IntelliNest

Prepared by Mark Schuessler

Hypertherm, Inc.

November 15, 2021

Table of Contents

Revision History

Name	Date	Reason for Changes	Version
Mark Schuessler	12/5/2010	Initial draft	1.0 Draft 1
Mark Schuessler	12/6/2010	Revised based on feedback from Doug Geiger	1.0 Draft 2
Mark Schuessler	12/27/2010	Revised based on feedback from John Pobedinsky	1.0 Draft 3
Mark Schuessler	3/1/2010	Adding scoring information	1.1 Draft 4
Mark Schuessler	11/15/2021	Edited for fun	2.0 Draft 1

1 Introduction

1.1 If you are reading this…

If you are looking for a high level overview of IntelliNest then you are in the right place. The first five sections of this document will describe the nesting problem, the general approach IntelliNest takes in solving the nesting problem, and the single, most important thing IntelliNest must do right to succeed.

If you are looking for detailed information about how scores are calculated for the approach then you, too, are a winner. Section 7 provides a detailed listing of the information we gather and generate for use in IntelliNest.

The final section discusses the shortcomings of IntelliNest.

2 The nesting problem

2.1 Why we nest

We nest things to minimize waste. That's basically it. Minimizing waste saves money. We try to fit the most parts in the least space, and we try to do it as quickly as possible. Our customers can usually produce a good result by manually nesting the parts themselves, but this is a time consuming, labor intensive effort that they would rather avoid. Our goal is to be able to match or surpass the utilization they produce manually and to do it in far less time. We want to be fast. Why? For one thing, it looks impressive on a demo. We don't want to look slow and clunky compared to the competition. Is the customer willing to trade some speed for better utilization? Yes, but it depends on how much speed and how much utilization. There is some mixture of utilization and speed that is optimal for every nesting situation and that's where we want to be. There's another reason we want to be fast. The faster you are, the more things you can try and the more things you can try the better the nesting result. In situations where utilization is the primary concern, speed still pays off.

2.2 Why nesting is hard

Nesting *is* hard. Why? The only way to produce an optimal result for a nest is to try every possible combination of values for every factor of the parts and plate to see what results in the best utilization. For the parts alone there are a number of different factors – the order in which to nest the parts, the rotation angles to try, the mirror states to try, the number of torches to try, the location of lead-ins and lead-outs and the offset to place each part.

The parts themselves may be of any shape or size. They may have cutouts in which other parts may be nested. Plates, too, come in different shapes and sizes. They may be rectangular, circular, remnants (any irregular shape) or skeletons, which are full plates or remnants that contain cutouts. Plates may have safe zone constraints. The machine may require the use of "work zones", which limit the effective area that may be accessed by the cutting head at a given time, resulting in repositioned safe zones and nesting zones.

All of this only addresses the problem of nesting a single plate. It doesn't even consider trying to find the best plate for a nest, or the optimal set of nests for a job. Nesting is hard because the only way to get the optimal result is to try everything and to try everything would take way, way too long.

2.3 If only we had more time…wait, we have a time machine!

Well, if we did have a time machine we wouldn't be having this conversation, because we could just try everything, produce the optimal result, go back in time and flash it on the user's screen. Nesting would be instantaneous (and optimal!) But we don't, so we have to try to do what we do as quickly as possible. This means we can't try every possible combination of nested parts. We can't even try the majority of them. We can only try a tiny fraction of the possibilities. But which ones?

3 Nesting solutions

3.1 Nesting strategies

This nesting approach is simple. We provide several different nesting technologies – profile nesting, pattern nesting and rectangular optimization – each of which produces a good result in the right situation. Rectangular optimization is a look-ahead algorithm designed to produce good nests when you can place 4-8 rectangles on each nest. However, it thinks everything is a rectangle and it doesn't even know about cutouts so it does no filling. Pattern array is another look-ahead algorithm that works well when you have large quantities of a single part. It seeks to maximize the number of parts placed by trying different patterns of the part paired with itself. Profile nesting is a general purpose, greedy algorithm that goes through the list of parts from largest to smallest, finding the best location for each part one part at a time.

3.1.1 How it works

These nesting technologies can be combined into nesting strategies, each of which also allows you to set the number of rotations to try along with various other settings. But all strategies use the same template:

For a given nest:

1. If the plate is not empty to start with, fill the already nested area (or if it's a skeleton, fill the region containing the cutouts)
2. Run rectangular optimization on the rectangular parts (if the option is selected), and then fill the nested region.
3. Grab a part and nest it using pattern nesting (if the option is selected) or profile nesting.
4. If using pattern nesting, fill the nested area. (Filling using pattern nesting is also an option.)
5. Repeat steps 3 and 4 until the nest is full.
6. Fill the entire nest for good measure.

It's a bit more complicated than that, but that's the gist of it. The order of things doesn't change. Yes, we can use different combinations of these nesting technologies (and different levels of pattern nesting –

basic, intermediate and advanced), try more orientations of each part until we find a fit, even use slightly different weightings to determine which position is better for a given part but there is one important point: *the decision of how we are going to nest a set of parts is made without considering anything about the parts.* And the decision has to be made by the user who has to set up the strategies and choose one to nest with. The chosen strategy is applied to the entire part set until the entire job has been nested. Even during nesting system optimization, which allows the user to try multiple strategies and pick the best result, each nest is produced by a single strategy.

3.1.2 Why it doesn't work better

Our nesting strategies produce pretty good results, all things considered, but have a number of shortcomings.

3.1.2.1 We always follow the same template

It doesn't matter what the set of parts is, or where we are on the nest, or what the situation is - we always do the same thing the same way in the same order. Obviously, this is not always going to lead to even a good nest, much less the best nest.

Take a simple example: say we nest a large part or pattern. Next, we will try to fill the nested area with smaller parts. Why? Because that's what the template says to do. But what if there was another large part that could interlock with that first large part you just nested? Too bad. You've already filled the nested region with smaller parts and you may no longer be able to fit the second large part on the nest. So instead of placing the second large part on the nest, you will now have to use even more small parts to fill the remainder of the nest. Then that second large part is pushed to the next nest and you may not have enough parts left over to fill the second nest. You will probably end up generating more scrap then if you had fit both large parts on the first nest.

That's just one example. Someone manually nesting would discard this template when it makes sense, and so should an automatic nesting system.

3.1.2.2 We can only use one nesting strategy per nest

Once the user selects a nesting strategy and hits go, that single strategy is used until nesting completes. Never mind that all the parts that could benefit from that strategy are nested on the first quarter of the first nest, we're sticking with that same strategy until the bitter end. That's crazy. A smart automatic nesting system would change strategies on the fly, depending on the current situation of the nest and the remaining parts. The nesting technologies we have developed are effective when they are used in the right situation with the right parts but there is an assumption that it is possible to pick a single strategy and use it to nest all of the remaining parts on all of the remaining plates. That is a bad assumption. It would be an improvement even to auto-select a new strategy at the beginning of each new nest. It would be even better to choose a strategy at each point in the nest where it would help.

3.1.2.3 We force the user to select a strategy

Even if there was a strategy that would lead to the best result for an entire job, is it realistic to expect every one of our users to know which one it is?

3.1.2.4 It is hard to handle new cases

We sometimes get feedback from customers that automatic nesting does not produce as good a result as expected. Sometimes we can rectify this with a simple tweak of the current algorithm (or by telling the user to change a setting or use a different strategy.) Other times we are stuck. Why? Because the current nesting algorithm is sort of a central planning design – most of the decisions are made in one place. This makes it complicated and difficult to modify. Moreover, modifications can have unintended consequences so we are reluctant to make modifications, because if you make one case better and ten cases worse, you end up worse off than you were. An automatic nesting system designed for easy addition of new cases would be a big improvement.

3.1.2.5 We don't recognize enough special cases

This is an extension of the previous item. We have always focused on solving general problems of automatic nesting but there are a number of special cases that come up often enough and if you handle them properly, you get a far superior nesting result. Often, these special cases are what a customer uses as a nesting benchmark when comparing nesting software, assuming if the nesting system can handle the special case well it will handle the general cases well. Because we haven't focused on these special cases (and because adding special cases to the current code base is problematic) we sometimes lose on these benchmarks. Often, if you simply recognize a nesting situation, you can place the first handful of parts down in a way that ensures a far better result than you would have gotten by following the same template you always follow.

3.1.2.6 We don't look ahead

We just nest whatever part (or parts, in the case of pattern array or rectangular optimization) we are currently nesting as best we can. We don't look to see what parts and space are going to remain, or whether they are likely to nest well together, or if what we are doing is going to lead to a good nest or a bad nest. We use the *Wall Street* (i.e. Gordon Gekko) "Greed is good" approach, trying to optimize the current step while ignoring the consequences for later steps. An improvement would be the *Beautiful Mind* (i.e. John Nash) approach where we try to maximize the utilization of what we are currently nesting while leaving a remaining part set and available space that are likely to lead to the best nest.

Some of our nesting technologies (i.e. pattern nesting and rectangular optimization) actually do look ahead to maximize nesting utilization but they do this only for the part or parts those technologies are nesting and not for the nest as a whole. So, although certain nesting technologies use a look ahead approach in a given region of the nest, the overall nesting approach does not look ahead.

3.2 Other existing solutions

There are a number of automatic nesting systems on the market. Most of them use one or more of the following approaches: genetic algorithms, vision nesting, pre-nesting parts and handling special cases.

3.2.1 Genetic algorithms

Genetic algorithms seek to emulate evolutionary processes of inheritance, mutation and natural selection. These are becoming more widely used and do a very good job particularly with irregularly

shaped, dissimilar parts, especially of low quantity. In these cases, genetic algorithms can produce nests that would be impossible to produce in any other way.

Pros	• Genetic algorithms can be applied to any part set. • They do an exceptional job for irregular shapes that have to be rotated at odd angles to nest well together. • The basic algorithms themselves are well known. This is established technology now. • It is a look-ahead approach in that it tries a number of different nests, each one attempting to improve upon its predecessor.
Cons	• It can be slow (sometimes needing to run overnight). • It can produce irregular, ugly looking nests. (Yes, utilization is king but all things being equal, users prefer an ordered looking nest). • Certain nests (i.e. patterns) can be generated faster and/or better using other approaches.

3.2.2 Vision nesting

This approach uses computer vision technology to "see" locations where a part will fit.

Pros	• It does a nice job of matching up the available parts to the available space. • It doesn't have to limit the parts by rotation angle.
Cons	• It is mainly a filling algorithm. • It focuses on finding the best fit for the current part and not on how that decision will affect the remainder of the nest.

3.2.3 Pre-nesting parts

This approach involves pre-grouping two or more parts to form a tightly nested rectangle and then nesting the rectangles, usually in rows and columns. It generally includes building pairs of a single part.

Pros	• It is easier to nest rectangles than irregular shapes. Because of this it probably lends itself well to looking ahead. • It produces regular; aesthetically pleasing nests that customers appreciate (provided the utilization is good.) • For the part sets it works well for, it probably out-performs genetic algorithms.
Cons	• It only works well for certain part sets, usually requiring parts that pair well together, are medium size or smaller and have sufficient quantity. • It doesn't work well for part sets containing large, irregularly shaped, small quantity parts.

3.2.4 Handling special cases

Most nesting systems handle special cases to some extent, which consists of recognizing some number of nesting situations and doing something unique to get the best result in those situations. For example, if you have two large L-shaped parts, it often helps to place them in opposite corners (if there are filler parts available) to maximize the remaining usable space.

Pros	• Handling special cases usually results in significant improvement for nests involving the special case. • Customers often set up their benchmarks to see how a nesting system handles their most difficult cases, assuming if it can handle the special case it will have no problems with the general case.
Cons	• This is not a general purpose approach. It only works for the cases you recognize. • Adding support for new special cases can have unintended side effects on general nesting.

4 IntelliChoice

IntelliNest, which is discussed in the next section, was developed to make intelligent decisions to nest instead of following a fixed template like our old nesting strategies. It uses multiple nesting technologies – Nesting Solutions, in IntelliNest speak – to generate a completed nest. This often works well, but sometimes produces results no better than the older nesting strategies, while taking longer to do it. Thus, IntelliChoice was developed as a higher level decision maker to choose the best approach for each nest from among our old nesting strategies and IntelliNest. This significantly increased the speed, while still relieving the user from having to select an appropriate nesting strategy.

5 IntelliNest

The biggest problem with our old nesting strategies was that each strategy followed a pre-determined set of steps to nest the parts, regardless of the properties of the parts or the plate. IntelliNest sought to overcome this by choosing the correct nesting approach for a subset of parts at each decision point in the nesting process.

5.1 In a nutshell

Here is the IntelliNest approach in a nutshell:

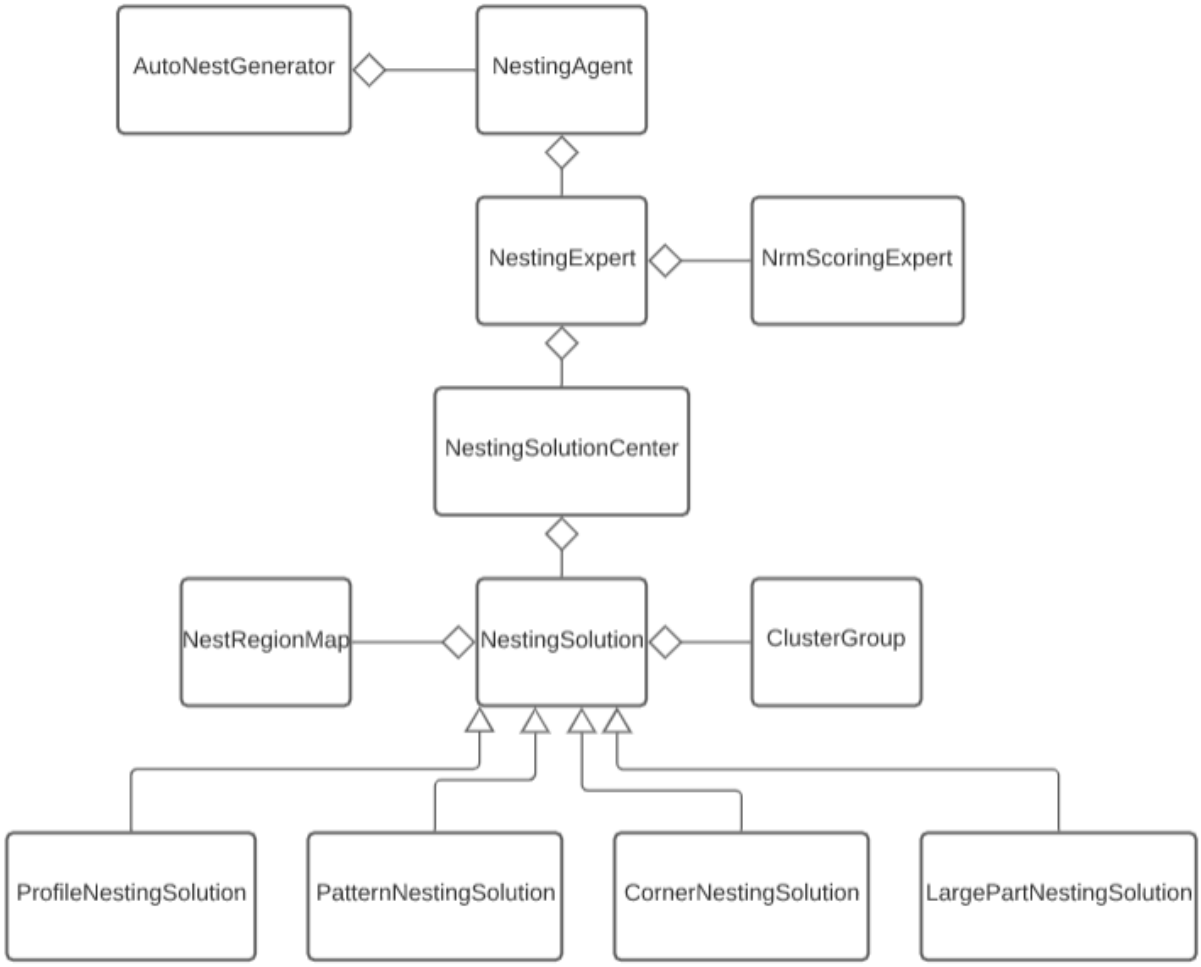
1. Based on the remaining parts and space, generate a set of proposed NestingSolutions, each of which is comprised of:
    a. A set of parts to nest
    b. A nesting algorithm (i.e. pattern nesting, profile nesting, rectangular optimization, etc.)
    c. A region of the plate to nest in

   d. (Note: this nesting solution will generally not complete the nest….it is only a single step towards doing so.)
2. Score these proposed NestingSolutions based on:
   a. Predicted utilization
   b. The set of parts that will be nested
   c. The set of parts that will still remain to be nested (larger parts are more important to nest than smaller parts)
   d. The set of parts that will no longer be able to be nested after this step
   e. The regions of the nest that will remain to be nested.
3. Sort by score and try the best *n* NestingSolutions.
4. Now that you have *n* partial nests, choose the best one based on:
   a. Utilization
   b. Which parts were nested
   c. Which parts still remain
   d. Which parts no longer fit on the nest
   e. The remaining available space
   f. (Note: It actually chooses the best *n* partial nests and looks ahead a certain number of steps to see which leads to the best result, but don't worry about that for now.)
5. Repeat steps 1-4 until you run out of parts or you can't fit any more parts on your plate.

5.2 A little bit louder now – with pictures

The previous section gave a very brief explanation of the approach. This section will go into a little more detail and, better yet, it has pictures. Here is a description of some important abstractions used in the approach.

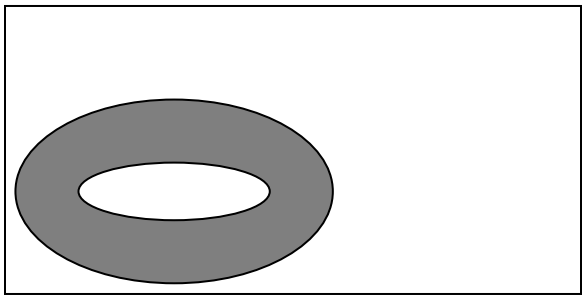
5.2.1 Architecture



5.2.2 NestRegionMap

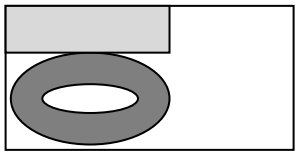
A NestRegionMap is simply a rectangular region on the nest. If you have an empty rectangular plate, the only available NestRegionMap is the region of the plate. But once some parts are nested, things get more interesting.

For example, let's place a part on the nest.

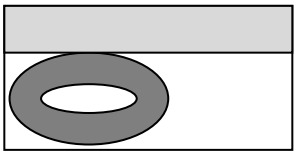


Now that we have a part nested, we can create eight NestRegionMaps as prospective regions for nesting more parts. Four of these are empty regions and four of them contain the nested parts and are considered fill regions. Here are the eight NestRegionMaps (shaded):

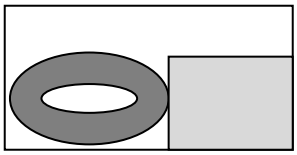
Empty NestRegionMaps:



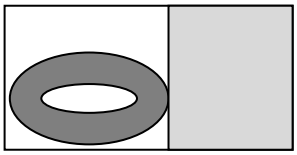
TopRegion



FullTopRegion

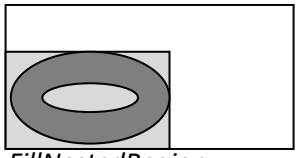


SideRegion

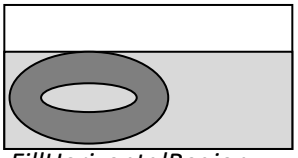


FullSideRegion

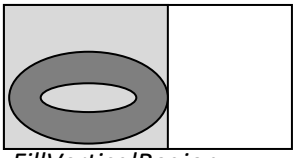
Fill NestRegionMaps:



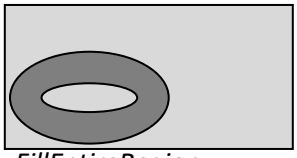
FillNestedRegion



FillHorizontalRegion



FillVerticalRegion



FillEntireRegion

So after each step where one or more parts are nested, we generate these eight NestRegionMaps as possible regions to attack in the next step. Because these are rectangular regions, it is not hard to see that this approach can be (and is!) applied recursively to all of the Empty NestRegionMaps. For example, nesting in the TopRegion is handled similarly to nesting on an empty plate, although scores are adjusted differently. Of course, the TopRegion is only above the nested parts when the nesting initialization point is at the bottom of the nest. If the nesting initialization point is at the top of the nest, TopRegion and Top Full Region will actually be below the nested parts. However, for simplicity, we always refer to these regions as TopRegion and Top Full Region.

5.2.3 NestingSolution

A NestingSolution nests parts in a NestRegionMap, but that is only one useful thing it does. Here is a list of things the nesting solution does (once it is provided with a NestRegionMap and a set of parts that still need to be nested):

1. It generates a list of parts that it will attempt to nest. This may be the entire list of remaining parts, as with ProfileNestingSolution, or only one part, as with PatternNestingSolution. The parts may be all rectangles, as with RectangularOptimizationNestingSolution, or one or more large parts, as with LargePartNestingSolution.
2. It generates a score based on the characteristics of the NestRegionMap and the parts it will nest. This score is then used, along with other factors, to determine which NestingSolutions to try.
3. It provides a nesting algorithm to nest parts in the NestRegionMap.

NestingSolutions offer competing bids for nesting work. They offer to nest a subset of the remaining parts in a subset of the remaining region and provide a score to tell you how good a job they expect to do (i.e. an estimate.)

Each NestingSolution is the ONLY abstraction that knows anything about itself. By this I mean, only the NestingSolution knows:

1. What sort of parts it can nest.
2. How it will nest them.
3. How well it is likely to nest a given part set in a given region.

All any other abstraction knows about a given NestingSolution is what the NestingSolution says about itself. This means that no other abstraction can say, "Aha, this looks like a job for Pattern Nesting!" because it doesn't even know what Pattern Nesting is, much less what is or is not a good situation for Pattern Nesting. Only the PatternNestingSolution knows this. This has two implications:

1. The knowledge of which nesting approaches are likely to work well in a given situation is distributed among the NestingSolutions themselves.
2. Any number of NestingSolutions can be plugged in to IntelliNest without having to modify the approach one iota, allowing for continuous improvement of the system.

5.2.4 NestingSolutionCenter

The NestingSolutionCenter generates NestingSolutions for every NestRegionMap available for the current step. For example, if you have eight types of NestingSolutions and eight NestRegionMaps, the NestingSolutionCenter will generate up to sixty four NestingSolutions, each of which is a bid to nest certain parts in a given region. In reality, less than sixty four NestingSolutions would be generated. Certain NestingSolutions require an empty region (i.e. Pattern Nesting), others require rectangular parts (i.e. rectangular optimization) and, others, large parts (LargePartNesting.) Some require an already nested region that is ready to fill. Also, there are times when you will have less than eight available NestRegionMaps to nest in. So you can see that the number of NestingSolutions generated at each step will always be less than *number of solutions * number of nest region maps.* Still, there will always be some number of NestingSolutions to pick from at each decision point and since IntelliNest was designed for the continuous addition of new NestingSolutions, there will be more and more to pick from as time goes on. Since it would take too long to try all of these for each step, there must be a way to select only the most promising approaches to actually execute.

5.2.5 NestingExpert

The NestingExpert is the central decision maker for the system. It directs the NestingSolutionCenter to generate all of the available NestingSolutions for the current situation. Then it decides what to try. How does it do this? It adjusts the score generated by each NestingSolution based on:

1. How well the NestingSolution says it will do.

2. The importance of the parts the NestingSolution says it will nest. (Larger parts are more important than smaller parts.)
3. The importance of the remaining parts (parts that will not be nested by the NestingSolution but will still fit elsewhere on the nest.)
4. The importance of the excluded parts (parts that will not be nested by the NestingSolution and will not fit elsewhere on the nest.)
5. The importance of the NestRegionMap used (i.e. if there are not enough parts to fill the entire nest, nesting in the TopRegion is preferable to nesting in the Full Top region.)
6. How well the remaining parts are likely to nest together.

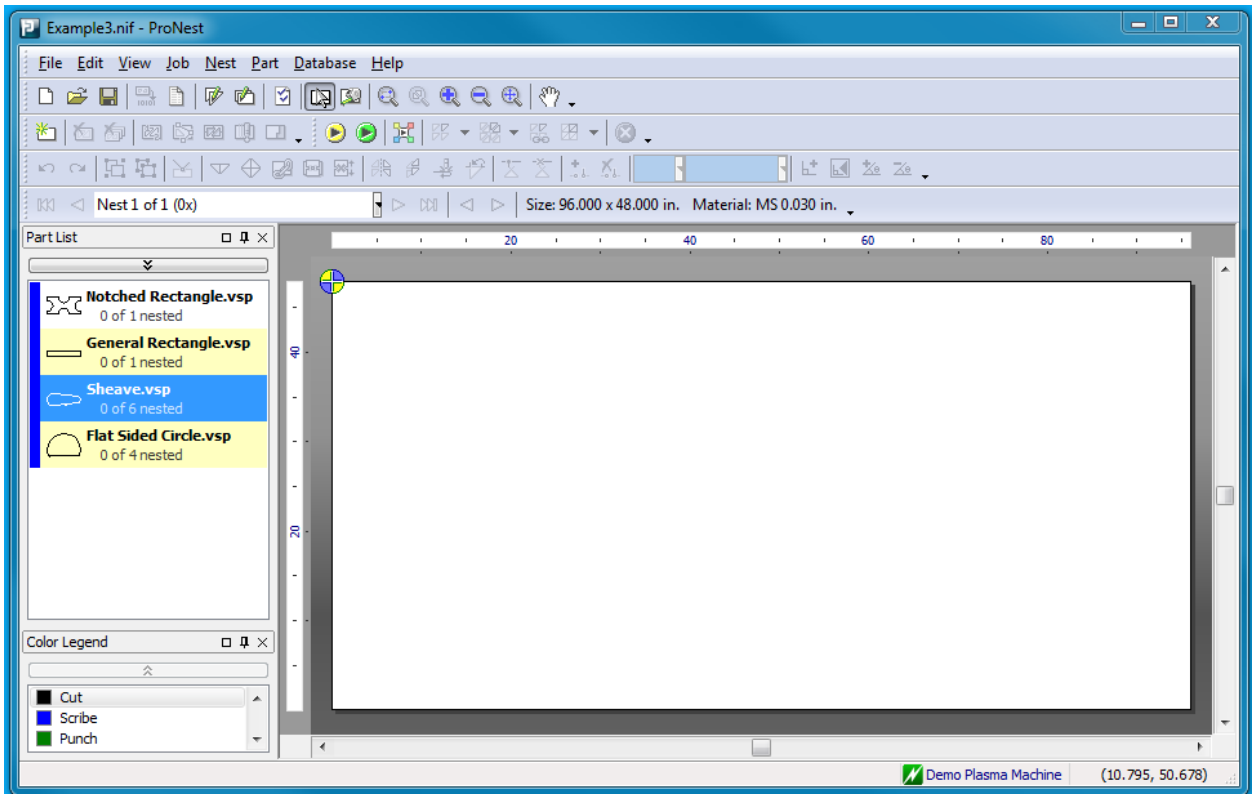
You can think of this adjusted score as a "big picture" score for each NestingSolution. In other words, the NestingSolution is like Gordon "Greed is good" Gekko – it just wants to get the best result it can in this one step. It doesn't know or care what happens after that. The NestingExpert is like John "Beautiful Mind" Nash, who wants to do what's best in this step but also leave what's best for all of the remaining steps. So the NestingSolution is just looking at what *it* will nest in the current step while the NestingExpert is looking at things in the context of the entire nest.

Once each NestingSolution has an adjusted score, the NestingSolutions are resorted and one or more of them is nested. The NestingExpert scores each of the partial nests. (There is a look-ahead component that allows each of the partial nests to be further nested some number of steps but we'll ignore that for now.) How is a partial nest scored? By looking at:

1. The nested utilization.
2. The importance of the nested parts.
3. The importance of the remaining parts.
4. The importance of the excluded parts.
5. How well the remaining parts are likely to nest together in the remaining space.

5.3 A little bit louder now –an example

To illustrate the approach, we will use a simple example of nesting one plate as follows. (This is an entirely fictitious example intended only to aid in understanding how IntelliNest works, so don't look at the scores, etc. too closely. I made them up.)



We will use a single 96 x 48 inch plate, four parts…

Part	Size	Quantity
Notched Rectangle	60 x 30	1
General Rectangle	94 x 15	1
Sheave	21 x 8	6
Flat Sided Circle	13 x 9	4

…and four NestingSolutions.

Name	Part Set	NestRegionMap	Comments
Profile Nesting	Any parts	Empty and Fill regions	This is our general purpose profile nesting algorithm. It works for all parts sets and is the only solution available to fill regions with parts already nested in them.
Pattern Nesting	Single part with quantity > 1	Empty regions only	This algorithm arranges multiple instances of a single part in a pattern in an empty region.
Large Part Nesting	One or more large parts	Empty regions only	This algorithm orients the largest part or parts in an empty region to maximize the remaining available space.

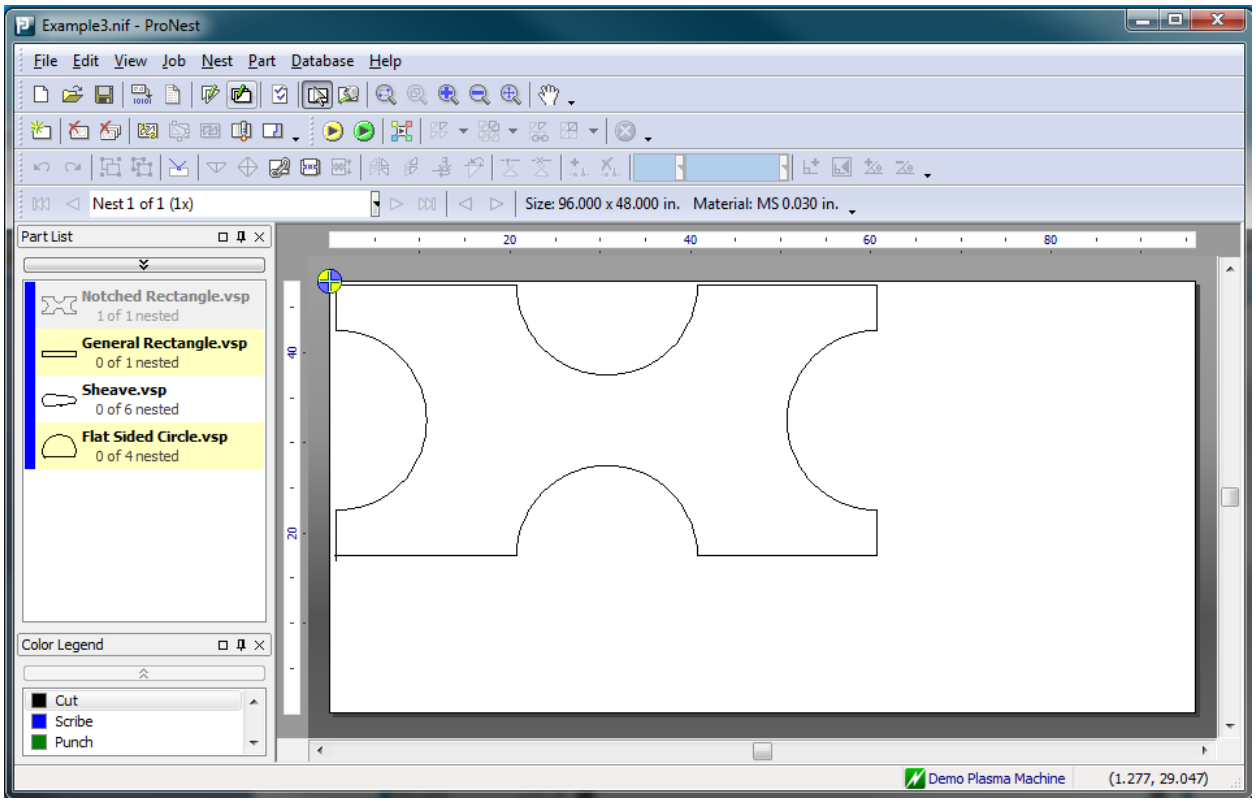
Rectangular Optimization	Large and medium rectangular parts	Empty regions only	This algorithm places 4 to 8 rectangular parts in an empty region.

Step 1:

We generate the following NestingSolutions for the only NestRegionMap that exists so far, the Full Region.

NestRegionMap	Solution	Parts	Score	Adjusted Score	Comments
Full Region	Large Part Nesting	Notched Rectangle	100	90	Special case to address the most important parts (but doesn't promise to nest everything.)
	Pattern Nesting	Sheave	75	50	Claims it will do a better than average job but doesn't address the two most important (largest) parts.
	Profile Nesting	Notched Rectangle General Rectangle Sheave Flat Sided Circle	50	75	Claims it will do an average job but addresses all of the parts.
	Rectangular Optimization	(none)	0	0	There is only one rectangle so the solution generates a zero score.

Four NestingSolutions were generated for the NestRegionMap by the NestingSolutionCenter. Each NestingSolution generated its own score, listed in the **Score** column. The NestingExpert then adjusted the score, listed in the **Adjusted Score** column, and resorted. For this example, we will only choose one NestingSolution for each step. For this step the winner is LargePartNesting, so the NestingExpert directs it to execute its nest. This is the result.



Step 2:

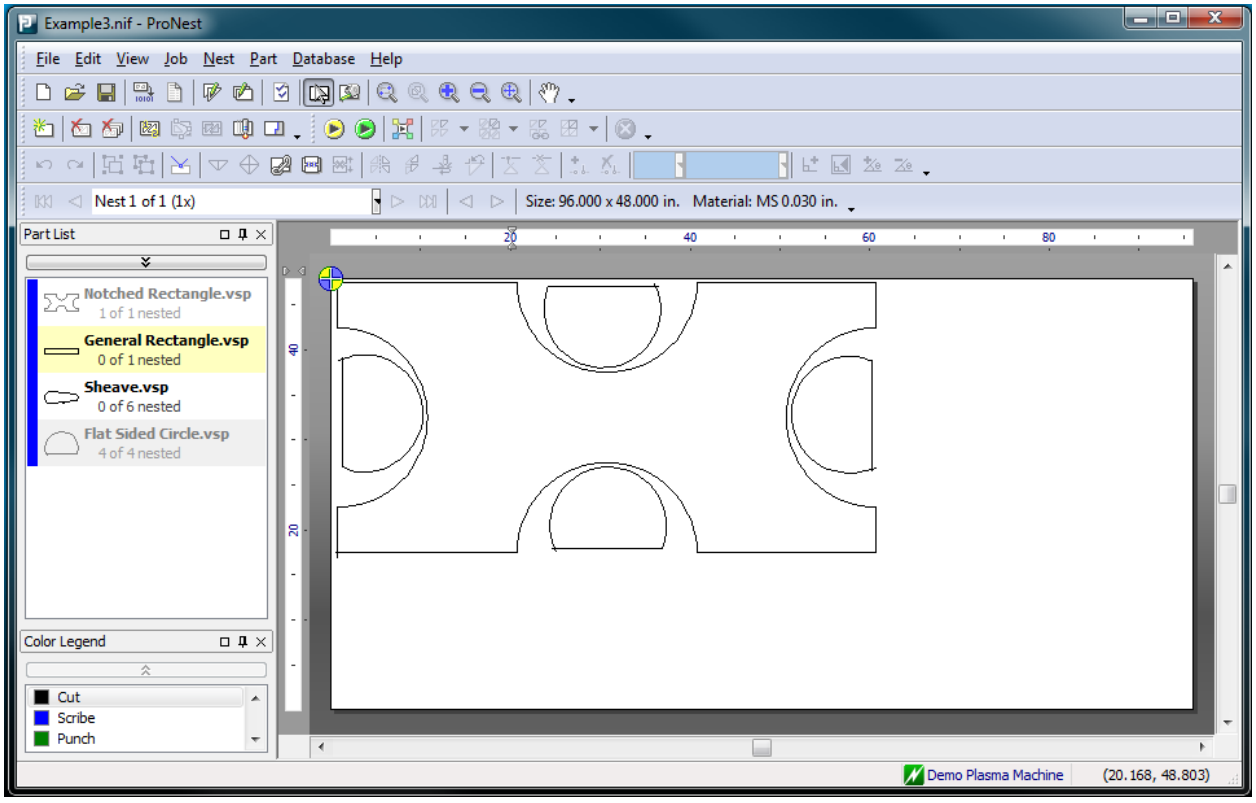
Now it gets interesting because we can generate all eight NestRegionMaps from the partial nest. Here they are, along with the NestingSolutions generated for each.

NestRegionMap	Solution	Parts	Score	Adjusted Score	Comments
Top Region	Pattern Nesting	Sheave	75	50	Does not nest the most important part.
	Profile Nesting	Sheave Flat Sided Circle	50	25	Does not nest the most important part.
Full Top Region	Large Part Nesting	General Rectangle	100	75	Nests the most important part but causes an L-shaped nested area.
	Pattern Nesting	Sheave	75	25	Not only does not nest the most important part, it actually excludes it from being nested.

	Profile Nesting	General Rectangle Sheave Flat Sided Circle	50	50	Nests the most important part but causes an L-shaped nested area.
Side Region	Pattern Nesting	Sheave	75	50	Does not nest the most important part.
	Profile Nesting	Sheave Flat Sided Circle	50	25	Does not nest the most important part.
Full Side Region	Pattern Nesting	Sheave	75	25	Not only does not nest the most important part, it actually excludes it from being nested.
	Profile Nesting	Sheave Flat Sided Circle	50	0	Not only does not nest the most important part, it actually excludes it from being nested.
Fill Nested Region	Profile Nesting	Flat Sided Circle	50	80	Doesn't nest the most important part but it doesn't exclude it and it fills in the most important region. The parts are also a perfect match for the space.
Fill Horizontal Region	Profile Nesting	Sheave Flat Sided Circle	50	25	Doesn't nest the most important part and extends the nest horizontally. Also, the parts are not designed to nest well together.
Fill Vertical Region	Profile Nesting	Sheave Flat Sided Circle	50	25	Doesn't nest the most important part and actually excludes it. Also, the parts are not designed to nest well together.
Fill Entire Region	Profile Nesting	General Rectangle	50	50	Nests everything but filling the entire plate this early with profile nesting is sort of like punting. Also, the parts are not designed to nest well together.

	Sheave Flat Sided Circle

Thirteen NestingSolutions were generated in this step. The adjusted scores mostly reflect whether a solution included, left alone or excluded (i.e. ensured that it would not fit on the nest) the most important remaining part (General Rectangle) and adjustment for which NestRegionMap was used. In general, extending the nest vertically is preferred to extending it horizontally and using a full top/side is discouraged unless it adds a lot of value. So the winning NestingSolution in this step is Profile Nesting in the FillNestedRegion. Here is the result.



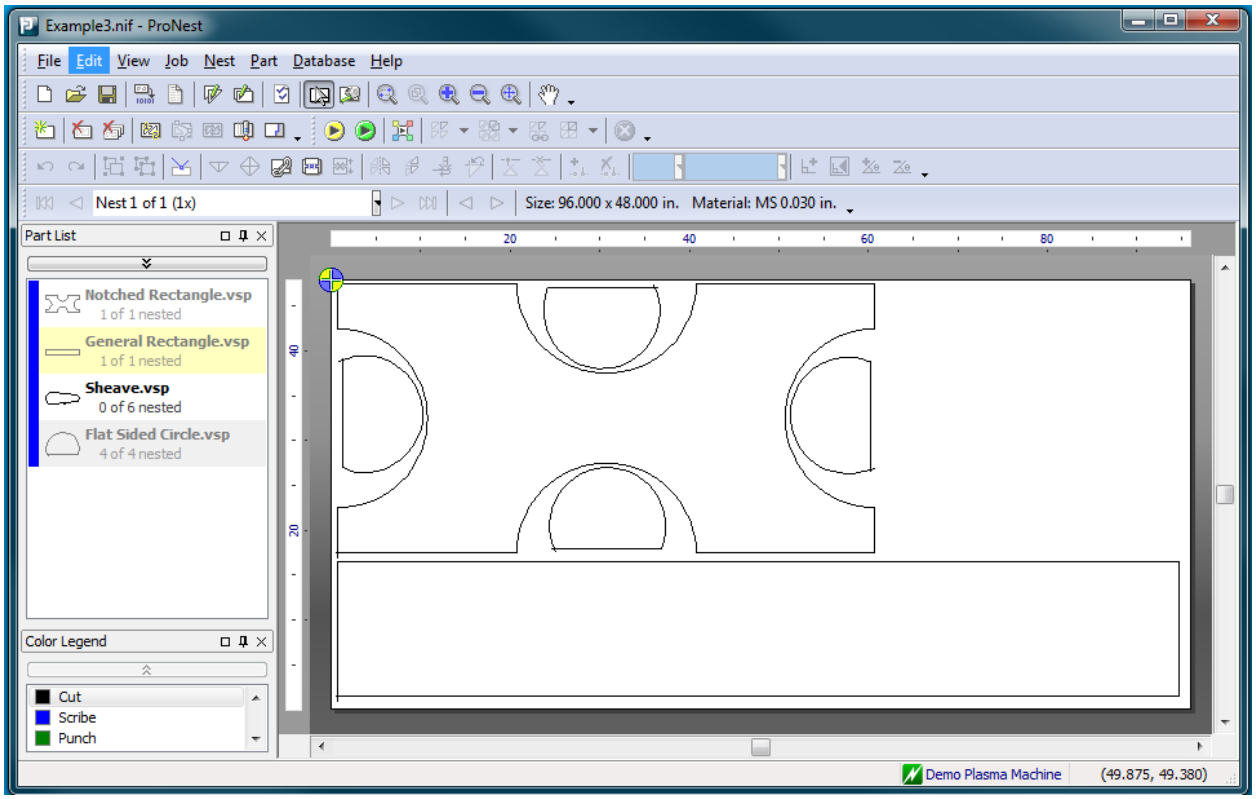
Step 3:

Now two parts have been nested and the original FillNestedRegion has been completed. The remaining NestingSolutions are revised and rescored.

NestRegionMap	Solution	Parts	Score	Adjusted Score	Comments
Top Region	Pattern Nesting	Sheave	75	50	Does not nest the most important part.

	Profile Nesting	Sheave	50	25	Does not nest the most important part.
Full Top Region	Large Part Nesting	General Rectangle	100	75	Nests the most important part but causes an L-shaped nested area.
	Pattern Nesting	Sheave	75	25	Not only does not nest the most important part, it actually excludes it from being nested.
	Profile Nesting	General Rectangle Sheave	50	50	Nests the most important part but causes an L-shaped nested area.
Side Region	Pattern Nesting	Sheave	75	50	Does not nest the most important part.
	Profile Nesting	Sheave	50	25	Does not nest the most important part.
Full Side Region	Pattern Nesting	Sheave	75	25	Not only does not nest the most important part, it actually excludes it from being nested.
	Profile Nesting	Sheave	50	0	Not only does not nest the most important part, it actually excludes it from being nested.
Fill Horizontal Region	Profile Nesting	Sheave	50	25	Doesn't nest the most important part and extends the nest horizontally.
Fill Vertical Region	Profile Nesting	Sheave	50	25	Doesn't nest the most important part and actually excludes it.
Fill Entire Region	Profile Nesting	General Rectangle Sheave	50	50	Nests everything but filling the entire plate this early with profile nesting is sort of like punting.

Even though it creates an L-shaped nested area, nesting the General Rectangle in the FullTopRegion gets the next most important part nested very efficiently and doesn't exclude any other parts from being nested. Here is the result.

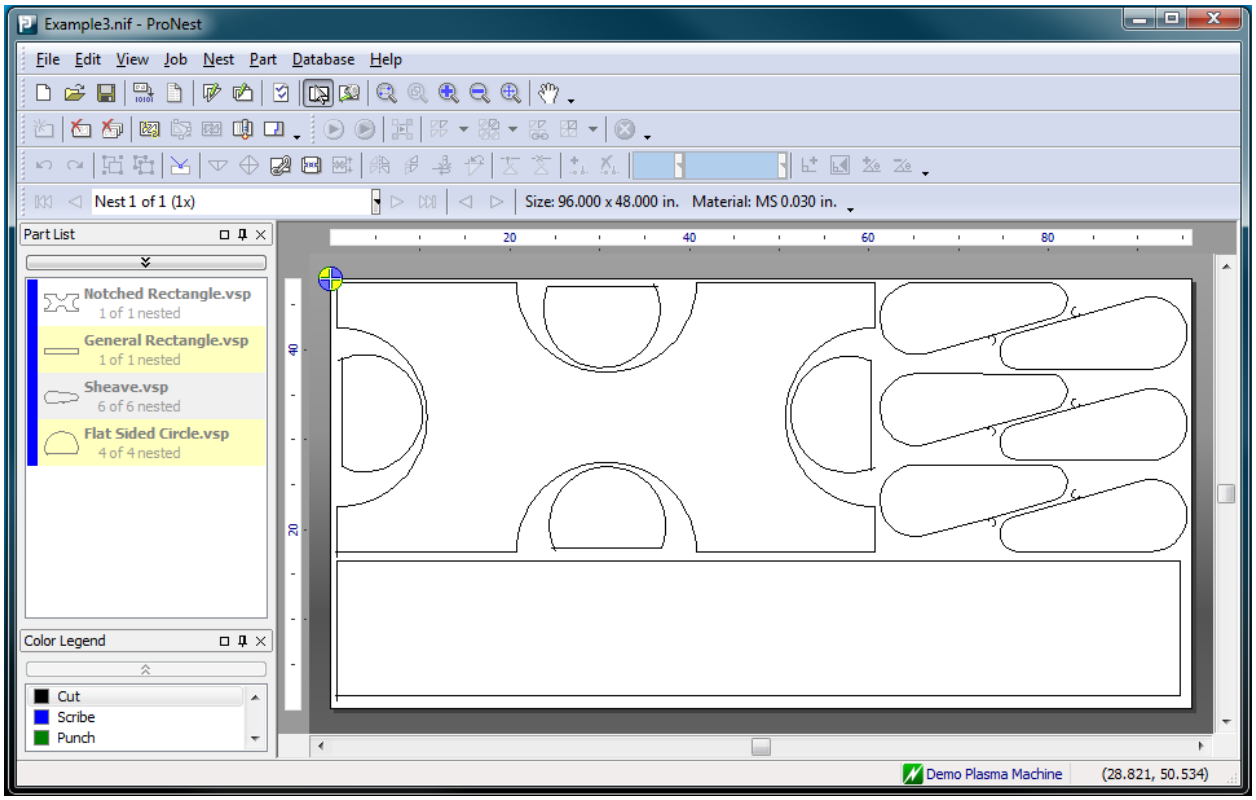


Step 4:

There is only one part left but there are still several ways to nest it. Here are the remaining NestingSolutions.

NestRegionMap	Solution	Parts	Score	Adjusted Score	Comments
Side Region	Pattern Nesting	Sheave	75	75	Nothing to adjust.
Fill Entire Region	Profile Nesting	Sheave	50	50	Filling the entire nest is preferable to just nesting on the side. However, there isn't much space available in the nested region so it's a wash.

Pattern Nesting in the SideRegion wins.



This was a trivial example just to give an idea of the process IntelliNest uses to produce a nest. The recursive nature of the solution was not illustrated, but it is easy enough (and important enough) to explain. Say, for example, the NestingExpert accepts the PatternNestingSolution's bid to nest in the TopRegion. Once Pattern Nesting has completed in the Top Region, the NestingExpert will direct the NestingSolutionCenter to generate the eight remaining NestRegionMaps from the TopRegion, and proceed as it normally does to nest the entire top region before returning to the previous frame of reference. Likewise, each of the eight remaining NestRegionMaps generated from the TopRegion will similarly be nested, and so on. So the problem is continually broken down in to smaller and smaller pieces and each piece is attacked in exactly the same way with every available NestingSolution being brought to bear on every available inch of the nest.

5.4 What's so great about IntelliNest?

The approach used in IntelliNest has a number of advantages:

1. It uses numerous nesting approaches to produce a nest rather than just a single approach. This is at least two levels better than our old nesting, which applies the same automatic nesting strategy to all nests in the job. Not only does IntelliNest apply different nesting approaches to each nest, it applies them to each decision point in each nest. In this way, it is able use the best of all nesting approaches when they are of the most benefit.
2. It automatically determines which nesting approach to use when based on the available parts and space, relieving the user of this responsibility.

3. It is highly recursive, allowing the same NestingSolutions to be applied to ever smaller parts of the problem. One example of this is the LargePartNestingSolution, which is designed to orient one or more large parts in an empty region to maximize the remaining available area for nesting. But what is large is relative to the region size; medium parts – and even small parts - become large parts as the problem is further decomposed into smaller and smaller regions, allowing this same solution to be applied to parts of all sizes in smaller areas of the nest.
4. It weighs the consequences of nesting particular parts in particular regions at each step along the way, balancing the need to nest effectively in the current step with the need to nest effectively in succeeding steps.
5. It allows for extensive special case handling. If we identify a special case we need to improve on we can simply design a new NestingSolution for this case and add it to the system, without having to change a single line of existing code. This new NestingSolution would only be executed when the special case was recognized and so would not affect any of the existing solutions.
6. It allows for the easy addition of new approaches, which will only be used when they are likely to outperform the other approaches. Part of what makes this possible is that the actual nesting expertise is not centrally located but is distributed among the NestingSolutions.
7. It allows for any number of NestingSolutions to be executed at any step and will pick the best of these to proceed. It determines which is best by looking not only at what has been nested, but also at what remains to be nested.
8. It will look ahead any number of steps to determine which solution actually leads to the best result. Of course, this is subject to time performance constraints.

5.5 Are we there yet?

No, and don't *make* me pull over. The success of IntelliNest depends upon its ability to choose the right nesting approach for the right parts in the right region of the nest at the right time. This is how humans nest – by choosing one or more parts, arranging them on the nest and then evaluating the remaining parts and space and deciding what to do next. For humans, these types of decisions are second nature. For computers? Not so much.

6 Keeping Score

6.1 Choose well

The IntelliNest approach is all about decision making. It is a flexible approach that can choose between different nesting approaches, different part sets and different regions of the nest. Instead of always doing the same thing in the same way, it chooses one of many possible paths based on the current state of the nest. Since it is highly extensible, the number of possible paths will continue to grow and so making the right choices will remain the central issue for this approach.

6.2 Keeping Score

There are three main scores the NestingExpert uses to decide what to do next.

Score	Generated By	Comments
Solution Score	NestingSolution	This score represents how effectively the NestingSolution is likely to nest a given part set in a NestRegionMap.
Adjusted Solution Score	NestingExpert	This score is a weighted adjustment of the Solution Score that also considers: • Which NestRegionMap is used • How well the remaining regions and part sets match up • The excluded part set
Partial Nest Score	NestingExpert	This score is used to compare multiple partial nests by considering: • The utilization of the nested parts • The remaining "usable" space (i.e. space that can be used by remaining parts) • How well the remaining parts are likely to nest together • How well the remaining parts are likely to nest in the remaining space • Note: This has not been implemented yet.

The challenge here is that, for IntelliNest to work, we will have to *predict* how well parts are going to nest with other parts in a particular space. What we have is a set of characteristics for the parts and the space. What we need is a prediction based on those characteristics of how effectively we can nest the parts in that space. What is the best way to accomplish this?

6.3 What about Artificial Neural Networks

We think we may be able to generate these scores through the use of Artificial Neural Networks, which are learning systems based on human biology. Through the use of training data, neural networks can learn to predict a result given a set of previously unseen input values. For us, this means that it may be possible to train a neural network to predict how well parts are likely to nest if we are able to supply it all of the relevant characteristics of the parts and the available space and enough of the right training data. This changes our task from developing formulas that estimate nested utilization to identifying the pertinent characteristics, gathering the required amount of training data and designing the proper type of neural network to accomplish our objective. Will this approach work?

Sadly, it did not. The shortcomings of IntelliNest are discussed in the final section.

7 Scores, Information, Etc.

This is where the friendly, high level overview ends and the serious, tedious detail begins. If you thought this has already been serious and tedious then you should run away and not look back. For the rest of us, here is all of the information we currently use to make decisions, broken down into three groups:

1. Major Scores – There are three highest level scores that are used to decide what to do next in this automatic nesting approach.
2. Supporting Scores- There are a number of supporting scores used to generate the three major scores. These are calculated using weighted formulas.
3. Other Useful Information – This is just data that is collected and calculated from our data model.

For each score, we will provide detail in the following format:

Name	The name of the score
Description	What the score represents
Purpose	Why the score is necessary and how it is used
Generated By	Which abstraction is responsible for generating the score
Used By	Who uses the score and why
Inputs	The information used to calculate the score
Outputs	The range of possible outputs and what they mean
Calculation Method	How the score is calculated
Comments	Miscellaneous information

7.1 Major Scores

There are three major scores used by IntelliNest to determine what to do next: Solution Score, Adjusted Solution Score and Partial Nest Score. Each of these scores uses supporting scores and information and probably depends on non-obvious relationships between many factors.

7.1.1 Nesting SolutionScore

A Nesting SolutionScore represents a prediction of how well a given NestingSolution will nest a set of parts in a NestRegionMap. There are a number of different nesting solutions, each of which generates its own score, using inputs and methods unique to itself. These scores are then used by the NestingExpert to determine which solutions to try at each point in the nesting process. Here is a detailed description of the scores for all existing NestingSolutions, understanding, of course, that many new NestingSolutions will be added to IntelliNest and each of these will provide its own score.

Although each NestingSolution has its own way of generating its score, they have many aspects in common:

Name	Any NestingSolutionScore

Description	This score represents a prediction of how well this nesting solution will nest a given part set in a given nest region map.
Purpose	This score is used to determine which nesting solutions to attempt at a given point in automatic nesting.
Generated By	The nesting solution itself.
Used By	NestingExpert – to determine which nesting solutions to try
Inputs	(vary by nesting solution)
Outputs	• 100 – a special case has been recognized • 75 – this nesting solution predicts it will outperform profile nesting • 50 – this score is reserved for profile nesting • 0 – will not outperform profile nesting so don't bother with it
Calculation Method	(varies by nesting solution)
Comments	The ProfileNestingSolution score will always be 50, and the other scores just denote predicted results compared to profile nesting.

7.1.2 AdjustedSolutionScore

To create the adjusted solution score, the nesting expert takes the solution score generated by a NestingSolution and adjusts it by combining:

- NestingSolutionScore * NestingSolutionScoreWeight
- AdjustedNestRegionMapScore * NestRegionMapWeight
- AdjustedPartSetScore * PartSetScoreWeight

The three weights are adjusted such that such that the NestRegionMapWeight and PartSetScoreWeight are halved each time you enter a new subregion of the problem (i.e. at each new level of recursion.) This makes the nested utilization of parts more and more important and selection of parts and nesting region less and less important the farther you move away from the top level of the nest. For example, when you first start to nest a blank plate, you *could* get good utilization by nesting a block of the smallest part in your list but then you might not be able to fit the largest part on the nest at all, so which part you nest first is very important. However, when you are simply trying to fill a small sub-region in the center of the nest, you probably just want to get the best utilization possible. That is what the weight adjustment tries to accomplish.

AdjustedPartSetScore is a combination of:

- EstimatedNestedPartSetScore – representing the set of parts to be nested
- EstimatedRemainingPartSetScore – representing the set of parts that will not be nested but are still available to nest elsewhere
- EstimatedExcludedPartSetScore – representing the set of parts that will no longer be nestable if this solution is executed

The AdjustedNestRegionMapScore is a combination of:

- NestedNestRegionMapScore – representing the nest region map used in the solution
- ExcludedNestRegionMapScore – representing the average score of all NestRegionMaps what would be excluded if the NestRegionMap in the solution was chosen

Name	AdjustedSolutionScore
Description	This score is an adjustment of a NestingSolution score in the context of the part set and the nest region map.
Purpose	This score is used to determine which nesting solutions to attempt at a given point in automatic nesting.
Generated By	NestingExpert
Used By	NestingExpert – to determine which nesting solutions to try
Inputs	• NestingSolutionScore • AdjustedPartSetScore • AdjustedNestRegionMapScore
Outputs	• 0..100
Calculation Method	NestingSolutionScore * NestingSolutionWeight + AdjustedPartSetScore * PartSetWeight + AdjustedNestRegionMapScore * NestRegionMapWeight
Comments	• AdjustedPartSetScore is calculated from the scores of NestedPartSet, RemainingPartSet and ExcludedPartSet • AdjustedNestRegionMapScore considers the NestRegionMapScore and the ExcludedNestRegionMapScore. • The PartSetWeight and NestRegionMapWeight are halved at each new level of recursion to decrease their effect

7.1.3 PartialNestScore (not implemented)

The PartialNestScore is used to compare two or more partially nested plates to see which will lead to the best nest. This is an important score because it allows IntelliNest to try more than one NestingSolution at a given point and then choose the best one. How to determine which is the best, though? In early versions of this approach IntelliNest simply used nested utilization, but that was like comparing apples to oranges, especially since it was often comparing two nests in which:

1. Different parts had been nested
2. Different NestRegionMaps had been nested

What is needed here is to consider:

1. Which parts have been nested
2. How efficiently they have been nested
3. Which parts remain to be nested
4. How compatible the remaining parts are with each other
5. How compatible the remaining parts are with the remaining space
6. Which parts are now excluded from being nested

PartialNestScore is a combination of:

1. NestedPartsScore – representing which parts have been nested and how efficiently they have been nested
2. RemainingPartsSpaceScore – representing what remains to be nested and how compatible the remaining parts are with each other and with the remaining space
3. ExcludedPartsScore – representing the part that can no longer be nested

NestedPartsScore is a combination of:

1. UsableSpaceUtilization – this is essentially NestedPartArea / (PlateArea – UsableRemainingSpace)
2. NestedPartSetScore – representing the nested parts

RemainingPartsSpaceScore is a combination of:

1. RemainingUsableSpace – the ratio of usable space to unnested space
2. RemainingPartSetScore – representing the parts that still remain to be nested
3. RemainingPartsNestingCompatibilityScore – how well the remaining parts are likely to nest together
4. RemainingPartsSpaceNestingCompatibilityScore – how well the remaining parts are likely to nest in the remaining space

Name	PartialNestScore
Description	This score represents a partial nest in the context of what has been nested, what remains to be nested and what is excluded from being nested.

Purpose	This score is used to compare two or more partial nests. It allows the system to try multiple solutions at any given point and choose the one that will lead to the best nest.
Generated By	NestingExpert
Used By	NestingExpert – to determine which nested solution will lead to the best nest
Inputs	• NestedPartsScore • RemainingPartsSpaceScore • ExcludedPartsScore
Outputs	• 0..100
Calculation Method	NestedPartsScore * w1 + RemainingPartsSpaceScore * w2 + ExcludedPartsScore * w3
Comments	• NestedPartsScore considers UsableSpaceUtilization and NestedPartSetScore • RemainingPartsSpaceScore considers RemainingUsableSpace, RemainingPartSetScore, RemainingPartsNestingCompatibilityScore and RemainingPartsSpaceNestingCompatibilityScore • This is not implemented or used yet (although some of the supporting information is being generated)

7.2 Supporting Scores

Supporting Scores are used in the generation of the three Major Scores (NestingSolutionScore, AdjustedNestingSolutionScore and PartialNestScore.) We may also be able to use neural networks to generate some of these scores.

7.2.1 PartSetScore

PartSetScore extends the concept of Part Importance from parts to part sets. This is done by weighting the importance score by the percentage of nested area (or estimated nested area) made up of parts from each part importance category.

Score = (w1 * NestedArea of Important parts + w2 * NestedArea of Average parts + w3 * NestedArea of Filler parts) / Total part area

Area is:

1. estimated area, for unnested parts
2. actual, for nested parts

Name	PartSetScore
Description	This score represents the importance a given part set.
Purpose	This score allows you to compare two or more part sets and determine which of them is best to have nested.
Generated By	MtcPartSetAnalyzer
Used By	NestingExpert – used to generate AdjustedSolutionScore and PartialNestScore.
Inputs	• PartImportanceScore • NestedArea of Important parts • NestedArea of Average parts • NestedArea of Filler parts
Outputs	• 0..100
Calculation Method	(w1 * NestedArea of Important parts + w2 * NestedArea of Average parts + w3 * NestedArea of Filler parts) / Total part area
Comments	• This is not implemented yet • For a nested part set, NestedArea is the actual nested area • For an unnested part set, NestedArea is estimated • How do we estimate the NestedArea of each category of parts for an unnested part set? Neural networks? • Weights must add up to 1.

7.2.2 NestRegionMapScore

For any given situation, choosing the correct NestRegionMap to nest in can lead to a better nest. Here is the table we currently use for generating the NestRegionMapScore:

Approach			Score (available nrmaps only)									Comments
	Will Fill Nest?	Will Fill Region?	FULL	TOP	FULL TOP	SIDE	FULL SIDE	FILL NESTED	FILL HORZ	FILL VERT	FILL ALL	
Empty Region			100									This is an empty region. The only available choice is FULL.
Nested Region				80	40	60	40	90	40	80	50	This is the base case where something has been nested and now everything (except FULL) is available.
	✓			80	60	70	60	90	60	80	50	
NR (no top)						60	40		40	80	50	This is the base case where something has been nested but there were no solutions for top.
	✓					70	60		60	80	50	

Top First							60			90	60	The top has been nested and everything else is available.
	✓						80			90	70	
						70				90	80	The full top has been nested and everything else is available.
	✓					80				90	70	
Side First					70				90		70	The side has been nested and everything else is available.
	✓				80				90		70	
				70					90		80	The full side has been nested and everything else is available.
	✓			80					90		70	
Fill First				90	60	80	60			90	70	
	✓											
Full Side Left							100				50	All that is left is the full side (or fill the entire nest.)
	✓						100				50	
Full Top Left					100						50	All that is left is the full top (or fill the entire nest.)
	✓				100						50	

These scores have been chosen through reflection and testing (also known as trial and error.) There probably is a better way to generate them.

Name	NestRegionMapScore
Description	This score represents the desirability of filling a particular NestRegionMap as compared to other available NestRegionMaps.
Purpose	This score allows you to compare two or more NestRegionMaps and determine the relative desirability of nesting in them.
Generated By	MtcNrmScoringExpert
Used By	NestingExpert – used to generate AdjustedSolutionScore.
Inputs	• NestWillBeFilled • RegionWillBeFilled • List of available NestRegionMaps
Outputs	• 0..100
Calculation Method	See table above.
Comments	• There must be a better way to generate these scores.

7.2.3 AdjustedNestRegionMapScore (not implemented)

This score tries to take into account the NestRegionMaps that would be excluded if this one was chosen. For example, If you choose the TopNestRegionMap, you exclude FullTop, FillVertical and FillAll.

Name	AdjustedNestRegionMapScore
Description	This score represents the NestRegionMapScore in the context of which NestRegionMaps would be excluded if this one was chosen.
Purpose	This score allows you to compare two or more NestRegionMaps and determine the relative desirability of nesting in them.
Generated By	MtcNrmScoringExpert
Used By	NestingExpert – used to generate AdjustedSolutionScore.
Inputs	• NestRegionMapScore • NestRegionMapType
Outputs	• 0..100
Calculation Method	A table, probably.
Comments	• This may already be accounted for by the relative scoring in NestRegionMapScore. I'm not sure.

7.3 Other Useful Information

There is a host of information that we generate and store (or *will* generate and store) about part sets, part categories, parts, profiles, concavities and space. Here is a subset of it.

7.3.1 Part Set Characteristics

7.3.1.1 MaxRegionDimensionRatio

Name	MaxRegionDimensionRatio
Description	The ratio of the largest dimension of any part in the part set to the largest dimension of the space region.
Purpose	To give an idea of how big the max dimension of the largest

	part is compared to the region.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• MaxPartRegionDimension • MaxSpaceRegionDimension
Outputs	• 0..1
Calculation Method	MaxPartRegionDimension / MaxSpaceRegionDimension
Comments	

7.3.1.2 MaxExteriorProfileAreaRatio

Name	MaxExteriorProfileAreaRatio
Description	The ratio of the area of the largest part in the part set to the area of the space.
Purpose	To give an idea of how big the area of the largest part is compared to the area of the space.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• MaxPartArea • SpaceArea
Outputs	• 0..1
Calculation Method	MaxPartArea / SpaceaArea
Comments	

7.3.1.3 UnnestedTotalFootprint

Name	UnnestedTotalFootprintArea

Description	The total footprint area of a part set is the total area of all remaining unnested parts in that part set that could be nested in a given region.
Purpose	To give an idea of how big the area of the largest part is compared to the area of the space.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• UnnestedPartQuantity • PartArea
Outputs	• 0..whatever
Calculation Method	Summation over all unnested parts of (UnnestedPartQuantity * PartArea)
Comments	• This should probably use use some footprint area to limit the total area to what could possibly fit in the region.

7.3.2 Part Characteristics

7.3.2.1 Size

Name	Size
Description	The size of a part relative to a given region.
Purpose	To give an idea of how big a part is compared to a region.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• PartRegionRatio
Outputs	• Very small – (100 parts per region) • Small – (20-100 parts per region) • Medium – (10-20 parts per region) • Large – (2-10 parts per region) • Very large – (1 part per region)

Calculation Method	If RegionRatio < 0.01) PartSize is Very small Else if RegionRatio < 0.05 PartSize is Small Else if RegionRatio < 0.1 PartSize is Medium Else if RegionRatio < 0.5 PartSize is Large Else PartSize is Very large
Comments	

7.3.2.2 Quantity

Name	Quantity
Description	The available quantity of the part that could fit into the available region.
Purpose	To give an idea of how many instances of a part could be nested in a region.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• PartRemainingQuantity • PartFootprintQuantity
Outputs	• 0..whatever
Calculation Method	Min(PartRemainingQuantity, PartFootprintQuantity)
Comments	• We could use PartInterlockingScore to better estimate how many parts could be nested in the region.

7.3.2.3 ExteriorProfileArea

Name	ExteriorProfileArea

Description	The area of the exterior profile of the part.
Purpose	Since the entire exterior profile needs to fit inside of some space when nesting, ExteriorProfileArea can be used as a quick check to see if there is enough available space remaining to nest the part.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• The exterior profile of the part
Outputs	• 0..whatever
Calculation Method	The way we've always done it.
Comments	

7.3.2.4 ExteriorProfileRatio

Name	ExteriorProfileAreaRatio
Description	The ratio of the area of the exterior profile to the area of the minimum bounding region of the exterior profile.
Purpose	To give us some idea of how much space inside of the part's bounding region will be available for nesting.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• The exterior profile of the part
Outputs	• 0..whatever
Calculation Method	PartExteriorProfileArea/PartMinRegionArea
Comments	

7.3.2.5 TotalArea

Name	TotalArea
Description	The total area for all instances of the part.
Purpose	This is used in other various ratios.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• The area of the part
Outputs	• 0..whatever
Calculation Method	PartArea * PartQuantity
Comments	

7.3.2.6 TotalInteriorProfileArea

Name	TotalInteriorProfileArea
Description	The combined area of all interior profiles of the part.
Purpose	This is used in other various ratios.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• The areas of each of the interior profiles belonging to the part
Outputs	• 0..whatever
Calculation Method	Summation of the area of each interior profile belonging to the part.
Comments	

7.3.2.7 TotalConcavityArea

The sum of the areas of all concavities of the part.

Name	TotalConcavityArea
Description	The combined area of all concavities of the exterior profile of a part.
Purpose	This is used in other various ratios.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• The areas of each of the concavities of the exterior profile
Outputs	• 0..whatever
Calculation Method	Summation of the area of each concavity of the exterior profile
Comments	

7.3.2.8 Slopiness

This is a part characteristic that refers to gradual decrease in width of a part over its length. Slopey parts tend to combine nicely into rectangular pairs. Slopey parts can “slide” up one another to effectively use a given region dimension. Slopiness refers to the angle between a pair of lines. Here are some examples of slopey parts:

- Certain shapes: non-rectangular trapezoid
- some arc sections
- parts where linear motions on opposite sides of the part make a certain angle
- parts where the general tendency of one side is slopey
- parts that, when rotated to maximize their length, have a gradual decrease in width towards one end or the other
- parts that, when paired with another slopey part, create a rectangular pair
- Triangles

Name	Slopiness
Description	This is a part characteristic that refers to gradual decrease in width of a part over its length.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable and which parts should be used.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MtcNestShape
Outputs	• 0..100
Calculation Method	Formula
Comments	

7.3.2.9 Edginess

Edgy parts have one or more sides that could be placed up against the region edge. Obviously, any shape containing a straight motion along an entire side is edgy. A circle is not. Between these extremes parts have various degrees of edginess. In general, parts are considered edgy if you can place them around the edge of a region and still leave good usable space. For large parts, this may be related to rotatability. Some examples:

- Rectangles
- “Outside” edges of L-bracket
- Parts with long shallow concavities along one edge

Name	Edginess
Description	This represents how well a part could be nested by placing it against a straight edge of the available space.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many

Inputs	• MtcNestShape
Outputs	• 0..100
Calculation Method	Formula
Comments	

7.3.2.10 Righteousness

This refers to the existence of a "corner" of a part that would fit nicely into the corner of a region. Obviously, a part with two straight sides that come together at 90 degrees would have the highest degree of righteousness. But a large radius arc section would also be righteous. Examples:

- Rectangles
- Parts with large arcs that could be rotated to maximize usage of one nesting region dimension
- Notched corner rectangle or clipped corner rectangles would have a lesser degree of righteousness than full rectangles (their convex corners could not fit into the corner of the region without overlapping it, but on a larger scale the "missing" portion of the corner may fit quite well.)

Name	Righteousness
Description	This refers to the existence of a "corner" of a part that would fit nicely into the corner of a region.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MtcNestShape
Outputs	• 0..100
Calculation Method	Formula
Comments	

7.3.2.11 Corneriness

A cornery part is one that leaves good, usable space when placed in a corner. Corneriness is related to righteousness. I think a part has to be righteous before it can be cornery. Some examples of cornery parts:

- L-bracket
- Windmill part
- Large semi-circles (see illustration for “Rotatability”)

Some examples of righteous parts which are not cornery:

- Rectangles (may not leave good usable space)

Name	Corneriness
Description	This refers a part that leaves good, usable space when placed in a corner.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MtcNestShape
Outputs	• 0..100
Calculation Method	Formula
Comments	

7.3.2.12 Bigness

Bigness relates the size of the part to the size of the nesting region under consideration. The bigness relation could involve the part area, part region area, the part region dimensions, the longest side dimension or any combination of these. Examples:

- A 50x50 rectangle being nested in a 60x60 region
- A 75” long part in a region that is 100” long

Name	Bigness

Description	Bigness relates the size of the part to the size of the nesting region under consideration.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• PartProfileRatio • PartRegionRatio • SpaceRegion
Outputs	• 0..100
Calculation Method	max(PartProfileRatio, max(RegionRatio, 0.01)) * 100.0))
Comments	

7.3.2.13 Rectangularishness

Rectangularishness relates the exterior geometry of a part to its minimum bounding rectangle, without considering leads. Obviously a rectangle would be 100% rectangular. A circle would not be rectangular. A right triangle would be 50% rectangular, although a right triangle paired with itself may be close to 100% rectangular. Examples:

- A true rectangle would be most rectangularish
- A part or cluster whose exterior boundary approximates a rectangle would be less rectangularish
- A circle would be not at all rectangularish.

Name	Rectangularishness
Description	Rectangularishness relates the exterior geometry of a part to its minimum bounding rectangle
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many

Inputs	• PartExteriorProfileArea • PartMinRegionArea
Outputs	• 0..100
Calculation Method	(PartExteriorProfileArea / PartMinRegionArea * 100.0) + 0.5
Comments	

7.3.2.14 EstimatedFootprintQuantity

Name	EstimatedFootprintQuantity
Description	The estimated number of parts that could block arrayed in a region.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• PartMinNonLeadsRegion • Space region
Outputs	• 0..whatever
Calculation Method	min(PartFootprintQuantity, PartQuantity)
Comments	

7.3.2.15 TotalProfileFootprint

Name	TotalProfileFootprint
Description	The sum of the exterior profile areas of all instances of the part in the footprint.

Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• PartTotalFootprintQuantity • PartExteriorProfileArea
Outputs	• 0..whatever
Calculation Method	PartTotalFootprintQuantity * PartExteriorProfileArea
Comments	

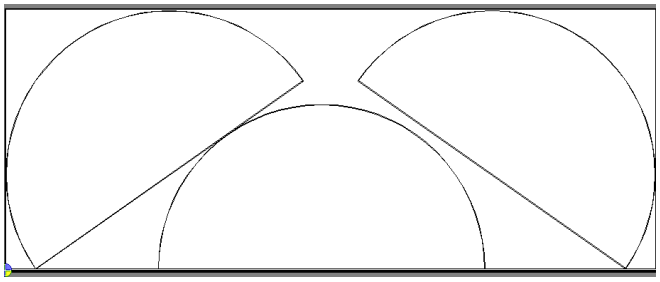
7.3.2.16 TotalRegionFootprint

Name	TotalRegionFootprint
Description	The sum of the region areas of all instances of the part in the footprint.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• PartTotalFootprintQuantity • PartMinNonLeadsRegionArea
Outputs	• 0..whatever
Calculation Method	• PartTotalFootprintQuantity * PartMinNonLeadsRegionArea
Comments	

7.3.2.17 Rotatability

Rotatability refers to the possibility of rotating part geometry to maximize usage of the region length or width. Examples:

- Parts whose maximum bounding rectangle dimensions are greater than or equal to the minimum region dimension: a 60” radius half circle on a 96” wide plate can be rotated at 35” to use the entire plate width. In fact, on a 240x96 plate, it is the only way to get 3 of these parts on the plate:



Name	Rotatability
Description	Rotatability refers to the possibility of rotating part geometry to maximize usage of the region length or width
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	
Outputs	
Calculation Method	
Comments	

7.3.2.18 InterlockabilityScore

Name	InterlockabilityScore
Description	Estimates the degree to which the region will be minimized when a part is paired with itself.
Purpose	This will used by various NestingSolutions (particularly PatternNesting) to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• ShapeType • ShapeTendency • ProfileRatio • RegionRatio • MinNonLeadsRegion • TotalConcavityArea
Outputs	0..100
Calculation Method	Hard coded for known shapes and calculated for unknown shapes.
Comments	• This concept could also be extended to how well a part could interlock with other parts (although to really calculate that score you may need both parts.)

7.3.2.19 MaximizeHeightAngle

Name	MaximizeHeightAngle
Description	The rotation angle at which the height of the part region is maximized.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer

Used By	Many
Inputs	• The part
Outputs	0..360
Calculation Method	We rotate the part until we find the angle at which the region height is maximized.
Comments	

7.3.2.20 MaximizeWidthAngle

Name	MaximizeWidthAngle
Description	The rotation angle at which the width of the part region is maximized.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• The part
Outputs	0..360
Calculation Method	We rotate the part until we find the angle at which the region width is maximized.
Comments	

7.3.2.21 ClcArrayabilityScore

Name	ClcArrayabilityScore
Description	Estimates how effectively a part can be CLC arrayed.

Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MtcNestShape • EstimatedFootprintQuantity • ProfileRatio • RegionRatio • ShapeType
Outputs	0..100
Calculation Method	Factor = 1.0 If EstimatedFootprintQuantity <= 2 Factor = 0.25 If EstimatedFootprintQuantity <= 4 Factor = 0.5 If EstimatedFootprintQuantity <= 6 Factor = 0.75 Score = ProfileRatio / RegionRatio * 100 * Factor
Comments	

7.3.2.22 PatternArrayabilityScore

Name	PatternArrayabilityScore
Description	Estimates how effectively a part can be pattern arrayed.
Purpose	This will used by various NestingSolutions to determine if a given solution is viable.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MtcNestShape • EstimatedFootprintQuantity • ProfileRatio

	• RegionRatio • ShapeType • MinNonLeadsRegion • TotalConcavityArea
Outputs	0..100
Calculation Method	Hard coded for known shapes and calculated for unknown shapes.
Comments	• This may be a job for neural networks.

7.3.2.23 PatternArrayStrategy

Name	PatternArrayStrategy
Description	Chooses a pattern array strategy for a part.
Purpose	This will used by various NestingSolutions to determine which pattern array strategy to use to pattern array a part.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• RectuangularishnessScore • MtcNestShape • EstimatedFootprintQuantity • ShapeType • ShapeTendency • ShapeNumberOfLines • ShapeNumberOfArcs • MinNonLeadsRegionArea • ExteriorProfileArea
Outputs	• Basic • Intermediate • Advanced
Calculation Method	If RectangularishnessScore > 90 Strategy = Basic Else if single part cluster Hard coded for known shapes

	Calculated by counting lines, arcs and concavities for unknown shapes Else if multi part cluster Strategy = Intermediate
Comments	• This may be a job for neural networks (but maybe only for the values that hard currently not hard-coded based on ShapeType)

7.3.2.24 PatternArrayRotation

Name	PatternArrayRotation
Description	Chooses a pattern array rotation angle for a part.
Purpose	This will used by various NestingSolutions to determine what rotation angle to use when pattern arraying a part.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• RectuangularishnessScore • MtcNestShape • EstimatedFootprintQuantity • ShapeType • ShapeTendency • ShapeNumberOfLines • ShapeNumberOfArcs • MinNonLeadsRegionArea • ExteriorProfileArea
Outputs	0..360
Calculation Method	If RectangularishnessScore > 90 Angle = 90 Else if single part cluster Hard coded for known shapes Calculated by counting lines, arcs and concavities for unknown shapes Else if multi part cluster Ratio = ExteriorProfileArea / MinNonLeadsRegionArea If Ratio > 85

```
    Angle = 45
  Else If Ratio > 75
    Angle = 30
  Else If Ratio > 65
    Angle = 15
  Else If Ratio > 55
    Angle = 10
  Else If Ratio > 45
    Angle = 5
  Else
    Angle = 1
```

Comments

7.3.2.25 FillerPartScore

Name	FillerPartScore
Description	Estimates how effectively a part can be used to fill in a region.
Purpose	This will used by various NestingSolutions in building part sets.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• ProfileRatio • ClusterPriority
Outputs	0..100
Calculation Method	score = (1 - ProfileRatio) * 50 + 50 * (ClusterPriority == 99)
Comments	• This was a very basic implementation of this score. • We may want to use the space map to generate this. • Filler (priority 99) parts get priority over non-filler parts.

7.3.2.26 ProfileRatio

Name	ProfileRatio
Description	Represents the ratio of the area of the exterior profile of the part to the available area in the nesting region.
Purpose	This will used by various NestingSolutions and others.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• ExteriorProfileArea • AvailableArea
Outputs	0..1
Calculation Method	ExteriorProfileArea / AvailableArea
Comments	

7.3.2.27 RegionRatio

Name	RegionRatio
Description	Represents the ratio of the area of the min non-leads region of the part to the available area in the nesting region.
Purpose	This will used by various NestingSolutions and others.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MinNonLeadsRegionArea • AvailableArea
Outputs	0..1
Calculation Method	MinNonLeadsRegionArea / AvailableArea
Comments	

7.3.2.28 MaxDimensionRatio

Name	MaxDimensionRatio
Description	Represents the ratio of the maximum dimension of the min non-leads region of the part to the maximum dimension of the nesting region.
Purpose	This will used by various NestingSolutions and others.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MinNonLeadsRegionMaxDimension • RegionMaxDimension
Outputs	0..1
Calculation Method	MinNonLeadsRegionMaxDimension / RegionMaxDimension
Comments	

7.3.2.29 MinDimensionRatio

Name	MinDimensionRatio
Description	Represents the ratio of the minimum dimension of the min non-leads region of the part to the minimum dimension of the nesting region.
Purpose	This will used by various NestingSolutions and others.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MinNonLeadsRegionMinDimension • RegionMinDimension

Outputs	0..1
Calculation Method	MinNonLeadsRegionMinDimension / RegionMinDimension
Comments	

7.3.2.30 ArcInformation

We collect information for arcs in five different size categories: very small, small, medium, large and very large. This information is generated for parts for use in determining nesting compatibility with other parts and available space.

a. NumberOfArcs – total number of arcs in this size category
b. TotalLengthOfArcs – total length of arcs in this size category
c. PerimeterRatio – ratio of the total length of arcs in this size category to the length of the perimeter

7.3.2.30.1 NumberOfArcs

Name	NumberOfArcs
Description	The total number of arcs in a particular size category.
Purpose	To provide the number of arcs in each size category.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MtcNestShape
Outputs	0..whatever
Calculation Method	Go through all the arcs and increment NumberOfArcs in the appropriate size category for each arc.
Comments	

7.3.2.30.2 TotalLengthOfArcs

Name	TotalLengthOfArcs
Description	The sum of the lenths of all arcs in a particular size category.
Purpose	This is used to develop ratios, etc.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• MtcNestShape
Outputs	0..whatever
Calculation Method	Go through all the arcs and increase TotalLengthOfArcs in the appropriate size category for each arc.
Comments	

7.3.2.30.3 PerimiterRatio

Name	PerimeterRatio
Description	The ratio of TotalLengthOfArcs for a particular size category to the ShapePerimeter.
Purpose	This is used to get a sense for how much of a profile is made up of arcs in a given size category.
Generated By	MtcPartAnalyzer
Used By	Many
Inputs	• ShapePerimeter • TotalLengthOfArcs
Outputs	0..1
Calculation Method	TotalLengthOfArcs / ShapePerimeter
Comments	

7.3.3 Part Category Characteristics

Parts are divided into five different categories by size (relative to the size of the available region):

1. Very small parts
2. Small parts
3. Medium parts
4. Large parts
5. Very large parts

We collect and calculate information about the parts in each category. This information is then used to determine the PartsNestingCompatibilityScore and the PartsSpaceNestingCompatibilityScore.

7.3.3.1 Size

Name	Size
Description	The size of a part relative to a given region.
Purpose	To give an idea of how big a part is compared to a region.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• PartRegionRatio
Outputs	• Very small – (100 parts per region) • Small – (20-100 parts per region) • Medium – (10-20 parts per region) • Large – (2-10 parts per region) • Very large – (1 part per region)
Calculation Method	If RegionRatio < 0.01) PartSize is Very small Else if RegionRatio < 0.05 PartSize is Small Else if RegionRatio < 0.1 PartSize is Medium Else if RegionRatio < 0.5 PartSize is Large Else PartSize is Very large
Comments	

7.3.3.2 AverageExteriorProfileRatio

Name	AverageExteriorProfileRatio
Description	The ratio of exterior profile area to the region area for all parts in this category.
Purpose	To give an idea of how big on average all parts in this category are compared to their region.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• TotalExteriorProfileArea • TotalPartRegionArea
Outputs	• 0..1
Calculation Method	TotalExteriorProfileArea / TotalPartRegionArea
Comments	

7.3.3.3 AveragePartAreaRatio

Name	AveragePartRatio
Description	The ratio of part area to region area for all parts in this category.
Purpose	To give an idea of how much of the space of their region on average the parts in this category take up.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• TotalPartArea • TotalPartRegionArea

Outputs	• 0..1
Calculation Method	TotalPartArea / TotalPartRegionArea
Comments	

7.3.3.4 AreaRatio

Name	AreaRatio
Description	The ratio of the total area of parts in this category to the total area of all parts in the part set.
Purpose	To give an idea of how much area the parts in this category contribute to the overall area of the part set.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• TotalPartArea • TotalPartSetArea
Outputs	• 0..1
Calculation Method	TotalPartArea / TotalPartSetArea
Comments	

7.3.3.5 FillAreaRatio1

Name	FillAreaRatio1
Description	This measures how well parts in this category fit inside interiors or concavities of all parts in the part set.
Purpose	To give an idea of how well parts in this category can fill and interlock with other parts in the part set.

Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• MaxContainedDistance • MaxDistFromBoundary • ProfileType
Outputs	• 0..1
Calculation Method	Using ProfileType, MaxContainedDistance and MaxDistFromBoundary for profiles and concavities, sum up the total part area from this category that could fit inside of any interior profile or concavity.
Comments	• This uses information generated from the space map. • To be considered, an exterior profile must be able to fit inside of the interior profile or fit at least halfway inside of a concavity.

7.3.3.6 FillAreaRatio2

Name	FillAreaRatio2
Description	This measures how well other parts fit inside of the interiors and concavities of the parts in this category.
Purpose	To give an idea of how well parts in this category can be filled by and interlock with other parts in the part set.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• MaxContainedDistance • MaxDistFromBoundary • ProfileType
Outputs	• 0..1
Calculation Method	Using ProfileType, MaxContainedDistance and MaxDistFromBoundary for profiles and concavities, sum up the total part area from this category that could fit inside of any interior profile or concavity.

Comments	• This uses information generated from the space map. • To be considered, an interior profile must completely contain an exterior profile or a concavity from an exterior profile must be able to fit half of an exterior profile.

7.3.3.7 QuantityRatio

Name	QuantityRatio
Description	The ratio of total quantity of all parts in this category to the total quantity of all parts in the part set.
Purpose	To give an idea of what percentage of total quantity of parts in the part set are in this category.
Generated By	MtcPartSetAnalyzer
Used By	Many
Inputs	• TotalPartQuantity • TotalPartSetQuantity
Outputs	• 0..1
Calculation Method	TotalPartQuantity / TotalPartSetQuantity
Comments	

7.3.4 Profile/Concavity Characteristics

Profile/Concavity characteristics are intended to be used in developing part, part category and part set characteristics. They area primarily generated through the use of the space map.

7.3.4.1 Type

Name	Type
Description	Represents whether the element is an interior profile, an exterior profile or a concavity.

Purpose	To identify the type of element.
Generated By	MtcSpaceMap
Used By	Many
Inputs	• MtcNestShape
Outputs	• Interior profile • Exterior profile • Concavity
Calculation Method	It just knows.
Comments	

7.3.4.2 Area

Name	Area
Description	The area inside the profile.
Purpose	To provide the area inside the profile.
Generated By	MtcSpaceMap
Used By	Many
Inputs	• MtcNestShape
Outputs	• 0..whatever
Calculation Method	It just knows.
Comments	• For concavities, a closed profile is obtained by connecting the end point to the start point.

7.3.4.3 MaxDistanceRequired/Available

Name	MaxDistanceRequired/Available
Description	The longest line between any two points on the profile that does not intersect the profile.
Purpose	This allows for a quick comparison to see if a profile could nest inside of another profile or inside of a space.
Generated By	MtcSpaceMap
Used By	Many
Inputs	• MtcSpaceMap
Outputs	• 0..whatever
Calculation Method	This is obtained from the max distance transform of the SpaceMap.
Comments	• This is only somewhat accurate as it only calculates length along lines at every 45 degrees.

7.3.4.4 MaxMinDistanceToBoundary

Name	MaxMinDistanceToBoundary
Description	This is maximum value of the distance from the boundary of the shape to any point inside the shape.
Purpose	This allows for a quick comparison to see if a profile could nest inside of another profile or inside of a space.
Generated By	MtcSpaceMap
Used By	Many
Inputs	• MtcSpaceMap
Outputs	• 0..whatever
Calculation Method	This is obtained from the distance transform of the SpaceMap.

Comments	• This is only somewhat accurate as it only calculates length along lines at every 45 degrees.

7.3.5 Space Characteristics

7.3.5.1 ArcInformation

This has not been thought through yet, but for a rough idea, see Part Characteristics | ArcInformation.

8 Why IntelliNest doesn't work as well as we had hoped

IntelliNest works very well for certain jobs, and pretty well for the remainder. On the whole, it is an improvement over any of the previous strategies. However, for a number of reasons, it does not work as well as we had hoped.

8.1 It was hard to predict which nesting solution would work best for a given part set

Having a set of effective nesting solutions available is one thing, choosing the best one for each sitation is another. This turned out to be one of the most difficult things to get right, and, in fact, we didn't get it right as often as we would like. After getting less than optimal results through sets of hand coded formulas, we tried another approach: neural netwoks.

8.2 Neural networks didn't help in the decision making

Neural networks excel at finding non-linear relationships between the inputs and outputs of a problem.This approach seemed promising. At first. However, there were challenges. First, for problems with a lot of features (or dimensions), a LOT of training data is required, and even though we were able to generate training data, it is not clear if it was enough data, or if the records adequately covered the problem space. Second, there is a non-continuous relationship between the output (say, nested utilization) and the inputs (say, part size) such that minor changes to part size could result in major changes to nested utilization. It wasn't clear to what degree neural networks could accurately predict an outcome. Finally, there was no expert on neural networks involved in the project, except for a consultant who came in for one day and thought we were following the correct approach, and that it should ultimately work. We continued to follow that approach, and, ultimately, it didn't work. As promising as it originally seemed, neural networks were a failure and were not used in IntelliNest.

8.3 It creates a single solution

IntelliNest uses analysis and classification to try to predict a set of steps that will lead to the best nested result. Ultimately, however, it only tries one approach at each decision point, and if that approach later turns out to be a poor choice, too bad. There is no going back. Our old nesting strategies basically found the "best" place for the next part in the list, then it placed it there and moved on. IntelliNest chooses a strategy, a part set, and a region to nest in, places all of the parts it can there, and then moves on to a

new strategy, part set and region. This is a step better. However, in recent years it has become clear that the nesting problem is best solved by optimization algorithms, which generate many solutions, evolves them over time, and selects the best.

IntelliNest was designed to overcome this limitation to some extent, through the use of a look-ahead feature. At each decision point, several nesting solutions could be applied, each of which would be further nested for a number of additional steps. This was (and maybe still is) a promising approach, but it had several problems and was shelved. This is detailed in the next section.

8.4 The look-ahead feature didn't work

There were several problems with the look-ahead feature. First, intermittent crashes occurred during execution of the look-ahead, and because we were attempting to release IntelliChoice ASAP, we chose to disable the feature rather than fix these problems. Another reason we decide to defer this feature was because it was SLOW, which is the second issue. In addition, since the lookahead was so slow, it would have to be multi-threaded. However, many of the nesting solutions are not thread safe. So even before we could multi-thread this area, all of the current nesting solutions would have to be thread safe. Finally, comparing two partial nests turned out to be another difficult problem. Given two partial nests, which may contain different parts and occupy different regions, which will lead to the best final nest? Not easy to determine. Of course, we also attempted to use neural networks to choose the best partial nest, and of course they didn't work.

8.5 It works at a nest level, not a job level

Even though the majority of jobs created by our customers are single nest jobs, benchmarks used to compare competing nesting application are often multiple nest jobs, with many nested sheets. IntelliNest tries to create the best possible nest from the remaining parts before moving to the next nest. But to do a good job on large, multi-nest jobs, the job itself has to be optimized. To win these benchmarks, you have to nest on fewer plates, and this can only be done by optimizing at the job level instead of at the nest level. IntelliNest didn't address this, so we didn't improve our performance on these types of customer benchmarks.

8.6 It was designed for expansion and was never expanded

An excellent aspect of the IntelliNest design is that new nesting solutions could be plugged into the system without having to refactor IntelliNest. In general, you can handle new nesting situations without breaking the old situations. However, in the years since IntelliChoice was released, this has not happened. It is possible that incremental improvements (i.e. new nesting solutions) to IntelliNest over the last decade could have closed the gap between IntelliChoice and our competitors, but those improvements were never made.