The Container Stacking Problem

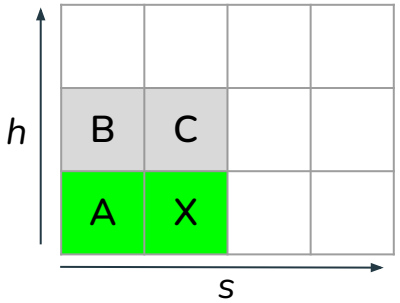
-

Final Presentation

Daniel Monteiro & Rafael Ribeiro

Previously on...

Problem definition

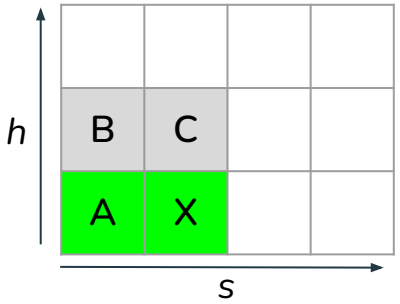


Given:

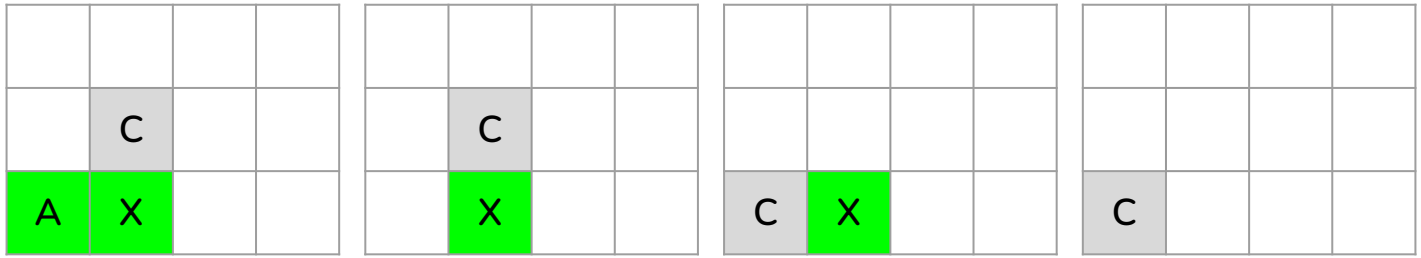
- The starting positions of the containers
- The number of stacks (*s*)
- The maximum height of the stacks (*h*)
- The maximum number of operations (*t*)
- A list of the containers we wish to remove (A and X, in this case)

Minimize the number of operations necessary to remove the desired containers

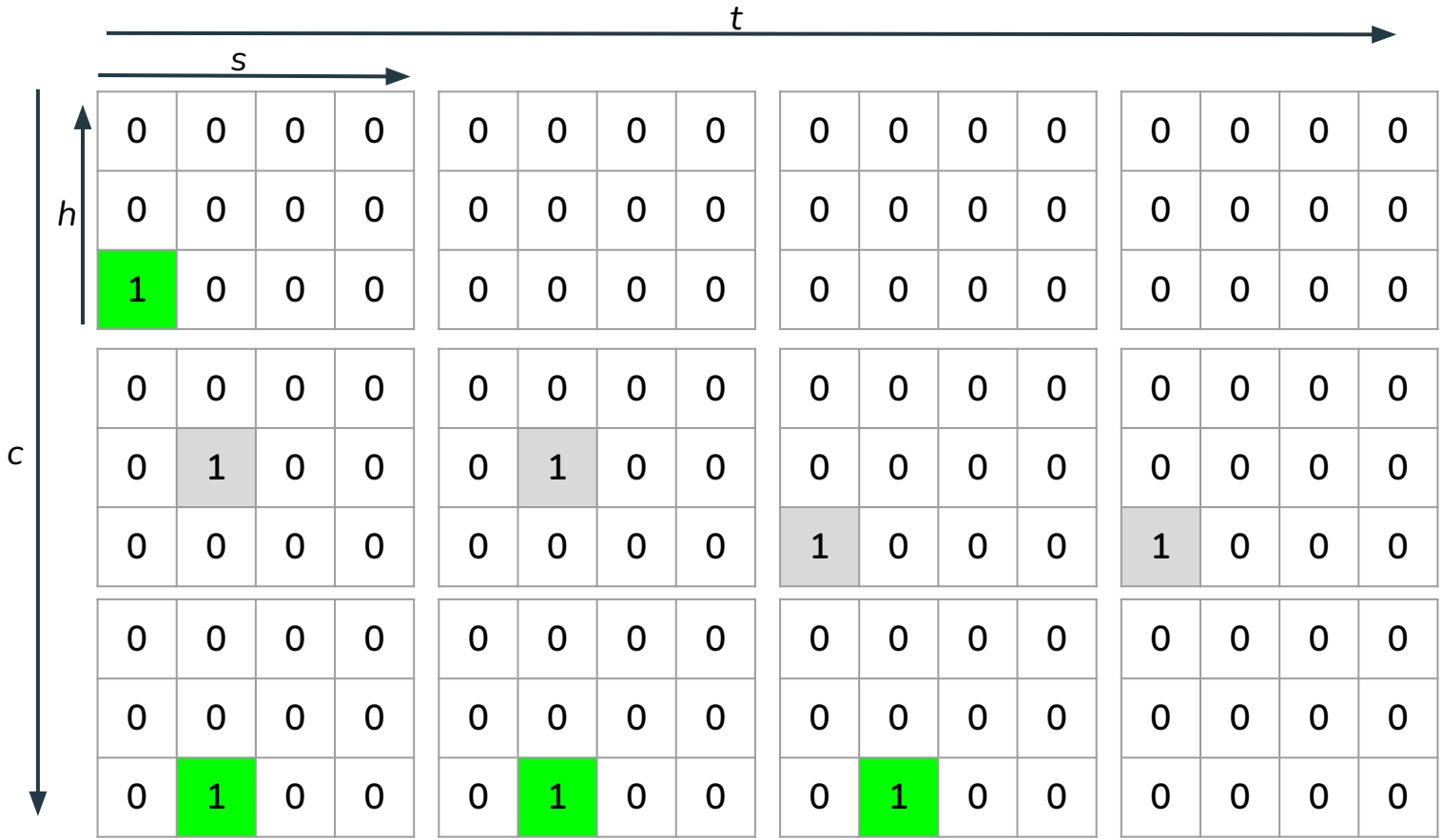
Three possible operations



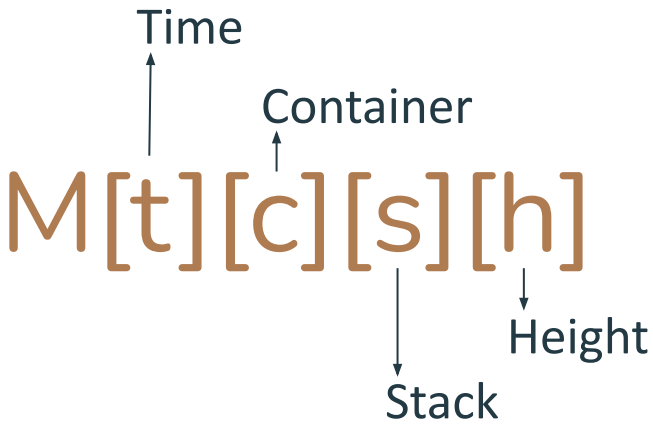
1. Move top container from stack *a* to stack *b*
2. Remove container from stack *a*
3. Idle (leave all the containers untouched)



Previously on...



Previously on...



Previously on...

	t=0	t=1	t=2	t=3
A	1	0	0	0
C	1	1	1	1
X	1	1	1	0

Lifetimes

	op1	op2	op3
Emplace	0	1	0
Idle	0	0	0
Remove	1	0	1

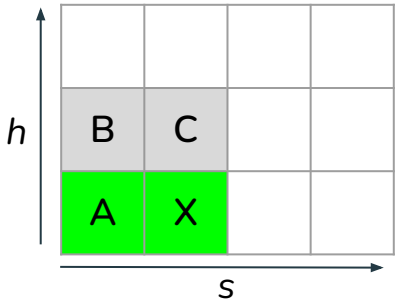
Operations

In	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	1	0	0	0	0	0	0	0
Out	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	1	0	0

Previously on...

Here’s what’s new

Problem definition

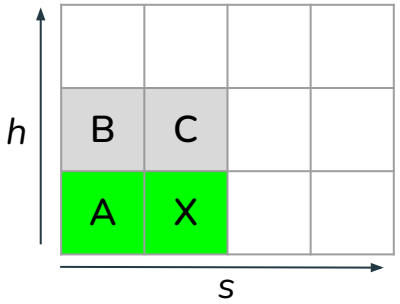


Given:

- The starting positions of the containers
- The weight of the containers (lighter containers on top of the heavier)
- The number of stacks (*s*)
- The maximum height of the stacks (*h*)
- The list of shipments and for how long they are docked
    - Each shipment can add or remove some containers to the problem

Minimize the number of operations necessary to remove the desired containers

Four possible operations



1. Move top container from stack *a* to stack *b*
2. Remove container from stack *a*
3. Idle (leave all the containers untouched)
4. Insert container to stack a

Example of a problem instance


```
{
   "containers" : [
       ["X", 0, 0],
       ["Y", 1, 0]
   ],
   "dimensions" : [4, 3],
   "shipments": [
       {
           "in": ["A", "B", "C", "D", "E", "F", "G"],
           "out": [],
           "duration": 10,
       },
       {
           "duration": 5,
       },
       {
           "in": ["1" ,"2" ,"3"],
           "out": ["B", "D", "G"],
           "duration": 10,
       },
       …
       {
           "in": [],
           "out": ["A" ,"C" ,"F"],
           "duration": 6
       },
       {
           "duration": 10
       },
       {
           "in": [],
           "out": ["1", "3", "Y"],
           "duration": 5,
       }
   ],
   "weights" : {
       "A" : 2,
       "1" : 2
   }
}
```

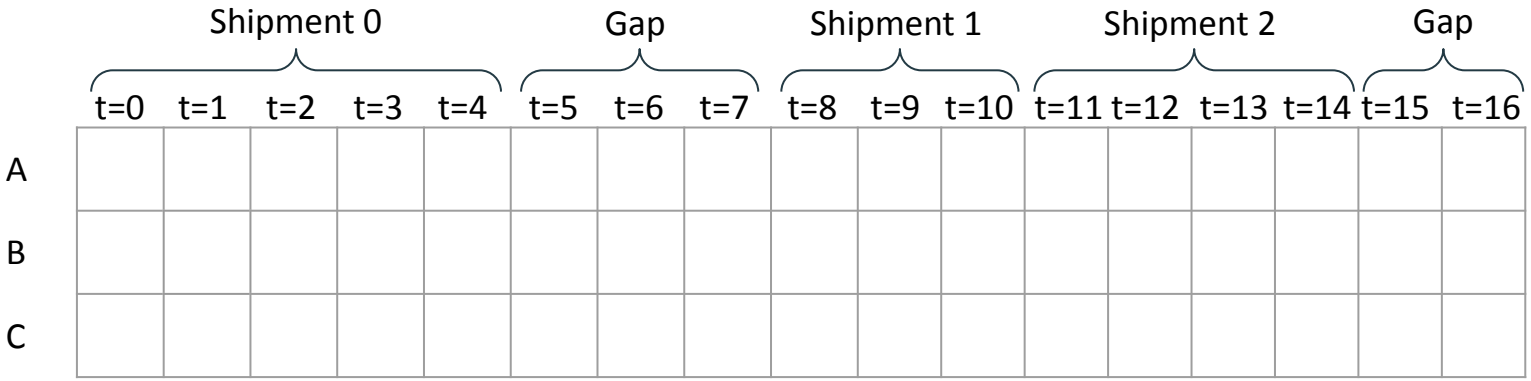
Feature highlight: shipments

Feature highlight: shipments

- Shipment 0
  - Duration: 5 | In: [**A**, **B**] | Out: []
- Gap
  - Duration: 3
- Shipment 1
  - Duration: 3 | In: [**C**] | Out: [**A**]
- Shipment 2
  - Duration: 4 | In: [] | Out: [**B**]
- Gap
  - Duration: 2

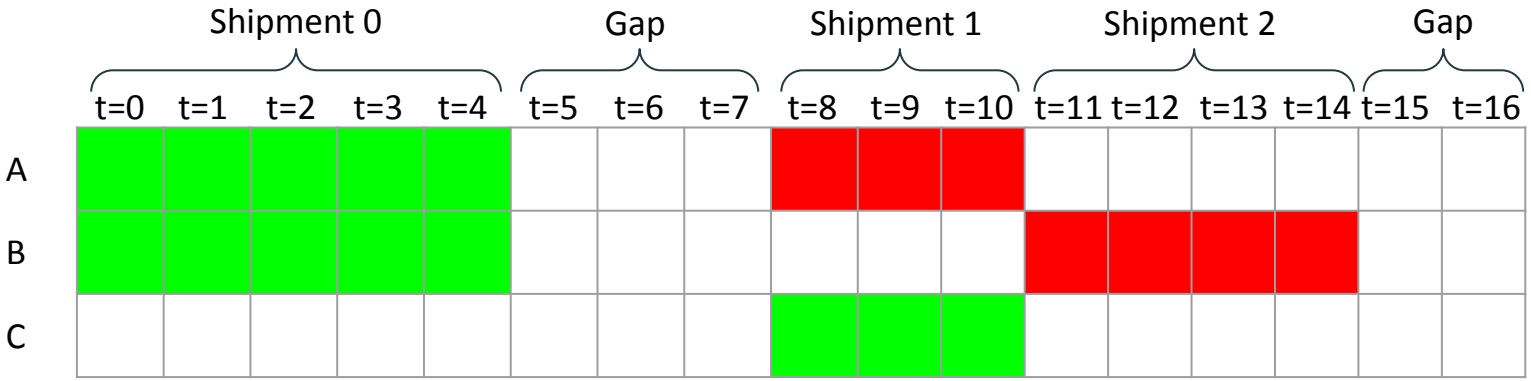
Dimension T = 5 + 3 + 3 + 4 + 2 = 17

Feature highlight: shipments



Lifetimes

Feature highlight: shipments

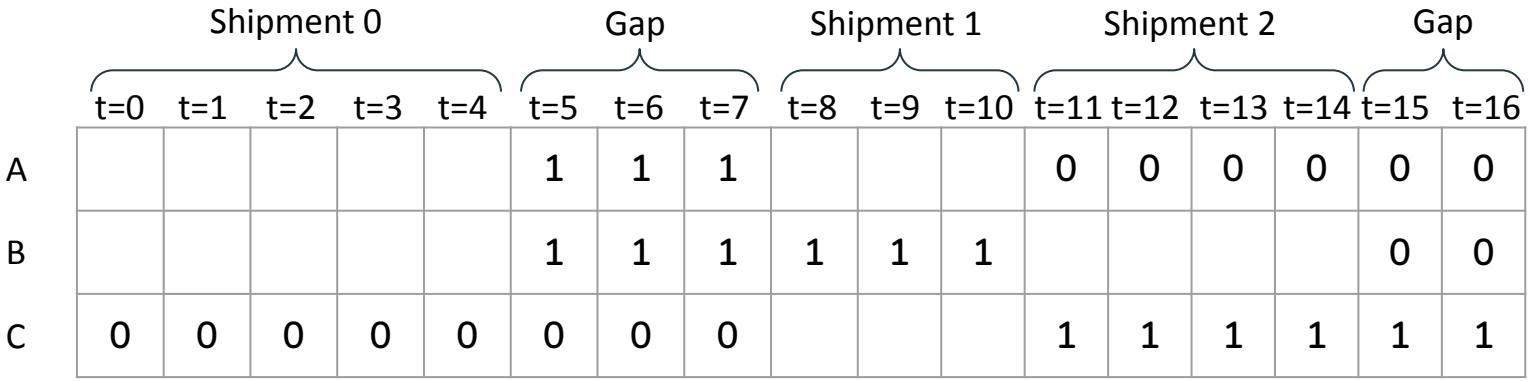


Lifetimes

Feature highlight: shipments

Lifetimes

Feature highlight: shipments



Lifetimes

The spaces in blank are left as an exercise to ~~the reader~~ cplex…

Feature highlight: weight restrictions

Feature highlight: weight restrictions

For every possible container location:
    if container exists:
        enforce lighter containers not being able to be placed below it

```
def enforce_weight_restrictions(model : Model, matrix : ContainerMatrix, weights : dict, index_lookup):
    weight_array = [0] * len(index_lookup)

    for c, weight in weights.items():
        weight_array[index_lookup[c]] = weight

    for t in range(matrix.t):
        for s in range(matrix.s):
            for h in range(matrix.h):
                for c in range(matrix.c):
                    container_is_here = model.NewBoolVar('b')
                    model.AddIf(matrix.get(t, c, s, h) == 1, container_is_here)
                    model.AddIf(matrix.get(t, c, s, h) == 0, model.Not(container_is_here))

                    for container in range(matrix.c):
                        for height in range(matrix.h):
                            if container != c and height < h and weight_array[container] < weight_array[c]:
                                model.AddIf(matrix.get(t, container, s, height) == 0, container_is_here)
```

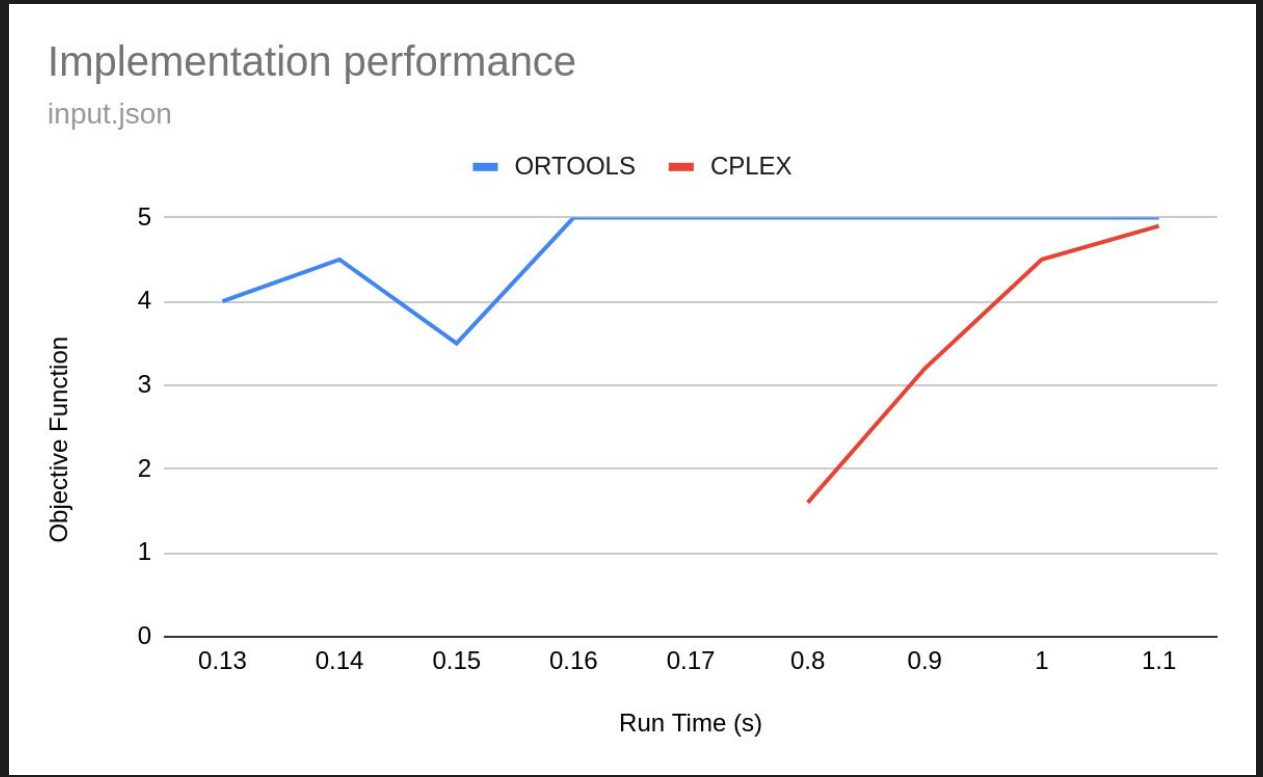
For every possible container location

If container is lower and lighter

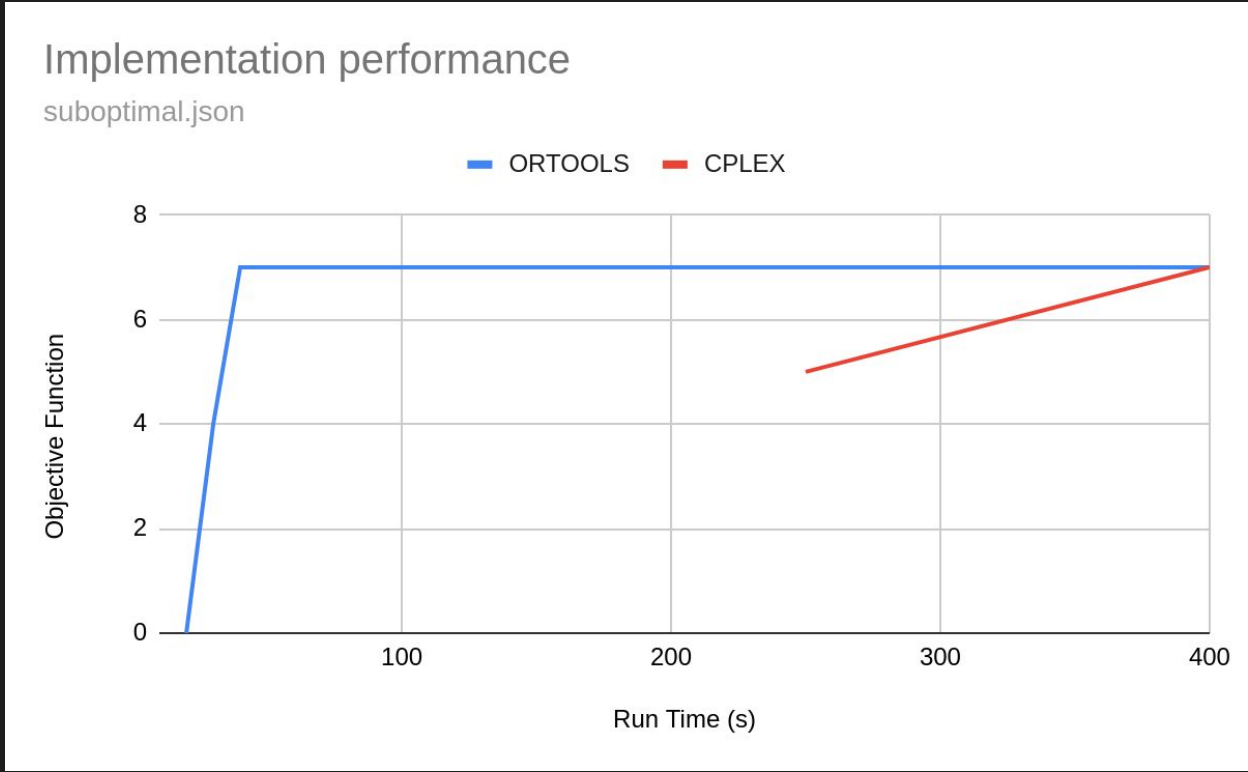
Don’t

Performance analysis: Intermediate version

```
{
    "containers" : [
        ["A", 0, 0],
        ["B", 0, 1],
        ["C", 1, 1],
        ["X", 1, 0]
    ],
    "dimensions" : [10, 4, 3],
    "remove" : ["A", "X"]
}
```

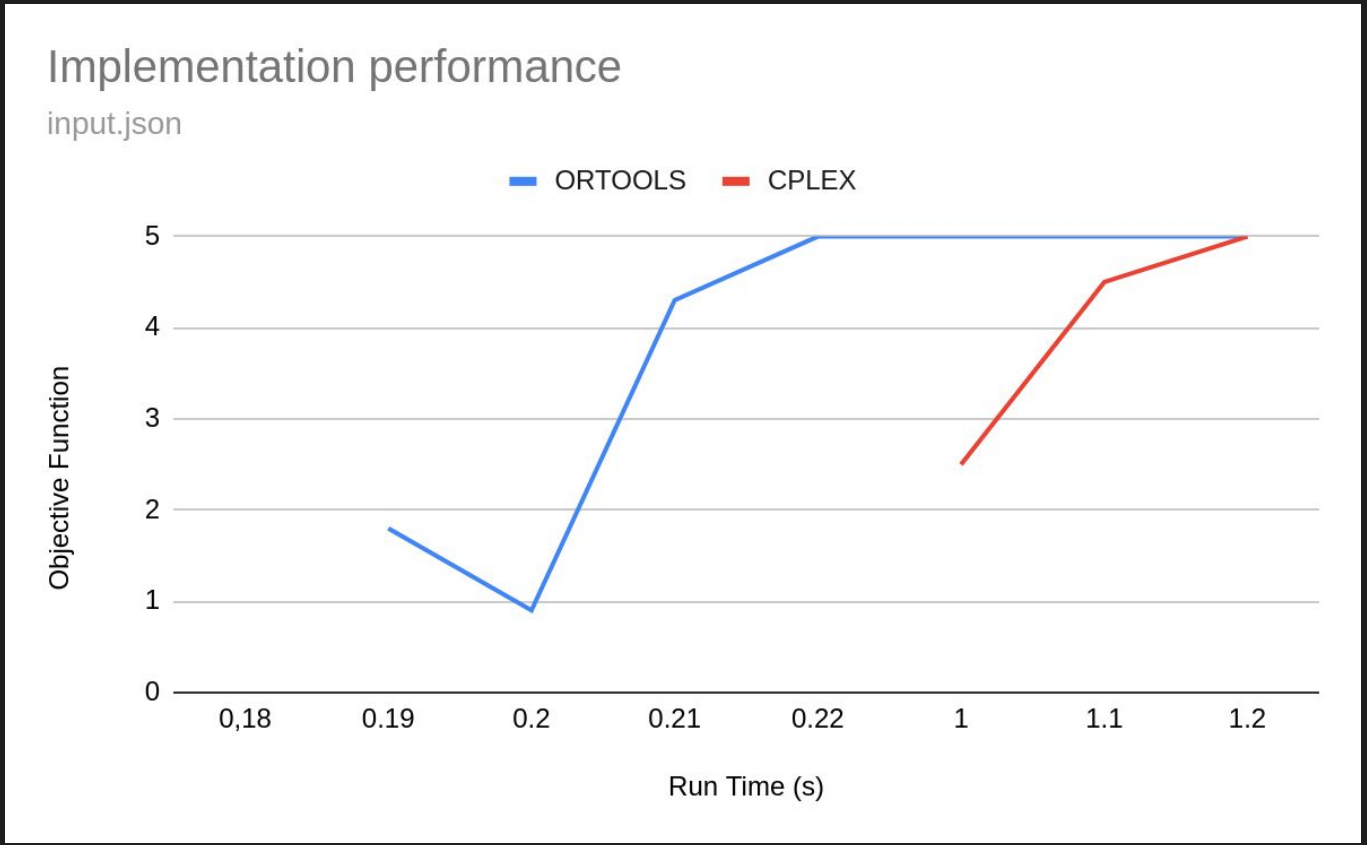


```
{
   "containers" : [
       ["A", 0, 0],
       ["W", 0, 1],
       ["B", 0, 2],
       ["C", 0, 3],
       ["X", 0, 4],
       ["D", 0, 5],
       ["E", 1, 0],
       ["Y", 1, 1],
       ["F", 1, 2],
       ["G", 1, 3],
       ["Z", 2, 0],
       ["H", 2, 1],
       ["I", 2, 2],
       ["J", 2, 3]
   ],
   "dimensions" : [20, 4, 6],
   "remove" : ["W", "X", "Y", "Z"]
}
```



Performance analysis: Final version

```
{
    "containers" : [
        ["A", 0, 0],
        ["B", 0, 1],
        ["C", 1, 1],
        ["X", 1, 0]
    ],
    "dimensions" : [4, 3],
    "shipments": [
        {
            "duration": 1
        },
        {
            "in": [],
            "out": [
        "A", "B", "C", "X"
             ],
            "duration": 10
        }
    ],
    "weights" : {
        "A" : 2,
        "B" : 2,
        "C" : 2,
        "X" : 2
    }
}
```



```
{
   "containers" : [
       ["9", 0, 0],
       ["8", 0, 1],
       ["7", 0, 2],
       ["6", 0, 3],
       ["5", 0, 4],
       ["4", 0, 5],
       ["3", 0, 6]
   ],
   "dimensions" : [3, 7],
   "shipments" : [
       {
           "duration" : 33
       },
       {
           "in": [],
           "out": ["9"],
           "duration" : 2
       }
   ],
   "weights" : {
       "9" : 9,
       "8" : 8,
       "7" : 7,
       "6" : 6,
       "5" : 5,
       "4" : 4,
       "3" : 3
   }
}
```