Applications of the Mixed Packing and Covering Problem

Florian Diedrich Klaus Jansen

Institut für Informatik, Universität zu Kiel

AEOLUS 2007

Outline

Introduction

Algorithm

Sketch of Analysis

Applications

Conclusion

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}^M_+$ vector of continuous convex functions
- $g : B \to \mathbb{R}^M_+$ vector of continuous concave functions
- $a, b \in \mathbb{R}^M_{++}$ positive vectors

Problem:

$$\begin{array}{c} \text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}_+^M$ vector of continuous convex functions
- $g : B \to \mathbb{R}_+^M$ vector of continuous concave functions
- $a, b \in \mathbb{R}_{++}^M$ positive vectors

Problem:

$$\begin{array}{c}\text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset\end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}^M_+$ vector of continuous convex functions
- $g : B \to \mathbb{R}^M_+$ vector of continuous concave functions
- $a, b \in \mathbb{R}^M_{++}$ positive vectors

Problem:

$$\begin{array}{c}\text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset\end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}_+^M$ vector of continuous convex functions
- $g : B \to \mathbb{R}_+^M$ vector of continuous concave functions
- $a, b \in \mathbb{R}_{++}^M$ positive vectors

Problem:

$$\begin{array}{c}\text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset\end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}_+^M$ vector of continuous convex functions
- $g : B \to \mathbb{R}_+^M$ vector of continuous concave functions
- $a, b \in \mathbb{R}_{++}^M$ positive vectors

Problem:

$$\begin{array}{c}\text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset\end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}_+^M$ vector of continuous convex functions
- $g : B \to \mathbb{R}_+^M$ vector of continuous concave functions
- $a, b \in \mathbb{R}_{++}^M$ positive vectors

Problem:

$$\begin{array}{c} \text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}_+^M$ vector of continuous convex functions
- $g : B \to \mathbb{R}_+^M$ vector of continuous concave functions
- $a, b \in \mathbb{R}_{++}^M$ positive vectors

Problem:

$$\begin{array}{c}\text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset\end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}^M_+$ vector of continuous convex functions
- $g : B \to \mathbb{R}^M_+$ vector of continuous concave functions
- $a, b \in \mathbb{R}^M_{++}$ positive vectors

Problem:

$$\begin{array}{c} \text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

The Problem

- $N, M \in \mathbb{N}$
- $\emptyset \neq B \subseteq \mathbb{R}^N$ convex, compact
- $f : B \to \mathbb{R}_+^M$ vector of continuous convex functions
- $g : B \to \mathbb{R}_+^M$ vector of continuous concave functions
- $a, b \in \mathbb{R}_{++}^M$ positive vectors

Problem:

$$\begin{array}{c}\text{find } x \in B \text{ such that } f(x) \leq a, \quad g(x) \geq b \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset\end{array} \qquad (MPC)$$

W.l.o.g. $a = e = b$, $e \in \mathbb{R}^M$ unit vector

Linear Case

- $B \subseteq \mathbb{R}_+^N$ polytope
- f consists of linear functions
- g consists of linear functions

In this case, *MPC* is an LP with nonnegative coefficients (feasibility version).

Linear Case

- $B \subseteq \mathbb{R}_+^N$ polytope
- *f* consists of linear functions
- *g* consists of linear functions

In this case, *MPC* is an LP with nonnegative coefficients (feasibility version).

Linear Case

- $B \subseteq \mathbb{R}_+^N$ polytope
- *f* consists of linear functions
- *g* consists of linear functions

In this case, *MPC* is an LP with nonnegative coefficients (feasibility version).

Linear Case

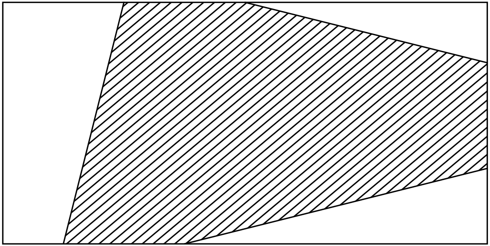
- $B \subseteq \mathbb{R}_+^N$ polytope
- f consists of linear functions
- g consists of linear functions

In this case, *MPC* is an LP with nonnegative coefficients (feasibility version).

Linear Case

- $B \subseteq \mathbb{R}_+^N$ polytope
- f consists of linear functions
- g consists of linear functions

In this case, *MPC* is an LP with nonnegative coefficients (feasibility version).



Motivation

LPs are well-studied; classical methods:

- Simplex Algorithm
- Ellipsoid Algorithm

Both aim at solving to optimality (or exact feasibility).
Drawbacks:

- "exact feasibility" limited by data structures
- paid for with excessive running time for massive instances
- input might be inexact

Motivation

LPs are well-studied; classical methods:

- Simplex Algorithm
- Ellipsoid Algorithm

Both aim at solving to optimality (or exact feasibility).

Drawbacks:

- "exact feasibility" limited by data structures
- paid for with excessive running time for massive instances
- input might be inexact

Motivation

LPs are well-studied; classical methods:

- Simplex Algorithm
- Ellipsoid Algorithm

Both aim at solving to optimality (or exact feasibility).
Drawbacks:

- "exact feasibility" limited by data structures
- paid for with excessive running time for massive instances
- input might be inexact

Motivation

LPs are well-studied; classical methods:

- Simplex Algorithm
- Ellipsoid Algorithm

Both aim at solving to optimality (or exact feasibility).
Drawbacks:

- "exact feasibility" limited by data structures
- paid for with excessive running time for massive instances
- input might be inexact

Motivation

LPs are well-studied; classical methods:

- Simplex Algorithm
- Ellipsoid Algorithm

Both aim at solving to optimality (or exact feasibility).

Drawbacks:

- "exact feasibility" limited by data structures
- paid for with excessive running time for massive instances
- input might be inexact

Motivation

LPs are well-studied; classical methods:

- Simplex Algorithm
- Ellipsoid Algorithm

Both aim at solving to optimality (or exact feasibility).
Drawbacks:

- “exact feasibility” limited by data structures
- paid for with excessive running time for massive instances
- input might be inexact

Motivation

LPs are well-studied; classical methods:

- Simplex Algorithm
- Ellipsoid Algorithm

Both aim at solving to optimality (or exact feasibility).
Drawbacks:

- “exact feasibility” limited by data structures
- paid for with excessive running time for massive instances
- input might be inexact

Motivation

LPs are well-studied; classical methods:

- Simplex Algorithm
- Ellipsoid Algorithm

Both aim at solving to optimality (or exact feasibility).
Drawbacks:

- “exact feasibility” limited by data structures
- paid for with excessive running time for massive instances
- input might be inexact

Alternative Approach

We drop the goal to solve exactly.
We like to approximate instead, within a better running time.

- $c \geq 1, \epsilon \in (0, 1)$

Restate the problem:

$$\text{find } x \in B \text{ such that } f(x) \leq c(1+\epsilon)a, \quad g(x) \geq (1-\epsilon)b/c \quad \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \qquad (MPC_{c,\epsilon})$$

Alternative Approach

We drop the goal to solve exactly.

We like to approximate instead, within a better running time.

- $c \geq 1, \epsilon \in (0, 1)$

Restate the problem:

$$\text{find } x \in B \text{ such that } f(x) \leq c(1+\epsilon)a, \quad g(x) \geq (1-\epsilon)b/c \quad \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \qquad (MPC_{c,\epsilon})$$

Alternative Approach

We drop the goal to solve exactly.
We like to approximate instead, within a better running time.

- $c \geq 1, \epsilon \in (0, 1)$

Restate the problem:

$$\begin{array}{c} \text{find } x \in B \text{ such that } f(x) \leq c(1+\epsilon)a, \quad g(x) \geq (1-\epsilon)b/c \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \end{array} \qquad (MPC_{c,\epsilon})$$

Alternative Approach

We drop the goal to solve exactly.
We like to approximate instead, within a better running time.

- $c \geq 1, \epsilon \in (0, 1)$

Restate the problem:

$$\text{find } x \in B \text{ such that } f(x) \leq c(1+\epsilon)a, \quad g(x) \geq (1-\epsilon)b/c \quad \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \qquad (MPC_{c,\epsilon})$$

Alternative Approach

We drop the goal to solve exactly.
We like to approximate instead, within a better running time.

- $c \geq 1, \epsilon \in (0, 1)$

Restate the problem:

$$\begin{array}{c} \text{find } x \in B \text{ such that } f(x) \leq c(1+\epsilon)a, \quad g(x) \geq (1-\epsilon)b/c \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \end{array} \qquad (MPC_{c,\epsilon})$$

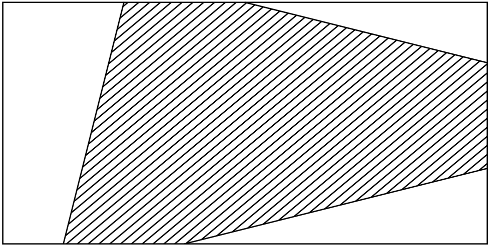
Alternative Approach

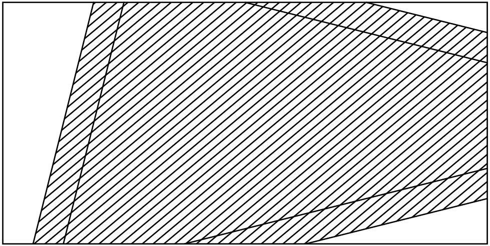
We drop the goal to solve exactly.
We like to approximate instead, within a better running time.

- $c \geq 1, \epsilon \in (0, 1)$

Restate the problem:

$$\begin{array}{c} \text{find } x \in B \text{ such that } f(x) \leq c(1+\epsilon)a, \quad g(x) \geq (1-\epsilon)b/c \\ \text{or decide that } \{x \in B | f(x) \leq a, g(x) \geq b\} = \emptyset \end{array} \qquad (MPC_{c,\epsilon})$$





Algorithm is based on the so-called Lagrangian decomposition.
Several key properties:

- iterative algorithms
- potentially faster
- potentially easier to implement
- potentially easier to parallelize
- generate only approximate solutions
- can handle models where N is exponential in a "compact formulation" of the instance (by column generation)

Algorithm is based on the so-called Lagrangian decomposition.

Several key properties:

- iterative algorithms
- potentially faster
- potentially easier to implement
- potentially easier to parallelize
- generate only approximate solutions
- can handle models where N is exponential in a "compact formulation" of the instance (by column generation)

Algorithm is based on the so-called Lagrangian decomposition.
Several key properties:

- iterative algorithms
- potentially faster
- potentially easier to implement
- potentially easier to parallelize
- generate only approximate solutions
- can handle models where N is exponential in a "compact formulation" of the instance (by column generation)

Algorithm is based on the so-called Lagrangian decomposition.
Several key properties:

- iterative algorithms
- potentially faster
- potentially easier to implement
- potentially easier to parallelize
- generate only approximate solutions
- can handle models where N is exponential in a "compact formulation" of the instance (by column generation)

Algorithm is based on the so-called Lagrangian decomposition.
Several key properties:

- iterative algorithms
- potentially faster
- potentially easier to implement
- potentially easier to parallelize
- generate only approximate solutions
- can handle models where N is exponential in a “compact formulation” of the instance (by column generation)

Algorithm is based on the so-called Lagrangian decomposition.
Several key properties:

- iterative algorithms
- potentially faster
- potentially easier to implement
- potentially easier to parallelize
- generate only approximate solutions
- can handle models where N is exponential in a "compact formulation" of the instance (by column generation)

Algorithm is based on the so-called Lagrangian decomposition.
Several key properties:

- iterative algorithms
- potentially faster
- potentially easier to implement
- potentially easier to parallelize
- generate only approximate solutions
- can handle models where N is exponential in a "compact formulation" of the instance (by column generation)

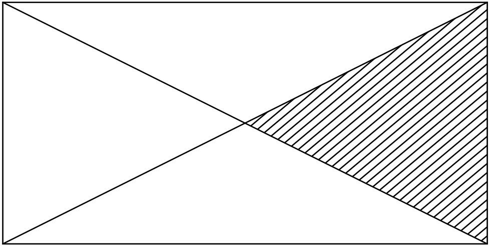
Algorithm is based on the so-called Lagrangian decomposition.
Several key properties:

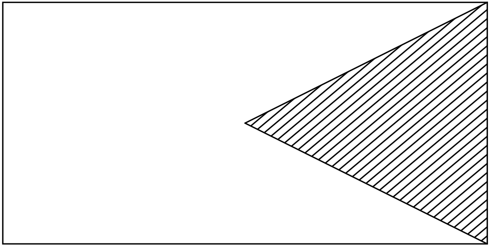
- iterative algorithms
- potentially faster
- potentially easier to implement
- potentially easier to parallelize
- generate only approximate solutions
- can handle models where N is exponential in a “compact formulation” of the instance (by column generation)

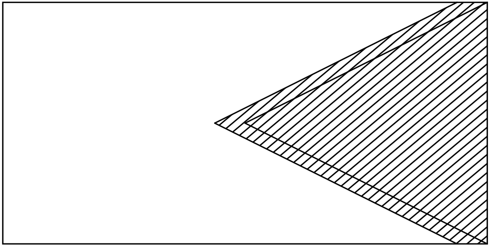
Sketch of Algorithm

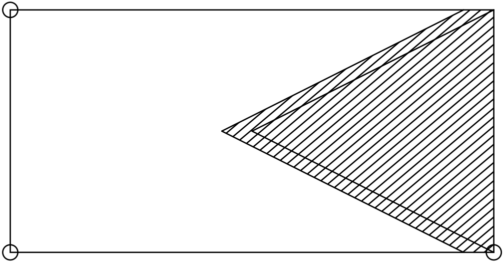
The algorithm can be sketched as follows.

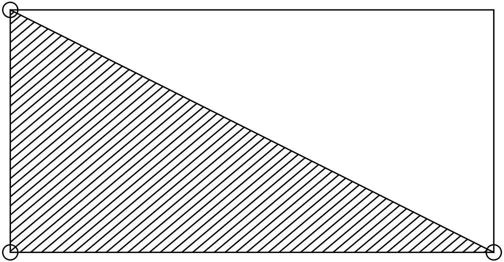
- compute an initial solution $x \in B$ via feasibility oracle
- as long as x is not “feasible enough”:
- find suitable $\hat{x} \in B$ via feasibility oracle
- set $x := (1 - \tau)x + \tau\hat{x}$ for a step length $\tau \in (0, 1)$
- assert that x becomes “more feasible”

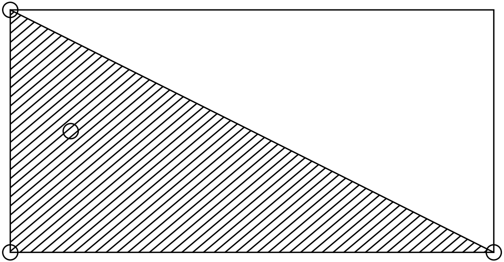


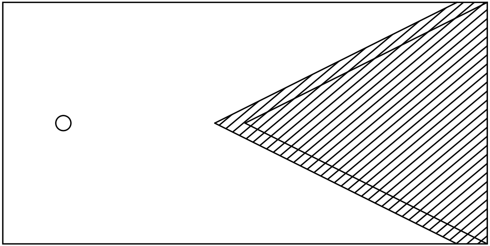


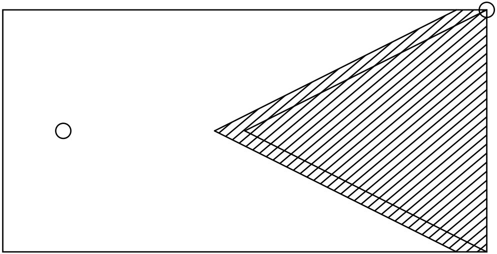


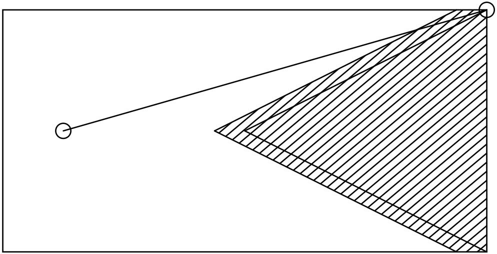


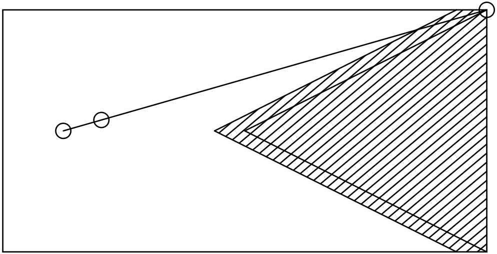


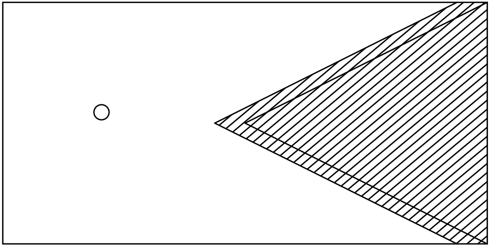


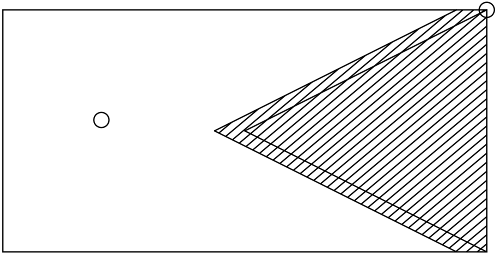


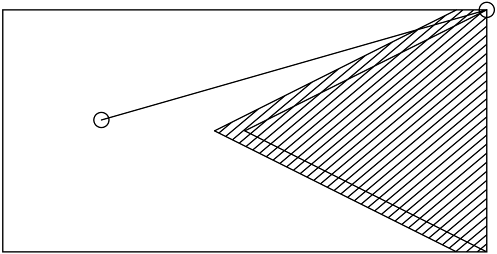


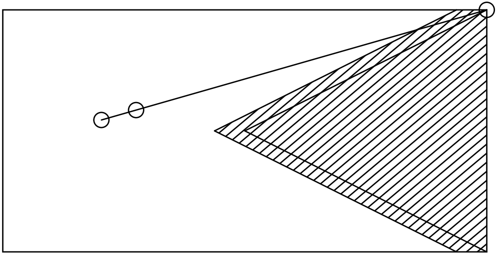


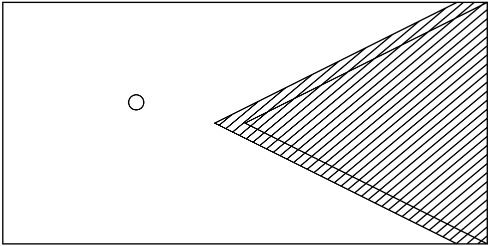


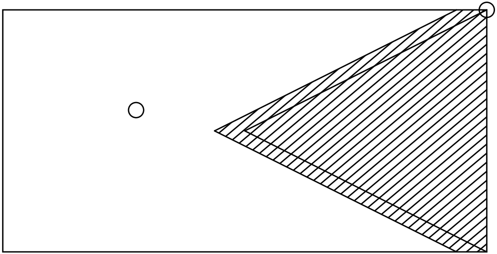


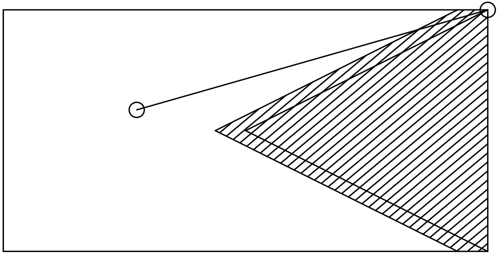


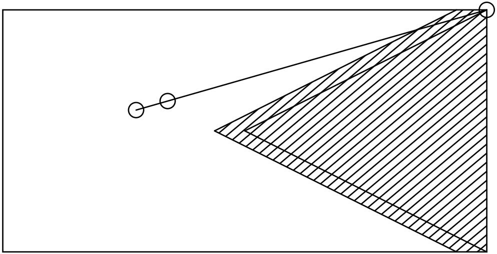


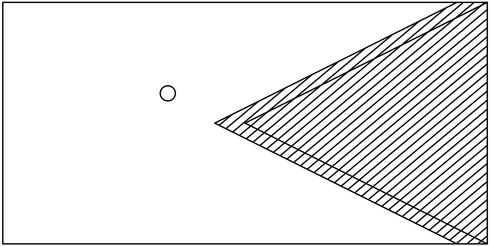


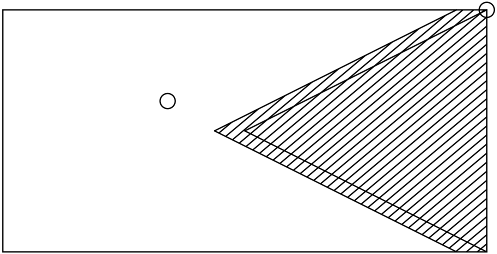


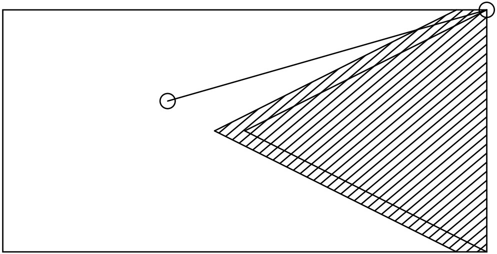


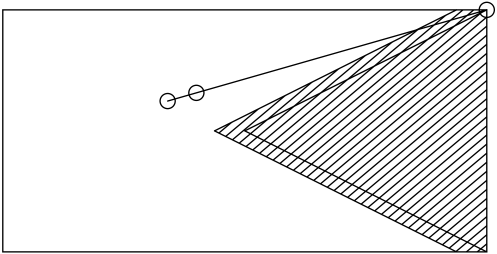


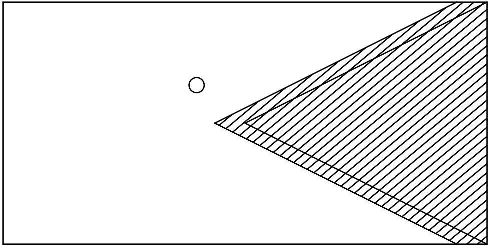


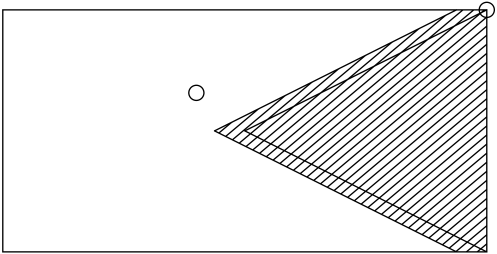


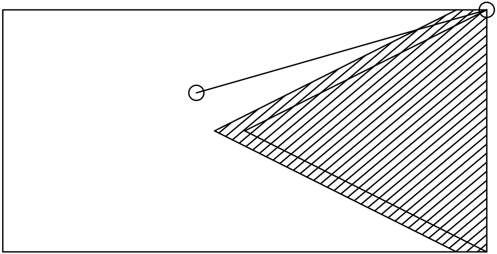


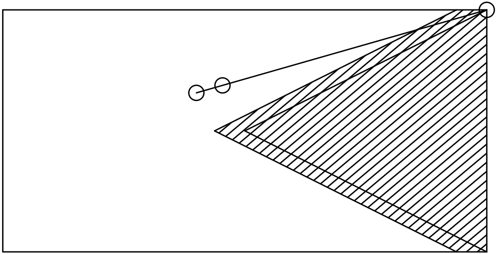


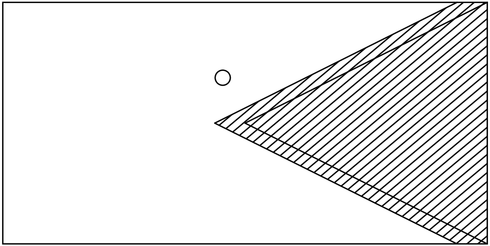


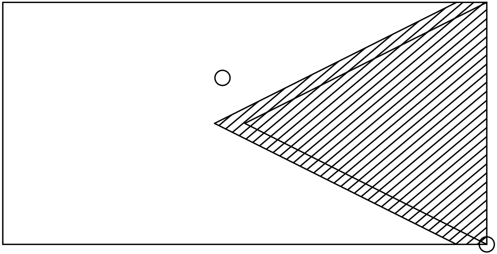


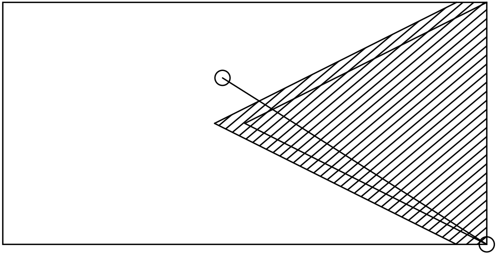


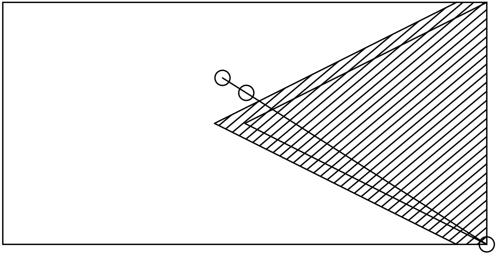


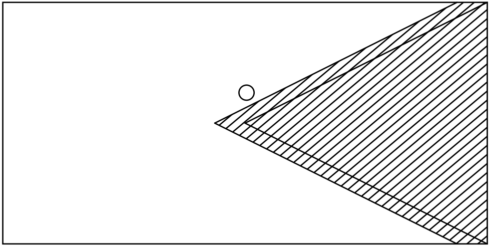


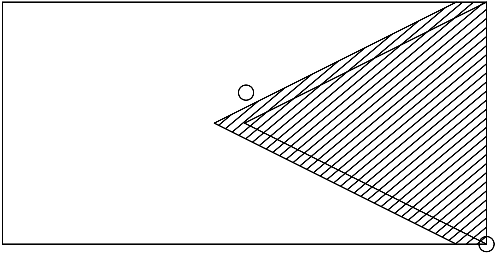


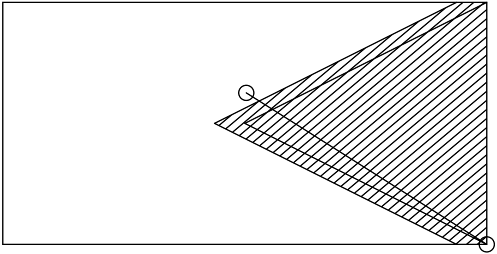


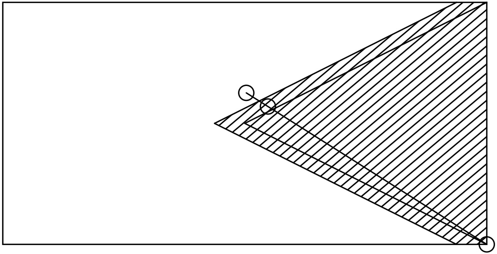


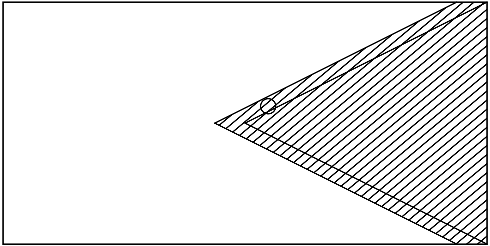












The Block Solver

The feasibility oracle is of the form

$$\text{find } \hat{x} \in B \text{ such that}$$

$$\frac{p^T f(\hat{x})}{c(1+t)(1+8/3t)} - q^T g(\hat{x}) c(1+t)(1+8/3t) \leq \alpha := 2e^T p - 1 - 2t$$

$$\text{or decide that there is no } x \in B \text{ with}$$

$$\frac{p^T f(\hat{x})}{(1+8/3t)} - q^T g(\hat{x})(1+8/3t) \leq \alpha$$

$$(ABS_c(p, q, \alpha, t))$$

where $p, q \in \mathbb{R}_+^M$ such that $\sum_{m=1}^M p_i + \sum_{i=1}^M q_i = 1$.

$ABS_c(p, q, \alpha, t)$ can be implemented by minimizing a convex function over B.

In the linear case it can be done by minimizing a linear function.

We aim at using fast combinatorial algorithms to implement $ABS_c(p, q, \alpha, t)$ for certain special cases of $(MPC_{c,\epsilon})$.

$ABS_c(p, q, \alpha, t)$ can be implemented by minimizing a convex function over B.

In the linear case it can be done by minimizing a linear function.

We aim at using fast combinatorial algorithms to implement $ABS_c(p, q, \alpha, t)$ for certain special cases of $(MPC_{c,\epsilon})$.

$ABS_c(p, q, \alpha, t)$ can be implemented by minimizing a convex function over B.

In the linear case it can be done by minimizing a linear function.

We aim at using fast combinatorial algorithms to implement $ABS_c(p, q, \alpha, t)$ for certain special cases of $(MPC_{c,\epsilon})$.

$ABS_c(p, q, \alpha, t)$ can be implemented by minimizing a convex function over B.

In the linear case it can be done by minimizing a linear function.

We aim at using fast combinatorial algorithms to implement $ABS_c(p, q, \alpha, t)$ for certain special cases of $(MPC_{c,\epsilon})$.

Theorem
The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

Theorem

The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2}\ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

Theorem

The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

Theorem

The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

Theorem

The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

Theorem

The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

Theorem

The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

Theorem

The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

Theorem

The algorithm solves $MPC_{c,\epsilon}$ in

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}))$$

iterations, where in each iteration $MPC_{c,\epsilon}$ is invoked once.

Some additional low-complexity coordination tasks in each iteration:

- evaluation of f, g
- interpolation in $\mathbb{R}^M$
- numerically finding a root of an equation
- comparison of vector entries
- administration of an index mask

However, the number of iterations is the primary measure of complexity.

More precisely, the algorithm aims at minimizing

$$\lambda_A : B \to \mathbb{R}_+ \cup \{\infty\}, \quad x \mapsto \max\{\max_{m\in[M]} f_m(x), \max_{m\in A} 1/g_m(x)\}$$

which "measures the infeasibility" of $x \in B$.
Here also the connection to the resource sharing algorithms is visible.

More precisely, the algorithm aims at minimizing

$$\lambda_A : B \to \mathbb{R}_+ \cup \{\infty\}, \quad x \mapsto \max\{\max_{m \in [M]} f_m(x), \max_{m \in A} 1/g_m(x)\}$$

which "measures the infeasibility" of $x \in B$.
Here also the connection to the resource sharing algorithms is visible.

More precisely, the algorithm aims at minimizing

$$\lambda_A : B \to \mathbb{R}_+ \cup \{\infty\}, \quad x \mapsto \max\{\max_{m \in [M]} f_m(x), \max_{m \in A} 1/g_m(x)\}$$

which "measures the infeasibility" of $x \in B$.
Here also the connection to the resource sharing algorithms is visible.

1. Setup some parameters; compute initial point $x^{(0)}$.
   If $\lambda(x^{(0)}) \leq c(1+\epsilon/2)$, go to Step 3.
2. Repeat Steps 2.1 – 2.3 {scaling phase s} until ϵ_s small enough or $\lambda(x^{(s)}) \leq c/(1-\epsilon)$.
   - 2.1. Set $\epsilon_s := \epsilon_{s-1}/2$, $x := x^{(s-1)}$, and T_s.
   - 2.2. Set $A := \{m \in [M] | g_m < T_s\}$.
   - 2.3. Repeat Steps 2.3.1 – 2.3.5 {coordination phase} forever.
     - 2.3.1. If $\lambda_A(x) \leq c/(1-\epsilon_s)$ go to Step 2.4.
     - 2.3.2. Compute θ, p and q, let $t_s := \epsilon_s/8$, $\alpha := 2\bar{p} - 1 - 2t_s$ and call $\hat{x} := ABS(p, q, \alpha, t_s)$.
     - 2.3.3. Compute suitable $\tau \in (0,1)$ and set $x' := (1-\tau)x + \tau\hat{x}$.
     - 2.3.4. If $\max\{(1-\tau)g_m + \tau\hat{g}_m | m \in A\} > T_s$ then reduce τ to τ' and set $x' := (1-\tau')x + \tau'\hat{x}$.
     - 2.3.5. Set $A := A \setminus \{m \in [M] | g_m(x') \geq T_s\}$ and $x := x'$.
   - 2.4. Set $x^{(s)} := x$. {end of scaling phase s}
3. Return the final iterate $x^{(s)} \in B$.

The analysis is based on a *logarithmic potential function* which also governs the choice of p, q and τ.
We use

$$\Phi_t(\theta, x, A) := 2\ln\theta - \frac{t}{CM}\Big[\sum_{m=1}^{M}\ln(\theta - f_m(x)) + \sum_{m\in A}\ln(g_m(x) - \frac{1}{\theta}) + (M - |A|)\ln T\Big]$$

where $C = 8$ is a constant.
It is based on two potential functions that have been used for the so-called *min-max* and *max-min* resource sharing problem.

The analysis is based on a *logarithmic potential function* which also governs the choice of p, q and τ.

We use

$$\Phi_t(\theta, x, A) := 2\ln\theta - \frac{t}{CM}\Big[\sum_{m=1}^{M}\ln(\theta - f_m(x)) + \sum_{m\in A}\ln(g_m(x) - \frac{1}{\theta}) + (M - |A|)\ln T\Big]$$

where $C = 8$ is a constant.

It is based on two potential functions that have been used for the so-called *min-max* and *max-min* resource sharing problem.

The analysis is based on a *logarithmic potential function* which also governs the choice of p, q and τ.
We use

$$\Phi_t(\theta, x, A) := 2 \ln \theta - \frac{t}{CM}[\sum_{m=1}^{M} \ln(\theta - f_m(x)) + \sum_{m \in A} \ln(g_m(x) - \frac{1}{\theta}) + (M - |A|) \ln T]$$

where $C = 8$ is a constant.
It is based on two potential functions that have been used for the so-called *min-max* and *max-min* resource sharing problem.

The analysis is based on a *logarithmic potential function* which also governs the choice of p, q and τ.
We use

$$\Phi_t(\theta, x, A) := 2\ln\theta - \frac{t}{CM}[\sum_{m=1}^{M}\ln(\theta - f_m(x)) + \sum_{m\in A}\ln(g_m(x) - \frac{1}{\theta}) + (M - |A|)\ln T]$$

where $C = 8$ is a constant.
It is based on two potential functions that have been used for the so-called *min-max* and *max-min* resource sharing problem.

For fixed A,x there is a uniquely determined $\theta \in \mathbb{R}_+$ that minimizes $\Phi_t(\theta, x, A)$.
This θ approximates $\lambda_A(x)$.
The corresponding minimum is denoted $\phi_t(x, A)$ and termed the *reduced potential* in x.
Key Ideas of the analysis:

- each iteration suitably decreases the reduced potential
- within a scaling phase, the possible difference between reduced potientials is bounded

For fixed A,x there is a uniquely determined $\theta \in \mathbb{R}_+$ that minimizes $\Phi_t(\theta, x, A)$.

This θ approximates $\lambda_A(x)$.

The corresponding minimum is denoted $\phi_t(x, A)$ and termed the *reduced potential* in x.

Key Ideas of the analysis:

- each iteration suitably decreases the reduced potential
- within a scaling phase, the possible difference between reduced potientials is bounded

For fixed A,x there is a uniquely determined $\theta \in \mathbb{R}_+$ that minimizes $\Phi_t(\theta, x, A)$.
This θ approximates $\lambda_A(x)$.
The corresponding minimum is denoted $\phi_t(x, A)$ and termed the *reduced potential* in x.
Key Ideas of the analysis:

- each iteration suitably decreases the reduced potential
- within a scaling phase, the possible difference between reduced potientials is bounded

For fixed A,x there is a uniquely determined $\theta \in \mathbb{R}_+$ that minimizes $\Phi_t(\theta, x, A)$.
This θ approximates $\lambda_A(x)$.
The corresponding minimum is denoted $\phi_t(x, A)$ and termed the *reduced potential* in x.
Key Ideas of the analysis:

- each iteration suitably decreases the reduced potential
- within a scaling phase, the possible difference between reduced potientials is bounded

For fixed A,x there is a uniquely determined $\theta \in \mathbb{R}_+$ that minimizes $\Phi_t(\theta, x, A)$.
This θ approximates $\lambda_A(x)$.
The corresponding minimum is denoted $\phi_t(x, A)$ and termed the *reduced potential* in x.
Key Ideas of the analysis:

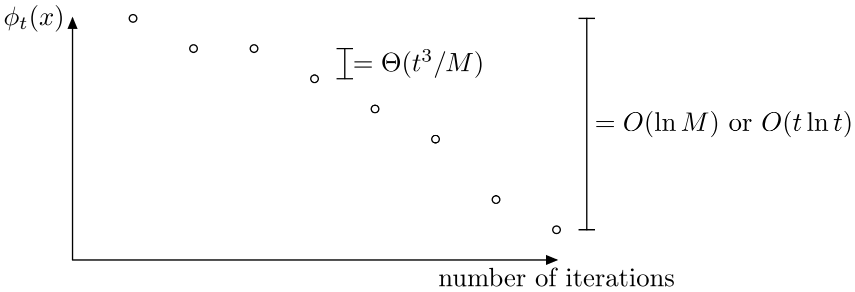
- each iteration suitably decreases the reduced potential
- within a scaling phase, the possible difference between reduced potientials is bounded

For fixed A,x there is a uniquely determined $\theta \in \mathbb{R}_+$ that minimizes $\Phi_t(\theta, x, A)$.
This θ approximates $\lambda_A(x)$.
The corresponding minimum is denoted $\phi_t(x, A)$ and termed the *reduced potential* in x.
Key Ideas of the analysis:

- each iteration suitably decreases the reduced potential
- within a scaling phase, the possible difference between reduced potientials is bounded

For fixed A,x there is a uniquely determined $\theta \in \mathbb{R}_+$ that minimizes $\Phi_t(\theta, x, A)$.
This θ approximates $\lambda_A(x)$.
The corresponding minimum is denoted $\phi_t(x, A)$ and termed the *reduced potential* in x.
Key Ideas of the analysis:

- each iteration suitably decreases the reduced potential
- within a scaling phase, the possible difference between reduced potientials is bounded


$\phi_t(x)$
$= \Theta(t^3/M)$
$= O(\ln M)$ or $O(t \ln t)$
number of iterations

Application: Fractional Multicommodity Flow

Given:

- directed graph $G = (V, E)$
- demands $d_i \in \mathbb{R}_{++}$ from s_i to t_i for each $i \in [k]$
- capacities c_e for each edge $e \in E$
- P_i set of all s_i-t_i-paths
- costs $w(p) \in \mathbb{R}_+$ for each $p \in \cup P_i$
- budget $W \in \mathbb{R}_+$

Application: Fractional Multicommodity Flow

Given:

- directed graph $G = (V, E)$
- demands $d_i \in \mathbb{R}_{++}$ from s_i to t_i for each $i \in [k]$
- capacities c_e for each edge $e \in E$
- P_i set of all s_i-t_i-paths
- costs $w(p) \in \mathbb{R}_+$ for each $p \in \cup P_i$
- budget $W \in \mathbb{R}_+$

Application: Fractional Multicommodity Flow

Given:

- directed graph $G = (V, E)$
- demands $d_i \in \mathbb{R}_{++}$ from s_i to t_i for each $i \in [k]$
- capacities c_e for each edge $e \in E$
- P_i set of all s_i-t_i-paths
- costs $w(p) \in \mathbb{R}_+$ for each $p \in \cup P_i$
- budget $W \in \mathbb{R}_+$

Application: Fractional Multicommodity Flow

Given:

- directed graph $G = (V, E)$
- demands $d_i \in \mathbb{R}_{++}$ from s_i to t_i for each $i \in [k]$
- capacities c_e for each edge $e \in E$
- P_i set of all s_i-t_i-paths
- costs $w(p) \in \mathbb{R}_+$ for each $p \in \cup P_i$
- budget $W \in \mathbb{R}_+$

Application: Fractional Multicommodity Flow

Given:

- directed graph $G = (V, E)$
- demands $d_i \in \mathbb{R}_{++}$ from s_i to t_i for each $i \in [k]$
- capacities c_e for each edge $e \in E$
- P_i set of all s_i-t_i-paths
- costs $w(p) \in \mathbb{R}_+$ for each $p \in \cup P_i$
- budget $W \in \mathbb{R}_+$

Application: Fractional Multicommodity Flow

Given:

- directed graph $G = (V, E)$
- demands $d_i \in \mathbb{R}_{++}$ from s_i to t_i for each $i \in [k]$
- capacities c_e for each edge $e \in E$
- P_i set of all s_i-t_i-paths
- costs $w(p) \in \mathbb{R}_+$ for each $p \in \cup P_i$
- budget $W \in \mathbb{R}_+$

Fractional Multicommodity Flow LP

Use a variable x_p for each $p \in \cup P_i$.

$$\begin{aligned}
\sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p &= W \\
\sum_{p \in P_i} x_p &\geq d_i \text{ for each } i \in [k] \\
\sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p &\leq c_e \text{ for each } e \in E \\
x_p &\geq 0 \text{ for each } p \in \cup P_i
\end{aligned}$$

Note that the flow conservation is not explicitly modelled.

Fractional Multicommodity Flow LP

Use a variable x_p for each $p \in \cup P_i$.

$$\begin{array}{rcl}
\sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p & = & W \\
\sum_{p \in P_i} x_p & \geq & d_i \text{ for each } i \in [k] \\
\sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p & \leq & c_e \text{ for each } e \in E \\
x_p & \geq & 0 \text{ for each } p \in \cup P_i
\end{array}$$

Note that the flow conservation is not explicitly modelled.

Fractional Multicommodity Flow LP

Use a variable x_p for each $p \in \cup P_i$.

$$\begin{array}{rcl} \sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p & = & W \\ \sum_{p \in P_i} x_p & \geq & d_i \text{ for each } i \in [k] \\ \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p & \leq & c_e \text{ for each } e \in E \\ x_p & \geq & 0 \text{ for each } p \in \cup P_i \end{array}$$

Note that the flow conservation is not explicitly modelled.

Formulation for Mixed Problem

We set

- $f_e(x) := \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p / c_e \leq 1$ for each $e \in E$
- $g_i(x) := \sum_{p \in P_i} x_p / d_i \geq 1$ for each $i \in [k]$

and furthermore

- $B := \{x_p | p \in \cup P_i, x_p \geq 0, \sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p = W\}$

Here B is a standard simplex (distorted by w) over which it is “easy” to optimize a linear objective since the vertices are easily found.

Formulation for Mixed Problem

We set

- $f_e(x) := \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p / c_e \leq 1$ for each $e \in E$
- $g_i(x) := \sum_{p \in P_i} x_p / d_i \geq 1$ for each $i \in [k]$

and furthermore

- $B := \{x_p | p \in \cup P_i, x_p \geq 0, \sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p = W\}$

Here B is a standard simplex (distorted by w) over which it is “easy” to optimize a linear objective since the vertices are easily found.

Formulation for Mixed Problem

We set

- $f_e(x) := \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p / c_e \leq 1$ for each $e \in E$
- $g_i(x) := \sum_{p \in P_i} x_p / d_i \geq 1$ for each $i \in [k]$

and furthermore

- $B := \{x_p | p \in \cup P_i, x_p \geq 0, \sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p = W\}$

Here B is a standard simplex (distorted by w) over which it is “easy” to optimize a linear objective since the vertices are easily found.

Formulation for Mixed Problem

We set

- $f_e(x) := \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p / c_e \leq 1$ for each $e \in E$
- $g_i(x) := \sum_{p \in P_i} x_p / d_i \geq 1$ for each $i \in [k]$

and furthermore

- $B := \{x_p | p \in \cup P_i, x_p \geq 0, \sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p = W\}$

Here B is a standard simplex (distorted by w) over which it is “easy” to optimize a linear objective since the vertices are easily found.

Formulation for Mixed Problem

We set

- $f_e(x) := \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p / c_e \leq 1$ for each $e \in E$
- $g_i(x) := \sum_{p \in P_i} x_p / d_i \geq 1$ for each $i \in [k]$

and furthermore

- $B := \{x_p | p \in \cup P_i, x_p \geq 0, \sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p = W\}$

Here B is a standard simplex (distorted by w) over which it is "easy" to optimize a linear objective since the vertices are easily found.

Formulation for Mixed Problem

We set

- $f_e(x) := \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p / c_e \leq 1$ for each $e \in E$
- $g_i(x) := \sum_{p \in P_i} x_p / d_i \geq 1$ for each $i \in [k]$

and furthermore

- $B := \{x_p | p \in \cup P_i, x_p \geq 0, \sum_{i=1}^{k} \sum_{p \in P_i} w(p) x_p = W\}$

Here B is a standard simplex (distorted by w) over which it is “easy” to optimize a linear objective since the vertices are easily found.

The Blocksolver

The resulting block solver is

$$\min p^T f(\hat{x})/Y(c,t) - q^T g(\hat{x}) Y(c,t) = \sum_{i=1}^{k} \sum_{p \in P_i} \Big(\sum_{e \in P} \frac{p_e}{c_e Y(c,t)} - \frac{q_i}{d_i} Y(c,t) \Big) x_p$$

where $Y(c,t) = c(1+t)(1+8/3t)$ is the parameter from the beginning. Let $\ell(p) = \sum_{e \in p} \frac{p_e}{c_e Y(c,t)}$ be the length of path p w.r.t. edge weights

$$\frac{p_e}{c_e Y(c,t)}.$$

The Blocksolver

The resulting block solver is

$$\min p^T f(\hat{x})/Y(c,t) - q^T g(\hat{x})Y(c,t) = \sum_{i=1}^{k}\sum_{p\in P_i}(\sum_{e\in P}\frac{p_e}{c_e Y(c,t)} - \frac{q_i}{d_i}Y(c,t))x_p$$

where $Y(c,t) = c(1+t)(1+8/3t)$ is the parameter from the beginning.
Let $\ell(p) = \sum_{e\in p}\frac{p_e}{c_e Y(c,t)}$ be the length of path p w.r.t. edge weights

$$\frac{p_e}{c_e Y(c,t)}.$$

The Blocksolver

The resulting block solver is

$$\min p^T f(\hat{x})/Y(c,t) - q^T g(\hat{x}) Y(c,t) = \sum_{i=1}^{k} \sum_{p \in P_i} \Big(\sum_{e \in P} \frac{p_e}{c_e Y(c,t)} - \frac{q_i}{d_i} Y(c,t) \Big) x_p$$

where $Y(c,t) = c(1+t)(1+8/3t)$ is the parameter from the beginning. Let $\ell(p) = \sum_{e \in p} \frac{p_e}{c_e Y(c,t)}$ be the length of path p w.r.t. edge weights

$$\frac{p_e}{c_e Y(c,t)}.$$

Since B is a distorted standard simplex, the optimum is attained for the incidence variable of exactly one path $p \in \cup P_i$.

Hence we can enumerate the k commodities and solve a shortest path problem to minimize $\ell(p)$ for $p \in P_i$.

Our approach decomposes Fractional Multicommodity Flow to a sequence of shortest path problems.

Overall running time is

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}) \cdot M^2) = O(M^3 \ln M + M^3 \epsilon^{-2} \ln \epsilon^{-1}).$$

Since B is a distorted standard simplex, the optimum is attained for the incidence variable of exactly one path $p \in \cup P_i$.

Hence we can enumerate the k commodities and solve a shortest path problem to minimize $\ell(p)$ for $p \in P_i$.

Our approach decomposes Fractional Multicommodity Flow to a sequence of shortest path problems.

Overall running time is

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}) \cdot M^2) = O(M^3 \ln M + M^3 \epsilon^{-2} \ln \epsilon^{-1}).$$

Since B is a distorted standard simplex, the optimum is attained for the incidence variable of exactly one path $p \in \cup P_i$.

Hence we can enumerate the k commodities and solve a shortest path problem to minimize $\ell(p)$ for $p \in P_i$.

Our approach decomposes Fractional Multicommodity Flow to a sequence of shortest path problems.

Overall running time is

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}) \cdot M^2) = O(M^3 \ln M + M^3 \epsilon^{-2} \ln \epsilon^{-1}).$$

Since B is a distorted standard simplex, the optimum is attained for the incidence variable of exactly one path $p \in \cup P_i$.
Hence we can enumerate the k commodities and solve a shortest path problem to minimize $\ell(p)$ for $p \in P_i$.
Our approach decomposes Fractional Multicommodity Flow to a sequence of shortest path problems.
Overall running time is

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}) \cdot M^2) = O(M^3 \ln M + M^3 \epsilon^{-2} \ln \epsilon^{-1}).$$

Since B is a distorted standard simplex, the optimum is attained for the incidence variable of exactly one path $p \in \cup P_i$.
Hence we can enumerate the k commodities and solve a shortest path problem to minimize $\ell(p)$ for $p \in P_i$.
Our approach decomposes Fractional Multicommodity Flow to a sequence of shortest path problems.
Overall running time is

$$O(M(\ln M + \epsilon^{-2} \ln \epsilon^{-1}) \cdot M^2) = O(M^3 \ln M + M^3 \epsilon^{-2} \ln \epsilon^{-1}).$$

Different Model

We study the following optimization variant ("throughput maximization")

$$\begin{aligned} \max \theta & \\ \text{s.t.} & \\ \sum_{p \in P_i} x_p &= \theta d_i \text{ for each } i \in [k] \\ \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p &\leq c_e \text{ for each } e \in E \\ x_p &\geq 0 \text{ for each } p \in \cup P_i \end{aligned}$$

Can be solved with a smiliar technique by Jansen & Zhang [IFIP TCS 2004] where also in each iteration M shortest path computations are necessary. Skip the details here.

Different Model

We study the following optimization variant ("throughput maximization")

$$
\begin{aligned}
&\max \theta \\
&\text{s.t.} \\
\sum_{p \in P_i} x_p &= \theta d_i \text{ for each } i \in [k] \\
\sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p &\leq c_e \text{ for each } e \in E \\
x_p &\geq 0 \text{ for each } p \in \cup P_i
\end{aligned}
$$

Can be solved with a smiliar technique by Jansen & Zhang [IFIP TCS 2004] where also in each iteration M shortest path computations are necessary. Skip the details here.

Different Model

We study the following optimization variant (“throughput maximization”)

$$\begin{aligned} \max \theta & \\ \text{s.t.} & \\ \sum_{p \in P_i} x_p &= \theta d_i \text{ for each } i \in [k] \\ \sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p &\leq c_e \text{ for each } e \in E \\ x_p &\geq 0 \text{ for each } p \in \cup P_i \end{aligned}$$

Can be solved with a smiliar technique by Jansen & Zhang [IFIP TCS 2004] where also in each iteration M shortest path computations are necessary. Skip the details here.

Different Model

We study the following optimization variant ("throughput maximization")

$$
\begin{aligned}
\max \theta & \\
\text{s.t.} & \\
\sum_{p\in P_i} x_p &= \theta d_i \text{ for each } i \in [k] \\
\sum_{i=1}^{k} \sum_{e\in p\in P_i} x_p &\leq c_e \text{ for each } e \in E \\
x_p &\geq 0 \text{ for each } p \in \cup P_i
\end{aligned}
$$

Can be solved with a smiliar technique by Jansen & Zhang [IFIP TCS 2004] where also in each iteration M shortest path computations are necessary. Skip the details here.

Different Model

We study the following optimization variant ("throughput maximization")

$$
\begin{array}{rcl}
\max \theta & & \\
\text{s.t.} & & \\
\sum_{p \in P_i} x_p & = & \theta d_i \text{ for each } i \in [k] \\
\sum_{i=1}^{k} \sum_{e \in p \in P_i} x_p & \leq & c_e \text{ for each } e \in E \\
x_p & \geq & 0 \text{ for each } p \in \cup P_i
\end{array}
$$

Can be solved with a smiliar technique by Jansen & Zhang [IFIP TCS 2004] where also in each iteration M shortest path computations are necessary. Skip the details here.

In total, large-scale mixed packing and covering problems can be solved efficiently (in theory).
So far, no experimental study of this algorithm.

In total, large-scale mixed packing and covering problems can be solved efficiently (in theory).

So far, no experimental study of this algorithm.

In total, large-scale mixed packing and covering problems can be solved efficiently (in theory).
So far, no experimental study of this algorithm.

Open Problems

- Possible to minimize budget for the mixed model?
- Possible to reduce the running times?
- Experimental comparision with algorithms by
  - Fleischer [Soda 2004]
  - Young [FOCS 2001]
  - Garg & Könemann [FOCS 1998]

Open Problems

- Possible to minimize budget for the mixed model?
- Possible to reduce the running times?
- Experimental comparision with algorithms by
  - Fleischer [Soda 2004]
  - Young [FOCS 2001]
  - Garg & Könemann [FOCS 1998]

Open Problems

- Possible to minimize budget for the mixed model?
- Possible to reduce the running times?
- Experimental comparision with algorithms by
  - Fleischer [Soda 2004]
  - Young [FOCS 2001]
  - Garg & Könemann [FOCS 1998]

Open Problems

- Possible to minimize budget for the mixed model?
- Possible to reduce the running times?
- Experimental comparision with algorithms by
  - Fleischer [Soda 2004]
  - Young [FOCS 2001]
  - Garg & Könemann [FOCS 1998]

Open Problems

- Possible to minimize budget for the mixed model?
- Possible to reduce the running times?
- Experimental comparision with algorithms by
  - Fleischer [Soda 2004]
  - Young [FOCS 2001]
  - Garg & Könemann [FOCS 1998]

Open Problems

- Possible to minimize budget for the mixed model?
- Possible to reduce the running times?
- Experimental comparision with algorithms by
  - Fleischer [Soda 2004]
  - Young [FOCS 2001]
  - Garg & Könemann [FOCS 1998]

End

Thanks for your attention!