Self-Stabilizing Task Allocation In Spite of Noise

Anna Dornhaus, Nancy Lynch, Frederik Mallmann-Trenn,
Dominik Pajak,Tsvetomira Radeva

(a) Foraging

(b) Brood care

(c) Farming aphids

(d) Cultivating fungi

Task Allocation

- We care about task allocation, i.e., assignment of ants to tasks
- Queen has no real power
- Decision must be made in distributed fashion
- From time to time the assignment needs to be adapted:
  - Say that there are currently 1000 foragers and an anteater eats half of them

  - Who replaces them?

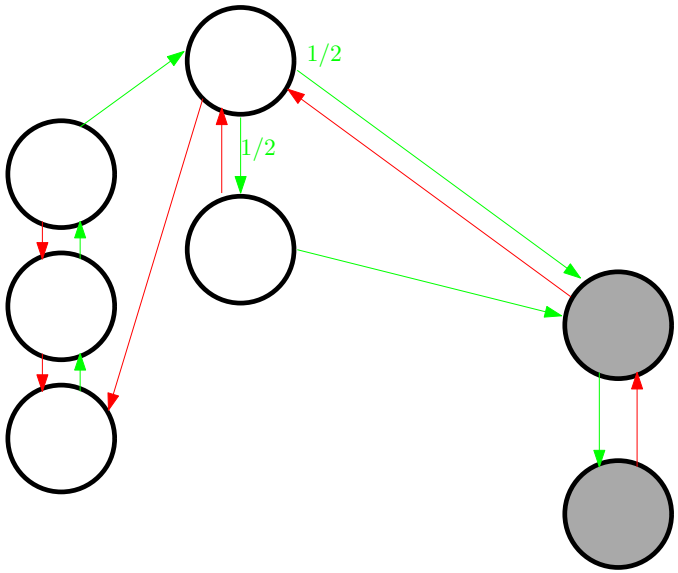
Model: Cornejo et al. 2014

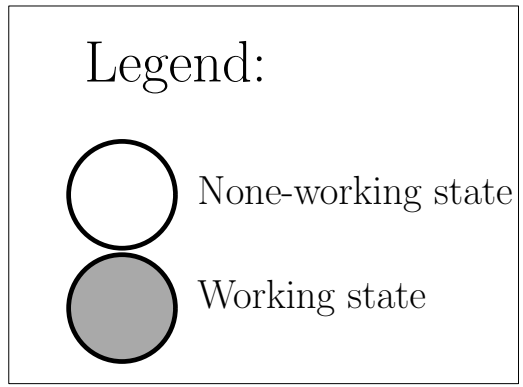
- Each task j has a demand $\mathrm{demand}^{(j)}$
- Ants act in synchronous rounds
- Number of ants working on task in round t: $\mathrm{load}_j^{(t)}$
- Ants receive binary feedback for each task:

$$F_j^{(t)} = \begin{cases} \texttt{overload} & \text{if } \mathrm{load}_j^{(t-1)} > \mathrm{demand}^{(j)} \\ \texttt{lack} & \text{otherwise} \end{cases}$$

- Based on feedback ants can choose to join a task or leave a task
- Synchronicity models the delay of information and movement

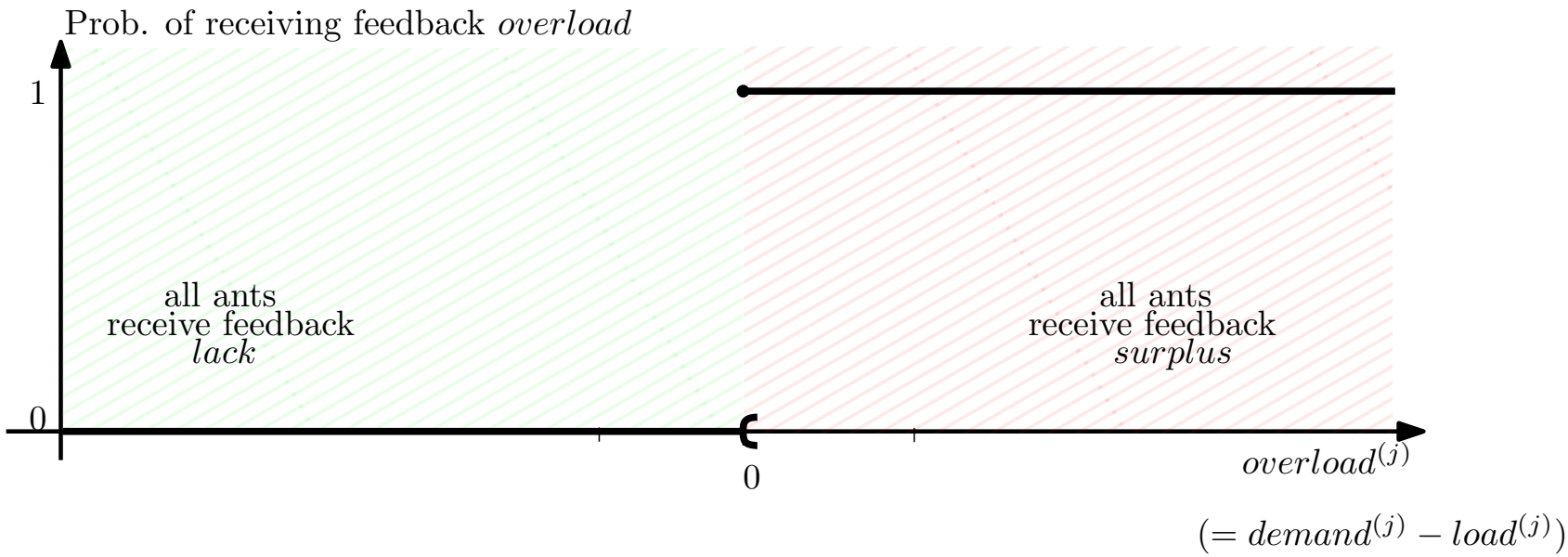
Binary Feedback—Cornejo et al. 2014



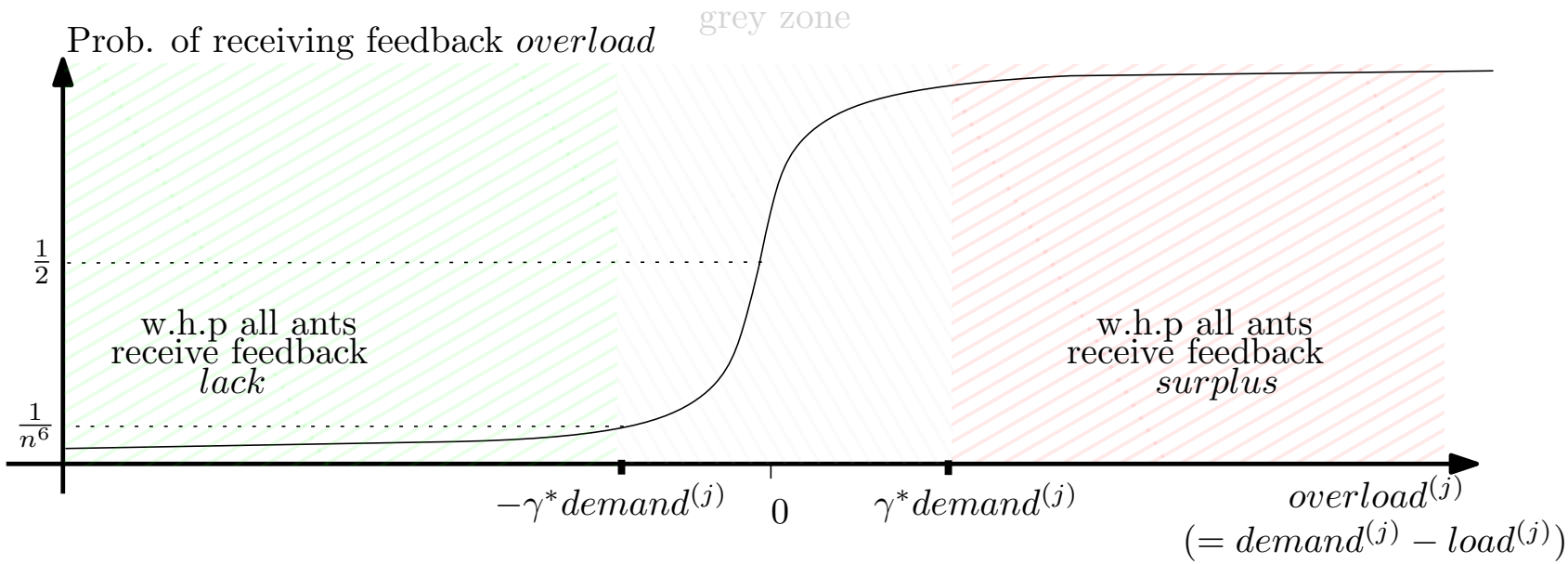


- A simple state-machine does the job.
- Transitions based on feedback: overload/lack
- Thm: For every $t = \Omega(\log n)$ rounds
  $|\,\mathrm{load}_j^{(t)} - \mathrm{demand}^{(j)}\,| \leq 1$ (i.e., optimal load ± 1)

Is this feedback realistic? —Cornejo et al. 2014

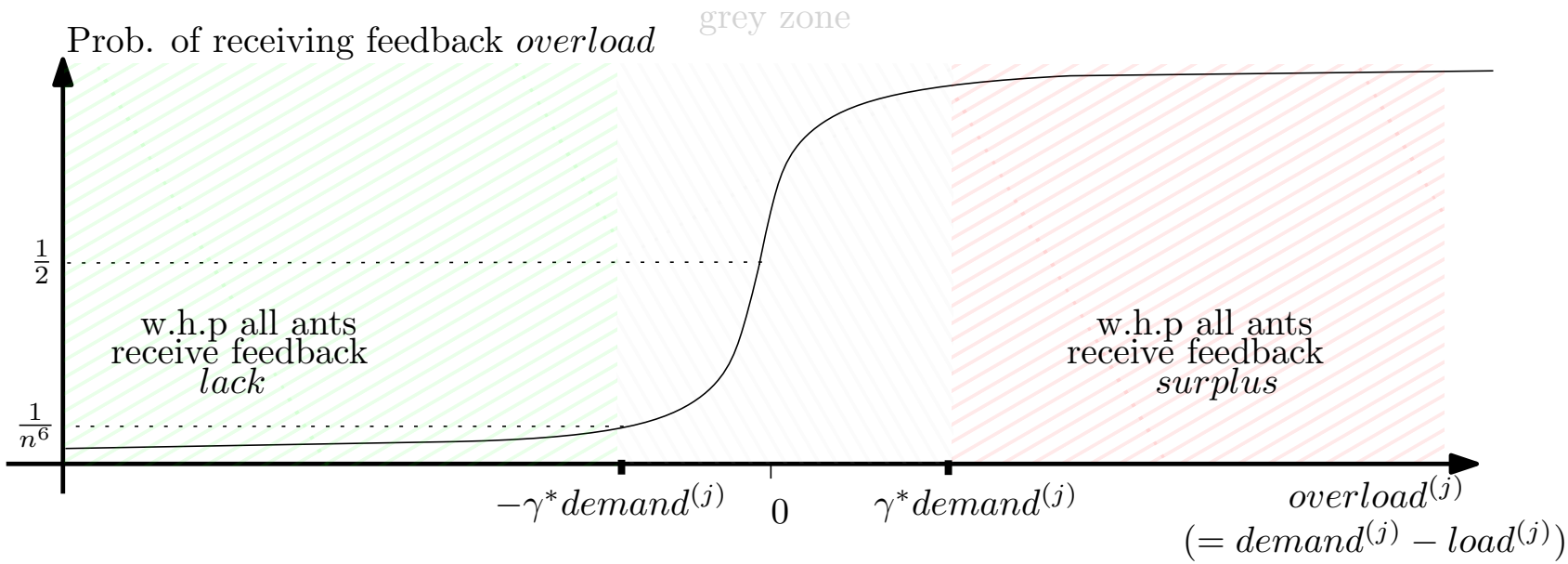


Noisy feedback! — Our model



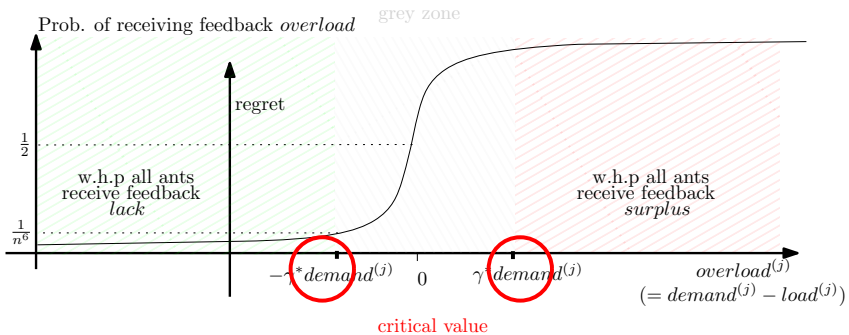
- critical value $\gamma^* \, \text{demand}^{(j)}$

Noisy feedback! — Our model



- critical value $\gamma^* \, \mathrm{demand}^{(j)}$
- sigmoid could be replaced by any function that
  - monotonic increasing
  - bounded away from 1
  - uncertainty maximized at 0
  - exponential decay

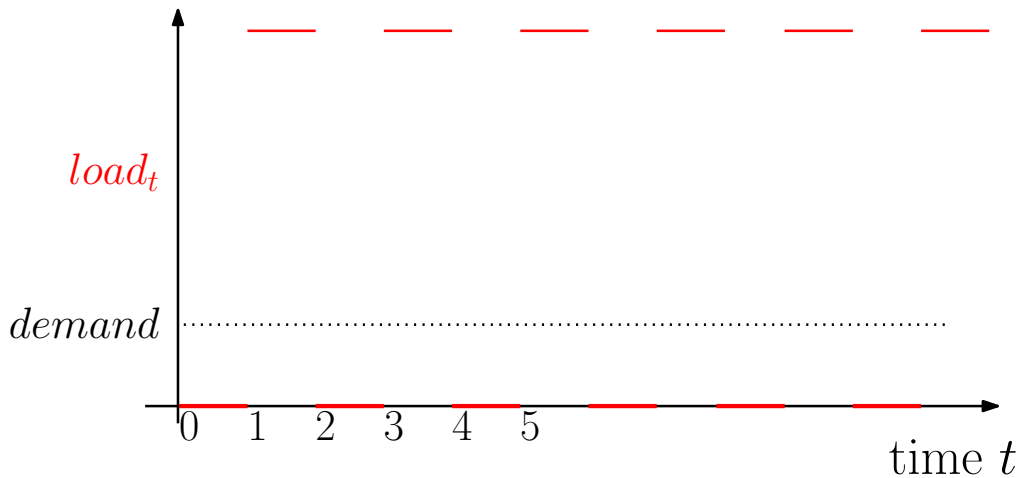
How can we use this feedback?



- As long we are in the green or red area all ants have the correct feedback
- If we are in the red area, then some ants should leave
- If we are in the green area, then some ants should join

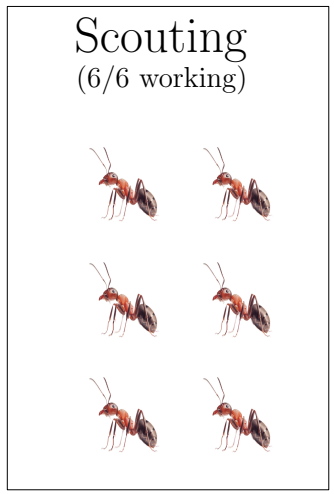
Challenge 1: Synchronicity

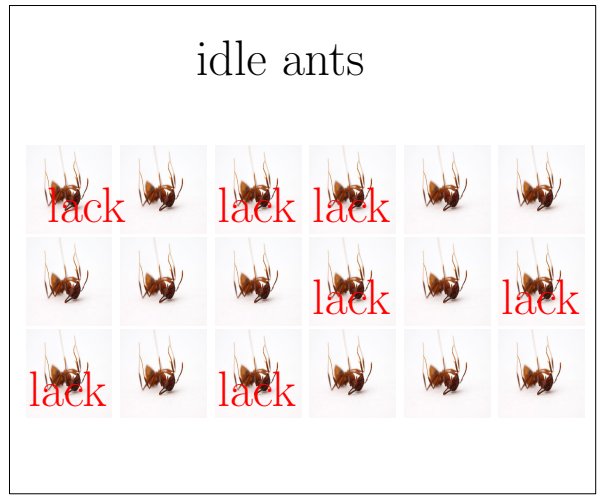
- All ants decide at the same time
- If ants simply leave when a task if overloaded and join if it's underloaded, then we get oscillations



Challenge 2: What to do in the grey zone?

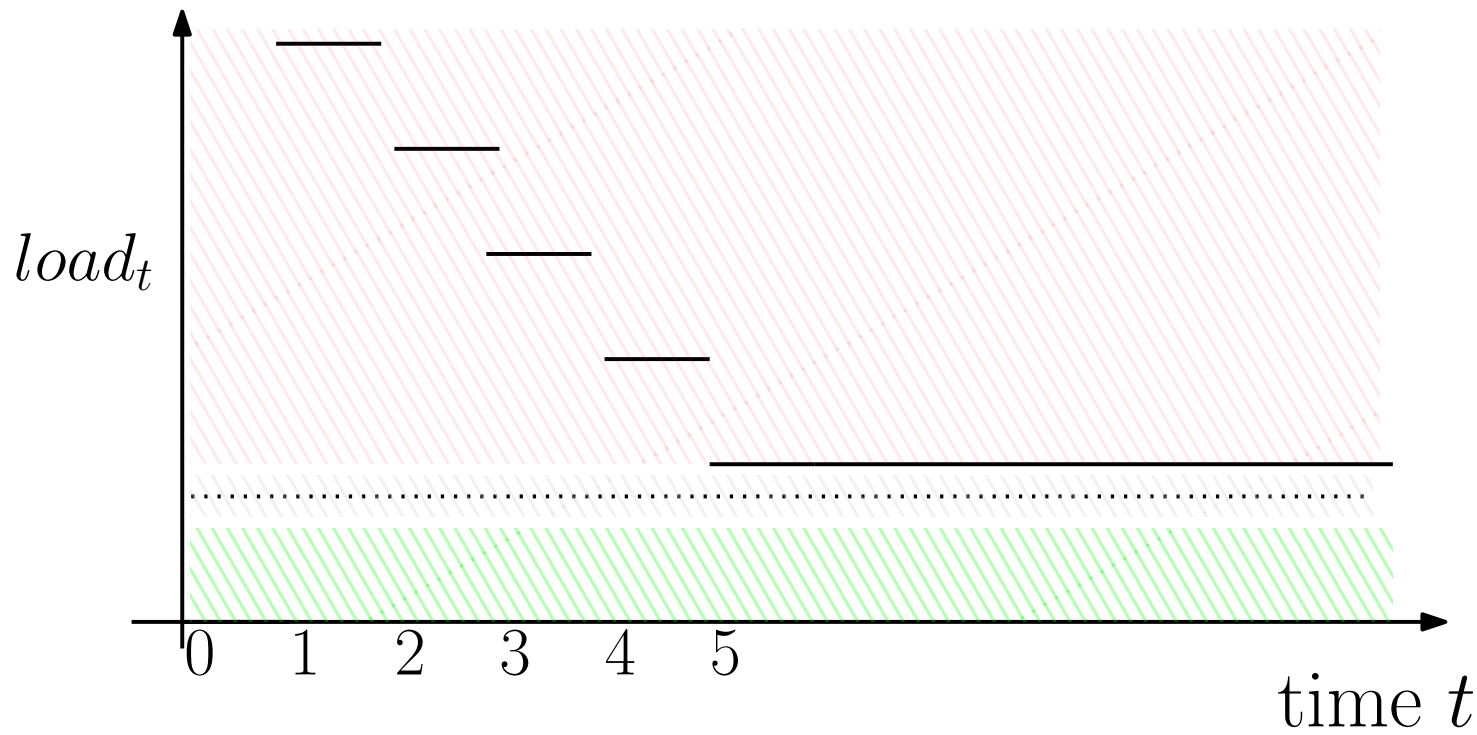
- Say $demand = 6$ and $load = 6$ ants currently working on the task.
- Then half of the idle ants receive feedback `lack`
- What should each ant do?
- Those ants can distinguish from the case where 0 ants work





Tasks performed by ants

- If in red area, slowly decrease!
- Avoid grey area!
- If in green, join if idle

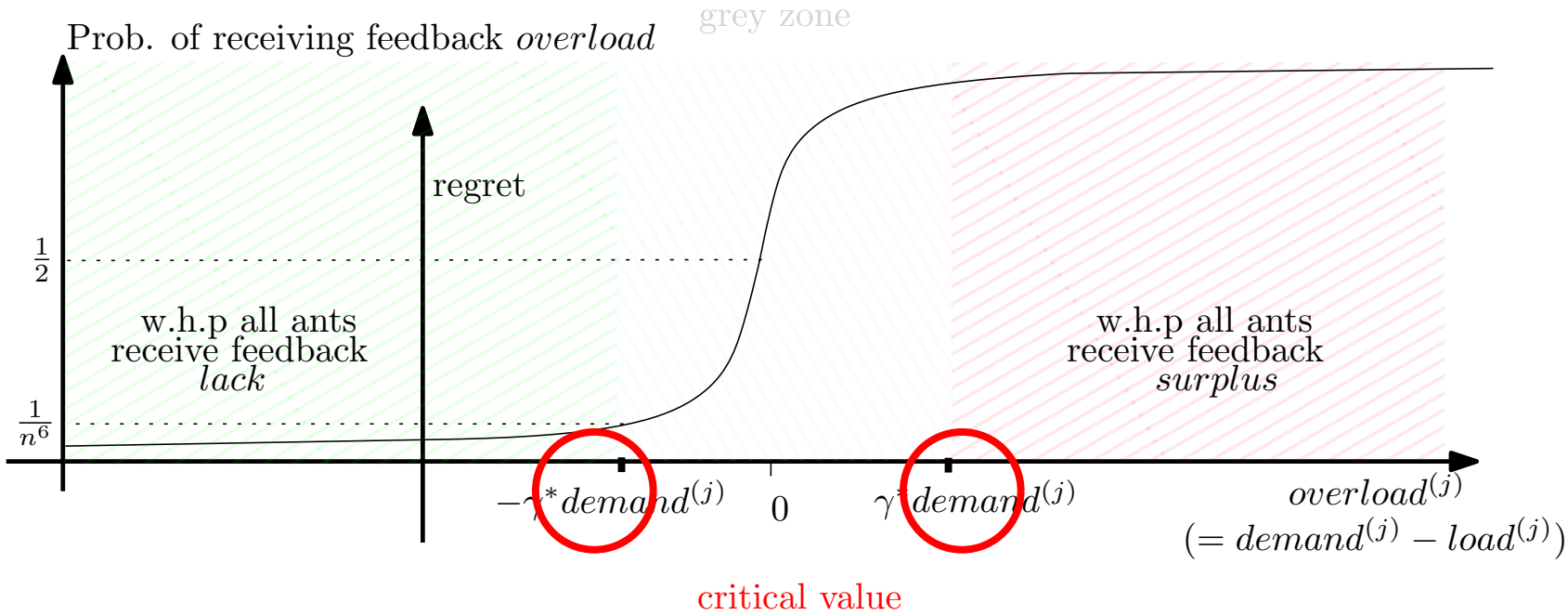


The algorithm

Intuition:

- Making a decision based only on the current load (sample) is tricky
- We need probabilistic choices
- Reduce the load carefully
- Avoid the grey zone ...

Assumptions



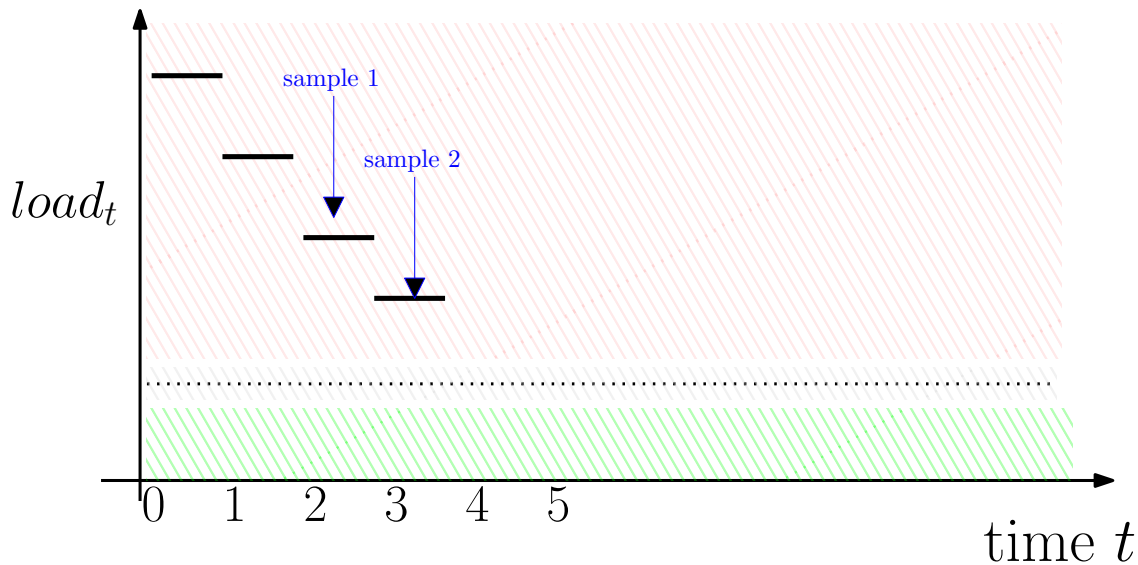
- We assume the ants have a rough upper bound γ with $\gamma^* \leq \gamma$.
- All ants are in the same cycle consisting of 2 rounds, *e.g.*, day and night
- Not all ants are required to work (slack) $\sum_j \text{demand}^{(j)} \leq 0.9n$

The algorithm

- Combine round $2i$ and $2i+1$ into phase i
- Always take two samples:
- Sample 1 at $\mathrm{load}_j^{(t)}$
- Sample 2 at $(1-3\gamma^*)\cdot \mathrm{load}_j^{(t)}$ (each ant flips a coin)
- If both samples indicate overload, then leave with small probability

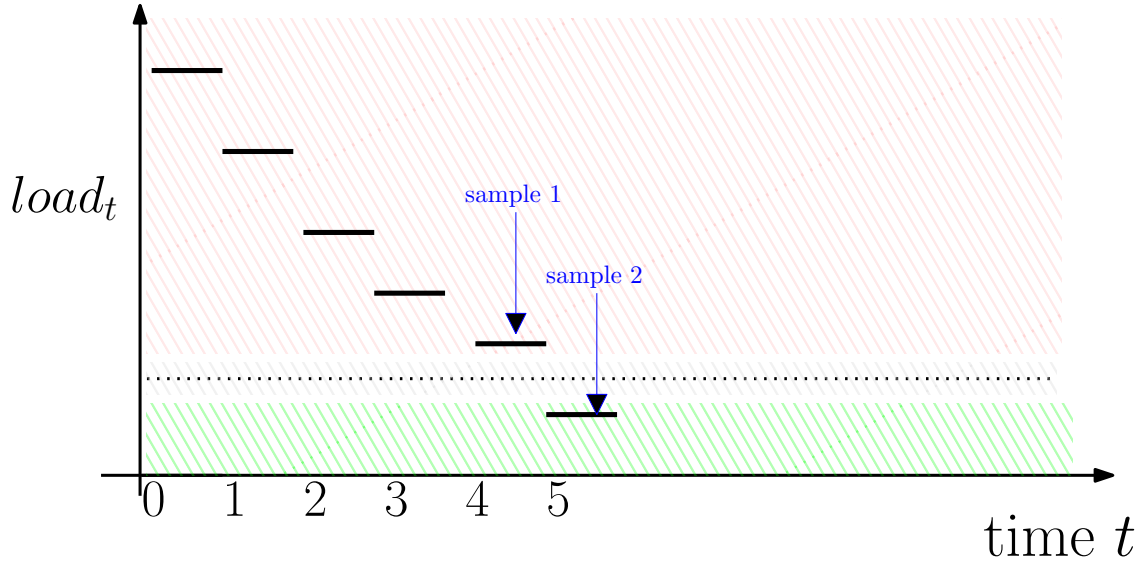
The algorithm

- Combine round $2i$ and $2i+1$ into phase i
- Always take two samples:
- Sample 1 at $\text{load}_j^{(t)}$
- Sample 2 at $(1-3\gamma^*)\cdot \text{load}_j^{(t)}$ (each ant flips a coin)
- If both samples indicate overload, then leave with small probability



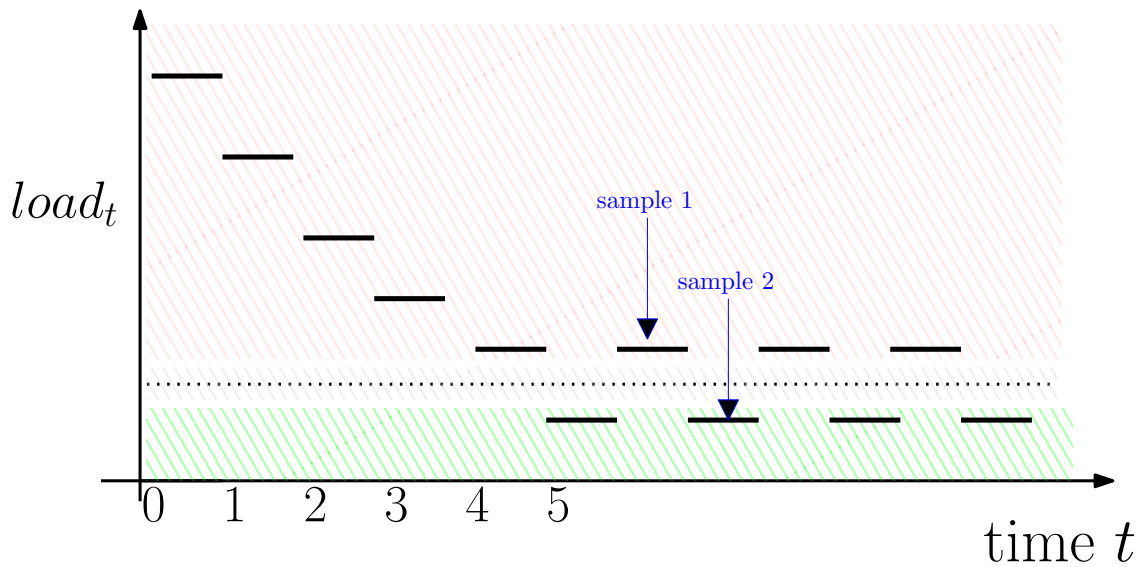
The algorithm

- Always take two samples—combine two rounds into one phase
- Sample 1 at $\text{load}_j^{(t)}$
- Sample 2 at $(1 - 3\gamma^*) \cdot \text{load}_j^{(t)}$
- If both samples indicate overload, then leave with small probability



The algorithm

- Always take two samples:
- Sample 1 at $\mathrm{load}_j^{(t)}$
- Sample 2 at $(1 - 3\gamma^*) \cdot \mathrm{load}_j^{(t)}$
- If both samples indicate overload, then leave with small probability



Performance Metric

- Regret $r(t)$, defined for time $t \in \mathbb{R}$ as:

$$r(t) = \sum_{j \in [k]} |\, \text{demand}^{(j)} - \text{load}_t^{(j)} \,| = \sum_{j \in [k]} |\, \text{deficit}_t^{(j)} \,|.$$

  Furthermore, we define the (total) regret up to time t as:

$$R(t) = \sum_{i \leq t} r(i).$$

Performance Metric

- Regret $r(t)$, defined for time $t \in \mathbb{R}$ as:

$$r(t) = \sum_{j \in [k]} |\, \text{demand}^{(j)} - \text{load}_t^{(j)} \,| = \sum_{j \in [k]} |\, \text{deficit}_t^{(j)} \,|.$$

  Furthermore, we define the (total) regret up to time t as:

$$R(t) = \sum_{i \leq t} r(i).$$

- Penalize under- and overload equally

Main Result

Theorem

Consider an arbitrary initial allocation at time 0*. Fix an arbitrary* $t \in \mathbb{N}$*. Algorithm Ant with learning rate* $\gamma \geq \gamma^*$ *has w.h.p. a total regret during the first* t *rounds that is bounded by*

$$R(t) \leq c\frac{nk}{\gamma} + 5\gamma \sum_{j \in [k]} \text{demand}^{(j)} \cdot t,$$

for some constant c*.*

More Result

Recall our main upper bound

$$R(t) \leq c\frac{nk}{\gamma} + 5\gamma \sum_{j\in[k]} \text{demand}^{(j)} \cdot t,$$

- Lower bound for any constant memory algorithm

$$R(t) = n/2 + \Omega\left(\gamma^* \sum_{j\in[k]} \text{demand}^{(j)} \cdot t\right)$$

More Result

Recall our main upper bound

$$R(t) \leq c\frac{nk}{\gamma} + 5\gamma \sum_{j\in[k]} \text{demand}^{(j)} \cdot t,$$

- Lower bound for any constant memory algorithm

$$R(t) = n/2 + \Omega\left(\gamma^* \sum_{j\in[k]} \text{demand}^{(j)} \cdot t\right)$$

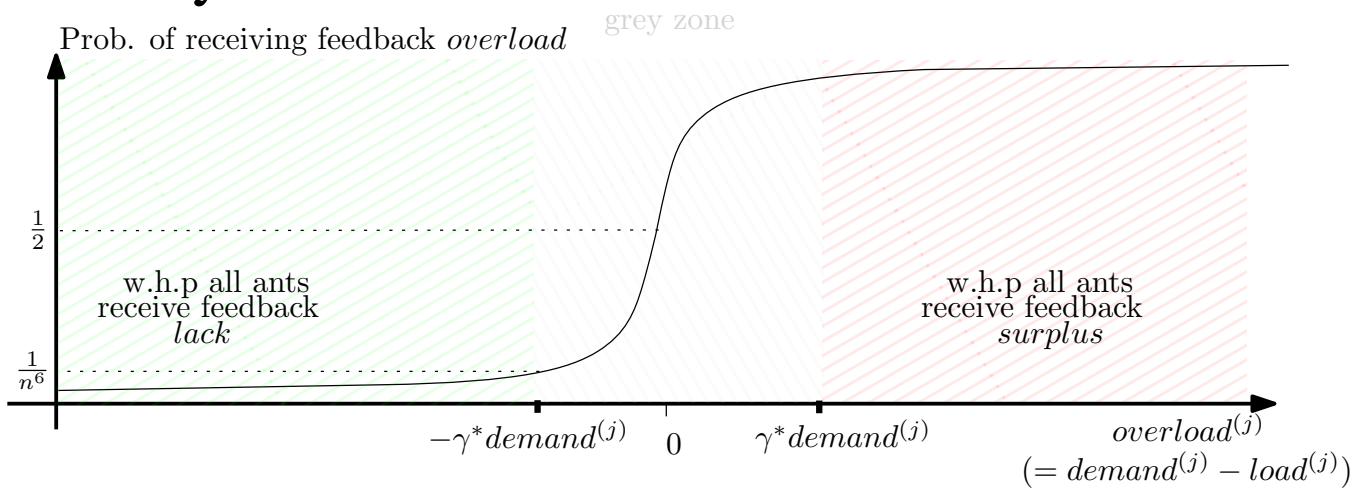
- Improved upper bound: Using more memory and synchronized rounds, we can decrease the constant in front of $\left(\gamma \sum_{j\in[k]} \text{demand}^{(j)}\right)$

Lower Bound

- Feedback in the grey area close to 0 is very fuzzy
- If ants stay for a long time in the grey area, then it's very likely that many ants receive the incorrect feedback for a number of steps
- This number exceeds the size of the state space
- In other they will join (otherwise they would never join)

Better Upper Bound

- How to beat the lower bound (less regret) if we allow more memory?



- Solution: Take more samples (longer phases)
- Less likely that each sample is correct (since samples are closer to 0 deficit)
- However, longer rounds allow us to amplify the probability to compensate completely for this
- From here on the analysis is equal to before

Open Problems

- Do we need odd/even rounds?
- Can we design an algorithm that will work for relaxed synchronicity? (random subset of ants chosen etc)
- How much does communication help?

Thank you! Questions?