TpaNestingOEM

4.1.0

TpaNestingOEM 库

Tecnologie e Prodotti per l'Automazione

摘要

1 简介

TpaNestingOEM.dll（又名 TPA_N）是一款专为软件开发人员设计的产品，可以集成为 32 位和 64 位 Windows 体系结构 的库。
TPA_N 是用于自动嵌套复杂 2D 形状的库，支持开发优化应用程序，实现不同应用领域的 2D 放置或形状切割。

库必须集成为产品的组件。

网站提供库演示 app www.tpaspa.com/oem-nesting-library 但应注意的是，这个 app 不显示所有可用的 TPA_N 函数。

1.1 版本

简介	*发布日期*	*注释*
1.0.0	14-04-2021	
1.5.0	11-01-2022	方法 *Compute*：修改了原型 添加的属性：*Version* 添加的方法：*SaveSolutionDXF*
1.8.0	14-06-2022	添加的属性：*TimeCompute*、*TimerProgres*
1.9.0	29-09-2022	产品维护更改
3.0.0	20-11-2023	整体产品检查
3.2.0	17-06-2024	结构 *NestBox*：添加的与放置整体尺寸有关的字段 添加的属性管理：*UseOnlyExtraPart* 改进了与板材边缘的间隔放置的应用标准。 改进了非矩形部件的*矩形*优化计算标准。 解决了内部产品验证过程中报告或产生的问题。
3.3.0	11-06-2024	可以请求使用部分或集群的位置数量（参见常量：MAX_ROW _N、MAX_CLUSTER_N） 产品维护更改
3.5.0	14-04-2025	产品维护更改
3.6.0	20-10-2025	产品维护更改
4.0.0	20-10-2025	产品维护更改 新的许可证管理制度
3.7.0	16-03-2026	版本更新（3.6.0） 矩形优化也增加了分步执行功能 增加了零件群组功能 增加了增量嵌套功能 *NestSheet*结构：增加了字段 *NestBox*结构：增加了字段 Square 优化：增加了面板岛的管理。 删除了废弃属性
4.1.0	16-03-2026	版本更新（4.0.0） 库规范已与 3.7.0 版本保持一致。

1.2 许可证

从 4.0.0 版本开始，TPA_N 操作需要软件许可证验证。
在本篇文档中，*密钥”一词作为“软件许可”的同义词使用*。
软件许可管理由专门的应用程序处理，详细信息请参阅该程序。

密钥可以指定两个等级的优化授权：*矩形*（订单代码：SW 2.4.01）和*真形状优化*（订单代码：SW 2.4.02）。
启用*“真形状”*优化 会自动 the*“矩形”*优化功能，并提供从 DXF 格式文件导入几何图形的附加功能。

导入 DXF 文件时仅解析 ASCII 版本。

除非另有说明，以下操作步骤均假定在拥有完整授权的情况下进行。

若无密钥，将无法执行任何项目优化。

1.3 系统要求

安装该应用程序的电脑最低要求如下：

- 操作系统：64 位 Windows 10 或更高版本
- 处理器：双核（推荐四核）：CPU 速度越快，优化计算速度越快
- 内存：1 GB 或更大
- 所需 Windows 扩展：.NET Framework 4.8 或更高版本
- 安装该套件需要管理员访问权限。

1.4 所有权和版权

1.5 联系方式

如需更多信息，请联系：

T.P.A. Srl Tecnologie e Prodotti per l'Automazione - Via Carducci, 221 - 20099 Sesto S. Giovanni (MI) - ITALY
电话：+390236527550
电子邮件：info@tpaspa.it

或访问我们的网站：
www.tpaspa.com

2 用户指南

TPA_N 在 ***TpaNestingOEM*** 命名空间指定 ***Nesting*** 类。

TPA_N 计算矩形或三角形放置区域的*形状*放置。
*形状*指多边形几何；
- 最简单的形状定义为矩形
- 形状可以定义一个外部几何，以及一个或多个内部几何（孤岛）。

*部件*是区域中的放置带来的单个实体
放置区域称为*板材*。

*部件*具有几何，通常指定为*形状*。赋值给*部件*的形状只有一个外部路径，可以具有孤岛：内部废料区域，可以用于放置其他较小*部件*。
*板材*赋值放置部件的材料板材。板材具有几何，通常也赋值为*形状*：通常*板材*为简单矩形，但可以赋值不规则形状和可能的内约束区域无法用于放置部件。

赋值*部件*可以定义基本几何转换，如旋转和/或镜像，以及常规（可用的数量、定位优先级）或通常技术赋值（厚度、刀具直径、材料、颜色、纹理…）。为每种类型的零件定义需要放置的数量和可能放置的最大数量，以填充已使用的板材。

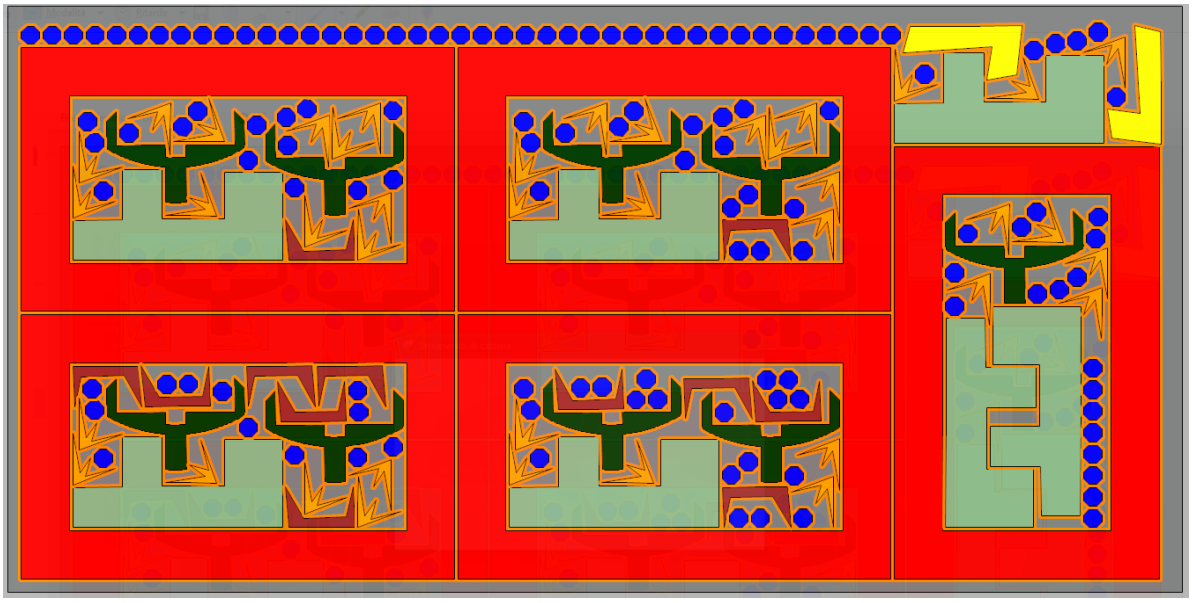
可以创建两个或多个部分的分组并直接使用这些组进行放置：我们正在讨论手动集群。

赋值*板材*允许定义一般的技术设置（厚度、材料、颜色、纹理…）。为每种板材定义可用的数量。

通常技术数据可以将*部件*和*板材*之间的匹配条件和/或筛选器应用到嵌套过程。

嵌套优化的*结果*赋值放在可用*板材*的*所有部件*的位置和旋转。嵌套*结果*可以赋值一个或多个*板材*：每个*板材*包含至少一个*部件*的放置。

下图显示用 *True Shape* 优化得到的矩形板材。



2.1 嵌套优化类型

带来嵌套求解的全部计算过程称为*优化阶段*。
TPA_N 管理两种优化模式：*Square* 和 *True Shape*.

Square 优化

- 管理矩形*板材*中矩形部*件的*放置
- 一个部件的放置可以应用0°或90°旋转（逆时针），镜像变换
- 可以计算后续的多次优化，并将其作为等效方案进行评估，或者直接请求获取最优结果。

如果所有*部件*和*板材*赋值为简单矩形，库默认进行*矩形*优化。

如果强制*矩形*优化：

- 为外形本身的边界框考虑赋值几何外形的*部件*，忽略赋值为孤岛的外形
- *手动集群*的处理是以其组成零件的整体边界框为准。
- 对于已分配几何轮廓的图纸，将考虑具有相同轮廓的边界框：在这种情况下，优化方案可能会在已分配图纸之外进行放置。
- 将指定为图纸约束区域的轮廓视为相同轮廓的边界框：在这种情况下，约束区域被高估了。

True Shape 优化

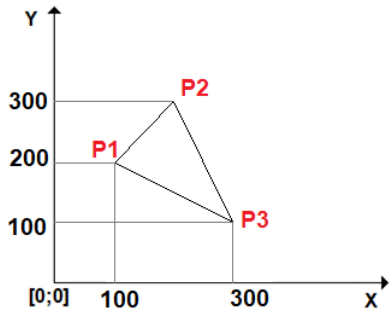
- 管理以任何方式赋值的*部件*和*板材*的放置
- *部件*可以包含一个或多个孤岛（内部废料区域），可以用于放置其他*部件*
- *板材*可以包含排除部件放置的约束区域
- 放置*部件*可以应用变量旋转和可能的镜像转换
- 在*板材*上放置*部件*遵循部件和板材的形状，结果是优化材料使用
- 可以计算连续优化，和同等替代一样重要。

2.2 单位和坐标

在 TPA_N 中，公制值（坐标，尺寸）在函数 *IniSettings(...)* 的调用中采用定义为 [mm] 或 [in] 的单位。

TPA_N 在 2D 笛卡尔坐标系中工作：
- 坐标点 [0;0] 是左下角
- X 是水平轴，向右为正
- Y 是垂直轴，向上为正。

图中显示赋值在 3 个点的一个简单多边形示例：P1(100;200)，P2(200;300)，P3(300;100)



将参考多边形的显著几何特性。我们看一看如何定义：
- *边框*：在 XY 平面完全包含外形的矩形区域
- *LB（左下）角*：边框左下角：
  - 我们将 LB 相对的角称为 RT（右上）
  - 左右指水平轴（X 轴）
  - 上下指垂直轴（Y 轴）
  - 如图所示：边框的极限角不一定属于外形。

如图所示：
- 边框由两个极限角标识：LB = {100; 100}，RT = {300; 300}。

所有角度以度和分为单位表示。旋转解释有意义的情况下：
- 逆时针旋转与正角度关联
- 顺时针旋转与负角度关联。

2.3 嵌套尺寸

TPA_N 可以在两种不同类型的板材中操作，取决于项目常规设置 *OneNestingDimension*。
- (*False*) 2D：板材赋值嵌套二维的平面部分
- (*True*) 1D：嵌套在线性支撑上，板材为棒、外形、管。

嵌套 2D

优化在板材约束的 2D 空间上操作。选择对应默认操作 (*OneNestingDimension=false*)。
典型应用与切割各种材料的规则或不规则形状板材有关：金属、树脂玻璃、木材、皮革、纸、布料。

嵌套 1D

典型应用与切割有关：
- 棒、管或金属外形
- 木梁。

选择强制使用*垂直方向*，但不限制水平方向的连续放置或部件高度处的板材高度。
图显示梁优化求解示例：左侧的两个三角形显示优化过程如何保持二维平面上的普通放置逻辑有效。

选择在条件选择中指导 TPA_N，研究以在线性支撑放置情况下获得更好的结果。不当选择可导致次优结果，尤其是从计算时间来看。
与选择相关的显著设置与以下有关：
- 选择部件刀具类型（*OneCutterDimension* 常规设置）
- 与板材边缘切割外形重叠有关的设置（常规设置：OverrunSheet、MaximizeOverrunSheet）
- 与部件镜像应用标准有关的选择（ModeMirrorPart 常规设置）

上面的示例：
- 可以用于线性类型刀具（例如刀片）
- 部件切割不包含沿梁上下部分的水平边缘：可以完全在梁边缘外应用切割
- 梁没有良好侧（上下相等）：因此可以根据更高效率，在正常或镜像模式应用部件。

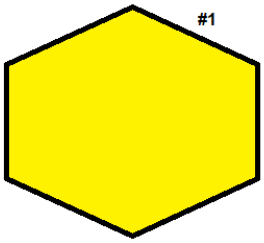
2.4 外形定义

部件或板材通过主几何外形和最终辅助外形 赋值 （用于零件的岛屿或用于板材的废料区域）；最简单的几何外形是矩形，可以用长度和高度尺寸直接赋值。

几何外形可以由直线或曲线（弧形）段组成，用作*闭合*，可以通过直线段闭合。正如上图中：外形由三个点 (P1, P2, P3) 赋值，可以用连接 P3 和 P1 的直线段闭合。

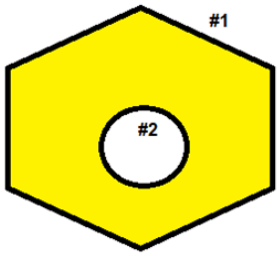
每个外形赋值一个*净*区域：
- 用于部件的切割区域
- 用于板材的边缘区域。

如果部件（或板材）赋值几何，则不使用部件的主要 2D 尺寸（部件/板赋值的 ***NestPart***/***NestSheet*** 结构的 *L* 和 *H* 字段）。



图形表示没有孤岛的部件情况：

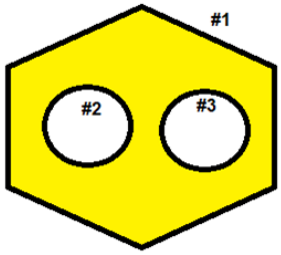
- #1 标记主要外部外形。



图形表示带 1 个孤岛的部件情况：

- #1 标记外部主要外形
- #2 标记内部废料外形。

赋值部件：可以利用废料外形放置其他部件。
赋值板材：废料外形表示无法用于放置的区域。



图形表示带 2 个孤岛的部件情况：

- #1 标记外部主要外形
- #2 和 #3 标记两个内部废料外形。

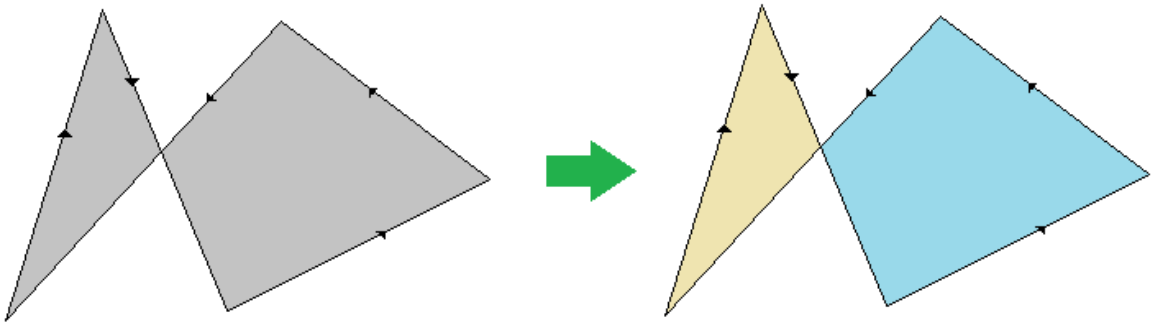
几何边界框可以赋值在任意位置，X 和/或 Y，正或负，区域：赋值正 XY 区域的几何和 (0;0) 最小整体尺寸可以方便编程，但不是必要条件。

简化外形

如果赋值*简单*几何，几何外形有效：没有空或交叉段，闭合非空区域。

要具有*简单*几何，通过TPA_N绘制每个外形：
- 移除重合（少于一个epsilon线性求解）或对齐顶点
- 移除交叉情况，可能拆分为更多外形。

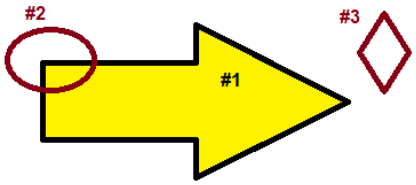
图中可以看到具有交叉段的外形示例：
- 左侧：原始外形
- 右侧，两个颜色区域突出显示外形如何拆分为两个外形，每个外形定义一个*简单*几何
- 赋值：
  - 部件：获得的两个简单外形相应连接，用作一个简单元素
  - 板材：可以在每个简化几何内分别放置。



嵌套过程丢弃无法归属为有效几何的外形，可能在放置计算中排除。

必须格外注意赋值内部外形（部件孤岛或板约束区域）：
- 必须完全在相应主要外形内
- 对于部件中的孤岛：仅当其内部不再包含更多孤岛，且不与其他孤岛相交时，才可用于放置其他部件。

图形显示错误放置情况：



- #1标记主要外形（内部区域颜色：黄色）
- #2标记*交叉*主要外形的废料外形
- #3标记主要外形*外部*的废料外形。

嵌套过程将忽略两个废料外形。具体来说：评估*板材*上的部件放置时，无法遵循两个废料外形使用的区域。
这里显示的情况要求特定放置重叠（#2，#3）所示外形区域的其他部件：调用应用程序负责检查和/或报告类似情况。

外形近似

外形曲线部分用一系列线段近似，定义最大允许准确度 0.3 mm:
- 意味着直线路径与*实际*弧线偏差的最大距离不超过 0.3 mm
- 弧线近似的线段数量根据弧线半径而不同：随半径减小而增加。

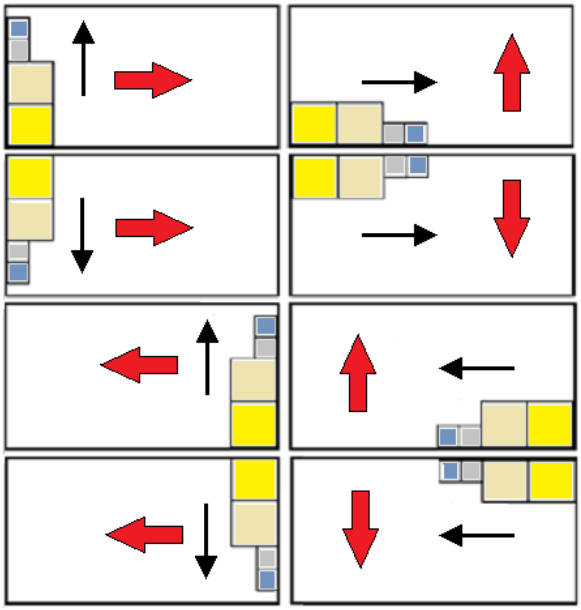
这里介绍的公差值表示为*弦误差*。

外形的一个或多个线段近似可能导致增加相邻放置之间或者部件与板材边缘之间的安全距离（至少等于*弦误差*）。

2.5 嵌套延展方向和顶点

嵌套方向赋值放置进料方向，填充板材：
- *水平方向*：首先垂直执行放置（图中：左侧，水平红色箭头）
- *垂直方向*：首先水平执行放置（图中：右侧，垂直红色箭头）

嵌套顶点赋值四种可能值中的放置起始顶点：



- 0 = 左下（图中：从上数第一排的情况）
- 1 = 左上（图中：从上数第二排的情况）
- 2 = 右下（图中：从上数第三排的情况）
- 3 = 右上（图中：从上数第四排即最后一排的情况）

信息符合项目总体赋值 Direction 和 Corner。

TPA_N 始终使用赋值的角设置。
TPA_N 尝试用修改方向值优化以获得更好求解。

2.6 匹配筛选器和组

赋值*部件*和*板材*需要通常技术设置，应用*部件*与*板材*之间的匹配条件和/或筛选器，即使不是正常使用条件。

首先看一看应用匹配条件时的设置。

这些设置对应 NestPart 和 NestSheet 结构中的字段：

- *S*：（double 类型）厚度
- *Fiber*：（integer 类型）用于常规材料赋值的
- *Colour*：（integer 类型）用于常规颜色赋值的
- *Grain*：（integer 类型）赋值颗粒方向。

S 区域：根据设定值相等与否评估匹配，除非直线比较的 epsilon 匹配 MinResolution 设置。
其他设置支持使用现场 *Fiber*、*Colour* 和 *Grain*（MatchType、MatchColor 和 MatchGrain）。

对部件和板厚度匹配组赋值的示例：

- 赋值两个部件（标识符：1、2），字段 *S*=18.0
- 赋值一个部件（标识符：3），字段 *S*=25.0
- 赋值一个板材（标识符：1），字段 *S*=0.0
- 赋值一个板材（标识符：2），字段 *S*=18.0

情况导致赋值 3 个匹配组：

- 组 1，匹配 *S*=18.0
- 组 2，匹配 *S*=25.0
- 组 3，匹配 *S*=0.0。

每个组单独优化：这相当于针对三个不同的项目请求优化。上面的示例正面仅组 1 如何生成嵌套求解，允许部件和板材之间关联；具有标识符（1、2）的部件将放置在具有标识符 2 的板材上。

对于多值赋值情况（例如：对于材料），部件和板材之间的匹配关联可以随复杂程度增加。示例：

- 组 1，匹配 *S*=18.0 和 *Fiber*=0
- 组 2，匹配 *S*=18.0 和 *Fiber*=1
- 组 3，匹配 *S*=25.0 和 *Fiber*=0。

纹理控制

还有一个字段显示在结构 NestPart 和 NestSheet 中，匹配颗粒方向赋值：
- 赋值发生在结构的 *Grain* 字段
- 此信息的技术含义取决于板材的材料（如：木材胶合板、金属）。

可赋值的值有三个：

- *0（无）：不赋值纹理*
- *1 (X)*：沿水平方向的纹理
- *2 (Y)*：沿垂直方向的纹理。

颗粒赋值不一定导致确定单独匹配组：*Grain* 字段值相同的部件可以放在不同组的板材和/或具有相同或不同 *Grain* 字段值。行为由 MatchGrain 属性决定。

如果 *MatchGrain=true*：每个 *Grain* 值导致一个单独匹配：
- *Grain=(1, 2)* 的部件只能放在具有相同颗粒的板材中
- *Grain=0* 的部件只能放在无颗粒的板材中。

看一看 *MatchGrain=false* 情况下应用的标准：
- *Grain=(1, 2)* 的部件可以和 *Grain=0* 的板材上的任何旋转放置在一起
- *Grain=(1, 2)* 的部件可以放在 *Grain=(1,2)*，旋转符合赋值两个方向的板材上
- *Grain=0* 的部件可以放置任何旋转，与板材的 *Grain* 字段无关。

如果 *Grain=(1, 2)* 的部件可以放在不同颗粒板材上：
- 不保证在具有相同纹理的板材上的优先放置
- 如果可以在赋值颗粒的板材类型上使用部件，不应用有效 *RctMinimize* 设置。

2.7 应用于部件的转换

对于部件，可以赋值两个几何转换的信息：
- 镜像
- 旋转。

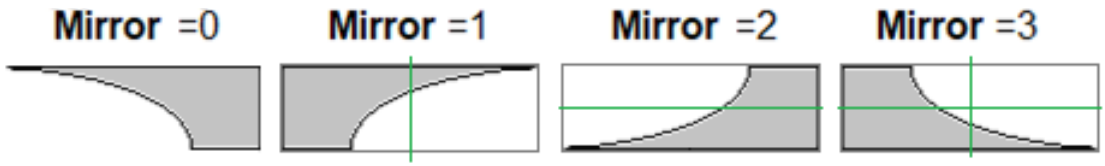
部件镜像

TPA_N 支持两种应用镜像部件的方式，按照 *ModeMirrorPart* 设置（**boolean** 类型）。

强制镜像

此默认功能对应 *ModeMirrorPart=false*。
可以沿一个或两个坐标轴请求镜像放置每个部件，四个选项可以选择：
- *0（无）*：不镜像部件
- *1 (X)*：沿自身边界矩形的垂直中心线镜像部件
- *2 (Y)*：沿自身边界矩形的水平中心线镜像部件
- *3 (X+Y)*：两个以前选项的总和。

部件赋值对应 NestPart 结构的 *Mirror* 字段：



使用部件和任何部件旋转前，应用几何。
模式对应有良好切割*侧*的板材情况：通常为上侧。
特定应用使用非线性支撑：木材或金属板、皮革、布料。

受控镜子

功能对应 *ModeMirrorPart=true*。
如果即使尝试不同旋转，也无法在正常模式下放置，可以对部件应用镜像。
选择包括四个选项，即使对于使用目的，前两个值之间的选择也足够了：
- *0（无）*：无法镜像部件
- *1 (X)*：可以镜像部件。

如果选择线性嵌套尺寸（*OneNestingDimension=true*）：仍可尝试镜像部件，即使可以在正常模式下放置部件。
模式对应没有良好切割*侧*的板材情况：上下侧相当。特定应用与线性支撑有关：杆、梁。

旋转控制

TPA_N 管理对部件应用旋转的可能。应用根据优化模式而不同：
- *Square*：部件可以逆时针旋转 90°
- *True Shape*：部件可以按角度步长旋转，可以更改此步长。

对于每个部件，可以赋值旋转允许的自由度，选项有三个：
- *0（无）*：部件不允许旋转
- *1 (90°)*：部件允许 90° 步长旋转（如果优化 *True Shape*）或逆时针 90° 步长旋转（如果优化 *Square*）
- *2 (Any)*：部件允许垂直步长旋转。如果优化 *Square*：选择对应上一个 (90°)。

部件赋值对应 NestPart 结构的 *Rotate* 字段。

Any 所选允许的旋转步长信息对应 StepAngle 常规赋值：
- 属性仅接受 1.0 到 90.0 之间的值
- 实际应用的旋转最小值是 360° 的约数。

设定值越小，放置计算阶段要求越高，包括需要的内存量和找到放置求解需要的时间：

- 设定值 45.0 允许最多 45.0° 最小步长旋转：一个部件具有 8 个可能放置求解，通过应用 45.0° 的所有倍数决定，即：0°、45°、90°、135°、180°、225°、270°、315°；
- 设定值 15.0 允许最多 15.0° 最小步长旋转：一个部件具有 24 个可能放置求解，通过应用 15.0° 的所有倍数（0°、15°、30°、□ 345°）决定。

然而，应用于给定部件的旋转列表可能会根据其他考虑而有所不同，例如：部件的尺寸和形状、指定的纹理。

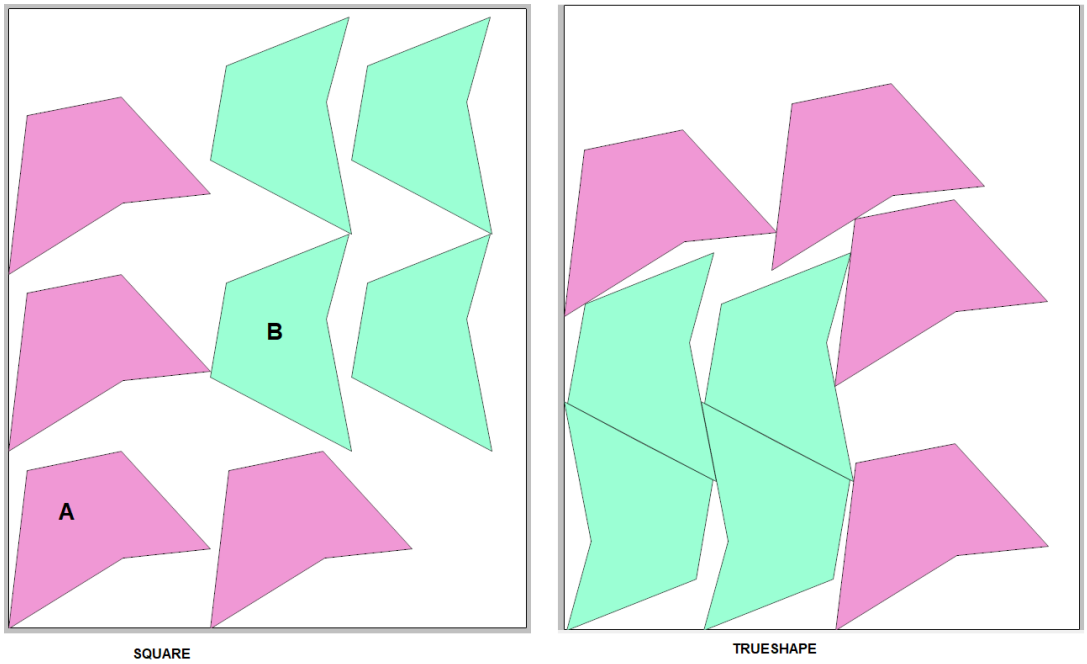
缩小部件边界框的选项

另一个常规***boolean***设置可以影响部件的旋转逻辑（RctMinimize）。
未分配矩形几何形状的部件可以应用此属性。

对于*Square*优化：
- *False*：部件按赋值使用。对于赋值几何：按原始或90°旋转（如果允许）放置
- *True*：可以提前更改部件初始放置，与原始比较，确定旋转以减小整体边界矩形。仅当部件可以旋转时可以进行：

对于*True Shape*优化：
- *False*：如果部件允许*any*旋转，则确定最小边界框的旋转
- *True*：等效设置，还用于允许90°步长旋转的部件。

图形对应用*Square*优化（左侧）、*True Shape*（右侧）以及*RctMinimize=true*的相同形状的执行两个优化：
- 图中：使用形状，无旋转可能。放置效果与原始几何相匹配
- **B**：使用形状，可以旋转(90.0°)。现在可以清楚看到每个放置如何从缩小相同形状边界矩形的旋转开始，然后提高放置优化的可能。



旋转和颗粒

部件颗粒赋值（X或Y）可以限制其旋转能力，或阻止在颗粒板材中放置：
- 首先，旋转的应用（*任何*）是被排除的：如果赋值，则减小为*(90°)*
- 如果部件和板材具有相同颗粒（例如X+X；Y+Y）：可以带旋转使用部件*(0°; 180°)*，如果部件无法旋转，则仅*(0°)*
- 如果部件和板材具有不同颗粒（例如X+Y；Y+X）：只有可以旋转并且旋转*(90°; 270°)*，可以使用部件。

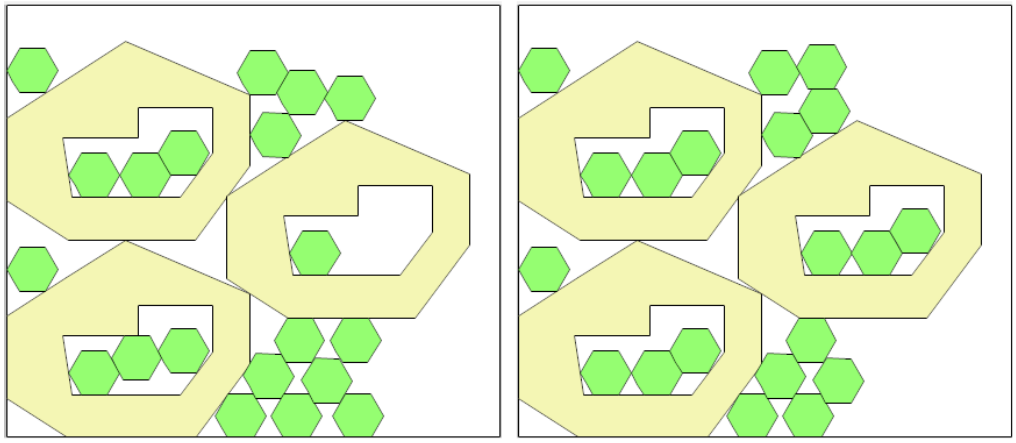
2.8 孤岛放置

*部件*可以指定可用于放置其他较小部件的孤岛。
该功能只能在*True Shape*优化中管理。
对于每个部件，可以启用孤岛内放置，对应NestPart结构的*UseIsle*字段。

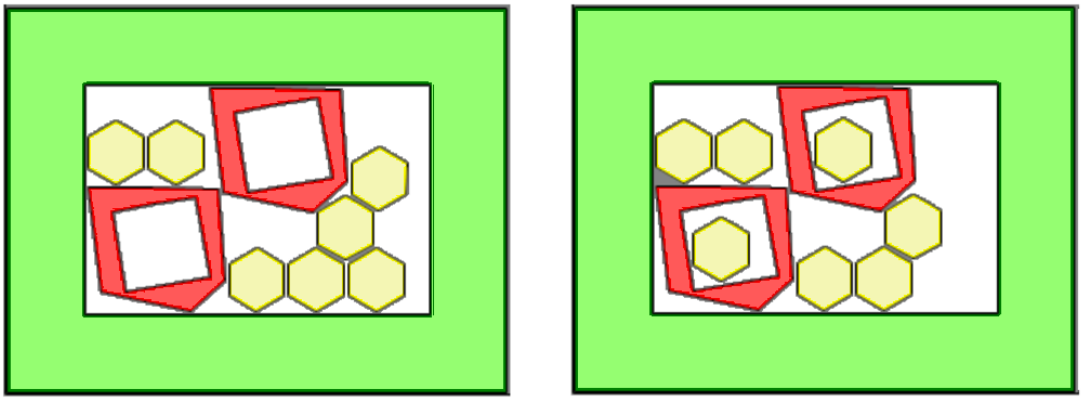
一些常规**boolean**设置控制孤岛放置：
- *PartInHole*：完全启用在部件废料区域内执行放置
- *PartInHoleMulti*：启用在部件废料区域内执行递归放置
- *PartInHoleBefore*：启用在部件废料区域内优先放置。

应考虑板材牢固和部件切割的相关技术因素，评估此功能的管理项。

图形比较*PartInHoleBefore*设置的效果：
- 左侧，*False*值情况：根据板材的使用部件最大填充的常规标准进行放置
- 右侧，*True*值情况：优化孤岛内的可能放置。



图形比较*PartInHoleMulti*设置的效果：
- 左侧，*False*值情况：*红色*显示部分放在*绿色*部分孤岛中；*红色*部件内没有其他放置
- 右侧，*True*值情况：在红色部件的孤岛内部也会进行放置。



2.9 控制放置间距

不同数据用于确定放置间距，以及放置和板材边缘之间间距。

部件切割直径

这些信息代表用于切割部件的刀具直径：对应CutterDiameter设置。
值可以为空(=0.0)用于未赋值自己刀具直径的部件。
对于每个部件，可以为主外形赋值一个刀具直径，为切割孤岛赋值一个，对应字段(*PartDiameter*、*IsleDiameter*)，NestPart结构：
- 如果两个字段具有有效（正）值：则部件为主外形和孤岛使用特定直径
- *IsleDiameter=0.0*：部件为孤岛使用*PartDiameter*

- *PartDiameter=0.0,IsleDiameter=0.0*：部件使用CutterDiameter用于主外形和孤岛。

对于一个部件的每个内部轮廓，也可以区分所使用的直径（参见 IniGeometry 函数）。除非另有说明，否则本文档假定一个部件的所有轮廓仅分配一个值。

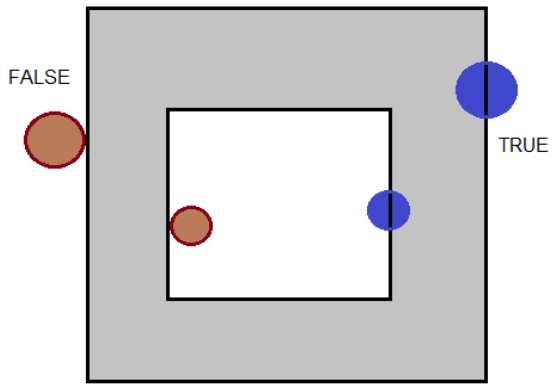
下面我们通常称*CutterDiameter*指示部件的切割直径：实际上，使用的值考虑这里显示的设置。

TPA_N 支持两种将工艺应用于部件的方式，对应于 *TecnoDistanceType*设置（*boolean*类型）。

如果*TecnoDistanceType=false*：围绕每个部件外形应用刀具：
- 在外部外形的外部部分，增加外形的有效区域
- 在废料切割外形的内部部分，减小废料区域。

如果*TecnoDistanceType= true*：刀具被应用于与每个部件的轮廓重叠：
- 外部轮廓的有效面积增加刀具尺寸的一半
- 一个废料区域的有效面积以同样的方式减少。

图中显示了一个带有孤岛的部件（两个轮廓均为矩形）：
- 两个轮廓之间的区域以灰色显示。
- 圆圈代表刀具：对孤岛使用了一个较小直径的刀具。
- 在左侧，圆圈（棕色）位于轮廓之外，对应于 *TecnoDistanceType=false*
- 在右侧，圆圈（蓝色）与轮廓重叠，对应于*TecnoDistanceType=true*。



除非另有说明，否则本文档假定：*TecnoDistanceType=false*的情况是有效的。

图形显示两个常规部件：
- 左侧，无孤岛的部件
- 右侧，有孤岛的部件

实心部分标记赋值给部件的路径：部件的净区域。
虚线路径通常标记部件的有效整体尺寸，添加切割刀具的整体尺寸。

零件切割机的类型

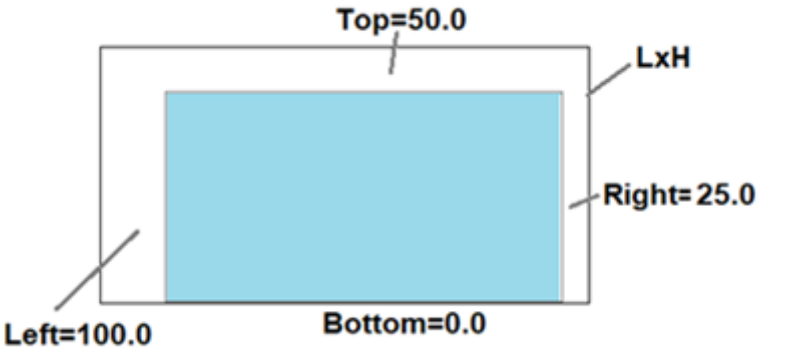
可以选择刀具类型，方式是配置OneCutterDimension设置（***boolean***类型）：
- *(False)* 刀具具有XY切割尺寸（例如铣削刀具、激光）
- *(True)* 刀具只有一个切割尺寸（例如刀片、刀具）。

在指出的第二种情况下，术语刀具直径应当被更准确地解释为*刀具尺寸*。
设置在*True Shape*优化中用作导致相邻放置之间或者部件与板材边缘之间间隔的因素之一：
- 使用刀具（例如铣削刀具、激光）会自动引入等于*弦误差*的最小间隔，因为刀具围绕边缘的移动跟踪曲线外部路径，必须用一系列线段来近似。 在图中，箭头指示了由刀具描绘的外部路径：原始矩形的每个边缘对应于一个圆弧的展开。

- 使用工具（例如刀片、切割器）会引入一个等于弦长误差的最小间距，这仅与零件原始轮廓的分配方式有关。一个特定的例子可能对应于矩形或凸形部件，其仅由直线段组成，并且使用单把刀具执行切割：在这种情况下，部件之间的间距可能不需要引入任何额外的补偿。

与板材边缘的间距

给定矩形*板材*的尺寸，可以赋值对放置无用和可以通过侧面区分的侧面区域。图中显示LxH尺寸板材，四个外部边缘赋值都不同：有用放置区域为彩色内部区域。



外部边距的通用设定对应于以下设置：
- MarginOuter：对于四个边缘的组合赋值
- MarginLeft, MarginRight, MarginBottom, MarginTop：对于边缘的区分赋值。

对于每张板材，可以分配其自身的外部边距，对应于 *NestSheet* 结构中的字段 (Border)：
- *Border*=0.0：板材使用通用设置
- 否则 (*Border>0.0*)：板材使用该结构中设定的值。

在所有情况下，可以赋值空或正值：页边距只会减少纸张的可用面积。
对于*True Shape*优化和非矩形板材：使用一个外部边缘。对于四个按边区分边缘，取最高设定值。

*最小间隔*板材边缘与放置之间，作为净区域，为0.0 mm。根据多个方面针对每个情况确定真实距离：
- *CutterDiameter*是部件切割技术的直径（最小有效值0.0）
- OverrunSheet设置
  - *False*：强制切割刀具不超过板材边距，因此与边缘保持最小距离，值（*CutterDiameter*）
  - *True*：允许刀具脱离板材边缘，基于MaximizeOverrunSheet设置，（在其全值或半值范围内）
  - 与边缘放置的最小距离评估可以添加更多间距，补偿：

- 不足外部边缘
- 应用*弦误差*后修改外形

- 在真实形状优化的情况下，如果为零件分配了不同的技术直径，则前面几点中要考虑的直径是最小直径。这意味着不同的零件可能以不同的方式接近板材边缘：零件的切割直径越大，与边缘的最小距离就越大。

板材内约束区域的间距

此问题仅涉及*True Shape*优化的案例。
前一段落中分配的边距不适用于板材内的约束区域。
在 Square 优化的情况下，约束区域按定义它们的轮廓的整体矩形来考虑，从而有效地增加了区域本身。

约束区域定义了无法进行放置的区域。
对于每张板材，可以在 *NestSheet 结构中*分配两个与废料板材*使用相关的字段*（IsleBorder，IsleDiameter）。对于这两个字段，正值或零值都是有意义的。
字段可以有多种用途，并且可以区分两种主要情况：
- 需要在放置点与板材约束区域之间分配最小距离
- 利用板材的约束区域分配初步放置，包括那些源自其他嵌套软件并按其自身运行逻辑管理的放置（参见：增量嵌套）。

这两种情况可能共存于一张板材的分配中。

现在让我们考虑如何将不同的情况形式化：
1. *(IsleBorder <=0.0, IsleDiameter<=0)*：在废料区域与部件的切割之间不应用边距（对于约束区域，OverrunSheet设置始终以 *false* 值应用，排除了与部件切割重叠的可能性）。

2. *(IsleBorder >0.0, IsleDiameter<=0)*：为约束区域分配一个外部边距（参见图中案例 A）。在这种情况下，OverrunSheet设置也以*false*值应用。

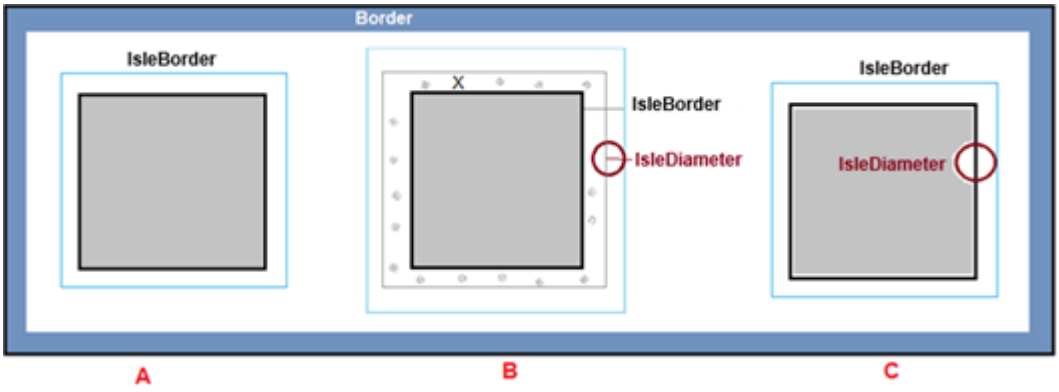
3. *(IsleBorder >0.0, IsleDiameter >0)*：约束区域现在可以对应于板材上已经存在的放置
   - *IsleBorder*设定约束区域切割刀具的外部整体矩形
   - *IsleDiameter*分配与约束区域关联的工艺。

   此处可以区分两种情况：
   - *IsleBorder >= IsleDiameter*：切割位于为约束区域分配的轮廓之外（参见图中案例 B）。
   - *IsleBorder < IsleDiameter*：切割部分位于为约束区域分配的轮廓内部。管理对应于 *IsleBorder=IsleDiameter*0.5*的极限条件（刀具中心沿区域轮廓运动：参见图中案例 C）。

   现在，约束区域像已经放置的部件一样被管理：
   - 应用对其他放置的切割轮廓的重叠和间距评估（设置：OverrunPolygon，MarginInner，OverrunSecurity）
   - 不应用TecnoDistanceType设置
   - 不应用OverrunSheet设置
   - 为空间需求和工艺分配差异化参数，对应于以下事实：已经分配的放置可能源自其他嵌套软件，按其自身逻辑运行，而两个参数的管理保证了最大的灵活性。

4. 最后，情况（*IsleBorder <=0.0, IsleDiameter >0*）：取*IsleBorder = IsleDiameter*，采用孤岛外部切割轮廓的方案。

图中显示了一张矩形板材，带有：
- 增加的外部边缘（蓝色边距）
- 三个标有字段（*IsleBorder, IsleDiameter*）的废料区域。每个区域用字母（A, B, C）标记。

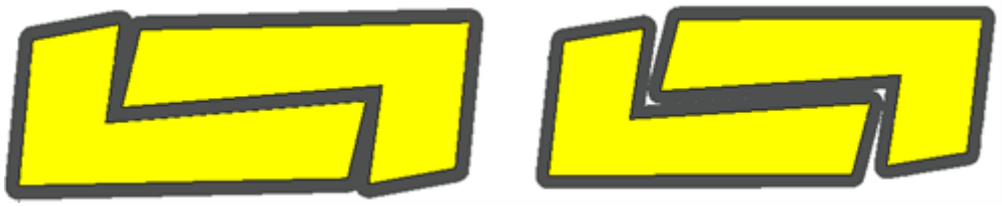


对于板材内的每个区域，也可以区分所使用的边距和直径（参见 IniGeometry 函数）。除非另有说明，否则本文档假定一张板材的所有内部轮廓使用单一分配。

放置间距

相邻放置之间*的可能最小间*距为 (CutterDiameter+OverrunSecurity)：

- OverrunPolygon设置管理支持部件切割区域超限。图中，超限（左侧）和不超限（右侧）的放置示例：灰色轮廓对应切割刀具路径



- OverrunSecurity设置赋值*安全距离*添加到放置，应用部件切割区域超限
- MarginInner设置赋值相邻放置之间的添加间隔。

现在我们看一些可能设置的示例：

CutterDiameter	*OverrunPolygon*	*OverrunSecurity*	*MarginInner*	放置间距
10.0	False	0.0	0.0	20.0
10.0	False	0.0	20.0	40.0
10.0	True	0.5	0.0	10.5
10.0	True	0.0	20	30.0

在*True Shape*优化的情况下，相对于为部件分配不同直径或变量几何形状而言，该表被简化了。在此情况下：

- 切割外形之间的最大重叠等于部件赋值较小直径的一半（所有可以放置的）
- 部件之间的最小距离可以增加一个间距，以补偿应用弦差*后轮廓段的变化*。

部件外部整体尺寸

对于部件，可以赋值四个解释为外部尺寸，按侧面区分，将添加到部件外形的字段：

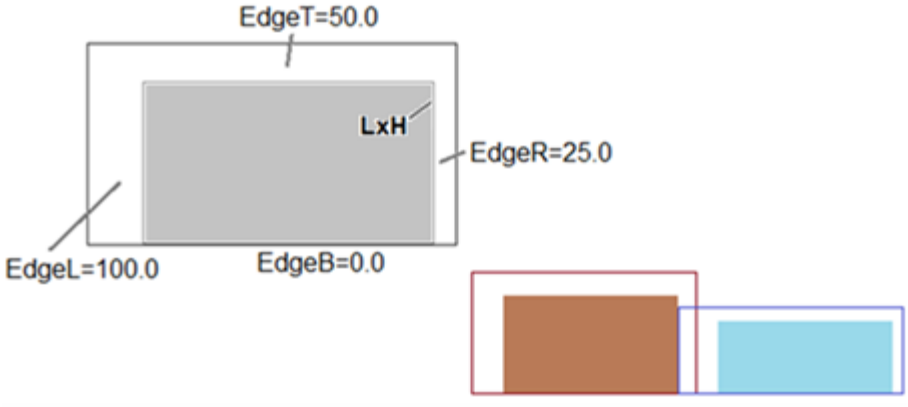
- 赋值在字段(*EdgeL*、*EdgeR*、*EdgeB*、*EdgeT*)，NestPart结构
  - 每个数值分配一个沿坐标轴（X 或 Y）测量的线性尺寸。
- 这些整体尺寸赋值区域视为报废：
  - 相邻外部尺寸重叠
  - 所有外部尺寸排除有用放置。

在*True Shape*优化中，通常相比*Square*优化过多应用外部整体尺寸：

- 如果部件赋值非矩形几何：使用单个数值，该数值等于所设置的最大尺寸。
- 增加部件与板材边缘的间隔。

使用外部整体尺寸是需要*方形*优化通常矩形部件的应用的特点。

在图的上部，部件具有分配的全部不同的外部尺寸（由LxH外形外部尺寸标识）。
在下部，两个部件之间最大水平接近示例和外边缘



对于部件对称和/或旋转，边缘*遵循*应用的变形。我们假定应用镜像X的部件情况：
- 字段（*EdgeL*、*EdgeR*）交换应用。

外部尺寸的含义完全是技术性的，取决于 TPA_N 库的具体应用。例如，执行与单个可放置部件相关的加工操作，需要使用该部件几何形状外部的空间。如前所述，外部尺寸的普遍使用应被视为需要对典型的矩形或类似矩形的部件进行“平方”优化的应用的一个特性。如果需要根据所需的优化来区分外部尺寸的用途，则客户有责任自行处理。

2.10 部件集群

TPA_N 可以激活部件集群用于放置而不是单个部件。

自动集群

自动*集群*将部件与旋转 180° 的副本匹配，生成为单个部件获得的组。
该函数只能在 *True Shape* 优化中管理。

图片左侧显示单个部件示例，右侧显示利用自动集群获得的*组*。

如果*组*的使用效率至少等于 *ClustersExpected* 属性赋值值，在自动集群模式下放置部件始终在单个放置之前，其中自动集群使用效率计算如下：

（单个部件面积 * 2 * 100）/（组长度 * 组高度）

对于每个部件，可以请求自动集群放置，对应 NestPart 结构中的 *AutoPair* 字段。自动集群机制需要部件可以旋转。

如果可行，TPA_N 应用尽可能减小部件整体尺寸的旋转，搜索最高效自动集群。
放置自动群集可以应用*组*的更多旋转，第一次尝试 90° 步进，后续步进减小（45°、30°…）。
还可以单独放置部件，无法在*组*中应用。

AutomaticCluster 设置赋值总启用以执行自动集群。

ClustersAbsolute 属性指定独立于相关部件和 *AutomaticCluster* 设置本身使用自动集群的效率：如果自动群集效率至少等于设定值，则始终激活群集放置。

自动群集应用可以强制激活或者在识别明显形状外形的情况不激活。例如：
- 以下情况不应用：矩形、圆形、椭圆、一些规则多边形
- 在一些规则多边形情况下强制应用。

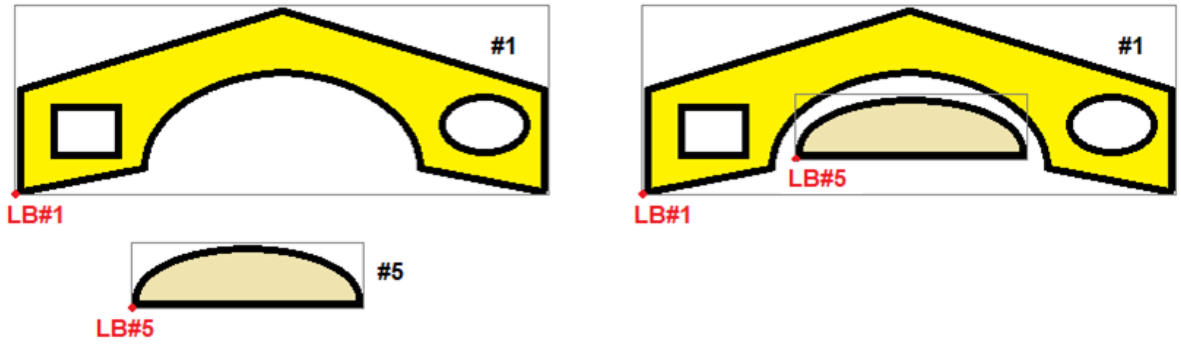
手动集群

此机制允许创建已经独立赋值的各个*部件*的集群：这种方式组合的部件将作为一个组放置，保持往复位置不变。
对于部件中的手动集群，TPA_N 创建所有部件的总图纸，包括可能的孤岛：然后将这样获得的图纸用作一个实体，放在板材上。

手动群集赋值在 *True Shape* 和 *Square* 优化之间控制，两个特定程序存在各自的差异。
下面将用*群集*表示通用手动群集。

图中表示两个部件：
- #1 部件具有 ID 1：主外形和 2 个孤岛
- #5 部件具有 ID 5：仅一个主外形。

为两个部件指示主外形边界矩形顶点 LB（左下角）。
右侧部件显示在相同表示平面，显示其往复定位。



集群的正确构造来自部件的正确赋值。例如：调用应用程序负责赋值部件，这样具有交叉或者几何不同。
实际考虑的部件只有启用的部件（NestPart 结构，字段 *Enable=true*）。

嵌套过程将手动群集作为单个部件处理：群集放置不是可选，而是必需的。
对于所有意图和目的，群集将理解为类似部件，但采用不同几何构造方法。
与单个部件赋值和使用有关的考虑可以扩展到群集（例如颗粒控制，应用转换，放置在自动群集中）：这也应用于接下来的考虑，除非另有指定。

手动群集通过特定方法和结构赋值。
在一个或多个群集组成中使用部件不排除获得直接放置的可能：在此情况下，我们讨论*剩余放置*。

和部件类似，群集可以赋值一组常规信息：
- 请求和最大可用数量
- 允许和请求转换（旋转，镜像）
- 匹配赋值（厚度、材料、颜色、纹理）
- 启用赋值（自动集群、孤岛放置、网格放置）
- 优先级。

群集的几何构造通过指示哪个部件对构成群集方案作出贡献来实现：
- 赋值的所有部件均可以使用
- 在构造一个或多个群集中，每个部件可以使用一次或多次
- 群集可以指示最多 100 次部件的独特使用
- 每次部件使用赋值要应用的变形：平移、镜像、旋转。

匹配组和筛选器

关于*部件*和*板材*之间匹配条件和/或筛选器的说法在*群集*和*板材*之间也有效。
*群集*和其中使用的*部件*之间不需要匹配，颗粒例外。我们了解一下：

- 赋值厚度 20 mm 的群集：每个部件可以在群集结构中使用，与部件厚度无关。尤其还可以使用不验证任何板材匹配的部件
- 赋值颗粒 X 的群集：仅当赋值 Y 以外的颗粒时，可以使用部件
- 赋值颗粒 Y 的群集：仅当赋值 X 以外的颗粒时，可以使用部件
- 赋值颗粒 OFF 的群集：仅当赋值 OFF 的颗粒时，可以使用部件

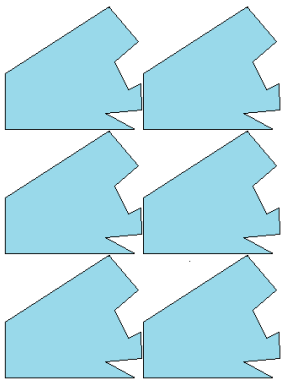
网格放置

对于一个部件，可以请求遵循网格模式的放置，对应 NestPart 结构中的 *AutoGrid* 字段。

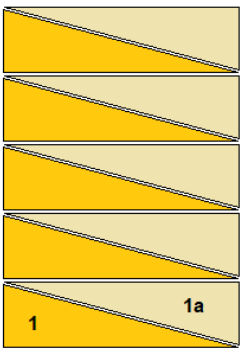
AutomaticGrid 设置赋值总启用以执行网格放置。
在 *True Shape* 优化中更有效地管理功能：在下方指示为 *True Shape* 优化的特权，但在 *Square* 优化中部分可以工作。

请求网格放置的工件在其他之前使用，按照*行*列*布置放置，考虑板材上的可用空间。要优化放置，可以应用自治集群策略，分析每个部件：
- 可以放置部件，重复可以对应始终以相同旋转重复的单个部件，或者应用自动集群的两个部件的单元
- 然后可以放置单个或两个部件的重复单元，评估 0° 或 90° 旋转变化。

下面是部件网格放置示例。



- 不应用自动集群
- 网格延展为：3 行 * 2 列



- 应用部件自动集群，配对部件 (**1;1a**)
- 网格延展为：5 行 * 1 列。

在 *True Shape* 优化中，网格放置应用可以强制激活，或者在识别明显形状外形的情况不激活。
在 *Square* 优化中，网格放置限制为仅选择的第一个元素类型。

部件组合

为了说明此功能，我们假设一个项目中的部件对应不同的最终产品。通常情况下，需要将同一产品的所有部件集中排布在同一张板材或连续的板材上。这是可以使用 *部件组合* 功能的典型场景。

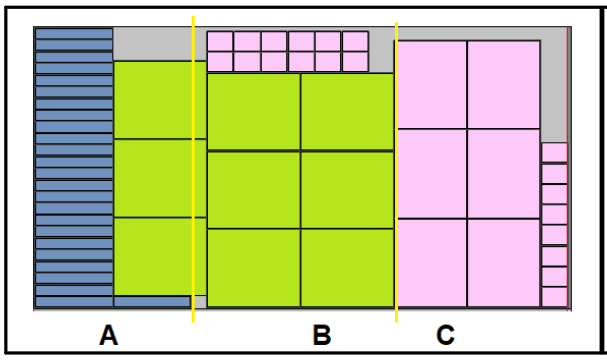
对于此功能，*Range* 字段在 NestPart和 NestCluster结构中设置，并且 UseRangePart 和DimRangePart设置也已设定。

具有相同 *Range* 值的部件和/或集群形成一个组，并且通过将可用区域划分为若干区域来进行放置，以便将组内的件聚集在一起：

- 可放置元素被排序，通过增加的组合值（0、1，……直到最大值）
- 顺序对应于排序顺序
- 并不保证，一个组的所有部件都将被放置在单张板材上。

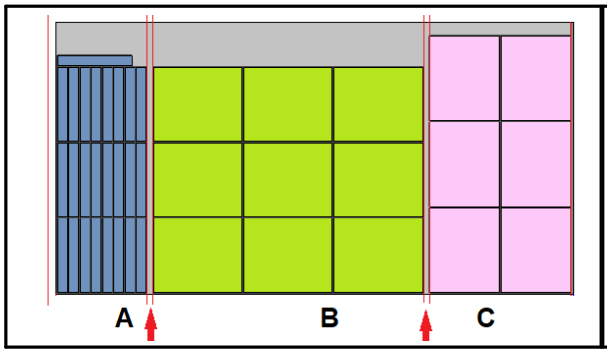
UseRangePart 设置分配该功能的总体授权，提供三个选项的选择：

- *0*：该功能被禁用（*Range* 字段的解析不激活）
- *1*：要求在连续区域内进行放置
  - 一个组的放置是连续的，但并不与前一个组的放置相分离，最多与前一个组的放置相混合
  - 涉及连续组放置的区域是相邻的，但没有明确的分隔。区域的相邻性遵循为嵌套选择的方向
    - *水平方向*：区域水平对齐
    - *垂直方向*：区域垂直对齐



图中显示了该功能：
A：图中：Range 元素 = 0 的区域极端（彩色部件：蓝色）
B：后续 Range 元素 = 1 的区域极端（彩色部件：绿色）
C：前一组合后续 Range 元素 = 2 的区域极端（彩色部件：紫红色）：range 部件也可以被放置在区域 B 中，但不能在 A 中

- *2*：要求在分隔区域内进行放置
  - 一个组的放置位于与前一个组相邻且分隔的区域内。DimRangePart设置设定相邻区域之间的间距。



图中显示了该功能：
- 指出了区域之间的两个分隔通道：此处通道是垂直的，对应于水平嵌套方向
- 通常假定分隔通道对应于板材随后的切割。

  - 所使用的设置可以追溯到前一个案例，在进行 True Shape 优化且使用非严格矩形板材的情况下。在这种情况下，遵循分隔区域内的放置方案是可能的，仅当板材被认为类似于矩形时。让我们看看验证一个通用板材类矩形条件的条件：
    - 仅有 1 个主区域（即：原始轮廓没有交集或极端收缩）
    - 不分配孤岛
    - 如果它不识别已知形状（矩形、椭圆、圆），它必须是凸的，并且总面积与板材整体矩形面积的比值必须至少为 0.75。

UseRangePart 设置被应用，仅当一个项目需要不同的*Range* 字段值时。

元素组合仅应用于具有严格为正的所需使用数量的部件和/或集群（NestPart和NestCluster结构中的 N 字段）：具有仅额外放置的元素被排除（参见段落：填充放置）。

一个放置的组合也包括任何为该组元素要求的额外放置（参见段落：额外放置）：

- 一个组的额外放置可能填满一张板材，但不开始下一张：因此，一个组的额外数量的总使用是不被保证的。
- 它们的使用不评估 ExtraFiller 设置。

任何填充放置在所有其他内容之后被评估，并影响板材的整个区域，但基于应用的 *UseRangePart* 值可能存在差异：

- 对于在连续区域内的放置：放置被放置在任何地方
- 对于在分隔区域内的放置：放置遵循区域之间的分隔通道。

2.11 部件和板材重复

*部件*和*板材*的正常放置条件与具有更多启用元素的列表一致（结构 NestPart 和 NestSheet 中的 *Enable* 字段）。
在此情况下：仅考虑赋值严格正请求/可用数量的元素（结构 *NestPart* 和 *NestSheet* 中的 *N* 字段）。
我们提到的所有内容指板材与部件之间的单个匹配组：如果存在多个组，则必须每个组应用考虑。

关于部件使用，对于群集放置有部分不同，请参考段落手动群集放置。

优化过程将工件放在最少可用数量面板上。如果该过程已放置可用数量的工件，对于其中一部分，设置**最大数量 > 请求数量**，该放置将填充已经使用的面板，直到最大设定值。
对于*部件的最大数量*字段管理，请参考主题额外放置。对于部件的最大数量字段管理，请参考主题额外放置。

我们将如下指示这种情况：*多部件*和*多板材*。
现在我们看看下面可以从此常规情况推导的具体情况。

从下面的段落可以看出，许多属性可以影响特定情况的识别，可能的不同解释归结于部件需要的部件。

如果对这些情况的解释不直观,则建议明确排除部件的使用,而不是将所需数量归零。

单部件和单板材

*部件*和*板材*列表只有一个启用元素。按照您的需求，可以选择特定优化：

NestPart.N	*NestSheet.N*	
0	0	该过程在 1 板材上放置最高数量的工件
0	> 0	该过程在总可用板材上放置最高数量的工件（所有得到的板材相同）
> 0	0	该过程计算放置所需数量的部件时所需要的板材数。如果为部件设置*最大数量 > 请求数量*，使用的最后一个板材填充至设置的最大值
> 0	> 0	该过程在最少可用板材上放置所请求数量的部件。如果该过程已放置可用数量，对于其中一部分，设置*最大数量 > 请求数量*，使用的最后一个板材填充至设置的最大值

多部件和单板材

*部件*列表具有多个启用元素，而*板材*列表只有一个。*可用工件数量*必须严格为正 (> 0)。
按照您的需求，可以选择特定优化：

NestSheet.N	
0	该过程计算放置所有请求部件时所需要的板材数。
> 0	该过程在最少数量的可用板材上放置最高数量的请求部件

在这两种情况下：如果该过程已放置可用数量工件，对于其中一部分，设置*最大数量 > 请求数量*，放置填充已经使用的板材，最多至最高设置值。

单部件和多板材

*部件*列表只有一个启用元素，而*板材*列表则有多个。*可用板材数量*必须严格为正 (> 0)。
按照需求，可以选择特定优化：

NestPart.N	
0	该过程在可用板材上放置最高数量的工件
> 0	过程在可用最少数量板材上放置要求的数量。如果该过程已放置可用数量，对于其中一部分，设置*最大数量 > 请求数量*该放置，放置填充已经使用的最后一个板材，最多至最高设置值

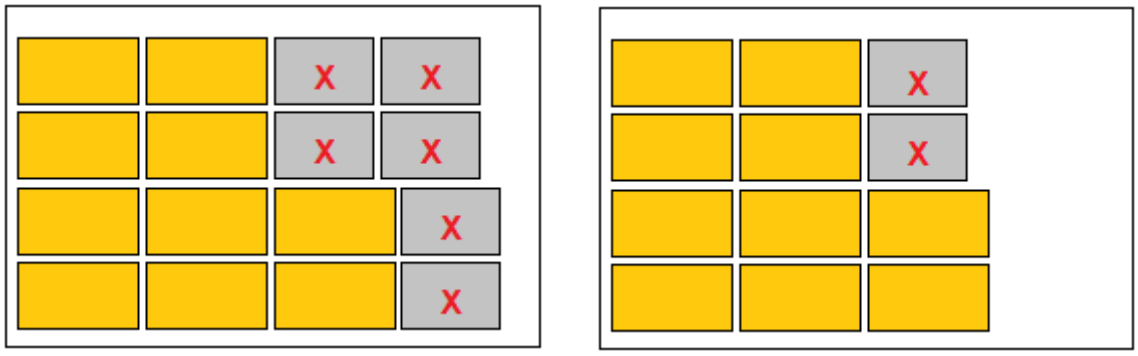
额外放置

除了实际要求的，还可以为部件和群集分配额外放置。
信息在NestPart结构的*Nmax*字段赋值。以下条件下使用该值：
- 为请求放置赋予严格正值（结构的（字段中）,或者UseOnlyExtraPart=true

值赋予的含义由*ModeExtraPart*设置决定（***boolean**类型*）：
- 如果是*ModeExtraPart=false*（*默认*）：则设置值大于其中一个要求的放置。在这些条件下，*额外放置*数量计算为：*(Nmax - N)*。
- 如果是*ModeExtraPart=true*则设置值为正，直接赋值额外放置数量。

除非另有声明，否则在文档中，以下情况视为有效：*ModeExtraPart*=false,*UseOnlyExtraPart*=false。
在板材上，只有检查无法放置更多请求部件后，才能使用额外放置。
根据这里的描述，可以清楚了解在任何情况下，都无法仅利用板材应用额外放置。
使用额外放置，可以选择两个模式，如设置ExtraFiller：
- *False*：应用额外放置填充板材
- *True*：仅应用额外放置填充请求部件已经使用的长度或高度。根据放置的进料方向选择填充方向（参见方向设置）：
  - 如果水平：填充应用于板材长度，不应用高度限制
  - 如果垂直：填充应用于板材高度，不应用长度限制

图片显示用不同属性值得到的结果：
- 假定使用：*Direction=0*，*Corner=0*
- 额外放置用'X'标记
- 左侧，*ExtraFiller=False*
- 右侧，*ExtraFiller=True*



有时仅需将ExtraFiller设置应用于最后一张板材，而非所有板材。这可以通过使用ExtraFillerLastSheet函数来实现。

填充放置

让我们考虑一些仅要求额外放置（在NestPart结构中：（=0,Nmax>0）的部件要点，这些部件也被称为*填充放置*。
只有当*UseOnlyExtraPart*=true时，才能使用这些部件。

涉及的部件始终在*所有可放置元素之后被使用*：
- 部件相互之间的使用顺序遵循常规排序准则（优先级、尺寸、类型、数量等）。
- 部件的使用会应用分配给板材填充的属性（*ExtraFiller*，*ExtraFillerLastSheet*）。

主要好处可能是向嵌套方案中添加放置，以减少浪费。例如，客户可以将相关部件分配到一个单独的列表中，以便根据需求使用该列表*来填充*嵌套方案。

手动群集放置

在手动群集中使用元素需要考虑一些其他因素。要求的数量赋值现在在*NestCluster*结构，*N*和*Nmax*字段中：
- 对于群集，不管理空请求数量（*N*字段 = 0）：在此情况下，自动排除群集
- 如果在群集中使用所有部件，赋值空可用数量（*NestPart*结构的*N*字段等于 0）

- 可以为群集放置的数量是 *NestCluster* 结构的字段（*N* 和 *Nmax*）中赋值的数量
- 群集结构中使用的部件无法用于直接放置

- 否则：为群集赋值的数量（*N* 和 *Nmax*）与为单个部件赋值的数量比较
  - 可以为群集放置的数量受部件数量限制，极有可能不应用相同群集
  - 如果部件可用数量足，则将相同部件用于直接放置。

在计算以上情况时，有必要牢记一些一般性问题：
- 部件可以在一个群集中使用多次，也可以在不同群集中使用
- 按照初始标识符顺序扫描群集列表，每次*预约*每个部件需要的数量：这可能导致接下来的群集数量不足。

特定群集使用可对应我们已经报告的情况：为群集中使用的所有部件赋值空可用数量 (= 0)。部件列表实际赋值可用形状库，而实际嵌套项目直接在群集上赋值。

我们用一些示例来说明。

A. 为群集赋值所需数量：*N=10*、*Nmax=20*。群集使用 3 个部件：
   - ID=2 的部件：*N=5*、*Nmax=10*
   - ID=3 的部件：*N=7*、*Nmax=10*
   - ID=5 的部件：*N=5*、*Nmax=15*

   - 可以为群集放置的数量为：*N=5*、*Nmax=10*
   - ID=2 的部件无法用于直接放置
   - ID=3：*N=2*、*Nmax=0*
   - ID=5 的部件只能用于直接放置额外类型（*Nmax=5*）。

B. 赋值和上一个示例相同的群集，但使用的部件数量不同：
   - ID=2 的部件：*N=0*
   - ID=3 的部件：*N=7*、*Nmax=10*
   - ID=5 的部件：*N=5*、*Nmax=15*

   - 无法放置群集
   - ID=2 的部件只能用作带板材的匹配组中的单个部件
   - 其余部件可以用于基于设置数量的直接放置。

C. 赋值和上一个示例相同的群集，但使用所有空部件的数量：
   - 可以为群集放置的数量为需要的数量：*N=10*、*Nmax=20*
   - 部件无法用于直接放置。

对于群集中使用的部件剩余放置，根据部件确认的匹配组使用每个部件。我们一起来看一个示例：
- 赋值部件厚度 = 20.0 mm
- 使用群集中的部件，应用群集厚度
- 使用直接放置中的部件，应用部件厚度。

在评估簇本身的实际布局可能性之前，需要先确定簇中所用零件的剩余布局。如果簇中所有必需的布局均未使用，则各个零件的剩余布局不会改变。

2.12 增量嵌套

Tpa_N 允许您在已分配放置的情况下开始优化。

有两种可用方法，也可以组合使用：
- 直接在单张或多张板材上输入放置列表（预放置）
- 将已放置的部件作为约束区域输入到单张或多张板材中

从外部视角来看，这两种方法的特性截然不同，显然它们对应着非常不同的需求和场景。

在 Tpa_N 中，初始放置方案可以以不确定的方式生成：
- 通过手动启动一次嵌套（在单张或多张板材上），然后请求自动完成该程序
- 从库生成的嵌套开始，随后在单张或多张板材中手动修改，然后请求在修改后的板材上完成该程序
- 由其他软件生成、且需要通过添加部件和/或手动集群和/或板材进行集成的嵌套
- 上述情况的混合场景。

预放置

可以从嵌套项目中已分配的部件和/或集群中，指定一系列手动集群或部件的预放置列表。

预放置：
- 使用特定的方法和结构进行分配。
- 应用于嵌套项目中指定的特定板材。
- 使用嵌套项目中已分配的部件或集群，并相应减少剩余可用数量。
- 使用几何变换（平移、镜像、旋转）来分配放置，这些变换与嵌套程序计算出的放置完全等效。

预放置的有效性在优化启动阶段进行验证（参见：优化标准和优先级）。

以下是该功能所执行的检查项以及所提供的功能特性摘要：
- 可以在一张或多张板材上分配预放置。
- 带有预放置的板材必须分配可用的单位数量，并会作为首选板材被优先使用。
- 一个放置动作可以作用于单个部件或一个手动集群：不可使用额外的放置，且所需的几何变换必须符合元素的分配设定（是否允许旋转、镜像）。
- 放置必须验证其与所分配板材的对应关系（参见：材质、颜色、厚度、纹理约束等）。
- 在使用 True Shape 优化时，可以使用部件的孤岛（前提是该区域未被其他预放置占用）。
- 任何完全超出对应板材可用区域的放置都将被删除。
- 无法要求系统对某个预放置的实际可用性进行预先评估。

预放置的定位是通过NestPlaced结构完成的：
- 坐标 (X,Y) 绝对应用于板材，参考板材整体矩形（边界框）的 LB（左下）顶点。
- 镜像和旋转使用与库计算出的放置相同的标准进行解释。

调用应用程序有责任确保板材上的预放置不会产生不必要的重叠。

当与部件分组结合使用时，预放置的分配可能相当不兼容，但在这种情况下，仍由客户端负责过滤或报告此类情况。
如果这两种功能共存，Tpa_N
- 会将板材的预放置分配给为该板材管理的第一个分组。
- 通常做法是限制或排除某些计算优化程序，从而导致嵌套效率降低。

预放置出现在嵌套结果中。

板材上的约束区域

现在，板材上已经包含的放置被作为板材自身的废料区域插入，并带有其自身在外形尺寸和刀具尺寸方面的工艺特性。

每个放置都被分配为废料几何形状（参见方法：IniGeometry），并且 Tpa_N 会避免在这些区域内放置任何项目。

此处分配的放置是项目部件列表之外的补充，并且不会出现在嵌套结果中。

2.13 TPA_N 中的优化

Tpa_N 中的优化的主要目的是在给定条件下（*常规设置*），最充分整体利用可用材料（*板材*）和请求放置（*部件*）。

优化通常执行几次*尝试*，选择最充分利用的尝试结果，即*最佳求解*。
不同尝试修改一些*参数*，以产生不同放置的情况。
我们通常的*参数*定义具有多种，可以与非常复杂的方面相关。一些更直接理解的*参数*示例：
- 部件放置顺序变化
- 板材填充逻辑变化
- 部件旋转角度变化
- 嵌套方向变化（水平或垂直）。

决定这些参数变化的模式是一个重要方面：
- 确定性

- 随机。

确定性模式执行的变化可以重复，不随时间变化。
随机模式执行的变化可以重复，但存在概率形。

目前，这两种优化都只应用确定性的变化。

对于*True Shape*优化，正在研究可以实现甚至随机变化的修改。

Square 优化

*Square*嵌套过程用于得出可重复确定的求解定义，保持可能影响其延展的所有初始设置不变。

研究这种优化尝试以检查任何可能的改进方法。

确定求解的设定时间不限制预定义尝试次数。

尝试次数是有限的。

在进行 Square 优化的情况下，可以使用 RetrySquare 在两种不同的操作模式之间进行选择：设置：

- *false*：确定求解的设定时间不限制预定义尝试次数。

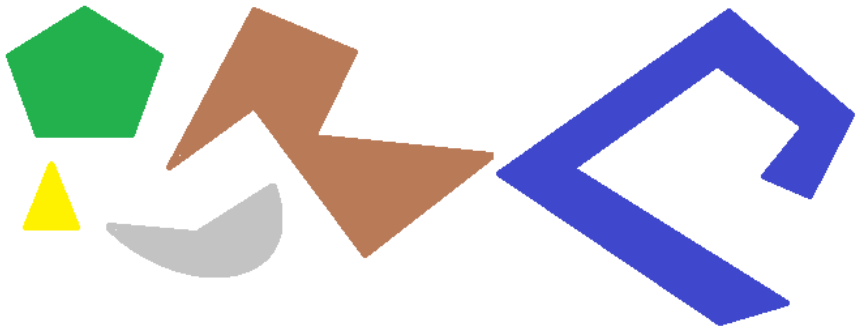
- *true*：受时间限制的影响，优化请求可能无法穷尽所有可能的尝试次数。

在第二种情况下，可以通过后续的尝试来计算更多的方案。

除非另有说明，否则本文档假定：*RetrySquare* =true 时，才能使用这些零件。

True Shape 优化

True Shape 嵌套过程对具有部分随机可变性的求解条件应用变化，因此得到可变求解定义，即使保持可能影响其延展的所有初始设置不变，随着时间推移也不一定可以实现。随机性通常采*用伪随机*方式应用：这意味着对于每次随机变化，给出此类条件以促进向更好的求解合并，事实上，限制了可能的变化数量。
随着该放置将*自由、凹面或凸面形状与具有可利用的内部孔在数量和可变旋转可能性进行关联*，现在嵌套求解变得复杂化了。形状之间互锁的可能性几乎没有穷尽。图形建议 5 种不同形状。



无疑，按类型放置具有 90° 步进旋转并占据更小更紧凑区域的单个形状的问题并不简单：每个形状具有 4 种可能放置求解（0°、90°、180°、270°），必须为其余图的每个可能放置和每种可能配对组合求解。
现在我们为每个形状赋值多个重复（例如 10）：每个形状重复必须与任何其他可放置的形状重复一起评估，总共 50 个形状。
现在我们假定角步长 45°：每个形状现在具有 8 个可能放置求解（0°、45°、90°、□ 315°）。
很明显问题变得非常复杂：普遍认为，求解此类问题不能单纯依靠推断，还必须依靠*随机性*。

这样做的第一个后果是无法保证将重复建议求解。另一个后果是新尝试始终可以改进求解：所以需要设置时间限制，可以添加更多时间以确定求解，从上一个确定的开始，从而能够评估不同求解。

如前所述，目前甚至连随机评估都被排除在外。为确定替代解决方案而进行的尝试应用了可重复的更改，因此也产生了可重复的解决方案。可计算的解决方案数量取决于多种因素，但目前数量有限。事实上，时间问题现在至关重要：完成所有必要的尝试可能需要相当长的时间，因此有必要设定一个时间限制。

连续优化

利用*True Shape*优化，可以计算更多求解，最多20个。

通过 Square 优化方案可获得的求解数量始终少于 20 个。

对于第一个优化，可以选择两种模式（参见函数：Compute）
- 在可用时间内返回计算出的最佳求解
- 返回计算出的第一个求解，并激活*StepByStep*功能。

对于接下来的优化，也可以选择不同模式（参见函数：RetryCompute）
- 按时间或按步骤
  - 定时：优化按照最大赋值时间停止
  - 逐步进行：返回计算出的第一个求解，并激活*StepByStep*功能。
- 仅当某个解优于前一个求解时才返回该求解。

如果满足下面的条件，可以计算接下来的优化：
- 已执行了第一次嵌套优化
- 所有设置保持不变（设置、部件、群集和板材列表）
- 已经计算不到20个优化
- TPA_N根据自己的可行性和有效使用性评估，没有独立停用该可能。

每个连续优化：
- 从上次执行优化留下的状态开始
- 正如所说，无法保证比上一个*更好*。

所有计算的优化保持可用，直到：
- 新总体优化（参见函数：Compute）
- 请求删除找到的求解（参见函数：ClearSolution）。

最佳求解

前面已经提到，优化*求解*是优化阶段已经执行的所有尝试求解之间的选择结果。完成的尝试次数根据不同因素变化。

在*Square*优化：
- 尝试次数由过程定义
- 赋值的优化时间通常足以运行预计尝试
- 或者，可以请求渐进式嵌套优化，如下文针对 True Shape 优化案例所示。

在*True Shape*优化：
- 第一次优化时（函数：*Compute*）即便不需要分步功能，仅在不同过程情况下尝试一次，以在最短时间内返回求解
- 连续优化时（函数：*RetryCompute*），运行模式由同一函数选择：
  - *按时间*：激活更多尝试，最多到可用最大时间。时间几乎用完后，该过程返回已经完成的尝试中计算的最佳求解。无疑，结束优化的最短时间是完成一次尝试需要的时间
  - *按步骤*：仅激活一次尝试。

每次优化返回对优化视为最佳的求解。这意味着连续的优化可能会得出不一定优于已计算出的求解。

运行 RetryCompute 函数时，您可以选择以下两种选项：

- 如果优化成功，则计算出的解决方案将成为当前解决方案。
- 只有当计算出的解决方案被评估为优于之前的解决方案时，它才会成为当前解决方案。

但是如何判断更好的求解？

答案并不总是显而易见，需要的评估标准可能在两个不同优化之间部分不同。
我们来看一些非常通用的标准。下面列出的点按照输入顺序应用，如果对上一个点无法操作更好的选择，则进入下一个点：

- 最大的放置区域优先。排列工件占据93.0%板材的求解优于88.00%填充的求解；
- 优先沿所选方向排列的求解：对于水平方向，沿Y轴，对于垂直方向，沿X轴；
- 优先“最整洁”的求解（根据放置工件边界矩形内的切余比较进行评估）

- 除了放置工件和内部切余的数量与大小，还应用工件排列的其他评估标准。

对上面列出的每个点应用相关可变权重，具有一定冗余边际，部分固定和部分调整以适应具体嵌套项目，从而允许组合评估最大数量的求解。
非常实际的示例：放置区域之间比较不是绝对的(92.7 < 93.0)，而是除了切割钻头直径外，还应用根据待放置工件最小尺寸评估的冗余区域。

优化标准和优先级

提出优化请求后，TPA_N开始分析赋值列表，进行优化阶段。
列表分析可能遇到错误，后果是取消优化。

我们考虑第一个分析阶段，区分部件、群集和板材。

*部件*列表检查：
- 未启用部件从优化排除（*NestPart*结构的*Enable*字段）
- 没有对应有效*板材*并且未在群集中使用的*部件*不优化（例如：*Fiber=1*的部件，没有相同设置的板材）
- 部件的一个或两个边界框尺寸小于最小分辨率值的，将被排除在优化之外。（常规设置 MinResolution）。

检查*群集*列表：
- 未启用群集（*NestCluster*结构中的*Enable*字段）或具有空请求数量（*NestCluster*结构中的*N*字段：具有值0）不优化
- 不对应有效*板材*的群集不优化（例如*Fiber=1*的群集，没有板材具有相同设置）
- 分配部件数量不足（少于2）或使用的部件未被标识或排除在外或在最小所需数量中不可用的群集，则不优化
- 一些部件检查会导致排除集群：
  - 带纹理（例如X）的群集无法使用具有不同纹理（例如Y）的部件
  - 不带纹理的群集无法使用带纹理（X或Y）的部件。

实际上，所指出的一些情况不一定导致简单排除群集，而是导致错误，阻止过程继续。可以分配常规设置，自动恢复错误情况，从而排除群集FixComputeError=*true*。

*板材*列表检查：
- 未启用板材从优化排除（*NestSheet*结构的*Enable*字段）
- 一个或两个边界矩形尺寸小于最小分辨率值 *50.0的板材从优化排除（常规设置MinResolution）
- 不对应有效*部件*或*群集*的板材不优化（例如*Fiber=1*的板材，没有部件或群集具有相同设置）。

报告分析的结果必须能够为每个部件和板材列表使用至少一个元素：
- 对于单元素，甚至可以请求空请求/可用数量（结构*NestPart*，*NestSheet*中的*N*字段），识别单个部件和/或板材情况
- 否则，具有空请求/可用数量的元素从优化排除。

检查*初步放置列表*：
- 与部件没有有效匹配的放置将被排除在优化之外（例如：旋转的放置与不可旋转的部件）
- 与板材没有有效匹配的放置将被排除在优化之外（板材未分配、未启用、可用数量不等于 1，或材料/颜色/纹理不兼容）。

即使在初步放置的情况下（与手动集群类似），上述情况并不一定会导致简单的放置排除，而是会导致阻止程序继续运行的错误情况。通过将通用设置FixComputeError=*true*，可以从错误情况中自动恢复，从而导致受影响的放置被排除。

嵌套过程首先应用特定标准，赋值部件、群集和板材的使用顺序，两种优化模式之间存在一些区别。

使用板材

板材使用遵循一个顺序，可以考虑不同情况：
- 首先对已分配初步排名的表格进行排序
- 先排序标记为废料的板材（*NestSheet* 结构中的 *IsScrap* 字段）
- 按优先级降序/升序排序（如果*UseOrderSheet=true*）
- 按增加类型排序（*NestSheet* 结构中的 *ID* 字段），条件和之前的点相同。

常规设置 UseBeforeScrap 支持应用*板材*鉴定字段作为废料。
常规 UseOrderSheet 设置支持应用*板材*优先级。
常规设置 ModeOrderItem 指定优先级解释规则：

- *False*：减小优先级进行排序（更高优先级的板材优先）
- *True*：增加优先级进行排序（更低优先级的板材优先）。

使用部件（Square）

部件包括可单独放置的部件和群集。

使用部件遵循一个顺序，可以考虑不同情况：
- 通用设置UseRangePart启用应用*部件*分组的功能
- 常规设置UseOrderPart支持应用*部件*优先级
- 常规设置ModeOrderItem指定优先级解释规则（参见上文）
- 常规设置*AutomaticGrid*允许网络放置*部件*。

现在我们看一看应用于初始部件排序的主条件。按照显示的顺序应用点，如果对上一个点无法进行之前的选择，则进入下一个点：

- 通过增加的分组值进行排序（如果*UseRangePart <0*)
- 减小/增加优先级进行排序（如果*UseOrderPart=true*）
- 具有网格放置请求的部件（如果*AutomaticGrid=true*）
- 在组件部件之前放置群集
- 通过面积递减进行排序（大件优先）。面积也与嵌套方向相对应地被评估：
  - 如果为水平方向：通过递减的高度进行排序
  - 如果为垂直方向：通过递减的长度进行排序
- 无法旋转的部件
- *矩形*部件
- 具有更大周长的部件
- 更大请求数量的部件
- 更大类型的部件（*NestPart*结构的*ID*字段）：群集ID按照升序追加到部件ID。

仅具有额外数量的部件位于排序过程的末尾。

与面积、尺寸、周长比较相关的评估实际比列表显示更加复杂：应提供常规框架。

使用部件（True Shape）

*部件*包括可单独放置的部件和群集。
使用部件遵循一个顺序，考虑相比*Square*情况更多的不同情况。

而且在此情况下：
- 常规设置UseRangePart 启用部件分组的*应用*
- 常规设置UseOrderSheet支持应用*部件*优先级
- 常规设置ModeOrderItem指定优先级解释规则
- 常规设置AutomaticGrid允许网络放置*部件*。

深入研究可能情况前，应明确，赋值的部件顺序现在仅视为初始情况，可能随优化尝试进行而变化。

现在我们看一看应用于初始部件顺序的主条件。下面处理的点按照输入顺序应用，如果对上一个点无法操作更好的选择，则进入下一个点：

- 按分组递增排序（如果*UseRangePart* > 0）
- 减小/增加优先级进行排序（如果*UseOrderPart=true*）
- 具有网格放置请求的部件（如果*AutomaticGrid=true*）
- 在组件部件之前放置群集
- 具有凹面形状或孤岛的部件，可能在凹面区域或孤岛内包含
- 更大面积部件
- 计算已知形状（矩形、多边形、圆形…）
- 更少旋转可能的部件
- 更大请求数量的部件
- 更大类型的部件（*NestPart*结构的*ID*字段）：群集ID按照升序追加到部件ID。

仅具有额外数量的部件位于排序过程的末尾。

2.14 管理嵌套项目支持

嵌套项目作为分配到 Tpa_N 库的所有信息集：
- 整体启用设置
- 部件、板材、手动群集列表。

Tpa_N 公开嵌套项目的序列化方法。这些方法对于测试情况尤其有用：
- 嵌套情况
- 集成 Tpa_N 情况，其中列表分配结构中提供的信息足够。

但是，通常集成 Tpa_N 需要此情况下的自定义。

我们看一看在 Tpa_N 中工作的嵌套项目的序列化方法，尤其对于整体启用设置。
默认所有设置直接与嵌套项目相关：
- 上面*常规功能*段落中的组除外
- 将项目保存到文件并加载，包含所有设置。

操作类型对于测试情况非常有用：需要测试特定优化，项目文件必须配备所有设置。因此，建议对于 Tpa_N 库的外部应用提供类似功能。

Tpa_N 还公开仅用于常规设置的序列化方法。

2.15 库的已知限制

毫无疑问，对于在数量、尺寸和形状方面均有限的部件，手动嵌套甚至可能取得比 Tpa_N 更好的结果。然而，只要给定案例的复杂度稍有增加，情况就很容易反转。 将拼图重新拼凑在一起的能力也并非 Tpa_N 的特权。
本节包含 Tpa_N 已知限制的列表。

赋值部件
- 可以赋值最多500个不同*部件*
- 对于每个*部件*，可以赋值的最大可放置数量为999
- 对于每个*部件*，最多可以分配 100 个孤岛
- 部件或部件孤岛的描述轮廓最多可以分配 10000 个元素
- 所有赋值*部件*和*群集*要求的最大总放置数量为99999。

群集赋值
- 可以赋值最多500个不同*群集*
- 对于每个*群集*，可以赋值的最大可放置数量为999
- 所有赋值*部件*和*群集*要求的最大总放置数量为99999
- 一个*群集*可以赋值最少2个部件，最多100个
- 使用手动集群不是可选的（即：您不能选择是使用手动集群还是单独使用其内部的部件）

赋值板材
- 可以赋值最多100个不同*板材*
- 对于每个*板材*，可以赋值最大可使用数量999
- 对于每张板材，您最多可以分配 100 个余料区域
- 板材或余料区域的描述轮廓最多可以分配 10000 个元素

初步放置的分配
- 最多可以分配 999 个初步放置

***Square**优化*
- 放置已赋值纹理的*部件*不会应用*RctMinimize*设置为激活（如果将其放在已赋值纹理的板材上）
- 仅部分求解*网格放置*。

***True Shape**优化*
- 并列赋值给*部件*的外部边距。

优化效率
有许多因素会导致降低嵌套优化的理论效率。 让我们来看看主要的几个：
- 部件聚合为手动集群：放置单个部件可能会提高效率
- 部件分组：效率损失不仅来自于被迫分组成组，还来自于自动优化实践的受限或取消（例如：改变方向的优化）
- 为部件分配外部边距

- 纹理匹配施加的限制
- 限制使用部件的凹陷区域
- 分配额外放置：虽然是为了响应特定的需求，但与相同数量的正常放置相比，对应额外数量的放置会导致效率降低。

3 库函数指南

本章详细介绍 TPA_N 的属性和函数。

3.1 许可证管理

IsSquareEnabled

Boolean 属性测试基本函数键的存在和状态（*Square* 优化）。

值
如果启用模块则为 *True*，否则为 *False*。

注
查询此属性作为实际键读取运行，但仅当没有嵌套优化运行时。
如果基本函数键无效，则无法执行优化。

IsShapeEnabled

Boolean 属性测试高级函数键的存在和状态（*True Shape* 优化）。

值
如果启用模块则为 *True*，否则为 *False*。

注
查询此属性作为实际键读取运行，但仅当没有嵌套优化运行时。
如果高级函数键无效，则无法执行部分库操作。

3.2 常量

TpaNestingOEM.Nesting类提供以下常量：

MAX_ROW_ITEMS	500	嵌套项目中可分配的最大零件类型数量
MAX_ROW_N	2000	为部件请求的放置最大值
MAX_ROW_ITEMSxN	99999	请求的总放置（部件和集群）最大值
MAX_SHEET_ITEMS	100	嵌套项目中可分配的最大板材类型数量
MAX_SHEET_N	999	板材可用的放置最大值
MAX_CLUSTER_ITEMS	500	嵌套项目中可分配的最大集群类型数量
MAX_CLUSTER_N	2000	为集群请求的放置最大值
MAX_ROW_INCLUSTER	100	集群分配中添加的部件最大数量
MAX_PLACED_N	999	嵌套项目中可分配的初步放置的最大数量
MAX_ITEMS_INGEO	10000	一个轮廓的几何项的最大数量
MAX_HOLE_N	100	部件的孤岛或板材余料面积的最大数量
SOLUTION_NUMBER	20	可计算方案的最大数量

3.3 枚举

TpaNestingOEM 命名空间提供以下枚举。

NestErrors

赋值库管理的错误

- *ErrorNone*：无错误
- *ErrorLicense*：未检验的基本许可证
- *ErrorLicenseHight*：未检验的高级许可证
- *ErrorReset*：用户中断的优化过程
- *ErrorMemory*：内存错误

- *ErrorPartsEmpty*：空工件列表或未启用或可放置工件
- *ErrorPartsTomany*：请求过多部件放置
- *ErrorSheetsEmpty*：空板材列表或未启用板材
- *ErrorSheetsTomany*：过多请求板材
- *ErrorSheetsMatch*：无板材列表对应匹配
- *ErrorInGeometry*：错误赋值几何元素（岛屿太多，物品太多， 空直线，无效弧线…）

- *ErrorClustersTomany*：请求的群集放置过多
- *ErrorInCluster*：手动群集赋值错误（部件或最小数量不可用）
- *ErrorClusterMatch*：检查特定匹配时手动群集赋值错误（群集具有的赋值纹理与部件纹理不同）

- *ErrorPlacedTomany*: 初步安置请求过多
- *ErrorPlacedMatch*: 初步定位检查错误（板材和/或零件的可用性、板材和零件的对应关系等）

- *ErrorIOProject*：管理项目文件错误（路径和/或文件访问错误，格式无效）
- *ErrorIOfile*：错误管理临时文件（路径和/或文件访问错误…）或附件（文件：DXF）
- *ErrorNoneSolution*：未赋值求解
- *ErrorBusy*：组件繁忙优化
- *ErrorContext*：指示当前上下文无效
- *ErrorUnexpected*：常规或未管理错误。

3.4 结构

TpaNestingOEM 命名空间提供以下结构。

NestPart

常规赋值*部件*。结构显示用默认值初始化字段的方法。

- *int ID*: *部件*数字标识符（唯一，严格正）{*默认* = 0}
  - *部件类型*也用于字段

- *bool Enable*: 启用使用部件（False = 不放置部件）{*默认* = True}

- *string Label*: 部件的描述性名称（可以是 =""；最大长度 50 个字符）{*默认* =""}

- *double L*: 长度 (>= 0.0) {*默认* = 0.0}

- *double H*: 高度 (>= 0.0) {*默认* = 0.0}
  - 如果为*部件*赋值多边形几何（如圆、多边形、多曲线）可以保留（L，H）字段 0.0
  - 值采用以下单位：Unit

- *double S*: 厚度 (>= 0.0) {*默认* = 0.0}
  - 如果需要应用匹配相应*板材*字段，则为字段赋值区分值
  - 值采用以下单位：Unit

- *int N*: 请求的放置数量 (>= 0) {*默认* = 0；最大值 = 999}

- *int Nmax*: 最大可用数量{*默认* = 0；最大值 = 999}（参见 *ModeExtraPart* 常规设置）
  - *N* 字段设置部件的请求数量。*Nmax* 字段设置最大可使用数量，如果赋值严格正值则有效。在此情况下：

  - 如果 *ModeExtraPart*=*false*：*Nmax* 必须赋值大于 *N* 的值，(*Nmax* - *N*) = 填充已赋值 *板材* 的可使用数量
  - 如果 *ModeExtraPart*=*true*：*Nmax* = 填充已赋值 *板材* 的可使用数量

仅当尝试放置请求数量的所有 *部件* 类型后。对应应用 *Nmax* 字段的放置还报告为 *额外放置*

- *short Fiber*： 材料 (>= 0) {*默认* = 0}
  - 如果需要应用匹配相应 *板材* 字段，则为字段赋值区分值
  - 字段的真实含义属于外部应用

- *int Colour*： 常规颜色信息 (>= -1) {*默认* = -1}
  - 如果需要应用匹配相应 *板材* 字段，则为字段赋值区分值字段赋值。默认值 = -1，允许赋值实际对应颜色的整数值（在此情况下：0 可以对应 *黑色*）
  - 字段的真实含义属于外部应用

- *short Grain*： 纹理方向（0 = 无；1 = 水平；2 = 垂直）{*默认* = 0}
  - 如果需要应用匹配相应 *板材* 字段，则为字段赋值值 1 或 2。仅当可以遵循纹理本身方向（可能应用旋转）时，具有字段（1，2）的部件可以放在具有纹理的 *板材* 上；无纹理的部件可以应用在具有任意 *Grain* 字段的板材上

- *short Order*： 使用优先级 (>= 0) {*默认* = 0}
  - 具有更高/更低优先级的 *部件* 在嵌套求解中具有放置优先级
  - 现场应用由常规函数设置 ***UseOrderPart*** 控制
  - 解释字段的规则由常规设置分配 ***ModeOrderItem***

- *short Rotate*： 旋转（0 = 无；1 = 90°；2 = any）{*单位* = 度和分；*默认* = 0}
  - 值 2 对应以可考虑角度步长激活旋转的可能（通过 ***StepAngle*** 属性赋值），只能用于 *True Shape* 嵌套。对于嵌套 *Square*，解释和值 1 相同
  - 值 1 对应启用 90° 旋转：对于 *True Shape* 嵌套，对应 90° 步长旋转可能

- *short Mirror*： 镜像放置（0 = 无；1 = x；2 = y；3 = x+y）{*默认* = 0}
  - 此字段指定赋值给原始部件的转换

- *bool UseIsle*： 在部件孤岛放置的设置 {*默认* = False}
  - 仅当 *True Shape* 嵌套和为 *部件* 赋值孤岛的情况下，此字段有效
  - 现场应用由常规函数设置 ***PartInHole*** 控制

- *bool AutoPair*： 启用以 *自动集群* 模式应用 *部件* {*默认* = False}
  - 仅当 *True Shape* 嵌套时此字段有效
  - 现场应用由常规函数设置 ***AutomaticCluster*** 控制

- *bool AutoGrid*： 启用按网格延展应用 *部件* {*默认* = False}
  - 现场应用由常规函数设置 ***AutomaticGrid*** 控制

- *double PartDiameter*：部件特定切割直径 {*默认* = 0.0}

- *double IsleDiameter*：部件孤岛特定切割直径 {*默认* = 0.0}

- *short Range*：元素（部件和/或群集）之间的分组值 (>= 0) {*默认* = 0}
  - 该字段的应用受 ***UseRangePart*** 设置的限制。
  - 该字段的真正含义取决于其外部应用。

- *double EdgeL*, *EdgeR*, *EdgeB*, *EdgeT*：零件的外部尺寸 {*默认* = 0.0}

NestSheet

常规赋值 *板材*。结构显示用默认值初始化字段的方法。

- *int ID*： *板材* 数字标识符（唯一，严格正）{*默认* = 0}
  - *板材类型* 也用于字段

- *bool Enable*： 启用使用 *板材*（False = 不放置板材）{*默认* = True}

- *string Label*： *板材* 的描述性名称（可以是 =""；最大长度 50 个字符）{*默认* =""}

- *bool IsScrap*： 将板材标识为恢复（例如以前优化板材的废料）{*默认* = False}

- 此类*板材*在嵌套求解中具有使用优先级
- 字段应用由设置分配 ***UseBeforeScrap***

- *double L*: 长度 (>= 0.0) {*默认* = 0.0}

- *double H*: 高度 (>= 0.0) {*默认* = 0.0}
  - 如果为板材赋值多边形几何（如圆、多边形、多曲线）可以保留 (L, H) 字段 0.0
  - 值采用以下单位：Unit

- *double S*: 厚度 (>= 0.0) {*默认* = 0.0}
  - 如果需要应用匹配相应*部件*字段，则为字段赋值区分值
  - 值采用以下单位：Unit

- *int N*: 可用放置数量 (>= 0) {*默认* = 0；最大值 = 100}

- *short Fiber*: 材料 (>= 0) {*默认* = 0}
  - 如果需要应用匹配相应*部件*字段，则为字段赋值区分值
  - 字段的真实含义属于外部应用

- *int Colour*: 常规颜色信息 (>= -1) {*默认* = -1}
  - 如果需要应用匹配相应*部件*字段，则为字段赋值区分值字段赋值。默认值 = -1，允许赋值实际对应颜色的整数值（在此情况下：0 可以对应*黑色*）
  - 字段的真实含义属于外部应用

- *short Grain*: 纹理方向（0 = 无；1 = x = 水平；2 = y = 垂直）{*默认* = 0}
  - 如果需要应用匹配相应*部件*字段，则为字段赋值值 1 或 2

- *short Order*: 使用优先级 (>= 0) {*默认* = 0}
  - 具有更高/更低优先级的*板材*在嵌套求解中具有使用优先级
  - 现场应用由常规函数设置 ***UseOrderSheet*** 控制
  - 解释字段的规则由常规设置分配 ***ModeOrderItem***

- *double Border*: 外部边距 (>= 0.0) {*默认* = 0.0}
  - 如果需要为工作表使用不同的外部边距，请为该字段分配一个值（必须为正数）。
  - 值采用以下单位： Unit

- *double IsleBorder*: 应用于床单岛的边距 (>=0.0) {*默认* = 0.0}
  - 值采用以下单位： Unit

- *double IsleDiameter*: 用于片状岛屿的技术 (>= 0.0) {*默认* = 0.0}
  - 值采用以下单位： Unit

NestCluster

*手动群集*中的常规赋值：有关单个字段的详细信息，请参考 NestPart。

结构显示用默认值初始化字段的方法。

- *int ID*: *群集*的数字标识符（唯一，严格为正）{*默认* = 0}
  - 拼写*群集类型*还用于字段

- *bool Enable*: 启用*群集*使用（False = 不使用）{*默认* = True}

- *string Label*: *板材*的描述名称（可以 =""；最大长度 50 个字符）{*默认* =""}

- *double S*: 厚度 (>= 0.0) {*默认* = 0.0}

- *int N*: 请求的放置数量 (>= 0) {*默认* = 0；最大值 = 999}

- *int Nmax*: 最大可用数量{*默认* = 0；最大值 = 999}

- *short Fiber*: 材料 (>= 0) {*默认* = 0}

- *int Colour*: 常规颜色信息 (>= -1) {*默认* = -1}
- *short Grain*: 纹理方向（0 = 无；1 = 水平；2 = 垂直）{*默认* = 0}
- *short Order*: 使用优先级 (>= 0) {*默认* = 0}
- *short Rotate*: 旋转（0 = 无；1 = 90°；2 = any）{*单位* = 度和分；*默认* = 0}
- *short Mirror*: 镜像放置（0 = 无；1 = x，2 = y，3 = x+y）{*默认* = 0}
- *bool UseIsle*: 启用在群集孤岛中放置 {*默认* = False}
- *bool AutoPair*: 启用在*自动群集*模式中应用*群集* {*默认* = False}
- *bool AutoGrid*: 启用通过网格发展应用*群集* {*默认* = False}
  - 参见：*NestPart* 结构中的类似字段
- *short Range*: 元素（部件和/或群集）之间的分组值 (>= 0) {*默认* = 0}

ItemCluster

*手动群集*中的单个项常规赋值。结构显示用默认值初始化字段的方法。

- *string NameID*: 部件标识符
  - 数字（严格正数）：对应 *NestPart* 结构中的 *ID* 字段
  - 否则：对应 *NestPart* 结构中的 *Label* 字段
- *double X, Y*: 部件原始边界矩形 LB（左下）角的放置 (X, Y) 尺寸
  - 尺寸与 XY 笛卡尔坐标系相关
- *short Mirror*: 部件需要的镜像
  - 0 = 无；1 = x；2 = y；3 = x + y
  - 轴应用转换的 (X, Y) 位置
  - 先应用转换，然后再进行任何旋转
- *double A*: 部件需要的旋转角度（单位：度和分）
  - 符号赋值旋转：正=逆时针；负=顺时针
  - (X, Y) 是旋转中心。

NestSize

赋值*部件*整体尺寸。

- *double LX, LY*: 最小整体 (X,Y) 尺寸
- *double HX, HY*: 最大整体 (X,Y) 尺寸。

NestGeometry

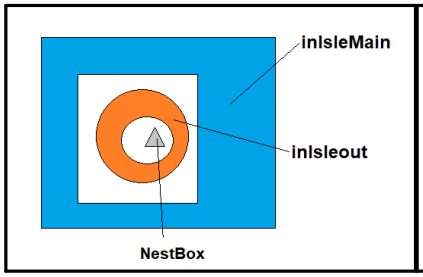
赋值多边形几何中的单个几何元素。

- *bool Isle*: True = 几何对应孤岛
- *int Type*: 几何的元素类型
  - *0* = 几何初始元素（(X;Y) = 多边形起点）
  - *1* = 直线元素
  - *2* = 顺时针圆弧
  - *3* = 逆时针圆弧
- *double X, Y*: 元素的最终 (X, Y) 坐标
- *double Xc, Yc*: 元素中心的 (X, Y) 坐标（如果圆弧）。

NestBox

常规赋值*板材*上的单个放置。结构显示用默认值初始化字段的方法。

- *int ID*: 对应放置的*部件*数字标识符
- *int Item*: 实习进展（>0）
- *double X, Y*: 部件原始边界矩形 LB（左下角）顶点的放置 (X, Y) 尺寸
- *short Mirror*: 放置需要的镜像
  - 0 = 无； 1 = x； 2 = y； 3 = x + y
  - 应用变换的轴穿过零件边界矩形的中心
    - 轴垂直于镜像 x
    - 轴与 y 轴水平
  - 先应用转换，然后再进行任何旋转
- *double A*: 对应放置的旋转角度（单位：度和分）
  - 符号赋值旋转：正 = 逆时针；负 = 顺时针
  - (X, Y) 是旋转中心
- *bool InIsle*: True = 放置在孤岛内（仅当 *True Shape* 嵌套时）。接下来的两个字段可以帮助我们重建岛屿的使用水平。
  - *int InIsleMain:* 渐进式最外层放置
  - *intIsleOut*: 紧邻外部位置的递增编号（可以等于 InIsleMain）



图中展示了岛屿中递归放置的一个示例

- *NestBox*: 表示最内层放置的部分（灰色三角形）
- *inIsleOut*: 它是 NestBox 结构中的一个字段，对应于紧邻的外部部分（带有圆形岛屿的橙色圆圈）的递增编号）
- *inIsleMain*: 它是 NestBox 结构中的一个字段，对应于最外层部分（带有矩形岛的蓝色矩形）的递进部分）

- *bool IsOver*: true= 位置对应于额外
- *bool IsInput*: true= 相当于初步安排
- *string InCluster*: 指示放置是否来自群集应用
  - "": 放置为单个
  - 否则：来自群集分解。Format："#;ID"
    - #：逐渐应用群集 (> 0)。具有相同值的放置来自相同群集放置
    - ID：群集的数字标识符（*NestCluster* 结构中的 *ID* 字段）。
- *doble Xl, Yl*: 放置最小整体尺寸的坐标 (X, Y)
- *double Xu, Yu*: 放置最大整体尺寸的坐标 (X, Y)

NestPlaced

在板材上分配单个初步放置的通用说明。 该结构公开了用于将字段初始化为默认值的方法。

- *bool Type*: 项目类型：false = 部件，true = 手动集群
- *int ID*: 与放置相对应的元素（部件或手动集群）的数字标识符
- *bool Enable*: 启用放置的使用（false = 不使用）{*默认*= true}
- *double X, Y*: 部件原始边界框左下角 (LB) 顶点的放置尺寸 (X, Y)
- *short Mirror*: 镜像请求
- *double A*: 与放置相对应的旋转角度（单位：度及小数度）

- 有关字段含义的详细信息，参见NestBox。

3.5 回调函数

TPA_N 无需管理可用回调函数。

Progres

允许管理优化器进展的函数。

void Progres (int nValue, ref bool bCancel, ref bool bPause)

参数
- *nValue*: 优化时间
- *bCancel*: 返回 *True* 以取消该过程
- *bPause*: 返回 *True* 以中断该过程。

注
管理事件允许取消或中断优化过程：
- 取消决定以完全取消该过程结束优化阶段
- 中断决定在完成第一个有效求解后结束优化阶段。

在这两种情况下，可能优化不立刻结束，因为安全终止模式仍激活。
事件连续发生间隔由 TimerProgres 属性决定。

3.6 功能定义

我们来看看如何赋值 Tpa_N 的所有函数：对于函数，也使用设置拼写。
在 Tpa_N 初始化时，所有设置赋值默认值，这里通过位置 {*默认* = …} 指示。

除非另有指定，本章介绍的所有赋值必须在章节（***IniSettings -> EndSettings***）内使用，*常规函数*组列出的属性除外。
以只读模式使用属性不受限制。

为了更加有组织性，赋值分为四组：

- *常规函数*：常规使用赋值
- *嵌套项目常规赋值相关函数*：具有嵌套项目常规自定义值的赋值
- *Square 嵌套相关函数*：应用于 *Square* 优化的赋值
- *True Shape 嵌套相关函数*：应用于 *True Shape* 优化的赋值。

IniSettings

此函数打开常规函数设置的赋值部分。

bool IniSettings (int Unit, bool Clear, bool Autoconv)

参数
- *Unit*: 线性单位 (0 = mm; 1 = inch) {*默认* = 0}
- *Clear*: True = 为设置赋值默认值
- *AutoConv*: True = 运行尺寸设置自动转换。

返回值
如果结果为正，则为 *True*，否则为 *False*。

注
函数返回 *True* 允许继续常规函数设置的全部或部分赋值。

- 如果*部件*、*群集*或*板材*赋值阶段激活：这将提前终止
- 如果 *Autoconv=true*：函数运行尺寸设置自动转换，但仅当 *Clear=false* 并且 *Unit* 相比当前单位改变时。对于部件或板材不自动进行任何操作。

对于赋值 *Unit*，赋值 0（零）：Unit=0 赋值 [mm] 单位；否则赋值 [inch] 单位。.
改变线性单位需要总体赋值：
- 尺寸设置（如果无法：*Autoconv=true*）

- *部件*、*群集*和*板材*列表。

如果 *Clear=true*：所有设置预设为默认值，一些保持不变的设置除外：FixComputeError、TimerProgres、DirectoryTemp, RetrySquare.

以调用 ***EndSettings()*** 函数完成赋值阶段。

函数返回 *False* 对应以下错误情况之一：
- 嵌套优化正在运行。

查询 *LastError* 属性，评估特定错误情况。

EndSettings

此函数打开常规函数设置的赋值部分。

bool EndSettings ()

返回值
如果结果为正，则为 *True*，否则为 *False*（调用不匹配 ***IniSettings***）。

常规函数

Version

String 类型属性返回库版本的属性。
字符串报告库的主要、次要、版本和修订编号。

LicenseType

Boolean属性，用于检查库所管理的许可证类型。
- false：库管理硬件许可证（USB 闪存驱动器）
- true：库管理软件许可证（库版本：4.0.0, 4.1.0）。

该属性可用于验证您操作环境的正确性。

ExpirationDateString

返回许可证到期日期的***String***属性。该日期采用计算机文化惯例格式化。如果识别到的是有限期许可证，则该属性值有效（即不等于“”）。空字符串对应的值可能表示未经验证的许可证或无时间限制的许可证。

IsSquareEnabled

Boolean 类型属性，检查基本许可证的有效性（*Square* 优化）。

IsShapeEnabled

Boolean 类型属性，检查高级许可证的有效性（*True Shape* 优化）。

LastError

NestErrors 类型属性，返回上次找到的错误。在可以诊断异常情况的所有过程中更新该值。

ErrorMessage

函数返回为报告错误赋值的消息。

string ErrorMessage (NestErrors Error)

参数
- *Error*：数值。

返回值
如果 *Error=NestError.ErrorNone*，函数返回 ""（空字符串）。
内部消息采用英语赋值。

FixComputeError

Boolean 类型属性，赋值/返回激活，自动解决赋值列表验证阶段遇到的错误情况。
该设置目前用于解决手动聚类和初步布局的列表检查问题，并避免报告错误：*NestErrors.ErrorInCluster*、*NestErrors.ErrorClusterMatch*, *NestErrors.ErrorPlacedMatch*。

TimeNesting

赋值/返回嵌套优化最大计算时间的 ***Integer*** 属性 {*单位* = 秒；*默认* = 30；*有效范围*：0/600}。
如果嵌套优化正在运行，则不赋值。

注意具体情况：
- 值 <= 0 或 = 600：它使用最大值 (600)
- 整数值 (1-20)：它使用最小值 (20)。

TimerProgres

Integer 类型属性，赋值/返回生成 ***Progres*** 事件的时间间隔 {*单位* = 秒；*默认* = 10；*有效性范围*：>= 5}。
如果嵌套优化正在运行，则不赋值。

值 (<= 0) 禁用 ***Progres*** 事件管理。

RetrySquare

Boolean用于分配/返回 Square 优化模式启用状态的属性 {*默认* = False}
- *false*：为确定解而设置的时间不会限制预定义尝试的次数。
- *true*由于应用了时间限制，优化请求可能无法穷举所有可能的尝试。

如果嵌套优化正在运行中，则不执行分配。
默认值对应于引入该属性之前的 TPA_N 版本（< ver4.0.1）中激活的模式。推荐值为：*true*。

DirectoryTemp

赋值/返回管理临时文件的文件夹的 ***String*** 属性 {*默认* =""}。
如果没有赋值有效文件夹（赋值空字符串或无法访问的文件夹），使用 TPA_N 库的存储路径。
赋值属性需要在部分 (***IniSettings -> EndSettings***) 中使用。

嵌套项目常规赋值相关函数

Unit

返回 *IniSettings* 函数赋值单位的 ***Integer*** 属性 {*默认* = 0}。

MinResolution

赋值/返回最小分辨率值的 ***Double*** 属性 {*单位*：Unit；*默认* = 0.1 [mm]；*有效范围*：1.0E - 5/1.0 [mm]}。
字段用于评估矩形（最小长度/高度值）或几何元素（如空线性段或无效圆弧）的有效性。

OneNestingDimension

Boolean 类型属性，赋值/返回两种不同板材的相关选择 {*默认* = False}：
- (*False*) 2D：板材赋值嵌套二维的平面部分
- (*True*) 1D：嵌套在线性支撑上，板材赋值棒、外形、管。

OneCutterDimension

Boolean 类型属性，赋值/返回部件刀具类型 {*默认* = False}：
- (*False*) 刀具具有 XY 尺寸（例如铣削刀具、激光）
- (*True*) 刀具只有一个尺寸（例如刀片、刀具）。

CutterDiameter

赋值/返回部件切割直径的 ***Double*** 属性，缺少单个部件的有效赋值 {*单位*：Unit；*默认* = 0.0；*有效范围*：>= 0.0}。
相邻*部件*之间的放置采用对应设定值的最小间隔。

TecnoDistanceType

Boolean用于分配/返回部件工艺应用模式的属性 {*默认*= False}
- *False*：刀具应用在每个部件的轮廓周围
- *True*：刀具应用在每个部件的轮廓上重叠。

OverrunPolygon

赋值/返回部件切割区域重叠激活的 ***Boolean*** 属性 {*默认* = False}。
最大允许重叠对应切割直径减去*安全距离*（用 ***OverrunSecurity*** 属性赋值）。
下图将行为差异显示为属性值变化：

- 左侧，放置值 *False*
- 右侧，放置值 *True*.

OverrunSecurity

赋值/返回添加到放置的*安全距离*的 ***Double*** 类型属性，应用部件切割区域重叠 {*单位*：Unit；*默认* = 0.1 [mm]；*有效范围*：0.0/10.0 [mm]}。

OverrunSheet

Boolean 类型属性，赋值/返回部件切割外形激活以重叠板材边缘 {*默认* = True}。
- *True*：部件切割外形可以重叠板材外边缘。板材边缘与放置之间最小距离的计算等于切割直径的一半 (CutterDiameter * 0.5) 或相同直径，根据设置 MaximizeOverrunSheet。
- *False*：在板材外部边距内应用部件切割外形。板材边缘与放置之间最小距离对应切割直径 (CutterDiameter)。

如前所述，在确定放置与边缘的最小距离时，可以考虑多种因素，进一步增加间距：
- 使用板材的边距赋值（参见：MarginLeft()、...）
- 通过应用*弦误差*，修改外形
- 为部件赋值不同技术直径。

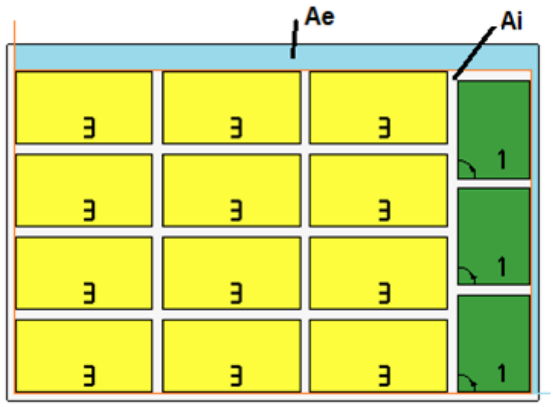
MaximizeOverrunSheet

Boolean 类型属性，赋值/返回最大化板材边缘切割外形重叠的激活。设置有效，如果 *OverrunSheet*=*true*。

SolutionMaxScrap

赋值/返回内部废料最大差值的 ***Double*** 类型属性 {*单位*：%；*默认* = 2.5；*有效范围*：0.5/50.0}。
将废料计算为包含板材放置的边界矩形的百分比，进行评估。这些信息与考虑的其他标准组合，确定不同求解中的最佳选择。
利用这些信息，和非矩形板材的情况一样高效，放置边界矩形还可包含板材外形外部、对放置无用的区域。

参考图：



- 外部矩形表示板材
- **Ai** 指放置相关面积：由部件限制坐标约束。**Ai** 面积与所有放置面积之间的差，对应嵌套*内部余切*。
- **Ae** 指放置外的面积，对应嵌套*外部余切*的面积。

SolutionExpected

赋值/返回最小预计板材值的 ***Double*** 属性 {*单位*：%；*默认* = 75.0；*有效范围*：25.0/95.0}。
将放置面积计算为包含板材所有放置的边界矩形的百分比（上图中的 **Ai**），进行评估。达到表示计算阶段完成条件的设定值，作为达到最大计算时间的替代。

ExtraFiller

赋值/返回额外放置应用模式的 ***Boolean*** 属性 {*默认* = False}。
- *False*：应用额外放置填充板材
- *True*：仅应用额外放置填充请求工件已经使用的长度或高度。根据放置的进料方向选择填充方向（参见 Direction 设置）：
  - 如果水平：填充应用于板材长度，不应用高度限制
  - 如果垂直：填充应用于板材高度，不应用长度限制。

ExtraFillerLastSheet

Boolean属性，用于分配/返回*ExtraFiller*设置的应用模式{*默认*= False}。
- *False*：将*ExtraFiller*应用于所有板材
- *True*：将*ExtraFiller*仅应用于最后一张板材（如果未重复）。

ModeExtraPart

Boolean 类型属性，赋值/返回部件和群集额外放置的解释标准 {*默认* = *False*}：
- *False*：设置对应可以放置的最大总数（包含请求的放置）
- *True*：设置对应将添加到请求放置的可放置最大数量。

UseOnlyExtraPart

Boolean 类型属性，赋值/返回授权使用仅赋值额外放置的部件 {default=False}：
- *false*：只能将额外放置应用添加到需要的放置
- *true*：额外放置应用不一定需要必要放置。

设置与应用自动 *false* 解释的手动集群无关。使用手动集群始终需要设置必要的布局。
属性有效值也影响部件重复情况的识别。

ModeMirrorPart

Boolean 类型属性，赋值/返回解释部件或群集镜像的应用标准 {*默认* = False}：
- *False*：请求并应用转换应用
- *True*：转换应用是无法在非镜像模式下放置的结果。

ModeOrderItem

Boolean 类型属性，赋值/返回解释部件、群集、板材优先级的应用标准 {*默认* = *False*}：
- *False*：按优先级降序排序（优先级较高的元素优先）
- *True*：按优先级升序排序（优先级较低的元素优先）。

UseOrderPart

Boolean 类型属性，赋值/返回*部件*和*群集*优先级应用激活 {*默认* = True}。
- *True*：激活 ***NestPart*** 和 ***NestCluster*** 结构中赋值的 *Order* 字段，用于赋值部件或群集
- *false*：忽略结构字段。

UseOrderSheet

Boolean 类型属性，赋值/返回*板材*优先级应用激活 {*默认* = True}。
- *True*：激活 ***NestSheet*** 结构中赋值的 *Order* 字段应用，用于赋值板材
- *False*：忽略结构字段。

UseBeforeScrap

Boolean 类型属性，赋值/返回首先使用标记为板材的废料的激活 {*默认* = False}。
- *True*：激活使用 ***NestSheet*** 结构中赋值的 *IsScrap* 字段，用于赋值板材
- *False*：忽略结构字段。

MatchType

Boolean 类型属性，赋值/返回部件和板材之间*材料*匹配应用激活 {*默认* = True}。
- *True*：激活结构（***NestPart***、***NestCluster***、***NestSheet***）中赋值的 *Fiber* 字段应用
- *False*：忽略结构字段。

MatchColor

Boolean 类型属性，赋值/返回部件和板材之间*颜色*匹配应用激活 {*默认* = True}。
- *True*：激活结构（***NestPart***、***NestCluster***、***NestSheet***）中赋值的 *Colour* 字段应用
- *False*：忽略结构字段。

MatchGrain

Boolean 类型属性，赋值/返回部件和板材之间*纹理*匹配应用激活 {*默认* = False}。
- *True*：激活结构中赋值的 *Grain* 字段应用（***NestPart***、***NestCluster***、***NestSheet***）
- *False*：忽略结构字段。

UseRangePart

Short类型属性，赋值/返回用于识别单个部件和/或群集分组的标准 {*默认* = 0}。
- *0*禁用该函数（停用对*Range*字段的解释）
- *1*：要求在公共区域内进行放置
- *2*：要求在由通道分隔的区域内进行放置。

DimRangePart

该双精度属性用于赋值/返回应用 UseRangePart 后相邻区域之间的分隔通道的大小 {*默认* = 0.0}。

RctMinimize

Boolean 类型属性，赋值/返回对应*部件*最小边界框的旋转搜索激活 {*默认* = True}。此激活可以用于赋值非矩形主几何的部件，可以旋转。

MarginOuter

用于将外部边缘分配给图纸的双精度浮点型属性 {*单位*：Unit；*默认* = 0.0；*有效范围*：>= 0.0}。
值整体赋值板材的四个外部边缘。

MarginLeft, MarginRight, MarginBottom, MarginTop

赋值/返回板材外部边缘的 ***Double*** 属性，分别在左、右、上、下侧 {*单位*：Unit；*默认* = 0.0；*有效范围*：>= 0.0}。
可以对矩形*板材*应用使用区分边缘。对于常规形状板材，应用对于四个值中最大值的唯一值。

MarginInner

赋值/返回放置之间额外应用距离的 ***Double*** 属性 {*单位*：Unit；*默认* = 0.0；*有效范围*：>= 0.0}。

Direction

赋值/返回放置进料方向的 ***Short*** 属性 {*默认* = 0；*有效范围*：0/1}：
- 0 = 水平方向
- 1 = 垂直方向。

对于 *OneNestingDimension=true*：对方向应用值 1（垂直方向）。

Corner

赋值/返回有效值中放置起点的 ***Short*** 属性 {*默认* = 0；*有效范围*：0/3}。
- 0 = 左下
- 1 = 左上
- 2 = 右下
- 3 = 右上。

Square 嵌套相关函数

目前暂无任何功能计划。

True Shape 嵌套相关函数

StepAngle

赋值/返回最小旋转角度的 ***Double*** 属性 {*默认* = 5.0；*单位* = 度和分；*有效范围*：1.0/90.0}。
设定值对应用于赋值部件的 ***NestPart*** 结构中赋值的 *Rotate* 字段值 2，群集赋值的 NestCluster 结构中。

PartInHole

赋值/返回部件废料区域内启用放置的 ***Boolean*** 属性 {*默认* = True}。
- *True*：激活用于赋值部件的 ***NestPart*** 结构中赋值的字段 *UseIsle=true* 应用，群集赋值的 NestCluster 结构中。

PartInHoleMulti

赋值/返回部件废料区域内激活*递归*放置的 ***Boolean*** 属性 {*默认* = False}。

PartInHoleBefore

赋值/返回部件废料区域内放*置优先*的 ***Boolean*** 属性 {*默认* = False}。

AutomaticCluster

Boolean 类型属性，赋值/返回应用*自动集群*启用 {*默认* = True}。
- *True*：激活用于赋值部件的 ***NestPart*** 结构中赋值的字段 *AutoPair=true* 应用，群集赋值的 NestCluster 结构中。

ClustersExpected

Double 类型属性，赋值/返回部件自动集群相对于单个放置的面积最小使用值 (%) {*默认* = 50.0；*有效范围*：50.0/95.0}。
此设置赋值为部件*自动集群*评估的最低效率，计算如下：

（单个部件面积 * 2 * 100）/（组长度 * 组高度）

ClustersAbsolute

Double 类型属性，含义与上一个 (*ClustersExpected*) 类似，用于自主应用自动集群 {*默认* = 50.0；*有效范围*：50.0/100.0}。

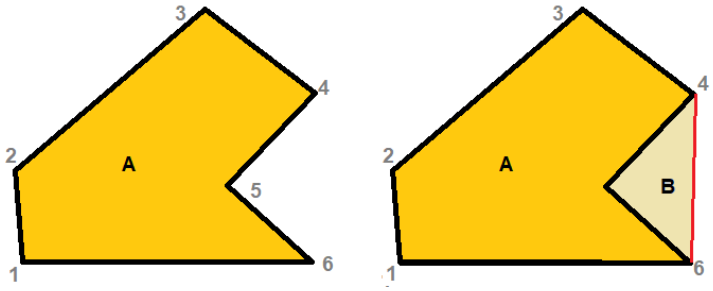
设定值不应低于之前的值。

AutomaticGrid

Boolean 类型属性，赋值/返回*网格放置*应用激活 {*默认* = True}。
- *True*：激活用于赋值部件的 ***NestPart*** 结构中赋值的字段 *AutoGrid=true* 应用 NestCluster 用于赋值群集的结构
- *False*：不包含应用已编程网格放置。

ExploreConcave

赋值/返回研究部件凹度启用的 ***Boolean*** 属性 {*默认* = True}。
图中，凹面部件情况：
- *True*：部件用作已赋值（图中左侧：顶点 1 至 6）
- *False*：使用部件，消除凹度。对于嵌套过程，如图右侧所示修改部件：顶点列表消除点 5，部件整体尺寸加入字母 B 指示的区域。



不使用凹形区域将加速计算过程，降低材料使用效率。

ConcaveDimension

Double 类型属性，赋值/返回用于减小部件凹度的长度 {*单位*：Unit；*默认* = 1.0；*有效范围*：>= 0.0}。
在 ***ExploreConcave=true*** 情况下使用该值：嵌套过程研究凹面区域，但简化。
对于上图所示凹度，凹度消除调整为验证：
- 部件 B 的面积小于半径等于设定值的圆面积；或
- 顶点 5 与连接顶点（4、6）的线段之间的距离小于设定值。

设置序列化函数

SaveSettings

函数将所有设置保存到文件（XML 格式）。

NestErrors SaveSettings (string pathName, bool bMode)

参数
- *pathName*: 文件路径
- *bMode*: *** 可用参数

返回值
NestError.ErrorNone，如果结果为正。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 函数在赋值部分（***IniSettings***、***IniSetPart***、***IniSetSheet***、***IniSetCluster***）使用
- (*NestError. ErrorIOProject*) 访问待写入文件时出错。

但是，一些设置不保存：FixComputeError、TimerProgres、DirectoryTemp、RetrySquare。

LoadSettings

函数从文件（XML 格式）读取所有设置。

NestErrors LoadSettings (string pathName, bool bMode)

参数
- *pathName*: 文件路径
- *bMode*: *** 可用参数

返回值
NestError.ErrorNone，如果结果为正。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 函数在赋值部分（***IniSettings***、***IniSetPart***、***IniSetSheet***、***IniSetCluster***）
- (*NestError.ErrorIOProject*) 访问待读取文件时出错。

所有设置之前赋值为默认值。
读取不包含一些保持不变的设置：FixComputeError、TimerProgres、DirectoryTemp、RetrySquare。

3.7 部件定义

IniSetPart

此函数打开部件的赋值部分。

bool IniSetPart (bool Clear)

参数
- *Clear*: True = 复位部件列表。

返回值
如果结果为正，则为 *True*，否则为 *False*。

注
此函数返回 *True* 允许全部或部分赋值部件。
以调用 ***EndSetPart()*** 函数完成赋值阶段。

此函数返回 *False* 对应以下错误情况之一：
- 发生严重系统状态，导致内存分配错误
- 嵌套优化正在运行。

查询 *LastError* 属性，评估特定错误情况。

EndSetPart

此函数关闭部件的赋值部分。

bool EndSetPart ()

返回值
如果结果为正，则为 *True*，否则为 *False*（调用不匹配 ***IniSetPart***，或者部件列表无效）。

AddPart

函数将部件添加到列表。

NestErrors AddPart (NestPart Item)

参数
- *Item*: 部件赋值结构。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 未在线段内使用函数（***IniSetPart -> EndSetPart***）
- (*NestError.ErrorPartsTomany*) 部件列表已经达到最大允许（500 个部件）或结构中请求的数量超过允许最大值 (999)
- (*NestError.ErrorUnexpected*) 没有赋值有效标识符（***Item.ID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 数字标识符 ***Item.ID*** 结果已赋值的部件。

对于无效数据情况，部件赋值可以使用相比结构修改的值。

RemovePart

函数消除部件或部件几何。

NestErrors RemovePart (int ItemID, bool OnlyGeometry)

参数
- *ItemID*: 部件标识符（> 0）
- *OnlyGeometry*: True = 仅消除额外几何赋值（如果已赋值）。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 未在线段内使用函数（***IniSetPart -> EndSetPart***）
- (*NestError.ErrorUnexpected*) 没有赋值有效标识符（***ItemID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 没有赋值数字标识符 ***Item.ID*** 结果为已赋值的部件。

WritePart

函数更改已经赋值的部件。

NestErrors WritePart (NestPart Item)

参数
- *Item*: 部件赋值结构。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 未在线段内使用函数（***IniSetPart -> EndSetPart***）

- (*NestError.ErrorUnexpected*) 没有赋值数字标识符 ***Item.ID*** 结果为已赋值的部件。

对于无效数据情况，部件赋值可以使用相比结构修改的值。

添加的几何分配保持不变。

CountPart

Integer 属性获得列表中的部件数量。
使用函数不需要在部分（***IniSetPart -> EndSetPart***）中工作。

ReadPart

函数搜索对应指定 ID 的部件。

NestErrors ReadPart (*int ItemID, ref *NestPart Item, ref NestSize ItemSize)

参数
- *ItemID*: 部件标识符（> 0）
- *Item*: 部件赋值结构
- *ItemSize*: 优化过程使用的边界矩形的赋值结构。

返回值
如果结果为正，*NestError.ErrorNone*。

注
使用函数不需要在线段内加工（***IniSetPart -> EndSetPart***）。

与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在线段内使用函数（***IniGeometry -> EndGeometry***）
- (*NestError.ErrorUnexpected*) 没有赋值有效标识符（***ItemID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 没有赋值数字标识符 ***Item.ID*** 结果为已赋值的部件。

ReadPartIndex

函数搜索对应指定索引的部件。

NestErrors ReadPartIndex (*int Index, ref *NestPart Item, ref NestSize ItemSize)

参数
- *Index*: （从零开始）部件列表的索引 (>= 0)
- *Item*: 部件赋值结构
- *ItemSize*: 优化过程使用的边界矩形的赋值结构。

返回值
如果结果为正，*NestError.ErrorNone*。

注
使用函数不需要在线段内加工（***IniSetPart -> EndSetPart***）。
此函数的主要用途是执行 ***LoadProject*** 函数后获取部件。

与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在线段内使用函数（***IniGeometry -> EndGeometry***）
- (*NestError.ErrorUnexpected*) 未赋值有效索引。

3.8 板材定义

IniSetSheet

此函数打开板材的赋值部分。

bool IniSetSheet (bool Clear)

参数
- *Clear*: True = 复位板材列表。

返回值
如果结果为正，则为 *True*，否则为 *False*。

注
此函数返回 *True* 允许全部或部分赋值板材。
以调用 ***EndSetSheet()*** 函数完成赋值阶段。

此函数返回 *False* 对应以下错误情况之一：
- 发生严重系统状态，导致内存分配错误
- 嵌套优化正在运行。

查询 *LastError* 属性，评估特定错误情况。

EndSetSheet

此函数关闭板材的赋值部分。

bool EndSetSheet ()

返回值
如果结果为正，则为 *True*，否则为 *False*（调用不匹配 ***IniSetSheet***，或者板材列表无效）。

AddSheet

函数将板材添加到列表。

NestErrors AddSheet (NestSheet Item)

参数
- *Item*: 板材赋值结构。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 未在线段内使用函数（***IniSetSheet -> EndSetSheet***）
- (*NestError.ErrorSheetsTomany*) 板材列表已经达到最大允许（100 个板材）或结构中请求的数量超过允许最大值 (999)
- (*NestError.ErrorUnexpected*) 板材未赋值有效标识符（***Item.ID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 有数字标识符 ***Item.ID*** 结果为赋值的板材。

对于无效数据情况，板材赋值可以使用相比结构修改的值。

RemoveSheet

函数消除板材或板材几何。

NestErrors RemoveSheet (int ItemID, bool OnlyGeometry)

参数
- *ItemID*: 板材标识符 (> 0)
- *OnlyGeometry*: True = 仅消除额外几何赋值（如果已赋值）。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 未在线段内使用函数（***IniSetSheet -> EndSetSheet***）
- (*NestError.ErrorUnexpected*) 没有赋值有效标识符（***ItemID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 没有按指定编号标识符 ***ItemID*** 结果的工作表。

WriteSheet

函数更改已经赋值的板材。

NestErrors WriteSheet (NestSheet Item)

参数
- *Item*： 板材赋值结构。

返回值
如果结果为正，则为 *True*，否则为 *False*。

注
此函数返回 *False* 对应以下错误情况之一：
- (*NestError.ErrorContext*) 未在线段内使用函数（***IniSetSheet -> EndSetSheet***）
- (*NestError.ErrorUnexpected*) 没有赋值数字标识符 ***Item.ID*** 结果为已赋值的板材。

对于无效数据情况，板材赋值可以使用相比结构修改的值。

添加的几何分配保持不变。

CountSheet

Integer 属性获得列表中的板材数量。
使用函数不需要在部分（***IniSetSheet -> EndSetSheet***）中工作。

ReadSheet

函数搜索对应指定 ID 的板材。

NestErrors ReadSheet (*int ItemID, ref *NestSheet Item, ref NestSize ItemSize)

参数
- *ItemID*： 板材标识符 (> 0)
- *Item*： 板材赋值结构
- *ItemSize*： 优化过程使用的边界矩形的赋值结构。

返回值
如果结果为正，则为 *True*，否则为 *False*。

注
使用函数不需要在线段内加工（***IniSetSheet -> EndSetSheet***）。

与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在线段内使用函数（***IniGeometry -> EndGeometry***）
- (*NestError.ErrorUnexpected*) 没有赋值有效标识符（***ItemID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 没有赋值数字标识符 ***Item.ID*** 结果为已赋值的板材。

ReadSheetIndex

函数搜索对应指定索引的板材。

NestErrors ReadSheetIndex (*int Index, ref *NestSheet Item, ref NestSize ItemSize)

参数
- *Index*： （从零开始）板材列表的索引 (>= 0)
- *Item*： 板材赋值结构

- *ItemSize*： 优化过程使用的边界矩形的赋值结构。

返回值
如果结果为正， *NestError.ErrorNone*。

注
使用函数不需要在线段内加工（***IniSetSheet -> EndSetSheet***）。
此函数的主要用途是执行 ***LoadProject*** 函数后获取板材。

与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在线段内使用函数（***IniGeometry -> EndGeometry***）
- (*NestError.ErrorUnexpected*) 未赋值有效索引。

3.9 多边形几何定义

我们将研究如何赋值部件和/或板材几何（*部件*）。
为了简化，下面假定我们处理部件列表赋值。

IniGeometry

此函数打开部件多边形几何的赋值部分。

bool IniGeometry (int ItemID, bool IsIsle, double X, double Y, double IsleBorder, double IsleDiameter)

参数
- *ItemID*： 部件标识符(> 0)
- *IsIsle*： True =该分配对应于零件岛/板材废料区
- *X, Y*： 几何起始坐标（值单位：Unit）。
- *IsleBorder*：为几何图形分配的边距（仅在以下情况有效：严格为正（即：>0）且*Isle*=true)
- *IsleDiameter*： 剪切直径（仅在以下情况有效：严格为正（即：>0）且*IsIsle=true*)

返回值
如果结果为正，则为*True*，否则为*False*。

注
此函数返回*True*允许继续赋值。
以调用***EndGeometry()***函数完成赋值阶段。

此函数返回*False*对应以下错误情况之一：
- （*NestError.ErrorContext*）未在线段内使用函数（***IniSetPart -> EndSetPart***或***IniSetSheet -> EndSetSheet***）
- （*NestError.ErrorUnexpected*）没有赋值有效标识符（***ItemID***值必须严格为正）
- （*NestError.ErrorUnexpected*）没有赋值数字标识符***Item.ID***结果为已赋值的部件
- （*NestError.ErrorUnexpected*）如果*IsIsle=false*：没有主外形必须为已经赋值
- （*NestError.ErrorInGeometry*)如果 *IsIsle=true*且孤岛列表 已达到允许的最大值（100 个元素）

查询*LastError*属性，评估特定错误情况。
如果*IsIsle=true*并且没有赋值主外形，将用部件赋值的矩形形状和尺寸（长度 * 高度）自动赋值。

参数（*IsleBorder, IsleDiameter*）仅在内部区域的情况下有效：

- 如果是部件孤岛：（*IsleDiameter >0*）为孤岛分配特定的剪切直径；*IsleBorder*无效
- 如果是板材废料：（*IsleBorder* ）在区域外部分配安全边距；（*IsleDiameter >0*）为区域分配剪切直径。

关于工艺参数的分配和使用情况，参见段落： 板材内约束区域的间距。

EndGeometry

此函数关闭部件多边形几何的赋值部分。

bool EndGeometry()

返回值
如果结果为正，则为 *True*，否则为 *False*。

注
此函数返回 *False* 对应以下错误情况之一：
- 调用不对应 ***IniGeometry***
- 赋值给部件的几何列表无效。

函数为每个几何运行赋值给部件的所有几何的总体有效性控制，不仅仅是当前段赋值的单个几何；对于每个几何：
- 必须赋值至少一个弧形元素或两个直线元素。

AddToGeometry_Line

函数将直线元素添加到多边形几何。

NestErrors AddToGeometry_Line (double Xend, double Yend)

参数
- *Xend*: 线段的最终 X 坐标（单位：Unit）
- *Yend*: 线段的最终 Y 坐标（单位：Unit）。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何函数返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 未在线段内使用函数 (***IniGeometry -> EndGeometry***)
- (*NestError.ErrorInGeometry*) 商品列表已达到允许的最大数量（10000 件）
- (*NestError.ErrorInGeometry*) 直线元素具有空长度 (<= ***MinResolution***)。

直线段起点对应上一个线段的终点。

AddToGeometry_Arc

函数将弧形元素（圆弧）添加到多边形几何。

NestErrors AddToGeometry_Arc (double Xend, double Yend, double Xcentre, double Ycentre, bool IsCCW)

参数
- *Xend, Yend*: 线段的最终 (X, Y) 坐标（单位：Unit）
- *Xcentre, Ycentre*: 线段的中心 (X, Y) 坐标（单位：Unit）
- *IsCCW*: True = 逆时针旋转。

返回值
如果结果为正，*NestError.ErrorNone*。

注
此函数返回 *False* 对应以下错误情况之一：
- (*NestError.ErrorContext*) 未在线段内使用函数 (***IniGeometry -> EndGeometry***)
- (*NestError.ErrorInGeometry*) 商品列表已达到允许的最大数量（10000 件）
- (*NestError.ErrorInGeometry*) 弧形元素无效（半径 <= ***MinResolution*** 或 |初始半径 - 最终半径| > ***MinResolution***）。

弧形段起点对应上一个线段的终点。

AddToGeometry_Circle

函数将圆元素添加到多边形几何。

NestErrors AddToGeometry_Circle (double Xcentre, double Ycentre, bool IsCCW)

参数
- *Xcentre, Ycentre*: 线段的中心 (X, Y) 坐标（单位：Unit）
- *IsCCW*: True = 逆时针旋转。

返回值
如果结果为正，*NestError.ErrorNone*。

注

参见 ***AddToGeometry_Arc*** 函数。

AddToGeometry

函数将常规元素添加到多边形几何。

NestErrors AddToGeometry (NestGeometry Item)

参数
- *Item*: 几何元素赋值结构。

返回值
如果结果为正，*NestError.ErrorNone*。

注
参见 ***AddToGeometry_**** 函数。

读取几何

ReadGeometry

函数递归调用读取为部件赋值结果的几何。

NestErrors ReadGeometry (int List, int ItemID, int IndexItem, int IndexElement, ref NestGeometry Item)

参数
- *List*: 选择列表：0 = 部件；1 = 板材
- *ItemID*: 元素标识符（部件或板材）(> 0)
- *IndexItem*: （从零开始）赋值几何外形列表中的索引
- *IndexElement*: （从零开始）几何外形中的几何元素索引
- *Item*: 几何元素赋值结构。

返回值
如果结果为正，*NestError.ErrorNone*。

注
此函数的主要用途是执行 ***LoadProject*** 函数后获取几何。

对应 (*List*, *ItemID*) 赋值，第一个调用必须使用 (*IndexItem=0*, *IndexElement=0*)。与 *NestError.ErrorNone* 不同的第一次调用任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在线段内使用函数 (***IniGeometry -> EndGeometry***)
- (*NestError.ErrorUnexpected*) 无效元素标识符或没有赋值几何的元素（主或辅助几何）。

ReadGeometryInfo

该函数调用用于读取分配给部件次要几何图形的辅助信息。

NestErrors ReadGeometryInfo (int List, int ItemID, int IndexItem, ref double IsleBorder, ref double IsleDiameter)

参数
- *List*: 选择列表：0 = 部件；1 = 板材
- *ItemID*: 项目标识符（部件或板材）（> 0）
- *IndexItem*: 分配的几何轮廓列表中的索引（从零开始）
- *IsleBorder*: 为几何图形分配的边距（参见：IniGeometry）
- *IsleDiameter*: 为 几何图形分配的切割直径（参见：IniGeometry）

返回值
*NestError.ErrorNone*如果结果为正。

注释
该函数的主要用途是在执行*LoadProject*函数后获取几何图形，且仅限于对应于部件孤岛或板材废料区域的几何图形情况。

若返回非*NestError.ErrorNone*则对应以下错误情况之一：
- （*NestError.ErrorBusy*）嵌套优化正在运行
- （*NestError.ErrorContext*）该函数在某个代码段内使用（*IniGeometry -> EndGeometry*）

- （*NestError.ErrorUnexpected*）项目标识符无效或属于主要分配项目。

使用外部几何

ShapeExtension

返回支持扩展列表用于解释外部文件几何的 ***String*** 属性。可能的情况如下：

- *=""*: 无有效识别。调用 *ReadShape* 函数失败
- *=".DXF"*: 识别 DXF 文件读取

ReadShape

函数从 DXF 或 DWG 文件解释几何。

NestErrors ReadShape (string pathOpen)

参数
- *pathOpen*：待读取的文件完整路径（例如："C:\PATTERN\A.DXF"）。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ERR_NONE* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在线段内使用函数（***IniGeometry -> EndGeometry***）
- (*NestError.ErrorLicense*) 检验高级函数键存在和状态时出现负结果
- (*NestError.ErrorIOfile*) 访问或解释 *pathOpen* 文件时出错
- (*NestError.ErrorIOfile*) 管理临时文件时出错。

函数不改变部件几何的任何赋值：运行格式解释，加载读取的几何，以供以后读取。

现在我们将查看图纸文件的常规解释规则：
- 解释 2D 图纸
- 仅解释外形赋值几何项（折线、圆、椭圆、样条）
- 仅闭合外形识别为有效（应用最大距离等于切割直径，评估闭合外形条件：参见 ***CutterDiameter***）
- 仅保留第一个内部层赋值的 1 个主外形（具有最大面积）及其最终孤岛。

ReadGeoInShape

函数递归调用读取调用 *ReadShape* 函数赋值的几何。

bool ReadGeoInShape (int IndexItem, int IndexElement, ref NestGeometry Item)

参数
- *IndexItem*: （从零开始）赋值几何外形列表中的索引
- *IndexElement*: （从零开始）几何外形中的几何元素索引
- *Item*: 几何元素赋值结构。

返回值
如果结果为正，则为 *True*，否则为 *False*。

注
第一个调用必须使用（*IndexItem=0*，*IndexElement=0*）。与 *False* 不同的第一次调用任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorUnexpected*) 没有赋值几何（无主或辅助几何）。

查询 *LastError* 属性以评估特定错误情况。

3.10 手动群集定义

IniSetCluster

此函数打开群集赋值部分。

bool IniSetCluster (bool Clear)

参数
- *Clear*: *True* = 复位群集列表。

返回值
True，如果结果为正；否则 *False*。

注
此函数返回 *True* 允许继续全部或部分赋值集群
调用 ***EndSetCluster()***，完成赋值阶段。

此函数返回 *False* 对应以下错误情况之一：
- 发生严重系统状态，导致内存分配错误
- 嵌套优化正在运行。

查询 *LastError* 属性以评估特定错误情况。

EndSetCluster

此函数关闭群集赋值部分。

bool EndSetCluster ()

返回值
True，如果结果为正；否则为 *False*（调用不匹配 ***IniSetCluster***，或者群集列表无效）。

AddCluster

函数将群集添加到列表。

NestErrors AddCluster (NestCluster Item)

参数
- *Item*: 群集赋值结构。

返回值
NestError.ErrorNone，如果结果为正。

注
如果函数返回值为*正*，则可以继续赋值群集的几何构成。

与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 没有在部分 (***IniSetCluster -> EndSetCluster***) 中使用函数
- (*NestError.ErrorClustersTomany*) 群集列表已经达到允许最大值（500 项）或结构中请求的数量超过允许最大值 (999)
- (*NestError.ErrorUnexpected*) 没有为群集赋值有效标识符（***Item.ID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 数字标识符 ***Item.ID*** 结果为赋值的群集。

对于无效数据情况，群集赋值可以使用相比结构修改的值。
新群集没有赋值几何构成。

AddToCluster

函数添加一个元素以定义群集的几何构成。

NestErrors AddToCluster (int ItemID, ItemCluster Item)

参数
- *ItemID*: 群集标识符 (> 0)
- *Item*: 部件赋值结构。

返回值
NestError.ErrorNone，如果结果为正。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 没有在部分 (***IniSetCluster -> EndSetCluster***) 中使用函数
- (*NestError.ErrorUnexpected*) 没有赋值有效标识符（***ItemID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 没有数字标识符 ***ItemID*** 结果为赋值的群集
- (*NestError.ErrorInCluster*) 没有数字标识符 ***Item.NameID*** 结果为赋值的部件（请记住，字段可以赋值数字标识符或文本别名）
- (*NestError.ErrorInCluster*) 赋值群集组成的元素列表已经达到允许最大值 (100)。

RemoveCluster

函数删除定义其组成的群集或元素。

NestErrors RemoveCluster (int ItemID, bool OnlyGeometry)

参数
- *ItemID*: 群集标识符 (> 0)
- *OnlyGeometry*: *True* = 仅删除添加的几何赋值（如果赋值）。

返回值
NestError.ErrorNone，如果结果为正。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 没有在部分 (***IniSetCluster -> EndSetCluster***) 中使用函数
- (*NestError.ErrorUnexpected*) 没有赋值有效标识符（***ItemID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 没有数字标识符***ItemID*** 结果为赋值的群集。

WriteCluster

函数更改已经赋值的群集。

NestErrors WriteCluster (NestCluster Item)

参数
- *Item*: 群集赋值结构。

返回值
NestError.ErrorNone，如果结果为正。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorContext*) 没有在部分 (***IniSetCluster -> EndSetCluster***) 中使用函数
- (*NestError.ErrorUnexpected*) 没有数字标识符 ***Item.ID*** 结果为赋值的群集。

对于无效数据情况，群集赋值可以使用相比结构修改的值。
添加的几何赋值不更改。

CountCluster

Integer 类型属性获得列表中的群集数量。
使用函数不需要在部分 (***IniSetCluster -> EndSetCluster***) 中工作。

ReadCluster

函数搜索对应指定 ID 的群集。

NestErrors ReadCluster (int ItemID, ref NestCluster Item)

参数
- *ItemID*: 群集标识符 (> 0)
- *Item*: 群集赋值结构。

返回值
NestError.ErrorNone，如果结果为正。

注
使用函数不需要在部分（***IniSetCluster -> EndSetCluster***）中工作。

与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorUnexpected*) 没有赋值有效标识符（***ItemID*** 值必须严格为正）
- (*NestError.ErrorUnexpected*) 没有数字标识符 ***ItemID*** 结果为赋值的群集。

ReadClusterIndex

函数搜索对应指定索引的群集。

NestErrors ReadClusterIndex (int Index, ref NestCluster Item)

参数
- *Index*: （从零开始）群集列表的索引 (>= 0)
- *Item*: 群集赋值结构。

返回值
NestError.ErrorNone，如果结果为正。

注
使用函数不需要在部分（***IniSetCluster -> EndSetCluster***）中工作。
此函数的主要用途是执行 ***LoadProject*** 函数后采集群集。

与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorUnexpected*) 没有赋值有效索引。

ReadInCluster

递归调用函数读取赋值给群集的组成元素。

NestErrors ReadInCluster (int ItemID, int IndexItem, ref ItemCluster Item)

参数
- *ItemID*: 群集标识符 (> 0)
- *IndexItem*: 群集中赋值的元素列表上的（从零开始）索引
- *Item*: 元素赋值结构。

返回值
NestError.ErrorNone，如果结果为正。

注
此函数的主要用途是执行 ***LoadProject*** 函数后采集群集。

对应 (*ItemID*) 的赋值，第一个调用必须使用 (*IndexItem=0*)。与 *NestError.ErrorNone* 不同的第一次调用的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在线段内使用函数（***IniGeometry -> EndGeometry***）
- (*NestError.ErrorUnexpected*) 分配了无效的集群标识符或未分配组合元素。

3.11 初步安置的定义

IniSetPlaced

该函数打开初步放置分配节。

bool IniSetPlaced (bool Clear)

参数
- *Clear*: true = 将初步放置列表清零。

返回值
如果结果为正，则返回 True；*否则返回*False。

注释
函数返回*True*允许您继续进行放置的全量或部分分配。
分配阶段通过调用*EndSetPlaced ()*函数关闭。

函数返回*False*对应于以下错误情况之一：
- 发生严重的系统状况，导致内存分配错误
- 嵌套优化正在运行

查询LastError属性以评估特定错误情况。

EndSetPlaced

该函数关闭初步放置分配节。

bool EndSetPlaced ()

返回值
*如果结果为正*则返回 True*否则返回 False*（调用与*IniSetPlaced*不匹配，或放置列表无效）。

AddPlaced

该函数向列表中添加一个部件。

NestErrors AddPlaced (int ID_sheet, NestPlaced Item)

参数
- *ID_Sheet*: 板材标识符（> 0）
- *Item*: 放置分配结构

返回值
*NestError.ErrorNone*如果结果为正

注释
返回非*NestError.ErrorNone*的值对应于错误情况：
- （*NestError.ErrorContext*）该函数未在某个代码段内使用（*IniSetPlaced -> EndSetPlaced*）
- （*NestError.ErrorPlacedTomany*）列表已达到允许的最大值（999 个项目）
- （*NestError.ErrorUnexpected*）为板材或部件分配了无效的标识符（*ID_sheet和Item.ID*必须严格为正）。

RemovePlaced

该函数删除一个或多个初步放置。

NestErrors RemovePlaced (int Index,*int ID_sheet)*

参数
- *Index*: 列表上的索引（从零开始计）（>= 0）
- *ID_Sheet*: 板材标识符（> 0）

返回值
*NestError.ErrorNone*如果结果为正

注释
返回非*NestError.ErrorNone*的值对应于错误情况：
- （*NestError.ErrorContext*）该函数未在某个代码段内使用（*IniSetPlaced -> EndSetPlaced*）
- （*NestError.ErrorUnexpected*）分配了无效参数

可以区分两种操作情况：
- （*Index>=0*）删除*(Index)*中指定的索引所对应的放置
- （*Index<0, ID_Sheet>0*）删除分配给*(ID_Sheet)*中指定的板材的所有放置。

CountPlaced

该*integer*属性获取列表中的放置数量。
使用该属性不需要在一个代码段（*IniSetPlaced -> EndSetPlaced*）内进行操作。

ReadPlacedIndex

该函数搜索位于指定索引处的放置。

*NestErrors ReadPlacedIndex (*int Index, *ref int ID_sheet, ref NestPlaced Item)*

参数
- *Index*: 列表上的索引（从零开始计）（>= 0）
- *ID_Sheet*: 板材标识符（> 0）
- *Item*: 放置分配结构

返回值
*NestError.ErrorNone*如果结果为正

注释
使用该函数不需要在一个代码段（*IniSetPlaced -> EndSetPlaced*）内工作。

返回非*NestError.ErrorNone*的值对应于错误情况：
- （*NestError.ErrorBusy*）嵌套优化正在运行
- （*NestError.ErrorContext*）该函数在某个代码段内使用（*IniGeometry -> EndGeometry*）
- （*NestError.ErrorUnexpected*）分配了无效的索引。

3.12 嵌套求解

Compute

函数开始优化过程。

NestErrors Compute (int OptiSelect, bool OnlyTest, bool StepByStep)

参数
- *OptiSelect*: 选择优化类型（-1 =自动；0 =*Square*；1 =*True Shape*）
- *OnlyTest*: True =仅运行优化前的赋值
- *StepByStep*: 选择优化加工类型。

返回值
如果结果为正，*NestError.ERR_NONE*。

注
与*NestError.ErrorNone*不同的任何返回值对应以下错误情况之一：
- （*NestError.ErrorBusy*）嵌套优化正在运行
- （*NestError.ErrorLicense*）检验基本函数键存在和状态时出现负结果
- （*NestError.ErrorLicenseHight*）请求*True Shape*优化，键状态不匹配高级函数
- （*NestErrors.ErrorPartsEmpty*）空部件列表或无启用项
- （*NestErrors.ErrorSheetsEmpty*）空板材列表或无启用项
- （*NestErrors.ErrorInCluster*）手动集群分配时出现错误，检查特定匹配（分配的集群粒度与部分粒度不同）
- （*NestErrors.ErrorClusterMatch*）检验特定匹配时群集分配出错（群集具有的赋值纹理与部件纹理不同）
- （*NestErrors.ErrorPlacedMatch*)分配初步放置时出错，在检查特定对应关系时出错
- （*NestError.ErrorReset*）调用应用取消该过程。

开始时，函数
- 闭合最终打开的赋值段（设置、部件、板材、群集）
- 运行初步检查（如上所述）
- 对部件、群集和板材列表运行检查
- 根据键检验和板材部件列表，评估所需优化类型。

如果执行列出的测试导致错误，函数仍返回，不执行优化。
否则根据***OnlyTest***继续：
- ***True***：不改变当前优化状态，仅返回结果（正或错误）
- ***False***：取消执行的优化状态，开始新优化。

StepByStep的值在优化过程的两种可能行为之间进行选择：
- *false*: 优化根据分配的最大时间停止
- *true*: 优化在计算出第一个方案时停止。

库运行*Square*优化：
- 如果所有*部件*和*板材*赋值为简单矩形，没有赋值群集，*OptiSelect=-1 [注释 (1)]*
- 如果*OptiSelect=0*
- 如果高级函数键存在和状态检查失败。

运行*Square*优化：
- 为外形本身的边界矩形考虑赋值几何外形的*部件*，忽略赋值为孤岛的外形
- 为外形本身的边界矩形考虑赋值几何外形的*板材*，忽略赋值为约束区域的外形：在此情况下，优化求解可以在赋值 板材外和/或约束区域内放置
- 为赋值的几何组成的整体边框考虑*群集*
- 进行增量优化的可能性由RetrySquare设置以及嵌套的复杂度决定。

*[注释 (1)]*为了执行 Square 优化，评估部件和板材的几何形状比上述陈述要稍复杂一些：
- 一个部件可以分配孤岛，但可以排除将其用于内部放置的可能性（系统会预先对可放置部件的面积与可用孤岛的面积进行比较）
- 部件的几何轮廓为矩形或类似于矩形：比例（轮廓面积/边界框面积）>= 85%，点沿轮廓的演进以及点相对于边界框的偏差在边界框最小尺寸的六分之一以内
- 板材的几何轮廓为矩形或类似于矩形：所有段均为直线，比例（轮廓面积/边界框面积）>= 95%，通过边界框的 4 个顶点，点沿轮廓的演进及其相对于边界框的偏差在设定的分辨率以内

IsComputed

返回执行优化信息的 ***Boolean*** 属性。
用 ***OnlyTest=false*** 和执行优化调用 ***Compute*** 函数后，设置值有效。

ModeCompute

返回应用于上次执行优化的信息的 ***Integer*** 属性：0 = *Square* 类型；1 = *True Shape* 类型。
用 ***OnlyTest=false*** 和执行优化调用 ***Compute*** 函数后，属性值有效。

TimeCompute

Double 类型属性，返回上次执行 ***Compute*** 函数或 ***RetryCompute*** 的相关计算时间 {*单位* = 秒}。
执行优化后，此属性值有效。

TimeOut

返回计算过程结束或达到超时信息的 ***Boolean*** 属性。
执行优化后，此属性值有效。

CanRetry

返回可以从上次找到求解请求新优化尝试的信息的 ***Boolean*** 属性。
能够计算更多求解的条件包括：
- 模式优化必须执行结果并完成
- 组件不得赋值任何变化（常规设置，部件和/或板材和/或群集列表）
- 未计算最大管理求解数量（赋值 = 20）
- 对于新计算尝试，优化过程具有运行限制。

RetryCompute

函数从找到的上一个求解开始优化过程。

NestErrors RetryCompute (bool StepByStep, bool RetryBest)

参数
- *StepByStep*：选择优化加工类型
- *RetryBest*：True = 仅当优于上一个时，赋值新求解。

返回值
如果结果为正，*NestError.ErrorNone*。

注

与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- 参见 ***CanRetry*** 属性情况
- (*NestError.ErrorReset*) 调用应用取消该过程。

RetryBest 值有两个可能选择：
- *False*：如果正确执行优化，计算求解变为当前
- *True*：仅当评估*优于*上一个时，计算求解变为当前。

StepByStep 值有两个可能优化过程选择：
- *False*：优化按照最大赋值时间停止
- *True*：优化在第一个计算求解停止。

ClearSolution

函数重置任何计算的求解。

NestErrors ClearSolution ()

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行。

3.13 嵌套结果

这里列出的所有属性和函数在以下错误情况下失败：

- 嵌套优化正在运行
- 没有计算有效优化。

Solution

赋值/返回当前求解数量的 ***Integer*** 属性。
返回值为：
- 0: 没有计算任何求解
- 1: 当前求解对应第一个（调用 ***Compute()*** 函数）
- >1: 当前求解对应调用 ***RetryCompute()*** 函数计算的求解。

赋值时：
- 属性允许将*当前求解*更改为已经计算的求解
- 如果值对于求解定义无效，则不赋值。

OfSolution

返回计算求解总数的 ***Integer*** 属性。
返回值为：
- 0: 没有计算任何求解
- 1: 仅计算调用 ***Compute()*** 函数的求解
- >1: 计算更多求解（调用 ***RetryCompute()*** 函数后调用 ***Compute()***）。

Fitness

返回*当前求解*效率信息的 ***Double*** 属性。
执行优化后，此属性值有效。
按照用于放置的面积与使用的面板总面积的 % 比率计算总效率：板材中的放置面积越大，计算的效率越高。

对于求解的最后一个板材，效率计算可以很方便地将可用于放置的面积限制为为已放置产生的总矩形：随着可用面积减小，为板材计算的效率增加。但是，存在以下限制：

- 执行板材时不带重复
- 板材形状为矩形，没有约束区域。

ReadNumResult

函数读取*当前求解*的板材数量（具有放置的板材）。

int ReadNumResult (int ID_Sheet)

参数
- *ID_Sheet*： 板材标识符 (> 0)。

注
函数返回当前求解或板材类型的数量：

ID_Sheet	
<=0	获得为求解计算的板材总数
>0	获得为求解计算的 (*ID_Sheet*) 类型板材数量

返回的值考虑了相同的重复表。

ReadNumPartInResult

函数读取当前求解板材中或整个求解的放置数量，可以仅匹配一种部件。

int ReadNumPartInResult (int Index, int ID_Part)

参数
- *Index*： 求解板材索引，从零开始
- *ID_Part*： 部件标识符 (> 0)。

注
函数读取对应赋值参数的放置数量：

Index	*ID_Part*	
-1	0	获得求解放置总数
-1	>0	获得对应所有求解板材中部件 (*ID_Part*) 的放置数量
>=0	0	获得具有 (*Index*) 索引的求解板材中的放置总数
>=0	>0	获得 (*Index*) 索引板材中的 (*ID_Part*) 部件对应的放置数量

如果负 *Index* (-1)，返回值考虑相同重复板材。
如果 *Index* >= 0，返回值计数单个板材放置，不考虑板材本身最终重复。

ReadNumClusterInResult

函数读取当前求解板材中或整个求解的群集放置数量，可以仅匹配一种群集。

int ReadNumClusterInResult (int Index, int ID_Item)

参数
- *Index*： 从零开始的求解板材索引
- *ID_Item*： 群集标识符 (> 0)。

注
函数读取匹配赋值参数的群集放置数量：

Index	*ID_Item*	
-1	0	获得求解群集放置总数
-1	>0	获得所有求解板材中匹配群集 (*ID_Item*) 的放置数量

>=0	0	获得求解的 (*Index*) 索引板材中的群集放置总数
>=0	>0	获得具有 (*Index*) 索引的板材中匹配群集 (*ID_Item*) 的放置数量

对于负 *Index* (-1) 操作，返回值考虑相等重复板材。
对于 *Index* >= 0 的操作，返回值计数单个板材放置，并不考虑板材本身任何重复。

ReadResult

函数返回对应指定索引的*当前求解*板材相关信息。

bool ReadResult (int Index, ref int ID_Sheet, ref int Item_Sheet, ref double AreaPercent, ref int NumPlaces, ref int Repetition)

参数
- *Index*: （从零开始）求解板材索引
- *ID_Sheet*: 板材数字标识符（*NestSheet* 结构的 ID 字段）
- *Item_Sheet*: 板材编号标识符类型的板材发生
- *AreaPercent*: 占据面积/总可用面积
- *NumPlaces*: 板材上放置的部件数量
- *Repetition*: 板材重复数量。

返回值
如果板材标识为有效，*True*。

注释
AreaPercent 参数与 Fitness 计算：板材中的放置越多，计算的效率越高。

ReadPartInResult

函数返回*当前求解*板材上放置的相关信息。

bool ReadPartInResult (int Index, int IndexPart, ref NestBox Item)

参数
- *Index*: （从零开始）求解板材索引
- *IndexPart*: （从零开始）求解板材上的放置索引
- *Item*: 放置赋值结构。

返回值
如果板材上的放置标识为有效，*True*。

注
正如前面提到的，*Item*结构返回放置相关信息：

- *int ID*: 匹配放置的*部件*数字标识符
- *int Item*: 放置的进度编号（>0）
- *double (X, Y)*: 部件原始边界矩形LB（左下角）顶点的放置(X, Y)坐标
- *short Mirror*: 放置需要的镜像
  - 0 =无；1 = x；2 = y；3 = x+y
  - 轴应用转换的(X, Y)位置
  - 先应用转换，然后再进行任何旋转
- *double A*: 对应放置的旋转角度（单位：度和分）
  - 符号赋值旋转：正 = 逆时针；负 = 顺时针
  - (X, Y)是旋转中心
  - A赋值相对角度：部件围绕(X, Y)旋转A
- *bool InIsle*: True =放置在孤岛中（仅当嵌套*True Shape*时）
  - 两个*int*字段*(InIsleMain, InIsleOut)*允许您完整地重建嵌套在孤岛中的放置进度。例如，客户可以使用不同的放置顺序，以便首先切割放置在孤岛内的部件。
- *string InCluster*：指示放置是否来自群集应用
  - *""*：放置为单个
  - 否则：来自群集分解。Format："#;ID"
    - #：逐渐应用群集(> 0)。具有相同值的放置来自相同群集放置
    - ID：群集的数字标识符（*NestCluster*结构中的*ID*字段）。
- *bool IsOver*: true =该放置对应于一个额外项目

- *bool IsInput*： true =对应于一个初步放置。
  - 然而，手动集群的初步放置被拆分为单个部件的放置。

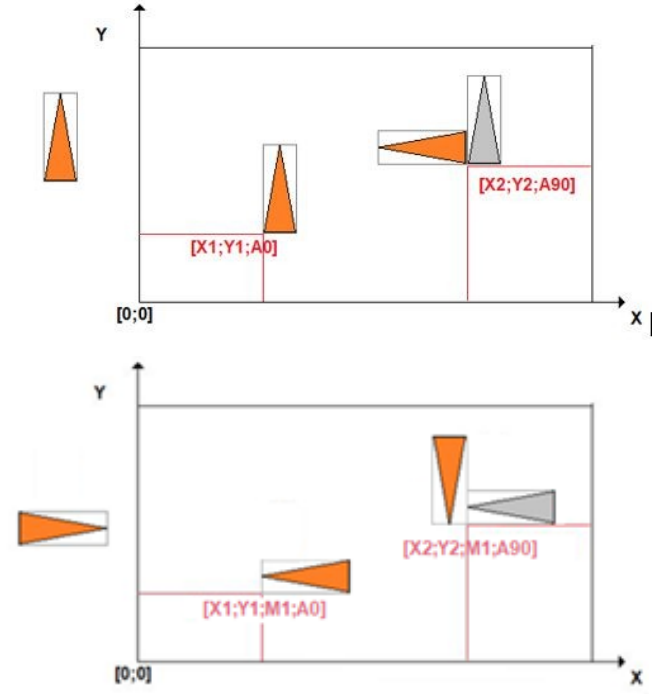
许多因素决定单个放置的镜像和旋转转换：

- 直接赋值源或原始群集：元素结构中的字段（*Mirror*、*Rotate*）
- *ModeMirrorPart*设置
- 原始群集中的几何组成。

这些图片展示了放置规则几何体零件的简单示例：

- 第一张图片应用了平移和旋转变换
- 第二幅图也应用了镜像变换

在坐标轴系统的左侧，显示了零件的原始几何形状



ReadGeoInResult

函数递归调用读取为当前放置赋值结果的几何。

bool ReadGeoInResult (int IndexItem, int IndexElement, ref NestGeometry Item)

参数

- *IndexItem*： （从零开始）赋值几何外形列表中的索引
- *IndexElement*： （从零开始）几何外形中的几何元素索引
- *Item*： 几何元素赋值结构。

返回值

如果板材上的放置标识为有效，*True*。

注

必须在调用 ***ReadPartInResult*** 函数后立刻调用该函数，必须使用第一个调用（*IndexItem=0*，*IndexElement=0*）。

此第一个调用返回 *False* 对应以下错误情况之一：

- 嵌套优化正在运行或者无效
- 当前放置结果无效。

函数运行重新构造当前放置关联*部件*，应用所有必要转换：平移、镜像、旋转。

如果部件形状赋值为纯矩形（部件 *L*，*H* 字段的尺寸），函数递归调用将返回对应矩形延展的四个线性元素。

SaveSolution

函数将*当前求解*保存到文件（XML 格式）。

NestErrors SaveSolution (string pathName)

参数
- *pathName*： 文件路径。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorNoneSolution*) 无当前求解结果
- (*NestError.ErrorIOfile*) 访问或解释待写入文件时出错。

外部应用程序可以将保存的文件解释为函数查询的替代。
对于板材放置，文件保存每个放置的基本信息作为来自 ***NestBox*** 结构。

SaveSolutionDXF

函数将*当前求解*保存到一个或多个文件（DXF 格式）。

NestErrors SaveSolutionDXF (string pathName, bool autoClose, string layerSheet)

参数
- *pathName*： 文件路径
- *autoClose*： True = 生成对应始终闭合部件的轮廓
- *layerSheet*： 板材分配轮廓的分配层名称。

返回值
NestError.ErrorNone 如果结果为正。

注
返回 *NestError.ErrorNone* 以外的值对应已经为 *SaveSolution* 函数提到的错误情况之一。

如果 *pathname* 分配 ".DXF" 以外的扩展名，将把相同内容添加到完整文件名。

函数为*当前求解*的每个板材（带位置的板材）保存一个文件。
分配了重复的板材仅保存一次。
第一个板材另存为 *pathName*。剩余板材通过添加 "_n" 修改原始名称，其中 n = 渐进值（1、2、□）。

如果 *layerSheet* 分配有效（示例："sheet"）：
- 文件还分配对应板材定义的折线，层字段对应指示字符串
- 具有相同层的其他折线可以分配板材内的孤岛
- 对应板材的折线始终显示闭合。

如果 *autoClose=true*：对应部件的轮廓显示为闭合；否则：为编程。

3.14 项目序列化函数

这些函数管理与文件之间序列化嵌套项目。

SaveProject

函数将*部件*和*板材*列表赋值保存到文件（XML 格式）。

NestErrors SaveProject (string pathName, bool bMode)

参数
- *pathName*： 文件路径
- *bMode*： *True* 要求保存常规函数设置。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在赋值部分使用函数 (***IniSettings -> EndSettings***、***IniSetPart -> EndSetPart***、***IniSetSheet -> EndSetSheet***、***IniSetCluster -> EndSetCluster***)
- (*NestError.ErrorIOProject*) 访问或解释待写入文件时出错。

使用 *bMode=true* 也保存常规函数设置（对应 ***IniSettings -> EndSettings*** 部分）。包含设置的保存用途对于测试情况很明显。
一些设置不保存 (FixComputeError、TimerProgres、DirectoryTemp、RetrySquare) 除了初步排名之外。

LoadProject

函数从文件（XML 格式）读取*部件*和*板材*列表赋值。

NestErrors LoadProject (string pathName, bool bMode)

参数
- *pathName*: 文件路径
- *bMode*: *True* 启用读取常规函数设置。

返回值
如果结果为正，*NestError.ErrorNone*。

注
与 *NestError.ErrorNone* 不同的任何返回值对应以下错误情况之一：
- (*NestError.ErrorBusy*) 嵌套优化正在运行
- (*NestError.ErrorContext*) 在赋值部分使用函数 (***IniSettings -> EndSettings***、***IniSetPart -> EndSetPart***、***IniSetSheet -> EndSetSheet***)
- (*NestError.ErrorIOProject*) 访问或解释待读取文件时出错。

使用 *bMode=true* 还读取常规函数设置（对应 ***IniSettings -> EndSettings*** 部分）：不从文件读取的设置初始化为默认值。包含设置的读取用途对于测试情况很明显。
读取不包含一些不更改的设置：FixComputeError、TimerProgres、DirectoryTemp、RetrySquare。

如果 *bMode=false*：常规函数设置保持不变。

所有初步分配的排名均重置为零。

4 库使用指南

本章介绍将 TPA_N 集成到应用程序所需的信息。

TPA_N 库必须作为产品的组件集成到 32 位和 64 位 Windows 架构中。

4.1 安装程序

安装程序可以以正常或静默模式启动。

正常安装

常规安装会创建一个完整的演示环境，从所选文件夹开始。默认文件夹为 "C:\TpaNestingOEM"。

安装过程中会创建一些子文件夹：

- "bin"：用于存放演示可执行文件 (TpaNestingDEMO.exe) 和所使用的库。
- "\help"：TPA_N 库教程
- "\NestingOemCfg"：附件配置文件夹
- "\Examples"：套料示例项目
- "TestNesting"：TPA_N库集成演示项目（编程语言：C#、C、VbNET）
- "HyLicenses"：用于管理软件许可证的程序文件夹。

对于库的初步评估，建议并有必要进行标准安装。该演示应用程序提供了 TPA_N 集成的示例：该环境以简单直观的方式开发，提供了几乎所有可用选项，允许您评估其应用效果。

演示应用程序的配置文件 (TpaNestingDEMO.exe.config) 位于 "bin" 文件夹中：它包含了许可证识别程序运行所需的信息，并可作为为您自己的应用程序创建等效文件的参考。

HyLicenses.Exe 可执行文件位于 "HyLicenses" 文件夹中：该程序管理许可证的安装、更新和转移。有关如何使用该程序的说明，请参阅专门的文档。

静默安装

静默（或隐藏）安装会将安装限制为仅安装将库集成到应用程序所需的文件。静默安装是指不会显示所有常见提示（例如目标文件夹或是否接受许可条款），并且所有操作对计算机用户而言都是不可见的。

静默安装的典型用途是调用自定义安装过程。

以静默模式启动安装可识别命令行上的命令分配。

命令	意义
/PATH /INSTALLDIR	完整的产品安装路径（例如："INSTALLDIR=c:\abc"）。 所有库集成所需的文件都直接复制到文件夹中
/HYLIC	需要安装 "HyLicenses" 文件夹（用于管理软件许可证的程序文件夹）。 该文件夹是从（/PATH）创建的
/S	需要静默安装（在没有其他命令的情况下请求该命令）。

4.2 典型流程图

下面显示的顺序描述库的基本使用：

1. 创建 *TpaNestingOEM.Nesting()* 类的实例
2. 运行常规初步函数检查（键存在检查和检验启用的功能级别：*Square* 或 *True Shape*）
3. 记录 *回调* 函数
4. 打开常规函数设置的赋值部分，调用 *IniSettings()* 函数

5. 赋值常规函数设置（例如：*DirectoryTemp*、*TimeNesting*、□）
6. 关闭该部分，调用 *EndSettings()* 函数

7. 打开*部件*赋值部分，调用 *IniSetPart()* 函数
8. 赋值*部件*，调用 *AddPart()* 函数：形状赋值可以发生在 *IniGeometry()*、□、*EndGeometry()* 部分
9. 关闭*部件*赋值部分，调用 *EndSetPart()* 函数

10. 打开*群集*赋值会话，调用 *IniSetCluster()* 函数
11. 赋值*群集*，调用 *AddCluster()* 函数：群集组成，调用 *AddToCluster()* 函数
12. 关闭*群集*赋值会话，调用 *EndSetCluster()* 函数

13. 打开*板材*赋值部分，调用 *IniSetSheet()* 函数
14. 赋值*板材*，调用 *AddSheet()* 函数
15. 关闭*板材*赋值部分，调用 *EndSetSheet()* 函数

16. 开始优化过程，调用 *Compute()* 函数
17. 获得调用函数的结果：*ReadNumResult()*、*ReadResult()*、*ReadPartInResult()*
18. 自治获得/解释嵌套过程效率的信息。

设置*部件*、*设置*、*群集*和*板材*的赋值顺序无关：此处建议仅为示例。
常规设置、部件、群集和板材集构成*嵌套项目*，标识执行优化需要的数据集。

TPA_N 函数的顺序

设置、部分、集群和工作表的分配顺序无关紧要：建议的顺序仅是指示性的，但也可以肯定的是，它对应于信息的逻辑流。

如我们所见，每个赋值组中的函数调用必须遵循特定的顺序。为了确保正确的顺序，几乎所有函数都会执行输入级检查，以确定在收到调用时库是否处于正确的状态。如果发现库的状态不正确，函数将返回一个错误来指示具体情况。

因此，建议用户检查每个函数的返回值，以便在应用程序测试阶段直接识别错误情况。

常规设置、零件、集群和工作表的集合构成了所谓的嵌套项目，因为它标识了执行优化所需的数据集。

4.3 初步检查

调整应用程序函数，需要初步检查键存在、启用的功能级别以及额外选项：
- 通过查询 IsSquareEnabled 检查键的存在和有效性
- 通过查询 IsShapeEnabled 检查 TPA_N 的高级功能级别
- 检查高级功能级别后，通过查询 ShapeExtension 检查从外部文件读取图纸时的可用选项。

4.4 设置赋值

通常需要对常规函数设置执行初步赋值：
- 要调用的第一个函数是 *IniSettings()*
- 要调用的最后一个函数是 *EndSettings()*。

与设置赋值有关的所有属性必须在上述两个函数之间使用。
这里应注意，可以在不同上下文读取相同设置。
还可以读取以前保存的文件完成初步赋值设置：参见函数 *SaveSettings()*、*LoadSettings()*。

常规函数设置的一部分无疑更改更频繁，因为与*嵌套项目*直接相关：外部应用程序决定如何组织设置，采用的标准，以及更新速率。

设置更改将禁止执行新优化尝试：以后调用 RetryCompute 函数将失败。

4.5 赋值部件

我们继续研究部件列表的赋值：
- 要调用的第一个函数是 *IniSetPart()*
- 要调用的最后一个函数是 *EndSetPart()*。

如果部件列表赋值为全部，用参数 *Clear=true* 调用 *IniSetPart*。
部件更改将禁止执行新优化尝试（以后调用 RetryCompute 函数将失败）。

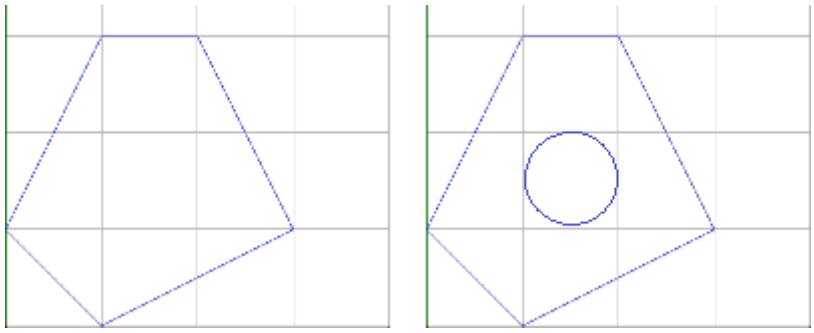
应强调部件赋值的一些基本方面：
- 每个部件具有严格为正 (> 0) 的唯一数字标识符：*NestPart* 结构的 *ID* 字段
- 因此无法将同一 *ID* 赋值给多个部件
- 标识符可以采用任何顺序赋值，不一定连续
- 应注意，ID 在部件排序中起重要作用：其他重要设置（如优先级、废料信息）相同的情况下，ID 决定哪个部件在优化过程中具有更高的优先级。（反向对应：ID越低，优先级越高）。

要添加部件，调用函数 *AddPart()*。
要取消部件，用参数 *OnlyGeometry=false* 调用函数 *RemovePart()*。
要更改部件，调用函数 *WritePart()*。
要更改已经插入部件的形状，用参数 *OnlyGeometry=true* 调用函数 *RemovePart()*。

如何赋值部件形状

可以为已经赋值的*部件*定义形状。
部件形状赋值必须在段内（***IniSetPart -> EndSetPart***）。

图中显示两个形状示例：
- 左侧，仅赋值外部几何的简单形状
- 右侧，还赋值了内部几何（孤岛）的更复杂形状。



每个几何独立赋值：
- 要调用的第一个函数是 *IniGeometry()*
- 要调用的最后一个函数是 *EndGeometry()*。

IniGeometry 函数指示：
- 进行赋值的部件
- 是外部还是内部几何。

只能为每个部件赋值一个外部几何，在内部几何前。
对于零件，可以仅分配内部几何形状：外部零件自动分配为矩形，应用为零件设置的尺寸，并将左下边缘定位在坐标轴的原点上。

示例代码

```
TpaNestingOEM.Nesting nestObj=new TpaNestingOEM.Nesting();

...
NestPart Item=new NestPart();

//打开部件赋值段
If (nestObj.IniSetPart(true))
{
    //部件结构最小赋值 (ID=1, N=20)
    Item.Initialize (1);
    Item.N=20;
    //添加部件
    nestObj.AddPart(Item);
    //打开 ID=1 的部件外部几何的赋值部分
```

```
        nestObj.IniGeometry(1, false, 0.0, 100);
        nestObj.AddToGeometry_Line(100, 300);
        nestObj.AddToGeometry_Line(200, 300);
        nestObj.AddToGeometry_Line(300, 100);
        nestObj.AddToGeometry_Line(100, 0);
        //关闭（外部）几何的赋值部分
        nestObj.EndGeometry();

        //打开 ID=1 的部件内部几何的赋值部分
        nestObj.IniGeometry(1, true, 100, 150);
        nestObj.AddToGeometry_Circle(150, 150);
        //关闭（内部）几何的赋值部分
        nestObj.EndGeometry();

        //添加其余部件
        …
        //关闭部件的赋值部分
        nestObj.EndSetPart();
    }
```

示例代码：外部几何采集循环

```
    bool GeometryFromFile (string pathName)
    {
        ////函数返回时管理错误
        if (nestObj.ReadShape (pathName) != NestErrors.ErrorNone) return false;

        //初始化读取循环使用的变量
        int IndexItem = 0;              //几何索引
        int IndexElement = 0;           //几何元素索引（IndexItem)
        NestGeometry Item=new NestGeometry();

        //采集几何循环
        while (nestObj.ReadGeoInShape (IndexItem, IndexElement, ref Item))
        {
            //……处理索引几何（IndexItem）的（Item)=1' 元素

            IndexElement++;             //几何元素索引增加（IndexItem)
            while (nestObj. ReadGeoInShape (IndexItem, IndexElement, ref Item))
            {
                //……处理（Item)= 索引几何（IndexItem）的新元素

                IndexElement++;         //几何元素索引增加（IndexItem)
            }
            IndexItem ++;               //增加几何索引
            IndexElement=0;             //初始化几何元素索引（IndexItem)
        }
        return true;
    }
```

4.6 赋值板材

我们继续研究板材列表的赋值：

- 要调用的第一个函数是 *IniSetSheet()*
- 要调用的最后一个函数是 *EndSetSheet()*。

如果板材列表赋值为全部，用参数 *Clear=true* 调用 *IniSetSheet*。
板材更改将禁止执行新优化尝试（以后调用 RetryCompute 函数将失败）。

和部件一样，强调板材赋值的一些基础方面：

- 每个板材具有严格为正 (> 0) 的唯一数字标识符：*NestSheet* 结构的 *ID* 字段
- 因此无法将同一 ID 赋值给多个板材
- 标识符可以采用任何顺序赋值，不一定连续
- 应注意，ID 在板材排序中起重要作用：其他重要设置（如优先级、废料信息）相同的情况下，ID 决定哪个板材在优化过程中具有更高的优先级。

要添加板材，调用函数 *AddSheet()*。
要移除板材，用参数 *OnlyGeometry=false* 调用函数 *RemoveSheet()*。
要修改板材，调用函数 *WriteSheet()*。
要更改已经插入的板材*形状*，用参数 *OnlyGeometry=true* 调用函数 *RemoveSheet()*。

如何赋值板材形状

可以为已经赋值的*板材*定义形状。
*板材*形状赋值必须在段内 (***IniSetSheet -> EndSetSheet***)。

赋值模式和*部件*相同。

4.7 如何赋值手动集群

我们继续研究群集列表的（可选）赋值：
- 调用的第一个函数为 *IniSetCluster()*
- 调用的最后一个函数为 *EndSetCluster()*。

如果群集列表赋值全部，调用 *IniSetCluster*，参数 *Clear=true*。
群集更改将无法执行新优化尝试（接下来调用 RetryCompute 函数失败）。

应强调群集赋值的一些基本方面：
- 每个群集具有严格为正 (>0) 的唯一数字标识符：*NestCluster* 结构中的 *ID* 字段
- 因此，无法向更多群集赋值同一 *ID*
- 标识符可以采用任何顺序赋值，不一定连续
- 应注意，ID 在群集排序中起重要作用：其他重要设置（如优先级、群集信息）相同的情况下，ID 决定哪个群集在优化过程中具有更高的优先级

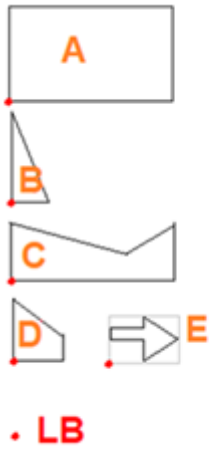
要添加群集，调用函数 *AddCluster()*。
要定义群集构成，调用函数 *AddToCluster()*。

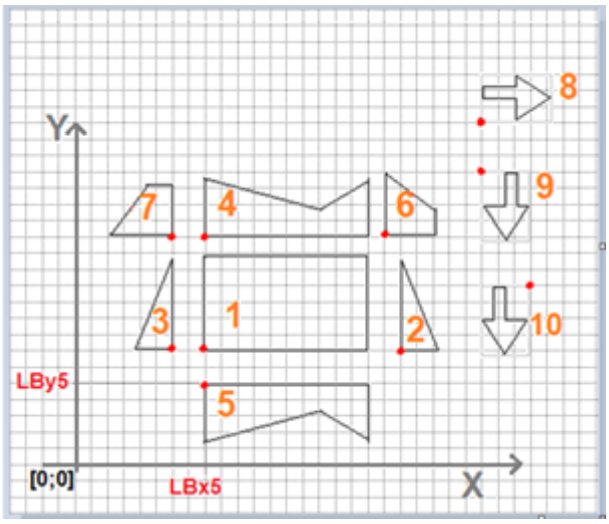
要移除群集，调用函数 *RemoveCluster()*，参数 *OnlyGeometry=false*。
要修改群集，调用函数 *WriteCluster()*。
要更改已经插入的群集*构成*，调用函数 *RemoveCluster()*，参数 *OnlyGeometry=true*。

如何赋值群集组成

可以为已经赋值的*群集*定义组成。
群集组成赋值必须在部分 (***IniSetCluster -> EndSetCluster***) 中。
定义群集组成，调用 ***AddToCluster*** 函数。

图片显示生成群集的规则示例：
- 左侧是可放置的部件
  - 字母 (A, B, C, D, E) 可以对应部件的说明名称（*NestPart* 结构的 *Label* 字段）
  - 每个部件上显示的红色圆圈为每个部件边框的 LB（左下）点
  - 仅对于 (E) 部件，LB 点不在外形上：在 (E) 周围显示边框

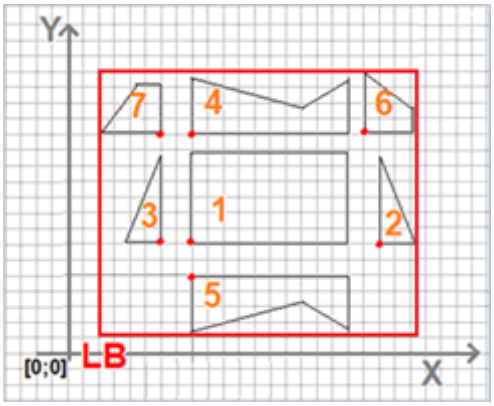




- 右侧可以在群集中使用部件
  - 指示 10 个部件（从 1 到 10 编号）
  - 表格显示群集构造信息（参见下文：示例代码）：

#	*部件*	X	Y	Mirror	A°
1	A	LBx1	LBy1	0	0
2	B	LBx2	LBy2	0	0
3	B	LBx3	LBy3	x	0
4	C	LBx4	LBy4	0	0
5	C	LBx5	LBy5	y	0
6	D	LBx6	LBy6	0	0
7	D	LBx7	LBy7	0	90
8	E	LBx8	LBy8	0	0
9	E	LBx9	LBy9	0	-90
10	E	LBx10	LBy10	x	90

- 前 7 个元素上的群集结构，图显示
  - 群集边框（红框）
  - 群集的 LB 点。



示例代码

```
TpaNestingOEM.Nesting nestObj=new TpaNestingOEM.Nesting();

...
NestCluster Item=new NestCluster();
ItemCluster OneItem=new ItemCluster();


//打开群集赋值部分
If (nestObj.IniSetCluster(true))
{
  //群集结构最小赋值 (ID=1, N=20)
  Item.Initialize (1);
  Item.N=20;
  //添加群集
  nestObj.AddCluster(Item);
  //赋值群集组成
  nestObj.AddTocluster(Item.ID, new ItemCluster() { NameID = "A", X = 160.0, Y = 140.0, Mirror = 0, A =
  0.0});
  nestObj.AddTocluster(Item.ID, new ItemCluster() { NameID = "B", X = 400.0, Y = 140.0, Mirror = 0, A =
  0.0});
  nestObj.AddTocluster(Item.ID, new ItemCluster() { NameID = "B", X = 120.0, Y = 140.0, Mirror = 1, A =
  0.0});
  nestObj.AddTocluster(Item.ID, new ItemCluster() { NameID = "C", X = 160.0, Y = 280.0, Mirror = 0, A =
  0.0});
  nestObj.AddTocluster(Item.ID, new ItemCluster() { NameID = "C", X = 160.0, Y = 100.0, Mirror = 2, A =
  0.0});
  nestObj.AddTocluster(Item.ID, new ItemCluster() { NameID = "D", X = 380.0, Y = 280.0, Mirror = 0, A =
  0.0});
  nestObj.AddTocluster(Item.ID, new ItemCluster() { NameID = "D", X = 120.0, Y = 280.0, Mirror = 0, A =
  90.0});
  ...

  //添加剩余群集
  ...
  //关闭群集的赋值部分
  nestObj.EndSetCluster();
}
```

4.8 初步放置的分配

我们进入（可选的）初步放置列表分配阶段：
- 第一个需要调用的函数是*IniSetPlaced()*
- 最后一个需要调用的函数是*EndSetPlaced()*。

如果列表分配是全量的，调用*IniSetPlaced*时需带上参数*Clear=true*。
初步放置的更改会抑制进行新优化尝试的可能性（随后调用RetryCompute函数将失败）。

要添加一个放置，调用函数*AddPlaced()*。
要删除一个放置，调用函数*RemovePlaced()*。

示例代码

```
TpaNestingOEM.NestingnestObj=new TpaNestingOEM.Nesting();

...
NestPlacedItem=newNestPlaced();


//打开初步放置的分配节
If (nestObj.IniSetPlaced(true))
{
```

```
    //在第一张板材上的第一个放置（ID=3 的集群）
    Item.Initialize (); Item.Type = true; Item.ID = 3;
    Item.X = 30.0; Item.Y = 50.0; Item.A = 0.0; Item.Mirror = 0;
    nestObj.AddPlaced(1, item);
    //在第一张板材上的第二个放置（ID=1 的部件）
    Item.Initialize (); Item.ID = 1;
    Item.X = 420.0; Item.Y = 50.0; Item.A = 30.0; Item.Mirror = 0;
    nestObj.AddPlaced(1, item);

    //添加剩余的放置
    …
    //关闭初步放置的分配节
    nestObj.EndSetPlaced();
}
```

4.9 执行嵌套优化

要开始嵌套项目优化，需要调用 Compute 函数。
在优化前运行部件和板材检查，还导致函数返回错误，取消优化：

- 必须为可用放置和板材请求部件
- 匹配筛选器检查（厚度、材料、颜色）必须能够匹配至少一个部件和一个板材。

优化时，外部应用程序可以管理回调函数，最终请求中断优化。

获取求解结果

完成优化后，可以获取嵌套求解结果。

一些属性/方法返回有关优化过程的常规信息：
- *IsComputed*：有效优化状态
- *ModeCompute*：执行优化的相关信息（*Square* 或 *True Shape*）
- *TimeOut*：计算过程是否因超时结束的信息
- *CanRetry*：请求新优化的可能。

一组函数支持获取优化求解的所有信息：
- *Fitness*：求解效率
- *ReadNumResult*：求解或一类板材总数的信息
- *ReadNumPartInResult*：秋季放置数量或板材和/或一种部件的信息
- *ReadNumClusterInResult*：求解放置数量或板材和/或群集类型信息
- *ReadResult*：单个求解板材的常规信息
- *ReadPartInResult*、*ReadGeoInResult*：一个求解板材上的单个放置相关信息
- *SaveSolution*：将求解保存到文件（XML 格式）
- *SaveSolutionDXF*：将求解保存到一个或多个文件（DXF 格式）。

示例代码：如何获取求解板材的常规信息

```
TpaNestingOEM.Nesting nestObj=new TpaNestingOEM.Nesting();

…
//组件查询的赋值字段
int ID_Sheet = 0, Item_Sheet = 0, NumPlaces = 0, Repetition= 0;
double AreaPercent = 0.0;

//求解板材的查询循环
int IndexSheet = 0; //板材索引
while (nestObj.ReadResult (IndexSheet, ref ID_Sheet, ref Item_Sheet, ref AreaPercent, ref NumPlaces, ref
Repetition))
{
    //…
    //板材放置的获取循环
    //……
    IndexSheet ++; //增加板材索引
}
```

示例代码：板材放置的获取循环

```
// IndexSheet = 板材索引（参见：上一个循环）
int IndexBox = 0; //板材上的放置索引
NestBox nestBox=new NestBox();

while (nestObj.ReadPartInResult (IndexSheet, IndexBox, ref nestBox))
{
    // ……
    // 参见：“放置几何的获取循环”
    //……

    IndexBox++; //增加放置索引
    // ……
}
```

示例代码：放置几何的获取循环

必须在调用函数 ***ReadPartInResult*** 后运行读取循环。

```
int IndexItem = 0;      //部件几何索引
int IndexElement = 0;   //几何元素索引（IndexItem）
NestGeometry Item=new NestGeometry();

//部件几何循环
while (nestObj.ReadGeoInResult (IndexItem, IndexElement, ref Item))
{
    //……处理索引几何（IndexItem）的（Item）= 1’元素

    IndexElement++;             //几何元素索引增加（IndexItem）
    while (nestObj.ReadGeoInResult (IndexItem, IndexElement, ref Item))
    {
        //……处理（Item）= 索引几何（IndexItem）的新元素

        IndexElement++;         //几何元素索引增加（IndexItem）
    }
    IndexItem ++;       //增加几何索引
    IndexElement=0;     //初始化几何元素索引（IndexItem）
}
```

计算和管理连续求解

利用 *True Shape* 优化，可以计算更多求解，最多 20 个：

- CanRetry 属性获得是否可以开始新优化的信息
- 调用 RetryCompute 函数运行新优化。

完成优化后，计算求解自动设为当前求解。
所有计算求解都是可用的，可以浏览，也可以将当前求解更改为特定求解。
可以调用 SaveSolution 函数，将当前求解保存到文件（或 SaveSolutionDXF）。

示例代码：浏览连续求解

```
TpaNestingOEM.Nesting nestObj=new TpaNestingOEM.Nesting();

// GoToNextSolution：转到下一个计算的求解
bool GoToNextSolution()
{
    If (nestObj.Solution < nestObj.OfSolution)
```

```
    {
        nestObj.Solution = nestObj.Solution+1;
        return true;
    }
    return false;
}

// GoToPrevSolution：转到上一个计算的求解
bool GoToPrevSolution()
{
    If (nestObj.Solution >1)
    {
        nestObj.Solution = nestObj.Solution-1;
        return true;
    }
    return false;
}
```

4.10 处理回调函数

该函数用于监测优化阶段的进展。此时仅一个函数可用。

Progres 函数

管理事件允许取消或中断优化过程。
- 取消导致结束优化阶段，完全删除该过程
- 将其中断，导致在完成第一个有效求解后结束优化阶段。

```
// Progres 函数
void ProgresFromNesting (int nValue, ref bool bCancel, ref bool bPause)
{
    //更新进料窗口
    ...
    //用户选择交互式管理：
    // CancelCondition=true -> 优化中断
    // StopCondition=true -> 快速退出优化的条件

    If (CancelCondition) bCancel=true;
    else if (StopCondition) bPause=true;
}
```

4.11 保存和读取 TPA_N 项目

可以通过函数 SaveProject 和 LoadProject 保存和检索项目。
这两个函数还支持保存和检索常规函数设置。
请求保存还包括常规设置的项目可以用于测试，如请求测试检索关键情况。

执行 *LoadProject* 后，外部应用程序必须更新部件和板材，从 *TPA_N* 读取信息。

示例代码

```
TpaNestingOEM.Nesting nestObj=new TpaNestingOEM.Nesting();

//结构
NestPart ItemPart=new NestPart();
NestSheet ItemSheet=new NestSheet();
NestSize ItemSize=new NestSize();
NestGeometry ItemGeo=new NestGeometry();
NestCluster ItemCluster=new NestCluster();
```

```
ItemCluster OneItemCluster=new ItemCluster();

//读取项目
if (nestObj.LoadProject ("C:\projects\one.xml", false) == NestErrors.ErrorNone)
{
    //---------- 读取部件
    for (int idxElement=0; idxElement< nestObj.CountPart; idxElement++)
    {
        nestObj.ReadPartIndex (idxElement, ref ItemPart, ref ItemSize);

        int IndexItem = 0;        //部件几何索引
        int IndexElement = 0;     //几何元素索引 (IndexItem)

        //部件几何循环
        while (nestObj.ReadGeometry (0, ItemPart.ID, IndexItem, IndexElement, ref ItemGeo)) ==
                                                                    NestErrors.ErrorNone)
        {
            // ……处理索引几何 (IndexItem) 的 (ItemGeo) = 1' 元素

            IndexElement++;      //几何元素索引增加 (IndexItem)
            while (nestObj ReadGeometry (0, ItemPart.ID, IndexItem, IndexElement, ref ItemGeo)) ==
                                                                        NestErrors.ErrorNone)
            {
                // ……处理 (ItemGeo) = 索引几何 (IndexItem) 的新元素

                IndexElement++; //几何元素索引增加 (IndexItem)
            }
            IndexItem ++;                  //增加几何索引
            IndexElement=0;                //初始化几何元素索引 (IndexItem)
        }
    }


    //---------- 读取板材

    for (int idxElement=0; idxElement< nestObj.CountSheet; idxElement++)
    {
        nestObj.ReadSheetIndex (idxElement, ref ItemSheet, ref ItemSize);
        int IndexItem = 0;        //板材几何索引
        int IndexElement = 0;     //几何元素索引 (IndexItem)

        //板材几何循环
        while (nestObj.ReadGeometry (1, ItemShet.ID, IndexItem, IndexElement, ref ItemGeo)) ==
                                                                    NestErrors.ErrorNone)
        {
            // …
        }
    }

    //----------- 读取群集
    for (int idxElement=0; idxElement< nestObj.CountCluster; idxElement++)
    {
        nesObj.ReadClusterIndex (idxElement, ref ItemCluster);

        int IndexItem = 0;        // 群集赋值索引

        //群集赋值循环
        while (nestObj.ReadInCluster ( ItemCluster.ID, IndexItem, ref OneItemCluster) ==
        NestErrors.ErrorNone)
        {
            // …
            IndexItem ++;        // 增加赋值索引
        }
```

```
        }
    }
```

Tecnologie e Prodotti per l'Automazione S.r.l.

Via Carducci 221
I - 20099 Sesto S. Giovanni (MI)
Ph. +393666507029

www.tpaspa.com

info@tpaspa.it

2026/3/6