Automatic Determination of Bending Sequence in Sheet Metal Products

M. Shpitalni (1), D. Saddan, Faculty of Mechanical Engineering, Technion, Haifa, Israel
Received on January 14, 1994

SUMMARY:

The bending stage is the most complex and critical stage in manufacturing sheet metal products. Moreover, the automatic determination of the bending sequence is essential within the framework of process planning. Basic examination of the problem immediately reveals that the determination of the bending sequence is a combinatorial problem. Furthermore, it is coupled with the problem of selection of the bending tools. In this paper, the problem of automatic tool selection and bending sequence determination is formulated as a graph search problem. Heuristics, reduction techniques and search methods based on the A-star search algorithm were used to accelerate the solution. The results are superior to previously published results.

KEYWORDS: CAPP, Sheetmetal, Bending.

Introduction

Automatic process planning is becoming more and more important as economic competition among manufacturers increases. Indeed, the need to manufacture small-sized batches of varied and more complex products is the driving force behind the development of flexible systems. Production of small-sized batches is characterized by the high cost per product associated with process planning and with programming of numerically controlled machines and industrial robots.

In most cases, process planning involves non-polynomial (exponential or combinatorial) problems which are practically impossible to solve optimally within a reasonable period of time. Therefore, sub-optimal solutions are searched for which can be obtained within an accepted period of time and can provide reasonable and practical results. One such example is determination of assembly order.

The bending stage is the most complex and critical stage in manufacturing sheet metal products. It is often this stage which produces a bottle neck in the whole process. In recent years, work in the area of automatic bending of sheet metal products has been accelerated with the introduction of bending robots (e.g., Amada's BM-100) and by the application of computer integrated manufacturing (CIM) and flexible manufacturing systems (FMS). Within this framework, the need for automatic determination of the bending sequence is essential. Basic examination of the problem immediately reveals that not only is determination of the bending sequence a combinatorial problem, but it is also coupled with the problem of selection of the bending tools.

The importance of the bending sequence problem has been recognized, and studies on the subject have been carried out [1-10]. Since the problem is combinatorial in nature, classical approaches, that is, expert system methods and heuristic-based graph search techniques, must be used to deal with the problem effectively.

This paper is concerned with strategies and techniques for automatic bending tool selection and bending sequence generation. The problem has been formulated as a graph search problem. Techniques for reducing the size of the problem, reducing the search domain by using heuristic rules, and iterating solutions were employed to accelerate the solution. The results of this study were implemented in an application package called BEND. The system can currently handle products with up to 16-18 bend lines.

The results are superior to previously published results which attain solutions (in reasonable time) for products with 8-12 bend lines. Furthermore, the proposed system considers factors which previously were not always taken into account, such as tool selection, handling, stability and number of tool changes.

In the following sections, the problem of tool selection and bend sequence determination is described and formulated. Next, the approach taken, including the heuristic rules, is discussed. Finally, examples are provided to demonstrate the capabilities of the proposed method, and a partial analysis of the results is presented.

The Problem

To produce a sheet metal product, a flat layout must be cut first. This layout, after being bent, results in the desired 3-D product. To bend this layout, a set of bending tools (punches and dies) is given. The punches (refer to Fig. 1) differ in their tips (radii and angles), shapes, lengths and holders. Each punch must also have a matching die, whose existence can be assumed.

Once the set of punches is given, the bending sequence must be determined and a matching punch must then be selected for every bend operation. The sequence is crucial because in many cases it determines whether a product can or cannot be produced. A feasible bending sequence is one in which no collisions (product-product, product-machine or product-punch) occur. In Fig. 2, two simple bending sequences of a simple profile are shown. The upper sequence ($2\rightarrow1\rightarrow3$) is a legal (feasible) one, while the lower one ($1\rightarrow3\rightarrow2$) is illegal and causes the product to collide with itself. Fig. 3 not only shows that many tools can be used to perform a required bend; it also raises another problem, the coupling between the tool selected and the bend to be performed. As illustrated in the figure, a certain bend may be feasible or not depending on the tool, implying that the two problems of tool assignment and bending sequence are coupled. Consequently, the complexity of the problem is increased significantly.

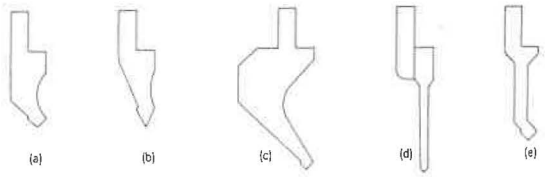

Fig. 1: Set of punches [7].

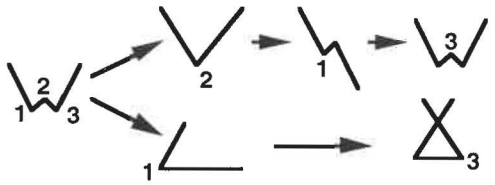

Fig. 2: Two simple bending sequences for a simple profile.

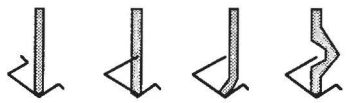

Fig. 3: Coupling of tool assignment and bending sequence.

Furthermore, most cases have many legal bending sequences and several legal punches. The selected bending sequence and the tools assigned must be reasonable (optimal or at least close to optimal) in terms of minimum time, minimum cost, minimum labor, etc.

Thus, the problem can be formulated descriptively as follows:

> *Given are a layout and a set of bending tools. A tool must be assigned to each bend line and the bending sequence must be determined such that:*
> *(a) The total bending operation is feasible and,*
> *(b) The bending is reasonable under given criteria.*

Using the following notation:

* A state in the bending process is defined by the pair S[Tools(p,d), Bends(b_1,...,b_i,...,b_n)].
* Tools(p,d) is a pair consisting of a punch p and a die d used to bend the current bend line (i.e., to transform the product from the previous state to the current state).
* Bends(b_1,...,b_i,...,b_n) describes the product's current bending state. If b_i=T the i-th bend line is bent; otherwise, b_i=F and the i-th bend line is not yet bent.
* A legal state is a state in which there is no collision, either product-product or product-tool or product-machine.

The problem can now be formulated as:

Given an initial state (layout): S[Tools(-,-), Bends(F,F,...,F)], and the final desired state describing the product S[Tools(,), Bends(T,T,...,T)]. Find an optimal (or close to optimal) sequence of legal states from the initial state to the final state.

A naive algorithm can immediately be proposed: generate all the possible bending sequences and check each one of them. Ignoring the tools, there are potentially n! such sequences (where n is the number of bend lines). In practice, there is usually more than one applicable tool pair, so the actual number is much greater. Assuming t applicable tools (on the average) for each bend line, the complexity of the problem is in the order of $O(t^n n!)$. That is, if an optimal solution is sought, an average of $t^n n!$ possibilities must be created and tested. For each of these possibilities, potential collisions should be examined and all sequences and tool combinations for which a collision is not detected must be evaluated for optimality. (Note: the issue of collision detection itself during bending is a serious computational problem. A special algorithm dedicated to the application of bending has been developed, but will not be discussed in this paper.) To understand the degree of complexity, assume t=5; then, for n=8 there are about $1.5.10^{10}$ possibilities and for n=10 the number increases to $3.5.10^{13}$ and for n=12 to $\sim 10^{17}$.

The dependency and the sensitivity of the problem to the number of bend lines can now be appreciated. Furthermore, the complete tree representing all possible states cannot be realized on a computer disk. For common (not complex and not extremely simple) products with n=8-12, each additional bend line increases the number of possibilities and the required calculations by at least one order of magnitude. It is obvious that the naive approach is not appropriate, and techniques to reduce the size of the problem and the search domain, such as heuristics, should be applied.

The Proposed Approach

The generation of a bending sequence naturally lends itself to a search problem formulation. Each state S[Tools, Bends] is represented by a node in a graph. The root of the graph S[Tools(-,-), Bends (F,F,..,F)] represents the initial state (flat pattern layout). Internal nodes, characterized by having at least one F value in the Bends set, represent intermediate states. Terminal nodes (leaves) are either goal nodes (Bend={T,T,...,T}) or "dead end" nodes, that is, no further bends can or should be performed, either due to collision and lack of needed punch or too high a cost. Every solution for the bending sequence is a directed path from the root node to a goal node. For simplicity, the graph is converted into a tree by duplicating nodes. A very simple tree, demonstrating illegal path, paths requiring tool changes, and an optimal path, is depicted in Fig. 4.

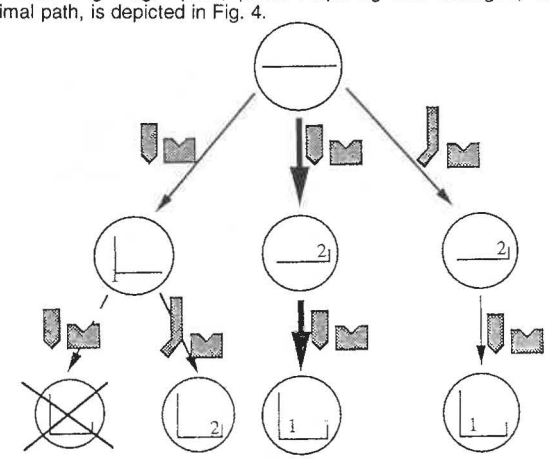

Fig. 4: Simple tree depicting illegal path, paths requiring tool changes, and an optimal path.

Due to the size of the problem, measures must be taken to control the size of the tree and to limit the development (open) of nodes. The development of nodes is selective and is governed by heuristics and constraints. For this reason, the A* search algorithm [11] has been selected, as will be discussed later.

The size of the tree is reduced by reversing the order from determination of the bending sequence to determination of the unbending sequence. That is, starting with the bent 3-D product, a directed path to the initial flat layout is searched for. Then, by reversing the order of the unbending sequence, the bending sequence is obtained. Thus, the root of the tree will represent the 3-D bent product, and the operations will correspond to unbend (flattening). Accordingly, goal nodes will now represent the initial flat layout.

The advantage of reversing the order is based on the fact that the initial bending operations (and therefore the last unbending operations) are usually easy to perform; also, many punches can be used for these operations. In contrast, the final bending operations (and thus the first unbending operations) are the most complex, because the product is much more compact at this stage. Consequently, by reversing the order, the more difficult bends are treated first, and impossible bendings can already be identified in the first stages. Thus, unnecessary computations are eliminated, and the size of the tree is reduced significantly.

Another technique used to reduce the computation time and to accelerate the search phase is based on the principle of separation, achieved by introducting a Tool Assignment phase as a pre-search stage. In this Tool Assignment phase, all the tools technically possible are assigned to each bend line. The criteria used in this phase are the criteria recommended by tool manufacturers and bend machine operators [see 7, 8]. The bending technique is dependent on the maximum load of the press brake, the tools, the material used, and the bend parameters (inner radius, length). The die and the punch are dependent on the bend technique, the bend parameters, and the material used.

Reduction in the search domain is accomplished by using the A* search algorithm. The A* algorithm uses domain-specific heuristics to accelerate the search and reduce the number of nodes developed and evaluated. The algorithm is a variant of the Best First Search method in which the node considered to be best is evaluated and expanded (open) in each cycle. Consequently, a front of open nodes is continuously being created. For each open node in the front, the cost to the root is calculated, and the cost to the target is estimated. The best candidate with minimal f(n) is then chosen, as shown schematically in Fig. 5. The algorithm uses the following function f(n) to evaluate a given node, n.

$$f(n) = g(n) + h(n) \tag{1}$$

where g(n) is the cost of the path from the root node (s) to node n and h(n) is an estimation (heuristic) of the cost of the path from node n to one of the goal nodes. Note that g(n) can be recursively computed by

$$g(n) = g(n') + C(n',n) \tag{2}$$

where C(n',n) is the cost of the transition from the parent node n' to its son node n. A* will always find an optimal path to a goal node if the heuristic function h(n) does not overestimate the actual cost to the goal node. However, the more accurate the estimation, the faster the algorithm converges to the solution. To insure a reasonable response time, it is necessary to guess h(n) accurately. In such cases, however, h(n) may be overestimated. Unfortunately, in real-life problems this is a very hard constraint to achieve because it is difficult even to express the various heuristic considerations in the same terms.

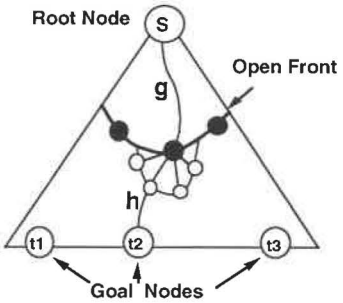

Fig. 5: Schematic path from the root node to a goal node.

Transition from a parent node to its son

Transition from a parent node to its son undergoes strict restriction by the no-collision constraint. Thus, if a collision occurs, the transition is considered unfeasible.

When no collision occurs, the transition cost from a parent node to its son is calculated. The transition cost consists of a minimal fixed price C_f and costs associated with various operations:

(a) Tool change. Tool changing is time-consuming, requiring adjustments and trial bendings. The following relative costs are used by the system:

$$C_t = \begin{cases} 0 & \text{no tool change} \\ 0.5 & \text{either punch or die change} \\ 1 & \text{both punch and die change} \end{cases} \tag{3}$$

(b) Manipulations. During the bending process, the product must usually be reoriented between one bending operation and the next. Rarely can all the bends be accomplished without any reorientation. Each axis of rotation is assigned a different value, reflecting the actual case on the production floor. For example, it is much easier to rotate a long product about an axis parallel to the bending machine than on an axis perpendicular to it. The cost of reorientation is found by first determining the rotation angles α, β, and γ, around the X, Y and Z-axes from the previous position to the current position ($\alpha, \beta, \gamma \in [-\pi, \pi]$). Translation is not considered. Next, the handling cost is computed by

$$C_m = \frac{Kx \cdot |\alpha| + Ky \cdot |\beta| + Kz \cdot |\gamma|}{(Kx + Ky + Kz) \cdot \pi} \quad (4)$$

where Kx, Ky and Kz are weights.

(c) Stability. The preferred state for bending occurs when the product's center of gravity (G.C.) is located between the operator and the machine. A less desirable but sometimes unavoidable state occurs when the G.C. is located at the back of the machine. BEND calculates the G.C. of the product in each step and then computes the relative cost:

$$C_g = \begin{cases} 0 & \text{G.C. is at the machine front} \\ 1 & \text{G.C. is at the machine back} \end{cases} \quad (5)$$

Thus, the transition cost from a parent node to its son is given by:

$$C(n`,n) = C_f + K_{tc}C_t + K_mC_m + K_{gc}C_g \quad (6)$$

where K_{tc}, K_m and K_{gc} are weights (constants) associated with tool change, product manipulation and center of gravity location relative to the machine and the punch.

Heuristic evaluation

In accordance with the A* terminology [11], penalties are assigned to express the heuristic rules. All penalties were normalized into the range [0,1] and were later multiplied by weight. The heuristic rules implemented in the present system are:

(a) A fixed price for punch and die set, P_f. This fixed price reflects the price and availability of the tools. It may also take into account factors such as standard or special tool, ease or difficulty of grinding, etc.

(b) Number of subtools comprising the tool. This penalty gives preference to the use of a single punch and die over grouping a number of short punches (subtools) in order to make a longer one. This penalty is expressed by:

$$P_t = 1 - 1/\#\text{subtools} \quad (7)$$

(c) Bend from outside to inside. Initially, the system finds the product central bend line by computing the distances of the bend lines from each other. The bend with the lowest maximum distance is considered to be at the product center. Each bend line b is given a penalty proportional to its distance from the central bend line bc:

$$P_c(b) = \frac{dis(b_i, bc)}{\max dis(b_j, bc)} \qquad j \in [1,n] \quad (8)$$

where dis(x,y) is the distance (measured in facets) between bend line x and bend line y.

(d) Bend short lines before long ones. The system calculates the penalty for each bend line b_i according to the ratio of its length to the length of the longest bend line, namely:

$$P_l = \frac{length(b_i)}{\max(length(b_j))} \qquad j \in [1,n] \quad (9)$$

Consequently, the heuristic function h(n), needed for Eq. (1), is estimated by:

$$h(n) = P_f + K_{tl}P_t + K_cP_c(b) + K_lP_l \quad (10)$$

where K_{tl} K_c and K_l are the weights associated with the number of subtools, the distance of a bend line from the center, and its length, respectively. Typical values for these weights are 0.5, 1 and 10, respectively. Thus, each open node is evaluated by substituting Eqs. (9) and (10) in Eqs. (2) and (1).

Examples

Two examples are shown below for products consisting of 12 (Fig. 6) and 16 (Fig. 7) bend lines. Several bending steps are shown for each case.

Comparing different evaluation functions

The original evaluating function of A*, assuring optimal solution, is:

$$f(n) = g(n) + h(n) \quad (11)$$

Different, not equal, interpolation between the cost g(n) (from the root to the current node n) and the heuristic h(n) (from the current node to a goal node) can be achieved. Such linear interpolation is given by:

$$f1(n) = w\, g(n) + (1-w)\, h(n) \quad (12)$$

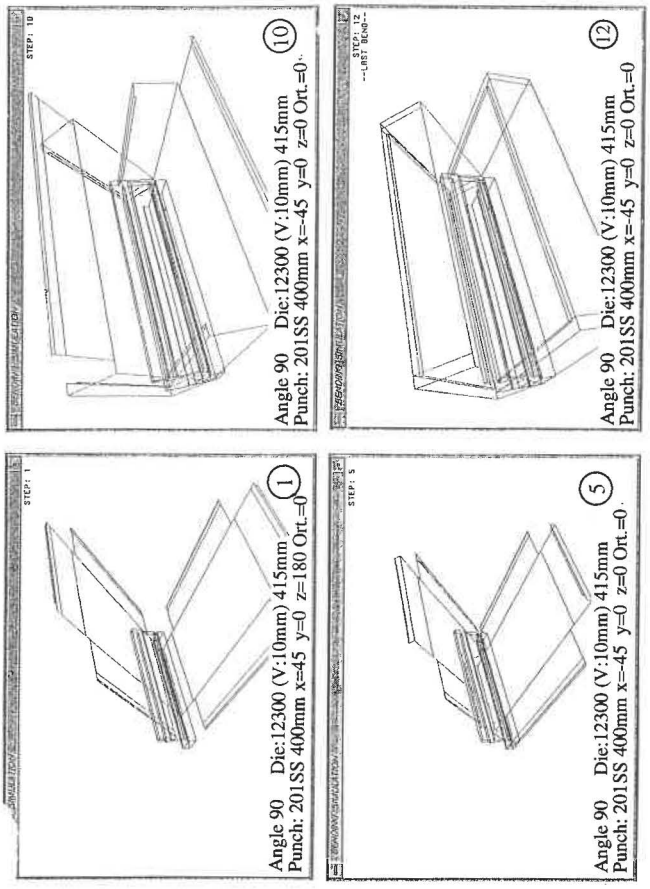

Fig. 6: Sample bending steps for product with 12 bend lines.

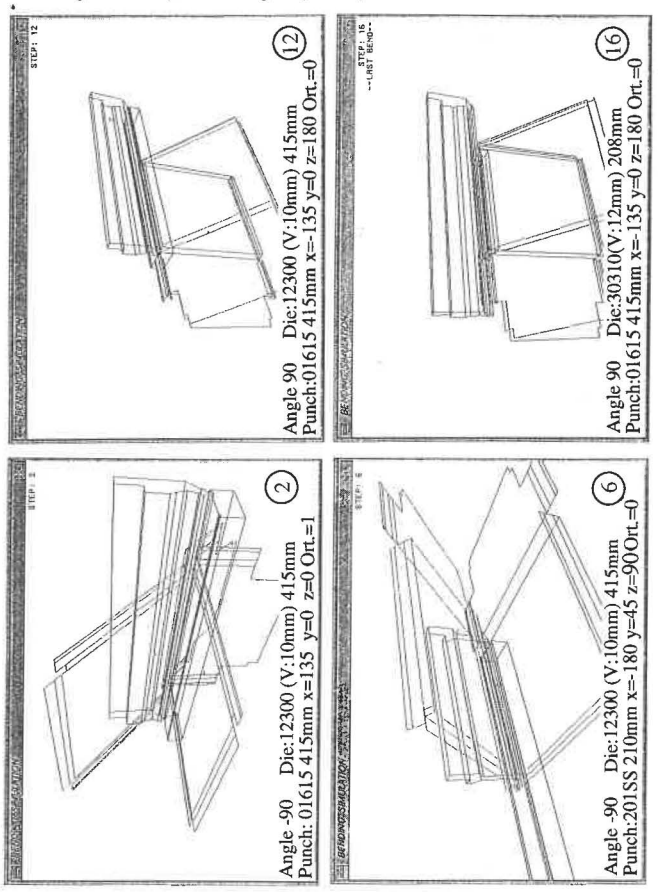

Fig. 7: Sample bending steps for product with 16 bend lines.

The parameter w (w∈[0,1]) can be used to control the algorithm's tendency to deepen the search instead of widening it for a given f function. Lowering the value of w increases the weight of the heuristic at the expense of the cost obtained up to the current node. Since the heuristic usually has a lower effect as the search nears the goal, the overall implication is that deeper nodes are preferred over shallower ones.

As will be shown, enhancing the algorithm's tendency to deepen the search not only accelerates the solution, but it is sometimes the only way to achieve a solution within an accepted time. Otherwise, the search tree becomes very wide and is in effect too large. However, with this reduction in run time, the optimal solution is no longer guaranteed.

A further mechanism for reducing the penalty on deeper nodes is to divide the heuristic factor by the current depth d(n) of the node. For our research, two other evaluation functions have been studied:

$$f2(n) = [wg(n) + (1-w)\, h(n)] / d(n) \qquad (13)$$

$$f3(n) = [wg(n) + (1-w)\, h(n)] / d^2(n) \qquad (14)$$

- f1 is not sufficiently powerful. No solutions were obtained for products with more than 4 bend lines even when w was lowered to 0.1.
- f2 is more powerful than f1, and good results were obtained up to 11-12 bend lines. When presented with a 16-bend line product, the run time became unacceptably long, even when w was lowered to 0.1. The conclusion is that even f2 is not powerful enough for more than 12 bend lines.
- f3 is the most powerful, and good results were obtained for a 16-bend line product.

A comparison between the performances of the two functions f2 and f3 for products consisting of 12 and 16 bend lines is given in Table 1.

		f2		f3	
bend lines	w	sec.	cost	sec.	cost
12	0.1	620	163	445	163
16	0.1	12489	385	311	436

Table 1: Comparison of f2 and f3.

As can be seen, f3 shortened the run time for both the 12- and 16--bend line products. However, the cost for the 16-bend line product rose from 385 to 436. This is due to the DFS-like (Depth first search) behavior of f3.

Analysis of results

The most interesting factors in evaluating the performances of the system are: the cost (goodness) of the bending sequence, the run time, and the number of nodes developed by the system. These results depend on the products themselves, the shape and number of bend lines, the relationship between the product and the heuristics, the evaluation function f(n) which controls the width of the search tree, and the weights. Therefore, it is extremely difficult to evaluate the performance of the system. Initial investigation has been carried out, and partial results are presented here.

(a) Run time. Different products with varying numbers of bend lines were tested. In general it can be concluded -- as expected -- that the run time increases with the number of bend lines, but this is not always true.

To overcome this difficulty and to use products as similar as possible, the following experiment was conceived: A product is chosen and its outer facets are picked one by one, very similar to picking the leaves of a flower. In this kind of experiment, the general structure of the product is modified only slightly from one step to the other. The result for the 12-bend-line and the 16-bend-line products are given in Figs. 8.a and 8.b, respectively.

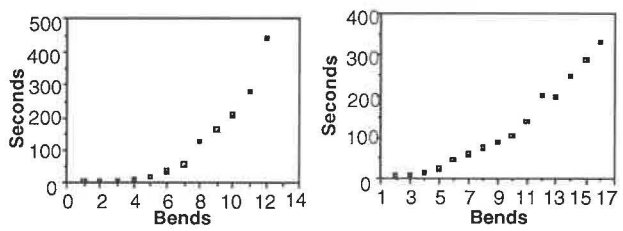

Fig. 8: Run time for (a) 12-bend line product leaves picking; (b) 16-bend-line product leaves picking.

(b) The influence of the relative weight of the heuristic is exemplified. The bending sequences for the two products were determined with different values of w. (Recall that increasing w reduces the influence of the heuristics.)

In Fig. 9.a, the run time and cost for the 12-bend line product is depicted. For determining the bending sequence, the system used the f2 function. It can be seen that while the cost is practically constant (142-143), the run time is significantly reduced when increasing the relative influence of the heuristic (smaller w), that is, run time of about 200 sec. for w=0.1, 0.2 as compared with run time of 4000 sec. for larger values of w 0.6 and 0.7. Good results were also obtained for the 16-bend line product with w=0.3, 0.4, as shown in Fig. 9.b. This time, function f3 was used.

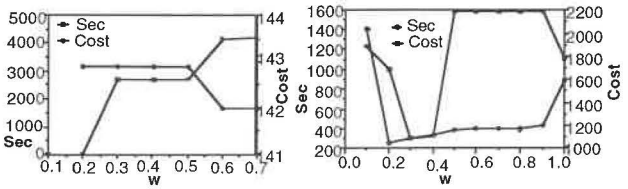

Fig. 9: Run time and cost for (a) 12-bend line product leaves picking; (b) 16-bend-line product leaves picking.

(c) The influence of the weights has also been studied. For example, the consequence of drastic changes in the weight (penalty) for tool change is summarized in Table 2.

bend lines	Kt	tool changes	run time (sec.)
10	100	9	154
10	0	3p	290
12	100	0	440
12	0	4p	702
13	100	1p+2d	105
13	0	5p+2d	421

Table 2: Number of tool changes as a function of Kt remark: p stands for a punch change and d stands for a die change.

It can be seen that when Kt was set to 0, the number of tool changes rose, but the solution time increased as well. The main reason for this is that lowering Kt increases the number of solutions and enlarges the search space.

Summary

This paper presented an application of the graph search method to the generation of bend sequences. The problem was formalized and then transformed into a search problem where the bending sequence is a path in the search tree from the root node to one of the goal nodes. The application, BEND, is based on the A* algorithm. The evaluation criteria and the heuristics used by the system were described in detail. By using these heuristics, BEND was able to generate good bending sequences in a relatively short time period. The ability of BEND to generate alternative bending sequences when various criteria are enabled or disabled or when their relative importance is changed makes it an effective tool for assessing the power of competing heuristics.

References

1. Inui, M., Kinosada, A., Suzuki, H., Kimura, F., Sata, T., 1987, "Automatic Process Planning for Sheet Metal Parts with Bending Simulation," ASME, WAM, Symposium on Intelligent and Integrated Manufacturing Analysis and Synthesis, PED 25:245-258.
2. Huwiler, B., Reissner, J., 1992, "Fertigung von Blechbiegeteilen Rechnerunterstützt Planen," Bänder Bleche Rohre, 10:126-129.
3. Geissler, V., Hoffmann, M., Geiger, M., 1991, "A Product Model for Sheet Metal Parts and its Applications," Manufacturing Systems, 20/3:257-262.
4. Cser, L., Geiger, M., Greska, W., Hoffmann, M., 1991, "Three Kinds of Case-based Learning in Sheet Metal Manufacturing," Computers in Industry, 17:195-206.
5. De Vin, L.J., de Vries, J., Streppel, A.H., Kals, H.J.J., 1992, "PART-S A CAPP System for Small Batch Manufacturing of Sheet Metal Components," 24th CIRP International Seminar on Manufacturing Systems, Copenhagen, 171-182.
6. Huwiler, B., Reissner, J., 1992, "CAPP Based on Rules and Algorithms in the Manufacturing of Bent Sheet Parts," 24th CIRP International Seminar on Manufacturing Systems, Copenhagen, 189-198.
7. Amada Sheet Metal Working Research Association, 1981, *New Knowhow on Sheet Metal Fabrication-Bending Technique,* Machinist Publishing Co. Ltd., Japan.
8. Amada Sheet Metal Working Research Association, 1986, *Bending Sequence,* Machinist Publishing Co. Ltd., Japan.
9. Saddan, D., 1993, "Bending Sequence and Tool Selection for Sheet Metal Product Manufacturing," M.Sc. Project Thesis (Hebrew), Technion, Haifa, Israel.
10. Shpitalni, M., 1993, "A New Concept for Design of Sheet Metal Products," Annals of the CIRP, 42/1:123-126.
11. Pearl, J., *Heuristics,* 1985, Addison-Wesley Publishers, New York.