Online optimization: Elevator and vehicle scheduling

CO@Work Berlin

Benjamin Hiller

ZIB

10/01/2009

Berlin Mathematical School

DFG Research Center MATHEON
Mathematics for key technologies

Online problems

high rack warehouse

elevator scheduling

dispatching service vehicles

Outline

Recap: Online optimization

Theoretical framework: Online Dial-a-Ride problems and competitive analysis

Online Optimization in Practice: Reoptimization Algorithms
- Dispatching the service vehicles of ADAC
- Controlling cargo elevators in a distribution center
- Controlling passenger elevators in high-rise buildings

Theory again: The Online Bin Coloring problem

Outline

Recap: Online optimization

Theoretical framework: Online Dial-a-Ride problems and competitive analysis

Online Optimization in Practice: Reoptimization Algorithms
- Dispatching the service vehicles of ADAC
- Controlling cargo elevators in a distribution center
- Controlling passenger elevators in high-rise buildings

Theory again: The Online Bin Coloring problem

What is online?

typical application: elevator control

- passenger calls arrive over time
- a new call must immediately be incorporated in the elevator schedule

Online optimization

In online optimization, we have to make decisions before all data are known.
Online problems are often real-time.

common assumption: nothing known about the future, not even stochastical information

Online algorithms

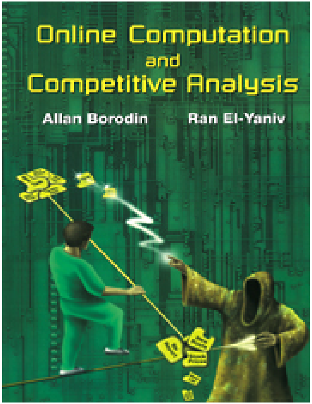
An online algorithm is a method to make a decision as soon as some new information becomes known.

Online algorithms

An online algorithm is a method to make a decision as soon as some new information becomes known.

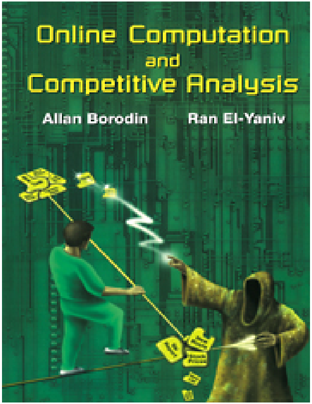
Usual theoretical assumptions:

- The decision is irrevocable.
- Running time does not matter.



Online algorithms

An online algorithm is a method to make a decision as soon as some new information becomes known.



Usual theoretical assumptions:

- The decision is irrevocable.
- Running time does not matter.

For our real-world applications:

- Decisions are (partially) revocable.
- Running time does matter.

Outline

Recap: Online optimization

Theoretical framework: Online Dial-a-Ride problems and competitive analysis

Online Optimization in Practice: Reoptimization Algorithms
- Dispatching the service vehicles of ADAC
- Controlling cargo elevators in a distribution center
- Controlling passenger elevators in high-rise buildings

Theory again: The Online Bin Coloring problem

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w \colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w: A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Solution: Schedule for a server

- A move is a triple $m = (s, t, L)$ with $(s, t) \in A$ and $L \subseteq R$.

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Solution: Schedule for a server

- A move is a triple $m = (s, t, L)$ with $(s, t) \in A$ and $L \subseteq R$.
- A schedule is a sequence of moves $\big(m_i = (s_i, t_i, L_i)\big)_{0 \leq i \leq k}$ with

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Solution: Schedule for a server

- A move is a triple $m = (s, t, L)$ with $(s, t) \in A$ and $L \subseteq R$.
- A schedule is a sequence of moves $\big(m_i = (s_i, t_i, L_i)\big)_{0 \leq i \leq k}$ with
    1. The first move starts and the last move ends at d.

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w \colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Solution: Schedule for a server

- A move is a triple $m = (s, t, L)$ with $(s, t) \in A$ and $L \subseteq R$.
- A schedule is a sequence of moves $\big(m_i = (s_i, t_i, L_i)\big)_{0 \leq i \leq k}$ with
  1. The first move starts and the last move ends at d.
  2. Move m_{i+1} starts at the target node t_i of its predecessor move m_i.

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Solution: Schedule for a server

- A move is a triple $m = (s, t, L)$ with $(s, t) \in A$ and $L \subseteq R$.
- A schedule is a sequence of moves $\big(m_i = (s_i, t_i, L_i)\big)_{0 \leq i \leq k}$ with
  1. The first move starts and the last move ends at d.
  2. Move m_{i+1} starts at the target node t_i of its predecessor move m_i.
  3. The requests $L_{i+1} \setminus L_i$ start at node t_i (are picked up) and the requests $L_i \setminus L_{i+1}$ end at node t_i (are dropped).

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Solution: Schedule for a server

- A move is a triple $m = (s, t, L)$ with $(s, t) \in A$ and $L \subseteq R$.
- A schedule is a sequence of moves $\big(m_i = (s_i, t_i, L_i)\big)_{0 \leq i \leq k}$ with
  1. The first move starts and the last move ends at d.
  2. Move m_{i+1} starts at the target node t_i of its predecessor move m_i.
  3. The requests $L_{i+1} \setminus L_i$ start at node t_i (are picked up) and the requests $L_i \setminus L_{i+1}$ end at node t_i (are dropped).
  4. Every request is picked up and dropped exactly once.

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Solution: Schedule for a server

- A move is a triple $m = (s, t, L)$ with $(s, t) \in A$ and $L \subseteq R$.
- A schedule is a sequence of moves $\big(m_i = (s_i, t_i, L_i)\big)_{0 \leq i \leq k}$.
- The time needed by a move $m = (s, t, L)$ is $w(s, t)$.

Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
- set R of node pairs (s_j, t_j), called requests

Solution: Schedule for a server

- A move is a triple $m = (s, t, L)$ with $(s, t) \in A$ and $L \subseteq R$.
- A schedule is a sequence of moves $\big(m_i = (s_i, t_i, L_i)\big)_{0 \leq i \leq k}$.
- The time needed by a move $m = (s, t, L)$ is $w(s, t)$.

Objective

- Find a schedule with minimum completion time.

Online Dial-a-Ride Problems

- system operates continuously, requests arrive over time
- online control algorithm does not know anything about future requests, note even their number
- schedule for server may change in response to new requests
- assumption: schedule may only change when the server is at a node

Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$

Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence σ of requests (s_j, t_j, τ_j), τ_j is release time

Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence σ of requests (s_j, t_j, τ_j), τ_j is release time

Solution: Online schedule for a server

- An online schedule is a sequence of moves with

Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence σ of requests (s_j, t_j, τ_j), τ_j is release time

Solution: Online schedule for a server

- An online schedule is a sequence of moves with
  1. The sequence is a feasible (offline) schedule.

Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w : A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence σ of requests (s_j, t_j, τ_j), τ_j is release time

Solution: Online schedule for a server

- An online schedule is a sequence of moves with
  1. The sequence is a feasible (offline) schedule.
  2. A request is not picked up before its release time τ_j.

Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w: A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence σ of requests (s_j, t_j, τ_j), τ_j is release time

Solution: Online schedule for a server

- An online schedule is a sequence of moves with
  1. The sequence is a feasible (offline) schedule.
  2. A request is not picked up before its release time τ_j.

Objectives

- Find a schedule with minimum completion time.

Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w\colon A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence σ of requests (s_j, t_j, τ_j), τ_j is release time

Solution: Online schedule for a server

- An online schedule is a sequence of moves with
  1. The sequence is a feasible (offline) schedule.
  2. A request is not picked up before its release time τ_j.

Objectives

- Find a schedule with minimum completion time.
- Find a schedule with minimum average/maximum waiting time.

Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w \colon A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence σ of requests (s_j, t_j, τ_j), τ_j is release time

Solution: Online schedule for a server

- An online schedule is a sequence of moves with
  1. The sequence is a feasible (offline) schedule.
  2. A request is not picked up before its release time τ_j.

Objectives

- Find a schedule with minimum completion time.
- Find a schedule with minimum average/maximum waiting time.
- Find a schedule with minimum average/maximum flow time.

Offline Dial-a-Ride problem

- The problem on general graphs is NP-hard (contains TSP).

Quality of Dial-a-Ride algorithms

Offline Dial-a-Ride problem

- The problem on general graphs is NP-hard (contains TSP).
- Algorithm Alg is called an α-approximation algorithm if

$$\textsc{Alg}(G, d, R) \leq \alpha \cdot \textsc{Opt}(G, d, R)$$

for all instances (G, d, R).

Quality of Dial-a-Ride algorithms

Offline Dial-a-Ride problem

- The problem on general graphs is NP-hard (contains TSP).
- Algorithm $\textsc{Alg}$ is called an α-approximation algorithm if

$$\textsc{Alg}(G, d, R) \leq \alpha \cdot \textsc{Opt}(G, d, R)$$

for all instances (G, d, R).

Online Dial-a-Ride problem

- An online algorithm $\textsc{Alg}$ is called a c-competitive algorithm if

$$\textsc{Alg}(G, d, R) \leq c \cdot \textsc{Opt}(G, d, R)$$

for all instances (G, d, R). Here, $\textsc{Opt}$ is the optimal offline algorithm that has access to the entire sequence, i.e. knows the future.

Construction of online algorithms

- for simple online problems like bin packing, special online algorithms have been designed

Construction of online algorithms

- for simple online problems like bin packing, special online algorithms have been designed
- for complex online problems like Dial-a-Ride, one wants to make use of offline algorithms

Construction of online algorithms

Replan-strategy

- As a new request arrives, compute a new schedule for the current set of requests using offline algorithm Alg.
- Replace the old schedule by the new one and follow this schedule until it is finished or replaced.

Construction of online algorithms

Replan-strategy

- As a new request arrives, compute a new schedule for the current set of requests using offline algorithm Alg.
- Replace the old schedule by the new one and follow this schedule until it is finished or replaced.

Ignore-strategy

- As a request arrives, serve it if the server is idle. If the server is not idle, ignore the request and complete the current schedule.
- As the server becomes idle, compute a schedule for the requests ignored so far using offline algorithm Alg and follow this schedule.

Competitive analysis results

Completion time [Ascheuer, Krumke, Rambau '00, Ausiello et al '01]

- Both REPLAN and IGNORE are 5/2-competitive if they use an optimal offline algorithm.

Competitive analysis results

Completion time [Ascheuer, Krumke, Rambau '00, Ausiello et al '01]

- Both REPLAN and IGNORE are 5/2-competitive if they use an optimal offline algorithm.
- REPLAN and IGNORE yield $5\alpha/2$-competitive algorithms if they employ an α-approximation algorithm.

Competitive analysis results

Completion time [Ascheuer, Krumke, Rambau '00, Ausiello et al '01]

- Both Replan and Ignore are 5/2-competitive if they use an optimal offline algorithm.
- Replan and Ignore yield $5\alpha/2$-competitive algorithms if they employ an α-approximation algorithm.
- There is an online algorithm that is 2-competitive.

Competitive analysis results

Completion time [Ascheuer, Krumke, Rambau '00, Ausiello et al '01]

- Both REPLAN and IGNORE are 5/2-competitive if they use an optimal offline algorithm.
- REPLAN and IGNORE yield $5\alpha/2$-competitive algorithms if they employ an α-approximation algorithm.
- There is an online algorithm that is 2-competitive.
- No (deterministic) online algorithm can be better than 2-competitive.

Competitive analysis results

Completion time [Ascheuer, Krumke, Rambau '00, Ausiello et al '01]

- Both Replan and Ignore are 5/2-competitive if they use an optimal offline algorithm.
- Replan and Ignore yield $5\alpha/2$-competitive algorithms if they employ an α-approximation algorithm.
- There is an online algorithm that is 2-competitive.
- No (deterministic) online algorithm can be better than 2-competitive.

Average/maximum flow/waiting time

- There is no c-competitive online algorithm for any $c \geq 1$.

Proof

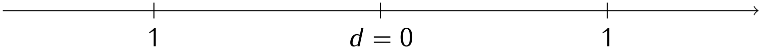
- Want to show: There is no c-competitive online algorithm for any $c \geq 1$.

Proof

- Want to show: There is no c-competitive online algorithm for any $c \geq 1$.
- Meaning: For any online algorithm $\textsc{Alg}$ and any $c \geq 1$, there is a sequence $\sigma^{\textsc{Alg}}$ s. t. $\textsc{Alg}^{\text{max-flow}}(\sigma^{\textsc{Alg}}) \geq c \cdot \text{OPT}^{\text{max-flow}}(\sigma^{\textsc{Alg}})$.

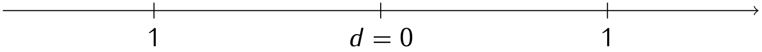
Proof

- Want to show: There is no c-competitive online algorithm for any $c \geq 1$.
- Meaning: For any online algorithm $\textsc{Alg}$ and any $c \geq 1$, there is a sequence $\sigma^{\textsc{Alg}}$ s. t. $\textsc{Alg}^{\text{max-flow}}(\sigma^{\textsc{Alg}}) \geq c \cdot \text{OPT}^{\text{max-flow}}(\sigma^{\textsc{Alg}})$.
- Consider the real line and a server moving at unit speed.



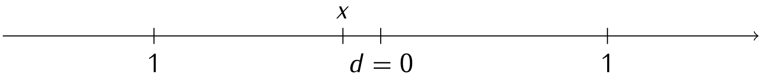
Proof

- Want to show: There is no c-competitive online algorithm for any $c \geq 1$.
- Meaning: For any online algorithm ALG and any $c \geq 1$, there is a sequence σ^{ALG} s. t. $\text{ALG}^{\text{max-flow}}(\sigma^{\text{ALG}}) \geq c \cdot \text{OPT}^{\text{max-flow}}(\sigma^{\text{ALG}})$.
- Consider the real line and a server moving at unit speed.
  Let x be position of the server controlled by ALG at time 1.



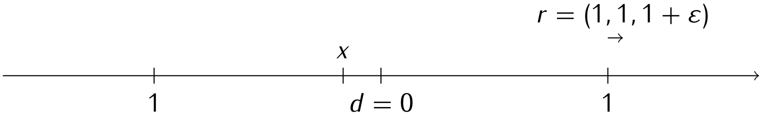
Proof

- Want to show: There is no c-competitive online algorithm for any $c \geq 1$.
- Meaning: For any online algorithm Alg and any $c \geq 1$, there is a sequence $\sigma^{\textsc{Alg}}$ s. t. $\textsc{Alg}^{\text{max-flow}}(\sigma^{\textsc{Alg}}) \geq c \cdot \text{OPT}^{\text{max-flow}}(\sigma^{\textsc{Alg}})$.
- Consider the real line and a server moving at unit speed.
  Let x be position of the server controlled by Alg at time 1.
- Case 1: $x \leq 0$



Proof

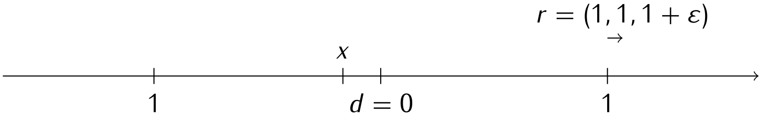
- Want to show: There is no c-competitive online algorithm for any $c \geq 1$.
- Meaning: For any online algorithm ALG and any $c \geq 1$, there is a sequence σ^{ALG} s. t. $\text{ALG}^{\text{max-flow}}(\sigma^{\text{ALG}}) \geq c \cdot \text{OPT}^{\text{max-flow}}(\sigma^{\text{ALG}})$.
- Consider the real line and a server moving at unit speed. Let x be position of the server controlled by ALG at time 1.
- Case 1: $x \leq 0 \quad \rightsquigarrow \sigma^{\text{ALG}} = \big(r = (1, 1, 1 + \varepsilon)\big)$.



Proof

- Want to show: There is no c-competitive online algorithm for any $c \geq 1$.
- Meaning: For any online algorithm $\textsc{Alg}$ and any $c \geq 1$, there is a sequence $\sigma^{\textsc{Alg}}$ s. t. $\textsc{Alg}^{\max-\text{flow}}(\sigma^{\textsc{Alg}}) \geq c \cdot \text{OPT}^{\max-\text{flow}}(\sigma^{\textsc{Alg}})$.
- Consider the real line and a server moving at unit speed. Let x be position of the server controlled by $\textsc{Alg}$ at time 1.
- Case 1: $x \leq 0 \quad \rightsquigarrow \sigma^{\textsc{Alg}} = \big(r = (1, 1, 1 + \varepsilon)\big)$.

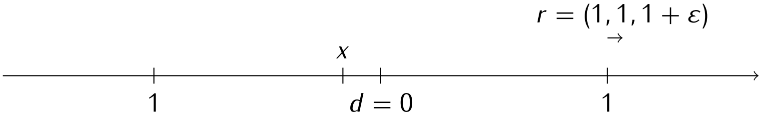
$$\textsc{Alg}^{\max-\text{flow}}(\sigma^{\textsc{Alg}}) \geq 1 + \varepsilon$$



Proof

- Want to show: There is no c-competitive online algorithm for any $c \geq 1$.
- Meaning: For any online algorithm $\textsc{Alg}$ and any $c \geq 1$, there is a sequence $\sigma^{\textsc{Alg}}$ s. t. $\textsc{Alg}^{\max-\text{flow}}(\sigma^{\textsc{Alg}}) \geq c \cdot \text{OPT}^{\max-\text{flow}}(\sigma^{\textsc{Alg}})$.
- Consider the real line and a server moving at unit speed.
  Let x be position of the server controlled by $\textsc{Alg}$ at time 1.
- Case 1: $x \leq 0 \quad \rightsquigarrow \sigma^{\textsc{Alg}} = \big(r = (1, 1, 1 + \varepsilon)\big)$.

$$\textsc{Alg}^{\max-\text{flow}}(\sigma^{\textsc{Alg}}) \geq 1 + \varepsilon$$
$$\text{OPT}^{\max-\text{flow}}(\sigma^{\textsc{Alg}}) = \varepsilon$$



Alternatives to pure competitive analysis

Theory

deterministic

- restrict input sequences
- resource augmentation
- max/max ratio
- worst order ratio
- bijective analysis
- ...many more

probabilistic

- randomized online algorithms
- average case analysis
- diffuse adversaries
- smoothed analysis
- stochastic dominance
- ...many more

Practice

- simulation
- Markov Decision Process policy evaluation

Alternatives to pure competitive analysis

Theory

deterministic

- restrict input sequences
- resource augmentation
- max/max ratio
- worst order ratio
- bijective analysis
- ...many more

probabilistic

- randomized online algorithms
- average case analysis
- diffuse adversaries
- smoothed analysis
- stochastic dominance
- ...many more

Practice

- simulation
- Markov Decision Process policy evaluation

Alternative analysis: Reasonable load

Observation

- request sequences are not restricted: arbitrarily many requests may arrive in a short period of time

Alternative analysis: Reasonable load

Observation

- request sequences are not restricted: arbitrarily many requests may arrive in a short period of time
- continuously operating systems should exhibit stability

Alternative analysis: Reasonable load

Observation

- request sequences are not restricted: arbitrarily many requests may arrive in a short period of time
- continuously operating systems should exhibit stability
- Queueing theory: arrival rate $\mu <$ service rate ϱ

Alternative analysis: Reasonable load

Observation

- request sequences are not restricted: arbitrarily many requests may arrive in a short period of time
- continuously operating systems should exhibit stability
- Queueing theory: arrival rate $\mu <$ service rate ϱ

Worst-case model: Reasonable load

- For request set R, let $\delta(R)$ denote the time span of R.

Alternative analysis: Reasonable load

Observation

- request sequences are not restricted: arbitrarily many requests may arrive in a short period of time
- continuously operating systems should exhibit stability
- Queueing theory: arrival rate $\mu <$ service rate ϱ

Worst-case model: Reasonable load

- For request set R, let $\delta(R)$ denote the time span of R.
- A request set R is called Δ-reasonable if

  For all $R' \subseteq R$ with $\delta(R') \geq \Delta$ we have

$$\textsc{Opt}^{\text{comp}}(R') \leq \delta(R').$$

Alternative analysis: Reasonable load

Observation

- request sequences are not restricted: arbitrarily many requests may arrive in a short period of time
- continuously operating systems should exhibit stability
- Queueing theory: arrival rate $\mu <$ service rate ϱ

Worst-case model: Reasonable load

- For request set R, let $\delta(R)$ denote the time span of R.
- A request set R is called Δ-reasonable if

  For all $R' \subseteq R$ with $\delta(R') \geq \Delta$ we have

$$\textsc{Opt}^{\text{comp}}(R') \leq \delta(R').$$

- A request sequence σ is called Δ-reasonable if the set of all requests in σ is Δ-reasonable.

Results for Dial-a-Ride under reasonable load

Theorem ([Hauptmeier, Krumke, Rambau '00])

1. *Let σ be a Δ-reasonable request sequence. Then maximal flow time achieved by* IGNORE *on σ is **bounded by** 2Δ.*

Results for Dial-a-Ride under reasonable load

Theorem ([Hauptmeier, Krumke, Rambau '00])

1. *Let σ be a Δ-reasonable request sequence. Then maximal flow time achieved by* Ignore *on σ is bounded by 2Δ.*
2. *There is a $\Delta \in \mathbb{R}$, a Δ-reasonable request sequence σ and a request r in σ such that the flow time for r achieved by* Replan *is unbounded.*

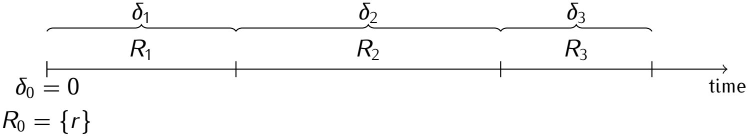
Proof of 2Δ bound for IGNORE

- Let r be the first request arriving at time $\delta_0 = 0$.

Proof of 2Δ bound for IGNORE

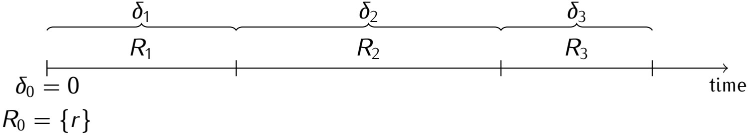
- Let r be the first request arriving at time $\delta_0 = 0$.
- Define
  - $R_0 = \{r\}$

Proof of 2Δ bound for IGNORE



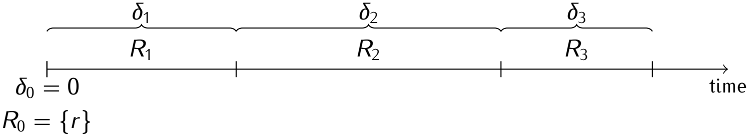
- Let r be the first request arriving at time $\delta_0 = 0$.
- Define
  - $R_0 = \{r\}$
  - $\delta_{i+1} :=$ time needed to serve R_i
  - $R_{i+1} :=$ set of requests arriving during serving R_i

Proof of 2Δ bound for IGNORE



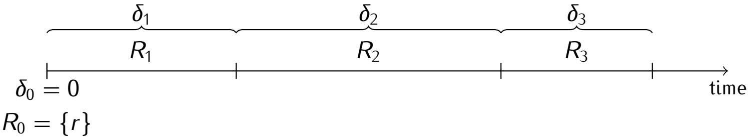
- Let r be the first request arriving at time $\delta_0 = 0$.
- Define
  - $R_0 = \{r\}$
  - $\delta_{i+1} :=$ time needed to serve R_i
  - $R_{i+1} :=$ set of requests arriving during serving R_i
- observation: $\text{IGNORE}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1}$

Proof of 2Δ bound for IGNORE



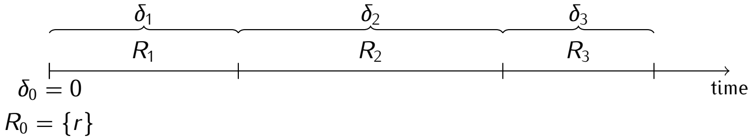
- Let r be the first request arriving at time $\delta_0 = 0$.
- Define
  - $R_0 = \{r\}$
  - $\delta_{i+1} :=$ time needed to serve R_i
  - $R_{i+1} :=$ set of requests arriving during serving R_i
- observation: $\textsc{Ignore}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1} \leq 2\Delta$
- suffices to show: $\delta_i \leq \Delta$ for all i

Proof of 2Δ bound for Ignore



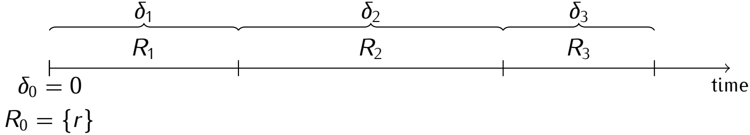
- Let r be the first request arriving at time $\delta_0 = 0$.
- Define
  - $R_0 = \{r\}$
  - $\delta_{i+1} :=$ time needed to serve R_i
  - $R_{i+1} :=$ set of requests arriving during serving R_i
- observation: $\textsc{Ignore}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1}$
- suffices to show: $\delta_i \leq \Delta$ for all i
- we have: $\delta_{i+1} = \textsc{Opt}^{\text{comp}}(R_i)$

Proof of 2Δ bound for IGNORE



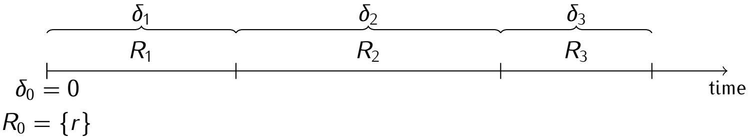
- Let r be the first request arriving at time $\delta_0 = 0$.
- Define
  - $R_0 = \{r\}$
  - $\delta_{i+1} :=$ time needed to serve R_i
  - $R_{i+1} :=$ set of requests arriving during serving R_i
- observation: $\textsc{Ignore}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1}$
- suffices to show: $\delta_i \leq \Delta$ for all i
- we have: $\delta_{i+1} = \textsc{Opt}^{\text{comp}}(R_i) \leq \delta(R_i) \leq \delta_i$

Proof of 2Δ bound for IGNORE



- Let r be the first request arriving at time $\delta_0 = 0$.
- Define
  - $R_0 = \{r\}$
  - $\delta_{i+1} :=$ time needed to serve R_i
  - $R_{i+1} :=$ set of requests arriving during serving R_i
- observation: $\textsc{Ignore}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1}$
- suffices to show: $\delta_i \leq \Delta$ for all i
- we have: $\delta_{i+1} = \textsc{Opt}^{\text{comp}}(R_i) \leq \delta(R_i) \leq \delta_i \leq \max\{\Delta, \delta_i\}$

Proof of 2Δ bound for IGNORE



- Let r be the first request arriving at time $\delta_0 = 0$.
- Define
  - $R_0 = \{r\}$
  - $\delta_{i+1} :=$ time needed to serve R_i
  - $R_{i+1} :=$ set of requests arriving during serving R_i
- observation: $\textsc{Ignore}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1} \leq 2\Delta$
- suffices to show: $\delta_i \leq \Delta$ for all i
- we have: $\delta_{i+1} = \textsc{Opt}^{\text{comp}}(R_i) \leq \delta(R_i) \leq \delta_i \leq \max\{\Delta, \delta_i\} \leq \Delta$.

Outline

Recap: Online optimization

Theoretical framework: Online Dial-a-Ride problems and competitive analysis

Online Optimization in Practice: Reoptimization Algorithms

- Dispatching the service vehicles of ADAC
- Controlling cargo elevators in a distribution center
- Controlling passenger elevators in high-rise buildings

Theory again: The Online Bin Coloring problem

Outline

Recap: Online optimization

Theoretical framework: Online Dial-a-Ride problems and competitive analysis

Online Optimization in Practice: Reoptimization Algorithms

- Dispatching the service vehicles of ADAC
- Controlling cargo elevators in a distribution center
- Controlling passenger elevators in high-rise buildings

Theory again: The Online Bin Coloring problem

Application #1: Dispatching the service vehicles of ADAC

Supereinsatz. In Berlin und Brandenburg mussten die Gelben Engel letztes Jahr mehr als 240 000 Mal ausrücken, um Havaristen in der Hauptstadt und auf 1700 Autobahnkilometern wieder flottzumachen – ein Rekordeinsatz. Einen Rückgang von zehn Prozent bei den Pannen registrierten dagegen die Gelben Engel in Mecklenburg-Vorpommern. Bei insgesamt 72 389 Einsätzen schafften sie jedoch auch einen Rekord: In 84 Prozent der Fälle konnten die Autofahrer mit dem Wagen weiterfahren.

Application #1: Dispatching the service vehicles of ADAC

Application #1: Dispatching the service vehicles of ADAC

phone call →

help center

Application #1: Dispatching the service vehicles of ADAC

phone call

help center

dispatch:
human operator

Application #1: Dispatching the service vehicles of ADAC

phone call

ADAC

help center

wireless transmission

Yellow Angel

dispatch:
human operator

Application #1: Dispatching the service vehicles of ADAC

phone call

help center

service

wireless transmission

dispatch:
human operator

Yellow Angel

ADAC break down service

Situation

- $\approx$ 1,700 Yellow Angels
- $\approx$ 1,200 road service partners with $\approx$ 5,000 vehicles
- 5 help centers
- on average 10,000 requests a day; peak: 45,000 requests in 4 hours

Task

Determine assignment of the requests to units and partners. Schedule corresponding tours for the units online and in real-time (computation time $\leq$ 10 seconds).

Goal

Minimize operating cost + lateness cost.

Difficulties and solution strategies

- The ADAC dispatching problem is an online problem.
  $\Rightarrow$ Reoptimization algorithm: At certain decision times, compute a (good) dispatch for the current situation.

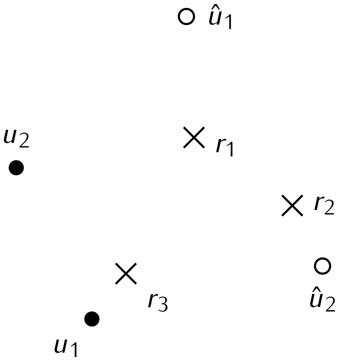
Difficulties and solution strategies

- The ADAC dispatching problem is an online problem.
  ⇒ Reoptimization algorithm: At certain decision times, compute a (good) dispatch for the current situation.
- The offline problem is NP-hard and there are real time requirements.
  ⇒ Abandon optimality and use efficient, problem-specific approximation algorithm based on an exact approach.

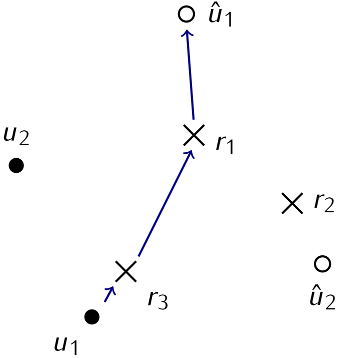
Difficulties and solution strategies

- The ADAC dispatching problem is an online problem.
  $\Rightarrow$ Reoptimization algorithm: At certain decision times, compute a (good) dispatch for the current situation.
- The offline problem is NP-hard and there are real time requirements.
  $\Rightarrow$ Abandon optimality and use efficient, problem-specific approximation algorithm based on an exact approach.
- Cost structure is complex: nonlinear lateness cost, discounted partner costs, ...
  $\Rightarrow$ tour-based model

Tour-based model: Details

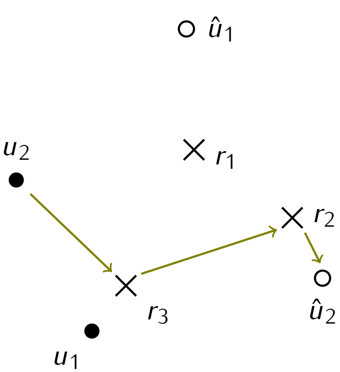


Tour-based model: Details



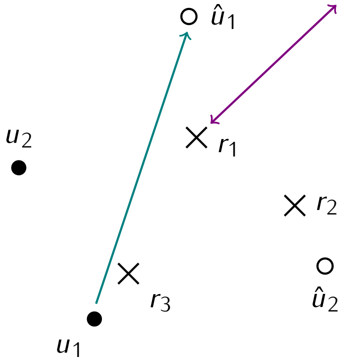
t_1	
1	r_1
0	r_2
1	r_3
1	u_1
0	u_2
c_{t_1}	

Tour-based model: Details



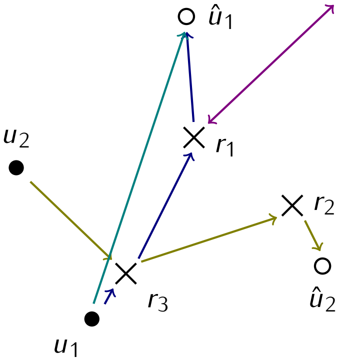
t_1	t_2	
1	0	r_1
0	1	r_2
1	1	r_3
1	0	u_1
0	1	u_2
c_{t_1}	c_{t_2}	

Tour-based model: Details



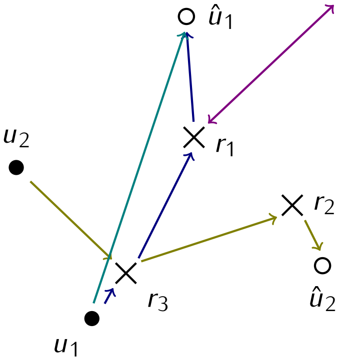
t_1	t_2	t_3	t_4	
1	0	0	1	r_1
0	1	0	0	r_2
1	1	0	0	r_3
1	0	1	0	u_1
0	1	0	0	u_2
c_{t_1}	c_{t_2}	c_{t_3}	c_{t_4}	

Tour-based model: Details



	t_1	t_2	t_3	t_4	...	t_N	
	1	0	0	1	...		r_1
	0	1	0	0	...		r_2
	1	1	0	0	...		r_3
	1	0	1	0	...		u_1
	0	1	0	0	...		u_2
	c_{t_1}	c_{t_2}	c_{t_3}	c_{t_4}	...	c_{t_N}	

Tour-based model: Details



	t_1	t_2	t_3	t_4	$\dots$	t_N	
	1	0	0	1	$\dots$		r_1
	0	1	0	0	$\dots$		r_2
A	1	1	0	0	$\dots$		r_3
	1	0	1	0	$\dots$		u_1
	0	1	0	0	$\dots$		u_2
c^T	c_{t_1}	c_{t_2}	c_{t_3}	c_{t_4}	$\dots$	c_{t_N}	
x^T	x_{t_1}	x_{t_2}	x_{t_3}	x_{t_4}	$\dots$	x_{t_N}	

obtain set partitioning IP:

$$\begin{aligned} \min \quad & c^T x \\ Ax &= 1 \\ x &\in \{0,1\}^n \end{aligned}$$

The reoptimization algorithm

Online problem

REPLAN strategy: When new information becomes known, compute new dispatch by solving an instance of the offline problem (the snapshot problem).

The reoptimization algorithm

Online problem

Replan strategy: When new information becomes known, compute new dispatch by solving an instance of the offline problem (the snapshot problem).

Offline problem

- extremely many feasible tours (= columns of A)
  $\Rightarrow$ dynamic column generation

The reoptimization algorithm

Online problem

REPLAN strategy: When new information becomes known, compute new dispatch by solving an instance of the offline problem (the snapshot problem).

Offline problem

- extremely many feasible tours (= columns of A)
  $\Rightarrow$ dynamic column generation
- column generation using Branch&Bound with special search strategy

The reoptimization algorithm

Online problem

REPLAN strategy: When new information becomes known, compute new dispatch by solving an instance of the offline problem (the snapshot problem).

Offline problem

- extremely many feasible tours (= columns of A)
  $\Rightarrow$ dynamic column generation
- column generation using Branch&Bound with special search strategy
- initial IP model contains
  - return-home tour for each unit
  - single request tour for each partner
  - a feasible dispatch from a simple best-insertion heuristic based on the current dispatch

The reoptimization algorithm

Online problem

REPLAN strategy: When new information becomes known, compute new dispatch by solving an instance of the offline problem (the snapshot problem).

Offline problem

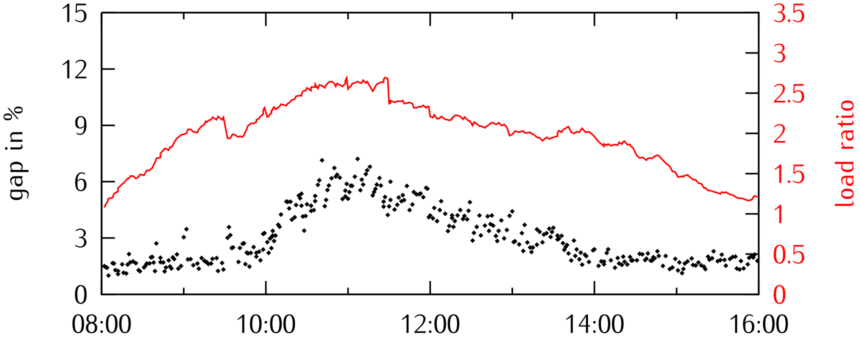
- extremely many feasible tours (= columns of A)
  $\Rightarrow$ dynamic column generation
- column generation using Branch&Bound with special search strategy
- initial IP model contains
  - return-home tour for each unit
  - single request tour for each partner
  - a feasible dispatch from a simple best-insertion heuristic based on the current dispatch

  $\Rightarrow$ reasonable dual prices

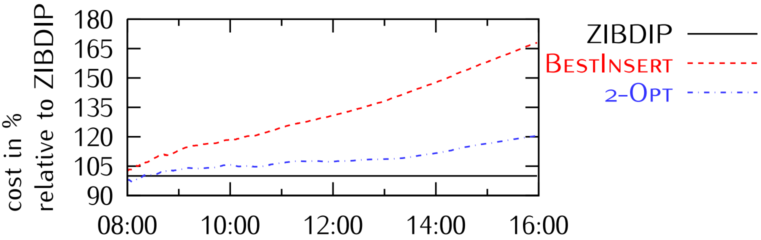
Data and simulation setup

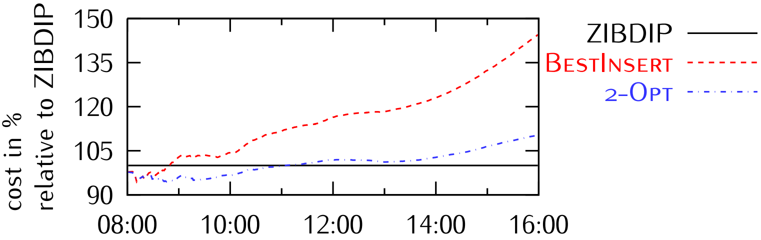
- ADAC provided production data of several days
  $\Rightarrow$ input for our online simulation
- reoptimization everytime a new request arrived
- computation time ZIBDIP: 10 seconds
- computer: Xeon 2.4 GHz, 2 GB RAM
- details and further results in [Hiller, Krumke, Rambau '06]

Snapshot solution quality over the day

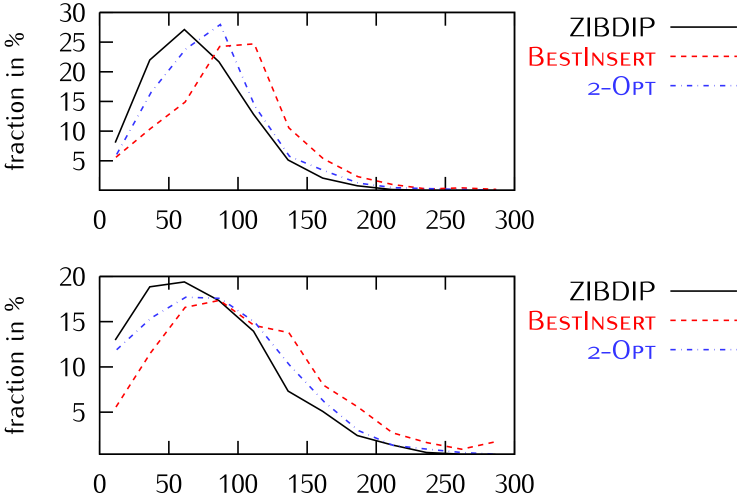


Comparison with heuristics over the day





Comparison waiting times (in minutes)



Outline

Recap: Online optimization

Theoretical framework: Online Dial-a-Ride problems and competitive analysis

Online Optimization in Practice: Reoptimization Algorithms

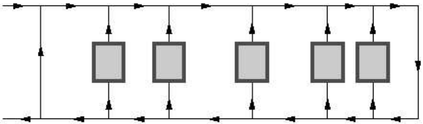
- Dispatching the service vehicles of ADAC
- Controlling cargo elevators in a distribution center
- Controlling passenger elevators in high-rise buildings

Theory again: The Online Bin Coloring problem

Movie

Application #2: Controlling cargo elevators in a distribution center

Application: elevators in a Herlitz high rack warehouse



idealized setup in [Friese, Rambau '06]:

- global queue for every floor
- local queue for every elevator on each floor
- pallet takes fixed time to travel from global to each local queue

Overview on considered control algorithms

Assign-first, route-second algorithms

FIFO elevator assignment according to round-robin; each elevator serves its requests in FIFO order

NN request is assigned to the elevator having least load including the new request; elevator serves nearest request next

REPLAN elevator assignment as NN; requests scheduled optimally w. r. t. completion time

Overview on considered control algorithms

Assign-first, route-second algorithms

FIFO elevator assignment according to round-robin; each elevator serves its requests in FIFO order

NN request is assigned to the elevator having least load including the new request; elevator serves nearest request next

REPLAN elevator assignment as NN; requests scheduled optimally w. r. t. completion time

Integrated algorithms

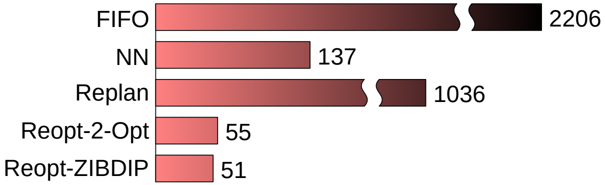
Reopt-2-OPT determines dispatch using a 2-exchange-heuristic

Reopt-ZIBDIP determines dispatch using modified ADAC algorithm ZIBDIP

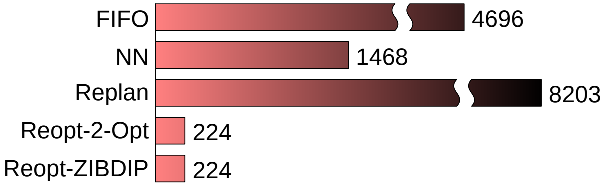
The simulated elevator system features 5 elevators and 16 floors.

Results

average waiting time (seconds)



maximum waiting time (seconds)



Results

- extremely high waiting times are due to instability of the system: number of unserved requests is ever-increasing

Results

- extremely high waiting times are due to instability of the system: number of unserved requests is ever-increasing
- control algorithm has a strong influence on the load a system can handle:

  FIFO 8 elevators needed

  NN 6 elevators needed

  Reopt-ZIBDIP 5 elevators needed

Results

- extremely high waiting times are due to instability of the system: number of unserved requests is ever-increasing
- control algorithm has a strong influence on the load a system can handle:

  FIFO 8 elevators needed

  NN 6 elevators needed

  Reopt-ZIBDIP 5 elevators needed
- for much more detailed results see [Friese, Rambau '06]:

Outline

Recap: Online optimization

Theoretical framework: Online Dial-a-Ride problems and competitive analysis

Online Optimization in Practice: Reoptimization Algorithms
- Dispatching the service vehicles of ADAC
- Controlling cargo elevators in a distribution center
- Controlling passenger elevators in high-rise buildings

Theory again: The Online Bin Coloring problem

Application #3: Controlling passenger elevators in high-rise buildings

Elements: floors, elevators, passengers, travel calls

Task
Serve all travel calls.

Goal
Small waiting and travel times.

Waiting time:
time for passenger to wait

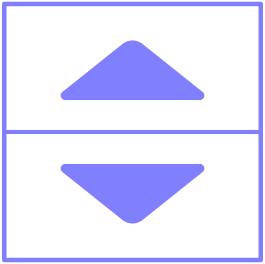
Travel time:
time for passenger to arrive at destination

Conventional System

Two-step input for each passenger:

- Outside car: landing call providing
  - start floor
  - travel direction
  - landing call release time

  unknown: number of passengers, actual passenger arrival times, destination floor(s)



Conventional System

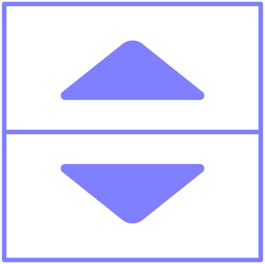
Two-step input for each passenger:

- Outside car: landing call providing
  - start floor
  - travel direction
  - landing call release time

  unknown: number of passengers, actual passenger arrival times, destination floor(s)

- Inside car: car call providing
  - destination floor

  unknown: number of passengers



Destination Call System

Outside car:
destination call providing

- start floor
- destination floor
- release time

Destination Call System

Outside car:
destination call providing

- start floor
- destination floor
- release time

Known at destination call release time:

- destination floor
- number of passengers*
- actual arrival time of each passenger*

No possibility to confirm destination inside car (no panel).

Two Variants of Destination Call Systems

Immediate Assignment (IA)

- after issuing a call, the passenger is immediately assigned to a serving elevator
- used by ThyssenKrupp, Schindler, Kollmorgen Steuerungstechnik

Two Variants of Destination Call Systems

Immediate Assignment (IA)

- after issuing a call, the passenger is immediately assigned to a serving elevator
- used by ThyssenKrupp, Schindler, Kollmorgen Steuerungstechnik

Delayed Assignment (DA)

- issuing a call and assignment of an elevator are separate steps
- each elevator signals shortly before arrival the floors it is going to serve, i. e., which passenger should board
- not yet implemented

Properties of passenger elevator systems

1. A passenger never moves in opposite direction of his travel direction (no turns)
   - Passenger boards only if elevator departs in travel direction
   - Elevator control has to ensure drop stops before changing direction

Properties of passenger elevator systems

1. A passenger never moves in opposite direction of his travel direction (no turns)
   - Passenger boards only if elevator departs in travel direction
   - Elevator control has to ensure drop stops before changing direction
2. Elevator stops at a floor: no control which passengers board car
   - enough capacity: all (assigned) waiting passengers with matching travel direction / destination floor board
   - not enough capacity: an arbitrary unknown subset; remaining passengers reissue travel call

Request model for elevator schedules

We employ a general model for elevator schedules applicable to all systems (conventional, IA, DA).

Request model for elevator schedules

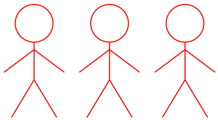
We employ a general model for elevator schedules applicable to all systems (conventional, IA, DA).

Request A set of calls that an elevator has to pickup together.
A request expresses that these calls can only be served by the same elevator.

Request model for elevator schedules

We employ a general model for elevator schedules applicable to all systems (conventional, IA, DA).

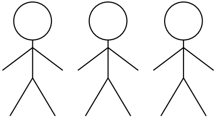
Request A set of calls that an elevator has to pickup together.
A request expresses that these calls can only be served by the same elevator.



Request model for elevator schedules

We employ a general model for elevator schedules applicable to all systems (conventional, IA, DA).

Request A set of calls that an elevator has to pickup together.
A request expresses that these calls can only be served by the same elevator.



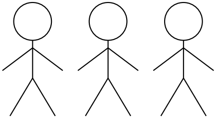
$\longrightarrow$

$$c_1 : 1 \to 5$$
$$c_2 : 1 \to 6$$
$$c_3 : 1 \to 5$$

Request model for elevator schedules

We employ a general model for elevator schedules applicable to all systems (conventional, IA, DA).

Request A set of calls that an elevator has to pickup together.
A request expresses that these calls can only be served by the same elevator.

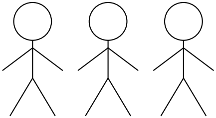


$$\longrightarrow \quad \begin{array}{l} c_1 : 1 \to 5 \\ c_2 : 1 \to 6 \\ c_3 : 1 \to 5 \end{array} \quad \xrightarrow{\text{IA}} \quad r = \{c_1, c_2, c_3\}$$

Request model for elevator schedules

We employ a general model for elevator schedules applicable to all systems (conventional, IA, DA).

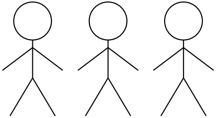
Request A set of calls that an elevator has to pickup together.
A request expresses that these calls can only be served by the same elevator.



$$\longrightarrow \quad \begin{array}{l} c_1 : 1 \to 5 \\ c_2 : 1 \to 6 \\ \boxed{c_3 : 1 \to 5} \end{array} \quad \xrightarrow{\text{IA}} \quad \begin{array}{l} r_1 = \{c_1, c_2\} \\ r_2 = \{c_3\} \end{array}$$

We employ a general model for elevator schedules applicable to all systems (conventional, IA, DA).

Request A set of calls that an elevator has to pickup together.
A request expresses that these calls can only be served by the same elevator.



$$\longrightarrow \boxed{\begin{array}{l} c_1 : 1 \to 5 \\ c_2 : 1 \to 6 \\ c_3 : 1 \to 5 \end{array}} \xrightarrow{\text{DA}} \begin{array}{l} r_1 = \{c_1, c_3\} \\ r_2 = \{c_2\} \end{array}$$

Exact Approach via Set Partitioning

- compute dispatch by solving set partitioning formulation

$$
\begin{array}{llll}
\min & \displaystyle\sum_{S\in\mathcal{S}} c_S x_S & & \text{cost } c_S \text{ of a schedule } S \text{ is weighted sum of waiting and travel times} \\
& \displaystyle\sum_{S\in\mathcal{S}\colon r\in S} x_S = 1 & r \in \mathcal{R}_u & \text{every unassigned request } r \in \mathcal{R}_u \text{ is served} \\
& \displaystyle\sum_{S\in\mathcal{S}_e} x_S = 1 & e \in E & \text{every elevator } e \text{ gets a feasible schedule } S \in \mathcal{S}_e \\
& x_S \in \{0,1\} & S \in \mathcal{S} & \text{feasible schedule } S \text{ can be used or not}
\end{array}
$$

Exact Approach via Set Partitioning

- compute dispatch by solving set partitioning formulation

$$
\begin{array}{rll}
\min \quad \displaystyle\sum_{S \in \mathcal{S}} c_S x_S & & \\
\displaystyle\sum_{S \in \mathcal{S}\colon r \in S} x_S = 1 & r \in \mathcal{R}_u & \\
\displaystyle\sum_{S \in \mathcal{S}_e} x_S = 1 & e \in E & \\
x_S \in \{0,1\} & S \in \mathcal{S} &
\end{array}
$$

cost c_S of a schedule S is weighted sum of waiting and travel times

every unassigned request $r \in \mathcal{R}_u$ is served

every elevator e gets a feasible schedule $S \in \mathcal{S}_e$

feasible schedule S can be used or not

- pricing problem: find a minimum cost feasible schedule, serving all requests assigned to this elevator plus some subset of the unassigned requests (reward: dual price)

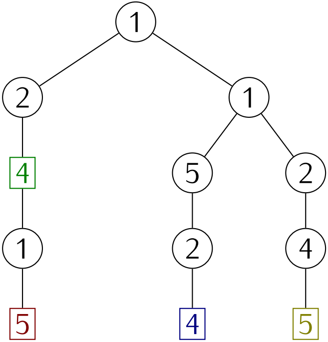
Branch & Bound Pricing Algorithm

$r_1 = \{c_1 \colon 2 \to 4\}, r_2 = \{c_2 \colon 1 \to 5\}$

Branch & Bound Pricing Algorithm

$r_1 = \{c_1 : 2 \to 4\}, r_2 = \{c_2 : 1 \to 5\}$

$A = \{r_1\}, O = \{r_2\}$



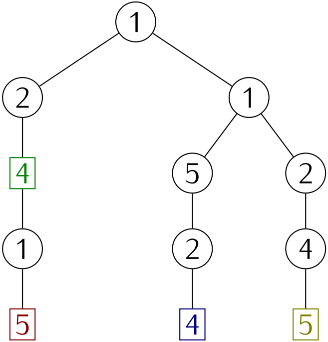
Node labels:

- A Set of not yet picked up assigned requests.
- O Set of not yet picked up optional requests.

Branch & Bound Pricing Algorithm

$$r_1 = \{c_1 : 2 \to 4\}, r_2 = \{c_2 : 1 \to 5\}$$

$$A = \{r_1\}, O = \{r_2\}$$



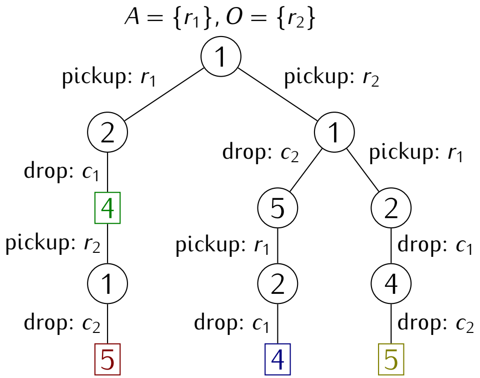
Node labels:

- A Set of not yet picked up assigned requests.
- O Set of not yet picked up optional requests.

- a node represents a partial schedule
- a node is feasible if the corresponding schedule is feasible

Branch & Bound Pricing Algorithm

$$r_1 = \{c_1 : 2 \to 4\}, r_2 = \{c_2 : 1 \to 5\}$$



Node labels:

- A Set of not yet picked up assigned requests.
- O Set of not yet picked up optional requests.

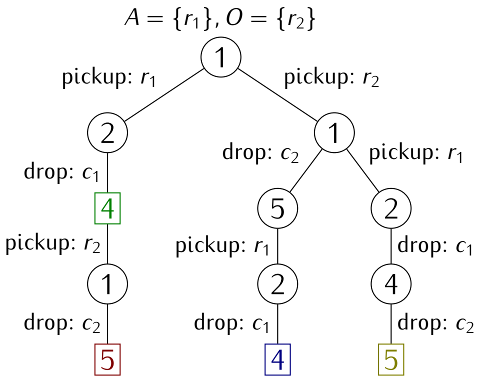
- a node represents a partial schedule
- a node is feasible if the corresponding schedule is feasible

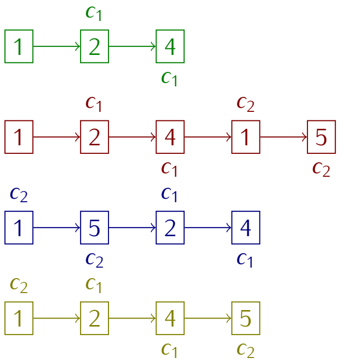
Branching

- drop call
- pick up request

Branch & Bound Pricing Algorithm

$r_1 = \{c_1\colon 2 \to 4\}, r_2 = \{c_2\colon 1 \to 5\}$





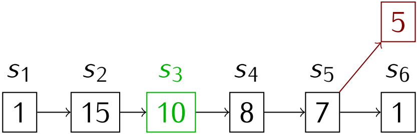
Greedy Lower Bounds

Example partial schedule in some node in the search tree:

$$\overset{s_1}{\boxed{1}} \longrightarrow \overset{s_2}{\boxed{15}} \longrightarrow \overset{s_3}{\boxed{10}} \longrightarrow \overset{s_4}{\boxed{8}} \longrightarrow \overset{s_5}{\boxed{7}} \longrightarrow \overset{s_6}{\boxed{1}}$$

Greedy Lower Bounds

Example partial schedule in some node in the search tree:



request 1: $5 \rightarrow 2$:

$$t^{+}(1) \geq s_5.\mathit{ArrivalTime} + \tau_{\mathrm{drv}}(7, 5)$$

Greedy Lower Bounds

Example partial schedule in some node in the search tree:

$$\begin{array}{ccccccccccccc} s_1 & & s_2 & & s_3 & & s_4 & & s_5 & & s_6 & & \\ \boxed{1} & \to & \boxed{15} & \to & \boxed{10} & \to & \boxed{8} & \to & \boxed{7} & \to & \boxed{1} & \to & \boxed{8} \end{array}$$

request 2: $8 \to 9$:

$$t^{+}(2) \geq s_6.\mathit{ArrivalTime} + \tau_{\mathrm{drv}}(1, 8)$$

Greedy Lower Bounds

Example partial schedule in some node in the search tree:

$$
\begin{array}{ccccccccccccc}
s_1 & & s_2 & & s_3 & & s_4 & & s_5 & & s_6 & & \\
\boxed{1} & \to & \boxed{15} & \to & \boxed{10} & \to & \boxed{8} & \to & \boxed{7} & \to & \boxed{1} & \to & \boxed{8}
\end{array}
$$

request 2: $8 \to 9$:

$$t^{+}(2) \geq s_6.\mathit{ArrivalTime} + \tau_{\mathrm{drv}}(1, 8)$$

- similar bounds for the travel time of each call
  $\rightsquigarrow$ bounds on cost for each request

Greedy Lower Bounds

Example partial schedule in some node in the search tree:

$$\begin{array}{ccccccc} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 & \\ \boxed{1} \to & \boxed{15} \to & \boxed{10} \to & \boxed{8} \to & \boxed{7} \to & \boxed{1} \to & \boxed{8} \end{array}$$

request 2: 8 $\to$ 9:

$$t^+(2) \geq s_6.\mathit{ArrivalTime} + \tau_{\text{drv}}(1, 8)$$

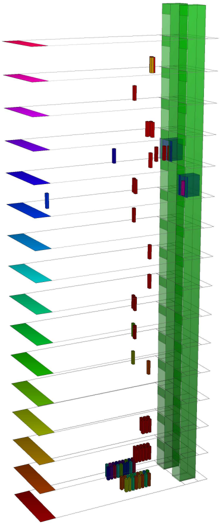
- similar bounds for the travel time of each call
  $\rightsquigarrow$ bounds on cost for each request
- comparing cost bound with dual price allows additional pruning

Simulation Model

- realistic time model: acceleration, maximum speed, ...
- waiting time of elevator at a floor depends on the number of passengers entering/leaving
- elevators stay at floor visited last if no further calls are assigned
- elevators cannot stop or reverse direction halfway between floors
- first-come-first-served boarding: at each stop, passengers enter elevator car in order of increasing waiting time

Simulation Data

- building with a group of 5 elevators serving 25 floors
- three entrance floors, two of which are only served by two elevators



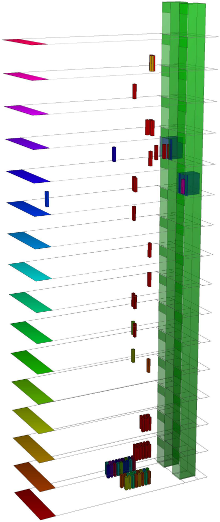
Simulation Data

- building with a group of 5 elevators serving 25 floors
- three entrance floors, two of which are only served by two elevators
- common types of traffic

  Up traffic (U): all calls start from entrance floors

  Interfloor traffic (I): uniformly distributed start/destination floors

  Down traffic (D): all calls to entrance floors



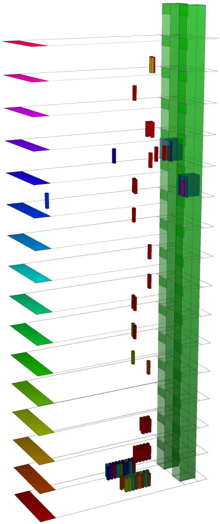
Simulation Data

- building with a group of 5 elevators serving 25 floors
- three entrance floors, two of which are only served by two elevators
- common types of traffic

  Up traffic (U): all calls start from entrance floors

  Interfloor traffic (I): uniformly distributed start/destination floors

  Down traffic (D): all calls to entrance floors
- here: one hour Real Up peak traffic with varying intensities



System Comparison: Waiting Times

evaluate control algorithms using median $\alpha_{0.5}$, 90% quantile $\alpha_{0.9}$, and average $\emptyset$ of waiting time

System Comparison: Waiting Times

evaluate control algorithms using median $\alpha_{0.5}$, 90% quantile $\alpha_{0.9}$, and average $\emptyset$ of waiting time

scenario	conventional			IA system			DA system		
	$\alpha_{0.5}$	$\alpha_{0.9}$	$\emptyset$	$\alpha_{0.5}$	$\alpha_{0.9}$	$\emptyset$	$\alpha_{0.5}$	$\alpha_{0.9}$	$\emptyset$
Real Up 80%	6	22	9	11	26	12	10	25	11
Real Up 100%	7	25	10	12	34	15	9	28	12
Real Up 144%	176	436	202	16	47	20	8	37	15
Real Up 168%	493	1181	535	42	161	65	–	–	–

waiting times in seconds

System Comparison: Travel Times

scenario	conventional			IA system			DA system		
	$\alpha_{0.5}$	$\alpha_{0.9}$	$\emptyset$	$\alpha_{0.5}$	$\alpha_{0.9}$	$\emptyset$	$\alpha_{0.5}$	$\alpha_{0.9}$	$\emptyset$
Real Up 80%	61	113	65	45	73	47	44	71	45
Real Up 100%	71	129	75	54	86	55	50	84	52
Real Up 144%	260	515	279	72	116	74	69	112	71
Real Up 168%	566	1253	611	106	235	127	–	–	–

travel times in seconds

Snapshot Solution Quality

Solution quality for snapshot problems solved for each new call

gap integrality gap between LP solution after complete pricing and final cost of the dispatch

time total solution time including pricing and solving IP to optimality

$|\mathcal{R}_u|$ number of unassigned requests

Snapshot Solution Quality

Solution quality for snapshot problems solved for each new call

gap integrality gap between LP solution after complete pricing and final cost of the dispatch

time total solution time including pricing and solving IP to optimality

$|\mathcal{R}_u|$ number of unassigned requests

quantile	IA system *(Real Up 168%)* gap [%]	time [s]	$\lvert\mathcal{R}_u\rvert$	DA system *(Real Up 144%)* gap [%]	time [s]	$\lvert\mathcal{R}_u\rvert$
$\alpha_{0.5}$	0.0	0.01	1	0.0	0.13	7
$\alpha_{0.75}$	0.0	0.02	1	0.0	0.27	10
$\alpha_{0.9}$	0.0	0.15	2	0.0	2.09	13
$\alpha_{1.0}$	0.1	1.21	5	0.7	2540.52	21

Conclusions

- destination call systems are offer higher capacity than conventional systems

Conclusions

- destination call systems are offer higher capacity than conventional systems
- DA systems allow to improve over both IA and conventional systems

Conclusions

- destination call systems are offer higher capacity than conventional systems
- DA systems allow to improve over both IA and conventional systems
- proven optimal dispatch often found in seconds, even for high intensity traffic
  $\Longrightarrow$ real-time compliant for IA systems

Conclusions

- destination call systems are offer higher capacity than conventional systems
- DA systems allow to improve over both IA and conventional systems
- proven optimal dispatch often found in seconds, even for high intensity traffic
  $\Longrightarrow$ real-time compliant for IA systems
- for DA systems computation times still too long for practical use; but already useful for assessing quality of heuristics

Outline

Recap: Online optimization

Theoretical framework: Online Dial-a-Ride problems and competitive analysis

Online Optimization in Practice: Reoptimization Algorithms

- Dispatching the service vehicles of ADAC
- Controlling cargo elevators in a distribution center
- Controlling passenger elevators in high-rise buildings

Theory again: The Online Bin Coloring problem

Online Bin Coloring [Krumke et al '01]

Task

Put a sequence of n colored items into m bins, subject to:

- As soon as a bin has B items it is replaced by an empty one (but no earlier).
- Items must be packed without knowledge of future items, i. e., online.
- Items may not be rearranged between the bins.

Online Bin Coloring [Krumke et al '01]

Task

Put a sequence of n colored items into m bins, subject to:

- As soon as a bin has B items it is replaced by an empty one (but no earlier).
- Items must be packed without knowledge of future items, i. e., online.
- Items may not be rearranged between the bins.

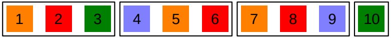
Goal

Minimize the maximum number of colors in a bin (*colorfulness*).

Three Algorithms

sequence for $m = 2$, $B = 3$: 1 2 3 4 5 6 7 8 9 10 sequence for $m = 2$, $B = 3$: 

OneBin Put all items in first bin.

1: 

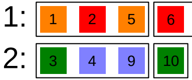
2:

Three Algorithms

sequence for $m = 2$, $B = 3$: 

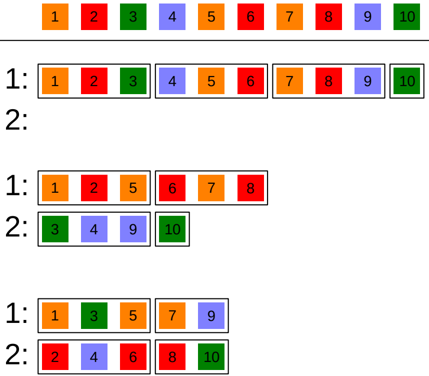
OneBin Put all items in first bin.

FixedColors Assign each bin equally many colors. Put only items with one of those colors in the bin.



Three Algorithms

sequence for $m = 2$, $B = 3$:



OneBin Put all items in first bin.

FixedColors Assign each bin equally many colors. Put only items with one of those colors in the bin.

GreedyFit Put an item with a new color in a bin which currently has least colorfulness.

Competitive Analysis Results [Krumke et al '01]

- GreedyFit has competitive ratio not greater than $3m$, but greater or equal to $2m$.
- OneBin has competitive ratio at most $2m - 1$.

Competitive Analysis Results [Krumke et al '01]

- GreedyFit has competitive ratio not greater than $3m$, but greater or equal to $2m$.
- OneBin has competitive ratio at most $2m - 1$.
- simulations (random data): GreedyFit significantly better than OneBin

Competitive Analysis Results [Krumke et al '01]

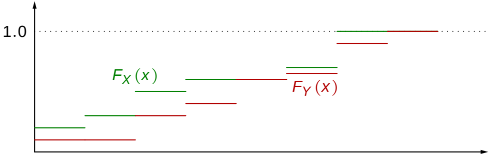
- GreedyFit has competitive ratio not greater than $3m$, but greater or equal to $2m$.
- OneBin has competitive ratio at most $2m - 1$.
- simulations (random data): GreedyFit significantly better than OneBin
- from now on: sequence is constructed by choosing each color independently according to distribution γ

New idea for analysis: Use stochastic dominance

Stochastic Dominance

Let X and Y be random variables with distribution functions F_X and F_Y. X is stochastically dominated by Y, written $X \leq_{st} Y$, if

$$F_X(x) \geq F_Y(x) \quad \text{for all } x \in \mathbb{R}.$$



Example: Stochastic Dominance for Bin Colorfulness

Colorfulness distribution for $m = 3$, $B = 5$, $C = 15$

#Items	FixedColors	GreedyFit
5	0.028, 0.657, 0.971, 1.000, 1.000	0.094, 1.000, 1.000, 1.000, 1.000

Shown is the cumulative distribution function!

Example: Stochastic Dominance for Bin Colorfulness

Colorfulness distribution for $m = 3$, $B = 5$, $C = 15$

#Items	FixedColors	GreedyFit
5	0.028, 0.657, 0.971, 1.000, 1.000	0.094, 1.000, 1.000, 1.000, 1.000
10	0.054, 0.619, 0.975, 1.000	0.189, 0.981, 1.000, 1.000
20	0.001, 0.192, 0.896, 1.000	0.009, 0.725, 0.999, 1.000
40	0.021, 0.766, 1.000	0.429, 0.999, 1.000
80	0.560, 1.000	0.149, 0.998, 1.000
160	0.299, 1.000	0.018, 0.993, 1.000
1000	1.000	0.956, 1.000

Shown is the cumulative distribution function!

A Stochastic Dominance Result

Theorem ([Hiller, Vredeveld '08])

Assume that the color sequence Σ is generated by choosing each color independently at random according to color distribution γ. Then

$$GF(\Sigma) \leq_{\mathrm{st}} OB(\Sigma).$$

Proof involves some probability tools ...

- Markov chains
- stopping times
- mixtures of probability distributions
- couplings

...but it is straightforward.

Consequences of the result

- $X \leq_{\text{st}} Y \Longrightarrow \mathbb{E}[X] \leq \mathbb{E}[Y]$

Consequences of the result

- $X \leq_{\mathrm{st}} Y \implies \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{\mathrm{st}} Y \implies f(X) \leq_{\mathrm{st}} f(Y)$ for every non-decreasing function f

Consequences of the result

- $X \leq_{\text{st}} Y \Longrightarrow \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{\text{st}} Y \Longrightarrow f(X) \leq_{\text{st}} f(Y)$ for every non-decreasing function f
- better *average* competitive ratio, i. e., $\frac{\mathbb{E}[\mathrm{GF}_n]}{\mathbb{E}[\mathrm{OPT}_n]} \leq \frac{\mathbb{E}[\mathrm{OB}_n]}{\mathbb{E}[\mathrm{OPT}_n]}$

Consequences of the result

- $X \leq_{\text{st}} Y \implies \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{\text{st}} Y \implies f(X) \leq_{\text{st}} f(Y)$ for every non-decreasing function f
- better *average* competitive ratio, i. e., $\frac{\mathbb{E}[\mathrm{GF}_n]}{\mathbb{E}[\mathrm{OPT}_n]} \leq \frac{\mathbb{E}[\mathrm{OB}_n]}{\mathbb{E}[\mathrm{OPT}_n]}$
- stochastic dominance for the uniform distribution is equivalent to a bijective analysis [Angelopoulos et al '07] result:

Consequences of the result

- $X \leq_{\mathrm{st}} Y \Longrightarrow \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{\mathrm{st}} Y \Longrightarrow f(X) \leq_{\mathrm{st}} f(Y)$ for every non-decreasing function f
- better *average* competitive ratio, i. e., $\frac{\mathbb{E}[\mathrm{GF}_n]}{\mathbb{E}[\mathrm{OPT}_n]} \leq \frac{\mathbb{E}[\mathrm{OB}_n]}{\mathbb{E}[\mathrm{OPT}_n]}$
- stochastic dominance for the uniform distribution is equivalent to a bijective analysis [Angelopoulos et al '07] result:

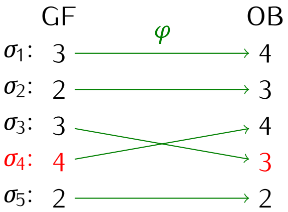
	GF	OB
σ_1:	3	4
σ_2:	2	3
σ_3:	3	4
σ_4:	4	3
σ_5:	2	2

Consequences of the result

- $X \leq_{\text{st}} Y \Longrightarrow \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{\text{st}} Y \Longrightarrow f(X) \leq_{\text{st}} f(Y)$ for every non-decreasing function f
- better *average* competitive ratio, i. e., $\frac{\mathbb{E}[\mathrm{GF}_n]}{\mathbb{E}[\mathrm{OPT}_n]} \leq \frac{\mathbb{E}[\mathrm{OB}_n]}{\mathbb{E}[\mathrm{OPT}_n]}$
- stochastic dominance for the uniform distribution is equivalent to a bijective analysis [Angelopoulos et al '07] result:

	GF	OB
σ_1:	3	4
σ_2:	2	3
σ_3:	3	4
σ_4:	4	3
σ_5:	2	2

- $X \leq_{\text{st}} Y \Longrightarrow \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{\text{st}} Y \Longrightarrow f(X) \leq_{\text{st}} f(Y)$ for every non-decreasing function f
- better *average* competitive ratio, i. e., $\frac{\mathbb{E}[\text{GF}_n]}{\mathbb{E}[\text{OPT}_n]} \leq \frac{\mathbb{E}[\text{OB}_n]}{\mathbb{E}[\text{OPT}_n]}$
- stochastic dominance for the uniform distribution is equivalent to a bijective analysis [Angelopoulos et al '07] result:



Consequences of the result

- $X \leq_{\text{st}} Y \implies \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{\text{st}} Y \implies f(X) \leq_{\text{st}} f(Y)$ for every non-decreasing function f
- better *average* competitive ratio, i. e., $\frac{\mathbb{E}[\text{GF}_n]}{\mathbb{E}[\text{OPT}_n]} \leq \frac{\mathbb{E}[\text{OB}_n]}{\mathbb{E}[\text{OPT}_n]}$
- stochastic dominance for the uniform distribution is equivalent to a bijective analysis [Angelopoulos et al '07] result: There is a bijective map φ on the sequences s. t.

$$\text{GF}(\sigma) \leq \text{OB}(\varphi(\sigma)) \quad \forall \sigma$$

$$\begin{tikzcd}[row sep=small]
 & \text{GF} & & \text{OB} \\
\sigma_1: & 3 \arrow[rr, "\varphi"] & & 4 \\
\sigma_2: & 2 \arrow[rr] & & 3 \\
\sigma_3: & 3 \arrow[rrd] & & 4 \\
\sigma_4: & 4 \arrow[rru] & & 3 \\
\sigma_5: & 2 \arrow[rr] & & 2
\end{tikzcd}$$

Summary

Theory

- Competitive analysis and its variants are the main tool to study online algorithms.
- Different measures and ways of analysis might provide more useful / interesting results.

Practice

- Exact reoptimization algorithms perform well in practice.
- Mathematical programming techniques can sometimes be used to obtain really fast algorithms.

References I

Spyros Angelopoulos, Reza Dorrigiv, and Alejandro López-Ortiz.
On the separation and equivalence of paging strategies.
In *SODA 2007*, pages 229–237, 2007.

Giorgio Ausiello, Esteban Feuerstein, Stefano Leonardi, Leen Stougie, and M. Talamo.
Algorithms for the on-line travelling salesman.
Algorithmica, 29(4):560–581, 2001.

Norbert Ascheuer, Sven O. Krumke, and Jörg Rambau.
Online Dial-a-Ride problems: Minimizing the completion time.
In *Proceedings of the 17st Symposium on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 639–650. Springer, 2000.

References II

Philipp Friese and Jörg Rambau.
Online-optimization of a multi-elevator transport system with reoptimization algorithms based on set-partitioning models.
Discrete Appl. Math., 154(13):1908–1931, 2006.
also available as ZIB Report 05-03.

Dietrich Hauptmeier, Sven O. Krumke, and Jörg Rambau.
The online Dial-a-Ride problem under reasonable load.
In *CIAC 2000*, volume 1767 of *Lecture Notes in Computer Science*, pages 125–136. Springer, 2000.

References III

Benjamin Hiller, Sven Oliver Krumke, and Jörg Rambau.
Reoptimization gaps versus model errors in online-dispatching of service units for ADAC.
Discrete Appl. Math., 154(13):1897–1907, 2006.
Traces of the Latin American Conference on Combinatorics, Graphs and Applications – A selection of papers from LACGA 2004, Santiago, Chile.

Benjamin Hiller and Tjark Vredeveld.
Probabilistic analysis of online bin coloring algorithms via stochastic comparison.
In *Proceedings of the 16th Annual European Symposium on Algorithms*, volume 5193 of *Lecture Notes in Computer Science*, pages 528–539. Springer, 2008.

Sven Oliver Krumke, Willem E. de Paepe, Leen Stougie, and Jörg Rambau.
Online bin coloring.
In Friedhelm Meyer auf der Heide, editor, *Proceedings of the 9th Annual European Symposium on Algorithms*, volume 2161 of *Lecture Notes in Computer Science*, pages 74–84, 2001.

Sven Oliver Krumke, Jörg Rambau, and Luis M. Torres.
Realtime-dispatching of guided and unguided automobile service units with soft time windows.
In Rolf H. Möhring and Rajeev Raman, editors, *Proceedings of the 10th Annual European Symposium on Algorithms*, volume 2461 of *Lecture Notes in Computer Science*, pages 637–648. Springer, 2002.