AUTOMATED DESIGN OF SHEET METAL BENDING TOOLS

Ujval Alva and Satyandra K. Gupta
Mechanical Engineering Department and Institute for Systems Research
University of Maryland
College Park, Md-20742

ABSTRACT: Sheet metal bending is a process in which bends are formed using a combination of a punch and a die. Bending tools need to satisfy the following two criteria: (1) tools should be able to withstand bending forces, and (2) tool shapes should be such that there is no tool-part interference. In this paper, we describe a systematic methodology for automatically synthesizing a punch shape that can be used to produce more than one type of part. We create a parametric geometric model of possible punches. This parametric model describes a family of possible punch shapes. Using the geometric models of various given part types and a parametric punch model, we automatically generate constraints on punch parameters that eliminate the possibility of part-tool interference. We use mixed integer programming techniques to solve these constraints and identify parameters of a punch shape that can work for the given set of parts. Finally, we perform strength analysis of the designed punch to verify that the designed punch is capable of withstanding the bending forces.

INTRODUCTION

Sheet metal bending is a very popular process and is used by the sheet metal industry to create large number of mechanical products such as furniture panels, shelves, cabinets, housing for electro-mechanical devices etc. We have developed a two step approach that allows us to design a common punch shape for multiple parts. The idea behind this approach is as follows. Rather than directly matching part features to manufacturing process, we first identify the constraints imposed by a part feature on tooling that will be used to create that feature. In the second step, we gather all the constraints imposed by various features in various parts and perform constraint-driven tool design to identify the punch shape that works for multiple parts.

This paper describes in detail the implementation of the mathematical formulation presented in our ASME's Design for Manufacturing Conference paper[1]. As described in the previous paper, bending tools need to satisfy the following two criteria: (1) tools should be able to withstand bending forces, and (2) tool shapes should be such that there is no part-tool interference. For a detailed description of sheet-metal bending processes, readers are referred to handbooks on this subject [2-4]

Our solution methodology involves creating a parametric geometric model of possible punch shapes. This parametric model describes a family of possible punch shapes. Using the given set of part geometries and the parametric punch shape model, we automatically generate constraints on tool parameters that eliminate the possibility of part-tool interference. We use mixed integer programming techniques to solve these constraints and identify parameters of a punch shape that can work for the given set of parts. Finally, we perform strength analysis of the designed punch shape to verify that the designed punch shape is capable of withstanding the bending forces. Using a single punch to bend multiple parts reduces the setup time and costs in the small batch manufacturing environment.

OVERVIEW OF SOLUTION METHODOLOGY

Problem Formulation

We assume that we are given the following information as input:

- *Parametric model of the punch*. Figure 1 shows a parametric model of the punch. Usually, there are restrictions imposed on values of these variables by punch manufacturers. These restrictions are captured as constraints on punch parameter values.

- *Geometric models for a set of parts.* Currently we restrict ourselves only to 2.5D parts. These types of parts are quite often referred to as sash type part in the sheet metal industry.

- *Operation sequences for each part in the given part set.* For each part, we assume that we are given an operation sequence that specifies the order in which various bending operations will be performed.

Our goal is to find a punch shape that is capable of producing each and every part in the given set of parts. If we can find such a punch, then the setup time can be significantly reduced and all the parts in the given set can be produced in the same setup.

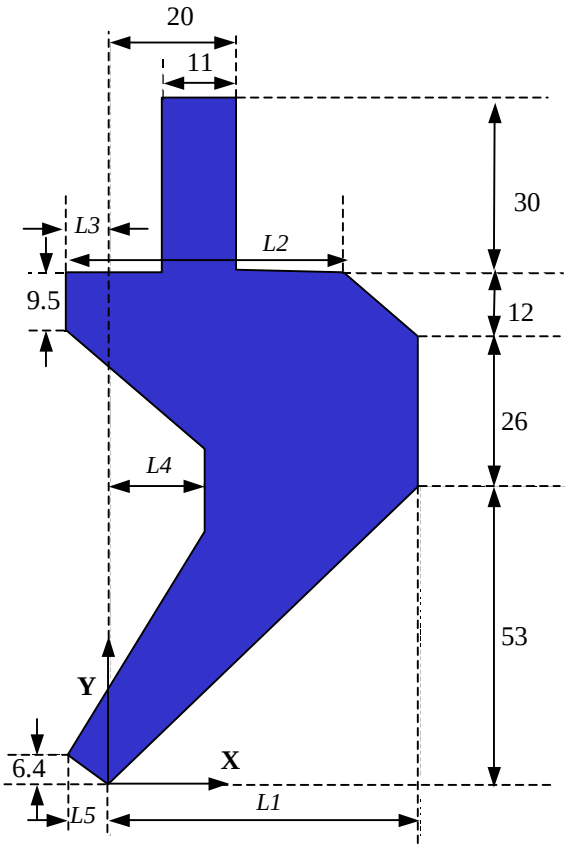

Figure 1. Parametric Model of a Punch

Overview of Algorithm

We are using the following approach to solve this problem:

1. *Generate constraints on punch parameters*. We first generate constraints on the punch parameters by performing interference checks between parametric punch shape and various intermediate workpiece shapes resulting during various bending operations.

2. *Find a punch shape that does not intersect with any intermediate workpiece shape and has the maximum strength*. As a second step, we use the constraints on punch parameters and try to find a punch shape that satisfies all interference constraints and maximize the punch strength. We use mixed integer programming formulation for carrying out this step.

3. *Verify that the designed punch can withstand stresses resulting from the bending forces*. Finally we verify if the designed punch shape can withstand the bending forces. We use Finite Element Analysis to carry out this verification.

PUNCH PARAMETER CONSTRAINT GENERATION

Generating Constraints to Eliminate Interference

As described in our previous paper[1], we can write the constraints for eliminating the intersection between two lines in a symbolic form. Let us assume that the first line is defined by end points (x_1, y_1) and (x_2, y_2). Similarly, let the second line be defined by end points (x_1', y_1') and (x_2', y_2'). Let $D = (x_2' - x_1')(y_2 - y_1) - (y_2' - y_1')(x_2 - x_1)$.

In order to eliminate the possibility of intersection between these two line segments, *one of the following four constraints* must be satisfied:

- Constraint 1:

If $D > 0$, then

$$x_2' y_1' - x_1' y_2' + x_1 y_2' - x_1 y_1' - y_1 x_2' + y_1 x_1' < 0 \quad (1)$$

If $D < 0$, then

$$-x_2' y_1' + x_1' y_2' - x_1 y_2' + x_1 y_1' + y_1 x_2' - y_1 x_1' < 0 \quad (2)$$

- Constraint 2:

If $D > 0$, then

$$x_2' y_2 - x_2' y_1' + x_1' y_2' - x_1' y_2 - y_2' x_2 + y_1' x_2 < 0 \quad (3)$$

If $D < 0$, then

$$-x_2' y_2 + x_2' y_1' - x_1' y_2' + x_1' y_2 + y_2' x_2 - y_1' x_2 < 0 \quad (4)$$

- Constraint 3:

If $D > 0$, then

$$x_1 y_2 - y_1 x_2 - x_1' y_2 + x_1' y_1 + y_1' x_2 - y_1' x_1 < 0 \quad (5)$$

If $D < 0$, then

$$-x_1y_2 + y_1x_2 + x_1'y_2 - x_1'y_1 - y_1'x_2 + y_1'x_1 < 0 \quad (6)$$

- Constraint 4:

If $D > 0$, then

$$x_2y_1 - x_1y_2 + x_2'y_2 - x_2'y_1 - y_2'x_2 + y_2'x_1 < 0 \quad (7)$$

If $D < 0$, then

$$-x_2y_1 + x_1y_2 - x_2'y_2 + x_2'y_1 + y_2'x_2 - y_2'x_1 < 0 \quad (8)$$

Using the constraint equations described above, for every pair of line segments on the punch profile and the part profile, we can write a set of constraints that eliminates the possibility of intersection between these two line segments. Therefore, if we consider all pairs of line segments on punch profiles and workpiece shapes (i.e., part shape during various bending operations), then we can generate a comprehensive set of constraints that eliminate the possibility of intersection between the punch shape and the various workpiece shapes.

The intersection constraints defined by the above equations are (1) conditional and (2) disjunctive in nature. Therefore, standard linear formulation based on conjunctive constraints[8] does not work with these types of constraints. We use additional integer constraint control variables to convert these constraints into standard conjunctive constraints. These additional integer variables selectively include or exclude constraints. Our methodology for introducing these integer constraint control variables is described in detail in our ASME's Design for Manufacturing Conference paper[1]. After converting these conditional and disjunctive constraints into standard conjunctive form, we use standard mixed integer programming formulation to identify punch parameters. The Integer programming engine automatically tries various appropriate combinations of constraint control variables to select the appropriate constraint.

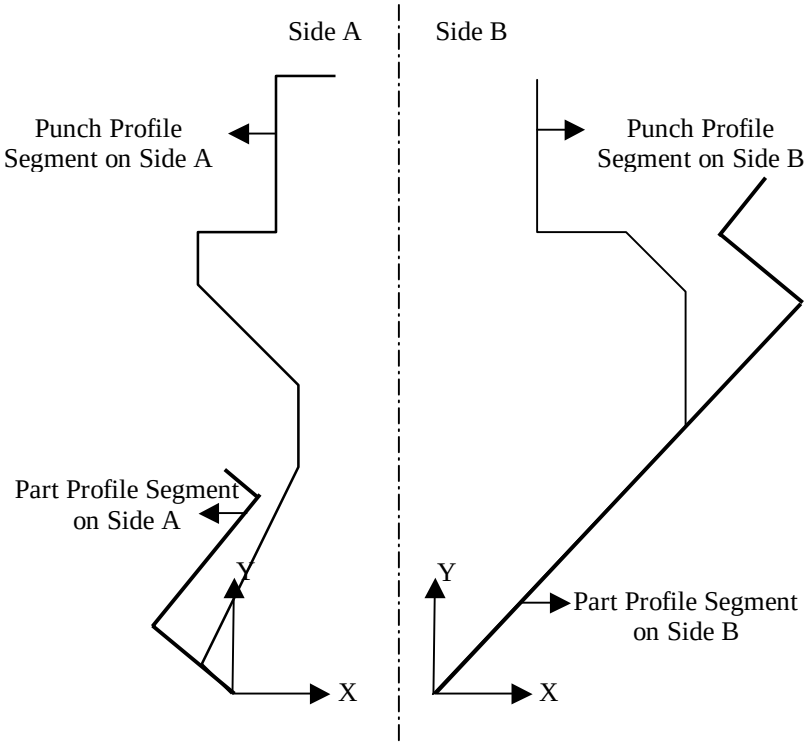

Figure 2. Partitioning Punch and Part profiles

Heuristics to Eliminate Unnecessary Constraints

The punch parameter constraint generation involves automatically generating constraints for eliminating intersection between every line segment of every workpiece shape with every line segment of the punch. This process results in a very large number of constraints. For example, a bending operation involving a gooseneck punch made up of 13 line segments and a part having 7 line segments in its cross section and 6 bends will result in 546 pairs of line segments. Such a large number of line segment pairs may generate a very large number of disjunctive and conditional constraints. This results in a large and time consuming optimization process. In order to reduce the number of constraints, we have developed two heuristics to remove redundant constraints. The following sections describe constraint reduction heuristics.

Profile Partitioning Heuristics. The first heuristic that eliminates redundant constraints involves dividing the punch and part profiles into two segments. The extreme points on a punch divide the punch profile into two profile segments as shown in Figure 2. Similarly the part profile can also be divided into two profile segments at the bend. The division of the profiles of the punch and part into segments helps in reducing the number of constraints in the following manner. The part-profile segments on side A need to be checked for interference with only the punch-profile segments on side A (see Figure 2). Similarly, the part-profile segments on side B need to be checked for interference with only the punch-profile segments on side B. This heuristic is based on the following observation. Any part-profile line segment on side A will intersect with the punch-profile line segment on side A before intersecting with the punch-profile line segment on side B. Therefore, if there is an intersection between the part-profile segment and punch-profile segment on side A, then we need not check for intersection between the part-profile segment on side A with the punch-profile segment on side B. This method reduces the total number of constraints that need to be included in the optimization process.

Heuristic for Identifying Redundant Constraints. The second heuristic is to eliminate constraints that correspond to those pairs of line segments that will never intersect. This is done to ensure that all constraints that play no role in finding the punch parameters are eliminated. As described in the section titled PROBLEM FORMULATION, the parameters that define the punch only have a certain range of values based on manufacturer specifications.

Similar to the non-intersection conditions described in the section titled GENERATING CONSTRAINTS TO ELIMINATE INTERFERENCE, we can also write intersection conditions that must be satisfied by two line segments in order for them to intersect. Such intersection conditions are then used to eliminate redundant constraints. If a line segment on the workpiece and a line segment on a punch do not satisfy intersection condition for all values in the given punch parameters' range, then it can be safely assumed that these two line segments will never intersect. Hence, all constraints that would otherwise have been generated can now be safely eliminated. For example, consider two line segments as shown in Figure 3. The first line segment is defined by end

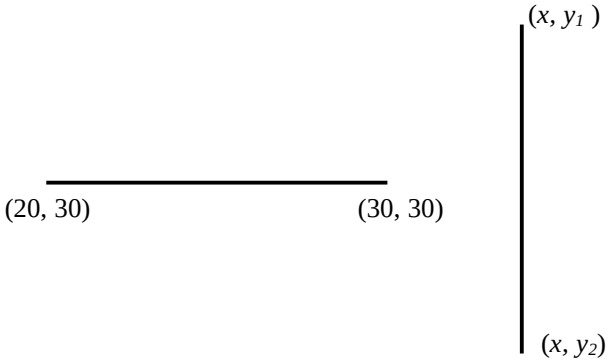

Figure 3. Heuristic for Identifying Redundant

points $(20,30)$ and $(30,30)$. Similarly, let the second line segment be defined by end points (x, y_1') and (x, y_2'). If the parameter range of x is $40 \le x \le 50$, then it can be concluded that the two line segments will never intersect.

This heuristic when used in tandem with the profile partitioning heuristic helps to eliminate a very large percentage of the redundant disjunctive constraints that were originally generated. From our implementation, we have observed approximately 75% reduction in the number of constraints by using these two heuristics. This reduction in constraints helps in making the punch design more tractable and easier to solve. The procedure of implementing these constraint reduction heuristics is described in detail in the next section.

OVERVIEW OF IMPLEMENTATION

The two main inputs are described below:

1. *The CAD Model of the parts*. The CAD model of the parts contains all the information about the part. The information includes the shape of the part, the dimensions of the part and the bending sequence that needs to be followed to produce the part. The CAD model information is stored in two types of input files, the first type of file contains the dimensions and the bend angles of each part. We only allow bend angles to be +90 degrees or –90 degrees. The second type of input file contains the bending sequence that needs to be followed to produce the part.

2. *The Parametric Shape of the Punch*. The shape of the punch can be parameterized as shown in Figure 1. The punch that was parameterized was the 90-degree gooseneck type of punch. The parameters that determine the shape of the punch were labeled as L_1, L_2, L_3, L_4 and L_5. The objective is to find the values of these parameters such that there is no interference between the part and the punch.

Figure 4 shows the major steps of our implementation graphically. As shown in this figure, the first C++ program takes the two input files containing the part dimensions and bend sequence information for every part, and automatically generates output files containing the coordinates of the various intermediate workpiece shapes that are generated during the bending of the part. For each part, the number of output files generated equals the number of bends in that particular part. Each output file contains the coordinate points of each line segment making up the intermediate workpiece.

These output files form the input for the second C++ program that automatically generates all the constraints that prevent part-tool interference. This program works in the following manner:

- This program takes various line segments from intermediate workpiece and pairs them with various line segments in the punch model, and generates the constraints as described in the constraint formulation discussed in the section titled GENERATING CONSTRAINTS TO ELIMINATE INTERFERENCE.

- The profile partitioning heuristic is a part of this program and is used to eliminate redundant constraints. The points making up the punch profile line segments are divided into side A and side B points as shown in Figure 2. The points making up the workpiece line segments are similarly divided into positive and negative points, depending on their position on x-axis (we assume that the punch tip is always located at $x = 0$ and we translate the intermediate workpiece shape appropriately). Interference check is carried out only between the punch profile line

segments on side A with the negative workpiece line segments and between punch profile line segments on side B with the positive workpiece line segments.

- The second heuristic is to eliminate constraints that correspond to those line segments that will never intersect. The program calls the AMPL constraint solver to carry out this function. A set of four constraints is generated based on intersection conditions. These four constraints along with the constraints on the parameters form the input file for AMPL (A Modeling Language for Mathematical Programming). AMPL solves the problem and checks if there exists a solution that satisfies these constraints. If there is no output solution at the end of this step, then it can be safely assumed that in the parameter range considered, there can never be interference between the two line segments. We prune such constraints.

- All the points that are not eliminated thus far then form the input for the final generation of constraints. These constraints are the necessary constraints that determine the parameters of the punch. The necessary constraints that are automatically generated at the end of the program are saved to a file that forms the input file to AMPL.

AMPL is a software package that solves a variety of optimization problems. It is a modeling language for linear, nonlinear, and integer programming problems. AMPL works with several solution generators like CPLEX, MINOS and OSL. AMPL can be used in both the interactive mode or in the batch environment. We use the batch environment where a file containing the objective function and constraints form the input to AMPL. AMPL is then used to solve this mixed integer formulation. The output of AMPL is also saved to a file.

In our formulation, we use punch strength as the objective function. Writing punch strength as a

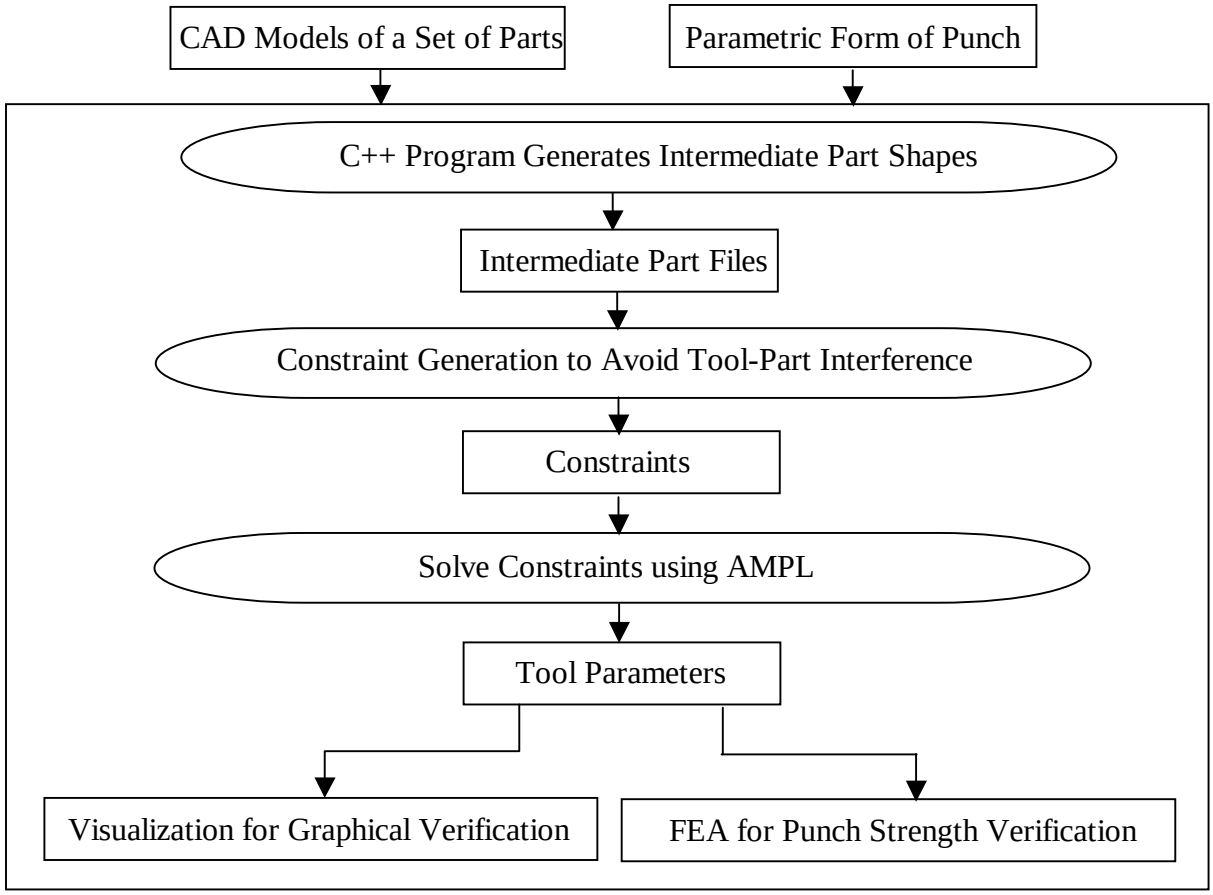

Figure 4. Implementation procedure

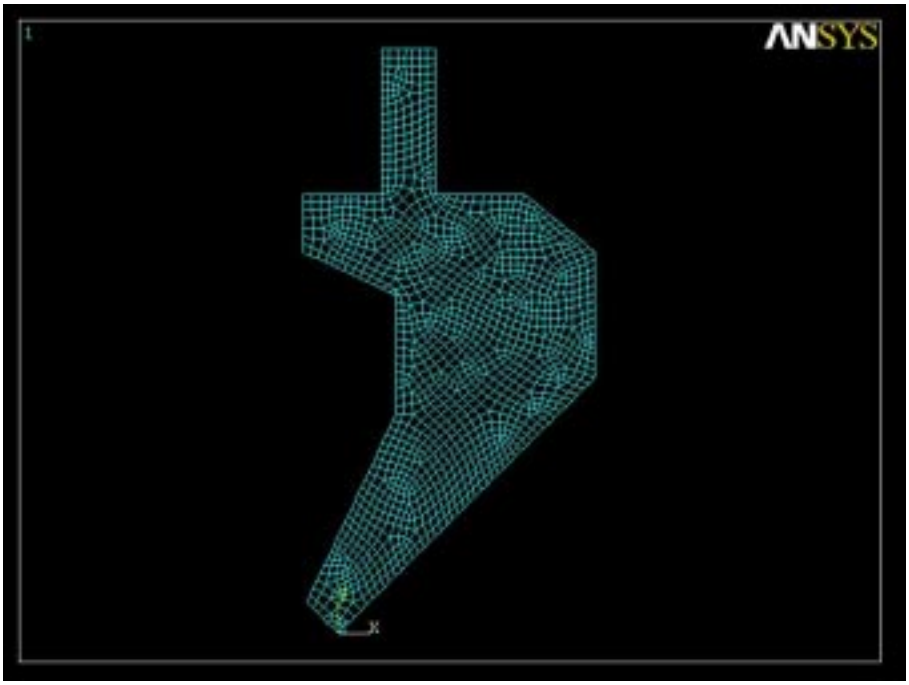

Figure 5. Punch Meshed using ANSYS

function of its shape parameter usually leads to non-linear objective functions. Currently, we are using linear programming techniques. Therefore we use *L1-L4* as an approximation for the punch strength. The value *L1-L4* increases the width of the main punch body and therefore increases the punch strength. We plan to experiment with non-linear programming in the next version of our program. This will allow us to use objective functions that will closely relate to the punch strength.

The results of the optimization step determine the parameters of the punch. The resultant parameters are those values for the parameters of the punch that ensure that there is no interference between various parts and the punch.

To graphically visualize the resultant punch we have used ACIS (a geometric modeling package from Spatial Technology). The resultant parameters are used to generate the shape of the punch. The punch shape is stored as an ACIS ".sat" file. Similarly the various intermediate part shapes are also stored as ACIS ".sat" files. These ACIS ".sat" files are retrieved in Test Harness and visually verified to ensure that there is no interference between the part and the punch.

It is important that the punch generated is able to withstand the forces of bending. We use Finite Element Analysis to verify the strength of the punch. The modeling and analyzing capabilities of ANSYS is used to carry out the strength verification of the resultant punch. Once the punch has been modeled and the material properties entered, we use ANSYS to automatically mesh the punch using a fine mesh. Figure 5 shows the meshed punch that is generated using ANSYS. The boundary conditions and the forces acting on the punch during the bending process are then entered. The bending force is assumed to be uniformly distributed along the radius of the punch tip. The formula for calculating the force of bending has been discussed in detail in our previous paper.[1] ANSYS carries out the finite element analysis of the resulting punch. The result of the analysis verifies if the punch material can withstand the stresses generated during the bending process with the required factor of safety.

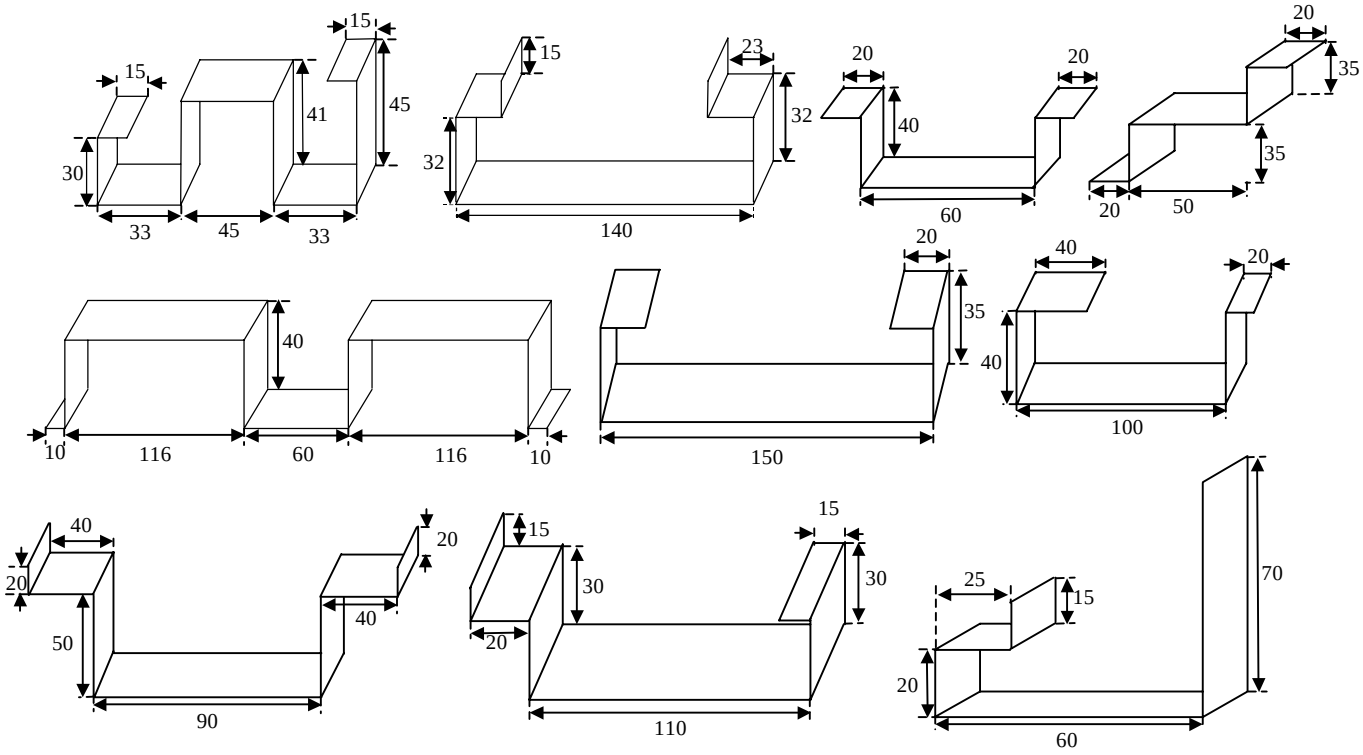

Figure 6: Given Set of Parts

EXAMPLE

Let us consider the ten parts shown in Figure 6. Let us assume that the material thickness for these parts is 1.5mm. For these parts, using the solution methodology described in previous sections, we have designed a punch.

We used the following constraints on the punch parameters:

$40 \leq L1 \leq 55$, $35 \leq L2 \leq 42$, $7 \leq L3 \leq 10$, $12 \leq L4 \leq 25$, $5 \leq L5 \leq 6.5$, $30 \leq L1 - L4$, $5 \leq L1 - L2 \leq 15$, $45 \leq L2 + L3 \leq 50$

The other constraints are the ones that are generated as a result of the interference check between the punch and the multiple parts. The designed punch has the following set of parameters:

- $L1 = 53$, $L2 = 38.4$, $L3 = 7$, $L4 = 12$, $L5 = 6.4$

We have visually verified that this punch does not intersect with any intermediate part shape. We assumed that the punch material is Tool Grade Steel having a HRC of 43-48. We carried out FEA to verify that this shape can safely withstand the bending forces generated during bending of these parts.

CONCLUSIONS

This paper describes a systematic implementation procedure for synthesizing punch shapes that do not intersect with a given set of part geometries and at the same time have enough strength to withstand bending forces. We believe that our approach will help process planners in selecting

punch shapes that will work for multiple types of parts. This will help in reducing the setup times and setup costs for small batch manufacturing environment.

We plan to extend our work in the following areas to overcome current limitations:

- *Non linear programming Formulation*. We plan to use non-linear programming techniques to allow us to use non-linear objective functions.

- *Three Dimensional Part Shapes*. Our current work is applicable to 2.5D parts. We plan to extend our approach to 3D part shapes.

- *Incorporating alternative operation sequences in generation of constraints.* Currently we only consider a given operation sequence for a part. We plan to extend our work to consider alternative operation sequences for a given part.

ACKNOWLEDGEMENTS

This research has been supported by the NSF grant DMI9896255. Opinions expressed in this paper are those of authors and do not necessarily reflect opinion of the National Science Foundation.

REFERENCES

1. Alva, Ujval and Gupta, Satyandra K. Automated Punch Shape Synthesis for Sheet Metal Bending Operations. *ASME Design Engineering Technical Conference*, Las Vegas, Nevada, 1999.

2. Amada Sheet Metal Working Research Association. *Bending Technique*. Machinist Publishing Company Limited, first edition, 1981.

3. Benson, S.D. *Press Brake Technology: A Guide to precision sheet metal bending*. Society of Manufacturing Engineers, 1997.

4. Eary, D.F. and Reed, E.A. *Techniques of Pressworking Sheet Metal-An Engineering Approach To Die Design*, 1974

5. Gupta, S.K, Bourne, D.A and Kim, K.H and Krishnan, S.S. Automated process planning for sheet metal bending operations. In *Journal of Manufacturing Systems*, 17(5):328--336, 1998.

6. Radin,B, Shpitalni,M and Hartman, I. Two stage algorithm for rapid determination of the bending sequence in sheet metal products. In *ASME Design Automation Conference*, Irvine, CA 1996.

7. Smith, J.S, Cohen, P.H, Davis, J.W and Irani, S.A. Process plan generation for sheet metal parts using an integrated feature-based expert system approach. *International Journal of Production Research*, 30(5):1175--1190, 1992.

8. Wolsey, L.H. *Integer Programming*. John Wiley & Sons Inc,1998.