A first-order $\varepsilon$-approximation algorithm for linear programs and a second order implementation

Ana Maria A.C. Rocha, University of Minho, Portugal
arocha@dps.uminho.pt

Edite M.G.P. Fernandes, University of Minho, Portugal
emgpf@dps.uminho.pt

João L.C. Soares, University of Coimbra, Portugal
jsoares@mat.uc.pt

Suntec City, Singapore, May 10, 2005
Outline

- Motivation
- Strategy
- Algorithm
- Approximating set-partitioning problems
- Computational results
  * first-order feasible directions method
  * second-order implementation
- Conclusions
Motivation

• Some linear programs arising from real-world applications have such a large number of variables and/or constraints that they hardly can be dealt with simplex type methods (they need a large amount of storage).

• We will be looking for an approximated solution to the linear program

\[ z^* \equiv \min \ cx \]
\[ \text{s.t.} \ Ax \geq b, \]
\[ x \in P, \] (1)

where \( A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m \) and \( P \subseteq \mathbb{R}^n \) is a set over which optimizing linear programs is considered "easy".

• We focus on obtaining a reasonable approximation (\( \varepsilon \)-feasible) to the optimal solution of (1) by a fast method and without too much storage.
A point $x$ is $\varepsilon$-feasible relatively to the constraints $Ax \geq b$ if

$$\lambda(x) \equiv \max_{i=1,\ldots,m} (b_i - a_i x) \leq \varepsilon$$

where $a_i$ denotes the $i$th row of the matrix $A$ and $e$ denotes a column vector of all-ones.

To obtain an $\varepsilon$-approximation to the optimal solution of (1), we solve a sequence of nonlinear programs of the form

$$\Phi(\varepsilon, z) = \min \phi_{\varepsilon}(x) \equiv \sum_{i=1}^{m} (\exp \alpha (b_i - a_i x))$$

s.t. $x \in P'(z) \equiv P \cap \{x : cx \leq z\}$

for given values of the parameters $\varepsilon$ and $z$. The scalar $\alpha$ is a positive penalty parameter that depends on $\varepsilon$ and $z$ is a guess for the value of $z^*$.

To solve the nonlinear program (2) we propose to use

- a first-order feasible directions method
- a second-order implementation.
Algorithm

We will choose $\alpha$ so that it may be possible to assert whether $x$ is $\varepsilon$-feasible from the value of $\phi_\varepsilon(x)$, as formally stated in the next lemma.

**Lemma 1.** If $\alpha \geq \ln((1 + \varepsilon)m)/\varepsilon$ then,

1. if there is no $\varepsilon$-feasible solution relatively to $Ax \geq b$ such that $x \in P'(z)$ then $\Phi(\varepsilon, z) > (1 + \varepsilon)m$.
2. if $x \in P'(z)$ and $\phi_\varepsilon(x) \leq (1 + \varepsilon)m$ then $x$ is $\varepsilon$-feasible solution relatively to $Ax \geq b$.

Thus, keeping the value of $\varepsilon$ fixed, we may use bisection to search for the minimum value of $z$ that produces a feasible solution $x$ in (1) with $cx \leq z$, i.e., $P'(z) \cap \{x : Ax \geq b\}$ is nonempty, being driven by (2).

The bisection search maintains an interval $[z_a, z_b]$ such that $P'(z_a) \cap \{x : Ax \geq b\}$ is empty, i.e.,

- there is no feasible solution $x$ in (1) satisfying $cx \leq z_a$, and
- there is some $x \in P'(z_b)$ that is $\varepsilon$-feasible relatively to $Ax \geq b$.

The search is interrupted when $z_b - z_a$ is small enough to guarantee $z_b \leq z_\ast$. 
Bisection search \((\varepsilon, x_a, z_a, x_b, z_b)\)

**Input:** \(\varepsilon > 0\), and \(x_a \in P'(z_a), x_b \in P'(z_b)\) such that
\[
\Phi (\varepsilon, z_b) \leq \phi_\varepsilon (x_b) \leq (1 + \varepsilon)m < \Phi (\varepsilon, z_a) \leq \phi_\varepsilon (x_a).
\]

**Initialization:** Set \((x^0_a, z^0_a, x^0_b, z^0_b) := (x_a, z_a, x_b, z_b)\) and \(k := 0\).

**Generic Iteration \(k\):**

**Step 1:** If \(z^k_b - z^k_a = 1\) then set \((x_a, z_a, x_b, z_b) := (x^k_a, z^k_a, x^k_b, z^k_b)\) and STOP.

**Step 2:** Set \(z := \lceil (z^k_a + z^k_b)/2 \rceil\) and obtain \(\bar{x} \in P'(z)\) such that
\[
\begin{align*}
\text{either } & \Phi (\varepsilon, z) \leq \phi_\varepsilon (\bar{x}) \leq (1 + \varepsilon)m; \\
\text{or } & (1 + \varepsilon)m < \Phi (\varepsilon, z) \leq \phi_\varepsilon (\bar{x}).
\end{align*}
\]

**Step 3:** Set \((x^{k+1}_a, z^{k+1}_a, x^{k+1}_b, z^{k+1}_b) := \begin{cases} (x^k_a, z^k_a, \bar{x}, z) & \text{if (3) holds}, \\ (\bar{x}, z, x^k_b, z^k_b) & \text{if (4) holds}. \end{cases}\)

Set \(k := k + 1\).
Main $\left(\varepsilon, z_a, x_a, x_b, z_b, \Delta\right)$

**Input:** Define $z_a \leq z_\ast$, $x_a \in P'(z_a)$, a positive integer $\Delta$.
Define $x_b$ (e.g., using Volume Algorithm) and set $z_b := z_a + \Delta$.

**Initialization:**

Choose $\varepsilon > 0$ so that $x_a$ do not overflow $\phi_\varepsilon(x_a)$ and $\Phi(\varepsilon, z_a) > (1 + \varepsilon)m$;

**Generic Iteration:**

*Step 1:* Call **Bisection search** $(\varepsilon, x_a, z_a, x_b, z_b)$.

*Step 2:* While ($\Phi(\varepsilon, z_b) \leq (1 + \varepsilon)m$ and $\varepsilon$ is not small enough)
redefine $x_a$ and set $\varepsilon := \varepsilon/2$.
If ($\varepsilon$ is small enough) then
set $x_b$ as the last solution found and STOP.

*Step 3:* Set $x_a$ as the last solution found, $z_a := z_b$ and $z_b := z_b + \Delta$
While ($\Phi(\varepsilon, z_b + \Delta) > (1 + \varepsilon)m$)
redefine both $x_a$ and $z_a$ and set $z_b := z_b + \Delta$.
Set $x_b$ as the last solution found and set $z_b := z_a + \Delta$.
Repeat *Step 1*. 
Approximating set-partitioning problems

Generically, fractional set-partitioning problems are of the form

\[
\begin{align*}
\text{min} & \quad cx \\
\text{s.t.} & \quad Ax = 1l \\
& \quad x \in [0, 1]^n
\end{align*}
\]

where \( A \) is a \( m \times n \) matrix of 0-1 coefficients and \( 1l \) denotes a vector of ones.

Our framework was tested on real-world set-partitioning problems mostly obtained from the OR-Library (see http://mscmga.ms.ic.ac.uk/info.html).

Barahona and Anbil [1] developed an extension of the subgradient algorithm, the Volume Algorithm, which produces approximate feasible primal and dual solutions to a linear program, much more quickly than solving it exactly.

Hence, the volume algorithm approximately solves the Lagrangian relaxation of a linear problem which is the following problem

\[
\begin{align*}
\text{max} & \quad \min \{ cx + y(1l - Ax) : x \in [0, 1]^n \} \\
\text{s.t.} & \quad y \in \mathbb{R}^m.
\end{align*}
\]
<table>
<thead>
<tr>
<th>Problem</th>
<th>Volume algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>n</td>
</tr>
<tr>
<td>sppnw08</td>
<td>24</td>
</tr>
<tr>
<td>sppnw10</td>
<td>24</td>
</tr>
<tr>
<td>sppnw12</td>
<td>27</td>
</tr>
<tr>
<td>sppnw15</td>
<td>31</td>
</tr>
<tr>
<td>sppnw20</td>
<td>22</td>
</tr>
<tr>
<td>sppnw21</td>
<td>25</td>
</tr>
<tr>
<td>sppnw22</td>
<td>23</td>
</tr>
<tr>
<td>sppnw23</td>
<td>19</td>
</tr>
<tr>
<td>sppnw24</td>
<td>19</td>
</tr>
<tr>
<td>sppnw25</td>
<td>20</td>
</tr>
<tr>
<td>sppnw26</td>
<td>23</td>
</tr>
<tr>
<td>sppnw27</td>
<td>22</td>
</tr>
<tr>
<td>sppnw28</td>
<td>18</td>
</tr>
<tr>
<td>sppnw32</td>
<td>19</td>
</tr>
<tr>
<td>sppnw35</td>
<td>23</td>
</tr>
<tr>
<td>sppnw36</td>
<td>20</td>
</tr>
<tr>
<td>sppnw38</td>
<td>23</td>
</tr>
</tbody>
</table>

Optimization: Theories and Applications (OTA) 2005
Computational results

Computational tests were performed on a PC with a 2.66GHz Pentium IV microprocessor and 512Mb of memory running RedHat Linux 8.0.

The algorithm was implemented in AMPL (Version 7.1) modeling language and when solving the nonlinear subproblem

$$\min_{\epsilon} \phi_{\epsilon}(x) \equiv \sum_{i=1}^{m} (e^{\alpha(1-a_ix)} + e^{\alpha(a_ix-1)})$$

s.t. $$x \in P'(z) \equiv \{x : cx \leq z \text{ and } x \in [0,1]^n\}$$

that arises in **Main algorithm** and **Bisection search** we will use

- a first-order feasible directions algorithm
- a second-order implementation.
A first-order algorithm

The direction of movement at a generic iterate $\bar{x} \in P'(z)$, that is not $\varepsilon$-feasible relatively to the constraints $Ax = 1l$, is determined from solving the following linear program

$$
\begin{align*}
\min \ & \phi_{\varepsilon}(\bar{x}) + \nabla \phi_{\varepsilon}(\bar{x})(x - \bar{x}) \\
\text{s.t.} \ & x \in P'(z).
\end{align*}
$$

(5)

If $\hat{x}$ is optimal in (5) then we reset $\bar{x}$ to $\bar{x} + \hat{\sigma}(\hat{x} - \bar{x})$, for some fixed stepsize $\hat{\sigma} \in (0, 1]$, and proceed analogously to the next iteration.

The conceptual algorithm is halted when $\phi_{\varepsilon}(\bar{x}) \leq (1 + \varepsilon)m$ or a maximum number of iterations is reached.

Note that (5) is equivalent to

$$
\begin{align*}
\max \ & (\bar{y}A) x \\
\text{s.t.} \ & cx \leq z, \\
x \in P,
\end{align*}
$$

(6)

for $\bar{y}$ defined componentwise by

$$
\bar{y}_i = \alpha(\exp(\alpha(1 - a_i\bar{x})) - \exp(\alpha(a_i\bar{x} - 1)))
$$

for $i = 1, 2, \ldots, m$. This is essentially because $\nabla \phi_{\varepsilon}(\bar{x}) = -\bar{y}A$. 

Optimization: Theories and Applications (OTA) 2005 11
Solve subproblem $(\bar{x}, \text{flag})$

**Input:** Given $\bar{x} \in P'(z)$ and a small positive tolerance $\delta$.

**Generic Iteration:**

1. **Step 1:** If $(\phi_\varepsilon(\bar{x}) \leq (1 + \varepsilon)m)$
   - Then set flag := CP1 and STOP;
   - Else if (maximum number of iterations is reached)
     - Then set flag := CP4 and STOP.

2. **Step 2:** Let $\hat{x}$ be optimal for (6).
   - If $(\phi_\varepsilon(\bar{x}) + \nabla \phi_\varepsilon(\bar{x})(\hat{x} - \bar{x}) > (1 + \varepsilon)m)$
     - Then set flag := CP2 and STOP;
   - Else if $(\phi_\varepsilon(\bar{x}) + \nabla \phi_\varepsilon(\bar{x})(\hat{x} - \bar{x}) > \Phi_\varepsilon(\bar{x}) - \delta))$
     - Then set flag := CP3 and STOP;
   - Else if $(\bar{y}A\hat{x} < \bar{y}b)$
     - Then set flag := CP5 and STOP.

3. **Step 3:** Set $\bar{x} := \bar{x} + \sigma_*(\hat{x} - \bar{x})$,
   where $\sigma_* \in \arg \min \{\phi_\varepsilon(\bar{x} + \sigma(\hat{x} - \bar{x})) : \sigma \in (0, 1]\}$.
   Repeat *Step 1*. 
Second order implementation

Calls to the nonlinear programming solver LOQO 6.0. [6]

LOQO is an implementation of a primal-dual interior point method (also known as barrier method) for solving nonlinear constrained optimization problems.
Details of implementation

From the (dual) lower bound thus derived from the Volume Algorithm, we obtain an integral lower bound $z_a$ on the value of $z^*$.

In our Main algorithm the initial $x_a$ was set to all-zeros.

Note that, since $c > 0$, $x_a \in P'(z_a)$.

The primal vector obtained through the Volume Algorithm is, on entry the Main algorithm, the $x_b$ and with this value we compute $z_b = cx_b$.

In our experiments we set:

- $\Delta = 1$
- Initial $\varepsilon = \|1 - Ax_a\| = 1$
- $\varepsilon \leq 10^{-4}$ for stopping criterion
<table>
<thead>
<tr>
<th>Name</th>
<th>Initial</th>
<th>First-order method</th>
<th>Loqo (Second-order)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sppnw08</td>
<td>35894</td>
<td>6.10E-05</td>
<td>4.20E-07</td>
</tr>
<tr>
<td>sppnw10</td>
<td>68147</td>
<td>0.01563</td>
<td>0.00269</td>
</tr>
<tr>
<td>sppnw12</td>
<td>14102</td>
<td>0.00781</td>
<td>0.00121</td>
</tr>
<tr>
<td>sppnw15</td>
<td>67714</td>
<td>0.03125</td>
<td>0.00521</td>
</tr>
<tr>
<td>sppnw20</td>
<td>16604</td>
<td>0.03125</td>
<td>0.00819</td>
</tr>
<tr>
<td>sppnw21</td>
<td>7371</td>
<td>0.03125</td>
<td>0.00573</td>
</tr>
<tr>
<td>sppnw22</td>
<td>6917</td>
<td>0.01563</td>
<td>0.00470</td>
</tr>
<tr>
<td>sppnw23</td>
<td>12301</td>
<td>0.01563</td>
<td>0.00385</td>
</tr>
<tr>
<td>sppnw24</td>
<td>5828</td>
<td>0.01563</td>
<td>0.00296</td>
</tr>
<tr>
<td>sppnw25</td>
<td>5830</td>
<td>0.01563</td>
<td>0.00280</td>
</tr>
<tr>
<td>sppnw26</td>
<td>6733</td>
<td>0.01563</td>
<td>0.00308</td>
</tr>
<tr>
<td>sppnw27</td>
<td>9871</td>
<td>0.00781</td>
<td>0.00175</td>
</tr>
<tr>
<td>sppnw28</td>
<td>8161</td>
<td>0.00781</td>
<td>0.00104</td>
</tr>
<tr>
<td>sppnw32</td>
<td>14560</td>
<td>0.01563</td>
<td>0.00296</td>
</tr>
<tr>
<td>sppnw35</td>
<td>7194</td>
<td>0.01563</td>
<td>0.00393</td>
</tr>
<tr>
<td>sppnw36</td>
<td>7260</td>
<td>0.01563</td>
<td>0.00302</td>
</tr>
<tr>
<td>sppnw38</td>
<td>5541</td>
<td>0.01563</td>
<td>0.00457</td>
</tr>
</tbody>
</table>
Comparing maximal violation

![Graph showing comparisons of maximal violation for different samples, with legend indicating VA, Loqo, and 1st-order categories.](image-url)
Conclusions

The algorithm herein proposed to obtain an $\varepsilon$-feasible solution of a linear programming problem, for a small and adequate $\varepsilon$ value,

- works quite well with the second-order implementation,

- the first-order method converges very slowly,

specially for problems that are not large-scale.

Our strategy enables to improve the quality of the solution found by volume algorithm.

Future developments will focus on improving convergence of the first-order algorithm.

We propose to combine this framework with an heuristic method to get the integer solution.
References


