joost R. Duflou, “Computer aided process planning for sheet metal bending: A state of the art”, *Computers in industry*, vol. 56, no.7, jpp.663-772