Bin Packing “How”

Basics

Intuition

Inspiration

ML2R

Autumn School

Sebastian Mueller

Outline

1. Task definition 2D Bin Packing
2. Heuristics and Data
3. Starting Points for ML
4. The Competition & Next Up

2D Bin Packing @ ML2R Autumn School

- Variation on 2D geometric bin packing
- Task Definition:

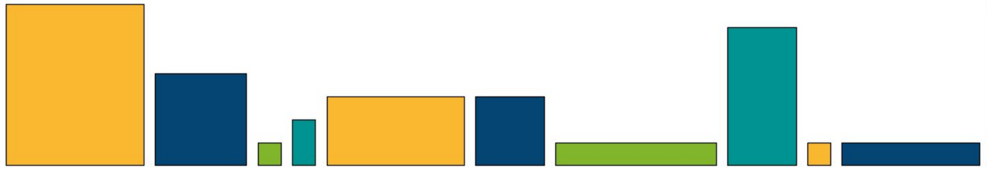
  Given a set of objects (boxes), described only by the two attributes width and height, place them inside as few identical square containers (bins) as possible, without the objects overlapping or extending beyond the bins' boundaries

2D Bin Packing @ ML2R Autumn School

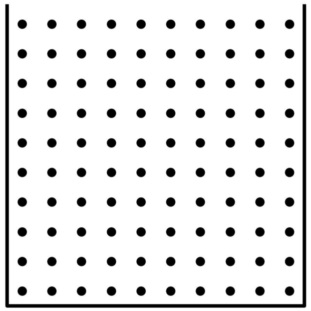
Setting:

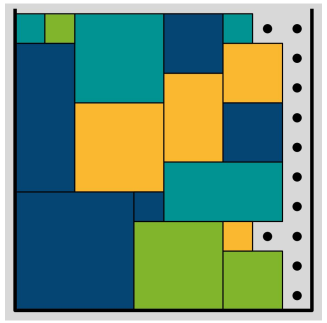
- Given: Set of boxes
- Return: For each box

  (box, bin_id, position)



- Bins: identical, 10x10
- Items in [1, 2, ..., 10]x[1, 2, ..., 10]
- **Items cannot be rotated**



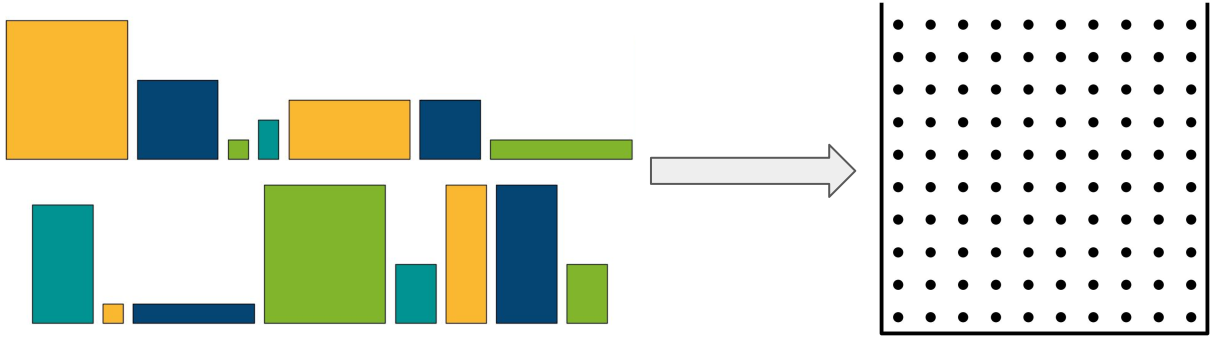


Outline

1. Task definition 2D Bin Packing ✓
2. Heuristics and Data
3. Starting Points for ML
4. The Competition & Next Up

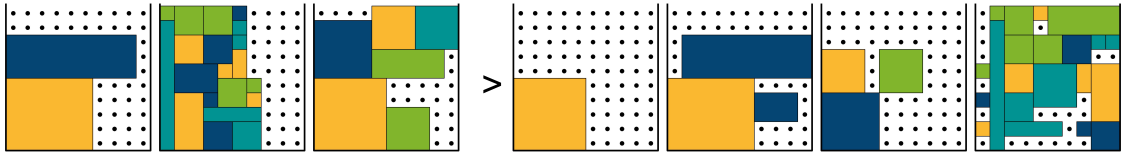
Simple Heuristics

- Next Fit
- First Fit
- Max Rest
- Best Fit
- First Fit Decreasing

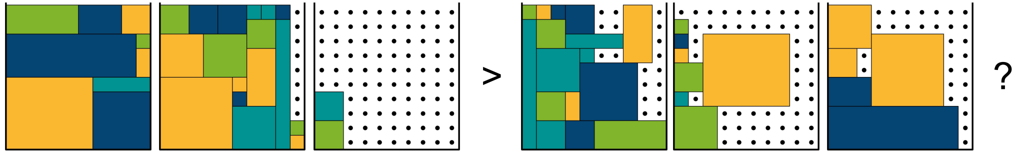


Comparing Heuristics - Criteria

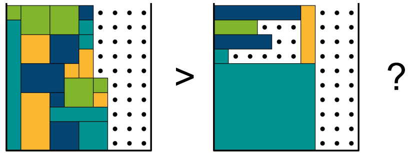
- Number of Bins



  - Density



    - Contiguity



- Runtime

Simple Heuristics - Next Fit (NF)

```
Next Fit(list Boxes):
    Bins = list []
    active_bin = new Bin()
    Bins = [active_bin]
    for box in Boxes:
        if fits(box, active_bin):
            place(box, active_bin)
        else:
            Bins.append(active_bin)
            active_bin = new Bin()
            place(box, active_bin)
    return Bins
```

Simple Heuristics - First Fit (FF)

```
First Fit(list Boxes):
    Bins = [new Bin()]
    for box in Boxes:
        placement_success = False
        for bin in Bins:
            if fits(box, bin):
                place(box, bin)
                placement_success = True
                break
        if not placement_succes:
            bin = new Bin()
            place(box, bin)
            Bins.append(bin)
    return Bins
```

Simple Heuristics - Max Rest (MR)

```
Max Rest(list Boxes):
    Bins = list []
    for box in Boxes:
        possible_bins = [bin for bin in Bins if fits(box, bin)]
        if possible_bins.empty():
            new_bin = new Bin()
            Bins.append(new_bin)
            possible_bins.append(new_bin)
        emptiest_bin = sort_descend(possible_bins, key=capacity)[0]
        place(box, emptiest_bin)
    return Bins
```

Simple Heuristics - Best Fit (BF)

```
Best Fit(list Boxes):
    Bins = list []
    for box in Boxes:
        possible_bins = [bin for bin in Bins if fits(box, bin)]
        if possible_bins.empty():
            new_bin = new Bin()
            Bins.append(new_bin)
            possible_bins.append(new_bin)
        fullest_bin = sort_ascend(possible_bins, key=capacity)[0]
        place(box, fullest_bin)
    return Bins
```

In other words:

Best Fit <-> “Least” Rest

Max Rest <-> “Worst” Fit

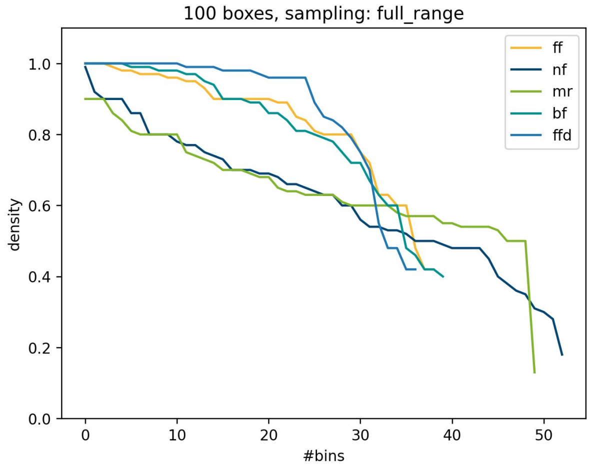
Simple Heuristics - First Fit Decreasing (FFD)

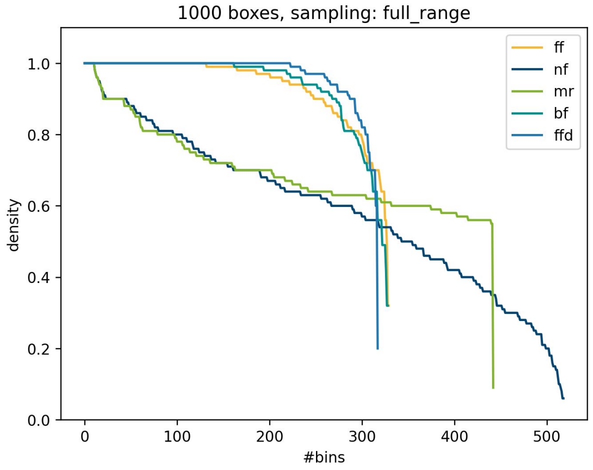
```
First Fit Decreasing(list Boxes):
    Bins = [new Bin()]
    sBoxes = sorted_descend(Boxes, key=area)
    for box in sBoxes:
        placement_success = False
        for bin in Bins:
            if fits(box, bin):
                place(box, bin)
                placement_success = True
                break
        if not placement_succes:
            bin = new Bin()
            place(box, bin)
            Bins.append(bin)
    return Bins
```

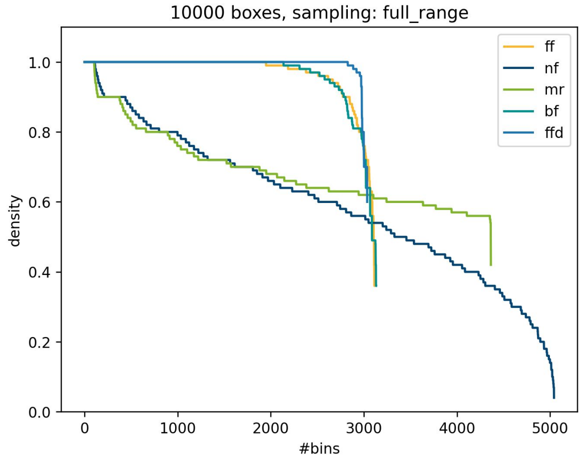
Comparing Heuristics - Sampling Boxes

- Creating Data: Sampling width and height **uniformly**
- Sets of various sizes: 100, 1k, 10k boxes

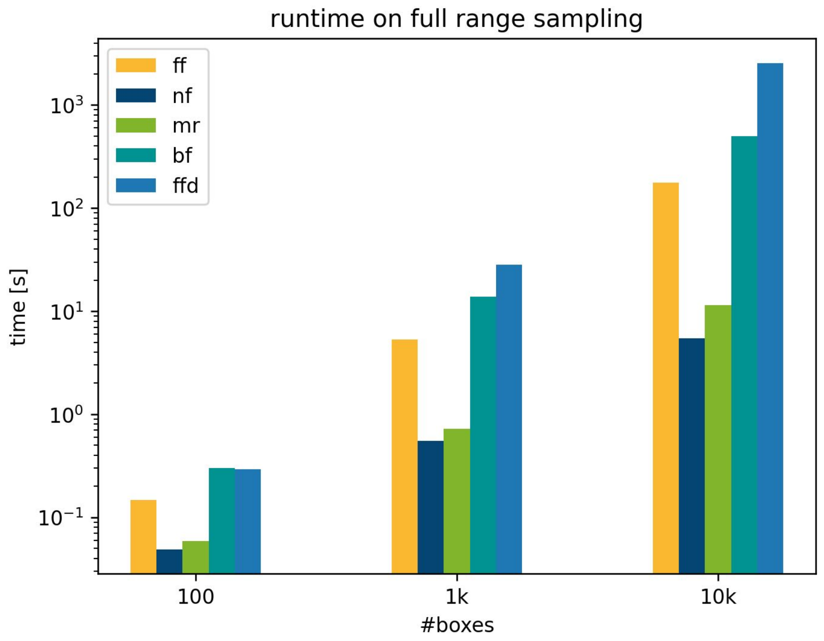
Comparison - Density







Comparison - Runtime



Comparing Heuristics - Sampling Boxes

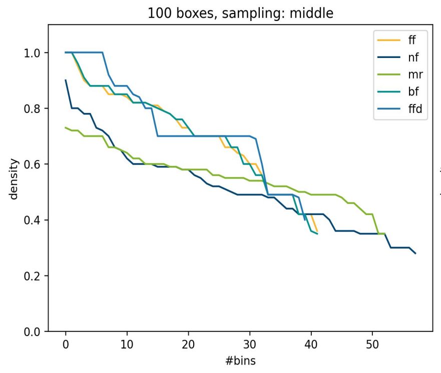
- Creating Data: Sampling width and height **uniformly -> “full range”**
- Sets of various sizes: 100, 1k, 10k boxes

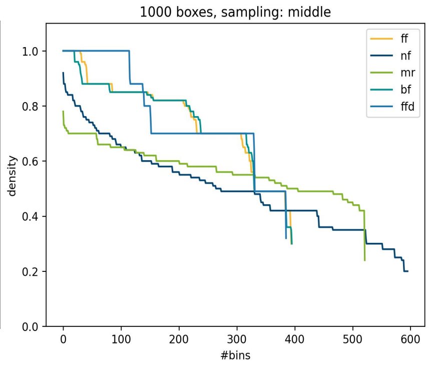
Slice the interval:

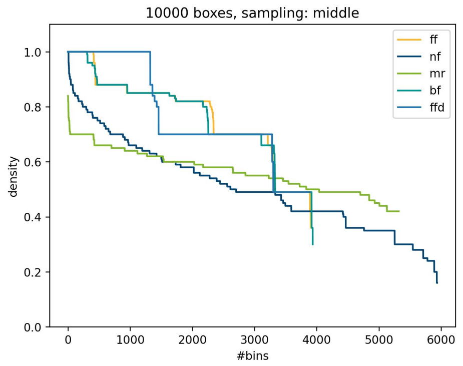
- “Medium”: boxes from [4,7]x[4,7]
- “Extremes”: boxes from [1,3]x[1,3] + [8,10]x[8,10]

Sampling - Effect on Density

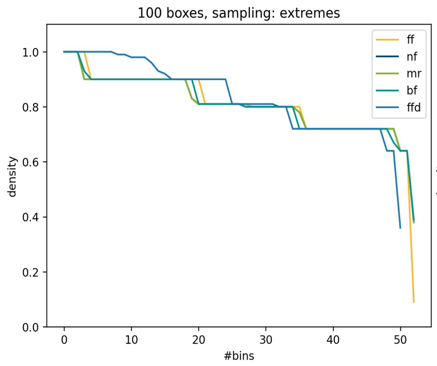
Medium

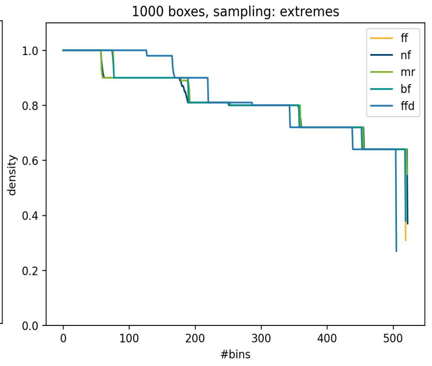


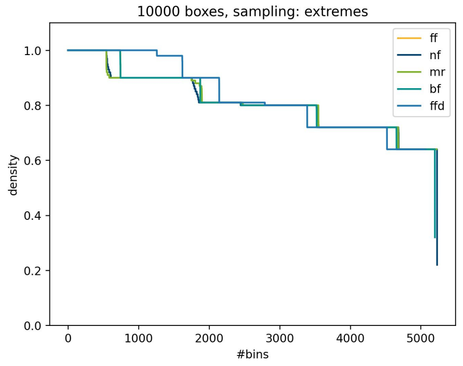




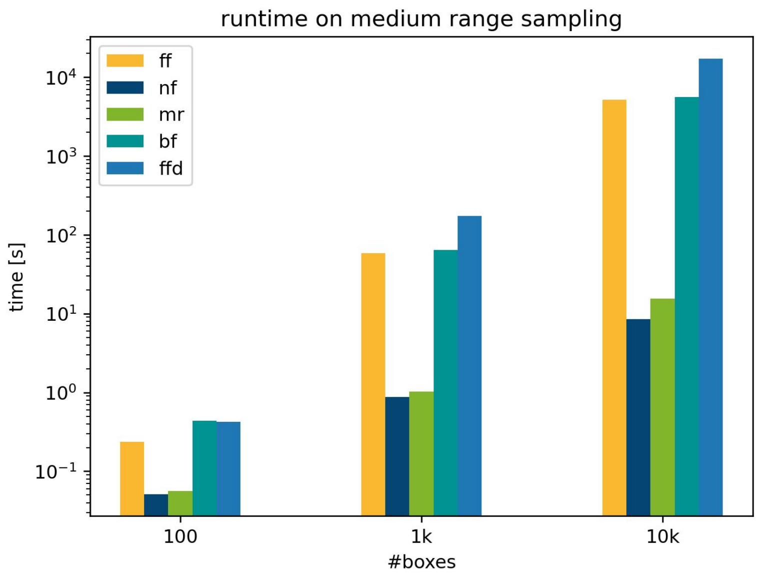
Extremes

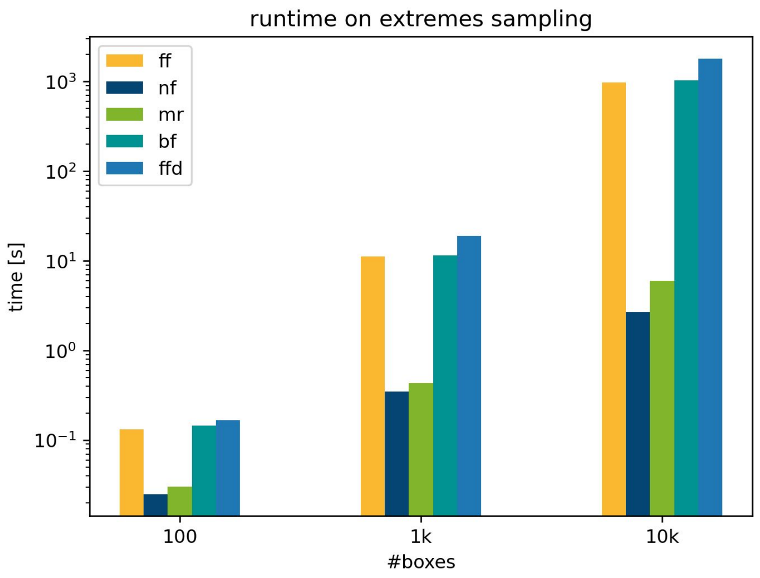






Sampling - Effect on Runtime





Simple Heuristics - Summary

Summary performance

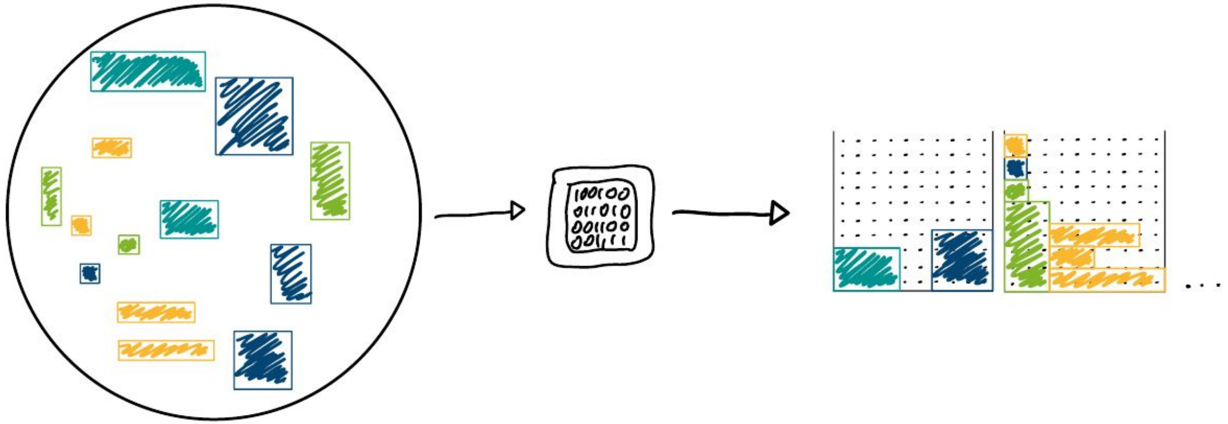
(better > worse)	Full range	Medium	Extremes
Density	FFD > BF >= FF >> MR > NF	FFD > BF >= FF >> MR > NF	FFD >= BF ~= FF ~= MR ~= NF
Runtime	NF > MR >> FF > BF >> FFD	NF > MR >>> FF = BF >> FFD	NF > MR >>> FF = BF > FFD

- Generally strong negative correlation between density and runtime
- Manipulating sampling had great impact on runtime of FF, FFD, BF
- Surprising: Heuristics perform very similar on “extreme” sampled sets

Outline

1. Task definition 2D Bin Packing ✓
2. Heuristics and Data ✓
3. Starting Points for ML
4. The Competition & Next Up

Starting Points for ML



Starting Points for ML

Ideas:

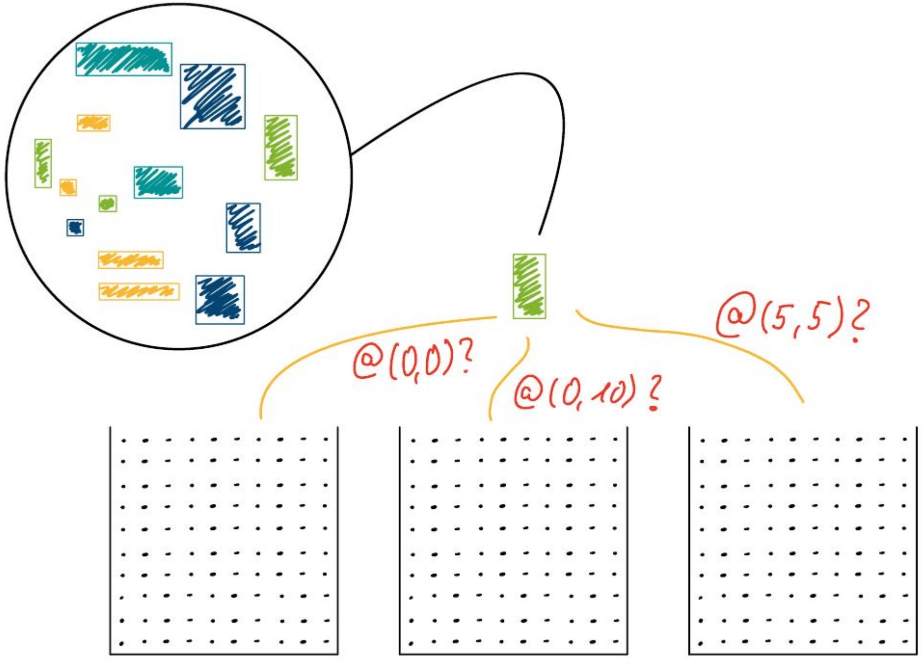
1. Learn to pick a bin and a position
2. Learn to pick a heuristic
3. Dense Prototypes
4. Re-ordering Sequences

Starting Points for ML

Two questions will reoccur:

1. What learning paradigm?
   - Reinforcement Learning: Intuitive; how to define fitness? (Suitable for this week?)
   - Supervised Learning: Less computational effort needed; how to generate training data?
2. How to represent the data or a solution?
   - Sequences? Sets? Histograms? Graphs?

Idea 1: Learn to pick a bin and a position



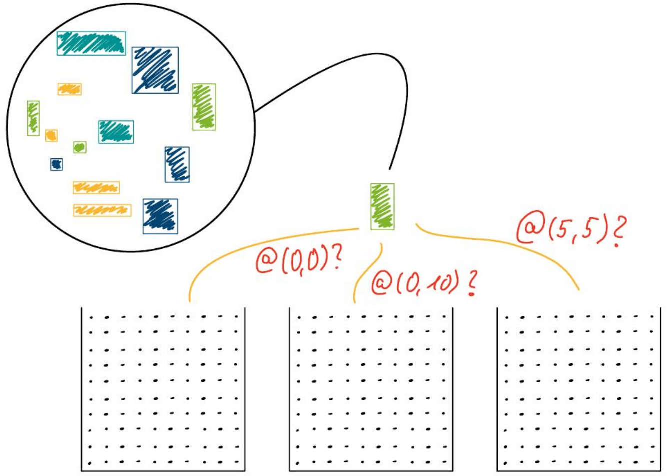
Idea 1: Learn to pick a bin and a position

+ Intuitive idea
+ Maybe the algorithm finds an interesting strategy
- Two choices each step: bin **and** position

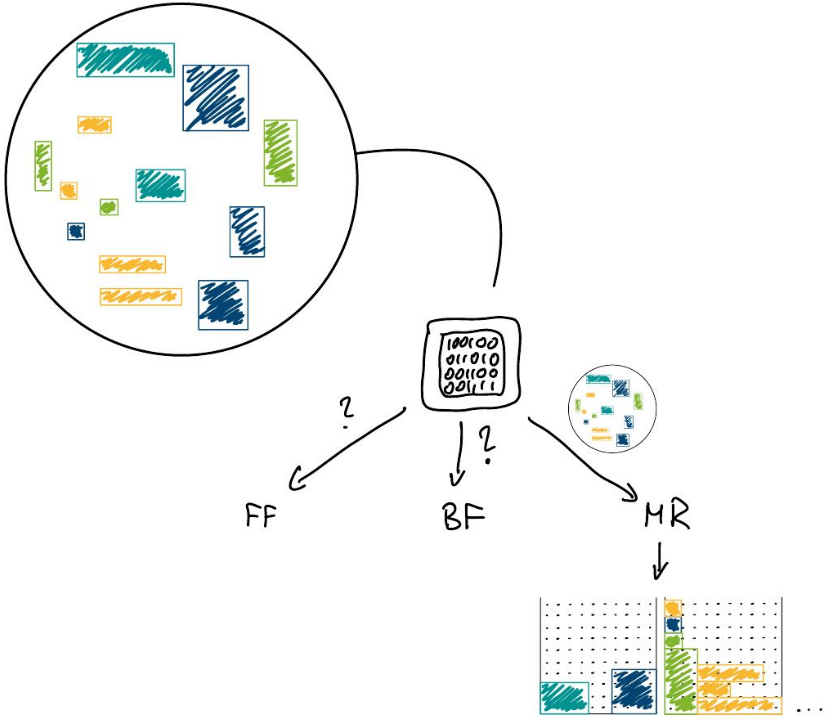
  -> not invariant against bin size

  -> harder the more constraints are on the placement

  -> fix number of bins? Dynamically add bins?



Idea 2: Learn to pick a heuristic

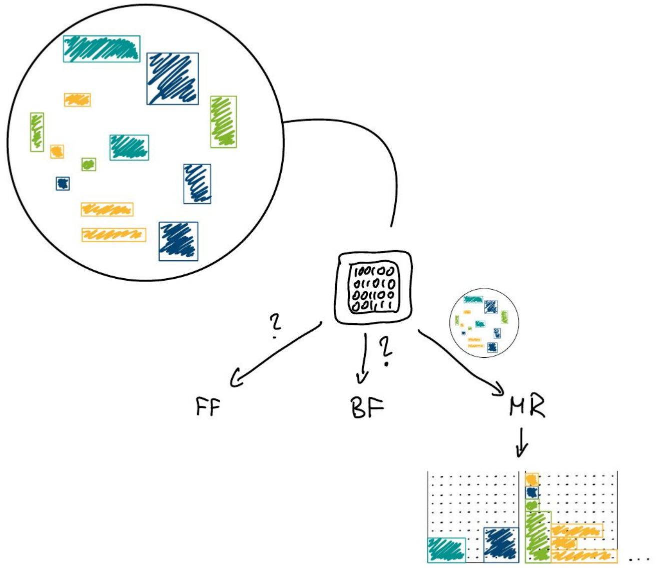


Idea 2: Learn to pick a heuristic

+ Theoretically independent of #boxes

- Performance bound by heuristics

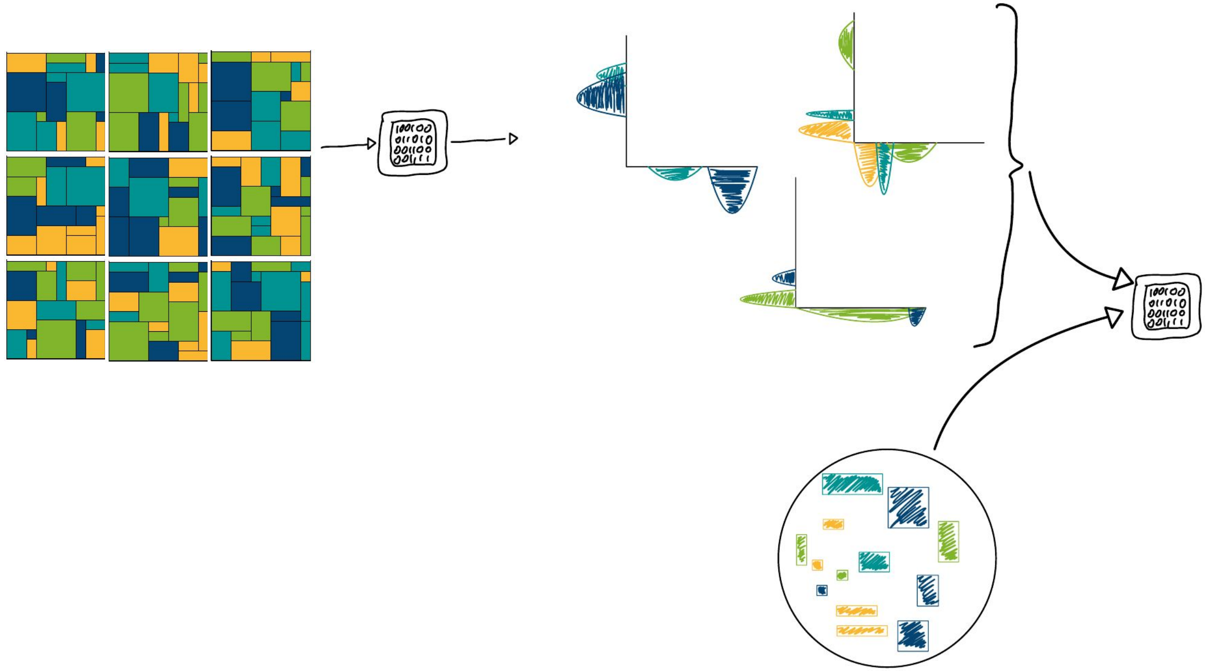
  - FFD will (probably) be overrepresented



How?

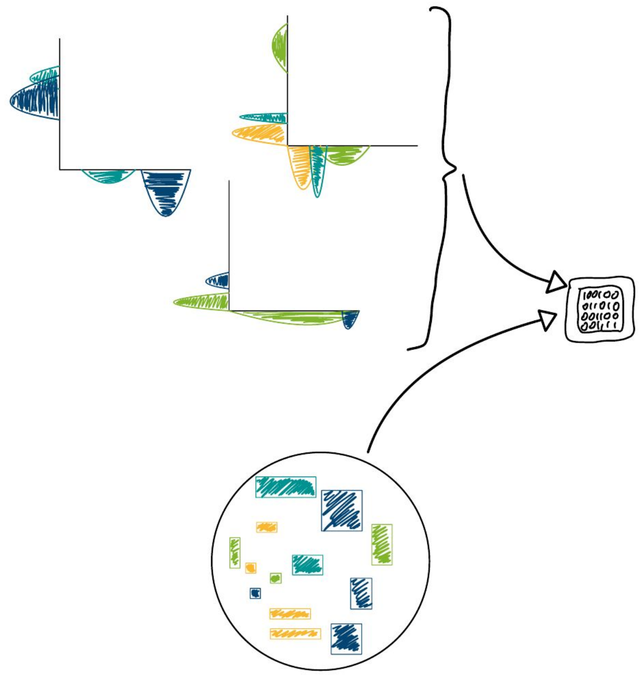
- Recurrent network + one hot output?
- Train Autoencoder; Decision Tree/ kNN on its latent space?

Idea 3: Dense Prototypes

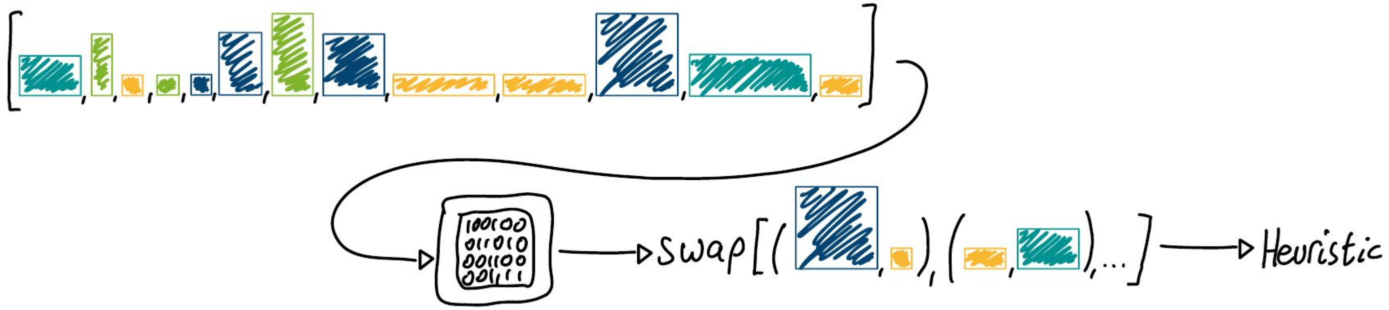


Idea 3: Dense Prototypes

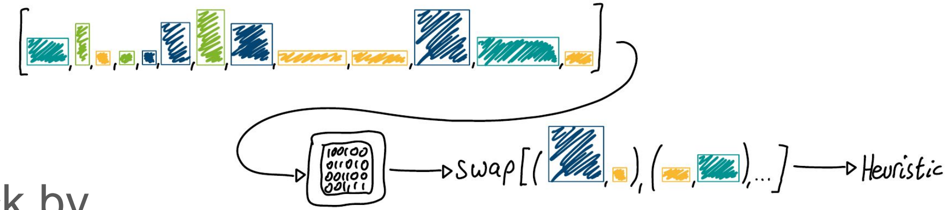
- Learn sequences/ sets that pack bins (close to) optimal
- Extract prototypes: **similar sets** of objects
  - Problem: Similar does not mean equally densely packable
- How to get varied prototypes?
- How to combine prototypes and given set?



Idea 4: Re-ordering Sequences



Idea 4: Re-ordering Sequences



- Training Data: Generate Samples, Pre-compute good packing, unpack by applying heuristic in reverse, compute swap operations

How?

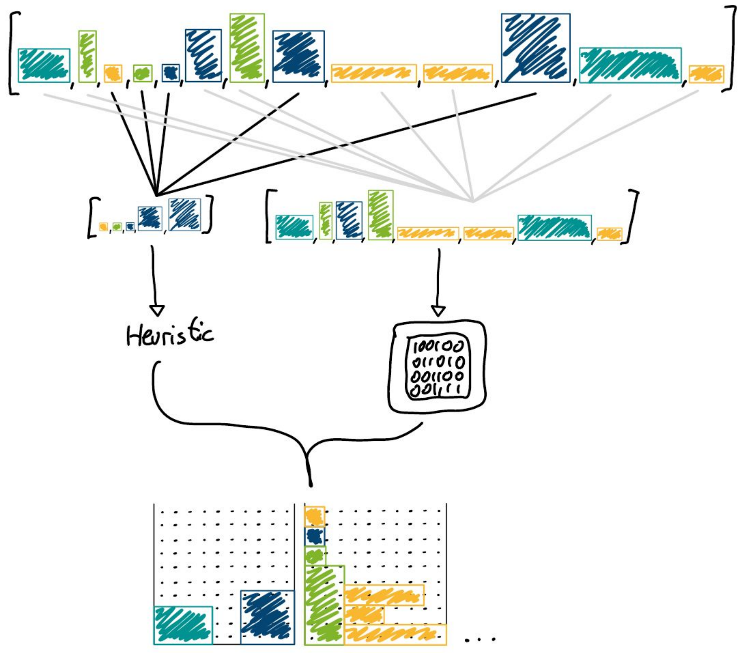
- Input: Sequence -> Seq2Seq Model
- Input: Histograms -> simple feed forward suffices?

Hybrid Approach

- Select some boxes for which an optimal packing is known or on which a heuristic is particularly good at

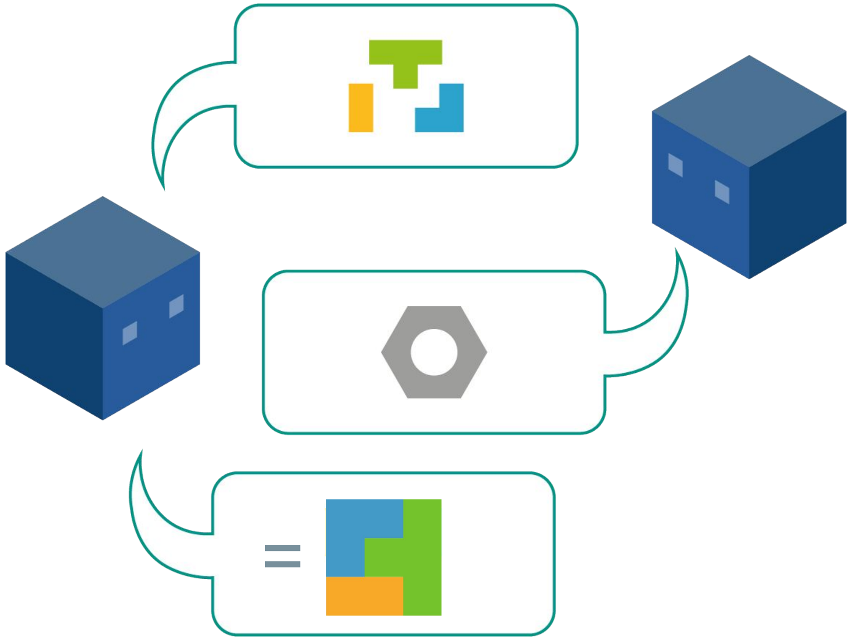
  ->eg pick all “Extreme” boxes, put them into an intelligent order, apply NF

- Use an ML approach to process the other boxes



Summary

- Supervised Pre-Training + Reinforced Fine-Tuning
- Representation
  - Sequence of boxes
  - Histograms
- Ideas
  - Learn to pick a bin and a position
  - Learn to pick a heuristic
  - Dense Prototypes
  - Re-ordering Sequences
  - Hybrid Approaches



Outline

1. Task definition 2D Bin Packing ✓
2. Heuristics and Data ✓
3. Starting Points for ML ✓
4. The Competition & Next Up

The Competition

- Work in teams! Approx. 5 members/group
- On the day of the submission we distribute unseen sets of boxes. You have to submit a packing within a given time frame.
  - The sizes of the sets will range from 42 - 10.000 boxes
  - The sets will follow different distributions

Certificate of Participation: Awarded for holding a presentation at the end of the week that presents your approach to the problem and/or what insights you gained on the problem -> independent from the competition

Let’s stay connected!

Share your #AutumnSchool experience with us by tagging @MLCompetence on **Twitter** and @ML2R | Maschinelles Lernen Rhein-Ruhr on **LinkedIn**!

Pictures taken in the context of the Autumn School can be sent to: autumnschool@ml2r.de

ML2R:	https://www.ml2r.de/en
	https://github.com/ML2R-center
	https://www.researchgate.net/project/Machine-Learning-Rhine-Ruhr-ML2R