Division of Labor in Ant Colonies

Alex Cornejo, Anna Dornhaus,
Nancy Lynch and Radhika Nagpal

October 13, 2014

What is division of labor?

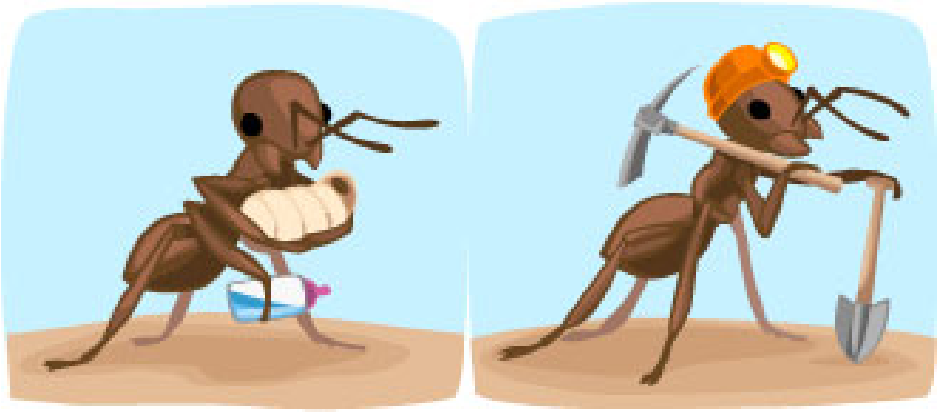
- Division of labor is the process by which individual ants in a colony decide which task to perform to ensure the survival of the colony.

What is division of labor?

- Division of labor is the process by which individual ants in a colony decide which task to perform to ensure the survival of the colony.

What is division of labor?

- Division of labor is the process by which individual ants in a colony decide which task to perform to ensure the survival of the colony.



What is division of labor?

- Division of labor is the process by which individual ants in a colony decide which task to perform to ensure the survival of the colony.

What is division of labor? (cont.)

- Much of the existing work focuses on exploring the correlation between temporal and morphological variations and worker behavior.

What is division of labor? (cont.)

- Much of the existing work focuses on exploring the correlation between temporal and morphological variations and worker behavior.

What is division of labor? (cont.)

- Much of the existing work focuses on exploring the correlation between temporal and morphological variations and worker behavior.

Contributions

- A very general mathematical formulation for the phenomenon of division of labor in ant colonies.

Contributions

- A very general mathematical formulation for the phenomenon of division of labor in ant colonies.
- A distributed algorithm that quickly reaches a near optimal task allocation and imposing only minimal assumptions on the capabilities of individual ants.

A model for division of labor

A model for division of labor

- A set A of ants.

A model for division of labor

- A set A of ants.
- A set T of tasks.

A model for division of labor

- A set A of ants.
- A set T of tasks.
- For task $\tau \in T$, ant $a \in A$ and time $t \in \mathbb{R}_{\geq 0}$:

A model for division of labor

- A set A of ants.
- A set T of tasks.
- For task $\tau \in T$, ant $a \in A$ and time $t \in \mathbb{R}_{\geq 0}$:
  - $d(\tau, t)$ is the total energy demand for task τ at time t.

A model for division of labor

- A set A of ants.
- A set T of tasks.
- For task $\tau \in T$, ant $a \in A$ and time $t \in \mathbb{R}_{\geq 0}$:
  - $d(\tau, t)$ is the total energy demand for task τ at time t.
  - $e(\tau, a, t)$ is the energy that can be supplied by ant a if engaged at task τ at time t.

Task Assignment

Task Assignment

- A task assignment is a function $y : A \times \mathbb{R}_{\geq 0} \to T \cup \{\perp\}$.

Task Assignment

- A task assignment is a function $y : A \times \mathbb{R}_{\geq 0} \to T \cup \{\perp\}$.
- Given y we define $Y(\tau, t)$ as the set of ants assigned to task τ at time t, and $I(t)$ as the set of ants assigned to no task at time t.

$$Y(\tau, t) = \{a \in A : y(a, t) = \tau\}$$
$$I(t) = \{a \in A : y(a, t) = \perp\}$$

Task Assignment

- A task assignment is a function $y : A \times \mathbb{R}_{\geq 0} \to T \cup \{\perp\}$.
- Given y we define $Y(\tau, t)$ as the set of ants assigned to task τ at time t, and $I(t)$ as the set of ants assigned to no task at time t.

$$Y(\tau, t) = \{a \in A : y(a, t) = \tau\}$$
$$I(t) = \{a \in A : y(a, t) = \perp\}$$

- Observe that by definition $A = Y(\tau, t) \cup I(t)$.

What is a good task allocation?

What is a good task allocation?

- $w(\tau, t) = \sum_{a \in Y(\tau,t)} e(\tau, a, t)$ is the energy supplied to task τ at time t.

What is a good task allocation?

- $w(\tau, t) = \sum_{a \in Y(\tau,t)} e(\tau, a, t)$ is the energy supplied to task τ at time t.
- $q(\tau, t) = d(\tau, t) - w(\tau, t)$, if negative represents a surplus of energy at task τ, if positive represents a deficit of energy at task τ, and if zero then the task τ is in equilibrium.

What is a good task allocation?

- $w(\tau, t) = \sum_{a \in Y(\tau,t)} e(\tau, a, t)$ is the energy supplied to task τ at time t.
- $q(\tau, t) = d(\tau, t) - w(\tau, t)$, if negative represents a surplus of energy at task τ, if positive represents a deficit of energy at task τ, and if zero then the task τ is in equilibrium.
- An *optimal* task assignment is one that minimizes $\sum_{\tau \in T} q(\tau, t)^2$

Restricted System Model

Restricted System Model

- We assume $e(\tau, a, t) = c$ for all $\tau \in T$, every ant $a \in A$, and every time $t \in \mathbb{R}_{\geq 0}$.

Restricted System Model

- We assume $e(\tau, a, t) = c$ for all $\tau \in T$, every ant $a \in A$, and every time $t \in \mathbb{R}_{\geq 0}$.

- We consider synchronous model where time proceeds in lock-step rounds $1, 2, \ldots$, and for the duration of each round $i \in \mathbb{N}$ each ant $a \in A$ works at task $y(a, i)$.

Capabilities of Individuals

Capabilities of Individuals

- Recall that $q(\tau, i) = d(\tau, i) - w(\tau, i)$ is the difference between the energy demand for task τ at round i and the energy supplied for task τ at round i.

Capabilities of Individuals

- Recall that $q(\tau, i) = d(\tau, i) - w(\tau, i)$ is the difference between the energy demand for task τ at round i and the energy supplied for task τ at round i.
- For each task $\tau \in T$ ants can sense only a binary feedback function $f(\tau, i)$ where:

$$f(\tau, i) = \begin{cases} +1 & q(\tau, i) \geq 0, \\ -1 & q(\tau, i) < 0. \end{cases}$$

Algorithm Idea

Algorithm Idea

- Intuition: Given sufficient memory, ants could use a randomized binary-search-like strategy, guided by the function f, to reach the optimal division of labor.

Algorithm Idea

- Intuition: Given sufficient memory, ants could use a randomized binary-search-like strategy, guided by the function f, to reach the optimal division of labor.
- Idea 1: Offload the burden of memory from the individuals to the colony.

Algorithm Idea

- Intuition: Given sufficient memory, ants could use a randomized binary-search-like strategy, guided by the function f, to reach the optimal division of labor.
- Idea 1: Offload the burden of memory from the individuals to the colony.
- Idea 2: Let ants use a response-threshold strategy to pick tasks, and allow them to specialize.

Algorithm Details

- Each ant stores a task $currentTask \in T \cup \{\perp\}$ and a potential table $\varrho[\tau]$.

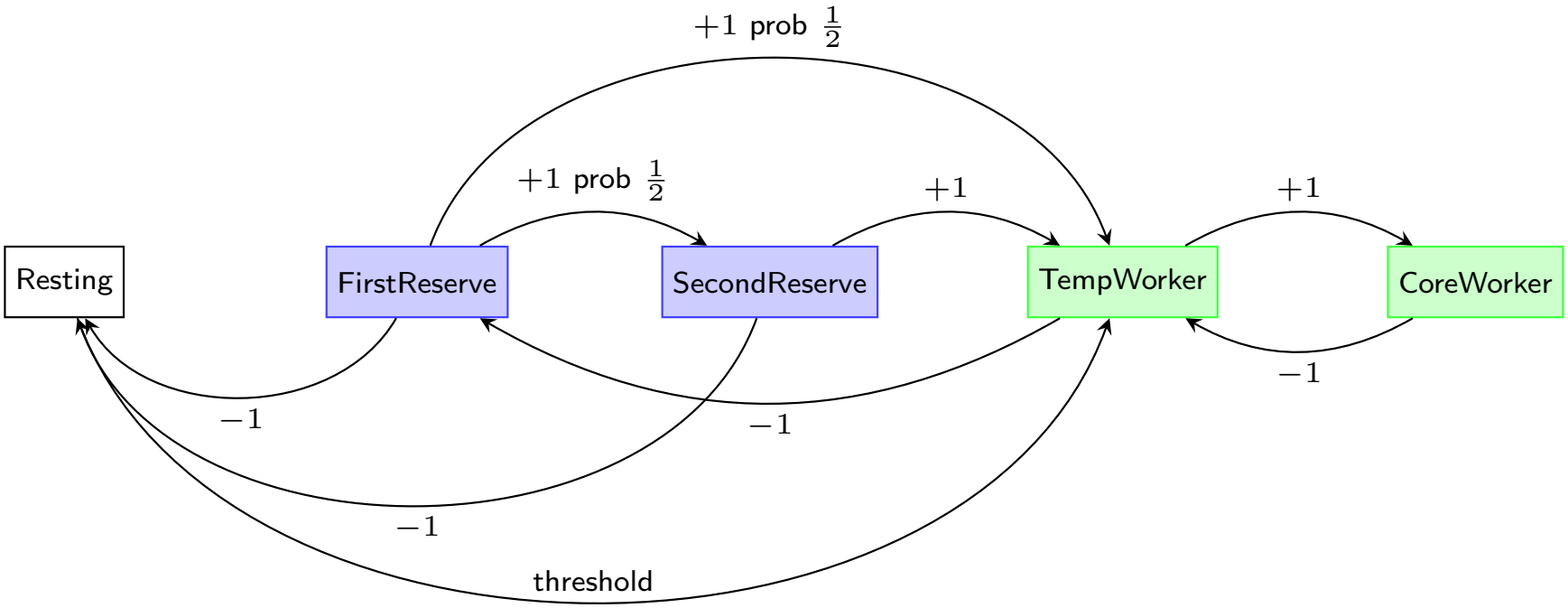
Algorithm Details

- Each ant stores a task $currentTask \in T \cup \{\perp\}$ and a potential table $\varrho[\tau]$.
- Initially ants start in the $\textsc{Resting}$ state with $currentTask = \perp$ and a potential of zero for every task $\forall \tau \in T, \varrho[\tau] = 0$.

Algorithm Details

- Each ant stores a task $currentTask \in T \cup \{\perp\}$ and a potential table $\varrho[\tau]$.
- Initially ants start in the $\textsc{Resting}$ state with $currentTask = \perp$ and a potential of zero for every task $\forall \tau \in T, \varrho[\tau] = 0$.
- Ants can be in one of the five states $\textsc{Resting}$, $\textsc{FirstReserve}$, $\textsc{SecondReserve}$, $\textsc{TempWorker}$ and $\textsc{CoreWorker}$.

Simplified State Machine



Resting State

$$\forall \tau \in T, \varrho[\tau] \leftarrow \begin{cases} 0 & \text{if } f(\tau, i) < 0 \\ \min(\varrho[\tau] + 1, 3) & \text{if } f(\tau, i) > 0 \end{cases}$$

Resting State

$$\forall \tau \in T, \varrho[\tau] \leftarrow \begin{cases} 0 & \text{if } f(\tau, i) < 0 \\ \min(\varrho[\tau] + 1, 3) & \text{if } f(\tau, i) > 0 \end{cases}$$

$$\text{candidate} \leftarrow \{\tau \in T \mid \varrho[\tau] = 3\}$$

Resting State

$$\forall \tau \in T, \varrho[\tau] \leftarrow \begin{cases} 0 & \text{if } f(\tau, i) < 0 \\ \min(\varrho[\tau] + 1, 3) & \text{if } f(\tau, i) > 0 \end{cases}$$

candidate $\leftarrow \{\tau \in T \mid \varrho[\tau] = 3\}$

if candidate $\neq \varnothing$ **then**

Resting State

$$\forall \tau \in T, \varrho[\tau] \leftarrow \begin{cases} 0 & \text{if } f(\tau, i) < 0 \\ \min(\varrho[\tau] + 1, 3) & \text{if } f(\tau, i) > 0 \end{cases}$$

candidate $\leftarrow \{\tau \in T \mid \varrho[\tau] = 3\}$

if candidate $\neq \varnothing$ **then**

  with probability $\frac{1}{2}$ **do**

    $\forall \tau \in T, \varrho[\tau] \leftarrow 0$

    $currentTask \leftarrow$ random candidate task

    state $\leftarrow$ TempWorker

  end with

end if

Working States

case state $= \textsc{TempWorker}$

Working States

case state $= \textsc{TempWorker}$
  if $f(currentTask, i) < 0$ **then**
    state $\leftarrow \textsc{FirstReserve}$

Working States

case state $=$ $\textsc{TempWorker}$
  if $f(currentTask, i) < 0$ **then**
    state $\leftarrow$ $\textsc{FirstReserve}$
  else
    state $\leftarrow$ $\textsc{CoreWorker}$
  end if

Working States

case state $=$ $\textsc{TempWorker}$
  if $f(currentTask, i) < 0$ **then**
    state $\leftarrow$ $\textsc{FirstReserve}$
  else
    state $\leftarrow$ $\textsc{CoreWorker}$
  end if
case state $=$ $\textsc{CoreWorker}$

Working States

```
case state = TempWorker
    if f(currentTask, i) < 0 then
        state ← FirstReserve
    else
        state ← CoreWorker
    end if
case state = CoreWorker
    if f(currentTask, i) < 0 then
        state ← TempWorker
    end if
end case
```

Reserve States

$$\textbf{case}\ \mathsf{state} = \textsc{FirstReserve}$$

Reserve States

case state $=$ $\textsc{FirstReserve}$
  if $f(currentTask, i) < 0$ **then**
    state $\leftarrow$ $\textsc{Resting}$
  else

Reserve States

```
case state = FirstReserve
    if f(currentTask, i) < 0 then
        state ← Resting
    else
        with probability 1/2 do
            state ← TempWorker
        otherwise
            state ← SecondReserve
        end with
    end if
```

Reserve States

```
case state = FirstReserve
    if f(currentTask, i) < 0 then
        state ← Resting
    else
        with probability 1/2 do
            state ← TempWorker
        otherwise
            state ← SecondReserve
        end with
    end if
case state = SecondReserve
```

Reserve States

```
case state = FirstReserve
    if f(currentTask, i) < 0 then
        state ← Resting
    else
        with probability 1/2 do
            state ← TempWorker
        otherwise
            state ← SecondReserve
        end with
    end if
case state = SecondReserve
    if f(currentTask, i) < 0 then
        state ← Resting
```

Reserve States

```
case state = FirstReserve
    if f(currentTask, i) < 0 then
        state ← Resting
    else
        with probability 1/2 do
            state ← TempWorker
        otherwise
            state ← SecondReserve
        end with
    end if
case state = SecondReserve
    if f(currentTask, i) < 0 then
        state ← Resting
    else
        state ← TempWorker
    end if
end case
```

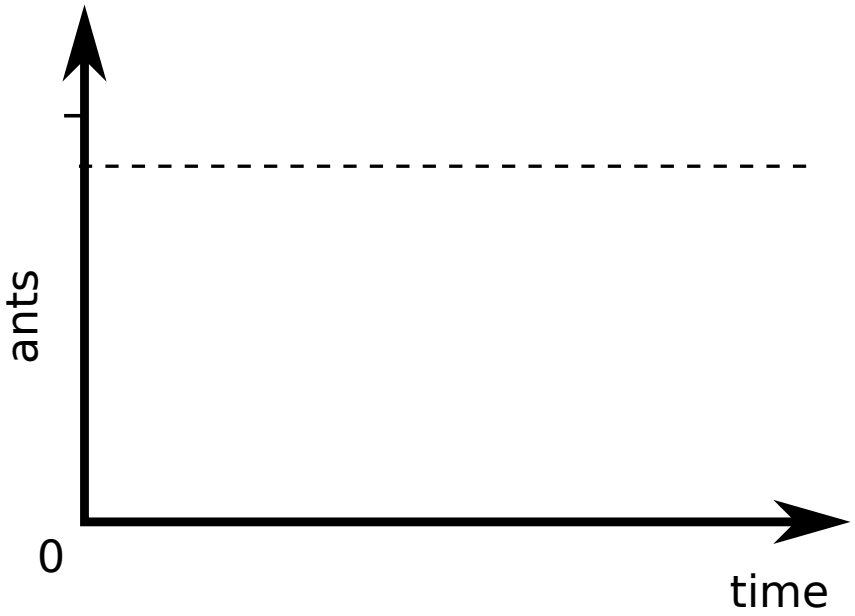
Analysis Outline

Analysis Outline

- **Lemma 1:** If during an interval tasks are not too oversatisfied, the probability there is a task with a deficit is exponentially small in the interval length.

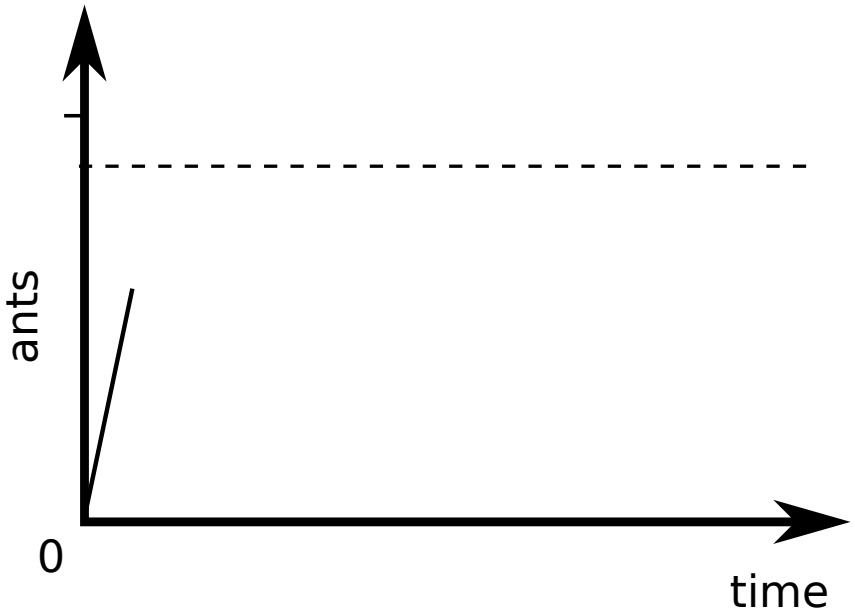
Analysis Outline

- **Lemma 1:** If during an interval tasks are not too oversatisfied, the probability there is a task with a deficit is exponentially small in the interval length.



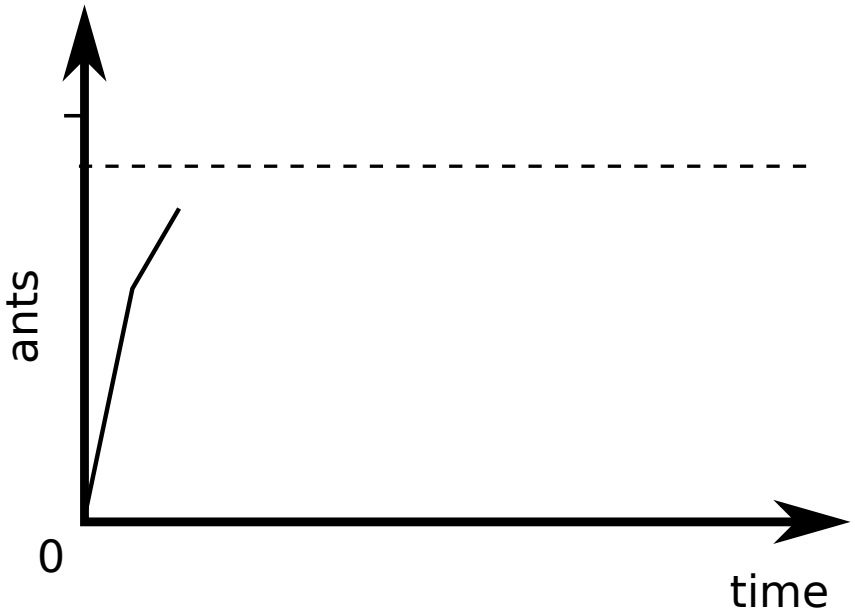
Analysis Outline

- **Lemma 1:** If during an interval tasks are not too oversatisfied, the probability there is a task with a deficit is exponentially small in the interval length.



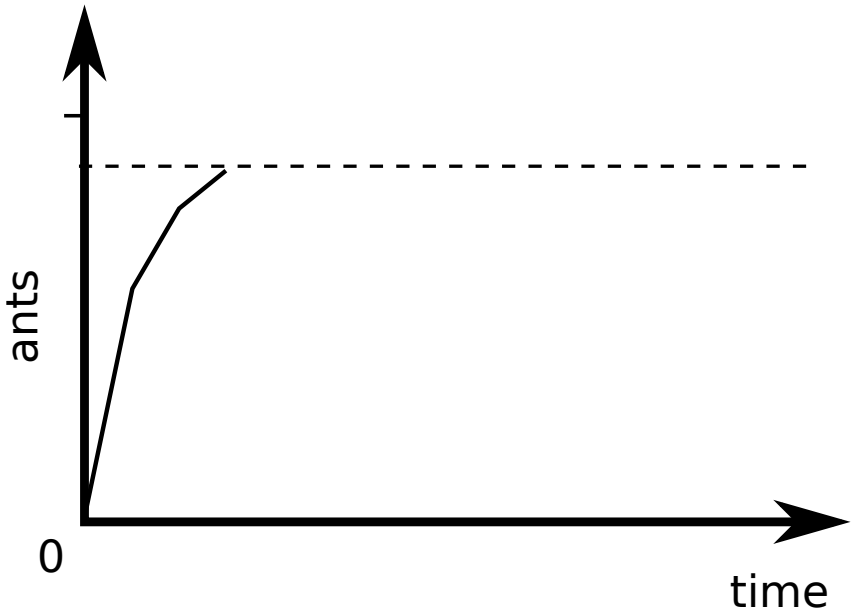
Analysis Outline

- **Lemma 1:** If during an interval tasks are not too oversatisfied, the probability there is a task with a deficit is exponentially small in the interval length.



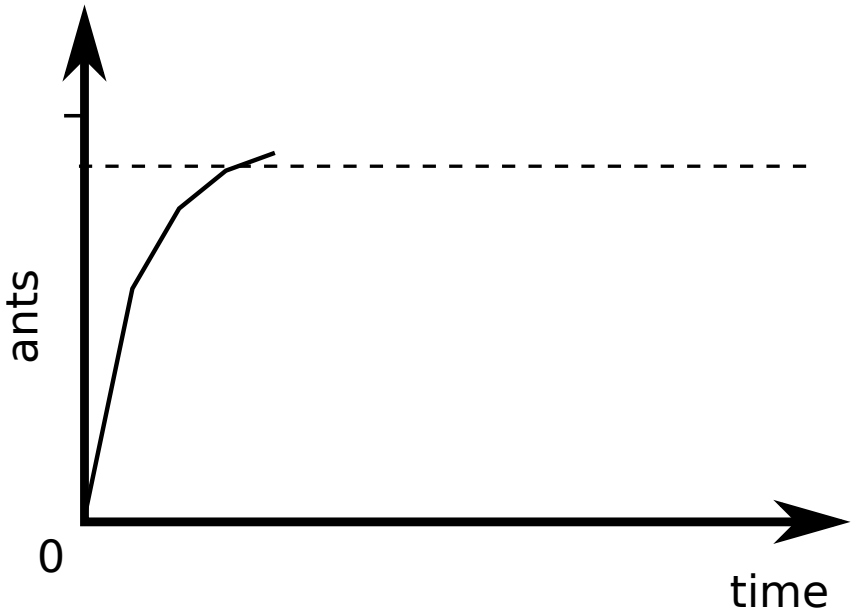
Analysis Outline

- **Lemma 1:** If during an interval tasks are not too oversatisfied, the probability there is a task with a deficit is exponentially small in the interval length.



Analysis Outline

- **Lemma 1:** If during an interval tasks are not too oversatisfied, the probability there is a task with a deficit is exponentially small in the interval length.

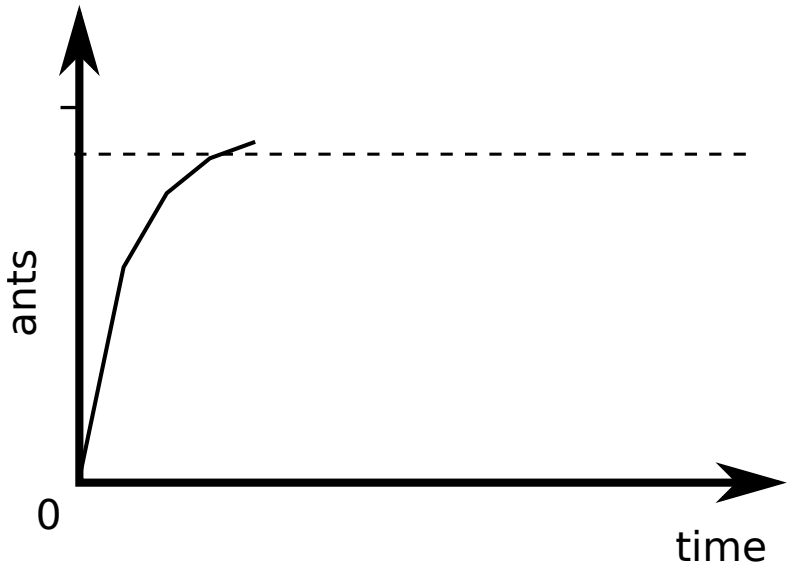


Analysis Outline

- **Lemma 2:** Once a task transitions from having a deficit to a surplus, it will keep oscillating every two or three rounds; moreover, with constant probability the number of ants involved in the oscillations is reduced by a constant fraction.

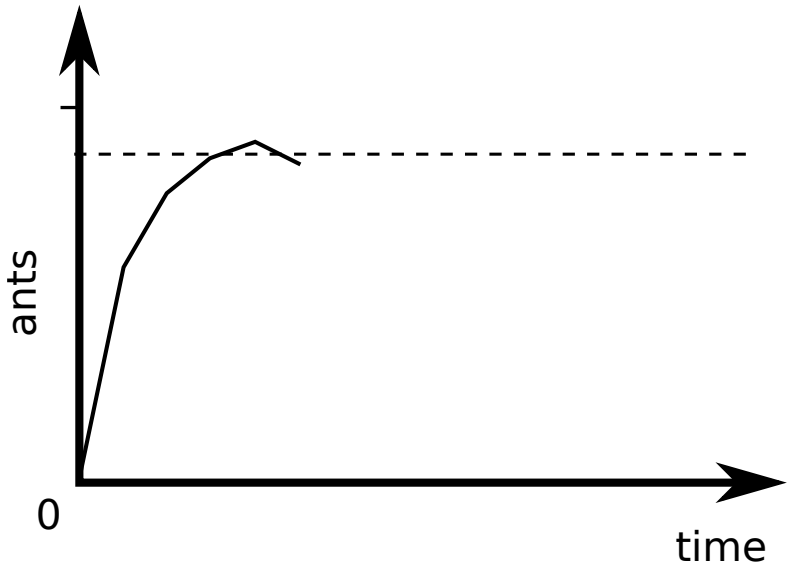
Analysis Outline

- **Lemma 2:** Once a task transitions from having a deficit to a surplus, it will keep oscillating every two or three rounds; moreover, with constant probability the number of ants involved in the oscillations is reduced by a constant fraction.



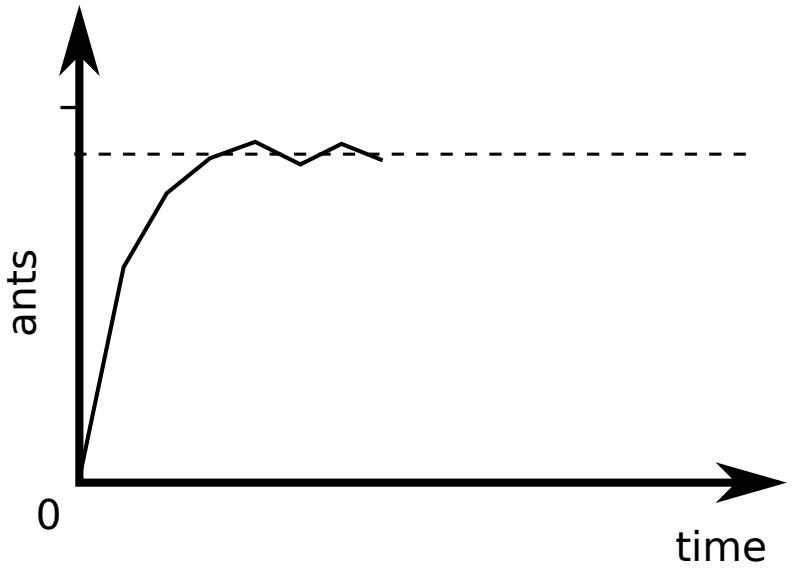
Analysis Outline

- **Lemma 2:** Once a task transitions from having a deficit to a surplus, it will keep oscillating every two or three rounds; moreover, with constant probability the number of ants involved in the oscillations is reduced by a constant fraction.



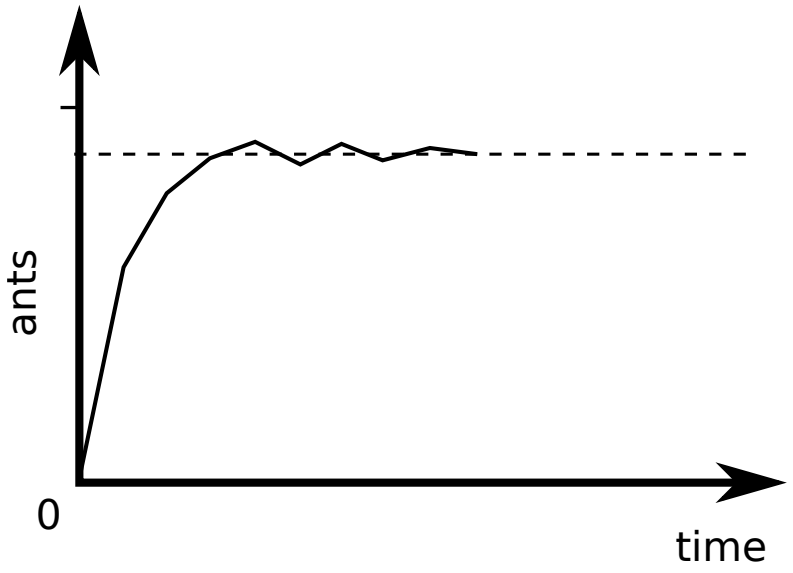
Analysis Outline

- **Lemma 2:** Once a task transitions from having a deficit to a surplus, it will keep oscillating every two or three rounds; moreover, with constant probability the number of ants involved in the oscillations is reduced by a constant fraction.



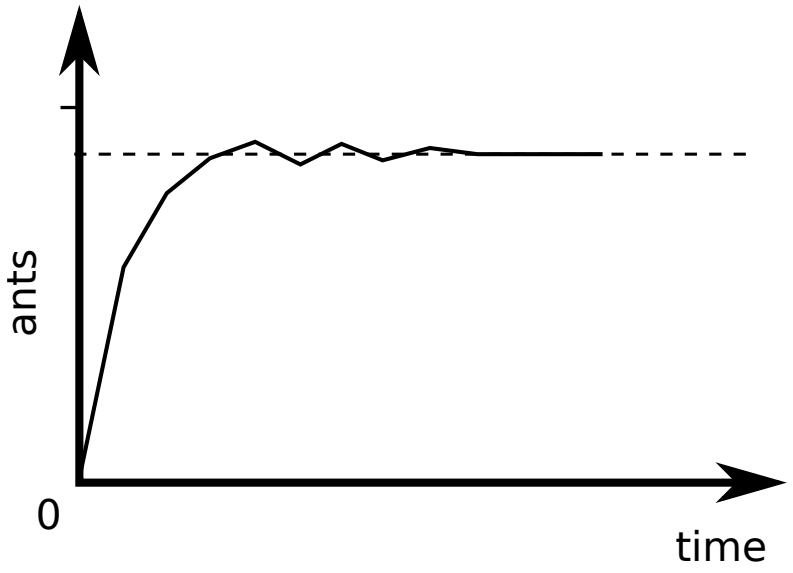
Analysis Outline

- **Lemma 2:** Once a task transitions from having a deficit to a surplus, it will keep oscillating every two or three rounds; moreover, with constant probability the number of ants involved in the oscillations is reduced by a constant fraction.



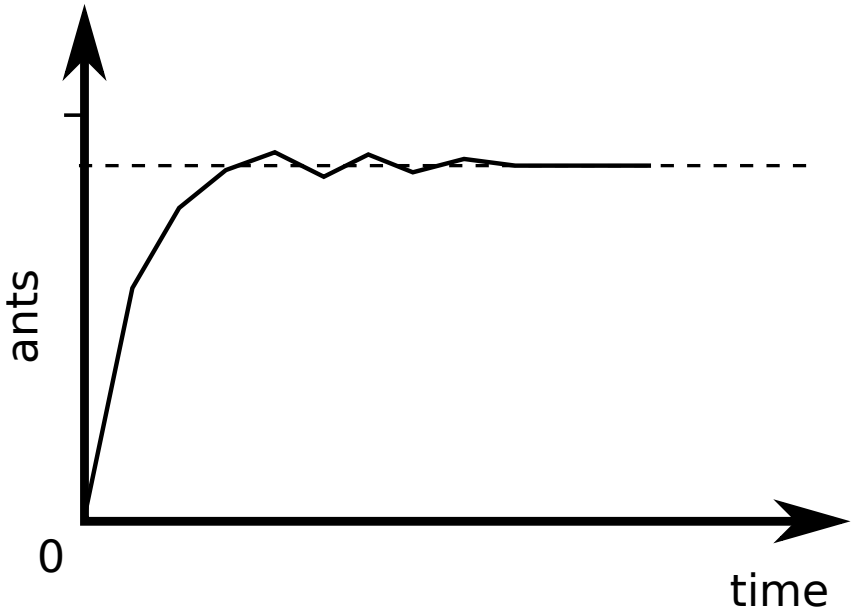
Analysis Outline

- **Lemma 2:** Once a task transitions from having a deficit to a surplus, it will keep oscillating every two or three rounds; moreover, with constant probability the number of ants involved in the oscillations is reduced by a constant fraction.



Analysis Outline

- **Theorem:** After $O(\log n)$ rounds, with high probability, ants reach an optimal task allocation.



Questions?