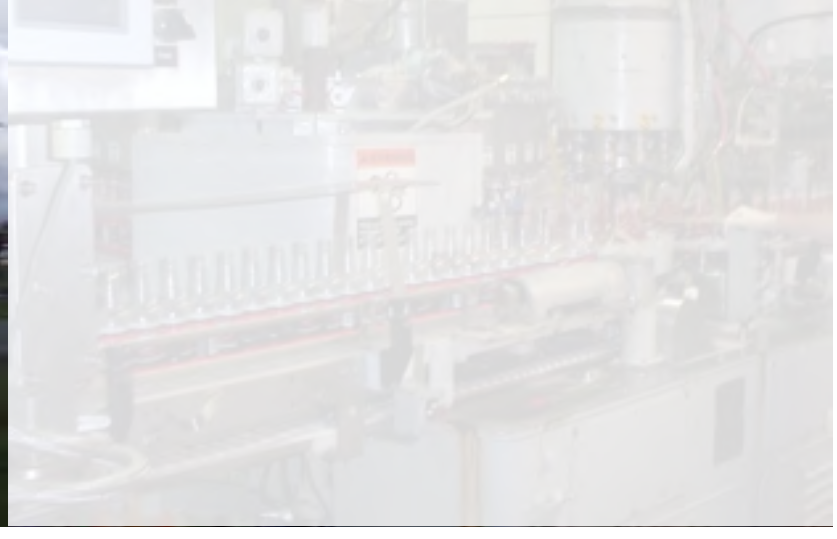
# Controller design for flow networks of switched servers with setup times

Erjen Lefeber

Goldrain, September 1, 2007

# Motivation

# Motivation



# Motivation

# Motivation

# Problem

## Problem

How to control these networks?

Decisions: When to switch, and to which job-type

Goals: Maximal throughput, minimal flow time

# Problem

## Problem

How to control these networks?

Decisions: When to switch, and to which job-type

Goals: Maximal throughput, minimal flow time

## Current approach

Start from policy, analyze resulting dynamics

# Problem

## Problem

How to control these networks?

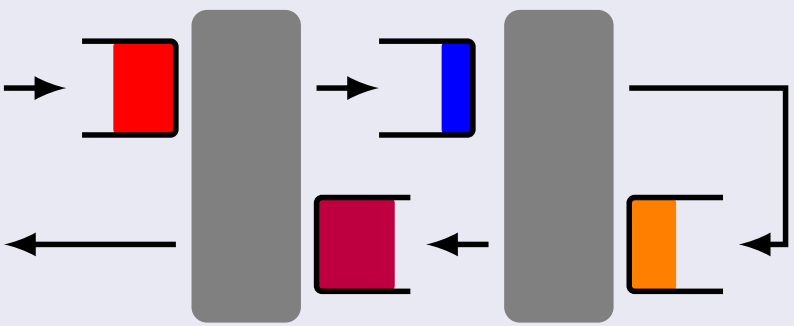
Decisions: When to switch, and to which job-type

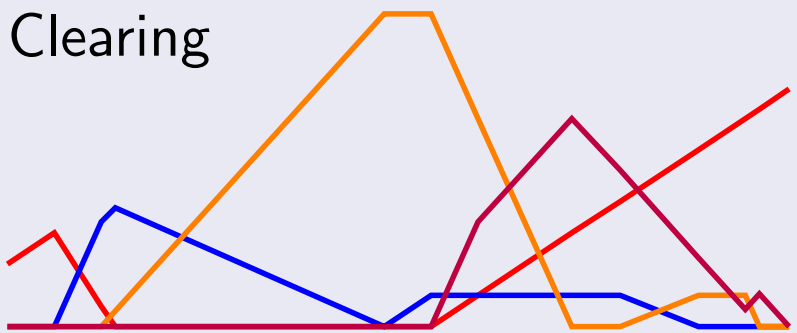
Goals: Maximal throughput, minimal flow time

## Current approach

Start from policy, analyze resulting dynamics

## Kumar, Seidman (1990)





# Problem

## Current status (after two decades)

Several policies exist that guarantee stability of the network

# Problem

## Current status (after two decades)

Several policies exist that guarantee stability of the network

## Remark

Stability is only a prerequisite for a good policy

# Problem

## Current status (after two decades)

Several policies exist that guarantee stability of the network

## Remark

Stability is only a prerequisite for a good policy

## Open issues

- Do existing policies yield satisfactory network performance?
- How to obtain pre-specified network behavior?

# Problem

## Current status (after two decades)

Several policies exist that guarantee stability of the network

## Remark

Stability is only a prerequisite for a good policy

## Open issues

- Do existing policies yield satisfactory network performance?
- How to obtain pre-specified network behavior?

## Main subject of study (modest)

Fixed, deterministic flow networks (not evolving, constant inflow)

# Main idea

## Important observation

"The main interest is in the resulting behavior. So why not use that as a starting point?"

# Main idea

## Important observation

"The main interest is in the resulting behavior. So why not use that as a starting point?"

## Approach

Start from desired behavior and *design* policy, instead of start from policy and analyze resulting dynamics

# Main idea

## Important observation

"The main interest is in the resulting behavior. So why not use that as a starting point?"

## Approach

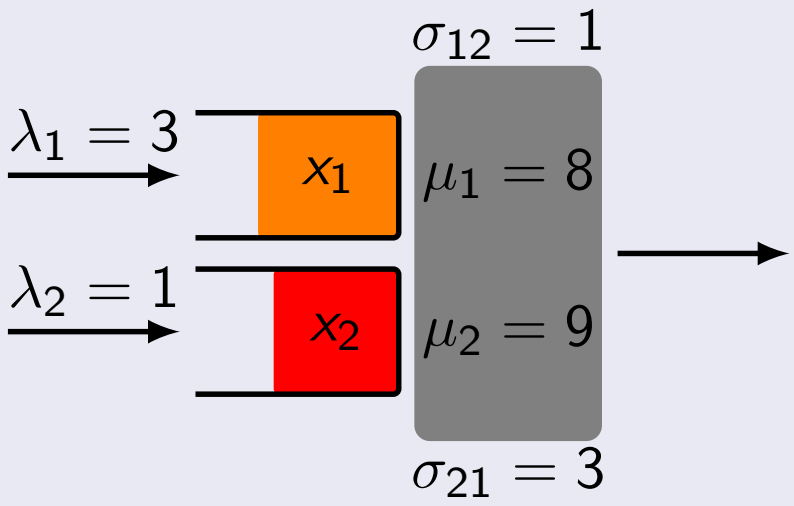
Start from desired behavior and *design* policy, instead of start from policy and analyze resulting dynamics

## Consequence

Separation of concern: desired behavior and controller can be designed separately.

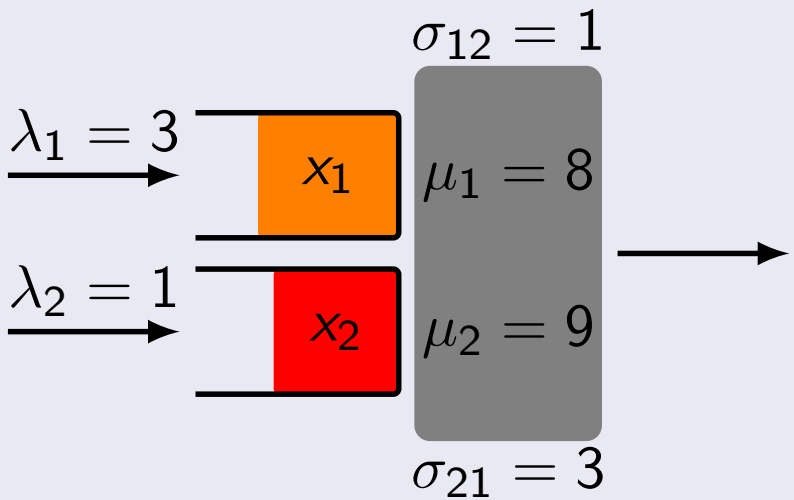
# Example

## Single machine

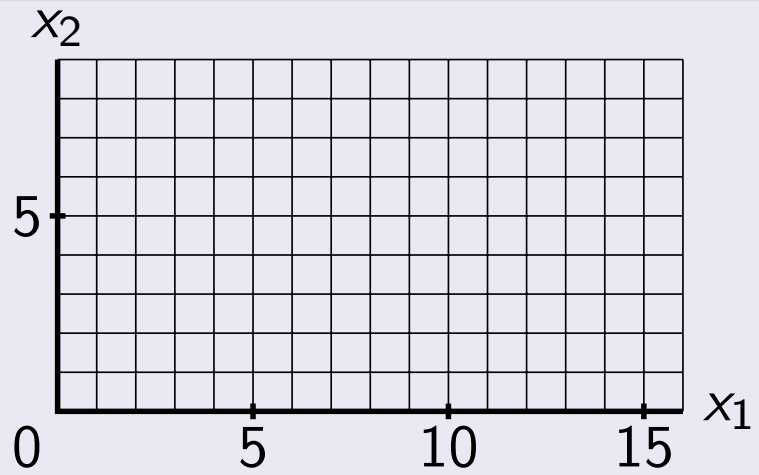


# Example

## Single machine




## Desired behavior

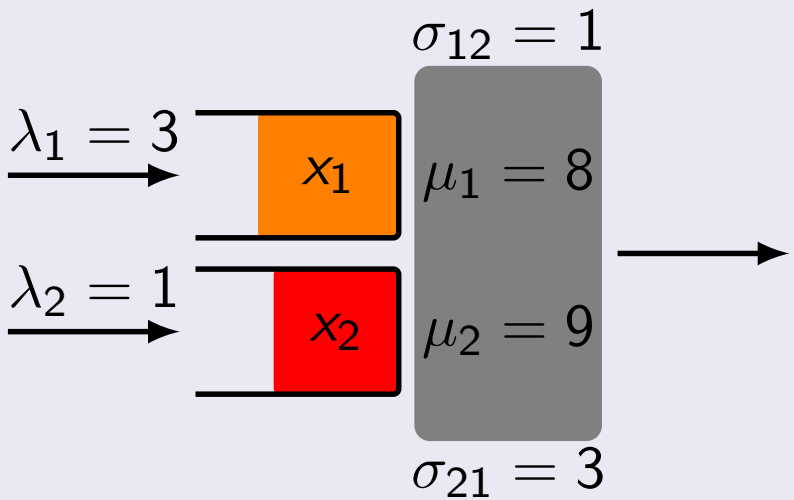



## Objective

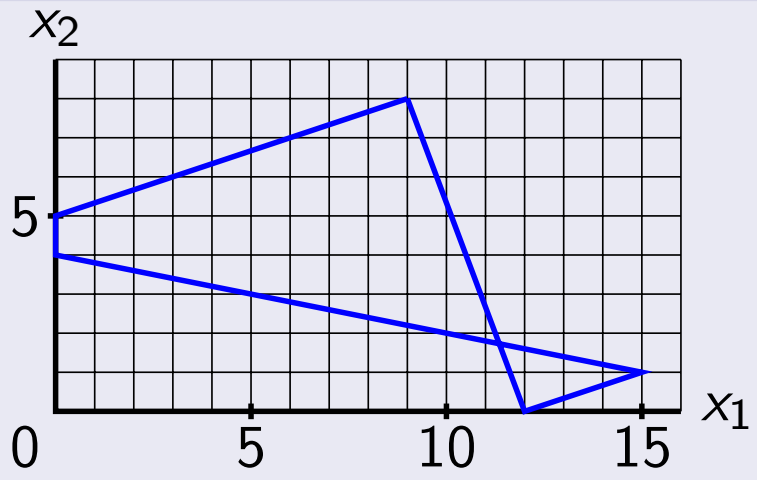
$$\text{minimize } \frac{1}{T}\int_0^T x_1(\tau) + x_2(\tau)\,\mathrm{d}\tau$$

# Example

## Single machine




## Desired behavior

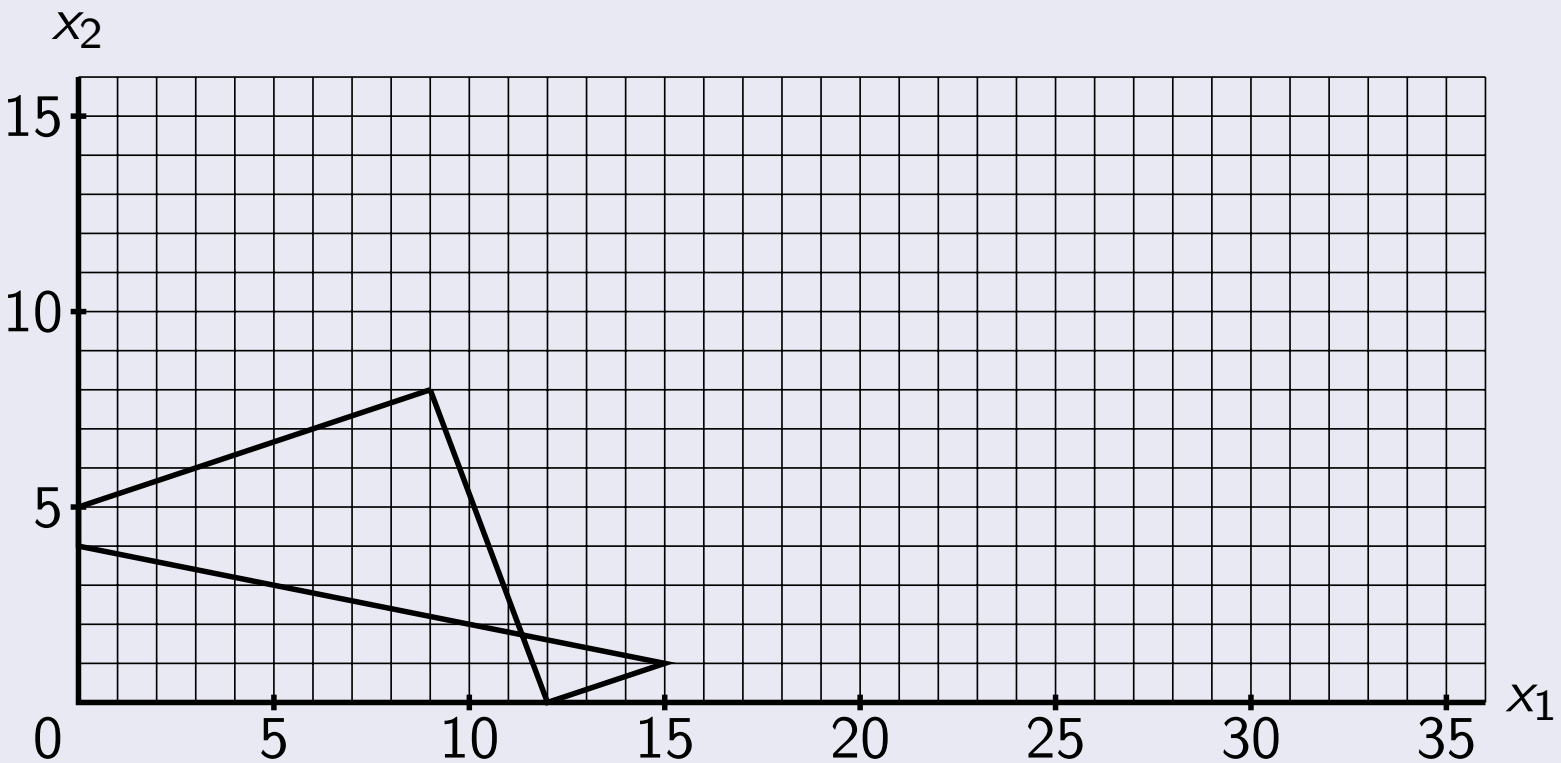



## Some remarks

- Desired behavior minimizes $\frac{1}{T}\int_0^T x_1(\tau) + x_2(\tau)\,\mathrm{d}\tau$
- Many existing policies assume non-idling a-priori
- No policy yet!

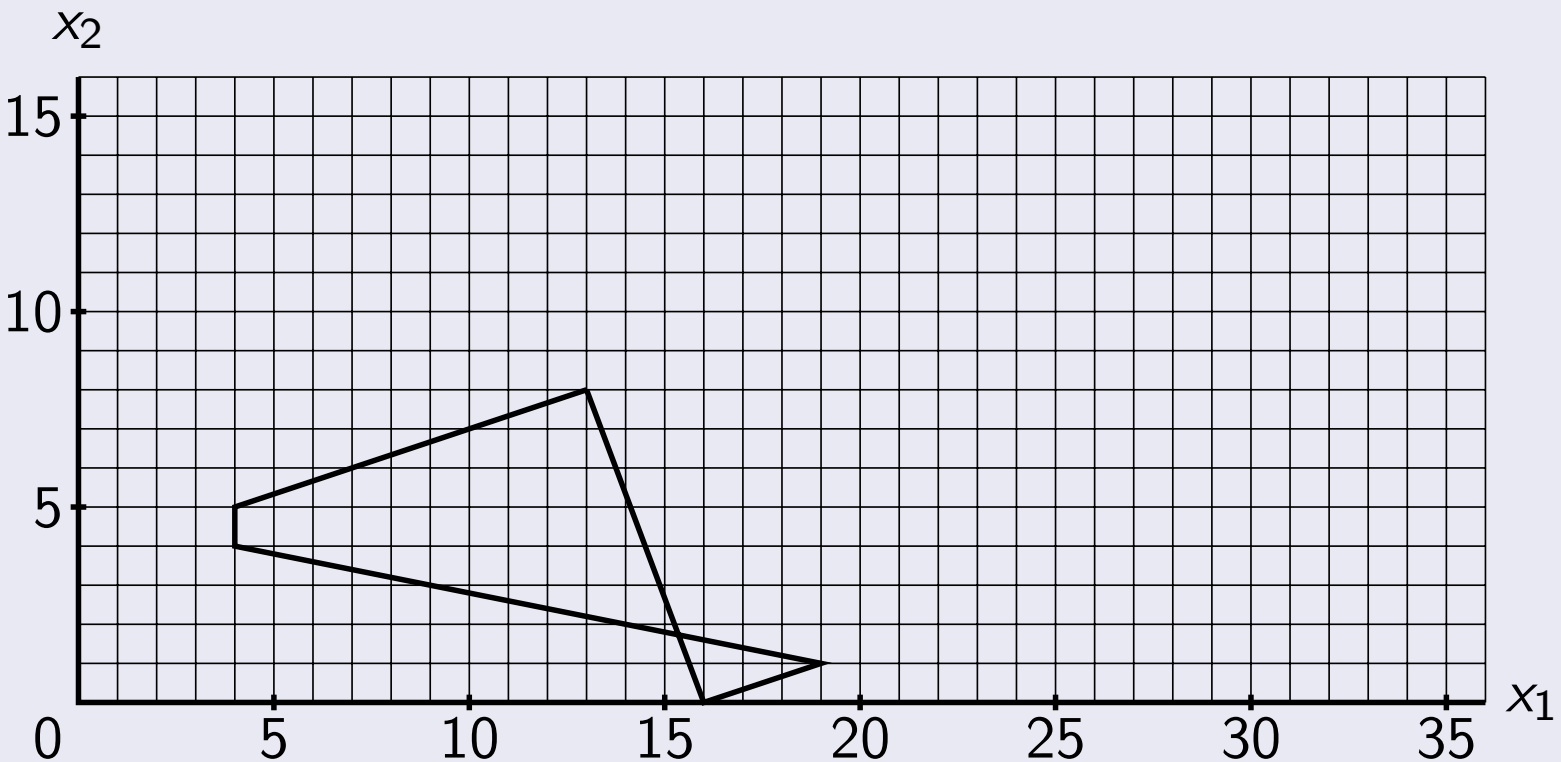
# Example

## Controller design [cf. Physica A (2006)]



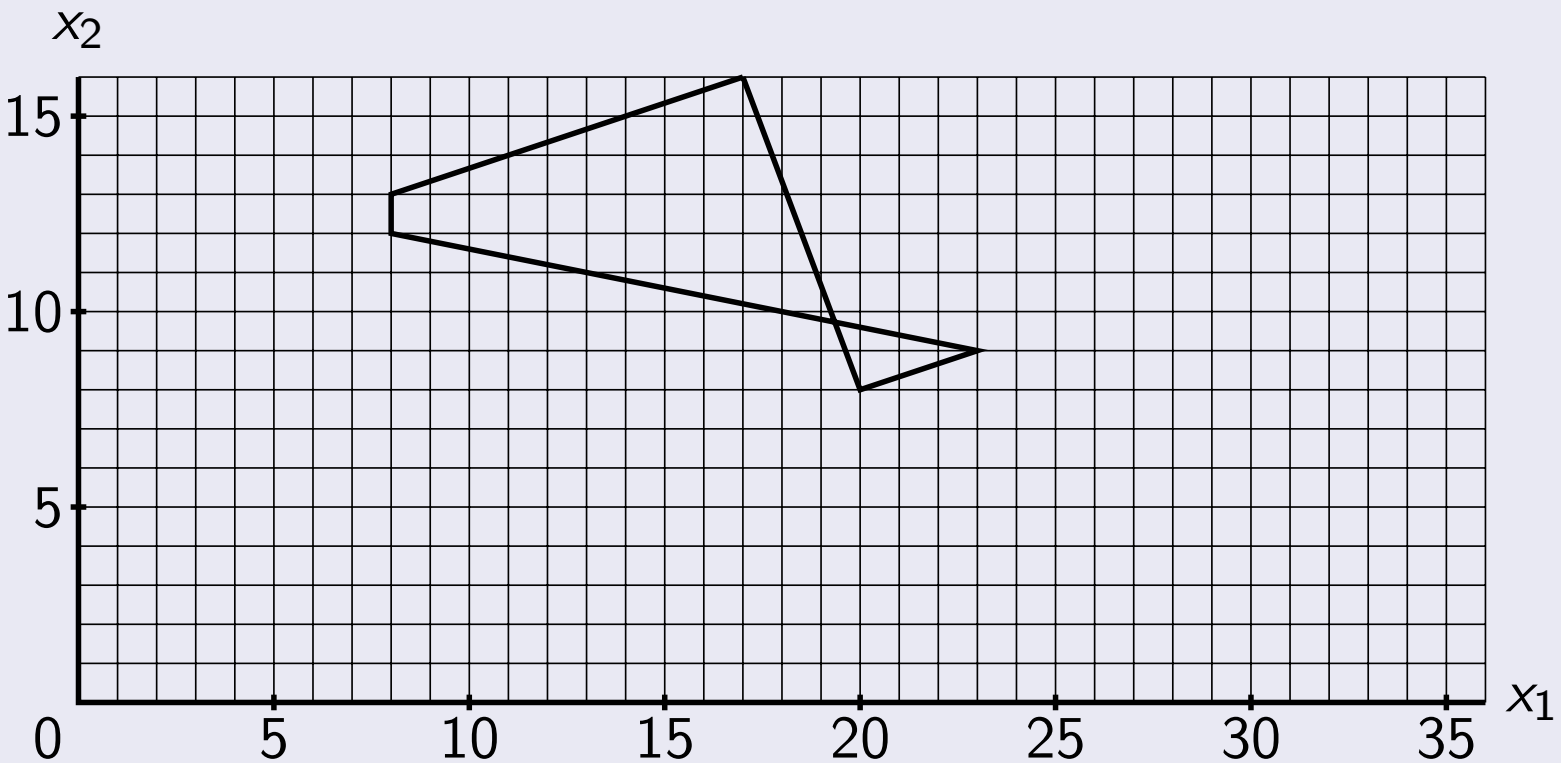
# Example

## Controller design [cf. Physica A (2006)]



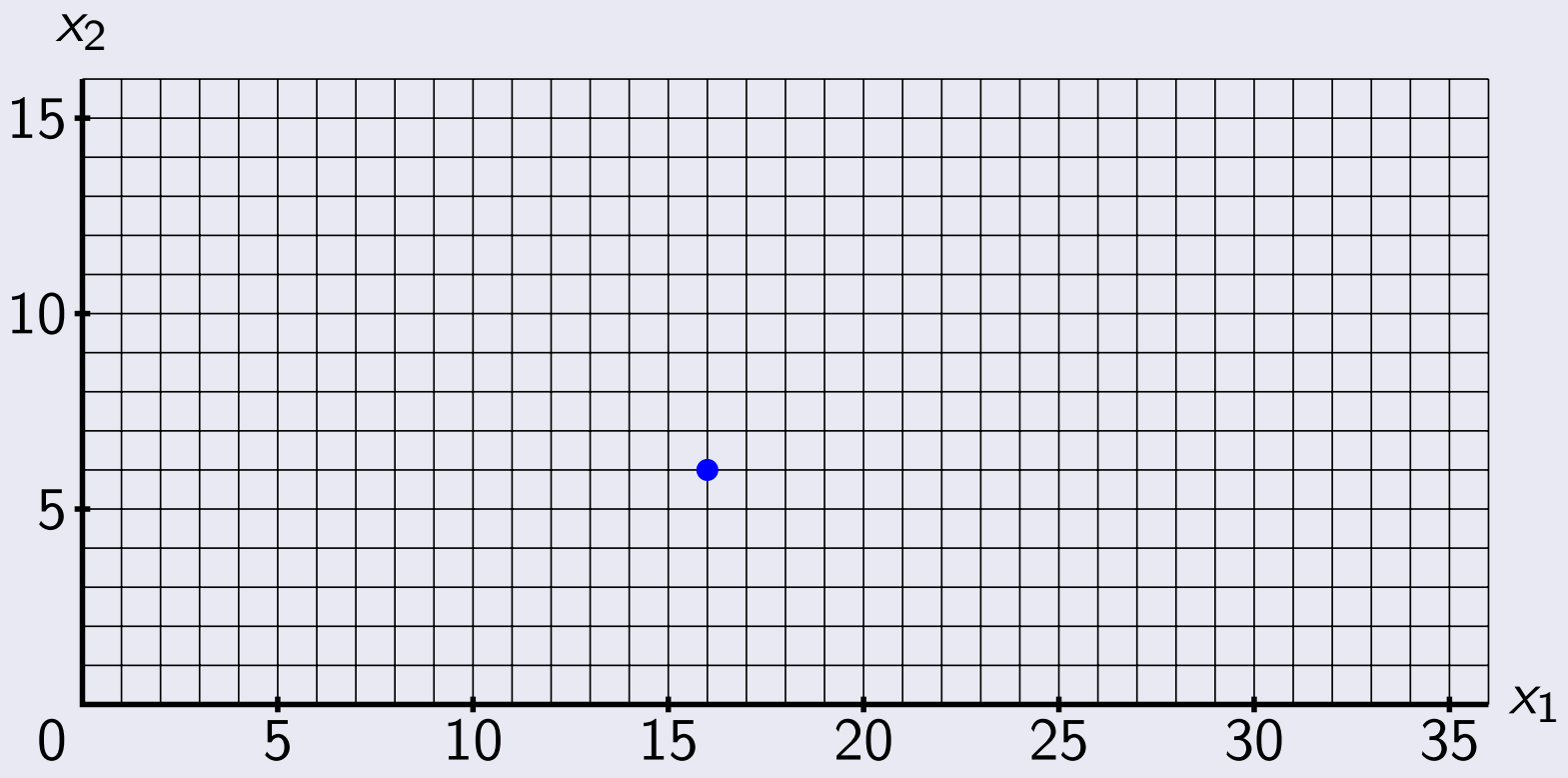
# Example

## Controller design [cf. Physica A (2006)]



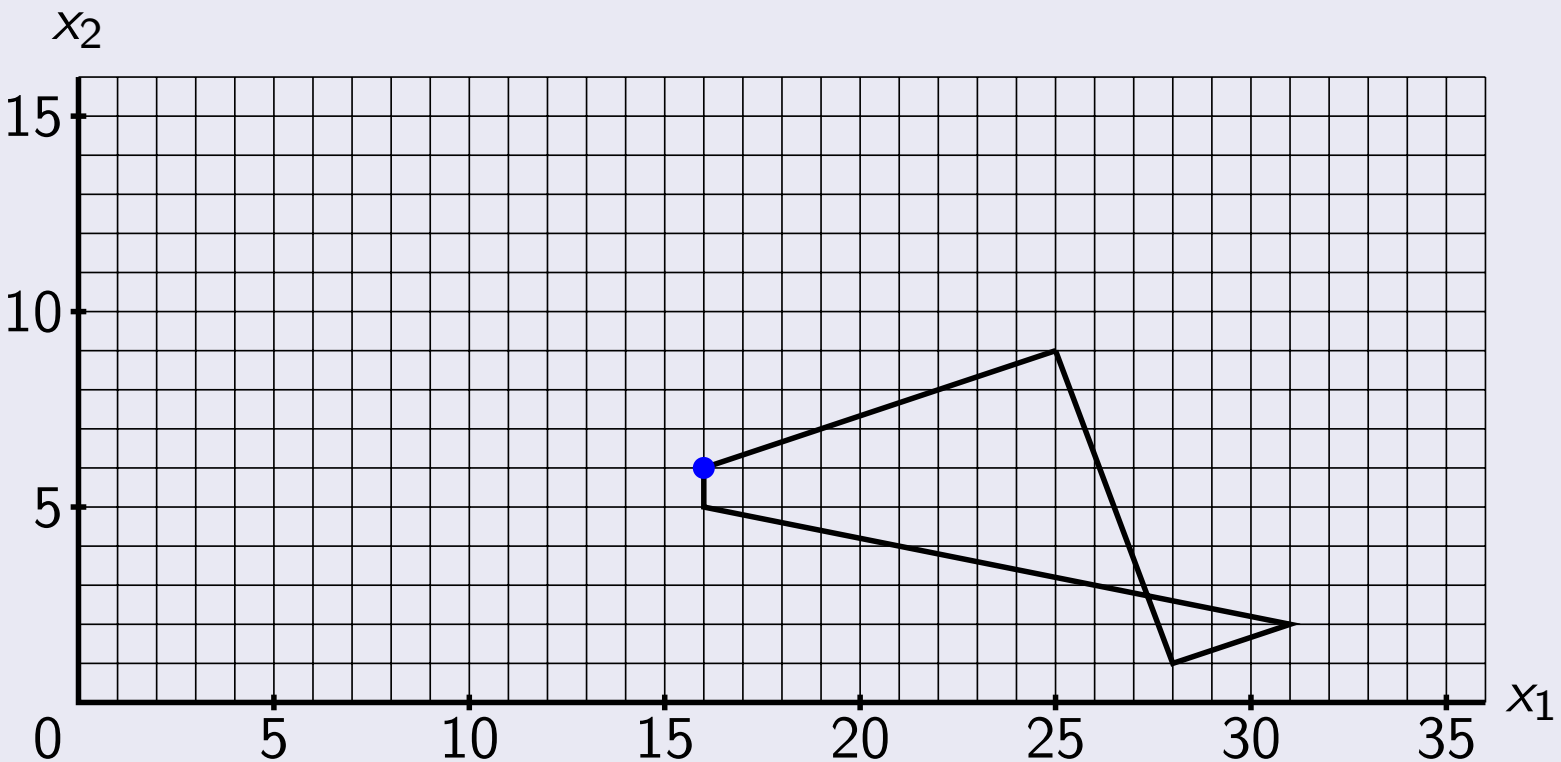
# Example

## Controller design [cf. Physica A (2006)]



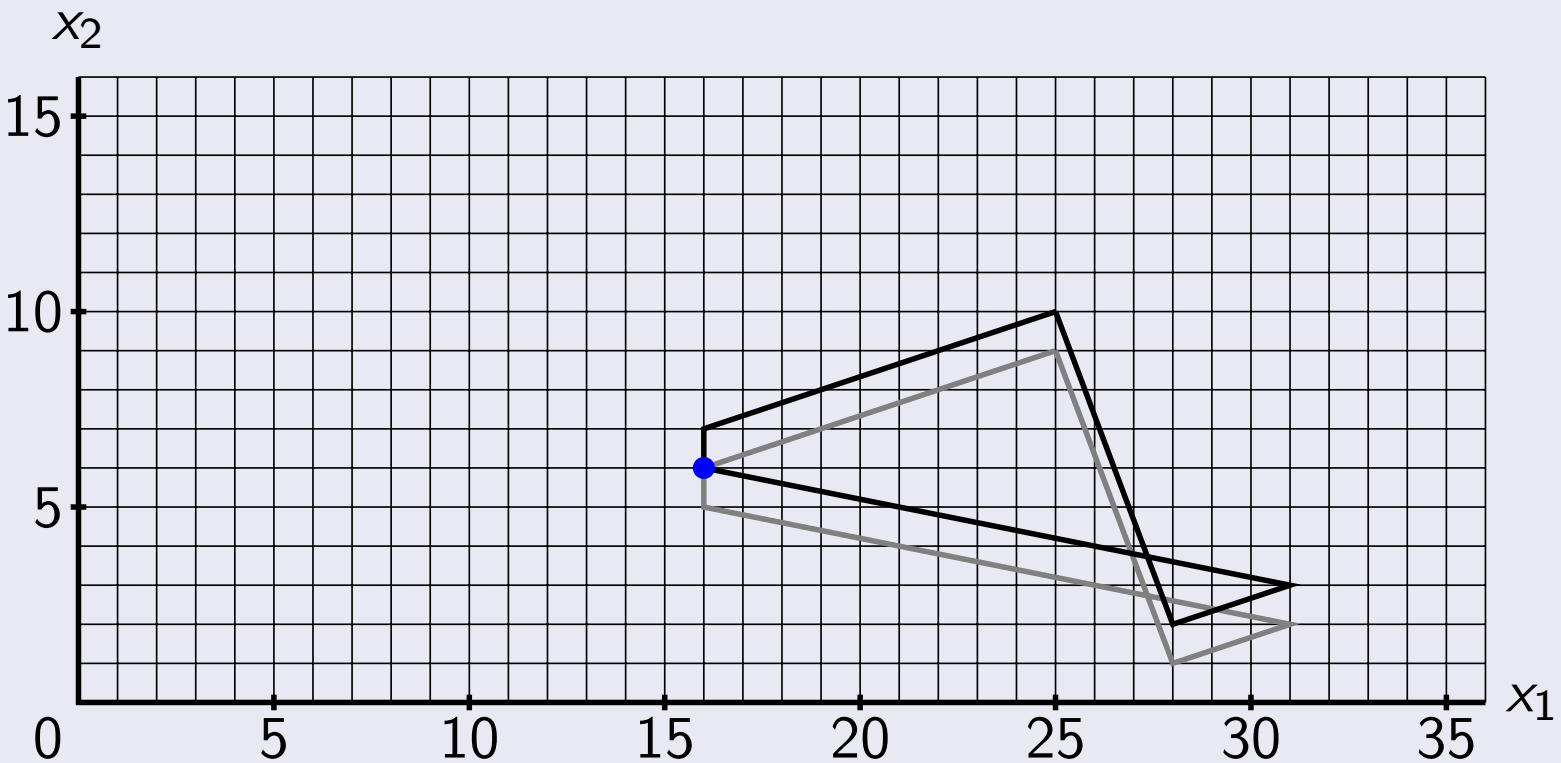
# Example

## Controller design [cf. Physica A (2006)]



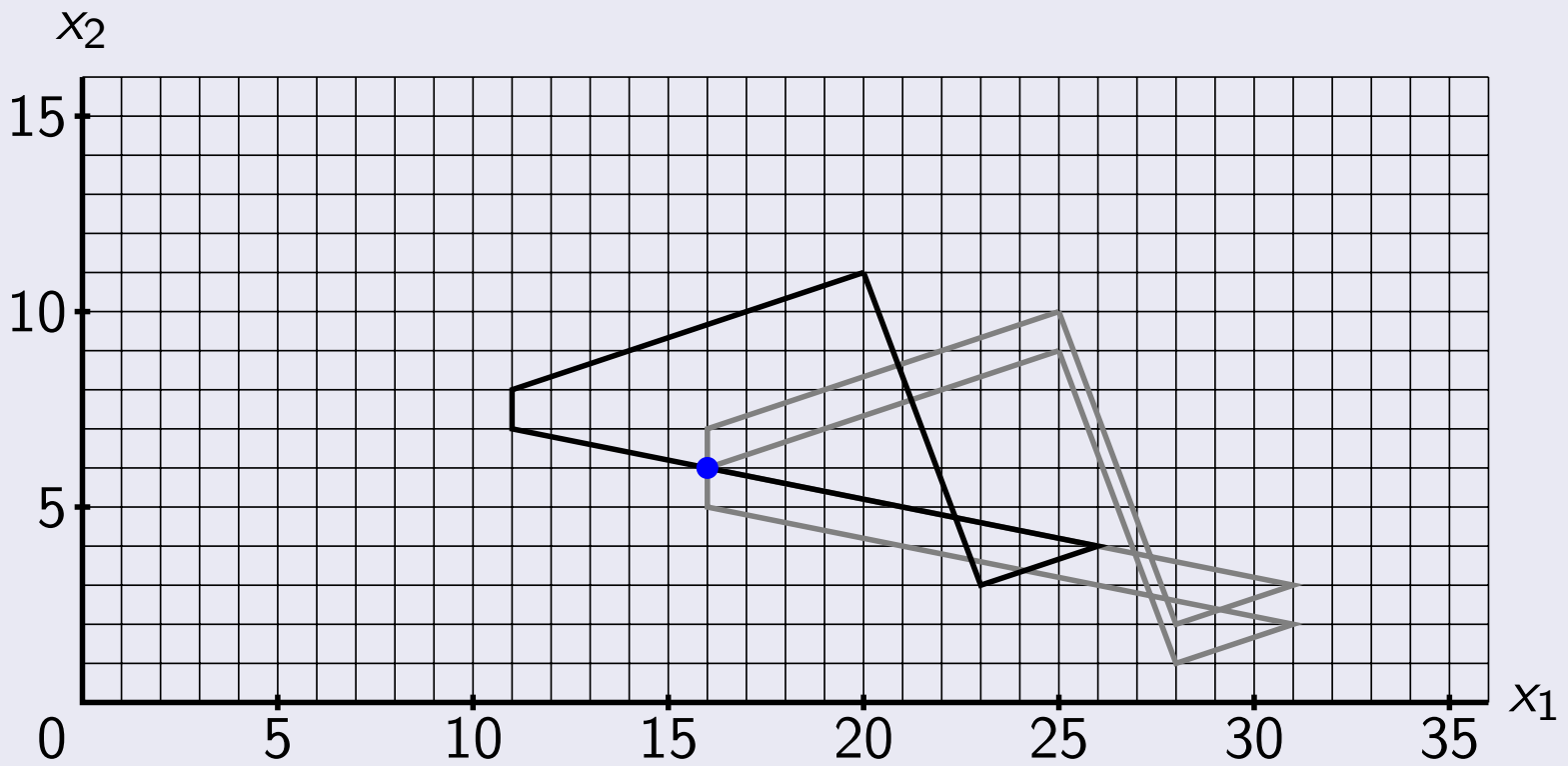
# Example

## Controller design [cf. Physica A (2006)]



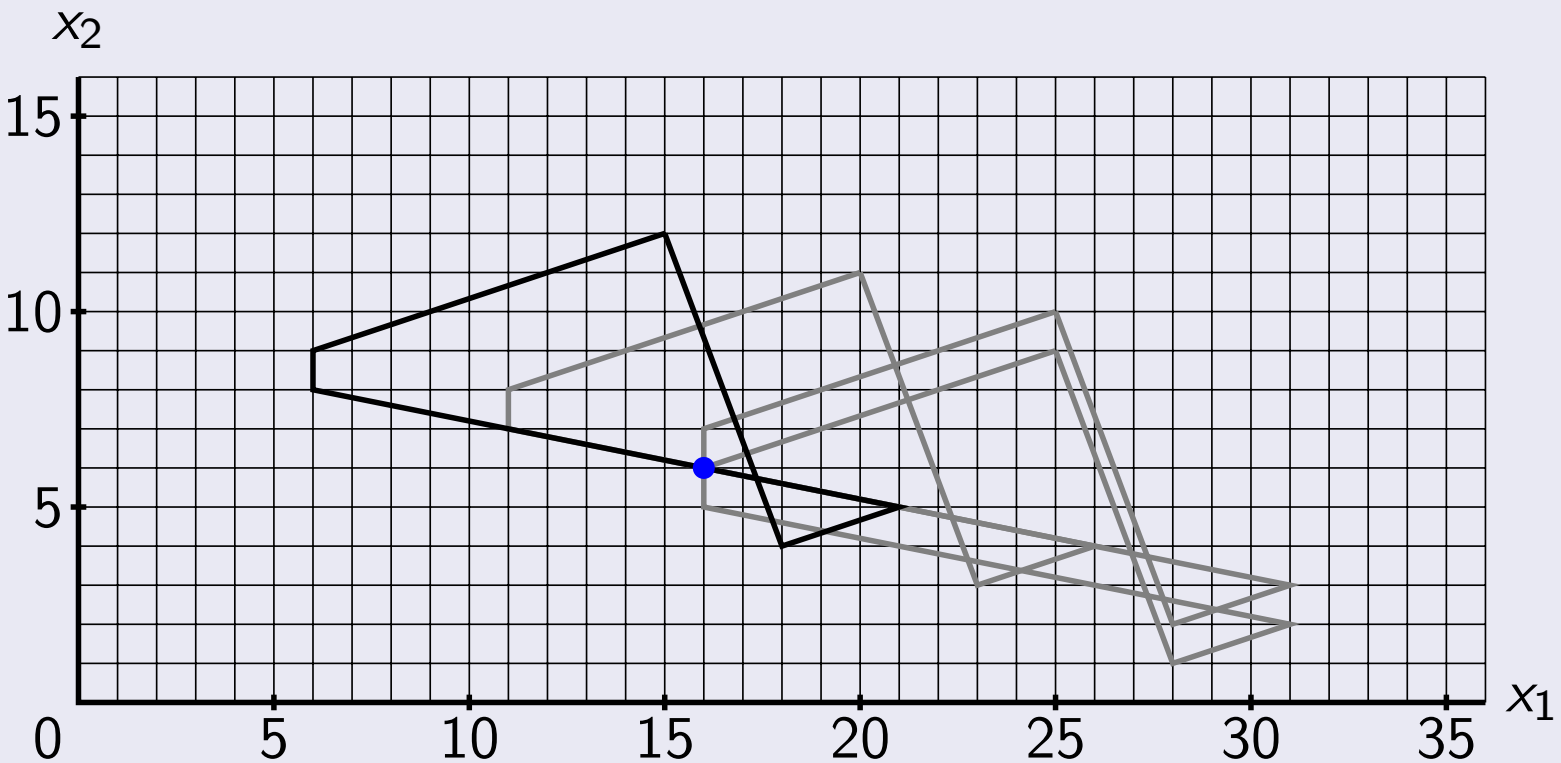
# Example

## Controller design [cf. Physica A (2006)]



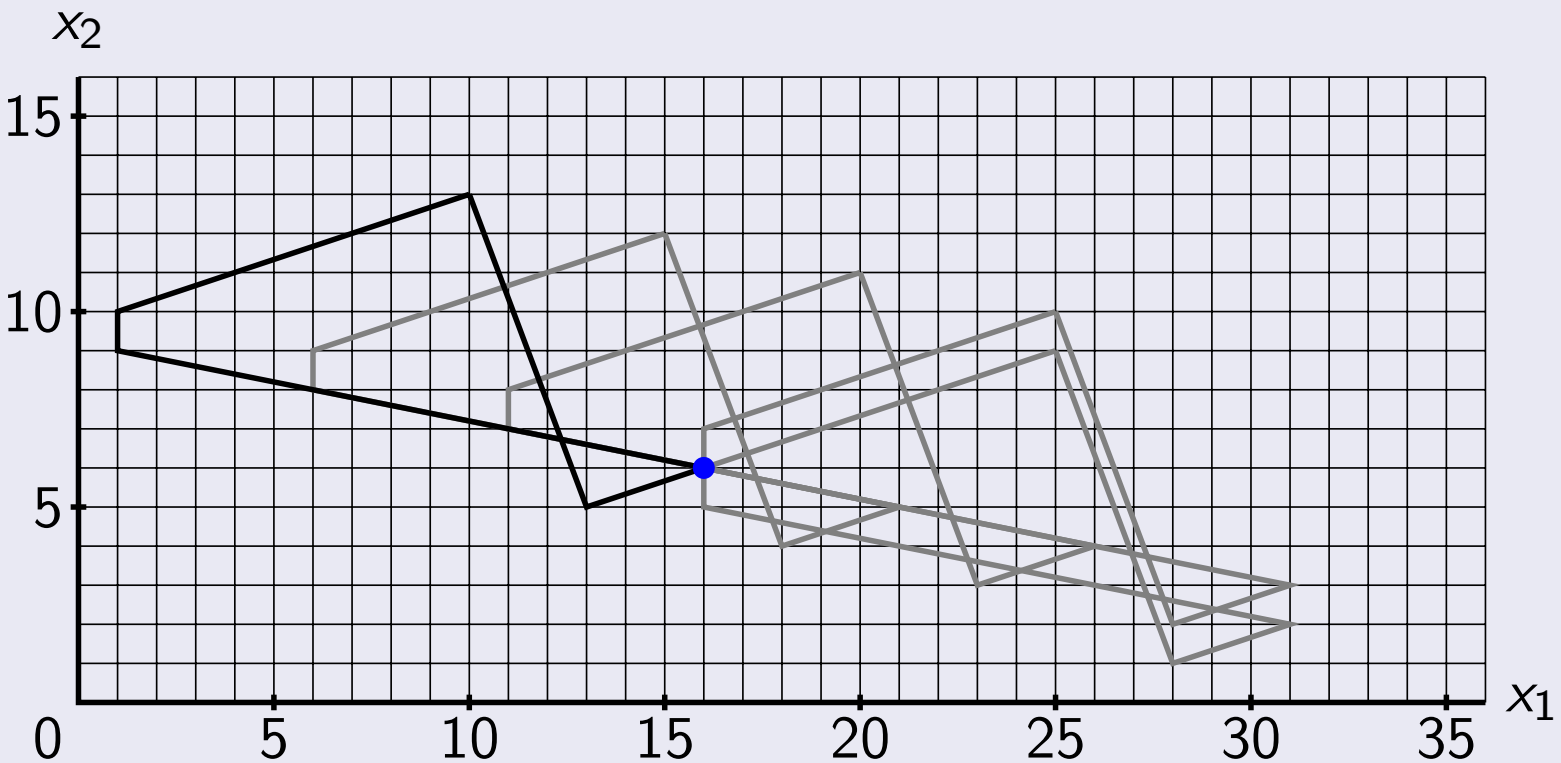
# Example

## Controller design [cf. Physica A (2006)]



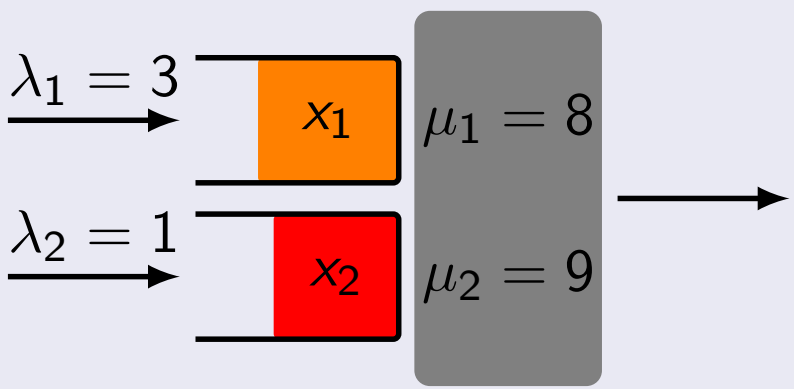
# Example

## Controller design [cf. Physica A (2006)]

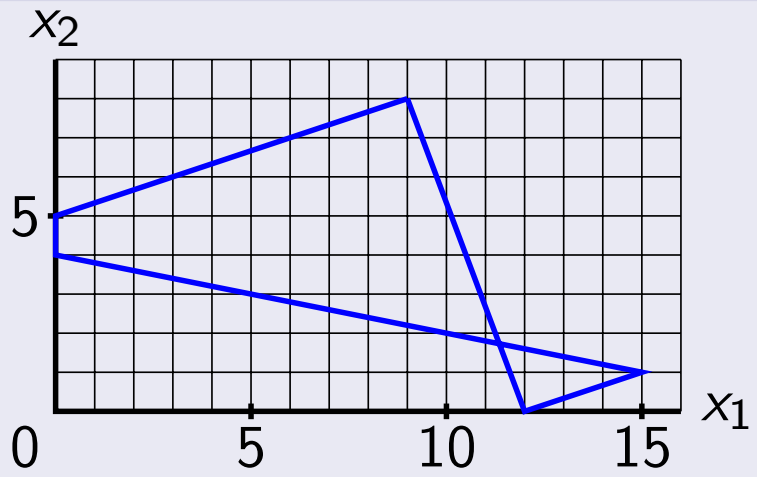


# Resulting controller

## Single machine

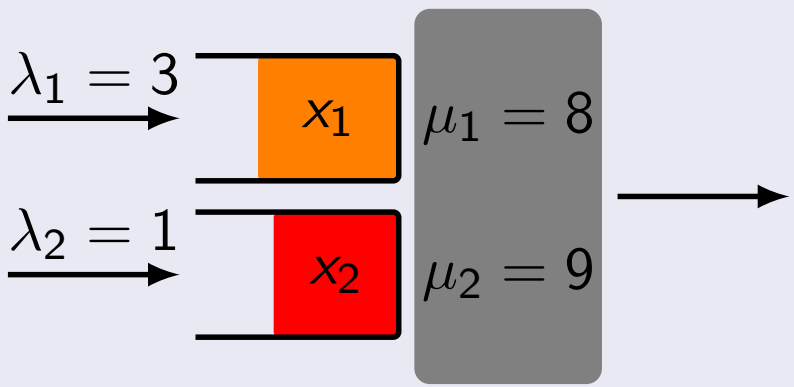



## Desired behavior

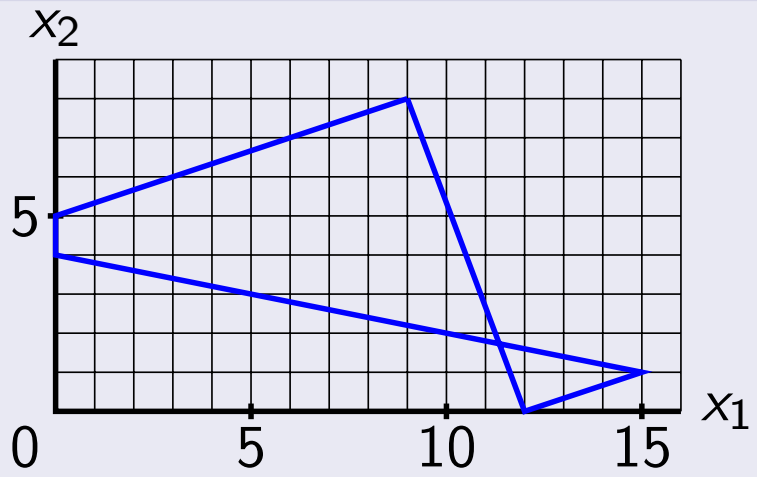


# Resulting controller

## Single machine




## Desired behavior

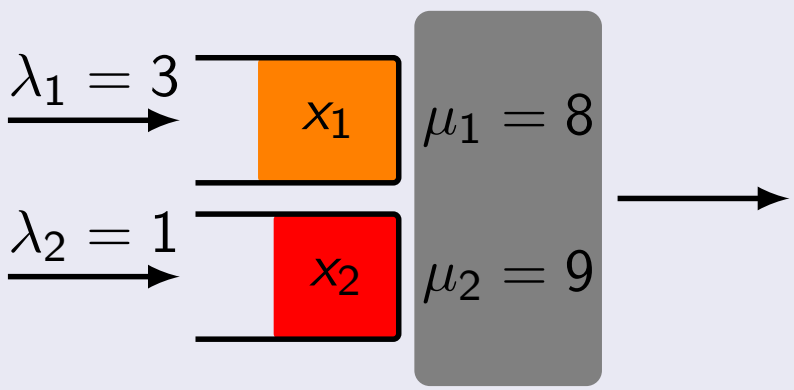



## Resulting Controller

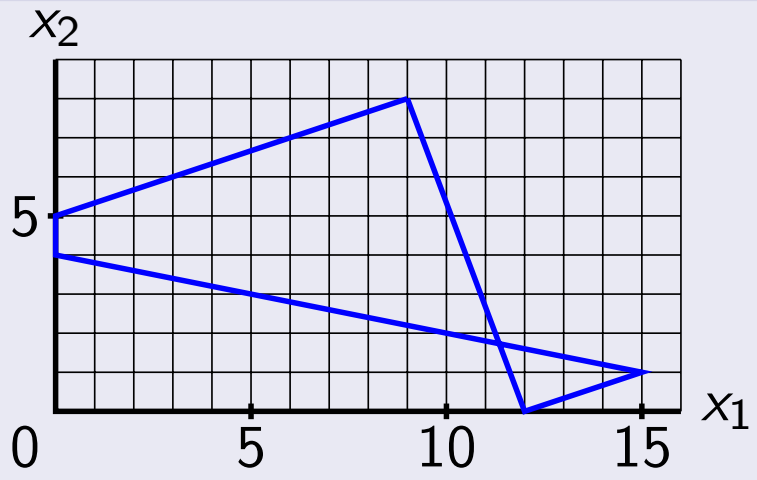
- When serving type 1:
  1. empty buffer
  2. serve until $x_2 \geq 5$
  3. switch to type 2
- When serving type 2:
  1. empty buffer
  2. serve until $x_1 \geq 12$
  3. switch to type 1

# Resulting controller

## Single machine




## Desired behavior

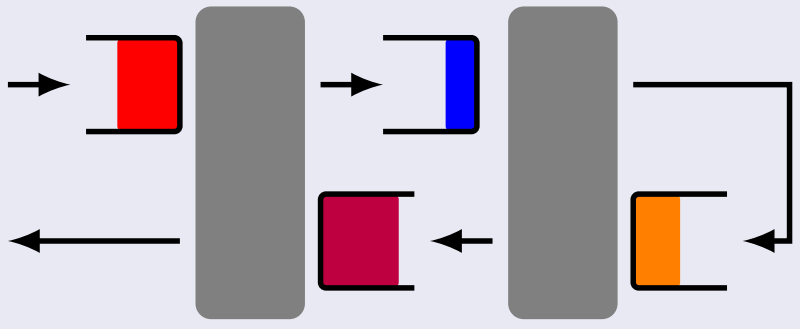



## Resulting Controller

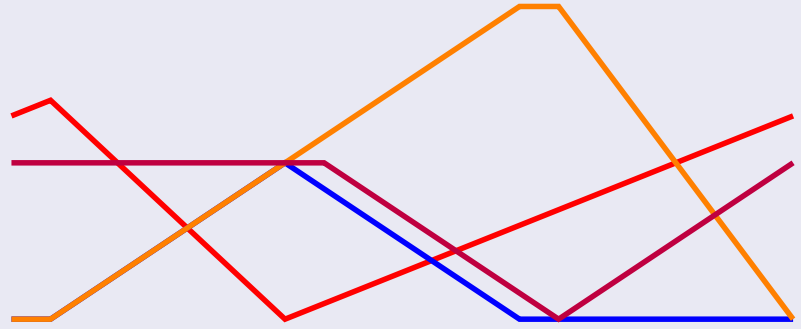
- When serving type 1:
  1. empty buffer
  2. serve until $x_2 \geq 5$
  3. switch to type 2
- When serving type 2:
  1. empty buffer
  2. serve until $x_1 \geq 12$
  3. switch to type 1

# Network setting (Kumar-Seidman)

## Network



## Desired behavior



## Controller

Mode (1,2): to (4,2) when both $x_1 = 0$ and $x_2 + x_3 \geq 1000$

Mode (4,2): to (4,3) when both $x_2 = 0$ and $x_4 \leq 83\frac{1}{3}$

Mode (4,3): to (1,2) when $x_3 = 0$

## Remark

Centralized controller, i.e. non-distributed

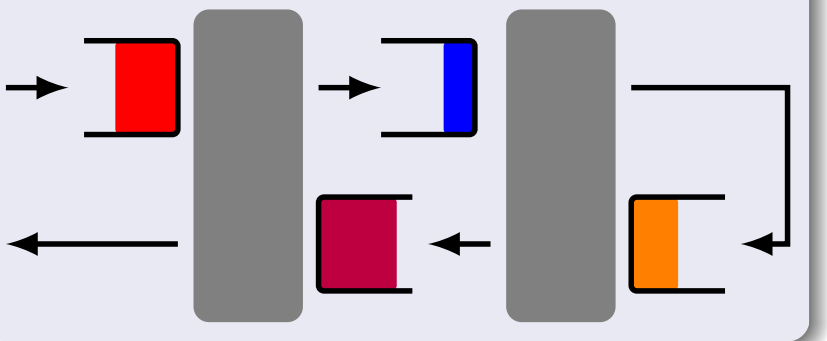
# Distributed control

## Important observation

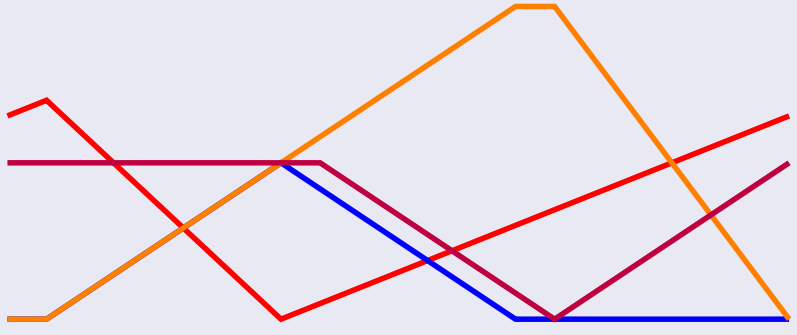
"Observing arrivals for a while provides information about the state in other parts of the network"

# Distributed controller for Kumar-Seidman network

## Network

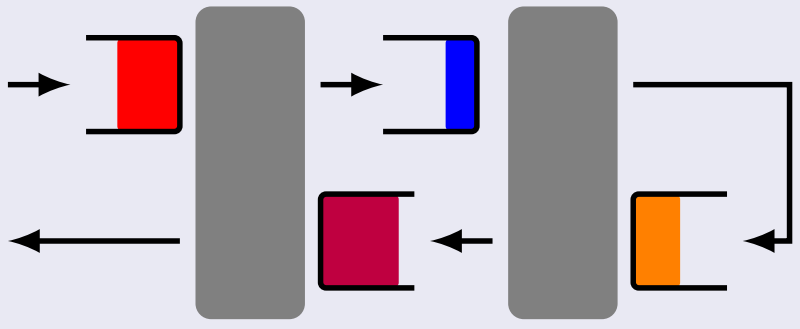


## Desired behavior

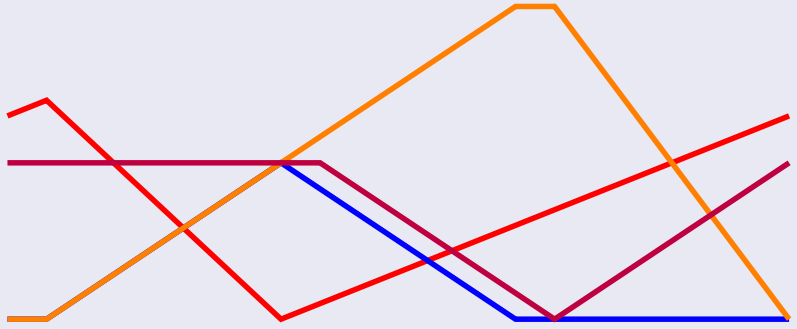


# Distributed controller for Kumar-Seidman network

## Network



## Desired behavior



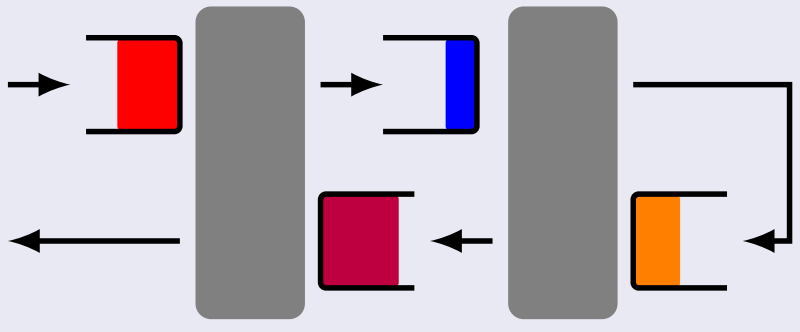
## Controller resulting from behavior with minimal number of jobs

Serving 2: Serve at least 1000 jobs until $x_2 = 0$, then switch.

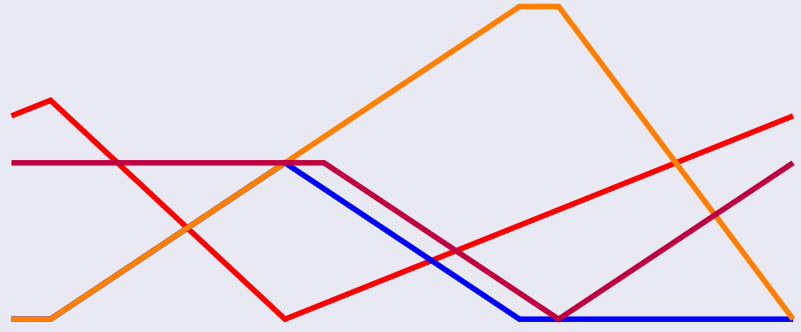
Serving 3: Empty buffer, then switch.

# Distributed controller for Kumar-Seidman network

## Network



## Desired behavior



## Controller resulting from behavior with minimal number of jobs

Serving 1: Serve at least 1000 jobs until $x_1 = 0$, then switch. Let $\bar{x}_1$ be nr of jobs served.

Serving 2: Serve at least 1000 jobs until $x_2 = 0$, then switch.

Serving 4: Let $\bar{x}_4$ be nr of jobs in Buffer 4 after setup. Serve $\bar{x}_4 + \frac{1}{2}\bar{x}_1$ jobs, then switch.

Serving 3: Empty buffer, then switch.

# Concluding remarks

## Ideas from control theory can be useful

1. Determine optimal behavior (trajectory generation)
2. Derive centralized controller (state feedback control)
3. Derive decentralized controllers (dyn. output feedback)

# Concluding remarks

## Ideas from control theory can be useful

1. Determine optimal behavior (trajectory generation)
2. Derive centralized controller (state feedback control)
3. Derive decentralized controllers (dyn. output feedback)

## Many questions remaining

- How to find good (or even optimal) network behavior?
- How to design decentralized controllers (observability)?
- Does feedback work well in stochastic environment?
- Robustness against parameter changes?
- What if network is modified?
- What if arrival rate not constant?
- What if routing is not fixed?