Explanation
Presentation

Best Fit Decreasing serves as the main algorithm, efficiently packing the items, especially when they are similar in size. However, when there are mixed products (items of varying sizes or shapes), or when we know the initial packing solution might be prone to errors (such as wasted space or inefficient arrangements), the Genetic Algorithm acts as a fine-tuning layer. It improves upon the initial packing by optimizing the arrangement and correcting any inefficiencies that the Best Fit Decreasing algorithm may have missed.

Best Fit Decreasing

- Step 1: Sort all the cubes by size in decreasing order.

First, we need to sort the items by their size. We can either go by item volume or just look at their biggest dimension (like height, width, or length).

Why? Because the big items are usually trickier to fit, and if we don't deal with them first, they might end up leaving a lot of empty space if we just toss them in later.

Step 2: Place Each Item into the Best Fitting Box

Now, it’s not just about shoving the item into any box where it fits by volume. We also need to make sure the item can fit into a box considering all three dimensions: height, width, and length.

For each item, we look at the boxes we already have and figure out which one it will fit into best—we’re trying to minimize the empty space left over. It’s like playing a 3D puzzle where you want each piece to fit just right.

If none of the existing boxes work (because the item doesn’t fit dimensionally), then we’ll open up a new box and start filling that one.

Step 3: Handle Box Rotation (Situational :))

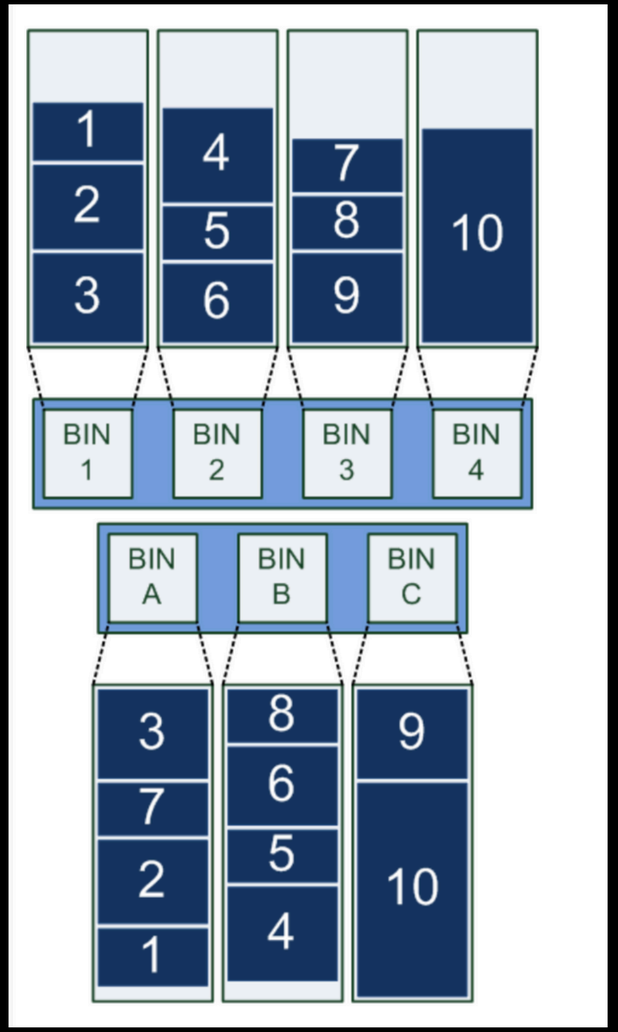
Sometimes, to get a better fit, we might need to rotate the items. If something doesn’t fit the way it’s currently oriented, we can try flipping or turning it to see if it fits better. For example, if a 4x4x10 item doesn’t fit in its current position in Box B, we can rotate it so the 10-length side lines up with a different side of the box.

Step 4: Create a New Box When No Existing Box Fits

If none of the boxes can fit the item, even after trying different orientations, then it’s time to open a new box. The process just keeps repeating like this: checking each item, rotating if needed, and adding new boxes as necessary until everything is packed.

Genetic Algorithm: Fine-Tuning the Packing Check the Fitness of the Current Solution

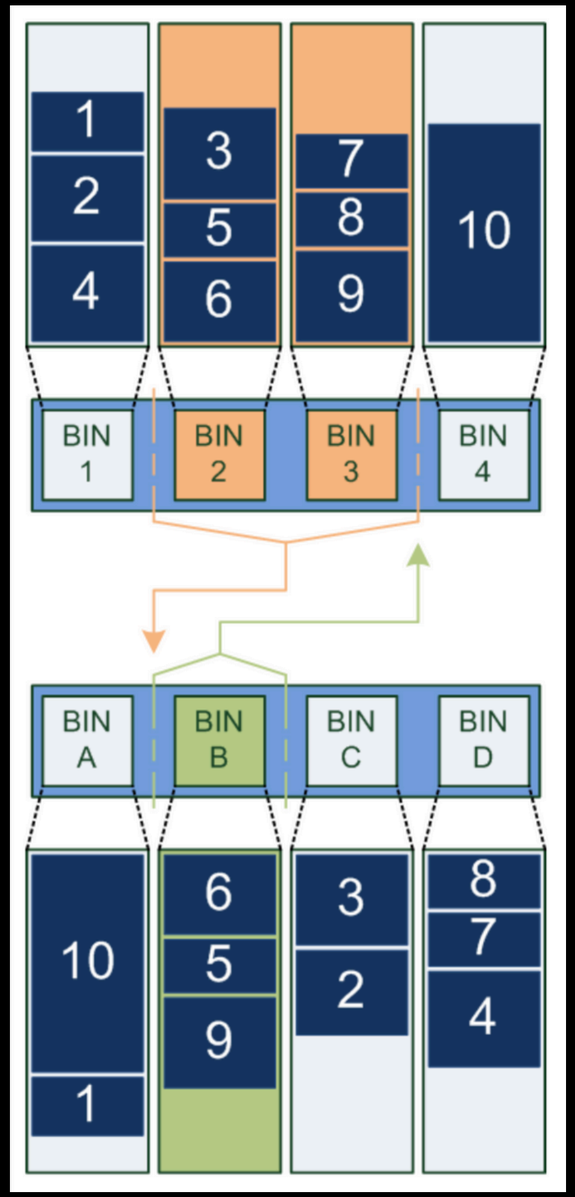
1. Start with what we've got: The first set of solutions comes from the packing we just did with BFD. We call these our population of solutions.
2. First, we evaluate how good our current packing solutions are. In the top row, you can see four bins (Bin 1, Bin 2, Bin 3, Bin 4) with items packed inside them. Some of these bins have extra space, which we want to reduce to improve efficiency.
3. Fitness is all about how well each bin is packed—fewer bins used and less wasted space means a better solution!



Crossover: Mixing the Good Solutions

Now, to improve this, we use crossover. Look at Bin 2 and Bin 3 in the middle. Items in these bins (like 3, 5, 6, 7, 8, and 9) are swapped between bins to try and find a better arrangement.

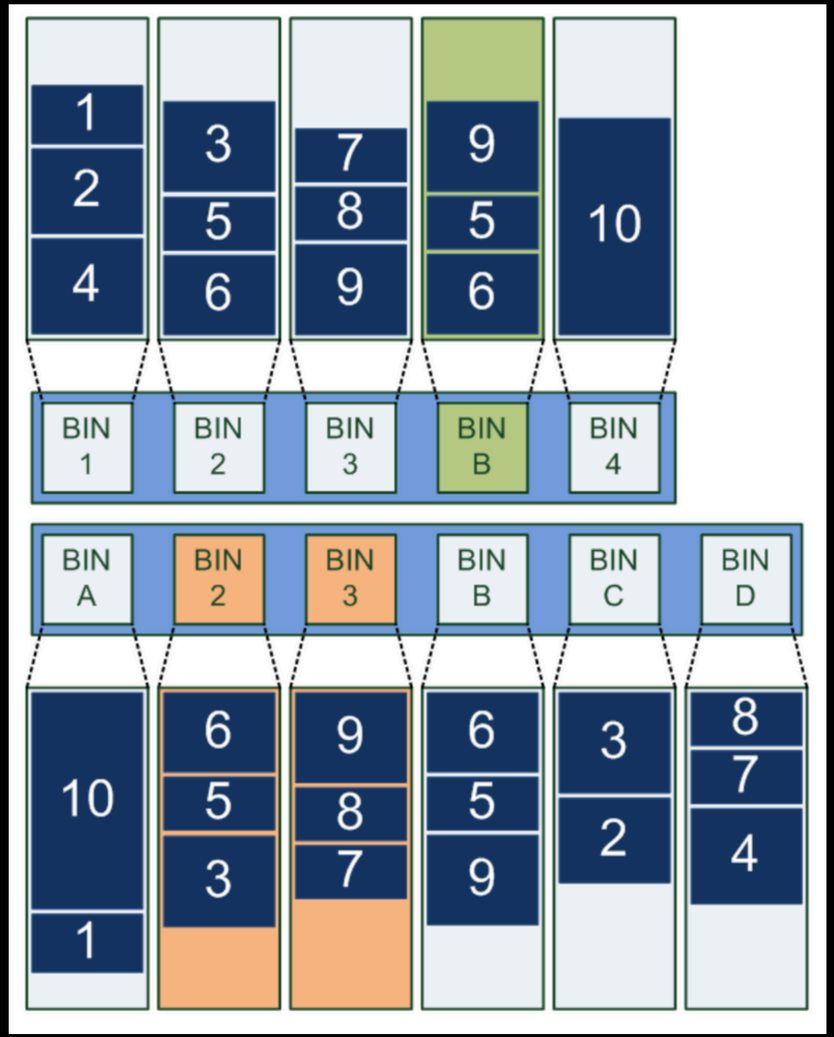
The idea here is that we take parts from two different "parent" bins and mix them to create a new combination, hoping it will pack better overall.



Mutation: Small Changes for Improvement

- Then comes mutation, where small random changes are made to the arrangement. For example, maybe an item like 6 gets moved from Bin 2 to Bin B , or items are rotated to fit better.

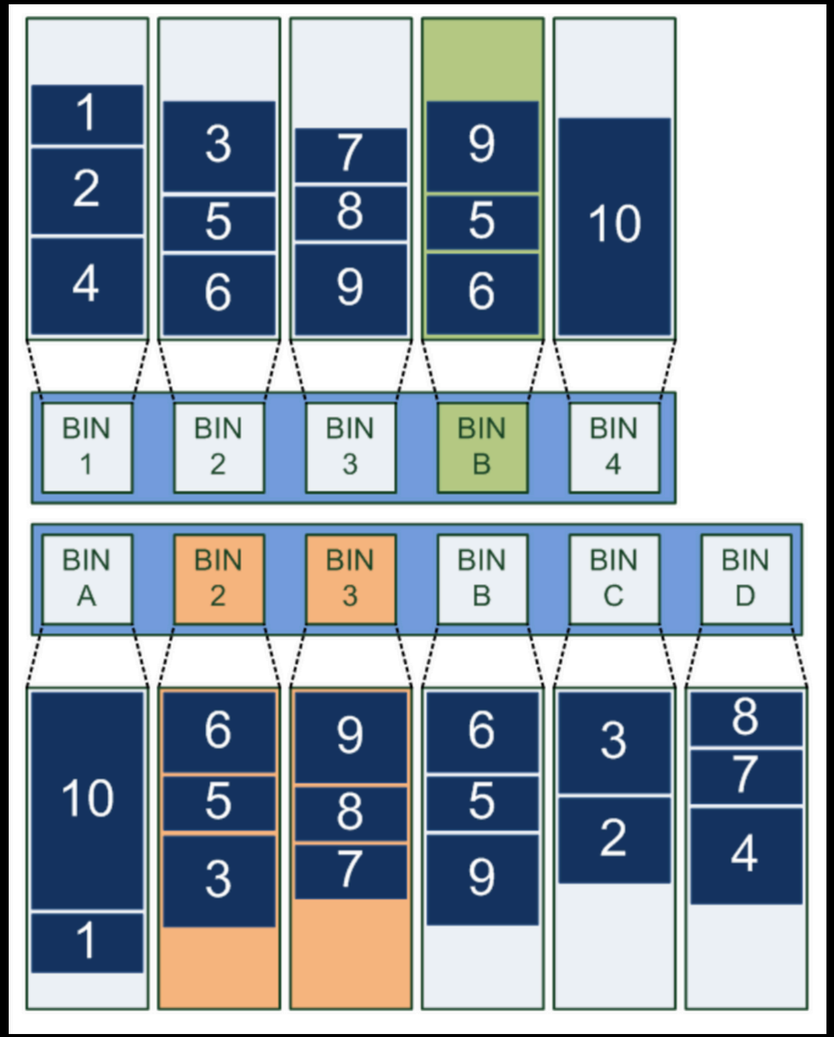
- These tiny adjustments give the algorithm more variety and help it explore different ways to optimize packing. We want to avoid getting stuck with just one approach, so mutation gives us new options to test.



Mutation: Small Changes for Improvement

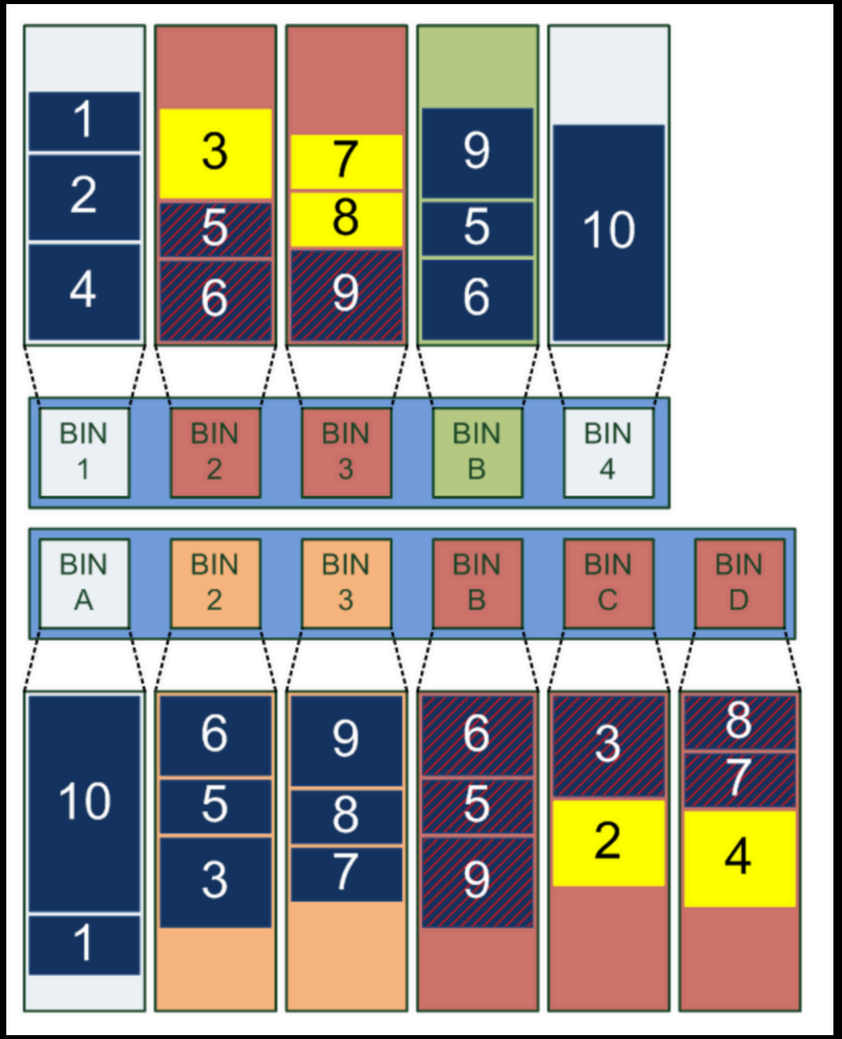
- Then comes mutation, where small random changes are made to the arrangement. For example, maybe an item like 6 gets moved from Bin 2 to Bin B , or items are rotated to fit better.

- These tiny adjustments give the algorithm more variety and help it explore different ways to optimize packing. We want to avoid getting stuck with just one approach, so mutation gives us new options to test.



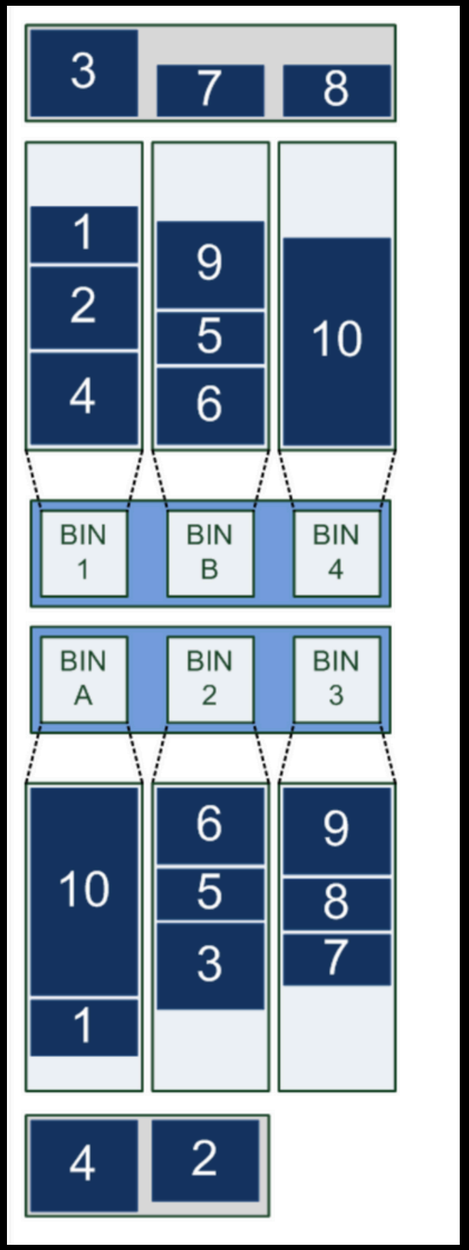
- In this image, we see an important challenge in the crossover process of the genetic algorithm: duplicates. When we copy bins from two different parents, sometimes items get duplicated, meaning they appear in more than one bin.

- To fix this, the algorithm performs a correction step. It identifies the bins where duplicate items are causing the problem.
-

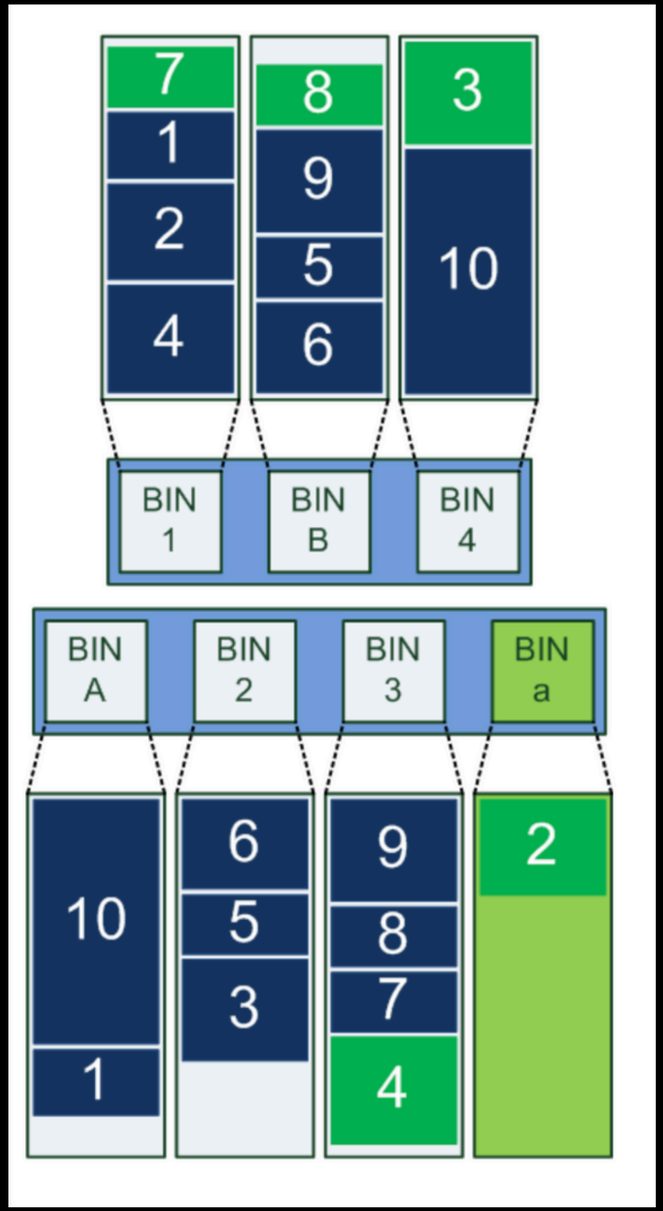


Duplicate Removal Causes Unassigned Items

Reinserting Unassigned Items



To handle unassigned items the algorithm uses first-fit decreasing as a final cleanup step.



Next Generation

In each new generation, the algorithm:

- Keeps the best packing arrangements (those with high fitness scores).
- Combines them with other good solutions (using crossover).
- Mutates parts of the packing to try new ideas and improve further.
- This cycle repeats, and with every new generation, the packing solutions get better and more optimized.