Fast and Memory-Efficient Algorithms for Evacuation Problems

Miriam Schlöter & Martin Skutella

Aussois 2017
Main Results
[S., Skutella, 2017]
Main Results
[S., Skutella, 2017]

New strongly polynomial time algorithm for the
Quickest Transshipment Problem
Main Results
[S., Skutella, 2017]

New strongly polynomial time algorithm for the **Quickest Transshipment Problem**

Polynomial space algorithm for the **Earliest Arrival Transshipment Problem**
Main Results
[S., Skutella, 2017]

New strongly polynomial time algorithm for the Quickest Transshipment Problem

Polynomial space algorithm for the Earliest Arrival Transshipment Problem
Flows over Time

Definition
Flows over Time

Definition

Many real life problems crucially depend on *time*:
- logistic
- public transport
- evacuation problems
Flows over Time

Definition

Many real life problems crucially depend on *time*:
- logistic
- public transport
- evacuation problems

*Flows over time* are like…

…like classical (static) network flows + *time component*:

- flow needs time to travel through an arc $a$:
  arc $a$ has a *transit time* $\tau_a$ (length)
- a bounded amount of flow can enter an arc $a$ per time unit:
  arc $a$ has a *capacity* $u_a$ (width)
Flows over Time

Definition
Flows over Time

Definition

dynamic network $\mathcal{N} = (G = (V, A), u, \tau, S^+, S^-)$: digraph $G = (V, A)$ with capacities $u$, transit times $\tau$, sources $S^+ \subset V$ and sinks $S^- \subset V$
**Flows over Time**

**Definition**

A *dynamic network* $\mathcal{N} = (G=(V,A), u, \tau, S^+, S^-)$: digraph $G = (V,A)$ with capacities $u$, transit times $\tau$, *sources* $S^+ \subset V$ and *sinks* $S^- \subset V$.
Flows over Time

Definition

dynamic network $\mathcal{N} = (G = (V, A), u, \tau, S^+, S^-)$: digraph $G = (V, A)$ with capacities $u$, transit times $\tau$, sources $S^+ \subset V$ and sinks $S^- \subset V$
**Flows over Time**

**Definition**

*dynamiic network* $\mathcal{N} = (G=(V,A), u, \tau, S^+, S^-)$: digraph $G = (V,A)$ with capacities $u$, transit times $\tau$, *sources* $S^+ \subset V$ and *sinks* $S^- \subset V$
**Flows over Time**

**Definition**

*dynamic network* \( \mathcal{N} = (G = (V,A), u, \tau, S^+, S^-) \): digraph \( G = (V,A) \) with capacities \( u \), transit times \( \tau \), *sources* \( S^+ \subset V \) and *sinks* \( S^- \subset V \)
**Flow over Time**

**Definition**

A *dynamic network* $\mathcal{N} = (G=(V,A),u,\tau,S^+,S^-)$ is a digraph $G = (V,A)$ with capacities $u$, transit times $\tau$, sources $S^+ \subset V$ and sinks $S^- \subset V$.
Flows over Time

Definition

dynamic network $\mathcal{N} = (G=(V,A),u,\tau,S^+,S^-)$: digraph $G = (V,A)$ with capacities $u$, transit times $\tau$, sources $S^+ \subset V$ and sinks $S^- \subset V$
**Definition**

*Dynamic network* $\mathcal{N} = (G=(V,A), u, \tau, S^+, S^-)$: digraph $G = (V,A)$ with capacities $u$, transit times $\tau$, *sources* $S^+ \subset V$ and *sinks* $S^- \subset V$
**Flows over Time**

**Definition**

A *dynamic network* $\mathcal{N} = (G=(V,A), u, \tau, S^+, S^-)$ is a directed graph $G = (V,A)$ with capacities $u$, transit times $\tau$, *sources* $S^+ \subset V$ and *sinks* $S^- \subset V$. The figure illustrates a dynamic network at time $t = 7$. The network consists of vertices connected by directed edges, each with a specified capacity and transit time.
Flows over Time

Definition

dynamic network $\mathcal{N} = (G = (V, A), u, \tau, S^+, S^-)$: digraph $G = (V, A)$ with capacities $u$, transit times $\tau$, sources $S^+ \subset V$ and sinks $S^- \subset V$
**Definition**

*dynamic network* \( \mathcal{N} = (G = (V, A), u, \tau, S^+, S^-) \): digraph \( G = (V, A) \) with capacities \( u \), transit times \( \tau \), *sources* \( S^+ \subset V \) and *sinks* \( S^- \subset V \)

**feasible flow over time** \( f \) in \( \mathcal{N} \) with *time horizon* \( T \):

- \( f : A \times [0, \infty) \rightarrow [0, \infty) \)
- \( f \) respects flow conservation and capacities
- no flow in \( \mathcal{N} \) after time \( T \)
Submodular Functions

**Definition:** finite set $U$, $g:2^U \rightarrow \mathbb{R}$ is *submodular* if:
$$g(X) + g(Y) \geq g(X \cup Y) + g(X \cap Y) \quad \text{for all } X, Y \subseteq U$$
Submodular Functions

**Definition:** finite set $U$, $g:2^U \to \mathbb{R}$ is submodular if:

$$g(X) + g(Y) \geq g(X \cup Y) + g(X \cap Y) \text{ for all } X, Y \subseteq U$$

**Submodular Function Minimization (SFM):**

$$\min\{g(X) | X \subseteq U\}$$
**Submodular Functions**

**Definition:** finite set $U$, $g: 2^U \rightarrow \mathbb{R}$ is **submodular** if:

$$g(X) + g(Y) \geq g(X \cup Y) + g(X \cap Y) \text{ for all } X, Y \subseteq U$$

**Submodular Function Minimization (SFM):**

$$\min \{ g(X) | X \subseteq U \}$$

Possible in strongly polynomial time:

- Ellipsoid Method [Grötschel et al., '81, '88]
- First combinatorial algorithms [Schrijver, Iwata et al., 2000]
- Fastest combinatorial algorithm, $O(n^3 \log^2 n \cdot \gamma + n^4 \log^{O(1)} n )$ [Lee et al., 2015]
Flows over Time

A Submodular Function

\[ \mathcal{N} = (G=(V,A), u, \tau, S^+, S^-) \]
Flows over Time

A Submodular Function

\[ \mathcal{N} = (G = (V,A), u, \tau, S^+, S^-) \]

**Definition:** For fixed \( T > 0 \), define

\[ o^T : 2^{S^+ \cup S^-} \rightarrow \mathbb{R} \] by,

\[ o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T \]
Flows over Time

A Submodular Function

Definition: For fixed $T > 0$, define

$$o^T: 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$$

by,

$$o^T(A) := \max \text{. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$
Definition: For fixed $T > 0$, define

$$o^T: 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$$

by,

$$o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$
**Definition:** For fixed $T > 0$, define

$$o^T : 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$$

by,

$$o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$

Fast and Memory-Efficient Algorithms for Evacuation Problems
**Definition:** For fixed $T > 0$, define $o^T : 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$ by,

$$o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$
**Definition:** For fixed $T > 0$, define

$$o^T: 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$$

by,

$$o^T(A) := \max \text{ amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$
**Definition:** For fixed $T > 0$, define

$$o^T : 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$$

by,

$$o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$
Flows over Time
A Submodular Function

**Definition:** For fixed $T > 0$, define $o^T : 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$ by,

$$o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$

- $U_a = \tau_a = 1$
- $S_1$
- $S_2$
- $T = 4$
- $o^T(\{s_1\}) = 2$
**Definition:** For fixed $T > 0$, define

$$o^T: 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$$

by,

$$o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$

\[ u_a = \tau_a = 1 \]

\[ o^T(\{s_1\}) = 2 \]

\[ o^T(\{s_2\}) = 2 \]

\[ T = 4 \]
Flows over Time
A Submodular Function

**Definition:** For fixed $T > 0$, define

$$o^T : 2^{S^+ \cup S^-} \rightarrow \mathbb{R}$$

by,

$$o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$

\[u_a = \tau_a = 1\]

\[T = 4\]

$$o^T(\{s_1\}) = 2\]

$$o^T(\{s_2\}) = 2\]

$$o^T(\{s_1, s_2\}) = 2$$
Definition: For fixed $T > 0$, define

$$o^T: 2^{S^+ \cup S^-} \rightarrow \mathbb{R} \text{ by,}$$

$$o^T(A) := \text{max. amount of flow that can be sent from sources in } A \text{ to sinks not in } A \text{ until time } T$$

The function $o^T$ is submodular!
Quickest Transshipments

Definition

\[ N \]

sources \( S^+ \)  
sinks \( S^- \)

Fast and Memory-Efficient Algorithms for Evacuation Problems
Quickest Transshipments

Definition

Supply/demand function $v: S^+ \cup S^- \rightarrow \mathbb{R}$

Sources $S^+$: $S_1, S_2, S_3, S_4$

Sinks $S^-$: $t_1, t_2, t_3$
Quickest Transshipments

Definition

supply/demand function \( v : S^+ \cup S^- \rightarrow \mathbb{R} \)

\( \mathcal{N} \)

sources \( S^+ \)

\[ v(s_i) \geq 0, \quad i = 1, \ldots, |S^+| \]

\[ v(t_i) \leq 0, \quad i = 1, \ldots, |S^-| \]

\[ v(S^+ \cup S^-) = v(s_1) + \ldots + v(s_4) + v(t_1) + \ldots + v(t_3) = 0 \]
Quickest Transshipments

Definition

A supply/demand function \( v: S^+ \cup S^- \rightarrow \mathbb{R} \)

Sources \( S^+ \):
- \( v(s_i) \geq 0, \ i = 1, \ldots, |S^+| \)
- \( v(t_i) \leq 0, \ i = 1, \ldots, |S^-| \)

Sinks \( S^- \):
- \( v(S^+ \cup S^-) = v(s_1) + \ldots + v(s_4) + v(t_1) + \ldots + v(t_3) = 0 \)

Quickest Transshipment problem: \((\mathcal{N}, v)\)

Find flow over time \( f \) from \( S^+ \) to \( S^- \) which fulfills the supplies and demands \textit{as quickly as possible}.

Fast and Memory-Efficient Algorithms for Evacuation Problems
Quickest Transshipment problem: $(\mathcal{N}, \nu)$
Find flow over time $f$ from $S^+$ to $S^-$ which fulfills the supplies and demands as quickly as possible
Quickest Transshipment problem: \((\mathcal{N}, \nu)\)

Find flow over time \(f\) from \(S^+\) to \(S^-\) which fulfills the supplies and demands as quickly as possible

Solution in two steps:
1. Compute minimal feasible time horizon \(T\)
2. Compute flow \(f\) solving the transshipment problem \((\mathcal{N}, \nu, T)\)
Quickest Transshipment problem: \((\mathcal{N}, \nu)\)

Find flow over time \(f\) from \(S^+\) to \(S^-\) which fulfills the supplies and demands as quickly as possible

**Solution in two steps:**

1. Compute minimal feasible time horizon \(T\)
2. Compute flow \(f\) solving the transshipment problem \((\mathcal{N}, \nu, T)\)

**1. Klinz:** \((\mathcal{N}, \nu, \theta)\) is feasible if and only if

\[\sigma^\theta(A) - \nu(A) \geq 0\]
for all \(A \subseteq S^+ \cup S^-\)
**Quickest Transshipment problem:** \((\mathcal{N}, \nu)\)

Find flow over time \(f\) from \(S^+\) to \(S^-\) which fulfills the supplies and demands as quickly as possible

**Solution in two steps:**
1. Compute minimal feasible time horizon \(T\)
2. Compute flow \(f\) solving the transshipment problem \((\mathcal{N}, \nu, T)\)

**1. Klinz:** \((\mathcal{N}, \nu, \theta)\) is feasible if and only if
   \[ \sigma^\theta(A) - \nu(A) \geq 0 \text{ for all } A \subseteq S^+ \cup S^- \]

   → Compute \(T\) via parametric submodular function minimization
Quickest Transshipment problem: $(\mathcal{N}, \nu)$
Find flow over time $f$ from $S^+$ to $S^-$ which fulfills the supplies and demands as quickly as possible

Solution in two steps:
1. Compute minimal feasible time horizon $T$
2. Compute flow $f$ solving the transshipment problem $(\mathcal{N}, \nu, T)$

2. Algorithm of Hoppe & Tardos:
   Needs many parametric submodular function minimizations
Quickest Transshipment problem: \((\mathcal{N}, v)\)
Find flow over time \(f\) from \(S^+\) to \(S^-\) which fulfills the supplies and demands as quickly as possible

Solution in two steps:
1. Compute minimal feasible time horizon \(T\)
2. Compute flow \(f\) solving the transshipment problem \((\mathcal{N}, v, T)\)

2. Algorithm of Hoppe & Tardos:
Needs many parametric submodular function minimizations

Our Contribution: Algorithm for solving a given (feasible) transshipment problem that only needs one submodular function minimization
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)

- A suitable convex combination can be computed by *one* submodular function minimization
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- A suitable convex combination can be computed by *one* submodular function minimization

**lex-max flow over time:**

Flow over time \(f_<\) in \(\mathcal{N}\) that *lexicographically maximizes* the amount of flow out of each terminal wrt. some total ordering \(<\) on \(|S^+ \cup S^-|\).
Quickest Transshipments

Main Result

Given: Feasible transshipment problem $(\mathcal{N}, v, T)$

Theorem:

- $(\mathcal{N}, v, T)$ can be solved by a convex combination of $d \leq |S^+ \cup S^-|$ many \textit{lex-max flows} over time with time horizon $T$

- a suitable convex combination can be computed by \textit{one submodular function minimization}

\textbf{lex-max flow over time:}
Flow over time $f_<$ in $\mathcal{N}$ that \textit{lexicographically maximizes} the amount of flow out of each terminal wrt. some total ordering $<$ on $|S^+ \cup S^-|$.

→ Strongly polynomial time algorithm by Hoppe and Tardos, ‘95
Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**
- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\).
- a suitable convex combination can be computed by *one* submodular function minimization.
Fast and Memory-Efficient Algorithms for Evacuation Problems

Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

Theorem:

\((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many \textit{lex-max flows} over time with time horizon \(T\)

- a suitable convex combination can be computed by \textit{one submodular function minimization}

\(\rightarrow\) New Algorithm for the Quickest Transshipment Problem \textbf{BUT} we only get a fractional solution
Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- A suitable convex combination can be computed by *one* submodular function minimization
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- a suitable convex combination can be computed by *one submodular function minimization*
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

Theorem:

\((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many \textit{lex-max flows} over time with time horizon \(T\)

- a suitable convex combination can be computed by \textit{one submodular function minimization}

\[ v(s_1) = v(s_2) = 1 \]
\[ u_a = \tau_a = 1 \]

\(T = 4\)
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- A suitable convex combination can be computed by *one* submodular function minimization

\(v(s_1) = v(s_2) = 1\)
\(u_a = \tau_a = 1\)

\(T = 4\)
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)

- A suitable convex combination can be computed by *one* submodular function minimization

\(v(s_1) = v(s_2) = 1\)

\(u_a = \tau_a = 1\)

\(T = 4\)
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

• \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)

• a suitable convex combination can be computed by one *submodular function minimization*
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- A suitable convex combination can be computed by one *submodular function minimization*

\(v(s_1) = v(s_2) = 1\)
\(u_a = \tau_a = 1\)

\(T = 4\)
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)

- A suitable convex combination can be computed by *one* submodular function minimization

\[v(s_1) = v(s_2) = 1\]
\[u_a = \tau_a = 1\]

\(T = 4\)
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)

- a suitable convex combination can be computed by *one submodular function minimization*

**Diagram:**

\[
v(s_1) = v(s_2) = 1
\]
\[
u_a = \tau_a = 1
\]
\[
T = 4
\]
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- A suitable convex combination can be computed by *one submodular function minimization*
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ u S^-|\) many *lex-max flows* over time with time horizon \(T\)
- A suitable convex combination can be computed by *one* submodular function minimization
**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\).

- A suitable convex combination can be computed by *one* submodular function minimization.
Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- a suitable convex combination can be computed by *one submodular function minimization*

\[ v(s_1) = v(s_2) = 1 \\
\text{and} \\
u_a = \tau_a = 1 \]

\[ T = 4 \]
Quickest Transshipments

Main Result

Given: Feasible transshipment problem $\mathcal{N}, v, T$

**Theorem:**

- $(\mathcal{N}, v, T)$ can be solved by a convex combination of $d \leq |S^+ \cup S^-|$ many *lex-max flows* over time with time horizon $T$
- A suitable convex combination can be computed by *one* submodular function minimization

![Diagram of transshipment problem with nodes $S_1$, $S_2$, and $t$, with $v(s_1) = v(s_2) = 1$, $u_a = \tau_a = 1$, and $T = 4$.]
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- A suitable convex combination can be computed by *one submodular function minimization*

![Diagram](image)

\(v(s_1) = v(s_2) = 1\)
\(u_a = \tau_a = 1\)

\(T = 4\)
**Quickest Transshipments**

**Main Result**

**Given:** Feasible transshipment problem \( (\mathcal{N}, v, T) \)

**Theorem:**

- \( (\mathcal{N}, v, T) \) can be solved by a convex combination of \( d \leq |S^+ \cup S^-| \) many *lex-max flows* over time with time horizon \( T \)

- A suitable convex combination can be computed by *one submodular function minimization*

\[ v(s_1) = v(s_2) = 1 \]
\[ u_a = \tau_a = 1 \]

\[ T = 4 \]
Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)
- A suitable convex combination can be computed by *one* submodular function minimization

\(v(s_1) = v(s_2) = 1\)
\(u_a = \tau_a = 1\)
\(T = 4\)
Theorem:

- $(\mathcal{N}, v, T)$ can be solved by a convex combination of $d \leq |S^+ \cup S^-|$ many *lex-max flows* over time with time horizon $T$
- A suitable convex combination can be computed by *one* submodular function minimization

**Given:** Feasible transshipment problem $(\mathcal{N}, v, T)$

**Main Result**
Quickest Transshipments

Main Result

Given: Feasible transshipment problem $(\mathcal{N}, v, T)$

Theorem:

- $(\mathcal{N}, v, T)$ can be solved by a convex combination of $d \leq |S^+ \cup S^-|$ many *lex-max flows* over time with time horizon $T$
- A suitable convex combination can be computed by *one submodular function minimization*

Given: Feasible transshipment problem $(\mathcal{N}, v, T)$

$\nu(s_1) = \nu(s_2) = 1$
$u_a = \tau_a = 1$

$T = 4$
Quickest Transshipments
Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Theorem:**

- \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many *lex-max flows* over time with time horizon \(T\)

- A suitable convex combination can be computed by *one* submodular function minimization
Quickest Transshipments

Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

Theorem:

• \((\mathcal{N}, v, T)\) can be solved by a convex combination of \(d \leq |S^+ \cup S^-|\) many \textit{lex-max flows} over time with time horizon \(T\)

• a suitable convex combination can be computed by \textit{one submodular function minimization}

Key idea for the proof: Open blackbox of SFM
Submodular Function Minimization

\[ g : 2^U \rightarrow \mathbb{R}, \text{ submodular function, } x \in \mathbb{R}^U, \ x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|}) \]
Submodular Function Minimization

$g : 2^U \rightarrow \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

Base Polytope $\mathcal{B}(g)$:

$\mathcal{B}(g) := \{x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U)\}$
Submodular Function Minimization

$g : 2^U \rightarrow \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

**Base Polytope** $\mathcal{B}(g)$:

$\mathcal{B}(g) := \{x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U)\}$

**min-max Theorem** [Edmonds, '70]:

$$\min\{g(X) | X \subseteq U\} = \max\{x^{-}(U) | x \in \mathcal{B}(g)\}$$

$x^{-}(U) := \text{sum of all negative components of } x \in \mathcal{B}(g)$
Submodular Function Minimization

\( g : 2^U \rightarrow \mathbb{R} \), submodular function, \( x \in \mathbb{R}^U \), \( x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|}) \)

**Base Polytope** \( \mathcal{B}(g) \):
\[
\mathcal{B}(g) := \{ x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U) \}
\]

**min-max Theorem** [Edmonds, ’70]:
\[
\min \{ g(X) | X \subseteq U \} = \max \{ x^- (U) | x \in \mathcal{B}(g) \}
\]
\( x^- (U) := \text{sum of all negative components of } x \in \mathcal{B}(g) \)

Idea for SFM:

**Base Polytope** \( \mathcal{B}(g) \)
Submodular Function Minimization

\( g : 2^U \rightarrow \mathbb{R} \), submodular function, \( x \in \mathbb{R}^U \), \( x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|}) \)

**Base Polytope** \( \mathcal{B}(g) \):

\[
\mathcal{B}(g) := \{ x \in \mathbb{R}^U \mid x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U) \}
\]

**min-max Theorem** [Edmonds, ’70]:

\[
\min \{ g(X) \mid X \subseteq U \} = \max \{ x^-(U) \mid x \in \mathcal{B}(g) \}
\]

\( x^-(U) := \text{sum of all negative components of } x \in \mathcal{B}(g) \)

Idea for SFM:

\[ \mathcal{B}(g) \]
Submodular Function Minimization

$g : 2^U \rightarrow \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

**Base Polytope $\mathcal{B}(g)$:**

$\mathcal{B}(g) := \{x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U)\}$

**min-max Theorem [Edmonds, ’70]:**

$\min\{g(X) | X \subseteq U\} = \max\{x^{-}(U) | x \in \mathcal{B}(g)\}$

$x^{-}(U) := \text{sum of all negative components of } x \in \mathcal{B}(g)$

Idea for SFM:

$\mathcal{B}(g)$
Submodular Function Minimization

\( g : 2^U \rightarrow \mathbb{R} \), submodular function, \( x \in \mathbb{R}^U \), \( x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|}) \)

**Base Polytope** \( \mathcal{B}(g) \):

\[ \mathcal{B}(g) := \{ x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U) \} \]

**min-max Theorem** [Edmonds, ’70]:

\[ \min \{ g(X) | X \subseteq U \} = \max \{ x^-(U) | x \in \mathcal{B}(g) \} \]

\( x^-(U) := \text{sum of all negative components of } x \in \mathcal{B}(g) \)

Idea for SFM:
Submodular Function Minimization

$g : 2^U \rightarrow \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

**Base Polytope** $\mathcal{B}(g)$:

$\mathcal{B}(g) := \{ x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subseteq U, x(U) = f(U) \}$

**min-max Theorem** [Edmonds, ’70]:

$\min \{ g(X) | X \subseteq U \} = \max \{ x^{-}(U) | x \in \mathcal{B}(g) \}$

$x^{-}(U) := $ sum of all negative components of $x \in \mathcal{B}(g)$

Idea for SFM:

$\mathcal{B}(g)$
Submodular Function Minimization

$g : 2^U \rightarrow \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

**Base Polytope** $\mathcal{B}(g)$:

$\mathcal{B}(g) := \{x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U)\}$

**min-max Theorem** [Edmonds, ’70]:

$$\min \{g(X) | X \subseteq U\} = \max \{x^-(U) | x \in \mathcal{B}(g)\}$$

$x^-(U) := \text{sum of all negative components of } x \in \mathcal{B}(g)$

Idea for SFM:

\[
\mathcal{B}(g)
\]
Submodular Function Minimization

\( g : 2^U \to \mathbb{R} \), submodular function, \( x \in \mathbb{R}^U \), \( x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|}) \)

**Base Polytope** \( \mathcal{B}(g) \):

\[
\mathcal{B}(g) := \{ x \in \mathbb{R}^U \mid x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U) \}
\]

**min-max Theorem** [Edmonds, ’70]:

\[
\min\{g(X) \mid X \subseteq U\} = \max\{x^-(U) \mid x \in \mathcal{B}(g)\}
\]

\( x^-(U) := \) sum of all negative components of \( x \in \mathcal{B}(g) \)

**Idea for SFM:**

\( \mathcal{B}(g) \)

\[ x^* = \arg\max\{x^-(U) \mid x \in \mathcal{B}(g)\} \]

as convex combination of vertices of \( \mathcal{B}(g) \)
Submodular Function Minimization

$g : 2^U \to \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

**Base Polytope** $B(g)$:

$B(g) := \{x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U)\}$

**min-max Theorem** [Edmonds, ‘70]:

$\min\{g(X) | X \subseteq U\} = \max\{x^{-}(U) | x \in B(g)\}$

$x^{-}(U) := \text{sum of all negative components of } x \in B(g)$

**Idea for SFM:**

Output: $x^* = \text{argmax}\{x^{-}(U) | x \in B(g)\}$ as convex combination of vertices of $B(g)$

**Question:** How do the vertices of $B(g)$ look like!?
Fast and Memory-Efficient Algorithms for Evacuation Problems

Submodular Function Minimization

\( f : 2^V \rightarrow \mathbb{R} \), submodular function, \( x \in \mathbb{R}^V \), \( x(A) = x(a_1) + x(a_2) + \ldots + x(a_{|A|}) \)

Base Polytope:

\[ \mathcal{B}(f) := \{ x \in \mathbb{R}^V \mid x(A) \leq f(A) \text{ for all } A \subset V, x(V) = f(V) \} \]

**min-max Theorem** [Edmonds, ’70]:

**Theorem** [Edmonds, ’70]:

vertices of \( \mathcal{B}(g) \) \( \leftrightarrow \) orderings \( < \) of \( U \)

Idea for SFM:

Output: \( x^* = \arg \max (x(V) \mid x \in \mathcal{B}(f)) \)
as convex combination of vertices of \( \mathcal{B}(f) \)

Problem: How to the vertices of \( \mathcal{B}(f) \) look like!?
Submodular Function Minimization

$g : 2^U \rightarrow \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

**Base Polytope** $\mathcal{B}(g)$:

$\mathcal{B}(g) := \{x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U)\}$

**min-max Theorem** [Edmonds, ’70]:

$$\min \{g(X) | X \subseteq U\} = \max \{x^-(U) | x \in \mathcal{B}(g)\}$$

$x^-(U)$: sum of all negative components of $x \in \mathcal{B}(g)$

**Idea for SFM:**

Output: $x^* = \arg\max \{x^-(U) | x \in \mathcal{B}(g)\}$ as convex combination of vertices of $\mathcal{B}(g)$
Submodular Function Minimization

$g : 2^U \rightarrow \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

**Base Polytope** $\mathcal{B}(g)$:

$\mathcal{B}(g) := \{ x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subset U, x(U) = f(U) \}$

**min-max Theorem** [Edmonds, ’70]:

$$\min\{g(X) | X \subseteq U\} = \max\{x^-(U) | x \in \mathcal{B}(g)\}$$

$x^-(U) := \text{sum of all negative components of } x \in \mathcal{B}(g)$

**Cunningham, 1984/85:**

$\mathcal{B}(g)$

Output: $x^* = \arg\max\{x^-(U) | x \in \mathcal{B}(g)\}$ as convex combination of vertices of $\mathcal{B}(g)$
Submodular Function Minimization

$g : 2^U \to \mathbb{R}$, submodular function, $x \in \mathbb{R}^U$, $x(A) := x(a_1) + x(a_2) + \ldots + x(a_{|A|})$

**Base Polytope $\mathcal{B}(g)$:**

$\mathcal{B}(g) := \{x \in \mathbb{R}^U | x(A) \leq g(A) \text{ for all } A \subseteq U, x(U) = f(U)\}$

**min-max Theorem [Edmonds, ’70]:**

$$\min\{g(X) | X \subseteq U\} = \max\{x^{-}(U) | x \in \mathcal{B}(g)\}$$

$x^{-}(U)$: sum of all negative components of $x \in \mathcal{B}(g)$

**Cunningham, 1984/85:**

Output: $x^* = \arg\max\{x^{-}(U) | x \in \mathcal{B}(g)\}$ as convex combination of vertices of $\mathcal{B}(g)$

→ Many combinatorial SFM algorithms use this principle
Quickest Transshipments

Proof of Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)
Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Know:** Algorithm to minimize \(o^T - v\) finds

\[ x^* = \arg \max \left\{ x^T (S^+ \cup S^-) \mid x \in B(o^T - v) \right\} \]

as a convex combination of vertices of \(B(o^T - v)\)
Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Know:** Algorithm to minimize \(o^T - v\) finds

\[ x^* = \arg\max\{x^*(S^+ \cup S^-) | x \in \mathcal{B}(o^T - v) \} \]

as a convex combination of vertices of \(\mathcal{B}(o^T - v)\)
Quickest Transshipments

Proof of Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Know:** Algorithm to minimize \(o^Tv\) finds

\[ x^* = \text{argmax}\{x^*(S^+ \cup S^-) | x \in \mathcal{B}(o^Tv)\} \]

as a convex combination of vertices of \(\mathcal{B}(o^Tv)\)
Quickest Transshipments
Proof of Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Know:** Algorithm to minimize \(o^T - v\) finds
\[ x^* = \arg\max\{x^{-1}(S^+ \cup S^-) | x \in \mathcal{B}(o^T - v)\} \]
as a convex combination of vertices of \(\mathcal{B}(o^T - v)\)

**Because of feasibility (Klinz) and min-max Theorem:**
\[ 0 = \max\{x^{-1}(V) | x \in \mathcal{B}(o^T - v)\} = \min\{o^T(X) - v(X) | X \subseteq S^+ \cup S^-\} \]
Quickest Transshipments

Proof of Main Result

**Given:** Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Know:** Algorithm to minimize \(o^T - v\) finds
\[
x^* = \arg\max\{x^-(S^+ \cup S^-) | x \in \mathcal{B}(o^T - v)\}
\]
as a convex combination of vertices of \(\mathcal{B}(o^T - v)\)

**Because of feasibility (Klinz) and min-max Theorem:**
\[
0 = \max\{x^-(V) | x \in \mathcal{B}(o^T - v)\} = \min\{o^T(X) - v(X) | X \subseteq S^+ \cup S^-\}
\]
Quickest Transshipments

Proof of Main Result

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Know:** Algorithm to minimize \(o^T - v\) finds

\[
x^* = \arg\max \{ x^-(S^+ \cup S^-) | x \in \mathcal{B}(o^T - v) \}
\]

as a convex combination of vertices of \(\mathcal{B}(o^T - v)\)

**Because of feasibility (Klinz) and min-max Theorem:**

\[
0 = \max \{ x^-(V) | x \in \mathcal{B}(o^T - v) \} = \min \{ o^T(X) - v(X) | X \subseteq S^+ \cup S^- \}
\]

Also:

\[
\mathcal{B}(o^T - v) + (v(S))_{s \in S^+ \cup S^-} = \mathcal{B}(o^T)
\]
**Quickest Transshipments**

*Proof of Main Result*

**Given:** Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Know:** Algorithm to minimize \(o^T - v\) finds
\[
x^* = \arg\max\{x^-(S^+ \cup S^-) | x \in \mathcal{B}(o^T - v)\}
\]
as a convex combination of vertices of \(\mathcal{B}(o^T - v)\)

**Because of feasibility (Klinz) and min-max Theorem:**
\[
0 = \max\{x^-(V) | x \in \mathcal{B}(o^T - v)\} = \min\{o^T(X) - v(X) | X \subseteq S^+ \cup S^-\}
\]

**Also:**
\[
\mathcal{B}(o^T - v) + (v(S))_{S \in S^+ \cup S^-} = \mathcal{B}(o^T)
\]
**Quickest Transshipments**

**Proof of Main Result**

**Given:** Feasible transshipment problem \((\mathcal{N}, v, T)\)

**Know:** Algorithm to minimize \(o^T - v\) finds

\[
x^* = \arg\max\{x^*(S^+ \cup S^-) | x \in \mathcal{B}(o^T - v)\}
\]
as a convex combination of vertices of \(\mathcal{B}(o^T - v)\)

**Because of feasibility (Klinz) and min-max Theorem:**

\[
0 = \max\{x^-(V) | x \in \mathcal{B}(o^T - v)\} = \min\{o^T(X) - v(X) | X \subseteq S^+ \cup S^-\}
\]

Also:

\[
\mathcal{B}(o^T - v) + (v(s))_{s \in S^+ \cup S^-} = \mathcal{B}(o^T)
\]
Quickest Transshipments

Proof of Main Result

lex-max flow

$\mathcal{B}(O^T)$

$(\nu(s))_{s \in S^+ \cup S^-}$
Quickest Transshipments

Proof of Main Result

ordering $< \text{ on } S^+ \cup S^- \iff \text{lex - max flow over time } f_\prec \leftrightarrow x_\prec \in \mathbb{R}^{S^+ \cup S^-}$

with $x_\prec(s) = |f_\prec(s)|_T$
Quickest Transshipments

Proof of Main Result

ordering \( < \) on \( S^+ \cup S^- \) \( \leftrightarrow \) lex - max flow over time \( f_\prec \leftrightarrow x_\prec \in \mathbb{R}^{S^+ \cup S^-} \)

with \( x_\prec(s) = |f_\prec(s)|_T \)

Also: Each ordering \( < \) on \( S^+ \cup S^- \) corresponds to a vertex \( v_\prec \) of \( \mathcal{B}(o^T) \)
Quickest Transshipments
Proof of Main Result

ordering $<$ on $S^+ \cup S^-$ $\leftrightarrow$ lex-max flow over time $f_\prec \leftrightarrow x_\prec \in \mathbb{R}^{S^+ \cup S^-}$ with $x_\prec(s) = |f_\prec(s)|_T$

Also: Each ordering $<$ on $S^+ \cup S^-$ corresponds to a vertex $v_\prec$ of $\mathcal{B}(o^T)$

Edmonds ’70, Minieka ’73, Megiddo ’74:
For each ordering $<$ on $S^+ \cup S^-$, we have $x_\prec = v_\prec$

Fast and Memory-Efficient Algorithms for Evacuation Problems
Quickest Transshipments

Proof of Main Result

ordering $< \text{ on } S^+ \cup S^-$ $\leftrightarrow$ lex-max flow over time $f_\prec \leftrightarrow x_\prec \in \mathbb{R}^{S^+ \cup S^-}$

with $x_\prec(s) = |f_\prec(s)|_T$

Also: Each ordering $< \text{ on } S^+ \cup S^-$ corresponds to a vertex $v_\prec$ of $\mathcal{B}(O^T)$

Edmonds ’70, Minieka ’73, Megiddo ’74:
For each ordering $< \text{ on } S^+ \cup S^-$, we have $x_\prec = v_\prec$

Fast and Memory-Efficient Algorithms for Evacuation Problems
Quickest Transshipments

Algorithm

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

Aim: Find a transshipment solving \((\mathcal{N}, v, T)\)
Quickest Transshipments

Algorithm

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

Aim: Find a transshipment solving \((\mathcal{N}, v, T)\)

1. minimize the submodular function \(o^T - v\):

\[
0 = \lambda_1 v_1 + \lambda_2 v_2 + \ldots + \lambda_d v_d,
\]

with: \(v_i\) vertex of \(B(o^T - v)\) corresponding to order \(<_i\) on \(S^+ \cup S^-\)
Quickest Transshipments
Algorithm

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)

Aim: Find a transshipment solving \((\mathcal{N}, v, T)\)

1. minimize the submodular function \(o^T-v\):
   \[
   0 = \lambda_1 v_1 + \lambda_2 v_2 + \ldots + \lambda_d v_d,
   \]
   with: \(v_i\) vertex of \(B(o^T-v)\) corresponding to order \(\prec_i\) on \(S^+ \cup S^-\)

2. compute lex-max flows over time \(f_1, f_2, \ldots, f_d\) in \(\mathcal{N}\) with time horizon \(T\) corresponding to \(\prec_1, \ldots, \prec_d\)
Quickest Transshipments

Algorithm

Given: Feasible transshipment problem \((\mathcal{N}, v, T)\)
Aim: Find a transshipment solving \((\mathcal{N}, v, T)\)

1. minimize the submodular function \(o^T - v\):
   \[
   0 = \lambda_1 v_1 + \lambda_2 v_2 + \ldots + \lambda_d v_d,
   \]
   with: \(v_i\) vertex of \(B(o^T - v)\) corresponding to order \(\prec_i\) on \(S^+ \cup S^-\)

2. compute lex-max flows over time \(f_1, f_2, \ldots f_d\) in \(\mathcal{N}\) with time horizon \(T\) corresponding to \(\prec_1, \ldots, \prec_d\)

3. \(f = \lambda_1 f_1 + \lambda_2 f_2 + \ldots + \lambda_d f_d\) solves \((\mathcal{N}, v, T)\)
Quickest Transshipments

Algorithm

Given: Quickest transshipment problem \((\mathcal{N}, v)\)
Aim: Find a transshipment solving \((\mathcal{N}, v)\)

1. do parametric submodular function minimization of \(o^\theta - v\):
   \[
   0 = \lambda_1 v_1 + \lambda_2 v_2 + \ldots + \lambda_d v_d
   \]
   minimal feasible time horizon \(T\)
   with: \(v_i\) vertex of \(B(o^T - v)\) corresponding to order \(\prec_i\) on \(S^+ \cup S^-\)

2. compute lex-max flows over time \(f_1, f_2, \ldots, f_d\) in \(\mathcal{N}\) with time horizon \(T\) corresponding to \(\prec_1, \ldots, \prec_d\)

3. \(f = \lambda_1 f_1 + \lambda_2 f_2 + \ldots + \lambda_d f_d\) solves \((\mathcal{N}, v)\)
Thank You!