Tp Final Otimização de grande porte two-dimensional non-guillotine cutting problems

Tasso Augusto Tomaz Pimenta 2021072198

Definição do problema via artigo

"Given a set of n demanded items with width wi and height hi, i = 1, ..., n,
and a set of m available objects with width Wj , height Hj , and cost cj
per unit of area, j = 1, ..., m, the nonguillotine cutting problem (without
residual pieces) is defined as the one of cutting the demanded items from the
available objects minimizing the cost of the used objects.
No rotations are allowed and there are no other constraints related to the
positioning of the items within the objects, or the types of cuts of the objects
(e.g. guillotine or staged cuts). We assume that the cuts of the objects are
infinitely thin (otherwise we consider that the saw thickness was added to the
dimensions of the objects and items). We also assume that the items' and objects'
dimensions are positive integers and the objects' costs per unit of area are
non-negative integers. These are not very restrictive hypotheses to deal with
real instances since, due to the finite precision of the cutting and measuring
tools and due to the finite precision used in any currency considered to define
the objects' costs, they can be easily satisfied by a change of scale.

Summing up, the MIP model of the tackled non-guillotine cutting problem is given by
minimizing (3) on:
uj {0, 1} (j = 1, ..., m), vij {0, 1} (i = 1, ..., n, j = 1, ..., m),
xi, yi R (i = 1, ..., n), ii´ {0, 1} (i = 1, ..., n, i = i + 1, ..., n),
and ii {0, 1}(q = 1, ..., p, i = oq + 1, ..., oq + nq, i = oq + nq + 1, ..., n)
subject to (1,2,4,5,6).

There are m+mn+n(n−1)− Pp q=1 nq(nq −1)/2 binary variables (which coincide with m+mn+n(n−1)
in the case in which there are no identical items), 2n continuous variables,
and 3mn + 3n + 2m Pp q=1 nq(nq − 1)/2 + 4m[n(n − 1)/2 −

```
Pp q=1 nq(nq - 1)/2] constraints (which coincide with3mn + 3n + n(n - 1)
in the case in which there are no identical items)."
```

{Andrade et al. (2014)}

```
from mip import Model, xsum, minimize, BINARY, CONTINUOUS, MINIMIZE, INTEGER
from time import time
import numpy as np
from data_tp_final import return_datas
```

Problema descrito no artigo principal

```
class AssortmentSolver:
    def __init__(self, data):
        self.data = data
        self.model = Model()
        self.solution = None
        self.setup_model()
    def setup_model(self):
        model   = self.model
        model.verbose = 0
        Objects = self.data['Objects']
        Items   = self.data['Items']
        n = len(Items)
        m = len(Objects)
        I, J =range(n), range(m)
        # Variables
        model.u   = u   = [model.add_var(var_type=BINARY) for j in J]
        model.v   = v   = [[model.add_var(var_type=BINARY) for j in J] for i in I]
        model.x   = x   = [model.add_var(var_type=CONTINUOUS) for i in I]
        model.y   = y   = [model.add_var(var_type=CONTINUOUS) for i in I]
        model.pi  = pi  = [[model.add_var(var_type=BINARY) for i_ in I] for i in I]
        model.tau = tau = [[model.add_var(var_type=BINARY) for i_ in I] for i in I]

        H = lambda j: Objects[j]['Height']
        max_H = max(H(j) for j in J)
        W = lambda j: Objects[j]['Length']
        max_W = max(W(j) for j in J)
        w = lambda i: Items[i]['Length']
        h = lambda i: Items[i]['Height']
```

```
        # Objective (3)
        model.objective = minimize(xsum(Objects[j]['Cost'] * W(j) * H(j) * u[j] for j in J))#

        # Constraints (1)
        for i in I:
            for j in J:
                model += (u[j] >= v[i][j])
        # Constraints (2)
        for i in I:
            model += xsum(v[i][j] for j in J) == 1
        # Constraints (4)
        for i in I:
            model += (x[i]-w(i))/2 >= 0
            model += (y[i]-w(i))/2 >= 0
            for j in J:
                model += (x[i]+w(i))/2 <= W(j) + (max_W- W(j))*(1-v[i][j])
                model += (y[i]+h(i))/2 <= H(j) + (max_H- H(j))*(1-v[i][j])
        for i in I:#complexidade alta
            for i_ in range(i + 1, len(Items)):
                for j in J:
        # Constraints (5)
                    model+=(x[i_]-w(i_))/2>=(x[i]+w(i))/2-max_W*(2-v[i][j]-v[i_][j]+pi[i][i_]
                    model+=(y[i_]-h(i_))/2>=(y[i]+h(i))/2-max_H*(3-v[i][j]-v[i_][j]-pi[i][i_]
        # Constraints (6)
                    model+=(x[i_]+w(i_))/2<=(x[i]-w(i))/2+max_W*(2-v[i][j]-v[i_][j]+pi[i][i_]
                    model+=(x[i_]-w(i_))/2>=(x[i]+w(i))/2-max_W*(3-v[i][j]-v[i_][j]+pi[i][i_]
                    model+=(y[i_]+h(i_))/2<=(y[i]-h(i))/2+max_H*(3-v[i][j]-v[i_][j]-pi[i][i_]
                    model+=(y[i_]-h(i_))/2>=(y[i]+h(i))/2-max_H*(4-v[i][j]-v[i_][j]-pi[i][i_]
    def solve(self):
        Objects = self.data['Objects']
        Items   = self.data['Items']
        model = self.model
        n = len(Items)
        m = len(Objects)
        I, J =range(n), range(m)
        inicio = time()
        self.model.optimize()
        self.time = time() - inicio
        if self.model.num_solutions:
            self.ub = model.objective_value
            self.solution = [(i, j) for i in I for j in J if model.v[i][j].x >= 0.99]
```

```
    def get_solution(self):
        return self.solution
    def get_ub(self):
        return self.ub
    def print_solution(self):
        if self.solution is not None:
            print('Solução encontrada em {:.2f} segundos'.format(self.time))
            print(f'Quantidade total de objetos é {len(self.data["Objects"])}')
            print(f'Quantidade total de Itemns é {len(self.data["Items"])}')
            chosen_item = set()
            chosen_object = set()
            for item, obj in self.solution:
                chosen_item.add(item)
                chosen_object.add(obj)
                x_pos = self.model.x[item].x
                y_pos = self.model.y[item].x
                item_length = self.data['Items'][item]['Length']
                item_height = self.data['Items'][item]['Height']
                obj_length = self.data['Objects'][obj]['Length']
                obj_height = self.data['Objects'][obj]['Height']
                print(f'Item {item+1} é colocado no objeto {obj+1}')
                print(f'  - Posição: ({x_pos:.2f}, {y_pos:.2f})')
                print(f'  - Tamanho do item: {item_length} x {item_height}')
                print(f'  - Tamanho do objeto: {obj_length} x {obj_height}')
                print()
            # List objects that were not chosen
            not_chosen_item = set(range(len(self.data['Items']))) - chosen_item
            not_chosen_obj = set(range(len(self.data['Objects']))) - chosen_object
            if not_chosen_item:
                print('Itens não escolhidos:')
                for item in not_chosen_item:
                    print(f'  - Itens {item+1} - tamanho {self.data["Items"][item]["Length"]]
            else:
                print('Todos os Itens foram escolhidos.')
            if not_chosen_obj:
                print('Objetos não escolhidos:')
                for obj in not_chosen_obj:
                    print(f'  - Objetos {obj+1} - tamanho {self.data["Objects"][obj]["Length'
            else:
                print('Todos os Objetos foram escolhidos.')
        else:
            print('Nenhuma solução viável encontrada.')
```

Data

A data foi retirada do Git: Oliveira, Gamboa, and Silva (n.d.)

Link para o dataset No seguinte formato

```
'''
datas = return_datas()
Objects    = [datas['Objects']]
J = range(Objects)
    Stock   =Objects[j for J]['Stock  ']
    Cost    =Objects[j for J]['Cost   ']
    Length  =Objects[j for J]['Length ']
    Height  =Objects[j for J]['Height ']

Items      = [datas['Items']]
I = range(Items)
    Demand   = Items[i for I]['Demand   ']
    DemandMax= Items[i for I]['DemandMax']
    Value    = Items[i for I]['Value    ']
    Length   = Items[i for I]['Length   ']
    Height   = Items[i for I]['Height   ']
'''
```

```
"\ndatas = return_datas()\nObjects    = [datas['Objects']]\nJ = range(Objects)\n    Stock   =
```

Usando relaxação lagrangeana para resolver o problema

```
def AssortmentSolverLagrangean(data, ub):
    Objects = data['Objects']
    Items   = data['Items']
    m = len(Objects)
    n = len(Items)
    I,J = range(n),range(m)
    H = lambda j: Objects[j]['Height']
    W = lambda j: Objects[j]['Length']
    w = lambda i: Items[i]['Length']
    h = lambda i: Items[i]['Height']
    c = lambda j: Objects[j]['Cost']
    # Initialize variables
```

```
    u = np.zeros(m)
    v = np.zeros((n, m))
    x = np.zeros(n)
    y = np.zeros(n)
    lambda_ij = np.zeros((n, m))
    mu = 1.0  # Initial step size
    lb = -float('inf')
    gap = float('inf')
    h_iter = 0
    def print_solution_lagrangian(solution):
        if solution is not None:
            print('Solução encontrada')
            print(f'Quantidade total de objetos é {len(data["Objects"])}')
            print(f'Quantidade total de Itens é {len(data["Items"])}')
            chosen_item = set()
            chosen_object = set()
            for item, obj in solution:
                chosen_item.add(item)
                chosen_object.add(obj)
                item_length = data['Items'][item]['Length']
                item_height = data['Items'][item]['Height']
                obj_length = data['Objects'][obj]['Length']
                obj_height = data['Objects'][obj]['Height']
                print(f'Item {item+1} é colocado no objeto {obj+1}')
                print(f'  - Tamanho do item: {item_length} x {item_height}')
                print(f'  - Tamanho do objeto: {obj_length} x {obj_height}')
                print()
            # List objects that were not chosen
            not_chosen_item = set(range(len(data['Items']))) - chosen_item
            not_chosen_obj = set(range(len(data['Objects']))) - chosen_object
            if not_chosen_item:
                print('Itens não escolhidos:')
                for item in not_chosen_item:
                    print(f'  - Itens {item+1} - tamanho {data["Items"][item]["Length"]} x {
            else:
                print('Todos os Itens foram escolhidos.')
            if not_chosen_obj:
                print('Objetos não escolhidos:')
                for obj in not_chosen_obj:
                    print(f'  - Objetos {obj+1} - tamanho {data["Objects"][obj]["Length"]} x
            else:
                print('Todos os Objetos foram escolhidos.')
```

```
        else:
            print('Nenhuma solução viável encontrada.')
    def solve_subproblem_u(lambda_ij):
        model = Model(sense=MINIMIZE)
        model.verbose = 0
        u_vars = [model.add_var(var_type=BINARY) for j in J]
        model.objective = xsum(c(j) * W(j) * H(j) * u_vars[j] - xsum(lambda_ij[i][j] * u_vars
        model.optimize()
        return model, u_vars
    def solve_subproblem_v(lambda_ij):
        model = Model(sense=MINIMIZE)
        model.verbose = 0
        v_vars = [[model.add_var(var_type=BINARY) for j in J] for i in I]
        for i in I:
            model += xsum(v_vars[i][j] for j in J) == 1
        model.objective = xsum(lambda_ij[i][j] * v_vars[i][j] for i in I for j in J)
        model.optimize()
        return model, v_vars
    while h_iter < 100 and gap > 1e-4:
        h_iter += 1
        subgradient = np.zeros((n, m))
        infimum = np.sum(lambda_ij)
        model_u, u_vars = solve_subproblem_u(lambda_ij)
        model_v, v_vars = solve_subproblem_v(lambda_ij)
        infimum += model_u.objective_value + model_v.objective_value
        for i in I:
            for j in J:
                subgradient[i][j] = v_vars[i][j].x - u_vars[j].x
        if infimum > lb:
            lb = infimum
        else:
            mu /= 2.0
        norm = np.sum(subgradient**2)
        gap = 100.0 * (ub - lb) / ub
        step = mu * (ub - lb) / (norm)
        lambda_ij += step * subgradient
        print(f"Iteration: {h_iter}, UB: {ub:.4f}, LB: {lb:.4f}, Gap: {gap:.4f}%")
        if norm < 1e-4 or mu < 1e-4:
            print(norm, mu,h_iter,gap, lb)
            break
        solution = []
        for i in I:
```

```
            for j in J:
                if v_vars[i][j].x >= 0.99:
                    solution.append((i, j))
        print_solution_lagrangian(solution)
```

Rodando problemas

Testando o Problema

```
    datas = return_datas()
    dataset = []
    for name, data in datas.items():
        print(name)
        print('\n')
        dataset.append(data)
        solver = AssortmentSolver(data)
        solver.solve()
        solver.print_solution()
        print('\n')
        AssortmentSolverLagrangean(data, solver.get_ub())
        print('\n')
```

```
ABMR_1.json


Solução encontrada em 0.05 segundos
Quantidade total de objetos é 2
Quantidade total de Itemns é 2
Item 1 é colocado no objeto 1
  - Posição: (2.00, 2.00)
  - Tamanho do item: 2 x 11
  - Tamanho do objeto: 22 x 17

Item 2 é colocado no objeto 1
  - Posição: (39.00, 5.00)
  - Tamanho do item: 5 x 5
  - Tamanho do objeto: 22 x 17
```

```
Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 14 x 30


Iteration: 1, UB: 139876.0000, LB: 0.0000, Gap: 100.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 2
Item 1 é colocado no objeto 1
  - Tamanho do item: 2 x 11
  - Tamanho do objeto: 22 x 17

Item 2 é colocado no objeto 1
  - Tamanho do item: 5 x 5
  - Tamanho do objeto: 22 x 17

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 14 x 30
Iteration: 2, UB: 139876.0000, LB: 139876.0000, Gap: 0.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 2
Item 1 é colocado no objeto 2
  - Tamanho do item: 2 x 11
  - Tamanho do objeto: 14 x 30

Item 2 é colocado no objeto 2
  - Tamanho do item: 5 x 5
  - Tamanho do objeto: 14 x 30

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 22 x 17


ABMR_2.json


Solução encontrada em 0.00 segundos
Quantidade total de objetos é 2
Quantidade total de Itemns é 1
```

```
Item 1 é colocado no objeto 2
  - Posição: (4.00, 4.00)
  - Tamanho do item: 4 x 10
  - Tamanho do objeto: 24 x 10

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 17 x 29


Iteration: 1, UB: 57600.0000, LB: 0.0000, Gap: 100.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 1
Item 1 é colocado no objeto 1
  - Tamanho do item: 4 x 10
  - Tamanho do objeto: 17 x 29

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 24 x 10
Iteration: 2, UB: 57600.0000, LB: 57600.0000, Gap: 0.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 1
Item 1 é colocado no objeto 2
  - Tamanho do item: 4 x 10
  - Tamanho do objeto: 24 x 10

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 17 x 29


ABMR_3.json


Solução encontrada em 0.00 segundos
Quantidade total de objetos é 2
Quantidade total de Itemns é 1
Item 1 é colocado no objeto 1
  - Posição: (5.00, 5.00)
  - Tamanho do item: 5 x 4
```

```
  - Tamanho do objeto: 18 x 19

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 26 x 22


Iteration: 1, UB: 116964.0000, LB: 0.0000, Gap: 100.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 1
Item 1 é colocado no objeto 1
  - Tamanho do item: 5 x 4
  - Tamanho do objeto: 18 x 19

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 26 x 22
Iteration: 2, UB: 116964.0000, LB: 116964.0000, Gap: 0.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 1
Item 1 é colocado no objeto 2
  - Tamanho do item: 5 x 4
  - Tamanho do objeto: 26 x 22

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 18 x 19


ABMR_4.json


Solução encontrada em 0.02 segundos
Quantidade total de objetos é 3
Quantidade total de Itemns é 3
Item 1 é colocado no objeto 3
  - Posição: (3.00, 3.00)
  - Tamanho do item: 3 x 3
  - Tamanho do objeto: 17 x 13

Item 2 é colocado no objeto 3
```

```
  - Posição: (30.00, 4.00)
  - Tamanho do item: 4 x 2
  - Tamanho do objeto: 17 x 13

Item 3 é colocado no objeto 3
  - Posição: (7.00, 25.00)
  - Tamanho do item: 7 x 1
  - Tamanho do objeto: 17 x 13

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 24 x 12
  - Objetos 2 - tamanho 15 x 18


Iteration: 1, UB: 48841.0000, LB: 0.0000, Gap: 100.0000%
Solução encontrada
Quantidade total de objetos é 3
Quantidade total de Itens é 3
Item 1 é colocado no objeto 3
  - Tamanho do item: 3 x 3
  - Tamanho do objeto: 17 x 13

Item 2 é colocado no objeto 3
  - Tamanho do item: 4 x 2
  - Tamanho do objeto: 17 x 13

Item 3 é colocado no objeto 3
  - Tamanho do item: 7 x 1
  - Tamanho do objeto: 17 x 13

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 24 x 12
  - Objetos 2 - tamanho 15 x 18
Iteration: 2, UB: 48841.0000, LB: 48841.0000, Gap: 0.0000%
Solução encontrada
Quantidade total de objetos é 3
Quantidade total de Itens é 3
Item 1 é colocado no objeto 1
  - Tamanho do item: 3 x 3
  - Tamanho do objeto: 24 x 12
```

```
Item 2 é colocado no objeto 1
  - Tamanho do item: 4 x 2
  - Tamanho do objeto: 24 x 12

Item 3 é colocado no objeto 1
  - Tamanho do item: 7 x 1
  - Tamanho do objeto: 24 x 12

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 15 x 18
  - Objetos 3 - tamanho 17 x 13


ABMR_5.json


Solução encontrada em 0.00 segundos
Quantidade total de objetos é 2
Quantidade total de Itemns é 2
Item 1 é colocado no objeto 1
  - Posição: (7.00, 7.00)
  - Tamanho do item: 7 x 1
  - Tamanho do objeto: 20 x 10

Item 2 é colocado no objeto 1
  - Posição: (11.00, 19.00)
  - Tamanho do item: 11 x 1
  - Tamanho do objeto: 20 x 10

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 29 x 12


Iteration: 1, UB: 40000.0000, LB: 0.0000, Gap: 100.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 2
Item 1 é colocado no objeto 1
  - Tamanho do item: 7 x 1
  - Tamanho do objeto: 20 x 10
```

```
Item 2 é colocado no objeto 1
  - Tamanho do item: 11 x 1
  - Tamanho do objeto: 20 x 10

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 29 x 12
Iteration: 2, UB: 40000.0000, LB: 40000.0000, Gap: 0.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 2
Item 1 é colocado no objeto 2
  - Tamanho do item: 7 x 1
  - Tamanho do objeto: 29 x 12

Item 2 é colocado no objeto 2
  - Tamanho do item: 11 x 1
  - Tamanho do objeto: 29 x 12

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 20 x 10


ABMR_6.json


Solução encontrada em 0.01 segundos
Quantidade total de objetos é 2
Quantidade total de Itemns é 3
Item 1 é colocado no objeto 1
  - Posição: (11.00, 11.00)
  - Tamanho do item: 11 x 11
  - Tamanho do objeto: 22 x 17

Item 2 é colocado no objeto 1
  - Posição: (42.00, 2.00)
  - Tamanho do item: 2 x 11
  - Tamanho do objeto: 22 x 17

Item 3 é colocado no objeto 1
  - Posição: (39.00, 29.00)
  - Tamanho do item: 5 x 5
```

```
  - Tamanho do objeto: 22 x 17

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 14 x 30


Iteration: 1, UB: 139876.0000, LB: 0.0000, Gap: 100.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 3
Item 1 é colocado no objeto 1
  - Tamanho do item: 11 x 11
  - Tamanho do objeto: 22 x 17

Item 2 é colocado no objeto 1
  - Tamanho do item: 2 x 11
  - Tamanho do objeto: 22 x 17

Item 3 é colocado no objeto 1
  - Tamanho do item: 5 x 5
  - Tamanho do objeto: 22 x 17

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 14 x 30
Iteration: 2, UB: 139876.0000, LB: 139876.0000, Gap: 0.0000%
Solução encontrada
Quantidade total de objetos é 2
Quantidade total de Itens é 3
Item 1 é colocado no objeto 2
  - Tamanho do item: 11 x 11
  - Tamanho do objeto: 14 x 30

Item 2 é colocado no objeto 2
  - Tamanho do item: 2 x 11
  - Tamanho do objeto: 14 x 30

Item 3 é colocado no objeto 2
  - Tamanho do item: 5 x 5
  - Tamanho do objeto: 14 x 30

Todos os Itens foram escolhidos.
```

```
Objetos não escolhidos:
  - Objetos 1 - tamanho 22 x 17


ABMR_7.json


Solução encontrada em 0.01 segundos
Quantidade total de objetos é 3
Quantidade total de Itemns é 2
Item 1 é colocado no objeto 2
  - Posição: (9.00, 9.00)
  - Tamanho do item: 9 x 6
  - Tamanho do objeto: 19 x 17

Item 2 é colocado no objeto 2
  - Posição: (5.00, 31.00)
  - Tamanho do item: 5 x 3
  - Tamanho do objeto: 19 x 17

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 27 x 23
  - Objetos 3 - tamanho 19 x 19


Iteration: 1, UB: 104329.0000, LB: 0.0000, Gap: 100.0000%
Solução encontrada
Quantidade total de objetos é 3
Quantidade total de Itens é 2
Item 1 é colocado no objeto 3
  - Tamanho do item: 9 x 6
  - Tamanho do objeto: 19 x 19

Item 2 é colocado no objeto 3
  - Tamanho do item: 5 x 3
  - Tamanho do objeto: 19 x 19

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 1 - tamanho 27 x 23
  - Objetos 2 - tamanho 19 x 17
Iteration: 2, UB: 104329.0000, LB: 104329.0000, Gap: 0.0000%
```

```
Solução encontrada
Quantidade total de objetos é 3
Quantidade total de Itens é 2
Item 1 é colocado no objeto 1
  - Tamanho do item: 9 x 6
  - Tamanho do objeto: 27 x 23

Item 2 é colocado no objeto 1
  - Tamanho do item: 5 x 3
  - Tamanho do objeto: 27 x 23

Todos os Itens foram escolhidos.
Objetos não escolhidos:
  - Objetos 2 - tamanho 19 x 17
  - Objetos 3 - tamanho 19 x 19
```


Andrade, Ricardo, Ernesto G Birgin, Reinaldo Morabito, and Ronconi. 2014. "MIP Models for Two-Dimensional Non-Guillotine Cutting Problems with Usable Leftovers." *Journal of the Operational Research Society*. Taylor & Francis.

Oliveira, Óscar, Dorabela Gamboa, and Elsa Silva. n.d. "RESOURCES FOR TWO-DIMENSIONAL (AND THREE-DIMENSIONAL) CUTTING AND PACKING SOLUTION METHODS RESEARCH." *Journal of the Operational Research Society*.