Background Info and Model Design

In the Portland market, 7% order requests were rejected because of the online parcel packing algorithm,

so this project is to develop a better packaging strategy with which the parcel dimension should be more reasonable.

Asssumption:

1. All item(SKU) shape are rectangle.
2. All parcel are rectangle.
3. weights are ignored.

Target:

1. parcel dimension (length, height, depth)should be smaller than dimension limit so the order won't be declined.
2. parcel volume should as small as possible so the delivery cost will be low.
3. target 1 has higher priority than target 2

The problem:

The packaging strategy is to solve a container loading problem. Basically, our model is going the answer the following 6 questions:

1. How to represent available spaces to load SKU in.
2. How to select a SKU to load in a selected space.
3. How to select a space to load a SKU in.
4. How to select a position in a space where the selected SKU should be load on.
5. How to decide the face direction of a SKU in a space.
6. How to update available spaces for next load task.

The following is how our model answer the 6 quesions:

1. we assume each space is a rectangle. In the 3d coordinate system (x, y, z coordinate system where x represents height, y represents length, z represents depth; also x is how hight, x is how right, z is how behind) , we can represent it by two points: the start point and end point. The Start point is the vertex of the rectangle with smallest x, y, z value and end point is with biggest x,y,z value.
2. We select the SKU with biggest sides to load first. The "big" is defined by the biggest dimension of a SKU. The motivation is stated by the following assumption.
    a. load a bigger SKU is a more challenge task than a small one.
    b. We have more space at earlier stage than latter stage.
    c. Thus to take less risk, we should load big item first.
3. Among all spaces that can include the SKU, we select the space with smallest start point to load SKU. By doing this, we re going to pick the space with lowest, leftest, frontest start point. The motivation behind this is that we want the each SKU be as close to (0,0,0)(lowest, leftest, frontest) point as possible. Thus, SKU can be loaded in a compact way.
4. We select the start point of the selected space to place SKU on. The motivation is that by doing this, SKU can be loaded in a compact way.
5. We try to pair the shortest side of the space with the biggest side of SKU. By doing this, we have more spacious space with bigger side not occupied by the SKU.
6. We maintain multiple spaces, once a SKU is loaded into any space, then we update maintained spaces. We only maintain spaces that cannot be included by others but these spaces can have overlap.
    a. For a space, if the space has no overlap with the SKU, then do nothing.
    b. For a space, if the space has overlap with the SKU, then for each surface of the SKU, if the surface is strictly in the space, it cuts the space and then generate a new space.
    c. For the new generated space, if it is included by any old space, then ignore it; if not included, then add it into maintained space.