Online optimization: Elevator and vehicle scheduling
CO@Work Berlin

Benjamin Hiller
ZIB
10/01/2009
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
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
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.
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.
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 $G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0})$
Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0})$
- special node $d \in V$, called depot
Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0})$
- 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 $m_i = (s_i, t_i, L_i)$, $0 \leq i \leq k$.
- The time needed by a move $m = (s, t, L)$ is $w(s, t)$.
Dial-a-Ride Problems (offline)

Input

- weighted graph $G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0})$
- 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 \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \)
- 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 \( \left( m_i = (s_i, t_i, L_i) \right)_{0 \leq i \leq k} \) with
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

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 $\left( m_i = (s_i, t_i, L_i) \right)_{0 \leq i \leq k}$ with
  1. The first move starts and the last move ends at $d$. 
Dial-a-Ride Problems (offline)

Input

- weighted graph \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \)
- 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 \( (m_i = (s_i, t_i, L_i))_{0 \leq i \leq k} \) 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 \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \)
- 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 \((m_i = (s_i, t_i, L_i))_{0 \leq i \leq k}\) 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 $G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0})$
- 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 $(m_i = (s_i, t_i, L_i))_{0 \leq i \leq k}$ 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.

Objective

- Find a schedule with minimum completion time.
Dial-a-Ride Problems (offline)

Input

- weighted graph \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \)
- 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 \( \left( m_i = (s_i, t_i, L_i) \right)_{0 \leq i \leq k} \).
- The time needed by a move \( m = (s, t, L) \) is \( w(s, t) \).
Dial-a-Ride Problems (offline)

Input

- weighted graph \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \)
- 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 \( \{m_i = (s_i, t_i, L_i)\}_{0 \leq i \leq k} \).
- 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 $G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0})$, depot node $d \in V$
Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w: A \to \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence $\sigma$ of requests $(s_j, t_j, \tau_j)$, $\tau_j$ is release time
Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence $\sigma$ of requests $(s_j, t_j, \tau_j)$, $\tau_j$ 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 \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \), depot node \( d \in V \)
- sequence \( \sigma \) of requests \((s_j, t_j, \tau_j)\), \( \tau_j \) 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 \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \), depot node \( d \in V \)
- sequence \( \sigma \) of requests \((s_j, t_j, \tau_j)\), \( \tau_j \) 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 \( \tau_j \).
Online Dial-a-Ride Problems

Input

- weighted graph $G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0})$, depot node $d \in V$
- sequence $\sigma$ of requests $(s_j, t_j, \tau_j)$, $\tau_j$ 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 $\tau_j$.

Objectives

- Find a schedule with minimum completion time.
Online Dial-a-Ride Problems

Input

- weighted graph \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \), depot node \( d \in V \)
- sequence \( \sigma \) of requests \((s_j, t_j, \tau_j)\), \( \tau_j \) 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 \( \tau_j \).

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 \( G = (V, A, w: A \rightarrow \mathbb{R}_{\geq 0}) \), depot node \( d \in V \)
- sequence \( \sigma \) of requests \( (s_j, t_j, \tau_j) \), \( \tau_j \) 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 \( \tau_j \).

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.
Quality of Dial-a-Ride algorithms

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 $\text{Alg}$ is called an $\alpha$-approximation algorithm if

$$\text{Alg}(G, d, R) \leq \alpha \cdot \text{OPT}(G, d, R)$$

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 Alg is called an $\alpha$-approximation algorithm if
  \[ \text{Alg}(G, d, R) \leq \alpha \cdot \text{Opt}(G, d, R) \]
  for all instances $(G, d, R)$.

Online Dial-a-Ride problem

- An online algorithm Alg is called a $c$-competitive algorithm if
  \[ \text{Alg}(G, d, R) \leq c \cdot \text{Opt}(G, d, R) \]
  for all instances $(G, d, R)$. Here, 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
Replan-strategy

- As a new request arrives, compute a new schedule for the current set of requests using offline algorithm \( \text{Alg} \).
- 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 $A_{\text{alg}}$.
- 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 $A_{\text{alg}}$ and follow this schedule.
Completion time [Ascheuer, Krumke, Rambau ’00, Ausiello et al ’01]

- Both \texttt{Replan} and \texttt{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 $\alpha$-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 $\alpha$-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 5α/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 \(\alpha\)-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 $c \geq 1$. 
Want to show: There is no $c$-competitive online algorithm for any $c \geq 1$.

Meaning: For any online algorithm $\text{Alg}$ and any $c \geq 1$, there is a sequence $\sigma^{\text{Alg}}$ 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 $x^{\text{Alg}}$ be the position of the server controlled by $\text{Alg}$ at time 1.

Case 1: $x^{\text{Alg}} \leq 0 \Rightarrow \sigma^{\text{Alg}} = (1/2, 1/2 + \epsilon)$. 

$\text{Alg}^{\text{max-flow}}(\sigma^{\text{Alg}}) \geq 1 + \epsilon \cdot \text{OPT}^{\text{max-flow}}(\sigma^{\text{Alg}}) = \epsilon$.
Proof

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

- Meaning: For any online algorithm $\text{Alg}$ and any $c \geq 1$, there is a sequence $\sigma^{\text{Alg}}$ 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.
Proof

- Want to show: There is no \( c \)-competitive online algorithm for any \( c \geq 1 \).
- Meaning: For any online algorithm \( \text{Alg} \) and any \( c \geq 1 \), there is a sequence \( \sigma^{\text{Alg}} \) 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 \( \text{Alg} \) at time 1.
Proof

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

► Consider the real line and a server moving at unit speed. Let $x$ be position of the server controlled by $\text{Alg}$ at time 1.

► Case 1: $x \leq 0 \Rightarrow \sigma^{\text{Alg}} = (r = (1, 1, 1 + \varepsilon))$. 

\[ r = (1, 1, 1 + \varepsilon) \]
Want to show: There is no $c$-competitive online algorithm for any $c \geq 1$.

Meaning: For any online algorithm $\text{Alg}$ and any $c \geq 1$, there is a sequence $\sigma_{\text{Alg}}$ s.t. $\text{Alg}^{\max-\text{flow}}(\sigma_{\text{Alg}}) \geq c \cdot \text{OPT}^{\max-\text{flow}}(\sigma_{\text{Alg}})$.

Consider the real line and a server moving at unit speed. Let $x$ be the position of the server controlled by $\text{Alg}$ at time 1.

Case 1: $x \leq 0 \leadsto \sigma_{\text{Alg}} = \left( r = (1, 1, 1 + \epsilon) \right)$.

\[
\text{Alg}^{\max-\text{flow}}(\sigma_{\text{Alg}}) \geq 1 + \epsilon
\]
Want to show: There is no $c$-competitive online algorithm for any $c \geq 1$.

Meaning: For any online algorithm $A_{\text{Alg}}$ and any $c \geq 1$, there is a sequence $\sigma^{\text{Alg}}$ s.t. $A_{\text{Alg}}^{\text{max-flow}}(\sigma^{\text{Alg}}) \geq c \cdot \text{OPT}^{\text{max-flow}}(\sigma^{\text{Alg}})$.

Consider the real line and a server moving at unit speed. Let $x$ be position of the server controlled by $A_{\text{Alg}}$ at time 1.

Case 1: $x \leq 0 \implies \sigma^{\text{Alg}} = (r = (1, 1, 1 + \varepsilon))$.

\[ A_{\text{Alg}}^{\text{max-flow}}(\sigma^{\text{Alg}}) \geq 1 + \varepsilon \]
\[ \text{OPT}^{\text{max-flow}}(\sigma^{\text{Alg}}) = \varepsilon \]
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 $\mu < \text{service rate } g$
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 $\rho$

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 $\mu < \text{service rate } g$

Worst-case model: Reasonable load

- For request set $R$, let $\delta(R)$ denote the **time span** of $R$.
- A request set $R$ is called $\Delta$-reasonable if
  
  For all $R' \subseteq R$ with $\delta(R') \geq \Delta$ we have
  
  $$\text{OPT}^{\text{comp}}(R') \leq \delta(R').$$
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 $\mu < \text{service rate } g$

Worst-case model: Reasonable load

- For request set $R$, let $\delta(R)$ denote the time span of $R$.
- A request set $R$ is called $\Delta$-reasonable if
  For all $R' \subseteq R$ with $\delta(R') \geq \Delta$ we have
  $$\text{OPT}^{\text{comp}}(R') \leq \delta(R').$$
- A request sequence $\sigma$ is called $\Delta$-reasonable if the set of all requests in $\sigma$ is $\Delta$-reasonable.
Theorem ([Hauptmeier, Krumke, Rambau ’00])

1. Let $\sigma$ be a $\Delta$-reasonable request sequence. Then maximal flow time achieved by $\text{IGNORE}$ on $\sigma$ is bounded by $2\Delta$. 
Theorem ([Hauptmeier, Krumke, Rambau ’00])

1. Let $\sigma$ be a $\Delta$-reasonable request sequence. Then maximal flow time achieved by IGNORE on $\sigma$ is bounded by $2\Delta$.

2. There is a $\Delta \in \mathbb{R}$, a $\Delta$-reasonable request sequence $\sigma$ and a request $r$ in $\sigma$ such that the flow time for $r$ achieved by REPLAN is unbounded.
Proof of $2\Delta$ bound for IGNORE

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

Define

- $R_0 = \{r\}$
Proof of $2\Delta$ bound for \textsc{Ignore}

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

Define

- $R_0 = \{r\}$
- $\delta_{i+1} := \text{time needed to serve } R_i$
- $R_{i+1} := \text{set of requests arriving during serving } R_i$
Proof of $2\Delta$ bound for \textsc{Ignore}

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

Define

- $R_0 = \{r\}$
- $\delta_{i+1} := \text{time needed to serve } R_i$
- $R_{i+1} := \text{set of requests arriving during serving } R_i$

Observation: $\textsc{Ignore}^{\max\text{-flow}}(R_i) \leq \delta_i + \delta_{i+1}$
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: $\text{IGNORE}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1} \leq 2\Delta$

Suffices to show: $\delta_i \leq \Delta$ for all $i$
Proof of $2\Delta$ bound for IGNORE

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

Define
- $R_0 = \{r\}$
- $\delta_{i+1} := \text{time needed to serve } R_i$
- $R_{i+1} := \text{set of requests arriving during serving } R_i$

Observation: $\text{IGNORE}^\text{max-flow}(R_i) \leq \delta_i + \delta_{i+1}$

Suffices to show: $\delta_i \leq \Delta$ for all $i$

We have: $\delta_{i+1} = \text{OPT}^{\text{comp}}(R_i)$
Proof of $2\Delta$ 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: $\text{IGNORE}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1}$

Suffices to show: $\delta_i \leq \Delta$ for all $i$

We have: $\delta_{i+1} = \text{OPT}^{\text{comp}}(R_i) \leq \delta(R_i) \leq \delta_i$
Proof of $2\Delta$ 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: $\text{IGNORE}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1}$

Suffices to show: $\delta_i \leq \Delta$ for all $i$

We have: $\delta_{i+1} = \text{OPT}^{\text{comp}}(R_i) \leq \delta(R_i) \leq \delta_i \leq \max\{\Delta, \delta_i\}$
Proof of $2\Delta$ bound for Ignore

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

Define

$R_0 = \{r\}$

$\delta_{i+1} := \text{time needed to serve } R_i$

$R_{i+1} := \text{set of requests arriving during serving } R_i$

Observation: $\text{Ignore}^{\text{max-flow}}(R_i) \leq \delta_i + \delta_{i+1} \leq 2\Delta$

Suffices to show: $\delta_i \leq \Delta$ for all $i$

We have: $\delta_{i+1} = \text{OPT}^{\text{comp}}(R_i) \leq \delta(R_i) \leq \delta_i \leq \max\{\Delta, \delta_i\} \leq \Delta$. 
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
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

Knapp 17.300 Gelebe Engel sind rund um die Uhr für Sie im Einsatz, Ihnen bei Pannen oder Unfällen auf Bundesautobahnen, in Ortschaften und in Einkaufszentren zu helfen. Mit moderner Technik ausgestattete Pannenhilfe-Zentrale nehmen Ihren Helferurlaub und organisieren den Helfer, der am schnellsten bei Ihnen ankommt.

Zusätzlich stehen mehr als 4.000 Pannenhilfe-Fahrzeuge der Straßendienste rund um die Uhr zur Verfügung.

Pannenhilfe-Bilanz
Neuer EinsatzRecord bei Straßensicherheit und Straßenverkehr. Mehr...

Pannenaufnahme
Ihre Pannen aufnehmen. Deutschland ohne Grenze. Mehr... per Online-Formular.

Die Pannenstatistik
Über 90 Modelleinheiten im Wandel der Pannenhilfe. Mehr...
Application #1: Dispatching the service vehicles of ADAC
Application #1: Dispatching the service vehicles of ADAC
Application #1: Dispatching the service vehicles of ADAC
Application #1: Dispatching the service vehicles of ADAC

- Phone call
- Wireless transmission
- Dispatch: Human operator

Help center

Yellow Angel

ADAC

Online optimization
Application #1: Dispatching the service vehicles of ADAC
Situation

- ≈ 1,700 Yellow Angels
- ≈ 1,200 road service partners with ≈ 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 ≤ 10 seconds).

Goal
Minimize operating cost + lateness cost.
The ADAC dispatching problem is an online problem.
⇒ 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. ⇒ 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.

- Cost structure is complex: nonlinear lateness cost, discounted partner costs, ... ⇒ tour-based model
Tour-based model: Details

\[ u_1, \hat{u}_1, u_2, r_1, r_2, u_2, \hat{u}_2, r_3 \]
Tour-based model: Details

\[ \begin{array}{c}
\text{\(u_1\)} \\
\text{\(r_1\)} \\
\text{\(r_2\)} \\
\text{\(r_3\)} \\
\end{array} \]

\[ \begin{array}{c}
\text{\(\hat{u}_1\)} \\
\text{\(\hat{u}_2\)} \\
\end{array} \]

\[ \begin{array}{c|c|c|c|c|c|c}
\text{\(t_1\)} & \text{\(r_1\)} & \text{\(r_2\)} & \text{\(r_3\)} & \text{\(u_1\)} & \text{\(u_2\)} \\
1 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{array} \]

\(c_{t_1}\)
Tour-based model: Details

\[
\begin{align*}
\hat{u}_1 & \quad \bullet \quad r_1 \\
\bullet & \quad u_1 \quad r_3 \\
\bullet & \quad u_2
\end{align*}
\]

\[
\begin{array}{c|c|c}
\hline
r_1 & r_2 & r_3 \\
\hline
1 & 0 & 0 \\
0 & 1 & 1 \\
1 & 1 & 0 \\
0 & 1 & 1 \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c}
\hline
t_1 & t_2 & c_{t_1} \\
\hline
1 & 0 & r_1 \\
0 & 1 & r_2 \\
1 & 1 & r_3 \\
0 & 1 & u_1 \\
1 & 0 & u_2 \\
\hline
\end{array}
\]

Obtain set partitioning IP:

\[
\min \sum_{i=1}^{n} \left( T_i \cdot A_i \right) \quad \text{s.t.} \quad \begin{cases} 
T_i = 1 & \text{if } A_i = 1 \\
T_i = 0 & \text{if } A_i = 0 
\end{cases}
\]

\[A_i \in \{0, 1\}\]
Tour-based model: Details

\begin{align*}
\text{\textbf{t}}_1 & \quad \text{\textbf{t}}_2 & \quad \text{\textbf{t}}_3 & \quad \text{\textbf{t}}_4 \\
1 & \quad 0 & \quad 0 & \quad 1 & \quad r_1 \\
0 & \quad 1 & \quad 0 & \quad 0 & \quad r_2 \\
1 & \quad 1 & \quad 0 & \quad 0 & \quad r_3 \\
1 & \quad 0 & \quad 1 & \quad 0 & \quad u_1 \\
0 & \quad 1 & \quad 0 & \quad 0 & \quad u_2
\end{align*}

\begin{align*}
\text{\textbf{c}}_{\text{\textbf{t}}_1} & \quad \text{\textbf{c}}_{\text{\textbf{t}}_2} & \quad \text{\textbf{c}}_{\text{\textbf{t}}_3} & \quad \text{\textbf{c}}_{\text{\textbf{t}}_4} \\
\end{align*}
Tour-based model: Details

### Table

<table>
<thead>
<tr>
<th>$t_1$</th>
<th>$t_2$</th>
<th>$t_3$</th>
<th>$t_4$</th>
<th>...</th>
<th>$t_N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>$r_1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$r_2$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$r_3$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>$u_1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>$u_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$c_{t_1}$</th>
<th>$c_{t_2}$</th>
<th>$c_{t_3}$</th>
<th>$c_{t_4}$</th>
<th>...</th>
<th>$c_{t_N}$</th>
</tr>
</thead>
</table>

- $u_1$, $u_2$, $r_1$, $r_2$, $r_3$
Tour-based model: Details

obtain set partitioning IP: \[
\min \quad c^T x \\
Ax = 1 \\
x \in \{0, 1\}^n
\]
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 $A$)
  ⇒ 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 $A$)
  ⇒ 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 $A$)
  ⇒ 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

⇒ reasonable dual prices
Data and simulation setup

- ADAC provided production data of several days ⇒ 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

Cost in % relative to ZIBDIP

08:00 10:00 12:00 14:00 16:00

ZIBDIP
BestInsert
2-OPT
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
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)

- FIFO: 2206 seconds
- NN: 137 seconds
- Replan: 1036 seconds
- Reopt-2-Opt: 55 seconds
- Reopt-ZIBDIP: 51 seconds

maximum waiting time (seconds)

- FIFO: 4696 seconds
- NN: 1468 seconds
- Replan: 8203 seconds
- Reopt-2-Opt: 224 seconds
- Reopt-ZIBDIP: 224 seconds

Extremely high waiting times are due to instability of the system: the number of unserved requests is ever-increasing. The 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]:

Benjamin Hiller (ZIB)
- 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*
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).
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).

* Depending on the system.
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
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
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
We employ a general model for elevator schedules applicable to all systems (conventional, IA, DA).
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.
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.
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.

\[
\begin{align*}
\text{IA 1} &= \{c_1 : 1 \rightarrow 5, c_2 : 1 \rightarrow 6, c_3 : 1 \rightarrow 5\} \\
\text{DA 1} &= \{c_1 : 1 \rightarrow 5, c_3 : 1 \rightarrow 5\} \\
\text{DA 2} &= \{c_2 : 1 \rightarrow 6\}
\end{align*}
\]
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.

\[
\begin{align*}
1 & : 1 \rightarrow 5 \\
2 & : 1 \rightarrow 6 \\
3 & : 1 \rightarrow 5
\end{align*}
\]

\[r = \{c_1, c_2, c_3\}\]
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.

\[
\begin{align*}
IA_1 &= \{c_1, c_2\} \\
IA_2 &= \{c_3\}
\end{align*}
\]
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.

\[
\begin{align*}
\text{IA} &= \{1, 2, 3\} \\
\text{DA} &= \{1, 3\} \quad r_1 = \{c_1, c_3\} \\
&\quad r_2 = \{c_2\}
\end{align*}
\]
compute dispatch by solving set partitioning formulation

\[
\min \sum_{S \in S} c_S x_S
\]

- cost \(c_S\) of a schedule \(S\) is weighted sum of waiting and travel times

\[
\sum_{S \in S: r \in S} x_S = 1 \quad r \in R_u
\]

- every unassigned request \(r \in R_u\) is served

\[
\sum_{S \in S_e} x_S = 1 \quad e \in E
\]

- every elevator \(e\) gets a feasible schedule \(S \in S_e\)

\[
x_S \in \{0, 1\} \quad S \in S
\]

- feasible schedule \(S\) can be used or not
compute dispatch by solving set partitioning formulation

\[
\begin{align*}
\min & \quad \sum_{S \in S} c_S x_S \\
\quad \sum_{S \in S: r \in S} x_S & = 1 \quad r \in R_u \\
\quad \sum_{S \in S_e} x_S & = 1 \quad e \in E \\
\quad x_S & \in \{0, 1\} \quad S \in S
\end{align*}
\]

cost \(c_S\) of a schedule \(S\) is weighted sum of waiting and travel times
every unassigned request \(r \in R_u\) is served
every elevator \(e\) gets a feasible schedule \(S \in S_e\)
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)
$r_1 = \{ c_1 : 2 \rightarrow 4 \}, \ r_2 = \{ c_2 : 1 \rightarrow 5 \}$
Branch & Bound Pricing Algorithm

\[ r_1 = \{c_1 : 2 \rightarrow 4\}, \quad r_2 = \{c_2 : 1 \rightarrow 5\} \]

\[ A = \{r_1\}, \quad O = \{r_2\} \]

**Node labels:**
- **A** Set of not yet picked up assigned requests.
- **O** Set of not yet picked up optional requests.
Branch & Bound Pricing Algorithm

\[ r_1 = \{ c_1 : 2 \rightarrow 4 \}, \quad r_2 = \{ c_2 : 1 \rightarrow 5 \} \]

\[ A = \{ r_1 \}, \quad O = \{ r_2 \} \]

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

\[ r_1 = \{ c_1 : 2 \rightarrow 4 \}, \quad r_2 = \{ c_2 : 1 \rightarrow 5 \} \]

\[ A = \{ r_1 \}, \quad O = \{ r_2 \} \]

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

Branching

- drop call
- pick up request
\[ r_1 = \{ c_1 : 2 \to 4 \}, \ r_2 = \{ c_2 : 1 \to 5 \} \]

\[ A = \{ r_1 \}, \ O = \{ r_2 \} \]
Greedy Lower Bounds

Example partial schedule in some node in the search tree:

\[ S_1 \rightarrow 15 \rightarrow 10 \rightarrow 8 \rightarrow 7 \rightarrow 1 \]
Greedy Lower Bounds

Example partial schedule in some node in the search tree:

\[ s_1 \rightarrow 1 \rightarrow s_2 \rightarrow 15 \rightarrow s_3 \rightarrow 10 \rightarrow s_4 \rightarrow 8 \rightarrow s_5 \rightarrow 7 \rightarrow s_6 \rightarrow 1 \]

request 1: 5 $\rightarrow$ 2:

\[ t^+(1) \geq s_5.\text{ArrivalTime} + \tau_{d_5}(7, 5) \]
Greedy Lower Bounds

Example partial schedule in some node in the search tree:

![Schedule Diagram]

request 2: 8 → 9:

\[ t^+(2) \geq s_6.\text{ArrivalTime} + \tau_{\text{drv}}(1, 8) \]
Greedy Lower Bounds

Example partial schedule in some node in the search tree:

\[
\begin{align*}
  s_1 & \rightarrow s_2 & s_3 & \rightarrow s_4 & s_5 & \rightarrow s_6 \\
  1 & \rightarrow 15 & 10 & \rightarrow 8 & 7 & \rightarrow 1 \rightarrow 8
\end{align*}
\]

request 2: 8 → 9:

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

- similar bounds for the travel time of each call
- \(\Rightarrow\) bounds on cost for each request
Greedy Lower Bounds

Example partial schedule in some node in the search tree:

\[ s_1 \rightarrow 1 \rightarrow 15 \rightarrow s_3 \rightarrow 10 \rightarrow 8 \rightarrow 7 \rightarrow s_5 \rightarrow 1 \rightarrow s_6 \rightarrow 8 \]

request 2: 8 → 9:

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

- similar bounds for the travel time of each call
- bounds on cost for each request
- comparing cost bound with dual price allows additional pruning
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
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
evaluate control algorithms using median $\alpha_{0.5}$, 90% quantile $\alpha_{0.9}$, and average $\bar{\alpha}$ of waiting time
evaluate control algorithms using median $\alpha_{0.5}$, 90% quantile $\alpha_{0.9}$, and average $\bar{\alpha}$ of waiting time

<table>
<thead>
<tr>
<th>scenario</th>
<th>conventional</th>
<th>IA system</th>
<th>DA system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_{0.5}$</td>
<td>$\alpha_{0.9}$</td>
<td>$\bar{\alpha}$</td>
</tr>
<tr>
<td>Real Up 80%</td>
<td>6</td>
<td>22</td>
<td>9</td>
</tr>
<tr>
<td>Real Up 100%</td>
<td>7</td>
<td>25</td>
<td>10</td>
</tr>
<tr>
<td>Real Up 144%</td>
<td>176</td>
<td>436</td>
<td>202</td>
</tr>
<tr>
<td>Real Up 168%</td>
<td>493</td>
<td>1181</td>
<td>535</td>
</tr>
</tbody>
</table>

waiting times in seconds
## System Comparison: Travel Times

<table>
<thead>
<tr>
<th>scenario</th>
<th>conventional</th>
<th>IA system</th>
<th>DA system</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\alpha_{0.5}$</td>
<td>$\alpha_{0.9}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Real Up 80%</td>
<td>61</td>
<td>113</td>
<td>65</td>
</tr>
<tr>
<td>Real Up 100%</td>
<td>71</td>
<td>129</td>
<td>75</td>
</tr>
<tr>
<td>Real Up 144%</td>
<td>260</td>
<td>515</td>
<td>279</td>
</tr>
<tr>
<td>Real Up 168%</td>
<td>566</td>
<td>1253</td>
<td>611</td>
</tr>
</tbody>
</table>

*travel times in seconds*
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

<table>
<thead>
<tr>
<th>quantile</th>
<th>IA system</th>
<th>(Real Up 168%)</th>
<th>DA system</th>
<th>(Real Up 144%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>gap [%]</td>
<td>time [s]</td>
<td>$</td>
<td>\mathcal{R}_u</td>
</tr>
<tr>
<td>$\alpha_{0.5}$</td>
<td>0.0</td>
<td>0.01</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>$\alpha_{0.75}$</td>
<td>0.0</td>
<td>0.02</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>$\alpha_{0.9}$</td>
<td>0.0</td>
<td>0.15</td>
<td>2</td>
<td>0.0</td>
</tr>
<tr>
<td>$\alpha_{1.0}$</td>
<td>0.1</td>
<td>1.21</td>
<td>5</td>
<td>0.7</td>
</tr>
</tbody>
</table>
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.
  → 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.
  \[\Rightarrow\] 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.
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
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).
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:

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

$1 3 5 7 9$

$2 4 6 8 10$
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.
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.
- **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 $\gamma$
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 $m = 3, B = 5, C = 15$

<table>
<thead>
<tr>
<th>#Items</th>
<th>FixedColors</th>
<th>GreedyFit</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.028, 0.657, 0.971, 1.000, 1.000</td>
<td>0.094, 1.000, 1.000, 1.000, 1.000</td>
</tr>
</tbody>
</table>

Shown is the cumulative distribution function!
Example: Stochastic Dominance for Bin Colorfulness

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

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

Shown is the cumulative distribution function!
Theorem ([Hiller, Vredeveld ’08])

Assume that the color sequence $\Sigma$ is generated by choosing each color independently at random according to color distribution $\gamma$. Then

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

Proof involves some probability tools …

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

…but it is straightforward.
Consequences of the result

- $X \leq_{st} Y \implies \mathbb{E}[X] \leq \mathbb{E}[Y]$
Consequences of the result

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

- $X \leq_{st} Y \implies \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{st} Y \implies f(X) \leq_{st} f(Y)$ for every non-decreasing function $f$
- better average competitive ratio, i.e., $\frac{\mathbb{E}[GF_n]}{\mathbb{E}[OPT_n]} \leq \frac{\mathbb{E}[OB_n]}{\mathbb{E}[OPT_n]}$
Consequences of the result

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

- $X \leq_{st} Y \implies \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{st} Y \implies f(X) \leq_{st} f(Y)$ for every non-decreasing function $f$
- Better average competitive ratio, i.e., $\frac{\mathbb{E}[GF_n]}{\mathbb{E}[OPT_n]} \leq \frac{\mathbb{E}[OB_n]}{\mathbb{E}[OPT_n]}$
- Stochastic dominance for the uniform distribution is equivalent to a bijective analysis [Angelopoulos et al '07] result:

<table>
<thead>
<tr>
<th>GF</th>
<th>OB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$: 3</td>
<td>4</td>
</tr>
<tr>
<td>$\sigma_2$: 2</td>
<td>3</td>
</tr>
<tr>
<td>$\sigma_3$: 3</td>
<td>4</td>
</tr>
<tr>
<td>$\sigma_4$: 4</td>
<td>3</td>
</tr>
<tr>
<td>$\sigma_5$: 2</td>
<td>2</td>
</tr>
</tbody>
</table>
Consequences of the result

- $X \leq_{st} Y \implies \mathbb{E}[X] \leq \mathbb{E}[Y]$
- $X \leq_{st} Y \implies f(X) \leq_{st} f(Y)$ 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:

<table>
<thead>
<tr>
<th>GF</th>
<th>OB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$: 3</td>
<td>4</td>
</tr>
<tr>
<td>$\sigma_2$: 2</td>
<td>3</td>
</tr>
<tr>
<td>$\sigma_3$: 3</td>
<td>4</td>
</tr>
<tr>
<td>$\sigma_4$: 4</td>
<td>3</td>
</tr>
<tr>
<td>$\sigma_5$: 2</td>
<td>2</td>
</tr>
</tbody>
</table>
Consequences of the result

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

\[
\begin{array}{ccc}
\text{GF} & \varphi & \text{OB} \\
\sigma_1: & 3 & \rightarrow 4 \\
\sigma_2: & 2 & \rightarrow 3 \\
\sigma_3: & 3 & \rightarrow 4 \\
\sigma_4: & 4 & \rightarrow 3 \\
\sigma_5: & 2 & \rightarrow 2 \\
\end{array}
\]
Consequences of the result

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

$$GF(\sigma) \leq OB(\varphi(\sigma)) \quad \forall \sigma$$

<table>
<thead>
<tr>
<th>$\sigma$</th>
<th>$GF$</th>
<th>$\varphi$</th>
<th>$OB$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sigma_1$</td>
<td>3</td>
<td>$\varphi$</td>
<td>4</td>
</tr>
<tr>
<td>$\sigma_2$</td>
<td>2</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>$\sigma_3$</td>
<td>3</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>$\sigma_4$</td>
<td>4</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>$\sigma_5$</td>
<td>2</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
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.


Philipp Friese and Jörg Rambau.
Online-optimization of a multi-elevator transport system with reoptimization algorithms based on set-partitioning models. 
also available as ZIB Report 05-03.

Dietrich Hauptmeier, Sven O. Krumke, and Jörg Rambau.
The online Dial-a-Ride problem under reasonable load.

Sven Oliver Krumke, Willem E. de Paepe, Leen Stougie, and Jörg Rambau.

Online bin coloring.

Sven Oliver Krumke, Jörg Rambau, and Luis M. Torres.

Realtime-dispatching of guided and unguided automobile service units with soft time windows.