Resource Sharing

Jens Vygen

Hangzhou, April 2009
Min-max resource sharing

Instance

- finite sets $\mathcal{R}$ of resources and $\mathcal{C}$ of customers
- for each $c \in \mathcal{C}$:
  - a convex set $\mathcal{B}_c$ of feasible solutions (a block) and
  - a convex resource consumption function $g_c : \mathcal{B}_c \to \mathbb{R}_+^\mathcal{R}$
- given by an oracle function $f_c : \mathbb{R}_+^\mathcal{R} \to \mathcal{B}_c$ with

$$\omega^T g_c(f_c(\omega)) \leq (1 + \epsilon_0) \inf_{b \in \mathcal{B}_c} \omega^T g_c(b)$$

for all $\omega \in \mathbb{R}_+^\mathcal{R}$ and some $\epsilon_0 \in \mathbb{R}_+$ (a block solver).

Task

- Find a $b_c \in \mathcal{B}_c$ for each $c \in \mathcal{C}$ with minimum congestion

$$\max_{r \in \mathcal{R}} \sum_{c \in \mathcal{C}} (g_c(b_c))_r .$$
Block solvers

A block solver is an oracle function $f_c : \mathbb{R}^R_+ \rightarrow B_c$ with

$$\omega^\top g_c(f_c(\omega)) \leq (1 + \epsilon_0) \text{opt}_c(\omega)$$

for all $\omega \in \mathbb{R}^R_+$ and some $\epsilon_0 \in \mathbb{R}_+$, where

$$\text{opt}_c(\omega) := \inf_{b \in B_c} \omega^\top g_c(b)$$

The block solver is called

- **strong** if $\epsilon_0 = 0$ or $\epsilon_0 > 0$ can be chosen arbitrary small
- **weak** otherwise

The block solver is called

- **bounded** if it can also optimize over

$$\{b \in B_c : g_c(b) \leq \mu 1\}$$

for any given $\mu > 0$ ($c \in C$).
- **unbounded** otherwise
Width

Let

$$\lambda^* := \inf \left\{ \max_{r \in \mathcal{R}} \sum_{c \in C} (g_c(b_c))_r : b_c \in \mathcal{B}_c(c \in C) \right\}$$

(the “optimum congestion”), and

$$\rho := \max \left\{ 1, \sup \left\{ \frac{(g_c(b))_r}{\lambda^*} : r \in \mathcal{R}, c \in C, b \in \mathcal{B}_c \right\} \right\}$$

(the supremum is sometimes called the “width” of the problem)

In case of a bounded block solver, and in most applications, we may assume $\rho = 1$ (“no bottleneck”).
## Summary of results

<table>
<thead>
<tr>
<th>min-max resource sharing</th>
<th>block solver</th>
<th>running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grigoriadis, Khachiyan [1994]</td>
<td>strong, bounded</td>
<td>$\tilde{O}(\epsilon^{-2}</td>
</tr>
<tr>
<td>Grigoriadis, Khachiyan [1996]</td>
<td>strong, unbounded</td>
<td>$\tilde{O}(\epsilon^{-2}</td>
</tr>
<tr>
<td>Jansen, Zhang [2008]</td>
<td>weak, unbounded</td>
<td>$\tilde{O}(\epsilon^{-2}</td>
</tr>
<tr>
<td>Müller, V. [2008]</td>
<td>weak, unbounded</td>
<td>$\tilde{O}(\epsilon^{-2}\rho</td>
</tr>
<tr>
<td>Müller, V. [2008]</td>
<td>weak, bounded</td>
<td>$\tilde{O}(\epsilon^{-2}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>fractional packing (all $g_c$ linear)</th>
<th>block solver</th>
<th>running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plotkin, Shmoys, Tardos [1995] *</td>
<td>strong, unbounded</td>
<td>$\tilde{O}(\epsilon^{-2}\rho</td>
</tr>
<tr>
<td>Young [1995]</td>
<td>weak, unbounded</td>
<td>$\tilde{O}(\epsilon^{-2}\rho</td>
</tr>
<tr>
<td>Charikar et al. [1998] *</td>
<td>weak, unbounded</td>
<td>$\tilde{O}(\epsilon^{-2}\rho</td>
</tr>
<tr>
<td>Bienstock, Iyengar [2004]</td>
<td>—</td>
<td>$\tilde{O}(\epsilon^{-1} \ldots)$</td>
</tr>
</tbody>
</table>

Algorithms compute a $(1 + \epsilon_0 + \epsilon)$-approximate solution.
Running times for fixed $\epsilon_0 \geq 0$. Logarithmic terms omitted.
Entries with * refer to the feasibility version ($\lambda^* = 1$).
Weak duality

Lemma (Weak duality)
Let $\omega \in \mathbb{R}_+^R$ be some cost vector with $\omega^\top 1 \neq 0$. Then

$$\sum_{c \in C} \frac{\text{opt}_c(\omega)}{\omega^\top 1} \leq \lambda^*.$$ 

Proof
Let $(b_c \in B_c)_{c \in C}$ be a solution with congestion $\lambda^*$. Then

$$\sum_{c \in C} \frac{\text{opt}_c(\omega)}{\omega^\top 1} \leq \frac{\sum_{c \in C} \omega^\top g_c(b_c)}{\omega^\top 1} = \frac{\omega^\top \sum_{c \in C} g_c(b_c)}{\omega^\top 1} \leq \frac{\omega^\top \lambda^* 1}{\omega^\top 1} = \lambda^*$$

$\square$
Bounding $\lambda^*$

Lemma (Weak duality)

Let $\omega \in \mathbb{R}^R_+$ be some cost vector with $\omega^\top 1 \neq 0$. Then

$$\frac{\sum_{c \in C} \text{opt}_c(\omega)}{\omega^\top 1} \leq \lambda^*.$$ 

Corollary

Let $b_c := f_c(1)$ ($c \in C$) and $\lambda^{ub} := \max_{r \in R} \sum_{c \in C} (g_c(b_c))_r$. Then

$$\frac{\lambda^{ub}}{|R|(1 + \epsilon_0)} \leq \frac{\sum_{c \in C} \text{opt}_c(1)}{|R|(1 + \epsilon_0)} \leq \lambda^* \leq \lambda^{ub}. \quad \square$$
Scaling and binary search

We know \( \frac{\lambda^{ub}}{|R|(1+\epsilon_0)} \leq \lambda^* \leq \lambda^{ub} \).

1. Set \( j := 0 \).
2. Scale \( g_c^{(j)}(b) := g_c(b) \frac{2^j}{\lambda^{ub}} \). Note that \( \lambda^*(j) \leq 1 \).
3. Find a solution with congestion \( \lambda(j) \leq (1 + \epsilon_0 + \frac{1}{4})\lambda^*(j) + \frac{1}{4} \).
4. If \( \lambda(j) \leq \frac{1}{2} \), then increment \( j \) and go to 2.
5. Now \( \frac{1}{5(1+\epsilon_0)} \leq \lambda^*(j) \leq 1 \).
6. Find a solution with congestion \( \lambda(j) \leq (1 + \epsilon_0 + \frac{\epsilon}{6})\lambda^*(j) + \frac{\epsilon}{6(1+\epsilon)} \).

Lemma (Main Lemma)

Let \( \delta, \delta' > 0 \). Suppose that \( \lambda^* \leq 1 \).
Then we can compute a solution with congestion at most

\[(1 + \epsilon_0 + \delta)\lambda^* + \delta'\]

in

\( O \left( (\delta\delta')^{-1}|C|\theta \rho (1 + \epsilon_0)^2 \log |R| \right) \)

time, where \( \theta \) is the time for an oracle call.
Core algorithm

**Input:** An instance of the min-max resource sharing problem.
**Output:** A convex combination of vectors in $B_c$ for each $c \in C$.

Set $t := \left\lceil \frac{4\rho (1 + \epsilon_0)^2 \ln |R|}{\delta' \min\{1, \delta\}} \right\rceil$.

Set $\alpha_r := 0$ and $\omega_r := 1$ for each $r \in R$.
Set $x_{c,b} := 0$ for each $c \in C$ and $b \in B_c$.

For $p := 1$ to $t$ do: \hspace{1cm} (perform $t$ phases)

For each $c \in C$ do:

AllocateResources($c$).

Set $x_{c,b} := \frac{1}{t} x_{c,b}$ for each $c \in C$ and $b \in B_c$. \hspace{1cm} (normalize)
Core algorithm: subroutine

Set $\epsilon_2 := \frac{\min\{1, \delta\}}{4\rho(1+\epsilon_0)^2}$.

Procedure AllocateResources($c$):

Set $b_c := f_c(\omega)$. \hspace{1cm} (call oracle)
Set $x_{c,b_c} := x_{c,b_c} + 1$.
Set $\alpha := \alpha + g_c(b_c)$. \hspace{1cm} (update resource consumption)
For each $r \in R$ with $(g_c(b_c))_r \neq 0$ do:
Set $\omega_r := e^{\epsilon_2 \alpha_r}$. \hspace{1cm} (update prices)
Proof of performance guarantee (sketch)

Lemma

Let \((x, \omega)\) be the output of the algorithm, and let

\[
\lambda_r := \sum_{c \in C} \left( g_c \left( \sum_{b \in B_c} x_{c,b} b \right) \right)_r
\]

and \(\lambda := \max_{r \in \mathcal{R}} \lambda_r\). Then

\[
\lambda \leq \frac{1}{\epsilon^2 t} \ln \sum_{r \in \mathcal{R}} e^{\epsilon^2 t \lambda_r} = \frac{1}{\epsilon^2 t} \ln (\omega^\top 1).
\]

Proof: Since the functions \(g_c\) are convex, we have for \(r \in \mathcal{R}\):

\[
\lambda_r \leq \sum_{c \in C} \sum_{b \in B_c} x_{c,b}(g_c(b))_r = \frac{\alpha_r}{t} = \frac{1}{\epsilon^2 t} \ln (e^{\epsilon^2 \alpha_r}) = \frac{1}{\epsilon^2 t} \ln \omega_r
\]
Proof of performance guarantee (sketch)

Lemma (Main Lemma)

Let $\delta, \delta' > 0$. Suppose that $\lambda^* \leq 1$.
Then the algorithm computes a solution with congestion at most

$$(1 + \epsilon_0 + \delta)\lambda^* + \delta'.$$

Sketch of proof:

- Congestion is at most $\frac{1}{\epsilon t} \ln((\omega^{(t)})^\top 1)$.
- Initially, we have $(\omega^{(0)})^\top 1 = |R|$.
- Short calculation yields

$$((\omega^{(p)})^\top 1 \leq (\omega^{(p-1)})^\top 1 + \epsilon' \sum_{c \in C} \text{opt}_c(\omega^{(p)}),$$

where $\omega^{(i)}$ is the price vector at the end of the $i$-th phase and $\epsilon' := \epsilon_2(1 + (e - 2)\rho \epsilon_2)(1 + \epsilon_0)$. 


Proof of performance guarantee (sketch)

We had \((\omega^{(p)})^\top \mathbf{1} \leq (\omega^{(p-1)})^\top \mathbf{1} + \epsilon' \sum_{c \in C} \text{opt}_c(\omega^{(p)})\).

By weak duality, \(\epsilon' \sum_{c \in C} \frac{\text{opt}_c(\omega^{(p)})}{(\omega(p))^\top \mathbf{1}} \leq \epsilon' \lambda^* < 1\), and we get

\[ (\omega^{(p)})^\top \mathbf{1} \leq \frac{1}{1 - \epsilon' \lambda^*} (\omega^{(p-1)})^\top \mathbf{1} \]

and thus

\[ (\omega^{(t)})^\top \mathbf{1} \leq \frac{|\mathcal{R}|}{(1 - \epsilon' \lambda^*)^t} = |\mathcal{R}| \left(1 + \frac{\epsilon' \lambda^*}{1 - \epsilon' \lambda^*}\right)^t \leq |\mathcal{R}| e^{t \epsilon' \lambda^*/(1 - \epsilon' \lambda^*)} \].

Together with \(\lambda \leq \frac{1}{\epsilon_2 t} \ln\left((\omega^{(t)})^\top \mathbf{1}\right)\), this proves the claim. \(\square\)
Main result

Theorem

*The presented algorithm computes a $(1 + \epsilon_0 + \epsilon)$-approximate solution in $O(\lvert C \rvert \theta \rho (1 + \epsilon_0)^2 \log \lvert R \rvert (\log \lvert R \rvert + \epsilon^{-2}(1 + \epsilon_0)))$ time, where $\theta$ is the time for an oracle call.*

(Müller, V. [2008])

Extensions for practical application:

- Most oracle calls not necessary; reuse previous result if still good enough. Use lower bounds to decide
- Speed-up heuristics
- Randomized rounding to extreme points of the blocks
- Re-choose where rounding violates constraints
Application to global routing

Given a global routing graph (3D grid with millions of vertices).

- **Customers** = nets (sets of pins; roughly: sets of vertices)
- **Resources** = edge capacities, power consumption, yield loss, timing constraints, ...
- Objective function is transformed into a constraint
- **Block** = (convex hull of) set of Steiner trees for a net, with space consumption for each edge
- Resource consumption is nonlinear (but convex) for yield loss, timing, power consumption
- **Block solver** = approximation algorithm for the Steiner tree problem in the global routing graph (with edge weights)
The algorithm in practice

- In practice, results are much better than theoretical performance guarantees. Usually 10–20 iterations suffice.
- Only few upper bounds are violated; these are corrected easily by *rip-up and re-route*.
- Detailed routing can realize the solution well, due to excellent capacity estimations.
- Small integrality gap and approximate dual solution implies that an infeasibility proof can be found for most infeasible instances.
Congestion map of a difficult instance
### Running time in practice

<table>
<thead>
<tr>
<th>Chip</th>
<th></th>
<th>C</th>
<th></th>
<th></th>
<th>R</th>
<th></th>
<th>1 thread</th>
<th>4 threads</th>
<th>8 threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>478,946</td>
<td></td>
<td>894,377</td>
<td></td>
<td>0:15:49</td>
<td>0:04:25</td>
<td>0:02:37</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>786,368</td>
<td></td>
<td>1,949,245</td>
<td></td>
<td>1:18:13</td>
<td>0:23:09</td>
<td>0:14:29</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>529,966</td>
<td></td>
<td>1,091,339</td>
<td></td>
<td>0:48:40</td>
<td>0:13:19</td>
<td>0:08:20</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>959,163</td>
<td></td>
<td>2,794,166</td>
<td></td>
<td>1:12:26</td>
<td>0:21:00</td>
<td>0:10:49</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>3,590,647</td>
<td></td>
<td>20,392,657</td>
<td></td>
<td>1:16:07</td>
<td>0:23:27</td>
<td>0:15:09</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>5,340,123</td>
<td></td>
<td>23,606,915</td>
<td></td>
<td>0:33:25</td>
<td>0:12:22</td>
<td>0:08:51</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>7,039,094</td>
<td></td>
<td>22,891,145</td>
<td></td>
<td>2:32:48</td>
<td>0:46:12</td>
<td>0:29:08</td>
<td></td>
</tr>
</tbody>
</table>
Summary

- Min-max resource sharing is a very general problem
- We can solve it efficiently for millions of customers and resources
- Yields provably near-optimum solutions for global routing
- Core global optimization of overall routing flow
Thank you!