THE FABRICATOR®

Software evolves to take a humanlike approach

The best nest considers the big picture

By Erik Kettenhofen
July 6, 2009

Nesting software has evolved to the point where it can "see" shape contours and can determine how they best fit together.

Some operators tend to tweak nests. Unfortunately, the time spent tweaking may affect throughput; why invest in automatic nesting if someone spends time to manually change it? Besides, an hour spent rearranging parts for 1 percent better material utilization costs more than using the original nest. Many see tweaking as just human nature, but often there's a reason a person has the urge to do it. As if solving a jigsaw puzzle, the operator can just "see" that the material nest may not use every bit of space as efficiently as it might.

Nesting software works within constraints, and material utilization isn't the only factor. Software can, for instance, give hot jobs priority. A part may be rotated certain ways because of a punch tool orientation; a subsequent bending operation or final part may require the metal's grain to be in a certain direction; or the programmer may set rotational values high—parts can be rotated only in 90-degree increments, for instance—to either speed software processing time or meet a design requirement.

Software also views the nest in a fundamentally different way. People working jigsaw puzzles concentrate on shapes and contours, as did skilled tradesmen who manually nested parts in past decades. A person looks at parts and sees immediately how they relate to other parts—for instance, how two semicircular parts could be nested in a circle with many smaller parts in the middle.

Software, however, must take a mathematical approach to any problem, and nesting software is no exception. Depending on the product, the software nests priority parts first, usually the largest ones, unless the programmer gives certain parts priority based on the schedule and due dates. Then comes the next-largest part, and so on, and the software rotates each by designated increments to find the best fit.

Software can't "learn" the way a skilled tradesperson does after years of nesting experience. The human naturally doesn't try fitting parts in a certain arrangement if it didn't work for a similar nest. Software can save certain nests in memory, but if jobs and part priority change continually—as they do at many contract manufacturers—the program must go through the same deductive process again and again, considering one part at a time. It's trial by error: working one part, rotating it, finding the best fit, then moving on to the next part and doing the same thing.

Because computers run thousands of calculations a second, the method has worked extremely well, particularly as computer processing power has skyrocketed during the past decade. What would take software several minutes or hours to accomplish likely would take a human several days. Sure, the human would produce a more elegant nest with better material utilization, but the hours or days it would take to make the nest would outweigh any material utilization gains.

In most circumstances, no computer in the world can calculate all the possible nests for a given set of parts. To put this in perspective, 50 parts can be arranged into 10100 different nests, if no parts are rotated. If rotation is considered, the problem becomes even larger. If a computer could generate 1 million nests per second (faster than any computer built so far), it would still take 1084 lifetimes to calculate all the possible nests for just 50 parts.

Some of the latest nesting software developments, though, hark back to the way skilled people developed nests manually years ago. Instead of placing a part, then moving on to the next part, software now, through new mathematical algorithms, can look at the specific angles and contours of parts to "see" how they fit against other part features. In other words, it can see how multiple shapes fit together. If the software places a part in a certain way, it does so because it sees other parts that could be placed around or inside it. For instance, two semicircular parts may, when placed together, form a large circular space in the middle that could make room for smaller parts in the nest. Software technology now can emulate how humans made decisions when developing a nest manually—only, of course, much faster. This ability to see how parts relate to each other means the computer (like people) need not go through trillions of possible solutions to find the best one.

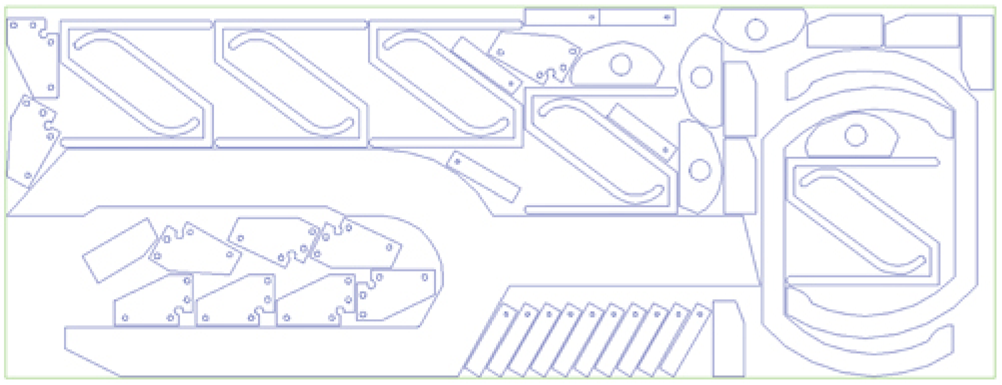
Note that material nesting is just one part of the big picture. Regardless of the nesting software a shop uses, how efficiently a company nests sheets matters little if the wrong information reaches the nesting program to begin with. Information flow between order entry and the shop floor is just as important as part flow between the primary fabrication processes and the finished product.

An efficient nesting method can help ensure smooth information flow before cutting and smooth part flow after. Hypothetical examples here show how one nesting algorithm technology—dubbed vision emulation—works. Note that the nests here do not include any constraints. No part is limited as to how it may be placed on the nest to account for, say, required grain direction or punch tool orientation. The software was set to calculate the final nest based on maximum material utilization, which is valuable if you're working with expensive metals, such as thick, high-strength plate, but perhaps not so valuable if hot jobs get pushed by the wayside.

Most shops cutting common metal like 20-gauge mild steel likely would use what's called a value approach to nesting, which considers scheduling priorities, available material, grain constraints, and other factors. For instance, if other orders were started on previous nests, the software considers those elements as part of a "broken order" and can make it a priority to place those parts on the next sheet available, so that all parts arrive at welding or assembly when needed.

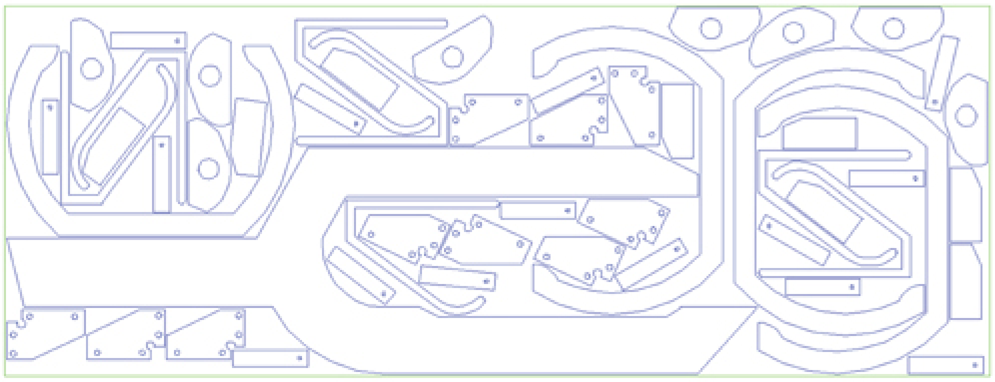
Still, the examples here do show how software can see shape relationships to produce a nest. And though it's only a part of the big picture, it can help propel an overall nesting strategy in an efficient direction.

First Fit Approach

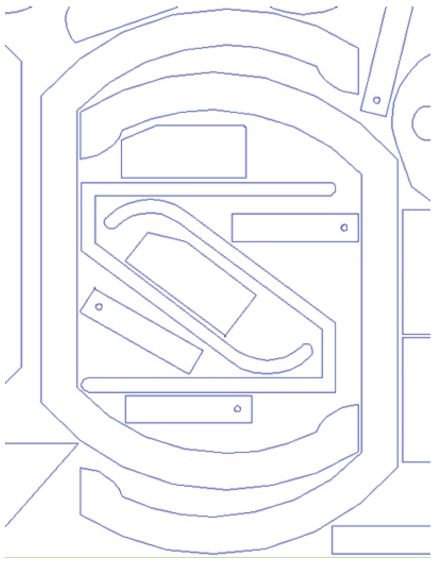


This nest (see left figure), using what's called a first fit approach, gives the larger and most complex parts priority placement, but considers each part individually, placing one part in the nest, rotating it for the best fit, then moving on to the next part. The tuning fork part was placed first at the bottom left; followed by the large, C-shaped parts at the right (grouped together in a so-called 180-degree pair); followed by the smaller, thin, complex-shape parts above the tuning fork on the top left and also inside that C-shaped, 180-degree pair. The smallest parts took the lowest priority and were placed last, inside the tuning fork shape and other empty spaces.

Vision Emulation (VE) Approach



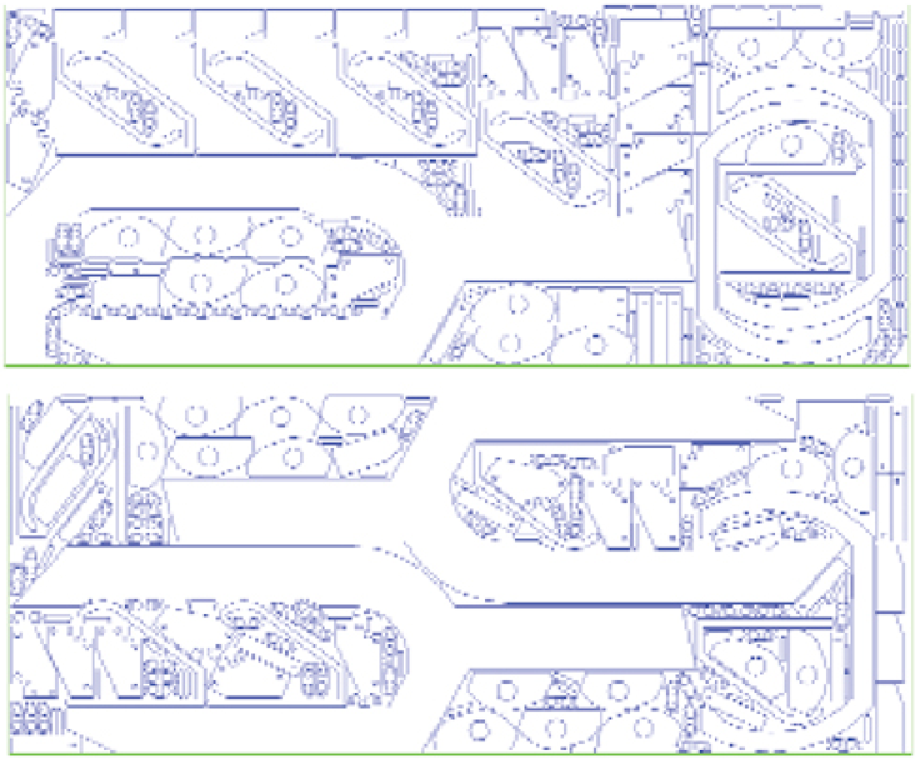
This nest (see left figure) takes what's called a vision emulation (VE) approach. Like the previous method, it prioritizes parts based on size, but this technology looks at how part shapes relate to each other. The first part is placed anywhere on the nest that would produce the most efficient nest arrangement. In this case, the tuning fork part—the largest and most complex—happened to remain on the bottom left, but by flipping it 180 degrees, the software could see that it would open up more nesting possibilities, such as the grouping of parts within the large, U-shaped part on the top left.



Seeing Part Complexity

This portion of the VE nest (see right figure) shows how the software prioritized parts by size as well as by shape complexity. The software has discovered that if it flips the two C-shaped parts 180 degrees, the arrangement allows enough space to fit in those smaller, lower-priority parts in the middle. It also views those thin, complex parts (which look like a field hockey stick at one end) as ones that could be grouped together to allow still more parts to be placed inside.

Determining Part Vale



These nests (see left figure) have those same tuning fork parts, only now more orders have been added to the mix. The nest on the bottom, which uses the VE algorithm, exhibits slightly greater material utilization than the one on the top. VE recognizes that not only is the tuning fork-shaped part large, but it also has an open space in the middle. That open space gives that complex part an even higher value, because it knows it will be able to squeeze more parts into those open spaces. For this reason, the algorithm made it a priority to fit as many of those tuning fork parts as possible onto the nest: in this case, two.

Again, regardless of the software algorithm used—VE, first fit, or anything else—material utilization remains just one part of the overall picture. The ability to prioritize parts based on the schedule, geometry constraints, punch tool orientation, and other limitations can be just as important. Software draws information directly from the job's CAD files, and in some instances the nesting engine can communicate directly with MRP or ERP systems to ease the flow of job information.

For instance, what if the above tuning fork had to be in a certain grain direction? In these circumstances, it's important to recognize this constraint and nest accordingly. Decreasing scrap and increasing material yields are of little benefit if the job has to be recut because of an unforeseen part requirement.

Erik Kettenhofen
Applications Engineer,
Optimation Inc.

18600 E. 37th Terrace S., Independence, MO 64057, 816-228-2100

www.optinest.com