Move Optimization in Multi-Slide Metal Forming

.

Chicago, IL. http://www.lindo.com

.

Keywords: Metal forming, PERT-CPM, Four-slide forming, Design optimization;

23 January 2006

Abstract

An important segment of the metal forming industry is small metal parts forming. A common machine type used in this segment is the "fourslide" or "multislide" forming machine. In these machines, a metal part is formed as a result of several rams or "slides" pressing against a metal part from several different directions, e.g., four or more. In order to form the part correctly, the rams must press against the part in a carefully choreographed sequence. A notable feature of these machines is high production rate, e.g., several parts per second. The production rate is limited by physical laws related to how much acceleration and resulting stress the components of the machine can tolerate as the machine runs faster. Usually there is some uncertainty about how much acceleration various parts of the forming machine can tolerate. We describe a method for designing this choreography of movement so as to maximize the production rate, subject to keeping the accelerations in an acceptable range. The method involves optimizing a model that is essentially a PERT/CPM critical path problem with "crashing" of activities.

1. Introduction

We encounter "formed" or "stamped" metal parts many times in a typical day. These are metal components that are formed from a metal strip or metal wire. These components are found in computers, switches, windows, and various other appliances around the home, office, and factory. Some examples of formed metal parts are shown in Figure 1.

A typical formed metal component is made as follows:

A metal blank is inserted into the forming machine.

A tool or ram moves into place to firmly hold the blank against a "bed".

A number of other tools, rams, or slides, in a well-coordinated sequence, press or strike the blank to form it into the desired shape. A typical move is very simple, e.g., a) the tool advances in a straight line to strike the blank, or b) the tool retracts and waits for other tools to complete their moves, as well as the arrival of the next blank. The slides or rams typically move in a common plane. At the end of the forming of a part, the part is usually ejected by a ram that moves in a direction perpendicular to the plane of the other slides.

In order to achieve high production rates, the movements of the tools that form the part must be carefully timed or coordinated. A typical production run for a single component may be for more than a million units. Being able to produce these parts quickly and efficiently is therefore of interest. A production rate of 10 parts per second on a single machine is not unusual.

2. The Slide Coordination Problem

The ability to quickly choreograph a set of slide or tool moves has two advantages: 1) a formed-parts manufacturer can respond more quickly to a prospective customer about the likely production rate for his part, and thus, the likely cost per unit, and 2) a faster production rate can be achieved and thus obtain a lower cost per unit than could otherwise be achieved by more ad hoc methods.

Figure 1. Formed metal parts.

This timing problem is very similar to the "PERT/CPM with crashing" problem that arises in project management. For a good introduction to PERT/CPM, see Hillier and Lieberman(2005). From a PERT/CPM perspective, the activities are the moves performed by the various tools during a single cycle that produces one part. The precedence constraints are of several types:

a) Obvious simple precedences for moves of a single tool, e.g., a tool must advance before it can retract.
b) Fairly obvious precedences among tools, e.g., the hold-down tool that holds the blank in in place, must advance before any other tools strike the blank. The hold-down tool cannot retract until all other tools have done their work. These precedences are complicated, however, if one exploits the fact that the movements of tool *x* and tool *y* may be partly overlapped. Tool *x* need not be completely retracted before tool *y* starts to advance. Loosely speaking, these constraints might be of the form: activity *x* must be at least 75% complete before activity *y* is at most 15% complete.
c) Collision constraints between tools. If two tools move through the same point in space, then the first tool must retract past the collision point before the second tool advances through the collision point. For example, in Figure 2, tool 3 must be substantially retracted before tool 4 starts to advance.
d) Pipelining constraints. The moves that produce part *n* may be partially overlapped with the moves that produce part *n*+1. Effectively, an activity late in the "project" (of part *n*) cannot occur too much later than an activity early in the "project" (of part *n*+1). An example is that the ejection tool need not be completely retracted after ejecting part *n*, before part *n*+1 starts to move into place.

A crude schematic of a part forming machine is shown in Figure 2.

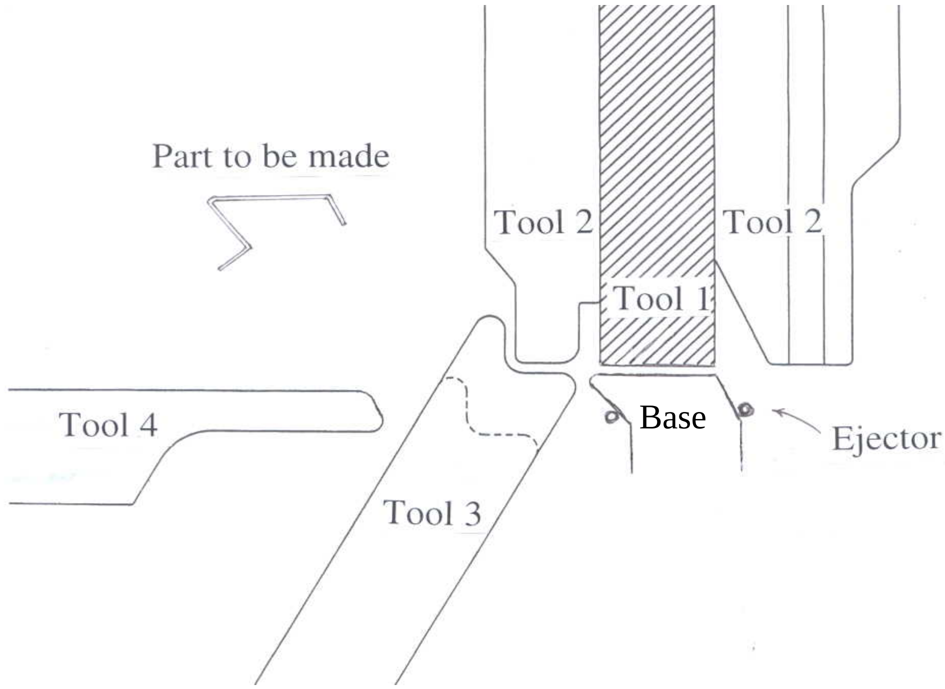

Figure 2. A multislide metal forming machine at the end of step 4.

The essential steps in forming the part in Figure 2 are:

1) The flat blank is brought in from the right,
2) Tool 1 comes down to hold the blank in place against the base,
3) Tool 2 comes down part way to provide an anvil for tool 3,
4) Tool 3 comes up from the lower left to form the clockwise right angle in the part,
5) Tool 3 retracts,
6) Tool 2 moves further down to start forming the counterclockwise angle in the part, and also to form the clockwise obtuse angle on the right,
7) Tool 2 retracts,
8) Tool 4 moves in from the left to complete the forming of the acute counterclockwise angle against the base,
9) Tool 4 retracts,
10) Tool 1 retracts
11) The ejector ejects the part towards the viewer.
12) The ejector retracts.

An obvious question is, how much time should be devoted to each move? If a tool must move a great distance or it is heavy, then it probably should be allotted more time. Not quite so obvious is that some of the moves can be overlapped, at least partially. For example, tools 1 and 2 can come down almost simultaneously. How do we represent the amount of overlap allowed?

Move i is described by the input data:

d_i = distance the tool is to be moved, > 0 for an advance,
< 0 for a retraction,

m_i = the mass of tool associated with move i,

u_i = upper limit on the force that can be applied to move i. This depends mainly upon the tool involved, but it is also related to the move, e.g., whether it is an extend or retract.

There are two decision variables for each move:

s_i = start time of move i,

Δ_i = the time taken to make move i,

An indirect decision variable is:

f_i = force required to make move i,

The objective is to minimize:

C = cycle time.

3. Limiting the forces required

The speed of a forming machine is usually continuously variable, with top speeds ranging from 200 to 800 cycles per minute. As you increase the speed of the machine, the machine will start to vibrate. At higher speed, bad things will begin to happen, typically, a component of the machine that was assumed to be a rigid structure, will in fact start to flex, and thus timing calculations based on rigid structures will be inaccurate, and things may get bent. So we want to keep the forces small. The force required by move i depends critically on three things: a) the mass of tool used in move i ; b) the distance d_i, and c) the time allowed, Δ_i.

From elementary physics, recall that the force required to accelerate an object is proportional to the mass times the acceleration, or algebraically, $f = m*a$. If a constant force is applied, then the force is inversely proportional to the square of the time allowed for the move, or algebraically, $d = a*t^2/2$, so $a = 2d/t^2$. A reasonable assumption is a tool is accelerated at a constant rate during the first half of a move and decelerated at a constant rate during the second half of the move. Thus, the acceleration force that needs to be applied during the first half of move i to move the tool distance d_i /2 in time Δ_i /2 is:

$$f_i = 4*m_i* d_i*/(\Delta_i*\Delta_i).$$

A similar comment applies for the deceleration during the second half of the move. So the mathematical model is:

Minimize C,

For each simple precedence pair i and j:

$$s_i + \Delta_i \leq s_j ,$$

Acceleration for each move:

$$f_i = 4*m_i* d_i/(\Delta_i*\Delta_i),$$

$f_i \le u_j$,

Cycle time:

$s_i + \Delta_i \le C$;

$0 \le s_i$, Δ_i

This looks like a nonlinear program because of the expression for f_i. Notice, however, that, $4*m_i* d_i/(\Delta_i*\Delta_i) \le u_j$, can be rewritten as the linear constraint:

$$(4*m_i* d_i/ u_j)^{0.5} \le \Delta_i$$

Thus, only a linear programming(LP) solver, rather than a nonlinear programming(NLP) solver is needed to solve the problem. LP tends to be faster and more robust.

4. Estimating the maximum acceptable force

It may not be obvious from first principles or a machine users manual, what is the maximum force that a particular tool can tolerate. If a spring is involved for a retraction move, we may have data on the force the spring can produce when it is extended. Otherwise, one may have to experiment and deduce the force. Recalling our fundamental formula from physics, if a constant force f, measured in newtons, is applied to an object initially at rest, of mass m, measured in kilograms, for a time t measured in seconds, then the distance d, measured in meters, that the object moves during this time is given by the formula:

$$d = f*t^2/ (2*m).$$

For a typical standard move curve, a reasonable approximation is that a constant force is applied during the first half of the move in the direction of the move to accelerate the tool. During the second half of the move, an equal but opposite constant force is applied to decelerate the tool. Analyzing the first half of move i, we can re-write the above as:

$$(d_i /2) = f*(\Delta_i /2)^2/ (2*m_i).$$

Solving for f we get:

$$f = 4*d_i *m_i /\Delta_i^{\ 2}.$$

Suppose we have empirically observed that a tool weighing 1.2 kilograms can be successfully moved a distance of 2 cm in 0.1 second. Thus, the implied force was:

$$f = 4*.02 *1.2 /.01^2 = 960 \text{ newtons.}$$

So we know this tool can tolerate at least 960 newtons.

The orientation of a tool makes a difference. If the tool is being accelerated on an up move or decelerated on a down move, then the force of gravity must also be overcome. The force of earth's gravity on one kg is about 9.7536 newtons. Thus, if a tool is in a vertical orientation, an additional force of 9.7536 newtons/kg is needed in either the accelerate or decelerate half of the move. A plausible first approximation is to simply subtract this amount from the upper limit on the force.

5. Partial move precedences and standard move curves

If you consider tools 2 and 3 in Figure 2, it is clear that tool 3 need not be completely retracted before tool 2 starts to make its second move down. For sake of definiteness, let us say that tool 3 must be 50% retracted before tool 2 begins its second descent. Somewhat similarly, if tools 2 and 4 are to avoid a collision, tool 2 need not be completely retracted before tool 4 starts to advance. E.g., the collision point might correspond to tool 2 being 45% retracted and tool 4 being 35% extended. It happens that we can represent these "partial precedences" if we require moves to follow a "standard move curve" defined below.

If a tool is allotted one second to make a move, then a standard move curve specifies the position of the tool at any intermediate time. Define:

$P_i(t)$ = relative position of the tool used in move i, for $0 \le t \le 1$.
Thus, in particular, $P_i(0) = 0$ and $P_i(1) = d_i$. A standard move curve is illustrated in Figure 3. Loosely speaking, the move curve tries to keep the acceleration, and thus the force, relatively constant throughout the move.

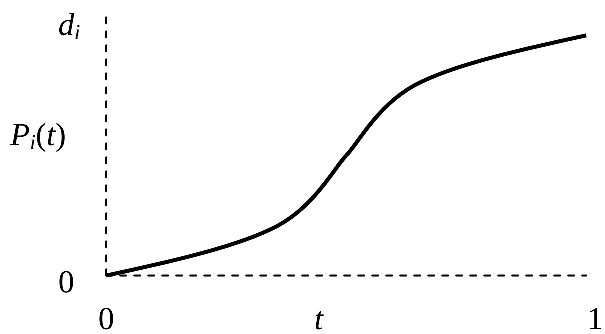

Figure 3. A standard move curve

If we allow an arbitrary amount of time, Δ_i, to make move i, we make the obvious scaling generalization that the position of the tool at time t, relative to the start of the move, is $P_i(t/\Delta_i)$, for $0 \le t \le \Delta_i$. Restricting moves to a standard move curve is quite reasonable, especially with cam driven tools. For example, almost all internal combustion engines, with the exception of Honda's variable timing VTEC engines, use a standard move profile for moving intake and exhaust valves via cams.

Suppose there is a partial move precedence between move i and move j, specifically, suppose there are constants δ_i and δ_j, such that the tool involved with move i (typically a retract) must have moved a distance of at least δ_i ($\leq d_i$) before the tool involved with move j (typically an extend)has moved a distance δ_j ($\leq d_j$) . In order to enforce this, we need to know the time at which each tool has reached its δ distance. Effectively, we want to solve:

$$\delta_i = P_i(t/\Delta_i),$$

that is, given δ_i and Δ_i , find t. Graphically, this is fairly intuitive: given a point δ_i on the vertical axis of the standard move curve, find the corresponding point t/Δ_i on the horizontal axis. Algebraically, if we denote the inverse function of $P_i()$, by $P_i^{-1}()$, then we can write:

$$t = P_i^{-1}(\delta_i)^* \Delta_i ,$$

Finally, the partial move precedence can be written as:

$$s_i + P_i^{-1}(\delta_i)^*\Delta_i \leq s_j + P_j^{-1}(\delta_j)^*\Delta_j.$$

Notice that the model is still linear in the decision variables(i.e., s_i, Δ_i, s_j, and Δ_j).

6. Pipeline precedences

A computer is said to be pipelined if sub-operations of operation n+1 are allowed to start(enter the pipeline) before all the suboperations of operation n are completed(exit the pipeline). This kind of overlapping of operations is also possible here. For the part being manufactured in Figure 2, the ejector, after ejecting part n, need not be completely retracted before part number n+1 starts to enter from the right. Notice that the ejector tool was wisely positioned so that it is not in the path of the incoming blank. For example, suppose that part n is well clear of the input area if the ejector is at least 90% extended. Thus, we might require that the ejector be at least 90% extended before the feed mechanism bringing in part n+1 is 20% extended. Similarly, it is clear that the ejector must be completely retracted before tools 2 and 4 advance. How do we represent these pipeline precedences?

The argument is fairly simple. If move j starts at time s_j on part n, then move j starts at time $s_j + C$ on part n+1. Thus, if we want move i on part n to have moved a distance of at least δ_i before move j on part n+1 has moved a distance δ_j, then we would write:

$$s_i + P_i^{-1}(\delta_i)^*\Delta_i \leq s_j + P_j^{-1}(\delta_j)^*\Delta_j + C.$$

Again, it is still linear.

In designing cams to drive a tool, a move is typically described in terms of degrees of rotation of the cam, where the cam makes one rotation every cycle. Thus, the degree at which move i starts is given by $360^* s_i / C$, and its duration is given by $360 * \Delta_i / C$. This is a nonlinear calculation, but it can be done after the optimization determines Δ_i and C.

8. Eliminating alternate optima and reducing the probability of failure

The model as formulated may have alternate optima, that is, alternate solutions, all of which have the same minimum cycle time. The solutions may differ in the move times of non-critical moves. A simple way of eliminating most of these alternate optima is with the slightly refined objective:

Minimize $C - \Sigma_i \Delta_i / M$,

Where M is a suitably big number. The purpose of the additional term, loosely speaking, is to cause the solver to choose the longest possible move time for a move, if it will not increase the cycle time. The probability of failure due to deformation of parts is less if move times are longer. We want the first term to dominate so that we first minimize the cycle time, and then given the minimum cycle time, make the sum of the move times as large as possible. Suppose there are N moves in total. In an extreme case, all moves could be done in parallel. In order for the first term to dominate in that case we would need $M > N$.

A third order consideration arises if two non-critical moves, say k and j, must be done in series. The above objective will be indifferent between making Δ_k small (corresponding to a large force) and Δ_j large, or making Δ_j small and Δ_k large. The probability of distortion and malfunction increases rapidly as force is increased, so a better solution is probably to make <u>both</u> Δ_j and Δ_k modestly larger than their lower limits. Define θ as the amount by which we would like a move to exceed its lower limit time if it will not increase the cycle time. Let the variable e_i be the minimum of θ and the amount by which Δ_i exceeds its minimum time. We can then get a solution that spreads the forces more evenly around among non-critical operations by using the objective:

Minimize $C - \Sigma_i (e_i + \Delta_i) / [2(N+1)]$,

and (recalling the expression for the lower bound on Δ_i), adding the constraints:

For all moves i:

$e_i \leq \theta$;

$e_i \leq \Delta_i - (4 * m_i * d_i / u_j)^{0.5}$;

If a typical cycle time is 1/5 of a second, and a typical move takes 1/5 of a cycle, then a plausible value for θ might be of the order of 1/100 second. To illustrate, suppose the lower bounds on Δ_i and Δ_j are both .05 and θ is chosen as .01. Further, suppose that each of the following three solutions are feasible for Δ_i and Δ_j: (.05, .07), (.07, .05), (.06, .06). The above mechanism will choose (.06, .06).

9. Summary and conclusions

The linear program model described earlier was setup in a spreadsheet so that it could be solved using the What's*Best* solver, see What's*Best*(2005). As input it required three data sets:

1) For each tool or slide:

Name, Max acceleration

2) For each move:

Move_name, Name of tool, Distance to move

3) For each predecessor/successor move pair:

Predecessor move, partial move dist,
Successor move, partial move dist, same cycle(Y or N)

The Output data are:

1) Time per cycle,
2) For each move: Start time, End time.

A preliminary application of the ideas described here allowed the production rate of a certain part to be increased to 35,000 parts per hour from 25,000 parts per hour by a single, simple change in the tool coordination.

References

Hillier, F. and G. Lieberman(2005), *Introduction to Operations Research*, McGraw Hill, New York.

What'sBest! User's Manual(2005), LINDO Systems, Chicago.