Standard Operating Procedure

Bundle Optimization Tool

Last Updated July 28, 2025

By Braedon McRae

# Scope and Purpose

This Standard Operation Procedure (SOP) document will cover the usage and technical details of the Bundle Optimization Tool (the “Program”). The tool is designed to efficiently pack multiple sub-bundles (each composed of 1+ SKUs) into optimized shipping bundles based on size, weight, stacking capability, and other packing rules.

This SOP applies to any users of the Program who are responsible for preparing and handling SKU bundles for shipment.

# Usage

## Usage Requirements

* Windows OS.
* A distribution of the Program.
* Excel file containing order data in the format of a “*SO-PackExportData*” sheet.
  + A sample version of this file can be found by clicking the “Open Example Input File” button in the User Interface (UI).
* A completed “*Sub-Bundle\_Data.xlsx*” file for dimensional information.
  + This file comes with the Program distribution and can be found in the “*\_internal*” directory. This file may be modified by the user as sub-bundle requirements change.
* (Optional) An already optimized file of orders, which the Program can open and add more orders to once bundle optimization is complete.

## Usage Instructions

1. The user may want specific SKUs to be placed into specific bundles. The user can do this by setting a unique identifier in the input file, in the “Bdl\_Override” (bundle override) column. The Program separates sub-bundles during packing based on this identifier.
2. Launch the Program.
3. Click the first “Browse” button to select the Excel file with the “*SO-PackExportData*” sheet.
4. (Optional) Click the second “Browse” button to select an existing file to append to. This file is the resultant file from a previous optimization using the Program.
5. Click “Perform Bundle Optimization” and wait for the process to complete. The Program may freeze when computing large orders, this is normal.
6. After optimization is complete:
   * Click “Open Images Folder” to view bundle diagrams for each order.
   * Click “Open Resultant Excel File” to view the resultant data.

## Output Files

* **Optimized\_Bundles.xlsx**: The resulting packed data file.
* **Bundle images**:Visuals of each packed bundle, saved as PNG files.

# Algorithm Mechanics

## Optimization Process Overview

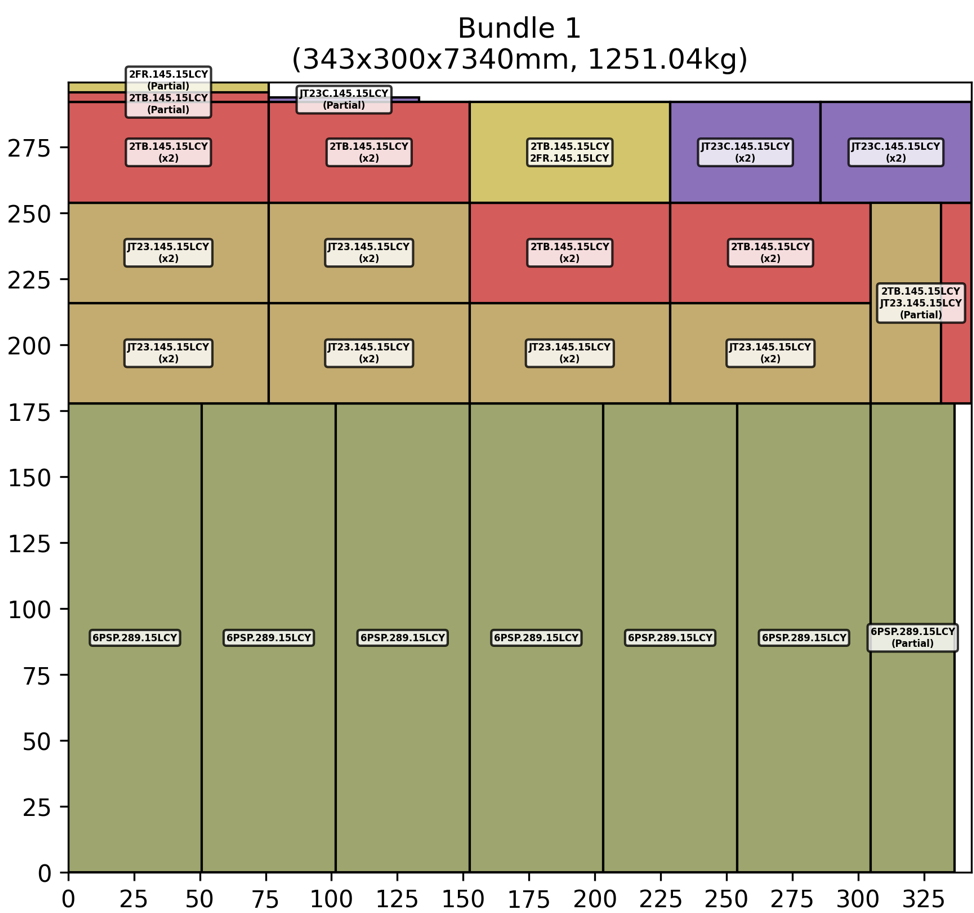
* **Data Preparation:**
  + Loads order data and pulls sub-bundle data from respective Excel files.
  + Cleans data, handles SKUs with empty or invalid data (they will be placed in bundle ‘0’).
  + SKUs flagged with an identifier in “Bdl\_Override” are processed into separate bundles.
  + Remaining SKUs are put into sub-bundle items and grouped by color.
  + If the piece amount specified in the order data is not enough to create a full sub-bundle, it is denoted with the “Partial” nomenclature in the inventory ID.
* **Row-Based Packing:**
  + SKUs that are eligible for bottom-row packing (as defined in sub-bundle data file) are prioritized to be packed along the bottom to ensure a heavy/strong base.
  + The first row of sub-bundles is packed vertically (on edge). Subsequent rows may have horizontal or vertical sub-bundles, but horizontal is preferred.
  + The Program tries to pack the same sub-bundles adjacent to one another to reduce variation.
  + Stacking is attempted (stacking multiple sub-bundles lengthwise) if available.
* **Greedy Filler Algorithm:**
  + Once a row is finished packing, the Program performs a greedy search over the whole bundle (up to the height of the current row) and checks if there are gaps where any of the remaining sub-bundles can be placed. If so, the sub-bundle is placed there.
  + Optional filler materials (Filler\_44 or Filler\_62) can be placed into these spaces if there are no more valid sub-bundles to place.
  + Short sub-bundles (length <= 609mm) may be inserted into filler material. If there are any that cannot be inserted into filler, they are placed at the top of the bundle.
* **Adaptive Layout Adjustment:**
  + If poor space usage is detected close to the top of the bundle, the algorithm attempts re-packing the bundle with reduced height or width.
  + Multiple iterations of this process may be run for each bundle to find optimal dimensions.
  + If the packing iterations complete and the bundle height is still greater than width, the sub-bundles are laid flat and stacked upwards. Supporting lumber is added to the packaging materials.
* **Merging and Packaging:**
  + Bundles of two different colors may be merged if all sub-bundles of both colors can fit into one bundle.
  + Standard packaging SKUs are added based on bundle dimensions and orientation (e.g., angleboards, dunnage, padding).
* **Final Output:**
  + Bundle visualizations are generated and saved as images.
  + Packed results are written to a new or appended Excel file, including timestamps and metadata.

## Feature Variables

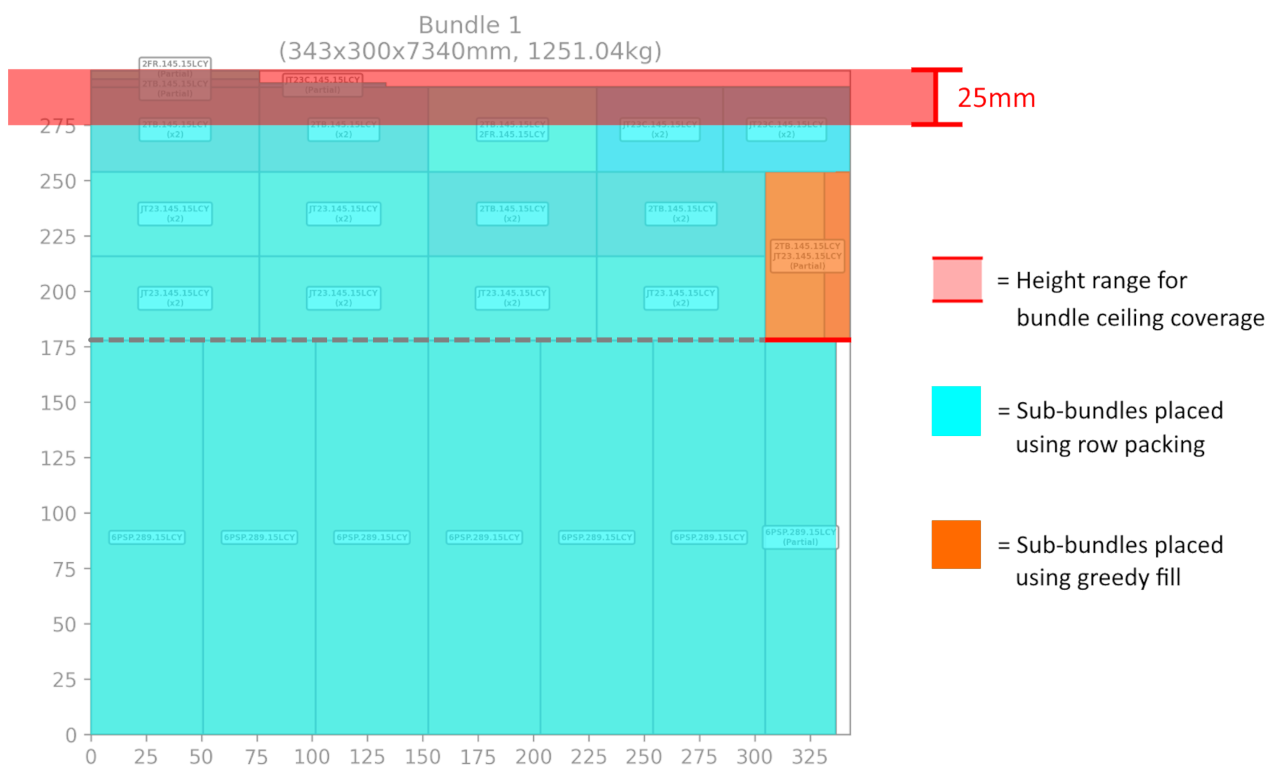
The following values impact how sub-bundles are optimally packed into bundles:

* SKU dimensions and weight.
* Length thresholds: SKUs > 3680mm trigger 7340mm bundles.
* Bundle max dimensions: Width = 559mm, Height = 559mm, Length = 3680/7340mm.
* Empty space limitations at the top of the bundle: at least 90% of the bundle’s width must have a sub-bundle at most 25mm from the bundle ceiling.
* Sub-bundles must have 85% coverage of their base to be considered supported and eligible for placement.
  + ‘Coverage’ is defined as: another sub-bundle exists ± 5mm vertically from the base of the sub-bundle currently being placed.
* Greedy filler uses a grid with 25mm increments to identify candidate points for new sub-bundles to be placed.
  + 5mm increments used for filler material at the end.
* Greedy filler prioritizes the right side of the bundle, intended to balance the left-leaning structure caused by packing rows left-to-right.
* Bundle height must be 0.5 – 1x the bundle width.
* Tick marks are used in the visualization images. Tick marks are shown every 25mm unless the dimension exceeds 400mm, then tick marks are shown every 50mm.

*Figure 1* (shown below) is an example bundle that is generated using the Program. *Figure 2* shows how these variables are used while packing this bundle.



**Figure 1:** Example bundle from Order 1013350



**Figure 2:** Example bundle with optimization features highlighted

From the figures, we see that:

1. This bundle is 7340mm long because there exist 7340mm sub-bundles (along the bottom row, stacked vertically). This allows shorter bundles to stack lengthwise, as shown.
2. This bundle is 343mm (width) x 300mm (height), satisfying the dimension restrictions.
3. The red band at the top of *Figure 2* shows the tolerance for what is an acceptable total height. 100% of this bundle is within that range, which satisfies this restriction.
4. All sub-bundles except one have 100% of their base covered. This sub-bundle (the one with the red line at its base) still has enough base coverage to be placed, at around 85%.
5. The same sub-bundle, colored in orange, is the only sub-bundle in this bundle that was placed using the greedy fill algorithm.
6. Tick marks are shown every 25mm in the visualization of this bundle.
7. The grey dashed line signifies the switch from vertical packing (first row) to horizontal.