Computing push plans for disk-shaped robots and dynamic labelings for moving points

Dirk Gerrits
“I do Computer Science.”
“I do Computer Science.”

“Cool!”
“I do Computer Science.”

“Cool! So I have this problem with my computer...”
"I do Computer Science."

"Cool! So I have this problem with my computer..."

**Not:** humans solving computer problems

**But:** computers solving human problems
"I do Computer Science."

"Cool! So I have this problem with my computer..."

**Not:** humans solving computer problems

**But:** computers solving human problems

- What problems can they solve?
- How efficiently can they solve them?
Problem 1: Pushing Objects with Robots
Problem 1: Pushing Objects with Robots

1. build a strong robot (easy)
Problem 1: Pushing Objects with Robots

1. build a strong robot (easy)
2. make it find the right pushing motion (hard)
Problem 1: Pushing Objects with Robots

1. build a strong robot (easy)
2. make it find the right pushing motion (hard)

“How hard could it be?
A child can do it!”
Baby Steps in Pushing
Baby Steps in Pushing

more complex shapes
Baby Steps in Pushing

more complex shapes
Baby Steps in Pushing

more complex shapes

more complex pushing tasks
Baby Steps in Pushing

more complex shapes

more complex pushing tasks
Intermezzo: Air-Traffic Control

≈100,000 flights/day, kept safe by air-traffic controllers.
Intermezzo: Air-Traffic Control

Airplanes $\rightarrow$ moving points + information labels

Watch for potential crashes
Intermezzo: Air-Traffic Control

Airplanes → moving points + information labels

Watch for potential crashes

... while moving labels around for readability!
Problem 2: Labeling Moving Points

We want to:

- **minimize** overlap,
  labeling all points
Problem 2: Labeling Moving Points

We want to:
- minimize overlap, labeling all points

We don’t want to:
- avoid overlap by only labeling some points
Problem 2: Labeling Moving Points

We want to:
- **minimize** overlap, labeling all points

We don’t want to:
- **avoid** overlap by only labeling some points
- relabel abruptly
Problem 2: Labeling Moving Points

We want to:
- **minimize** overlap, labeling all points
- move labels smoothly

We don’t want to:
- **avoid** overlap by only labeling some points
- relabel abruptly