

This document contains the draft version of the following paper:

S.K. Gupta. Sheet metal bending operation planning: Using virtual node generation to improve search efficiency. *Journal of Manufacturing Systems*, 18(2):127-139, 1999.

Readers are encouraged to get the official version from the journal's web site or by contacting Dr. S.K. Gupta ([skgupta@umd.edu](mailto:skgupta@umd.edu)).

# Sheet Metal Bending Operation Planning: Using Virtual Node Generation to Improve Search Efficiency

Satyandra K. Gupta

Mechanical Engineering Department and Institute for Systems Research  
University of Maryland  
College Park, MD 20742

**Keywords:** manufacturing operations planning, state-space search, sheet-metal bending, and setup planning.

## Abstract

A large number of manufacturing operation planning problems can be formulated as state-space search problems. In case of sheet-metal bending operation planning, processing a search node involves extensive geometric reasoning. Such computation-intensive node-processing limits the number of search nodes that can be expanded in a reasonable amount of time, making it difficult to solve real-life operation planning problems.

In this paper, we describe a scheme to speed up operation planning by virtual generation of state-space nodes. In this scheme, we eliminate unnecessary computation at the time of node generation by extracting the required information from already generated nodes. Although generation of two different search nodes rarely involves identical computation steps, there is considerable overlap in node generation steps. We have divided the node generation step into a number of computation subproblems. When we need to generate a new node, we first try to see if any of the node generation subproblems have been solved for any of the already generated nodes. If any subproblem has already been solved for some other node, then we use the solution of that subproblem to save computation time. In such cases, we do not perform node generation computation steps, and therefore we call such node generation virtual.

The scheme presented in this paper increases the node generation capability and allows us to consider many more search nodes. The ability to consider more search nodes helps us in solving more complex problems and finding better operation plans.

## 1 Introduction

In order to gain acceptance in today's competitive industry, automated operation planning systems need to be able to solve complex problems and produce high quality plans. More-

over, there is an increasing demand for faster response time to make the operation planning function suitable for interactive applications (e.g., manufacturability analysis during design stage).

State-space search techniques have been used to solve a variety of operation planning problems[10, 13, 19]. In state-space search formulations, the starting blank/stock is considered the initial state and the final target part is considered the goal state. Various manufacturing operations act as the search operators that transform one search state to another search state. The operation planning problem is defined as the problem of finding a sequence of operations that transform the initial state to the goal state. In order to use state space search techniques effectively and efficiently, we need to overcome the following two challenges.

1. *Very high node processing time.* Generation of a new state requires significant geometric reasoning. In most sophisticated process planning systems, this step involves making queries to a geometric kernel. Such computation is time consuming and leads to very high node processing time, making operation planning computation-intensive.
2. *Large number of search-states.* In most operation planning problems, the search space is very large and it typically grows exponentially with the number of features on the part. Slower node processing makes it difficult to explore large portions of the search space.

In order to do operation planning for a complex part, we need to considerably improve node processing time to allow effective exploration of the state space. In this paper we describe techniques for speeding up sheet metal bending operation planning.

In general, tooling and setup parameters of a bend depend on the operation sequence. However, in practice only a selected number of bends in the part can influence bending parameters of a given bend. This means that the parameters of a bend may remain unchanged in many different operation sequences. Existing search schemes for exploring alternative operation sequences result in repetitive calculations.

Bend-lines that do not interact mutually are not likely to affect the manufacturing operation of each other in any significant way. In many cases, simple geometric tests can be used to rule out the possibility of interactions among two bends. This observation raises the following question. Can we perform the necessary computations for a search node and reuse results of those computations later for other search nodes without going through computation intensive steps? It turns out that the answer to this question is affirmative.

Although the generation of two different search nodes rarely involves completely identical computation steps, there is considerable overlap in node generation steps. In this paper, we show that the reuse of computation steps is possible and propose a scheme for handling sheet-metal bending operation planning. The scheme presented in this paper involves decomposing the node generation problem into a number of computation subproblems. When we need to generate a new node, we first try to see if any of the node generation subproblems has been solved for any of the already generated nodes. If the answer is affirmative, then we

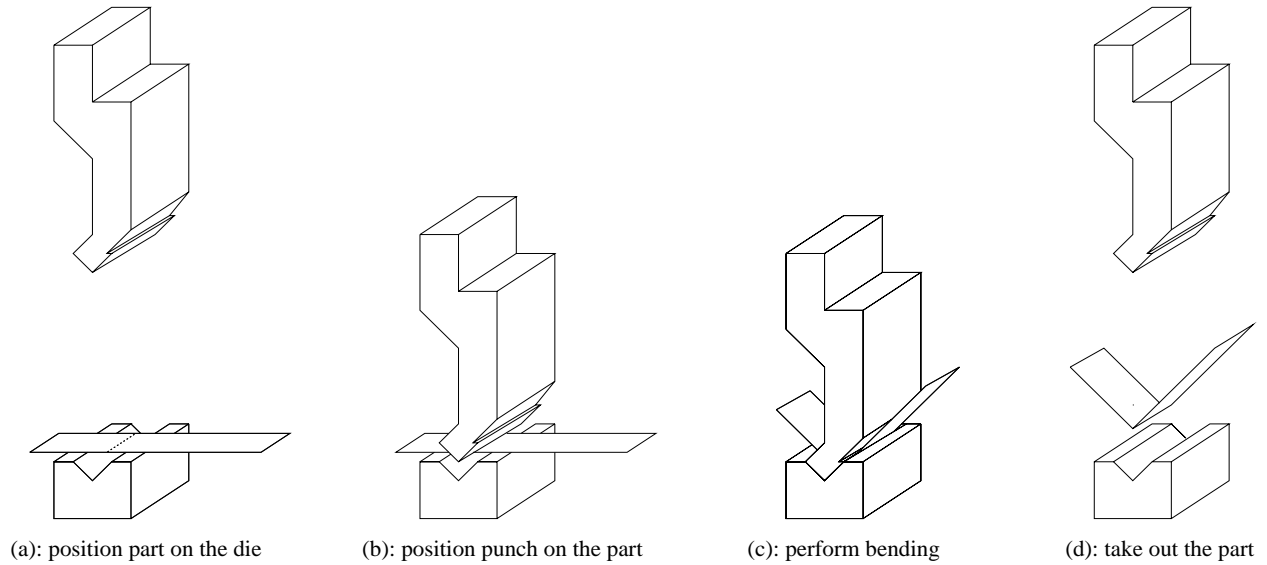


Figure 1: Sheet metal bending process.

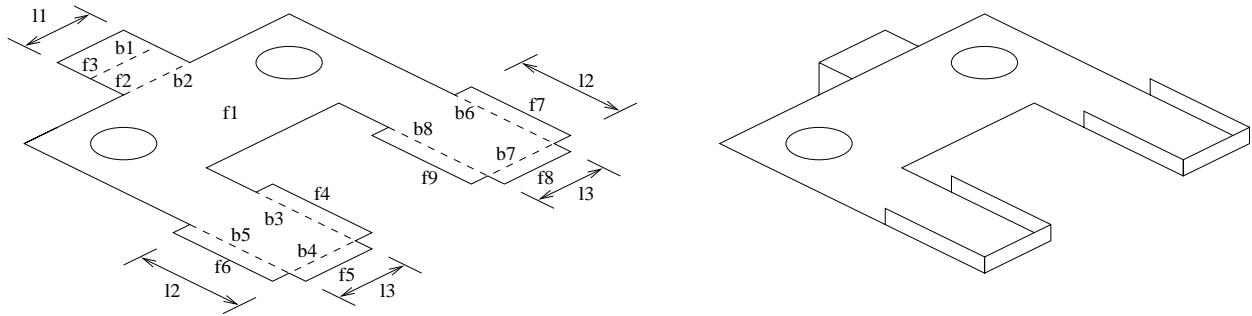


Figure 2: A sheet metal bending operation planning problem.

use the solution of the already solved subproblem and save computation time. Nodes that are generated through reuse of precomputed results do not go through real node generation steps, therefore we use the term *virtual node generation* to describe such node generation.

The reuse of subproblem results is a similar concept to data caching, which is used to speed up data retrieval. However, recognizing when results of an already solved subproblem can be used for a given subproblem is a challenging task. The efficiency of a result reuse scheme depends on the effectiveness of the conditions that are used to identify a match between various subproblems. Moreover, there is no unique way to decompose a problem into subproblems. Some decompositions may work better than others. Therefore, selecting the right problem decomposition is also very important.

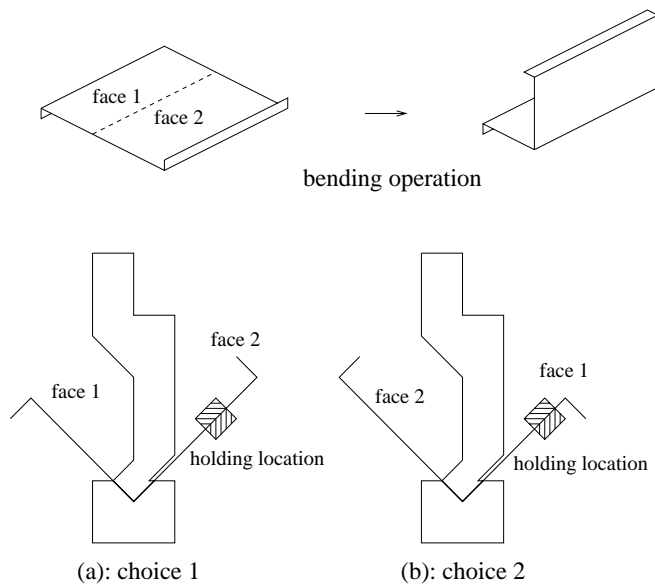


Figure 3: Two different choices for doing a bending operation.

## 2 Background

### 2.1 Sheet Metal Bending Operation Planning

In sheet metal bending, a flat part is bent using a set of punches and dies. We will refer to a set of punches and dies as tools. The punch and the die are mounted on a pressbrake, which controls the relative motion between the punch and die, and provides the necessary bending pressure. Figure 1 illustrates the sheet-metal bending process. For a detailed description of sheet-metal bending processes, readers are referred to handbooks on this subject [2, 3, 21]. In a typical problem, we are given a final part and a starting flat part. The flat part needs to be bent along the bend lines to create the final part. Figure 2 shows an example part and the corresponding starting flat. In this Figure,  $b_1, b_2, b_3, b_4, b_5, b_6, b_7$ , and  $b_8$  are bend-line numbers and  $f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8$ , and  $f_9$  are face numbers.

It should be noticed that each bending operation can be performed in two different ways. Each bend line connects two faces. Either of these two faces can be kept outside the pressbrake, resulting in two different possibilities for orienting the part in the pressbrake. Figure 3 shows two different ways of performing the bending operation. Many times the intermediate workpiece geometry is such that only one of these choices will work.

Intermediate workpiece shapes imposes restrictions on the tooling length that can be used to perform a bending operation. We refer to such constraints as setup constraints. Figure 4 shows an example of setup constraint. In this case, tabs on both side of the bend may intersect with the die during bending. Therefore, this bend cannot be performed on an arbitrarily long tooling stage. The minimum tooling stage length for this operation is  $L$ . In order to avoid interference between the tool and the intermediate workpiece, the maximum

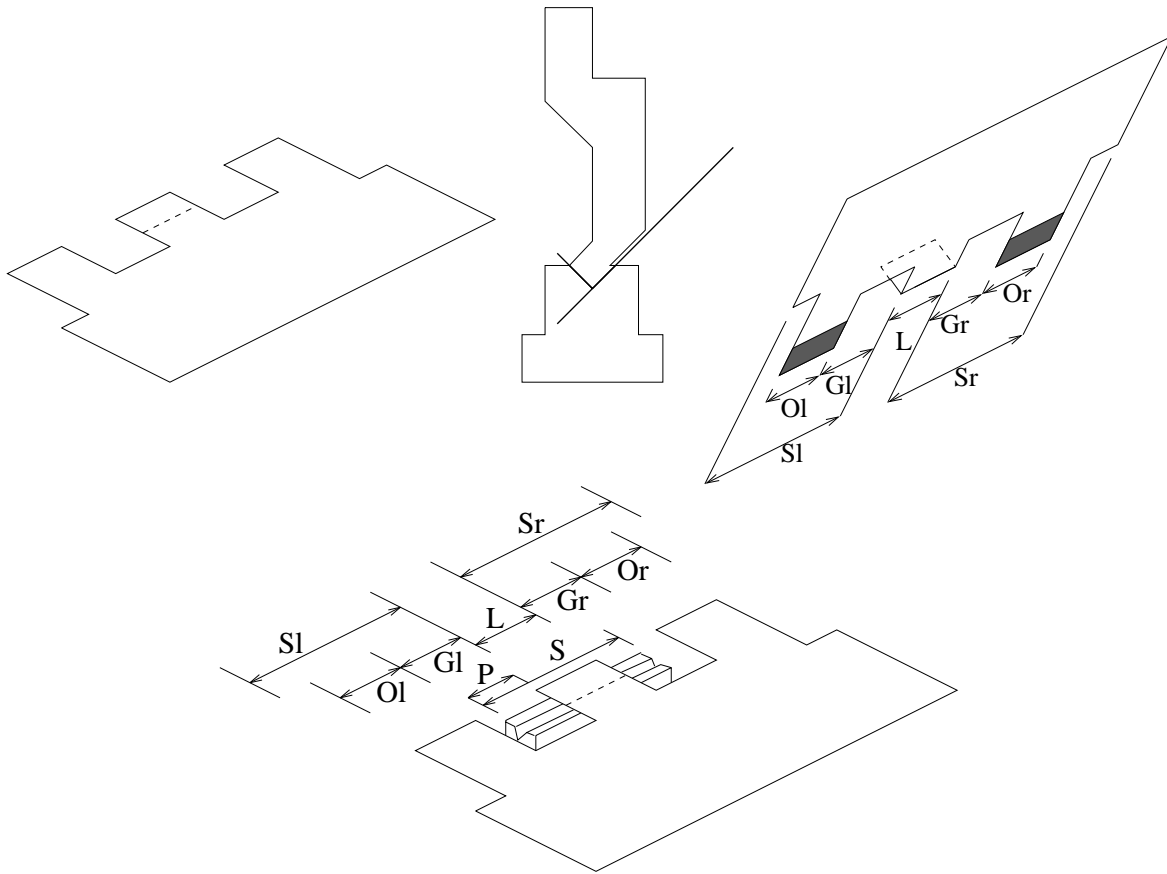


Figure 4: Setup constraint parameters.

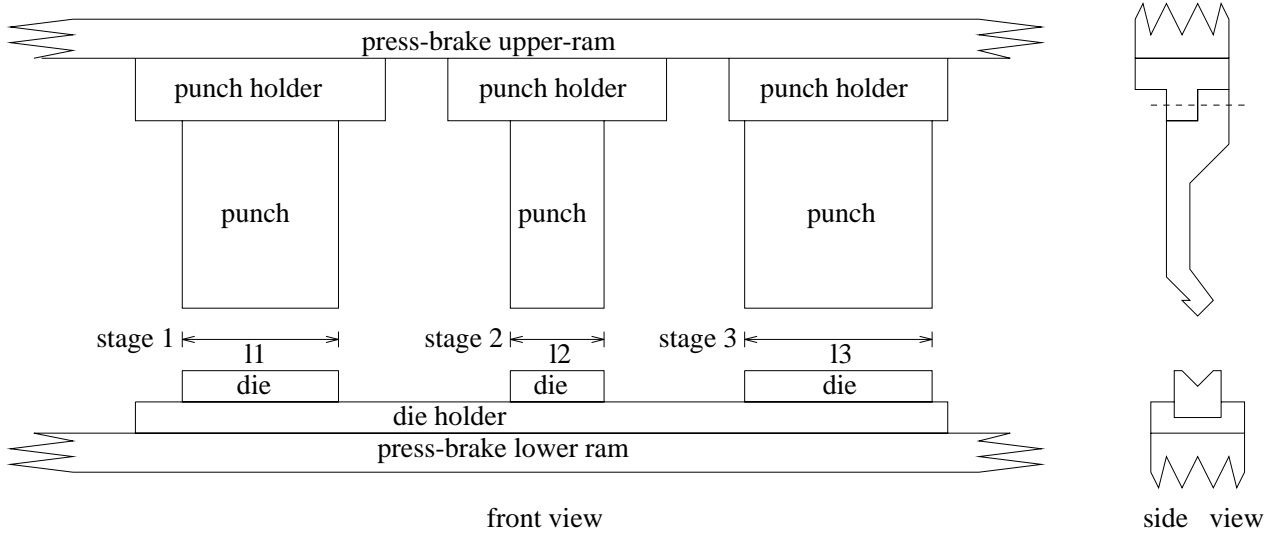


Figure 5: A bending pressbrake setup.

allowed tooling stage length is  $Gr + Gl + L$ . Besides these restrictions, adjacent stages should also clear the safety margins. For each bend-line, we use six parameters (shown in Figure 4) to capture the setup constraints associated with the bend-line. These parameters define the following setup constraints:

$$Gr + Gl + L \geq S \geq L$$

$$Gl \geq P$$

$$Gr \geq S - P - L$$

$$Sr \leq S - P - L + Dr$$

$$Sl \leq P + Dl$$

where,  $Dl$  is the distance between the current stage and the left adjacent stage.  $Dr$  is the distance between the current stage and the right adjacent stage.  $L$  is the length of bend-line.  $S$  is the length of tooling stage.  $P$  is the relative position of the bend line with respect to the left edge of the tooling stage.

In sheet metal bending operation planning, the objective is to find a sequence of operations that minimizes overall production time. Overall production time consists of setup time and operation time. Because of its strong influence on the quality of an operation plan, setup planning is an integral part of operation planning. A pressbrake setup consists of one or more tooling stages. Figure 5 shows a pressbrake setup consisting of three tooling stages. Depending upon its intermediate workpiece geometry, each bending operation is assigned to the appropriate stage in the setup.

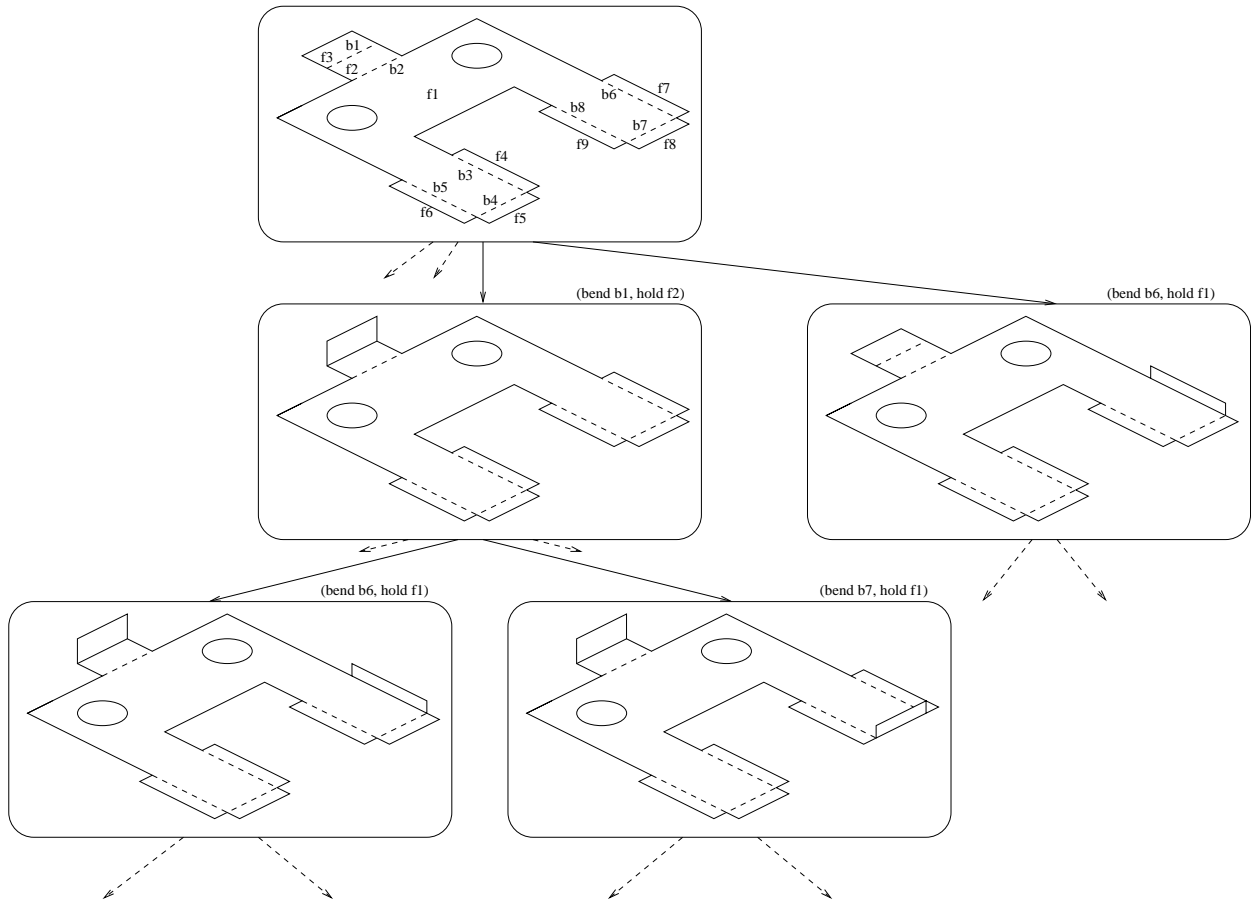


Figure 6: Partial search state-space for the part shown in Figure 2.

## 2.2 Related Work

Automated process planning is a very active research area. Over the last two decades, many different automated process planning systems have been developed. Surveys of various systems and techniques can be found in [1, 11, 4].

Many different AI techniques have been used for process planning. Wang and Wysk developed a knowledge based technique for process planning [20]. Hayes and Wright developed a rule based technique for process planning [12]. Nau developed a frame-based technique for process planning [14]. Increasingly, heuristic search is being used as the main problem solving technique [10, 13, 19] in process planning systems. The exact nature of search algorithm and heuristics depend on the particular planning problem being solved.

A number of systems have been developed to automate various process planning functions for sheet metal parts [5, 16, 18, 22]. Many of these system attempt to handle a wide variety of sheet metal processes and attempt to sequence various operations based on high level interactions among them.

The following two research efforts specifically addressed the problem of finding an optimal operation sequence for sheet metal bending. de Vin *et al* [6, 7] have developed a process planning system for finding a feasible operation sequence. Their system addresses the part-tool collision and tolerance constraints. In addition, they use heuristics for minimizing material handling time to guide the search. Radin *et al* [17] have developed a variation of branch and bound search technique to find the optimal operation sequence. They first try to find a feasible solution and then try to improve it in subsequent iterations. Their cost criteria is based on the number of tool changes and material handling time.

Both of the above described efforts use search based techniques to identify optimal operation sequences. Therefore, virtual node generation techniques described in this paper can be used in both of these cases to improve computational efficiency.

## 2.3 State Space Formulation

For sheet metal bending, one can develop many different types of state space formulations. We will use the forward chaining formulation for explaining the scheme described in this paper. The search space is represented as a tree. Each node in the search tree represents an intermediate workpiece and the associated setup plan. Each edge in the search tree represents a bending operation, described by a bend line and a holding face (i.e., the face that is used to hold the part). The root node of the search tree is the flat blank with an empty setup. Leaf nodes of the search tree are either nodes corresponding to the final part, or nodes corresponding to the intermediate workpiece shapes which have infeasible pressbrake setups. Leaf nodes that do not correspond to the final part are called *blocked nodes*. New search nodes cannot be generated from blocked nodes. Figure 6 shows a portion of the state space for the part shown in Figure 2.

Many different types of search techniques can be used on this state space formulation to find the desired solution. Readers are referred to Nilsson's book [15] for background information on search techniques. In our research, we use  $A^*$  search algorithm. The main

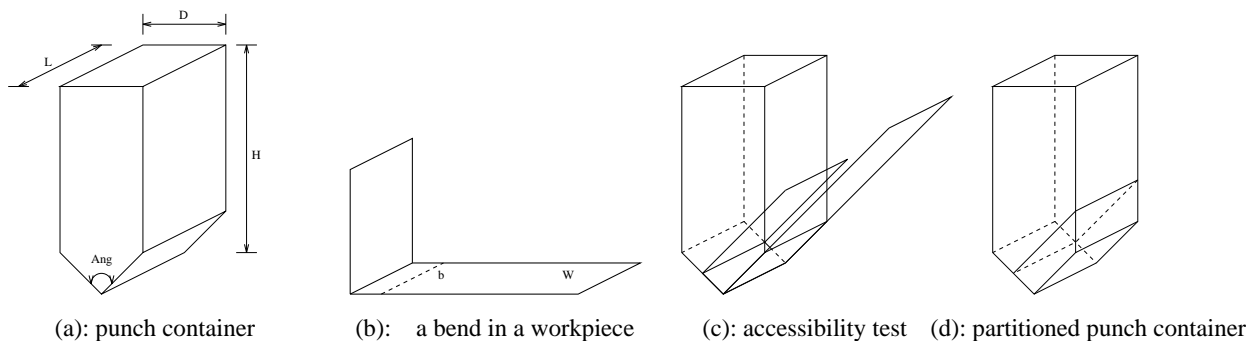


Figure 7: Bend line accessibility.

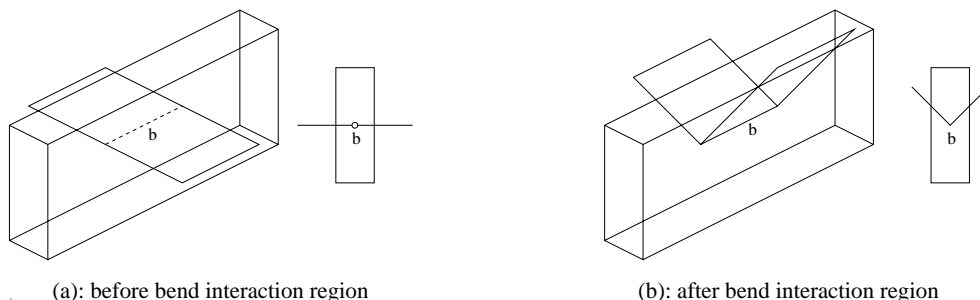


Figure 8: Interaction regions.

emphasis of this paper is speeding up the node generation and not the underlying search procedure. The speed up in node generation steps can be used to improve the computational efficiency of a wide variety of search procedures. Summary of our previous work on operations planning can be found in [8, 9].

## 3 Definitions

### 3.1 Inaccessibility

A bend line may not be accessible in every possible operation sequence. In fact, depending upon the intermediate workpiece geometry, a bend line might be inaccessible for every available punch. In order to eliminate infeasible operation sequences, we need to find bend lines that are inaccessible in a given workpiece for every available punch. Inaccessibility of a bend line  $b$  in a workpiece  $W$  is examined in the following manner:

1. Construct the after-bend model  $P$  by bending  $b$  in  $W$ . (For a given bend line, we can construct two different types of workpiece models: before-bend models, representing the workpiece shape before bending; and after-bend models, representing the workpiece shape after bending.)

2. Construct the universal punch container  $C_p$ . The universal punch container is a parametrized geometric shape capable of containing every available punch. For air bending operations, the universal punch container can be defined by parameters  $H$ ,  $D$ ,  $L$ , and  $Ang$ . An example is shown in Figure 7(a). Parameter  $L$  is set to the bend length. Parameter  $Ang$  is set to the bend angle. Parameters  $H$  and  $W$  are initialized based on the dimensions of the available punches. Parameters  $H$  and  $W$  are selected such that the universal punch container has the minimum possible volume and is capable of containing every available punch of length  $L$ .
3. Subtract  $P$  from  $C_p$ . If the subtraction process splits  $C_p$  into two or more solids, then the bend line  $b$  is *inaccessible* in the workpiece  $W$ . The reason for this is as follows. If the intermediate workpiece shape splits the punch container into two parts, then there exists no punch geometry that can fit in the punch container and connect the top face with the bottom tip without interfering with the workpiece geometry. For example, in Figure 7(c), the bend line  $b$  is inaccessible in the workpiece  $W$ .

### 3.2 Precedence Constraints

Due to accessibility requirements and preferred manufacturing practices certain bending operations need to be done before or after certain other operations. Precedence constraints resulting from accessibility requirements can be generated by pair-wise examination of various bend lines. For example, if performing bend  $b$  before bend  $b'$  makes  $b'$  inaccessible, then  $b'$  needs to be done before  $b$ . For example for the part shown in Figure 2, accessibility requirements introduce the following set of precedence constraints:  $\{(b1 \text{ before } b2)\}$ .

### 3.3 Interaction

Only a portion of the workpiece geometry which is close to the bend line affects the bending operation (i.e., accessibility and setup constraints). Based on the dimensions of the available tools, we can define *interaction regions* which characterize the portions of the workpiece that can affect the bending operation. The interaction regions should be large enough to contain every available tool. Figure 8 shows the interaction regions of a bend line.

In order to eliminate unnecessary computation, for every bend  $b$  we want to identify the set of bends that will not affect bending of  $b$ . Intuitively, a bend  $b'$  will not affect the bend  $b$ , if bending  $b'$  will never result in an intermediate workpiece shape having a face in the interaction regions for bend  $b$ . Interaction information is represented in a directed graph called an *interaction graph*. Nodes in the interaction graph represent bends. Edges in the interaction graph represent interactions. There is an edge in the graph from node  $b$  to  $b'$ , if we cannot eliminate the possibility of  $b'$  affecting the interaction regions for  $b$ . The interaction graph is built in the following manner:

1. Initialize the interaction graph by adding a node for every bend.

2. For every bend  $b$ , do the following:

- (a) Construct the after-bend and before-bend interaction regions for  $b$ .
- (b) Find set of faces  $F_N$  that cannot reach the interaction regions of  $b$ . How to find  $F_N$  is described below.
- (c) Let  $F$  be the set of all faces in the part. For every face  $f \in F - F_N$ , do the following:
  - i. Find the set of intermediate bends  $B_f$  that connect face  $f$  to bend  $b$ .
  - ii. For every bend  $b' \in B_f$ , add an edge  $(b, b')$  in the interaction graph (i.e., the possibility of  $b'$  affecting interaction regions of  $b$  cannot be eliminated).

The set of faces  $F_N$  for bend  $b$  in Step 2(b) is computed by analyzing every face  $f$  of the part in the intermediate workpiece geometry that brings the face  $f$  closest to the interaction region of bend  $b$ . Intuitively,  $F_N$  represents the set of faces that do not enter the interaction regions of  $b$  under the conservatively estimated worst case workpiece shape. The closest position of a face  $f$  with respect to an interaction region is estimated in the following manner.

- Find the sequence of bends  $B_{b,f}$  which connect the face  $f$  to the bend  $b$ . If  $B_{b,f}$  is empty, then stop (i.e.,  $f$  is directly connected to  $b$ , and it cannot be included in  $F_N$ ).
- Find the first bend  $b'$  in the sequence  $B_{b,f}$  which is not perpendicular to  $b$  and can be done before  $b$  as per precedence constraints. If no such bend exists, than intersect before-bend and after-bend models for bend  $b$  with the respective interaction regions. If  $f$  does not intersect with interaction regions, then include  $f$  in the set  $F_N$  and stop.
- If  $b'$  exist, than find the minimum distance  $d_{b'}$  of  $b'$  from the interaction region of  $b$ .
- Let  $F_{b',f}$  be the set of faces that connect the bend  $b'$  to the face  $f$ . Find the maximum distance  $d_m$  between any two points lying on the faces in the face set  $F_{b',f} \cup f$ .
- If  $d_m$  is greater than  $d_{b'}$ , then  $f$  will never enter the interaction region. Therefore, include  $f$  in the set  $F_N$  and stop.

For bend  $b_2$  (see Figure 2),  $F_N$  is  $\{f_4, f_5, f_6, f_7, f_8, f_9\}$ . Figure 9 shows the complete interaction graph for this part. The edge from node  $b_1$  to node  $b_2$  implies that the possibility of  $b_2$  affecting the tooling and setup parameters for  $b_1$  cannot be ruled out. There is no edge from  $b_1$  to  $b_3$ . This implies that  $b_3$  cannot affect the tooling and setup parameters for  $b_1$ .

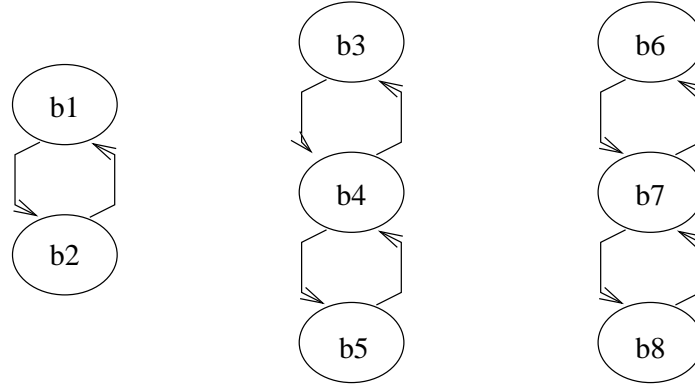


Figure 9: Interaction graph for the part shown in Figure 2.

## 4 Decomposing Node Generation Problem

A new node is generated by taking an already generated node in the search tree and applying a search operator to it. We will refer to the node from which the new node is generated as the starting node and the node which is being generated as the target node. For example, if the starting node is the root node (i.e., node corresponding to the flat blank), we can bend any of the accessible bend lines to generate a new node which corresponds to an intermediate workpiece with one bend. The node generation step can be decomposed into a number of subproblems described in the following subsections. In order to generate a new search node, we need to solve each of these subproblems.

### 4.1 Subproblem 1: Feasibility of Target Nodes

In this step, we try to examine whether the target node (i.e., node being generated) is a feasible state or not. Any state that has no feasible operation sequence associated with it is considered an infeasible state. In order to be feasible, a state should have at least one feasible bending operation, or it should be a state corresponding to the final part. For the given target node, we first compute the *inaccessible bend set* by analyzing accessibility of various unfinished bends in the intermediate workpiece. Then, we compute the *feasible bend set* by subtracting the inaccessible bend set from the set of unfinished bends. Input and output information for this subproblem is given below:

*Input:* A starting node and a search operator.

*Output:* A set of bending operations which will be inaccessible in the target state (i.e., inaccessible bend set) and a set of bending operations which will be feasible in the target state (i.e., feasible bend set).

## 4.2 Subproblem 2: Setup Constraint Generation

As described in the background section, various bending operations impose constraints on tooling stage lengths. In order to do setup planning, we need to compute setup constraints resulting from various bending operations. Setup constraints are generated by analyzing any potential interference problem between the geometric models of the tool and the intermediate workpiece. Figure 10 shows examples of setup constraint generation for the part shown in Figure 2. For example, consider the first bending operation in the operation sequence shown in Figure 10(a). In this operation, bend  $b8$  is created by holding the workpiece at face  $f1$ . The minimum tooling stage length for this operation is  $l2$  (see Figures 2 and 10). There is no restriction on the maximum tooling stage length for this bend. Now, consider the third bending operation in the same operation sequence. In this operation, bend  $b7$  is created by holding the workpiece at face  $f1$ . The minimum tooling stage length for this operation is  $l3$ . In order to avoid interference between the tool and the intermediate workpiece, the maximum allowed tooling stage length is  $l3$  (see Figures 2 and 10). Therefore, the operation sequence [(bend  $b8$ , hold  $f1$ )(bend  $b6$ , hold  $f1$ )(bend  $b7$ , hold  $f1$ )] (shown in Figure 10(a)) results in the following constraints.

Operation (bend  $b8$ , hold  $f1$ ): Tooling stage length needs to be between  $l2$  and  $\infty$ .

Operation (bend  $b6$ , hold  $f1$ ): Tooling stage length needs to be between  $l2$  and  $\infty$ .

Operation (bend  $b7$ , hold  $f1$ ): Tooling stage length needs to be  $l3$ .

Similarly, the operation sequence [(bend  $b7$ , hold  $f1$ )(bend  $b8$ , hold  $f1$ )(bend  $b6$ , hold  $f1$ )] (shown in Figure 10(b)) results in the following constraints.

Operation (bend  $b7$ , hold  $f1$ ): Tooling stage length needs to be between  $l3$  and  $\infty$ .

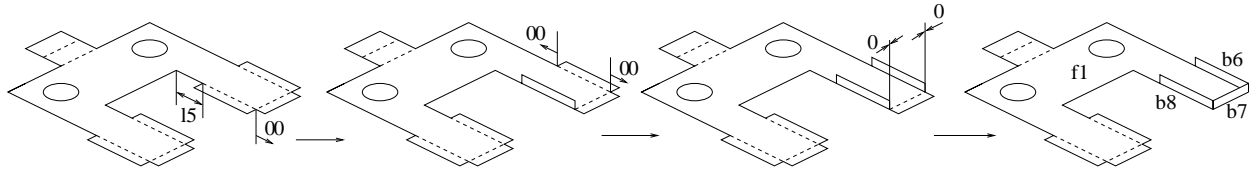
Operation (bend  $b8$ , hold  $f1$ ): Tooling stage length needs to be between  $l2$  and  $l2 + l5$ .

Operation (bend  $b6$ , hold  $f1$ ): Tooling stage length needs to be between  $l2$  and  $\infty$ .

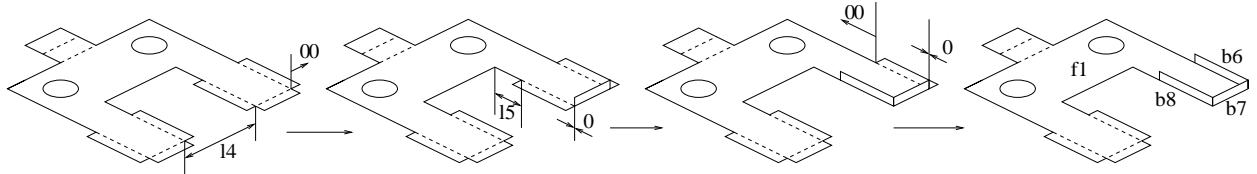
Setup constraints for the new node are generated by adding the setup constraints for the current bending operation to the set of setup constraints for the starting node. Input and output information for this subproblem is given below:

*Input:* A starting node and a search operator.

*Output:* A set of setup constraints for the target node.

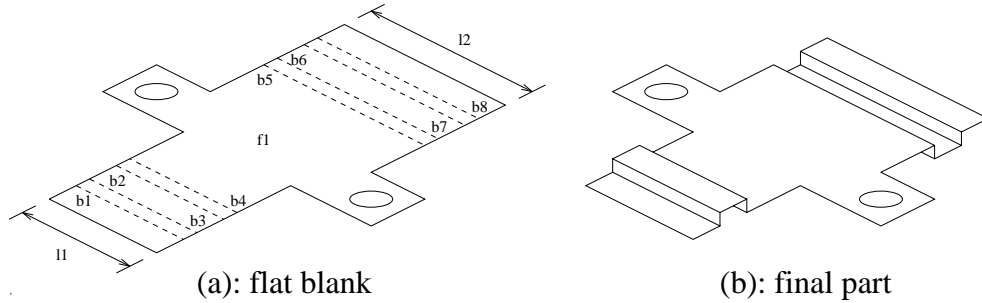


(a): SEQUENCE [(bend b8, hold f1)(bend b6, holdf1)(bend b7, hold f1)]



(b): SEQUENCE [(bend b7, hold f1)(bend b8, holdf1)(bend b6, hold f1)]

Figure 10: Generation of setup constraints for the part shown in Figure 2.



(a): flat blank

(b): final part

Figure 11: An example of constraint subsumption.

### 4.3 Subproblem 3: Setup Planning

Once we have computed the setup constraints for a partial or a complete bend sequence, we can perform the setup planning. Setup planning is done by a constraint propagation algorithm. The objective of setup planning is to create a pressbrake setup which requires the minimum production time. In most cases, we are interested in creating setups that involve the minimum number of tooling stages which fit on the die rail of the pressbrake. Detailed description of our setup planning algorithm is described in [8, 9].

If  $l2 > l3$  (see Figures 2 and 10), then bends  $b6$ ,  $b7$ , and  $b8$  cannot be done on same tooling stage in the operation sequence shown in Figure 10(a). On the other hand, bends  $b6$ ,  $b7$  and  $b8$  can be done on the same tooling stage in the operation sequence shown in Figure 10(b). This example shows that the number of stages in a press-brake setup depends on the operation sequence.

In some operation sequences, some constraints are subsumed by other constraints. In such

cases, we only need to deal with more restrictive constraints. Consider the part shown in Figure 11. Every feasible bend sequence for this part results in the following stage constraints.

Operations  $b_1$ ,  $b_2$ ,  $b_3$ , and  $b_4$ : Tooling stage length needs to be between  $l_1$  and  $\infty$ .

Operations  $b_5$ ,  $b_6$ ,  $b_7$ , and  $b_8$ : Tooling stage length needs to be between  $l_2$  and  $\infty$ .

If  $l_2 > l_1$ , then any tooling state which satisfies the second constraint will automatically satisfy the first constraint. Therefore, we can assign all operations that result in the first constraint to the tooling selected for the second constraint. In such cases, we can significantly reduce the size of the constraint set by performing constraint subsumption checks.

*Input:* A set of setup constraints associated with the node being generated.

*Output:* A pressbrake setup and a setup plan which assigns operations to various tooling stages in the setup.

## 5 Virtual Node Generation Scheme

Subproblems 1-3, described in the previous section require significant computation time. In this section, we describe how the results of previously solved subproblems can be reused to eliminate unnecessary computations.

### 5.1 Reusing Results of Subproblem 1

The output of this subproblem only depends on the intermediate workpiece shape of the node being generated. Therefore, any two subproblems which have the same intermediate workpiece shape will have the same result. Any two operation sequences which cover the same bends will lead to the same intermediate part shape. For example, bend sequences [(bend  $b_1$ , hold  $f_1$ ), (bend  $b_2$ , hold  $f_3$ ), (bend  $b_3$ , hold  $f_5$ )] and [(bend  $b_3$ , hold  $f_6$ ), (bend  $b_2$ , hold  $f_4$ ), (bend  $b_1$ , hold  $f_2$ )] will result in the same intermediate part shape. This observation can be used to develop a scheme for reusing the results of previously solved subproblems.

We can further improve the condition for reusing subproblem results by making the following observation. When we apply a search operator to a node, various choices for the search operator correspond to the bends in the feasible bend set of the starting node. If the bend corresponding to the search operator does not interact with any other bends in the inaccessible and feasible bend sets, then the feasible bend set for the new node can be simply computed by removing the current bend from the feasible bend set of the parent node. The inaccessible bend set for the new node is same as the inaccessible bend set for the parent node. We use the following scheme for solving these subproblems.

1. Compute the bend set for the node being generated. This is done by adding the bend in the current operation to the bend set for the parent node.

2. If any of the already solved subproblems has an identical bend set, then reuse the result of that subproblem.
3. If the bend in the current operation does not interact with any of the bends in the inaccessible and feasible bend sets of the parent node, then the inaccessible bend set for the new node is the same as the inaccessible bend set of the parent node; and the feasible bend set for the new state can be generated by removing the bend in the current operation from the feasible bend set for the parent node.
4. If the previous two steps do not allow reuse of results, then solve this subproblem.

Consider the part shown in Figure 2. Now let us consider the search node corresponding to the following partial sequence [(bend  $b_6$ , hold  $f_1$ ), (bend  $b_5$ , hold  $f_1$ ) (bend  $b_1$ , hold  $f_2$ )]. The feasible bend set for this search node is  $\{b_2, b_3, b_4, b_7, b_8\}$ . The inaccessible bend set for this search node is empty. Now suppose that we want to generate a new search node corresponding to the operation sequence [(bend  $b_6$ , hold  $f_1$ ), (bend  $b_5$ , hold  $f_1$ ) (bend  $b_1$ , hold  $f_2$ ), (bend  $b_2$ , hold  $f_1$ )]. As operation  $b_2$  does not interact with any bends in the inaccessible and feasible bend sets of the parent node, we can create inaccessible and feasible bend sets for this subproblem by simply modifying its parent’s inaccessible and feasible bend sets. The feasible bend set for this new search node will be  $\{b_3, b_4, b_7, b_8\}$ . The inaccessible bend set for this new search node will be empty.

## 5.2 Reusing Results of Subproblem 2

Setup constraints for a bending operation depend on the orientation of intermediate workpiece geometry. For example, operation (bend  $b_3$ , hold  $f_5$ ) in sequences [(bend  $b_1$ , hold  $f_1$ ), (bend  $b_2$ , hold  $f_3$ ), (bend  $b_3$ , hold  $f_5$ )] and [(bend  $b_2$ , hold  $f_4$ ), (bend  $b_1$ , hold  $f_2$ ), (bend  $b_3$ , hold  $f_5$ )] results in identical setup constraints. We can make use of this observation in reusing results of previously solved subproblems.

For any bending operation, setup constraints are only affected by a small portion of the workpiece that falls under the interaction regions of the bend. Any bend that does not interact with the current bend will not affect its setup constraints. Two subproblems that have the same bending operations and the same workpiece geometry in the interaction region will have the same setup constraints. We use the following scheme for solving these subproblems:

1. Compute the bend set for the node being generated.
2. Find the set of bends in the bend set that do not interact with the bend in current operation. Compute the interacting bend set by subtracting the non-interacting bends from the bend set. If a subproblem with the same operation and the same interacting bend set has already been solved, then reuse the setup constraints for the matching operation from that subproblem. Find the setup constraints for the new node by appending the setup constraints for the current operation to the parent node.

3. If the previous step does not allow reuse of results, then solve this subproblem.

For example, consider the part shown in Figure 2. For this part, the operation (bend  $b6$ , hold  $f1$ ) has identical stage constraints in sequences [(bend  $b6$ , hold  $f1$ )] and [(bend  $b1$ , hold  $f2$ ), (bend  $b2$ , hold  $f1$ ), (bend  $b6$ , hold  $f1$ )]. Bends  $b2$  and  $b1$  do not interact with  $b6$ . Therefore, setup constraints for operation (bend  $b4$ , hold  $f1$ ) remain same in both subproblems. If we have solved either of the two subproblems, then we can reuse the results in the other subproblem.

### 5.3 Reusing Results of Subproblem 3

In this subproblem, a set of setup constraints are used to create a setup for the search node. Any two subproblems that have the same constraints will have an identical solution. We compare the setup constraints of a given subproblem with the already solved subproblems. If the constraints for the current subproblem match with any of the already solved subproblems, then we can reuse the results. We can use the following scheme for solving these subproblems:

1. Remove redundant constraints from the constraint set.
2. Compare current constraints with constraints of already solved subproblems. If a match is found, then reuse the setup of the matching subproblem. Assign various operations to the compatible stages in the setup.
3. If no match is found in the previous step, then solve this subproblem.

Consider the part shown in Figure 11. The operation sequence [(bend  $b1$ , hold  $f1$ ), (bend  $b8$ , hold  $f1$ )] results in the following setup constraints:

Operations  $b1$ : Tooling stage length needs to be between  $l1$  and  $\infty$ .

Operations  $b8$ : Tooling stage length needs to be between  $l2$  and  $\infty$ .

The second constraint subsumes the first constraint. Therefore, the setup constraints for this sequence is same as the setup constraint for the operation sequence [(bend  $b8$ , hold  $f1$ )]. Both of these subproblems require an identical one stage solution.

## 6 Improvements in Computational Efficiency

This section describes how reusing results of already solved subproblems affects the computation efficiency.

## 6.1 Size of Search Tree

For the state space formulation presented in the background section, the number of nodes in the complete search tree (in absence of any blocked node) are:

$$\sum_{i=1}^n 2^i n! / (n - i)!$$

where  $n$  is the number of total number of bending operations. Presence of blocked nodes reduces the number of nodes in the search tree.

If we were to explore the full search tree without any virtual node generation scheme, in the worst case, we will need to solve each subproblem for every node in the search tree. For the part shown in Figure 2 there are more than  $10^8$  nodes in the complete search tree.

## 6.2 Analysis of Subproblem 1

There are  $2^n$  number of distinct intermediate part shapes. Therefore  $2^n$  is the maximum number of subproblems that need to be solved. Moreover, any node whose current operation's interacting bends have been already done can reuse the results of its parent node. Therefore, depending upon the size of interaction graph, the maximum number of distinct subproblems is further reduced. The upper bound on the maximum number of this type of subproblems for the part shown in Figure 2 is 256.

## 6.3 Analysis of Subproblem 2

This subproblem needs to be solved for every combination of bends and the intermediate workpiece geometries that affects the interaction regions of that bend. The maximum number of subproblems to be solved is:

$$\sum_{i=1}^n 2^{I_i+1}$$

where,  $I_i$  is the number of edges from the  $i^{th}$  bend in the interaction graph.

As clear from the above formula, the number of distinct subproblems that needs to be solved depends on the number of edges in the interaction graph. The smaller the number of edges, the lesser the number of distinct subproblems. For example, consider the part shown in Figure 2. The maximum number of this type of subproblems for this part is:

$$2^{1+1} + 2^{1+1} + 2^{1+1} + 2^{1+2} + 2^{1+1} + 2^{1+1} + 2^{1+2} + 2^{1+1} = 40$$

## 6.4 Analysis of Subproblem 3

In a typical part, multiple bend lines have the same bend length. In such cases, for setup planning, we can use the most restrictive stage constraints for the bend length. The maximum number of this type subproblems depends on the nature of setup constraints for a

particular problem. However, we can derive an upper bound on the maximum number of this type of subproblems.

Let  $n$  be the number of bend lines and  $m$  be the number of distinct bend lengths. Let  $n_i$  be the number of bends in the  $i^{\text{th}}$  bend length. The maximum number of subproblems to be solved is bounded by the following number:

$$\prod_{i=1}^m \sum_{j=1}^{n_i} 2^{I_{j,i}+1}$$

where,  $I_{j,i}$  is the number of edges in the interaction graph from the node corresponding to the  $j^{\text{th}}$  bend in the  $i^{\text{th}}$  bend length. The above formula produces close upper bounds when  $m$  is much smaller than  $n$ , and the number of edges in the interaction graph are of the order of  $n$ . For example, consider the part shown in Figure 2. There are three distinct bend lengths for this part. Bends  $b_1$  and  $b_2$  have length  $l_1$ . Bends  $b_3$ ,  $b_5$ ,  $b_6$ , and  $b_8$  have length  $l_2$ . Bends  $b_4$  and  $b_7$  have length  $l_3$ . The upper bound on the maximum number of this type of subproblems for this part is:

$$(2^{1+1} + 2^{1+1}) \times (2^{1+1} + 2^{1+1} + 2^{1+1} + 2^{1+1}) \times (2^{1+2} + 2^{1+2}) = 2048$$

## 6.5 Efficiency Gained by Virtual Node Generation

Finding an already solved matching subproblem also requires some computation. So, we would like to find out the overall efficiency gained by reusing old results rather than computing them from the scratch. Let  $\alpha$  be the ratio of subproblem matching time to actual computation time. Let  $\beta$  be the fraction of subproblems that can be reused. Let  $T_v$  be the computation time with virtual node generation, i.e., reusing results of old subproblems. Let  $T$  be the computation time without virtual node generation.

$$T_v/T = 1 - \beta(1 - \alpha)$$

As can be seen by this formula that if  $\beta(1 - \alpha)$  is expected to be much higher than zero, then one should try to reuse old results.

Consider the case of subproblem 2 for the part shown in Figure 2.  $\alpha$  is approximately 0.1 for this subproblem.  $\beta$  is close to one due to very small number of subproblems that need to be solved for this part. So we get a speed up by a factor of 10 for this case.

In our experiments  $\alpha$  ranges from 0.05 to 0.25 for different types of subproblems and parts. Complex part geometry leads to smaller value of  $\alpha$  due to the following reason. Most of the geometric reasoning algorithms are  $O(n^2)$  in nature, where  $n$  is the number of edges in the part. On the other hand, caching procedures do not directly use part geometry. So their performance does not get affected by the complexity of part geometry. The speed up factor varies from 20 to 2 depending upon a particular part ( $\beta$  depends on the part).

## 6.6 When to Start Virtual Node Generation?

Examining whether a given subproblem has already been solved requires computation time. In the beginning phase of the search process when there are very few nodes, there is very little probability of finding a matching subproblem. Therefore, we want to trigger virtual node generation scheme only when sufficient number of nodes have been generated and the probability of reusing results becomes reasonably high. In practice, we wait for a fixed number of nodes to be generated before triggering the virtual node generation scheme.

## 7 Conclusions

Manufacturing planning problems typically require computation-intensive node generation. In order to solve practical problems, we need some way to speed up the node generation process. Making reuse of previously performed computations is one way to achieve this. However, if we look at the node generation problem as a single monolithic problem, there is very little opportunity to reuse any of the already performed computations. Decomposing node generation problem into a number of simpler subproblems presents us with an opportunity to develop reuse conditions for similar subproblems.

In this paper, we present a virtual node generation scheme for sheet metal parts. We decompose the node generation problem into a number of subproblems. Analysis of various subproblems reveals that reuse conditions can be developed for each of the subproblems. Whenever we need to solve a node generation subproblem, we first try to find out if we can reuse results of any of the already solved subproblem. If reuse is possible, then we solve the subproblem virtually by reusing results and save computation time. Furthermore, we show that in case of each subproblem, the number of distinct subproblems are strongly related to the number of interactions among bending operations. We also provide a summary of the expected efficiency gains.

It should be noted that the virtual node generation scheme achieves speed up in node generation by trading computation time with memory. Therefore, this technique is useful when one needs to solve the problem quickly and can provide sufficient memory to save results of previously solved subproblems.

The scheme presented in this paper increases the node generation capability and speeds up the search process, allowing us to consider more possibilities and solve more complex operation planning problems. We believe that the concept presented in this paper is quite general and can be applied to other process planning domains. Specifically, process planning of manufacturing processes that create products by a sequence of discrete operations such as machining and assembly can also benefit from virtual node generation.

## Acknowledgements

This research has been supported by the NSF grant DMI9896255.

## References

- [1] L. Alting and H. Zhang. Computer aided process planning: The state of the art survey. *Int. J. of Prod. Res.*, 27(4):553–585, 1989.
- [2] Amada Sheet Metal Working Research Association. *Bending Technique*. Machinist Publishing Company Limited, first edition, 1981.
- [3] Steve D. Benson. *Press Brake Technology: A Guide to precision sheet metal bending*. Society of Manufacturing Engineers, 1997.
- [4] Tien-Chien Chang. *Expert Process Planning for Manufacturing*. Addison-Wesley Publishing Co., 1990.
- [5] L. Cser, M. Geiger, W. Greska, and M. Hoffman. Three kinds of case-based learning in sheet metal manufacturing. *Computers in Industry*, 17:195–206, 1991.
- [6] L.J. de Vin, J. de Vries, A.H. Streppel, and H.J.J. Kals. PART-S: a CAPP system for small batch manufacturing of sheet metal components. In *Proceedings of the 24th CIRP International Seminar on Manufacturing Systems*, pages 171–182, Copenhagen, 1992.
- [7] L.J. de Vin, J. de Vries, A.H. Streppel, E.J.W. Klaassen, and H.J.J. Kals. The generation of bending sequences in a CAPP system for sheet metal components. *Journal of materials processing technology*, 41:331–339, 1994.
- [8] S.K. Gupta and D.A. Bourne. Multi-part setup planning for sheet metal bending operations. In *ASME Computers in Engineering Conference*, Sacramento, CA, September 1997
- [9] S.K. Gupta, D.A. Bourne, and K.H. Kim, and S.S. Krishnan. Automated process planning for sheet metal bending operations. In *Journal of Manufacturing Systems*, 17(5):338–336, 1998.
- [10] S.K. Gupta, D.S. Nau, W.C. Regli, and G. Zhang. A methodology for systematic generation and evaluation of alternative operation plans. In Jami J. Shah, Martti Mantyla, and Dana S. Nau, editors, *Advances in Feature Based Manufacturing*, pages 161–184. Elsevier Science Publishers, 1994.
- [11] I. Ham and S. Lu. Computer aided process planning, the present and the future. *Annals of CIRP*, 37(2), 1988.
- [12] C. C. Hayes and P. Wright. Automatic process planning: using feature interaction to guide search. *Journal of Manufacturing Systems*, 8(1):1–15, 1989.
- [13] Caroline Hayes. A manufacturing process planner for a concurrent engineering environment. In *IEEE International Symposium on Assembly and Task Planning*, 1995.

- [14] D. S. Nau. Automated process planning using hierarchical abstraction. *TI Technical Journal*, pages 39–46, Winter 1987.
- [15] Nils Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann Publishers Incorporated, CA, 1980.
- [16] B.O. Nnaji, T.S. Kang, S.C. Yeh, and J.P. Chen. Feature reasoning for sheet metal components. *International Journal of Production Research*, 29(9), 1991.
- [17] B. Radin, M. Shpitalni, and I. Hartman. Two stage algorithm for rapid determination of the bending sequence in sheet metal products. In *ASME Design Automation Conference*, Irvine, CA 1996.
- [18] J.S. Smith, P.H. Cohen, J.W. Davis, and S.A. Irani. 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.
- [19] D.N. Sormaz and B. Khoshnevis. Process sequencing and process clustering in process planning using state space search. *Journal of Intelligent Manufacturing*, 7:189–200, 1996.
- [20] H.P. Wang and R.A. Wysk. A knowledge-based approach for automated process planning. *International Journal of Production Research*, 26(6):999–1014, 1988.
- [21] C. Wick, J.T. Benedict, and R.F. Veilleux, editors. *Forming*, volume 2 of *Tool and Manufacturing Engineers Handbook*. Society of Manufacturing Engineers, fourth edition, 1983.
- [22] G. Yut and T.C. Chang. A study of automated process planning for sheet metal products. In *NSF Design and Manufacturing Systems Conference*, January 1993.