Technische Universität Braunschweig

Online Algorithms - Tutorial 05

Summer term 2022, 04. July 2022

Plan for the upcoming tutorials

- 11th July (Michael & Peter): Preparation for the exams
  - Introduction to Online algorithms (basic definitions)
  - Ski Rental
  - Paging: algorithms, competitive-ratios, proof ideas
  - Randomized OA for Paging
- 18th July (Peter): Homeworks & Preparation for the exams
  - Presentation of the last exercise sheet
  - File migration
  - To be announced
- 25th July: Preparation for the exams
  - To be announced

Adversary Models

Randomized online algorithm adversaries

Oblivious adversary

- Adversary knows A
- Adversary generates σ and optimal offline solution $OPT(\sigma)$, A runs on σ, generating $A(\sigma)$

$$c = \sup_{\sigma} \frac{\mathbb{E}[A(\sigma)]}{OPT(\sigma)}$$

Adaptive online adversary

- Adversary knows A
- Adversary generate σ_i in response to a previous output (state of A known)
- Next input or end possible

Which of these adversaries is stronger?

Randomized File Migration

Claim from the lecture:

Any (deterministic or randomized) online file migration algorithm has a competitive ratio of at least 3.

How did the proof go? What was the input sequence?

Always request the file where the algorithm does not have it.

Can an oblivious adversary do that?

No! Can an adaptive online algorithm do that?

Against an **adaptive online adversary,** any randomized online file migration algorithm has a competitive ratio of at least 3.

Randomized online algorithm adversaries

Oblivious adversary

- Adversary knows A
- Adversary generates σ and optimal offline solution $OPT(\sigma)$, A runs on σ, generating $A(\sigma)$

Adaptive online adversary

- Adversary knows A
- Adversary generate σ_i in response to a previous output (state of A known)

Adaptive offline adversary

- Adversary knows A
- Adversary can generate σ_i in response **and** knows everything, even the random number generator

What adversary model is the one that we use for deterministic algorithms?

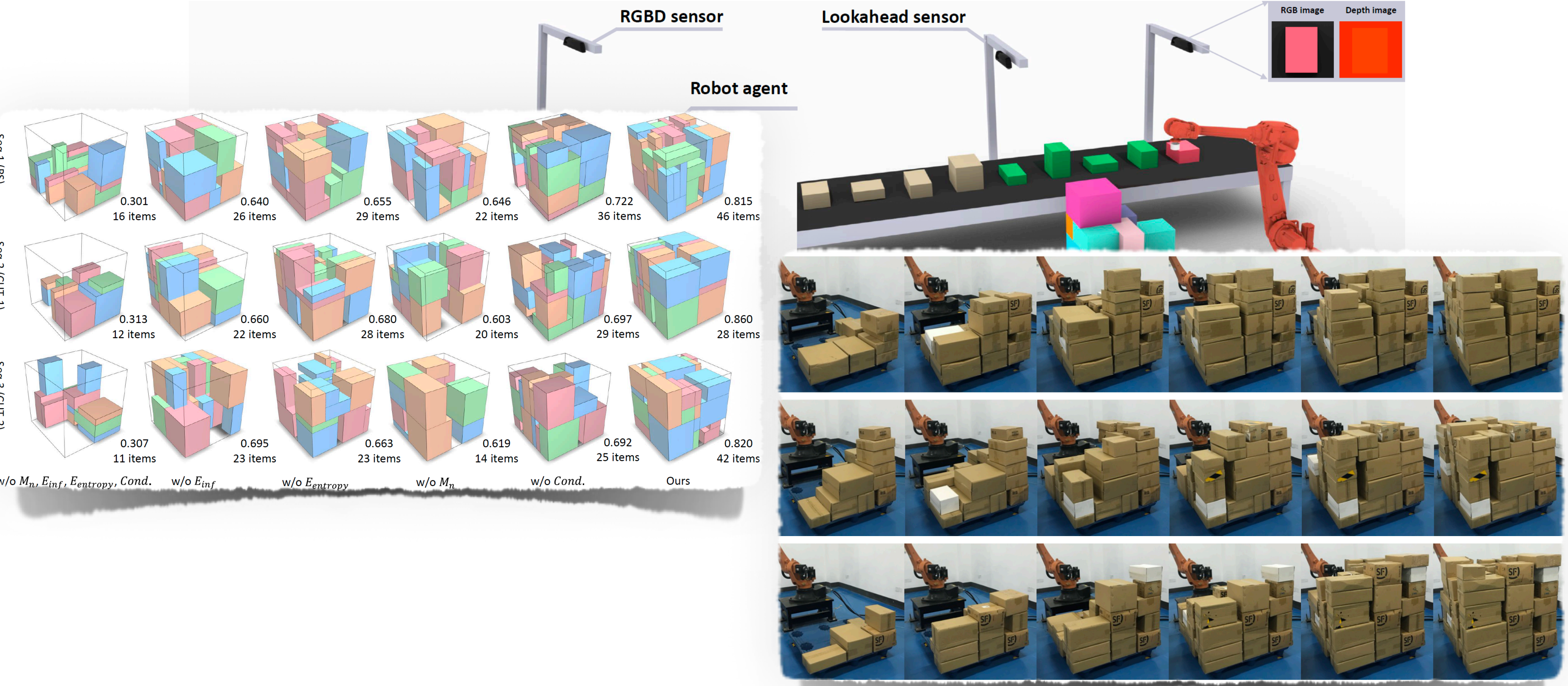
Bin Packing

Bin Packing

- Any Fit algorithms are 1.7-competitive
- But there are better algorithms!
- Idea: Categorize items by size (**Harmonic**)
    - Categories $(\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots$
    - Next Fit within categories
- With sufficient categories, better than 1.7

- Better algorithms: More categories, more complex packing
- Currently: 1.57829... (**Advanced Harmonic**) (2018)
- Current best lower bound: 1.54278... (2020)

Source: „Learning Practically Feasible Policies for Online 3D Bin Packing" SCIENCE CHINA Information Sciences 2022



Online Scheduling

Online Scheduling

Another classic problem: Distribute jobs on machines

Different scenarios:

- Precedence constraints
- Preemption
- Machine faults
- Unsure job running time
- Different machines (speed, possible jobs)

Different objectives:

- Minimum makespan
- Minimum waiting time, equal load, . . .

Online Scheduling

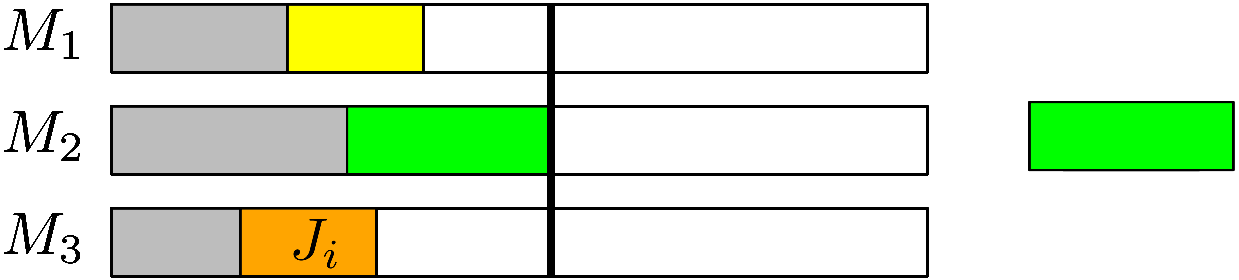
Our variant: Minimum Makespan

- m identical machines, n jobs, running times $t(J_i)$
- Minimize makespan
- Assign job J_i to some machine before getting J_{i+1}

Idea for simple online algorithm?

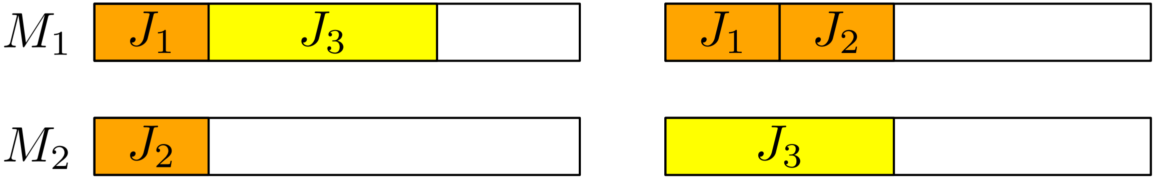
Online Scheduling - List Scheduling

Always put the next job on the machine with least load



Online Scheduling - List Scheduling

Competitive ration for $m = 2$:



$t(J_3) = 2t(J_1) = 2t(J_2)$ OPT

Competitive ratio: $\frac{3}{2}$

Arbitrary m: $m(m-1)$ jobs with time 1, 1 job with time m

List Scheduling: $(m-1)+m$ OPT: m $c \geq 2 - \frac{1}{m}$

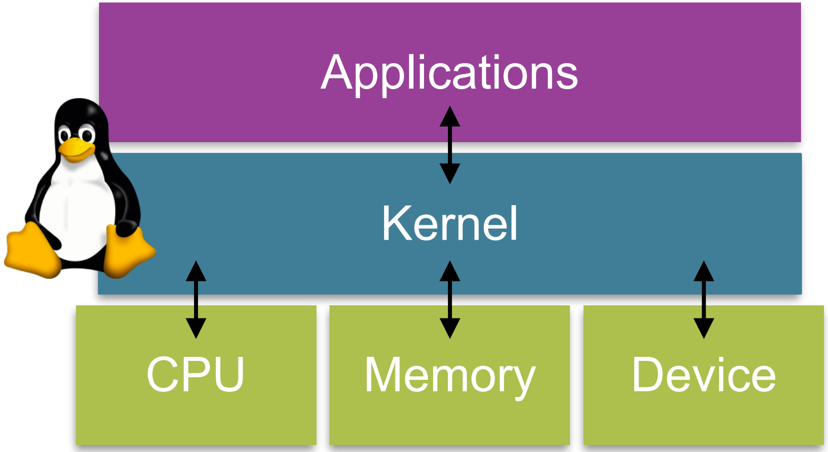
Online Scheduling - Randomization

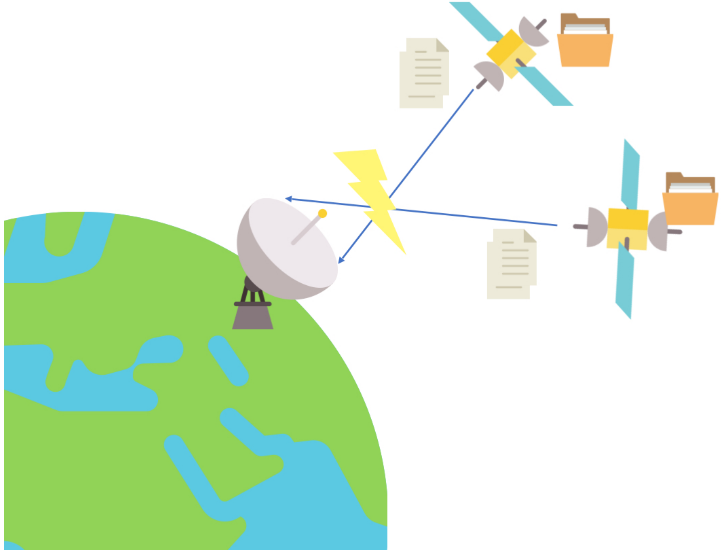
What about randomized online algorithms?

Theorem 5.1

Against an **oblivious adversary**, any randomized online list scheduling algorithm has a competitive ratio of at least $\frac{4}{3}$

Online Scheduling - Applications





Dynamic Data Structures

Dynamic Data Structures

- We are keeping collections of items in a data structure
    - Array/List
    - Binary search tree
    - Hash table
- DDS: Data structure may change on search requests

Goal: Minimize the total cost of all requests

Applications: Data compression, optimized search data structure

Important data structure: Splay trees

Minimum Spanning Tree

Minimum Spanning Tree

- Minimum weight connected subset of edges of a graph G that connects all vertices
- Offline: Prim's , Kruskal

Idea for simple online algorithm?

Prim's strategy: Always connect an incoming vertex to the closest one.

Applications:

- Solving online TSP (Christofides approximation)
- Handwriting recognition
- Cluster analysis: cluster points in the plane
- ....