ACO with Dynamic Mutation

In [8]:

```
import numpy as np
import random
import logging
import time

# Logging setup
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(me

# # Problem definition
# stock_lengths = [4300, 4250, 4150, 3950, 3800, 3700, 3550, 3500]
# stock_costs = [86, 85, 83, 79, 68, 66, 64, 63]
# piece_lengths = [2350, 2250, 2200, 2100, 2050, 2000, 1950, 1900, 1850, 1700, 1650
# quantities = [2, 4, 4, 15, 6, 11, 6, 15, 13, 5, 2, 9, 3, 6, 10, 4, 8, 3]

# Problem Definition
stock_lengths = [120, 115, 110, 105, 100]
stock_costs = [12, 11.5, 11, 10.5, 10]
piece_lengths = [21, 22, 24, 25, 27, 29, 30, 31, 32, 33, 34, 35, 38, 39, 42, 44, 45
quantities = [13, 15, 7, 5, 9, 9, 3, 15, 18, 17, 4, 17, 20, 9, 4, 19, 4, 12, 15, 3,


# ACO Parameters
num_ants = 10
num_iterations = 100
alpha = 1.0
beta = 1.0
decay = 0.1
initial_pheromone = 0.1
mutation_base_rate = 0.01

# Initialize pheromones
pheromones = np.full((len(stock_lengths), len(piece_lengths)), initial_pheromone)

def heuristic_value(stock_index, piece_index):
    piece_utilization = piece_lengths[piece_index] / stock_lengths[stock_index]
    return piece_utilization / stock_costs[stock_index]

def mutate(solution, mutation_rate):
    for stock_index, activities in enumerate(solution):
        if random.random() < mutation_rate:
            for activity in activities:
                if activity and random.random() < mutation_rate:
                    piece_to_mutate = random.choice(activity)
                    activity.remove(piece_to_mutate)
                    # Try to insert the mutated piece into a different position
                    possible_positions = range(len(piece_lengths))
                    new_position = random.choice(possible_positions)
                    activity.insert(new_position, piece_to_mutate)
    #logging.debug("Post-mutation solution: {}".format(solution))

def adjust_mutation_rate(previous_cost, current_cost, base_rate):
    if current_cost < previous_cost:  # Improvement found
        return max(base_rate / 2, 0.001)  # Decrease mutation rate
    else:
        return min(base_rate * 2, 0.1)  # Increase mutation rate if stagnated

def solve_aco():
    best_solution = None
```

```
    best_cost = float('inf')
    current_mutation_rate = mutation_base_rate

    for iteration in range(num_iterations):
        solutions = []
        for _ in range(num_ants):
            remaining_quantities = quantities[:]
            solution = construct_solution(pheromones, remaining_quantities)
            cost = calculate_fitness(solution)
            solutions.append((solution, cost))
            if cost < best_cost:
                best_cost = cost
                best_solution = solution
            #logging.debug(f"Iteration {iteration}, Ant {_}, Cost: {cost}, Solution

        # Update pheromones
        update_pheromones(pheromones, solutions, best_cost)

        # Mutation step
        for solution, cost in solutions:
            mutate(solution, current_mutation_rate)

        # Adjust mutation rate based on performance
        current_mutation_rate = adjust_mutation_rate(best_cost, cost, current_mutat
        #logging.debug(f"Current mutation rate: {current_mutation_rate}")

    return best_solution, best_cost

def construct_solution(pheromones, remaining_quantities):
    solution = []
    remaining_quantities = remaining_quantities[:]
    for stock_index in range(len(stock_lengths)):
        activities = []
        while any(remaining_quantities):
            activity = []
            current_length = stock_lengths[stock_index]
            while current_length > 0 and any(remaining_quantities):
                probs = [pheromones[stock_index][j] * alpha * heuristic_value(stock
                         if remaining_quantities[j] > 0 and piece_lengths[j] <= cur
                         for j in range(len(piece_lengths))]
                total_prob = sum(probs)
                if total_prob > 0:
                    probs /= total_prob
                    chosen_piece_index = np.random.choice(len(piece_lengths), p=prc
                    activity.append(chosen_piece_index)
                    remaining_quantities[chosen_piece_index] -= 1
                    current_length -= piece_lengths[chosen_piece_index]
                else:
                    break
            if activity:
                activities.append(activity)
        if activities:
            solution.append((stock_index, activities))
    return solution

def update_pheromones(pheromones, solutions, best_cost):
    for solution, cost in solutions:
        for stock_index, activities in solution:
            for activity in activities:
                for piece_index in activity:
                    pheromones[stock_index][piece_index] += 1 / (cost + 1)
    pheromones *= (1 - decay)

def calculate_fitness(solution):
```

```
    cost = 0
    for stock_index, activities in solution:
        for activity in activities:
            if activity:
                cost += stock_costs[stock_index]
    return cost

def calculate_waste(solution):
    total_waste = 0
    for stock_index, activities in solution:
        for activity in activities:
            used_length = sum(piece_lengths[piece_index] for piece_index in activit
            waste_per_piece = stock_lengths[stock_index] - used_length
            total_waste += waste_per_piece
    return total_waste


def print_solution(solution, cost):
    start_time = time.time()
    best_solution, best_cost = solve_aco()
    end_time = time.time()
    computation_time = end_time - start_time
    print(f"Best Cost: {cost}")
    print("Total waste :", calculate_waste (solution))
    print("Computation time :", computation_time)
    print("Solution:")
    for stock_index, activities in solution:
        for activity in activities:
            pieces = [piece_lengths[piece_index] for piece_index in activity]
            print(f"Stock Type {stock_index} (Length {stock_lengths[stock_index]}):

# Main execution block
if __name__ == "__main__":
    best_solution, best_cost = solve_aco()
    print_solution(best_solution, best_cost)
```

```
Best Cost: 1872
Total waste : 1087
Computation time : 51.43298935890198
Solution:
Stock Type 0 (Length 120): Pieces cut: [63, 38]
Stock Type 0 (Length 120): Pieces cut: [67, 44]
Stock Type 0 (Length 120): Pieces cut: [56, 44]
Stock Type 0 (Length 120): Pieces cut: [44, 22, 35]
Stock Type 0 (Length 120): Pieces cut: [52, 50]
Stock Type 0 (Length 120): Pieces cut: [56, 22, 22]
Stock Type 0 (Length 120): Pieces cut: [50, 38, 31]
Stock Type 0 (Length 120): Pieces cut: [35, 56, 29]
Stock Type 0 (Length 120): Pieces cut: [45, 35, 35]
Stock Type 0 (Length 120): Pieces cut: [63, 46]
Stock Type 0 (Length 120): Pieces cut: [22, 25, 52, 21]
Stock Type 0 (Length 120): Pieces cut: [46, 31, 32]
Stock Type 0 (Length 120): Pieces cut: [57, 61]
Stock Type 0 (Length 120): Pieces cut: [66, 24, 29]
Stock Type 0 (Length 120): Pieces cut: [30, 35, 33, 22]
Stock Type 0 (Length 120): Pieces cut: [56, 57]
Stock Type 0 (Length 120): Pieces cut: [56, 49]
Stock Type 0 (Length 120): Pieces cut: [32, 33, 38]
Stock Type 0 (Length 120): Pieces cut: [44, 54, 22]
Stock Type 0 (Length 120): Pieces cut: [51, 45, 22]
Stock Type 0 (Length 120): Pieces cut: [49, 52]
Stock Type 0 (Length 120): Pieces cut: [66, 53]
Stock Type 0 (Length 120): Pieces cut: [49, 63]
Stock Type 0 (Length 120): Pieces cut: [47, 66]
Stock Type 0 (Length 120): Pieces cut: [51, 33, 34]
Stock Type 0 (Length 120): Pieces cut: [65, 53]
Stock Type 0 (Length 120): Pieces cut: [52, 54]
Stock Type 0 (Length 120): Pieces cut: [38, 51, 22]
Stock Type 0 (Length 120): Pieces cut: [24, 51, 35]
Stock Type 0 (Length 120): Pieces cut: [30, 63, 27]
Stock Type 0 (Length 120): Pieces cut: [51, 67]
Stock Type 0 (Length 120): Pieces cut: [63, 50]
Stock Type 0 (Length 120): Pieces cut: [57, 35, 27]
Stock Type 0 (Length 120): Pieces cut: [53, 52]
Stock Type 0 (Length 120): Pieces cut: [34, 50, 33]
Stock Type 0 (Length 120): Pieces cut: [47, 67]
Stock Type 0 (Length 120): Pieces cut: [56, 52]
Stock Type 0 (Length 120): Pieces cut: [67, 32, 21]
Stock Type 0 (Length 120): Pieces cut: [31, 54, 33]
Stock Type 0 (Length 120): Pieces cut: [35, 57, 21]
Stock Type 0 (Length 120): Pieces cut: [67, 50]
Stock Type 0 (Length 120): Pieces cut: [46, 32, 31]
Stock Type 0 (Length 120): Pieces cut: [49, 39, 25]
Stock Type 0 (Length 120): Pieces cut: [65, 33, 22]
Stock Type 0 (Length 120): Pieces cut: [63, 44]
Stock Type 0 (Length 120): Pieces cut: [60, 39, 21]
Stock Type 0 (Length 120): Pieces cut: [59, 57]
Stock Type 0 (Length 120): Pieces cut: [59, 35, 22]
Stock Type 0 (Length 120): Pieces cut: [51, 63]
Stock Type 0 (Length 120): Pieces cut: [38, 67]
Stock Type 0 (Length 120): Pieces cut: [32, 56, 22]
Stock Type 0 (Length 120): Pieces cut: [38, 51, 24]
Stock Type 0 (Length 120): Pieces cut: [63, 42]
Stock Type 0 (Length 120): Pieces cut: [61, 55]
Stock Type 0 (Length 120): Pieces cut: [46, 44, 25]
Stock Type 0 (Length 120): Pieces cut: [67, 47]
Stock Type 0 (Length 120): Pieces cut: [49, 57]
Stock Type 0 (Length 120): Pieces cut: [56, 63]
Stock Type 0 (Length 120): Pieces cut: [61, 59]
Stock Type 0 (Length 120): Pieces cut: [61, 51]
```

```
Stock Type 0 (Length 120): Pieces cut: [67, 51]
Stock Type 0 (Length 120): Pieces cut: [57, 63]
Stock Type 0 (Length 120): Pieces cut: [24, 67, 22]
Stock Type 0 (Length 120): Pieces cut: [63, 51]
Stock Type 0 (Length 120): Pieces cut: [49, 39, 32]
Stock Type 0 (Length 120): Pieces cut: [54, 38, 24]
Stock Type 0 (Length 120): Pieces cut: [59, 54]
Stock Type 0 (Length 120): Pieces cut: [63, 50]
Stock Type 0 (Length 120): Pieces cut: [31, 50, 33]
Stock Type 0 (Length 120): Pieces cut: [47, 61]
Stock Type 0 (Length 120): Pieces cut: [61, 57]
Stock Type 0 (Length 120): Pieces cut: [67, 51]
Stock Type 0 (Length 120): Pieces cut: [46, 67]
Stock Type 0 (Length 120): Pieces cut: [44, 66]
Stock Type 0 (Length 120): Pieces cut: [65, 49]
Stock Type 0 (Length 120): Pieces cut: [45, 57]
Stock Type 0 (Length 120): Pieces cut: [54, 22, 44]
Stock Type 0 (Length 120): Pieces cut: [45, 49, 24]
Stock Type 0 (Length 120): Pieces cut: [47, 65]
Stock Type 0 (Length 120): Pieces cut: [44, 66]
Stock Type 0 (Length 120): Pieces cut: [57, 38, 24]
Stock Type 0 (Length 120): Pieces cut: [27, 56, 35]
Stock Type 0 (Length 120): Pieces cut: [29, 67, 21]
Stock Type 0 (Length 120): Pieces cut: [67, 49]
Stock Type 0 (Length 120): Pieces cut: [63, 32, 22]
Stock Type 0 (Length 120): Pieces cut: [50, 38, 29]
Stock Type 0 (Length 120): Pieces cut: [42, 56, 22]
Stock Type 0 (Length 120): Pieces cut: [66, 49]
Stock Type 0 (Length 120): Pieces cut: [63, 32, 21]
Stock Type 0 (Length 120): Pieces cut: [51, 31, 27]
Stock Type 0 (Length 120): Pieces cut: [32, 44, 44]
Stock Type 0 (Length 120): Pieces cut: [33, 31, 44]
Stock Type 0 (Length 120): Pieces cut: [65, 33, 21]
Stock Type 0 (Length 120): Pieces cut: [44, 27, 49]
Stock Type 0 (Length 120): Pieces cut: [49, 57]
Stock Type 0 (Length 120): Pieces cut: [66, 51]
Stock Type 0 (Length 120): Pieces cut: [35, 35, 48]
Stock Type 0 (Length 120): Pieces cut: [54, 63]
Stock Type 0 (Length 120): Pieces cut: [35, 46, 31]
Stock Type 0 (Length 120): Pieces cut: [47, 49, 21]
Stock Type 0 (Length 120): Pieces cut: [63, 51]
Stock Type 0 (Length 120): Pieces cut: [47, 66]
Stock Type 0 (Length 120): Pieces cut: [51, 59]
Stock Type 0 (Length 120): Pieces cut: [32, 47, 38]
Stock Type 0 (Length 120): Pieces cut: [57, 46]
Stock Type 0 (Length 120): Pieces cut: [55, 63]
Stock Type 0 (Length 120): Pieces cut: [47, 47, 25]
Stock Type 0 (Length 120): Pieces cut: [38, 31, 42]
Stock Type 0 (Length 120): Pieces cut: [42, 53, 21]
Stock Type 0 (Length 120): Pieces cut: [67, 27, 25]
Stock Type 0 (Length 120): Pieces cut: [47, 33, 29]
Stock Type 0 (Length 120): Pieces cut: [44, 29, 33]
Stock Type 0 (Length 120): Pieces cut: [35, 32, 49]
Stock Type 0 (Length 120): Pieces cut: [34, 67]
Stock Type 0 (Length 120): Pieces cut: [66, 46]
Stock Type 0 (Length 120): Pieces cut: [67, 38]
Stock Type 0 (Length 120): Pieces cut: [57, 56]
Stock Type 0 (Length 120): Pieces cut: [38, 66]
Stock Type 0 (Length 120): Pieces cut: [57, 49]
Stock Type 0 (Length 120): Pieces cut: [35, 49, 29]
Stock Type 0 (Length 120): Pieces cut: [35, 46, 38]
Stock Type 0 (Length 120): Pieces cut: [56, 32, 27]
Stock Type 0 (Length 120): Pieces cut: [56, 34, 27]
Stock Type 0 (Length 120): Pieces cut: [38, 50, 32]
```

```
Stock Type 0 (Length 120): Pieces cut: [67, 50]
Stock Type 0 (Length 120): Pieces cut: [31, 33, 47]
Stock Type 0 (Length 120): Pieces cut: [56, 63]
Stock Type 0 (Length 120): Pieces cut: [32, 56, 31]
Stock Type 0 (Length 120): Pieces cut: [38, 32, 39]
Stock Type 0 (Length 120): Pieces cut: [56, 50]
Stock Type 0 (Length 120): Pieces cut: [56, 63]
Stock Type 0 (Length 120): Pieces cut: [32, 27, 30, 31]
Stock Type 0 (Length 120): Pieces cut: [49, 29, 31]
Stock Type 0 (Length 120): Pieces cut: [39, 47, 31]
Stock Type 0 (Length 120): Pieces cut: [44, 29, 47]
Stock Type 0 (Length 120): Pieces cut: [50, 47, 21]
Stock Type 0 (Length 120): Pieces cut: [33, 32, 38]
Stock Type 0 (Length 120): Pieces cut: [46, 63]
Stock Type 0 (Length 120): Pieces cut: [56, 57]
Stock Type 0 (Length 120): Pieces cut: [31, 50, 39]
Stock Type 0 (Length 120): Pieces cut: [57, 35, 21]
Stock Type 0 (Length 120): Pieces cut: [44, 44, 21]
Stock Type 0 (Length 120): Pieces cut: [56, 50]
Stock Type 0 (Length 120): Pieces cut: [49, 57]
Stock Type 0 (Length 120): Pieces cut: [49, 61]
Stock Type 0 (Length 120): Pieces cut: [49, 33, 33]
Stock Type 0 (Length 120): Pieces cut: [44, 57]
Stock Type 0 (Length 120): Pieces cut: [44, 38, 32]
Stock Type 0 (Length 120): Pieces cut: [46, 57]
Stock Type 0 (Length 120): Pieces cut: [59, 55]
Stock Type 0 (Length 120): Pieces cut: [38, 38, 39]
Stock Type 0 (Length 120): Pieces cut: [55, 33, 21]
Stock Type 0 (Length 120): Pieces cut: [39, 39, 33]
Stock Type 0 (Length 120): Pieces cut: [55, 46]
Stock Type 0 (Length 120): Pieces cut: [48, 60]
Stock Type 0 (Length 120): Pieces cut: [60, 48]
```

Hybrid ACO with Local Search

In [6]:

```python
import numpy as np
import random
import logging
import time



# # Problem Definition
# stock_lengths = [4300, 4250, 4150, 3950, 3800, 3700, 3550, 3500]
# stock_costs = [86, 85, 83, 79, 68, 66, 64, 63]
# piece_lengths = [2350, 2250, 2200, 2100, 2050, 2000, 1950, 1900, 1850, 1700, 1650
# quantities = [2, 4, 4, 15, 6, 11, 6, 15, 13, 5, 2, 9, 3, 6, 10, 4, 8, 3]

stock_lengths = [120, 115, 110, 105, 100]
stock_costs = [12, 11.5, 11, 10.5, 10]
piece_lengths = [21, 22, 24, 25, 27, 29, 30, 31, 32, 33, 34, 35, 38, 39, 42, 44, 45
quantities = [13, 15, 7, 5, 9, 9, 3, 15, 18, 17, 4, 17, 20, 9, 4, 19, 4, 12, 15, 3,


logging.basicConfig(level=logging.DEBUG, format='%(asctime)s - %(levelname)s - %(me



# Parameters
num_ants = 10
num_iterations = 10
alpha = 1.0  # Pheromone influence
beta = 1.0   # Heuristic influence
```

```
decay = 0.1  # Pheromone decay rate
initial_pheromone = 0.1

# Initialize pheromones
pheromones = np.full((len(stock_lengths), len(piece_lengths)), initial_pheromone)

# Update heuristic to consider current stock usage
def heuristic_value(stock_index, piece_index):
    noise = random.uniform(0.9, 1.1)
    piece_utilization = piece_lengths[piece_index] / stock_lengths[stock_index] * r
    return piece_utilization / stock_costs[stock_index]

def apply_local_search(solution, remaining_quantities):
    improved = True
    while improved:
        improved = False
        for stock_index, activities in solution:
            for pieces in activities:  # Now correctly handling 'pieces' as each ir
                if not pieces:
                    continue
                gaps = stock_lengths[stock_index] - sum(piece_lengths[p] for p in p
                for j in range(len(piece_lengths)):
                    if piece_lengths[j] <= gaps and remaining_quantities[j] > 0:
                        pieces.append(j)
                        remaining_quantities[j] -= 1
                        gaps -= piece_lengths[j]
                        improved = True

                # Try rearranging pieces within and between stocks to minimize the
                for other_index, other_activities in enumerate(solution):
                    if stock_index == other_index:
                        continue
                    for other_pieces in other_activities:
                        other_gaps = stock_lengths[other_index] - sum(piece_lengths
                        for piece in list(pieces):
                            if piece_lengths[piece] <= other_gaps:
                                # Move piece to another stock
                                other_pieces.append(piece)
                                pieces.remove(piece)
                                other_gaps -= piece_lengths[piece]
                                gaps += piece_lengths[piece]
                                improved = True
                                break  # Reevaluate after each move
    return solution



def update_pheromones(pheromones, solutions, best_cost):
    for solution, cost in solutions:
        for stock_index, activities in solution:
            for activity in activities:
                for piece_index in activity:
                    # Update pheromones for each piece used in this activity
                    pheromones[stock_index][piece_index] += 1 / (cost + 1)
    pheromones *= (1 - decay)


def construct_solution(pheromones, remaining_quantities):
    solution = []
    remaining_quantities = remaining_quantities[:]
    for stock_index in range(len(stock_lengths)):
        activities = []  # List of activities for this stock type
        while any(remaining_quantities):
            activity = []
```

```
            current_length = stock_lengths[stock_index]
            while current_length > 0 and any(remaining_quantities):
                probs = [pheromones[stock_index][j] * alpha * heuristic_value(stock
                         if remaining_quantities[j] > 0 and piece_lengths[j] <= cur
                         for j in range(len(piece_lengths))]
                total_prob = sum(probs)
                if total_prob > 0:
                    probs /= total_prob
                    chosen_piece_index = np.random.choice(len(piece_lengths), p=prc
                    activity.append(chosen_piece_index)
                    remaining_quantities[chosen_piece_index] -= 1
                    current_length -= piece_lengths[chosen_piece_index]
                else:
                    break
            if activity:
                activities.append(activity)
        if activities:
            solution.append((stock_index, activities))
    return solution


def calculate_fitness(solution):
    cost = 0
    for stock_index, activities in solution:
        for activity in activities:
            if activity:  # Check if this activity is non-empty
                cost += stock_costs[stock_index]  # Each activity uses a new stock
    return cost


def calculate_waste(solution):
    total_waste = 0
    for stock_index, activities in solution:
        for activity in activities:
            used_length = sum(piece_lengths[piece_index] for piece_index in activit
            waste_per_piece = stock_lengths[stock_index] - used_length
            total_waste += waste_per_piece
    return total_waste

def print_solution(solution, cost):
    start_time = time.time()
    best_solution, best_cost = solve()
    end_time = time.time()
    computation_time = end_time - start_time
    total_waste = calculate_waste(solution)
    print("Total Waste:", total_waste)
    print("Computation time :", computation_time)
    print(f"Best Cost: {cost}")
    print("Solution:")
    for stock_index, activities in solution:
        for activity in activities:
            pieces = [piece_lengths[piece_index] for piece_index in activity]
            print(f"Stock Type {stock_index} (Length {stock_lengths[stock_index]}):


def solve():
    best_solution = None
    best_cost = float('inf')
    for _ in range(num_iterations):
        solutions = []
        for _ in range(num_ants):
            remaining_quantities = quantities[:]
            #logging.debug(f"Starting quantities for ant: {remaining_quantities}")
            solution = construct_solution(pheromones, remaining_quantities)
```

```
            #logging.debug(f"Solution before local search: {solution}")
            solution = apply_local_search(solution, remaining_quantities)
            #logging.debug(f"Solution after local search: {solution}")
            cost = calculate_fitness(solution)
            #logging.debug(f"Cost of solution: {cost}")
            solutions.append((solution, cost))
            if cost < best_cost:
                best_cost = cost
                best_solution = solution
        update_pheromones(pheromones, solutions, best_cost)
    return best_solution, best_cost

if __name__ == "__main__":
    best_solution, best_cost = solve()
    print_solution(best_solution, best_cost)
```

```
Total Waste: 1102
Computation time : 5.8081748485565186
Best Cost: 1884
Solution:
Stock Type 0 (Length 120): Pieces cut: [47, 61]
Stock Type 0 (Length 120): Pieces cut: [27, 29, 22, 35]
Stock Type 0 (Length 120): Pieces cut: [22, 46, 47]
Stock Type 0 (Length 120): Pieces cut: [32, 48, 21]
Stock Type 0 (Length 120): Pieces cut: [57, 59]
Stock Type 0 (Length 120): Pieces cut: [53, 63]
Stock Type 0 (Length 120): Pieces cut: [56, 51]
Stock Type 0 (Length 120): Pieces cut: [35, 57, 24]
Stock Type 0 (Length 120): Pieces cut: [33, 60, 22]
Stock Type 0 (Length 120): Pieces cut: [49, 65]
Stock Type 0 (Length 120): Pieces cut: [63, 49]
Stock Type 0 (Length 120): Pieces cut: [50, 59]
Stock Type 0 (Length 120): Pieces cut: [63, 27, 24]
Stock Type 0 (Length 120): Pieces cut: [35, 65]
Stock Type 0 (Length 120): Pieces cut: [51, 51]
Stock Type 0 (Length 120): Pieces cut: [22, 66, 31]
Stock Type 0 (Length 120): Pieces cut: [63, 49]
Stock Type 0 (Length 120): Pieces cut: [44, 66]
Stock Type 0 (Length 120): Pieces cut: [47, 34, 38]
Stock Type 0 (Length 120): Pieces cut: [63, 49]
Stock Type 0 (Length 120): Pieces cut: [32, 66, 22]
Stock Type 0 (Length 120): Pieces cut: [49, 21, 44]
Stock Type 0 (Length 120): Pieces cut: [21, 33, 47]
Stock Type 0 (Length 120): Pieces cut: [49, 24, 31]
Stock Type 0 (Length 120): Pieces cut: [34, 49, 34]
Stock Type 0 (Length 120): Pieces cut: [29, 53, 33]
Stock Type 0 (Length 120): Pieces cut: [47, 57]
Stock Type 0 (Length 120): Pieces cut: [44, 44, 31]
Stock Type 0 (Length 120): Pieces cut: [22, 38, 57]
Stock Type 0 (Length 120): Pieces cut: [38, 49, 24]
Stock Type 0 (Length 120): Pieces cut: [44, 32, 38]
Stock Type 0 (Length 120): Pieces cut: [50, 67]
Stock Type 0 (Length 120): Pieces cut: [44, 66]
Stock Type 0 (Length 120): Pieces cut: [59, 38, 21]
Stock Type 0 (Length 120): Pieces cut: [55, 27, 32]
Stock Type 0 (Length 120): Pieces cut: [61, 54]
Stock Type 0 (Length 120): Pieces cut: [65, 30, 22]
Stock Type 0 (Length 120): Pieces cut: [66, 49]
Stock Type 0 (Length 120): Pieces cut: [67, 47]
Stock Type 0 (Length 120): Pieces cut: [67, 29, 21]
Stock Type 0 (Length 120): Pieces cut: [55, 63]
Stock Type 0 (Length 120): Pieces cut: [21, 31, 52]
Stock Type 0 (Length 120): Pieces cut: [46, 51, 21]
Stock Type 0 (Length 120): Pieces cut: [48, 33, 39]
Stock Type 0 (Length 120): Pieces cut: [35, 48, 33]
Stock Type 0 (Length 120): Pieces cut: [67, 49]
Stock Type 0 (Length 120): Pieces cut: [67, 44]
Stock Type 0 (Length 120): Pieces cut: [59, 57]
Stock Type 0 (Length 120): Pieces cut: [32, 60, 21]
Stock Type 0 (Length 120): Pieces cut: [47, 49, 24]
Stock Type 0 (Length 120): Pieces cut: [32, 51, 33]
Stock Type 0 (Length 120): Pieces cut: [21, 52, 42]
Stock Type 0 (Length 120): Pieces cut: [61, 25, 29]
Stock Type 0 (Length 120): Pieces cut: [63, 50]
Stock Type 0 (Length 120): Pieces cut: [50, 59]
Stock Type 0 (Length 120): Pieces cut: [54, 46]
Stock Type 0 (Length 120): Pieces cut: [39, 39, 38]
Stock Type 0 (Length 120): Pieces cut: [63, 55]
Stock Type 0 (Length 120): Pieces cut: [38, 51, 29]
Stock Type 0 (Length 120): Pieces cut: [46, 60]
```

```
Stock Type 0 (Length 120): Pieces cut: [45, 67]
Stock Type 0 (Length 120): Pieces cut: [53, 35, 32]
Stock Type 0 (Length 120): Pieces cut: [54, 63]
Stock Type 0 (Length 120): Pieces cut: [46, 47, 22]
Stock Type 0 (Length 120): Pieces cut: [65, 54]
Stock Type 0 (Length 120): Pieces cut: [51, 54]
Stock Type 0 (Length 120): Pieces cut: [49, 47, 21]
Stock Type 0 (Length 120): Pieces cut: [31, 67, 21]
Stock Type 0 (Length 120): Pieces cut: [63, 54]
Stock Type 0 (Length 120): Pieces cut: [67, 44]
Stock Type 0 (Length 120): Pieces cut: [38, 49, 29]
Stock Type 0 (Length 120): Pieces cut: [53, 57]
Stock Type 0 (Length 120): Pieces cut: [45, 44, 21]
Stock Type 0 (Length 120): Pieces cut: [49, 31, 22]
Stock Type 0 (Length 120): Pieces cut: [46, 66]
Stock Type 0 (Length 120): Pieces cut: [29, 55, 32]
Stock Type 0 (Length 120): Pieces cut: [55, 33, 29]
Stock Type 0 (Length 120): Pieces cut: [57, 63]
Stock Type 0 (Length 120): Pieces cut: [46, 52, 21]
Stock Type 0 (Length 120): Pieces cut: [33, 65, 22]
Stock Type 0 (Length 120): Pieces cut: [63, 50]
Stock Type 0 (Length 120): Pieces cut: [39, 50, 29]
Stock Type 0 (Length 120): Pieces cut: [33, 22, 57]
Stock Type 0 (Length 120): Pieces cut: [67, 42]
Stock Type 0 (Length 120): Pieces cut: [47, 42, 24]
Stock Type 0 (Length 120): Pieces cut: [67, 51]
Stock Type 0 (Length 120): Pieces cut: [45, 50, 24]
Stock Type 0 (Length 120): Pieces cut: [49, 47, 22]
Stock Type 0 (Length 120): Pieces cut: [33, 51, 31]
Stock Type 0 (Length 120): Pieces cut: [35, 31, 49]
Stock Type 0 (Length 120): Pieces cut: [33, 63, 22]
Stock Type 0 (Length 120): Pieces cut: [57, 63]
Stock Type 0 (Length 120): Pieces cut: [63, 35, 22]
Stock Type 0 (Length 120): Pieces cut: [66, 50]
Stock Type 0 (Length 120): Pieces cut: [63, 52]
Stock Type 0 (Length 120): Pieces cut: [50, 31, 35]
Stock Type 0 (Length 120): Pieces cut: [32, 51, 35]
Stock Type 0 (Length 120): Pieces cut: [61, 32, 27]
Stock Type 0 (Length 120): Pieces cut: [45, 50, 22]
Stock Type 0 (Length 120): Pieces cut: [56, 49]
Stock Type 0 (Length 120): Pieces cut: [56, 49]
Stock Type 0 (Length 120): Pieces cut: [67, 47]
Stock Type 0 (Length 120): Pieces cut: [63, 47]
Stock Type 0 (Length 120): Pieces cut: [49, 25, 38]
Stock Type 0 (Length 120): Pieces cut: [57, 50]
Stock Type 0 (Length 120): Pieces cut: [39, 38, 32]
Stock Type 0 (Length 120): Pieces cut: [63, 51]
Stock Type 0 (Length 120): Pieces cut: [47, 61]
Stock Type 0 (Length 120): Pieces cut: [44, 56]
Stock Type 0 (Length 120): Pieces cut: [56, 44]
Stock Type 0 (Length 120): Pieces cut: [49, 52]
Stock Type 0 (Length 120): Pieces cut: [33, 46, 38]
Stock Type 0 (Length 120): Pieces cut: [63, 35, 21]
Stock Type 0 (Length 120): Pieces cut: [31, 38, 46]
Stock Type 0 (Length 120): Pieces cut: [57, 44]
Stock Type 0 (Length 120): Pieces cut: [44, 42, 25]
Stock Type 0 (Length 120): Pieces cut: [63, 32, 25]
Stock Type 0 (Length 120): Pieces cut: [33, 27, 56]
Stock Type 0 (Length 120): Pieces cut: [31, 61, 27]
Stock Type 0 (Length 120): Pieces cut: [56, 39, 25]
Stock Type 0 (Length 120): Pieces cut: [56, 59]
Stock Type 0 (Length 120): Pieces cut: [39, 33, 38]
Stock Type 0 (Length 120): Pieces cut: [66, 38]
Stock Type 0 (Length 120): Pieces cut: [46, 35, 33]
```

```
Stock Type 0 (Length 120): Pieces cut: [54, 50]
Stock Type 0 (Length 120): Pieces cut: [66, 46]
Stock Type 0 (Length 120): Pieces cut: [38, 33, 33]
Stock Type 0 (Length 120): Pieces cut: [66, 47]
Stock Type 0 (Length 120): Pieces cut: [38, 57, 21]
Stock Type 0 (Length 120): Pieces cut: [56, 57]
Stock Type 0 (Length 120): Pieces cut: [67, 38]
Stock Type 0 (Length 120): Pieces cut: [44, 57]
Stock Type 0 (Length 120): Pieces cut: [67, 38]
Stock Type 0 (Length 120): Pieces cut: [46, 56]
Stock Type 0 (Length 120): Pieces cut: [32, 51, 31]
Stock Type 0 (Length 120): Pieces cut: [57, 38, 21]
Stock Type 0 (Length 120): Pieces cut: [31, 57, 32]
Stock Type 0 (Length 120): Pieces cut: [56, 57]
Stock Type 0 (Length 120): Pieces cut: [39, 44, 35]
Stock Type 0 (Length 120): Pieces cut: [44, 38, 31]
Stock Type 0 (Length 120): Pieces cut: [27, 56, 35]
Stock Type 0 (Length 120): Pieces cut: [51, 27, 35]
Stock Type 0 (Length 120): Pieces cut: [31, 30, 35, 21]
Stock Type 0 (Length 120): Pieces cut: [56, 56]
Stock Type 0 (Length 120): Pieces cut: [56, 51]
Stock Type 0 (Length 120): Pieces cut: [35, 39, 44]
Stock Type 0 (Length 120): Pieces cut: [67, 52]
Stock Type 0 (Length 120): Pieces cut: [56, 57]
Stock Type 0 (Length 120): Pieces cut: [44, 44, 32]
Stock Type 0 (Length 120): Pieces cut: [67, 35]
Stock Type 0 (Length 120): Pieces cut: [34, 67]
Stock Type 0 (Length 120): Pieces cut: [57, 56]
Stock Type 0 (Length 120): Pieces cut: [56, 56]
Stock Type 0 (Length 120): Pieces cut: [67, 50]
Stock Type 0 (Length 120): Pieces cut: [51, 32, 32]
Stock Type 0 (Length 120): Pieces cut: [61, 32, 27]
Stock Type 0 (Length 120): Pieces cut: [50, 30, 21]
```

In [ ]: