A First-Order ε-Approximation Algorithm for Linear Programs and a Second-Order Implementation

Ana Maria A.C. Rocha[1], Edite M.G.P. Fernandes[1], and João L.C. Soares[2]

[1] Departamento de Produção e Sistemas, Universidade do Minho, Portugal
{arocha, emgpf}@dps.uminho.pt
[2] Departamento de Matemática, Universidade de Coimbra, Portugal
jsoares@mat.uc.pt

Abstract. This article presents an algorithm that finds an ε-feasible solution relatively to some constraints of a linear program. The algorithm is a first-order feasible directions method with constant stepsize that attempts to find the minimizer of an exponential penalty function. When embedded with bisection search, the algorithm allows for the approximated solution of linear programs. We present applications of this framework to set-partitioning problems and report some computational results with first-order and second-order implementations.

1 Introduction

Set-covering, -partitioning and -packing models arise in many applications like crew scheduling (trains, buses or airplanes), political districting, protection of microdata, information retrieval, etc. Typically these models are suboptimally solved by heuristics because an optimization framework (usually of the branch-and-price type) has to be rather specialized, if feasible at all. Moreover, a branch-and-price framework requires the solution of large linear programs at every node of the branch-and-price tree and these linear programs may take a long time and storage to be solved to optimality.

Our framework attempts to find reasonable approximate solutions to those models quickly and without too much storage, along the lines of Lagrangian relaxation. The approximation obtained may serve the purpose of speeding-up the optimal basis identification by simplex-type algorithms. We will be looking for an approximated solution of a linear program in the following form

$$\begin{aligned} z_* \equiv \min\ & cx \\ \text{s.t. } & Ax \geq b\ , \\ & x \in P\ , \end{aligned} \tag{1}$$

where A is a real $m \times n$ matrix, b is a real m-dimensional vector and $P \subseteq \mathbb{R}^n$ is a compact set (possibly, a lattice) over which optimizing linear programs is considered "easy". For example, in a set-covering model, A is a matrix of zeros

O. Gervasi et al. (Eds.): ICCSA 2005, LNCS 3483, pp. 488–498, 2005.

and ones, b is a vector of all-ones, and the set P is the lattice $\{0,1\}^n$, or the hypercube $[0,1]^n$ in the fractional version. If P includes a budget constraint then P becomes the feasible region of a $0-1$ knapsack problem or a fractional $0-1$ knapsack problem, respectively. We also assume that $\text{conv}(P)$ is a polyhedron.

In recent years, several researchers have developed algorithms (see [3, 4, 2]) that seek to produce approximate solutions to linear programs of this sort, by constructing an exponential potential function which serves as a surrogate for feasibility and optimality.

We focus on obtaining a reasonable approximation to the optimal solution of (1) by an ε-feasible solution. We say that a point x is ε-feasible relatively to the constraints $Ax \geq b$ if

$$\lambda(x) \equiv \max_{i=1,\ldots,m} (b_i - a_i x) \leq \varepsilon \ , \tag{2}$$

where a_i denotes the ith row of the matrix A. To achieve this, we will attempt to solve the following convex program

$$\begin{aligned} \Phi(\alpha, z) = \min \phi_\alpha(x) &\equiv \sum_{i=1}^{m} \exp\left(\alpha\left(b_i - a_i x\right)\right) \\ \text{s.t. } x \in P'(z) &\equiv \text{conv}\left(P \cap \{x\colon cx \leq z\}\right) \end{aligned} \tag{3}$$

where $\text{conv}(\cdot)$ denotes convex hull, for several adequate values of the parameters α and z. The scalar α is a penalty parameter and z is a guess for the value of z_*. To solve the nonlinear program (3) we propose a first-order feasible directions method with constant stepsize whose running time depends polynomially on $1/\varepsilon$ and the *width* of the set $P'(z)$ relatively to the constraints $Ax \geq b$. Significantly, the running time does not depend explicitly on n and, hence, it can be applied when n is exponentially large, assuming that, for a given row vector y, there exists a polynomial subroutine to optimize yAx over $P'(z)$.

This paper is organized as follows. In Sect. 2 we describe the ε-approximation algorithm used in this work. Then, in Sect. 3 we present the first-order algorithm with fixed stepsize. Our computational experience on set-partitioning problems is presented in Sect. 4 and Sect. 5 contains some conclusions and future work.

2 Main ε-Approximation Algorithm

If x is feasible in (1) then $\phi_\alpha(x) \leq m$, while if x is simply ε-feasible then $\phi_\alpha(x) \leq m\exp(\alpha\varepsilon)$. On the other hand, if x is not ε-feasible then $\phi_\alpha(x) > \exp(\alpha\varepsilon)$. We will choose α so that it may be possible to assert whether x is ε-feasible from the value of $\phi_\alpha(x)$, as formally stated in the next lemma.

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

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

Proof. If there is no ε-feasible solution relatively to '$Ax \geq b$' such that $x \in P'(z)$ then $\varepsilon < \varepsilon' \equiv \min\{\lambda(x)\colon x \in P'(z)\}$. Thus, $\Phi(\alpha, z) > \exp(\alpha\varepsilon') \geq (1+\varepsilon)m$. If $\phi_\alpha(x) \leq (1+\varepsilon)m$ then $\exp(\alpha(b_i - a_i\bar{x})) \leq (1+\varepsilon)m$, for any $i = 1, \ldots, m$. Equivalently, $b_i - a_i\bar{x} \leq \ln((1+\varepsilon)m)/\alpha \leq \varepsilon$, for any $i = 1, \ldots, m$. □

Keeping the value of α fixed, we will use bisection to search for the minimum value of z such that $P'(z) \cap \{x\colon Ax \geq b\}$ is nonempty, being driven by (3). The bisection search maintains an interval $[z_a, z_b]$ such that $P'(z_a) \cap \{x\colon Ax \geq b\}$ is empty, and there is some $x \in P'(z_b)$ that is ε-feasible relatively to '$Ax \geq b$'. The search is interrupted when $z_b - z_a$ is small enough to guarantee $z_b \leq z_*$. This does not imply any bound on how much z_b differs from z_*. It may be possible that z_b is much less than z_* though very unlikely for α large, as it is implicit from Proposition 1 below.

Proposition 1. *Let the sequence* $\{\alpha_k\}$ *be such that* $\lim_k \alpha_k = +\infty$ *and* y^k *be a vector defined componentwise by*

$$y_i^k = \alpha_k \exp\left(\alpha_k \left(b_i - a_i x^k\right)\right) \quad (i = 1, 2, \ldots, m) \ , \tag{4}$$

where x^k *is optimal in* $\Phi(\alpha_k, z_*)$*. Then, for every accumulation point* $(\bar{x}, \bar{y})$ *of the sequence* $\{(x^k, y^k)\}$*,* $\bar{x}$ *is optimal for program (1).*

Proof. Assume $\lim_{k\in K}(x^k, y^k) = (\bar{x}, \bar{y})$. Since P is closed, $\bar{x} \in P'(z_*)$. If we show that $A\bar{x} \geq b$ then $\bar{x}$ must be optimal in (1). Since $\lim_{k\in K} \alpha_k \exp\left(\alpha_k \left(b_i - a_i x^k\right)\right)$ exists then $\bar{y}_i = \lim_{k\in K} \alpha_k \exp(\alpha_k (b_i - a_i\bar{x}))$, from where we conclude that $a_i\bar{x} \leq b_i$, for every $i = 1, 2, \ldots, m$. □

If $[z_a^k, z_b^k]$ is the last bisection interval of the search for a given ε_k then, we may restart the bisection search with $[z_b^k, z_b]$ for some $\varepsilon_{k+1} < \varepsilon_k$, for example, $\varepsilon_{k+1} = \varepsilon_k/2$. The value of z_b denotes a proven upper bound on the value of z_*. Note that an ε_k-feasible solution is known at the left extreme of the new interval and furthermore it belongs to $P'(z)$, for any $z \in [z_b^k, z_b]$. Thus, it may serve as initial solution for the minimization of ϕ_α when $\varepsilon = \varepsilon_{k+1}$ and will not *overflow* the exponential function evaluations.

Each iteration of our main algorithm consists of a number of iterations of bisection search on the z value for a fixed value of the parameter ε. Then, ε is decreased, the bisection interval is updated and bisection search restarts. This process is terminated when ε is small enough.

From Lemma 1, if $\Phi(\varepsilon, z) \leq (1+\varepsilon)\, m$ then there is $x \in P'(z)$ that is ε-feasible relatively to '$Ax \geq b$' (for example, the optimal solution); otherwise, there is no feasible solution x in (1) satisfying $cx \leq z$. The new interval $[z_a^{k+1}, z_b^{k+1}]$ is adequately updated and k is increased by one unit. Termination occurs when $z_b^k - z_a^k$ is small enough. The algorithm is formally described in Algorithm 1. Step 2 of the algorithm **Bisection search** involves applying an off-the-shelf convex optimization method to (3). Note that, we always have $x_a^k \in P'(z)$ which makes it a natural starting point for the corresponding optimization algorithm.

Algorithm 1. Bisection search $(\varepsilon, x_a, z_a, x_b, z_b)$
Given $0 < \varepsilon$, and $x_a \in P'(z_a)$, $x_b \in P'(z_b)$ such that

$$\Phi(\varepsilon, z_b) \le \phi_\varepsilon(x_b) \le (1+\varepsilon)m < \Phi(\varepsilon, z_a) \le \phi_\varepsilon(x_a) \quad . \tag{5}$$

Initialization: Set $(x_a^0, z_a^0, x_b^0, z_b^0) := (x_a, z_a, x_b, z_b)$ and $k := 0$.

Generic Iteration k:

Step 1: If $z_b^k - z_a^k = 1$ then set $(x_a, z_a, x_b, z_b) := (x_a^k, z_a^k, x_b^k, z_b^k)$ and STOP.
Step 2: Set $z := \lceil (z_a^k + z_b^k)/2 \rceil$ and obtain $\bar{x} \in P'(z)$ such that

$$\text{either} \quad \Phi(\varepsilon, z) \le \phi_\varepsilon(\bar{x}) \le (1+\varepsilon)m \quad , \tag{6}$$

$$\text{or} \quad (1+\varepsilon)m < \Phi(\varepsilon, z) \le \phi_\varepsilon(\bar{x}) \quad . \tag{7}$$

Step 3: Set $(x_a^{k+1}, z_a^{k+1}, x_b^{k+1}, z_b^{k+1}) := \begin{cases} (x_a^k, z_a^k, \bar{x}, z) & \text{if (6) holds,} \\ (\bar{x}, z, x_b^k, z_b^k) & \text{if (7) holds.} \end{cases}$
Set $k := k+1$.

We may now present a formal description of the **Main** algorithm (see Algorithm 2). On entry: $z_a = \lceil \min\{cx - y(Ax-b) \colon x \in P\} \rceil$ is an integral lower bound on the value of z_*, for some fixed $y \ge 0$; $x_a \in P'(z_a)$ and Δ is a positive integer related to the initial amplitude of the starting intervals in each bisection search. Overall, we have the following convergence result. After a finite number of calls, the last interval $[z_b - 1, z_b]$ of the **Bisection search** routine is such that $z_b = \lceil \bar{z} \rceil$, where $\bar{z} = \min\{z \colon x \in P'(z), Ax \ge b\}$. In general $\bar{z} \le z_*$, with equality if P is a polyhedron.

Algorithm 2. Main $(z_a, x_a, z_b, x_b, \Delta)$
Given $z_a \le z_*$, $x_a \in P'(z_a)$ and a positive integer Δ.
Initialization:

Choose $\varepsilon > 0$ so that x_a do not *overflow* $\phi_\varepsilon(x_a)$ and $\Phi(\varepsilon, z_a) > (1+\varepsilon)m$.
While $\Phi(\varepsilon, z_a + \Delta) > (1+\varepsilon)m$ redefine x_a and set $z_a := z_a + \Delta$.
Define x_b as the last solution found and set $z_b := z_a + \Delta$.

Generic Iteration:

Step 1: Call **Bisection search** $(\varepsilon, x_a, z_a, x_b, z_b)$.
Step 2: While $(\Phi(\varepsilon/2, z_b) \le (1+\varepsilon/2)m$ and ε is not small enough)
redefine x_b and set $\varepsilon := \varepsilon/2$.
If (ε is small enough)
Then set x_b as the last solution found and STOP.
Step 3: Set $x_a := x_b$ and $z_a := z_b$.
While $(\Phi(\varepsilon, z_b + \Delta) > (1+\varepsilon)m$) redefine x_a, z_a and set $z_b := z_b + \Delta$.
Set x_b as the last solution found and set $z_b := z_b + \Delta$.
Repeat *Step 1.*

3 A First-Order Algorithm with Fixed Stepsize

Our conceptual algorithm to solving (3) is a first-order iterative procedure with a fixed stepsize. The algorithm coincides with the algorithm Improve-Packing proposed in [5] but the stepsize and the stopping criterion are different. The direction of movement at a generic iterate $\bar{x} \in P'(z)$, that is not ε-feasible relatively to the constraints '$Ax \geq b$', is determined from solving the following linear program

$$\begin{array}{l} \min \phi_\varepsilon(\bar{x}) + \nabla\phi_\varepsilon(\bar{x})(x - \bar{x}) \\ \text{s.t. } x \in P'(z) \ . \end{array} \tag{8}$$

If $\hat{x}$ is optimal in (8) 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. Of course, in practice other stopping criteria should be accounted for. For example, notice that (8) is equivalent to

$$\begin{array}{ll} \max & (\bar{y}A)\, x \\ \text{s.t.} & cx \leq z \ , \\ & x \in P \ , \end{array} \tag{9}$$

for $\bar{y}$ defined componentwise by $\bar{y}_i = \alpha \exp(\alpha(b_i - a_i\bar{x}))$, for $i = 1, 2, \ldots, m$. This is essentially because $\nabla\phi_\varepsilon(\bar{x}) = -\bar{y}A$. Then, since $\bar{y} \geq 0$, the inequality '$\bar{y}Ax \geq \bar{y}b$' is valid for the polyhedron $\{x\colon Ax \geq b\}$. Thus, if at some point of the solution procedure of (3) we have that the optimal value of (9) is smaller than $\bar{y}b$ then we may immediately conclude that $P'(z) \cap \{x\colon Ax \geq b\}$ is empty.

Theorem 1 below presents one particular choice for the fixed stepsize $\hat{\sigma}$. It depends on the following quantity, introduced as the *width* of $P'(z)$ relatively to the constraints '$Ax \geq b$' in [5],

$$\rho \equiv \left\{ \begin{array}{l} \sup \|Ax - Ay\|_\infty \\ \text{s.t. } x, y \in P'(z) \end{array} \right\} = \left\{ \max_{i=1,2,\ldots,m} \left(\begin{array}{l} \sup \mid a_i x - a_i y \mid \\ \text{s.t. } x, y \in P'(z) \end{array} \right) \right\} \ . \tag{10}$$

In [5], ρ is defined differently depending when whether the matrix A is such that $Ax \geq 0$, for every $x \in P'(z)$, or not. If yes, then the two definitions coincide with $\rho = \max_i \max_{x \in P'(z)} a_i x$. If not, then our definition of ρ is half of the ρ that is proposed in [5].

Theorem 1. *([6]) Assume that $z_* \leq z$, $\bar{x} \in P'(z)$ and it is not ε-feasible (relatively to '$Ax \geq b$'), for some $\varepsilon \in (0, 1)$, $\hat{x} \in P'(z)$ is optimal for program (8), ρ is given by (10) and*

$$\alpha \geq \max\left(\frac{\ln(m(3+\varepsilon))}{\varepsilon}, \frac{1}{\rho \ln 2} \right) \ . \tag{11}$$

Then, for $\hat{\sigma} = 1/(\alpha\rho)^2$ we have that

$$\phi_\varepsilon\left(\bar{x} + \hat{\sigma}(\hat{x} - \bar{x})\right) < \left(1 - \frac{1}{4(\alpha\rho)^2} \frac{1+\varepsilon}{3+\varepsilon}\right) \phi_\varepsilon(\bar{x}) \ . \tag{12}$$

In summary, assuming that $z_* \leq z$, if the first k iterates are not ε-feasible then

$$\phi_\varepsilon(x^{k+1}) < \left(1 - \frac{1}{4(\alpha\rho)^2}\frac{1+\varepsilon}{3+\varepsilon}\right)^k \phi_\varepsilon(x^0) \ , \tag{13}$$

where, we note, that the right hand side goes to zero as k goes to $+\infty$. The following corollary states a worst-case complexity bound on the number of iterations of the algorithm.

Corollary 1. *([6]) If α satisfies (11) and $\varepsilon \in (0,1)$ then our algorithm, using $\hat{\sigma} = 1/(\alpha\rho)^2$ and starting from $x^0 \in P'(z)$, terminates after*

$$\frac{\ln(m) + \ln(1+\varepsilon) - \ln \phi_\varepsilon(x^0)}{\ln\left(1 - \frac{1}{4(\alpha\rho)^2}\frac{1+\varepsilon}{3+\varepsilon}\right)} < 16\alpha^3\rho^2\lambda(x^0) \tag{14}$$

iterations, with an $x \in P'(z)$ that is ε-feasible relatively to 'Ax $\geq$ b' or, otherwise, with the proof that there is no x feasible in (1) such that $cx \leq z$.

Note that the right hand side of (14) is $\mathcal{O}(\ln^3(m)\varepsilon^{-3}\rho^2\lambda(x^0))$. If $\lambda(x^0) = \mathcal{O}(\varepsilon)$ then only $\mathcal{O}(\ln^3(m)\varepsilon^{-2}\rho^2) = \tilde{\mathcal{O}}(\varepsilon^{-2}\rho^2)$ iterations are required. This complexity result is related to Karger and Plotkin's [4–Theorem 2.5] ($\tilde{\mathcal{O}}(\varepsilon^{-3}\rho)$) and Plotkin, Shmoys and Tardos's [5–Theorem 2.12] ($\tilde{\mathcal{O}}(\varepsilon^{-2}\rho\ln\varepsilon^{-1})$). The result of Karger and Plotkin is valid even if the budget constraint is included in the objective function of (3) without counting in the definition of ρ. We recall that a function $f(n)$ is said to be $\tilde{\mathcal{O}}(g(n))$ if there exists a constant c such that $f(n)\ln^c(n) \geq \mathcal{O}(g(n))$.

Given an initial point $\bar{x} \in P'(z)$, the Algorithm 3 contains a practical implementation of the first-order algorithm to solving (3).

Algorithm 3. Solve_subproblem $(\bar{x}, \text{flag})$
Given $\bar{x} \in P'(z)$ and a small positive tolerance δ.
Generic Iteration:

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.

Step 2: Let $\hat{x}$ be optimal for (9).
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.

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*.

On exit of this routine, $\bar{x}$ should be understood as optimal in (3) when flag $\neq$ **CP4**. When this is the case, output should be interpreted as follows: $\Phi(\varepsilon, z) \leq (1+\varepsilon)m \iff \text{flag} \in \{\mathbf{CP1}, \mathbf{CP3}\}$.

Step 2 of the routine **Solve_subproblem** requires solving a program with a linear objective function. Interesting possibilities are: (1) $P = [0,1]^n$, in which case (9) can be solved by a greedy type algorithm; (2) $P = \{0,1\}^n$, in which case (9) is a *0-1 knapsack* problem and can be solved by the CPLEX MIP solver; (3) P is a generic polyhedron in which case (9) can be solved by the CPLEX solver.

In what follows we will expand on how the line search is performed assuming that we are looking for an ε-feasible solution relatively to an equality system $Ax = b$, as this is the case with the test problems studied. Step 3 of the algorithm **Solve_subproblem** aims at finding a minimizer σ_* of the function $g\colon [0,1] \to \mathbb{R}$ defined by $g(\sigma) \equiv \phi_\varepsilon(\bar{x} + \sigma d) \equiv \sum_{i=1}^m \phi_i(\bar{x} + \sigma d) \equiv 2\sum_{i=1}^m \cosh(a_i\bar{x} - b_i + \sigma a_i d)$, where $d = \hat{x} - \bar{x}$ and cosh denotes de hyperbolic cosine function. Note that g is convex and $g'(0) < 0$. The numerical method of choice would be the unidimensional Newton's method for it is, in this case, globally convergent and possesses locally quadratic convergence.

However, since the functions ϕ_i are defined by exponential functions, Newton's method may require too many iterations to reach its region of quadratic convergence. A method like bisection search may be more adequate at early stages of the minimization process. Our initial bisection interval is implied by the following result.

Theorem 2. *([6]) For every $i \in \{1, 2, \ldots, m\}$, let U_i be an upper bound on the value of $\phi_i(\bar{x} + \sigma_* d)$. Then, for every i such that $a_i d \neq 0$, the following holds,*

$$\sigma_* \leq \frac{1}{\alpha a_i d} \ln\left(\frac{U_i + \operatorname{sgn}(a_i d)\sqrt{U_i^2 - 4}}{2\exp\left(\alpha\left(a_i\bar{x} - b_i\right)\right)}\right) \quad . \tag{15}$$

where $\operatorname{sgn}(\cdot)$ *denotes the sign function.*

The bound (15) is important for the search for σ_* not to overflow the exponential function evaluations.

Bienstock [2] proposed to use bisection search until a sufficiently small interval is found and only then start with Newton's method. We propose a slightly different procedure. We propose to try two Newton's steps departing from the left limit of the current bisection interval. By using these two Newton points we make a judgement of whether the region of quadratic convergence for Newton's method was achieved. If yes, we switch to the pure Newton's method. More precisely, if σ_{N1} is the first Newton point and σ_{N2} is the second Newton point then we consider that the region of quadratic convergence is achieved if

$$\left(c_1 g'(\sigma_{\text{N2}})^2 \leq |g'(\sigma_{\text{N1}})| \text{ if } |g'(\sigma_{\text{N2}})| > 1\right) \text{ or } \left(|g'(\sigma_{\text{N2}})| \leq c_2 g'(\sigma_{\text{N1}})^2 \text{ if } |g'(\sigma_{\text{N2}})| \leq 1\right) \tag{16}$$

where c_1, c_2 are positive constants. Otherwise, the interval is updated by using the information on these points, the midpoint σ_{mid} is also tried and the bisection interval is again updated. The process restarts.

4 Computational Results with Set-Partitioning Problems

In this section we report computational results for the first-order method previously discussed and with a second-order nonlinear programming package. The 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.

Here, we are looking for finding "good" ε-feasible solutions of fractional set-partitioning problems arising in airline crew scheduling. Some of these linear programs are extremely difficult to solve with traditional algorithms. Generically, fractional set-partitioning problems are of the form

$$\begin{aligned} &\min\ cx \\ &\text{s.t.}\ \ Ax = \mathbb{1} \\ &\qquad x \in [0,1]^n\ , \end{aligned} \tag{17}$$

where A is a $m \times n$ matrix, with 0-1 coefficients, and $\mathbb{1}$ denotes a vector of ones. Our framework was tested on real-world set-partitioning problems obtained from the OR-Library (http://www.brunel.ac.uk/depts/ma/research/jeb/info.html).

Table 1. Set-partitioning problems

Name	m	n	z^{LP}	**Vol It.**	**Vol Dual**	**Vol Primal**	**Vol Viol.**	**Vol Time**
sppnw08	24	434	35894	356	35894.0	36188.0	0.01889	0.02
sppnw10	24	853	68271	501	68146.8	68510.8	0.01974	0.04
sppnw12	27	626	14118	447	14101.4	14222.6	0.01859	0.03
sppnw15	31	467	67743	483	67713.8	67407.0	0.01934	0.03
sppnw20	22	685	16626	408	16603.4	16645.2	0.01991	0.02
sppnw21	25	577	7380	358	7370.8	7387.1	0.01875	0.03
sppnw22	23	619	6942	340	6916.8	6917.4	0.01903	0.02
sppnw23	19	711	12317	357	12300.1	12412.8	0.01971	0.03
sppnw24	19	1366	5843	313	5827.8	5885.3	0.01924	0.04
sppnw25	20	1217	5852	305	5829.8	5851.3	0.01763	0.04
sppnw26	23	771	6743	304	6732.8	6742.4	0.01584	0.03
sppnw27	22	1355	9877.5	363	9870.0	9934.8	0.01933	0.05
sppnw28	18	1210	8169	342	8160.1	8167.5	0.01995	0.04
sppnw32	19	294	14570	430	14559.9	14559.6	0.01748	0.01
sppnw35	23	1709	7206	334	7194.0	7262.6	0.01633	0.06
sppnw36	20	1783	7260	631	7259.2	7225.3	0.01995	0.13
sppnw38	23	1220	5552	321	5540.5	5526.0	0.01951	0.04

Before a call to **Main** algorithm we apply the volume algorithm, developed by Barahona and Anbil [1], that is an extension of the subgradient algorithm, which produces approximate feasible primal and dual solutions to a linear program, much more quickly than solving it exactly. This algorithm approximately solves the Lagrangian relaxation of the problem (17), i.e.,

$$\begin{aligned} &\max \min \{cx - y\,(Ax - \mathbb{1}) : x \in [0,1]^n\} \\ &\text{s.t.} \quad y \in \mathbb{R}^m \end{aligned}$$

requiring a not too demanding number of iterations.

The main characteristics of the selected problems are described in Table 1.

The first four columns of this table identify the problem by its name, m, n, and z^{LP}, the known optimal value of the linear programming relaxation. The remaining columns are related to the volume algorithm [1], namely, the number of iterations up to finding a primal vector where each constraint is violated by at most 0.02 or the difference between the dual lower bound and the primal value is less than 1%. We also present the time required by the volume algorithm (in seconds).

From the (dual) lower bound derived from volume algorithm we obtain an integral lower bound z_a on the value of z_*. The initial x_a was set to all-zeros because often the primal solution obtained through the volume algorithm would not satisfy $cx \leq z_a$. Note that, since $c > 0$, $x_a \in P'(z_a)$. In our experiments we have chosen to set $\Delta = 1$.

We implemented the first-order feasible directions minimization algorithm, presented in Sect. 3, to solve the nonlinear problem (3). In Step 2 of the **Solve_subproblem** algorithm we used the software CPLEX (version 7.1) to get an optimal solution for the linear problem.

Table 2 is divided into four parts. The first part identifies the set-partitioning problem. Next, we have the initial value for z_a (input to **Main**), that is the lower bound found by volume algorithm.

Table 2. Results of solving (3) with first-order and second-order implementations

		First-order				Second-order		
Name	**Initial**	**Final**	**Max.**	**Out.**	**Time**	**Max.**	**Out.**	**Time**
	z_a	ε	**Viol.**	**It.**	**(sec.)**	**Viol.**	**It.**	**(sec.)**
sppnw08	35894	6.10E-05	4.20E-07	1	469.86	1.28E-14	1	21.48
sppnw10	68147	0.01563	0.00269	4*	26404.26	9.15E-10	6	3546.41
sppnw12	14102	0.00781	0.00121	3*	17301.78	7.41E-14	4	134.80
sppnw15	67714	0.03125	0.00521	1*	4522.86	5.79E-09	5	124.87
sppnw20	16604	0.03125	0.00819	1*	5653.53	9.88E-09	4	275.65
sppnw21	7371	0.03125	0.00573	1*	4752.56	8.96E-10	4	110.24
sppnw22	6917	0.01563	0.00470	3*	7919.76	2.68E-10	5	230.26
sppnw23	12301	0.01563	0.00385	2*	9034.06	2.74E-11	5	15173.3
sppnw24	5828	0.01563	0.00296	3*	44584.35	1.38E-09	4	2107.83
sppnw25	5830	0.01563	0.00280	6*	59902.49	1.14E-09	5	1718.74
sppnw26	6733	0.01563	0.00308	3*	16722.60	1.91E-08	4	275.63
sppnw27	9871	0.00781	0.00175	3*	23535.21	2.18E-14	5	1588.36
sppnw28	8161	0.00781	0.00104	4*	23150.30	8.65E-15	4	1147.54
sppnw32	14560	0.01563	0.00296	3*	3059.28	8.72E-10	4	12.51
sppnw35	7194	0.01563	0.00393	2*	19234.21	1.92E-09	5	5420.31
sppnw36	7260	0.01563	0.00302	1*	32498.39	4.23E-13	1	3049.8
sppnw38	5541	0.01563	0.00457	2*	26496.43	1.43E-10	4	1677.83

In the third part, we present the results of the first-order implementation. For all these problems the initial value for ε is 1, because at the beginning of the algorithm the primal vector x_a is a vector of zeros and $\varepsilon = \|\mathbb{1} - Ax_a\|_\infty = 1$. We exhibit the final value for ε, the maximal violation obtained for the constraints, the number of outer iterations, which corresponds to the number of bisection calls, and the time (in seconds) required. For all problems, except sppnw08, no optimal $\bar{x}$ was found (flag=**CP4**), as the maximum number of iterations (set to 20000) was reached, as pointed out with the character $*$ in the table. Nevertheless, the maximal constraint violation obtained is reduced at least 60% (compare with Table 1).

Then we analyze the strategy of solving the nonlinear subproblem (3) using solver LOQO 6.0. LOQO is an implementation of a primal-dual interior point method for solving nonlinear constrained optimization problems. The fourth part of Table 2 summarizes the results. We remark that, for all the problems, the algorithm halted because the stopping criterion in the **Main** algorithm ($\varepsilon < 10^{-4}$) was achieved. The maximal violation of the constraints is now much smaller than the one obtained with the volume algorithm. A different ε reduction scheme ($\varepsilon := \varepsilon/10$) in the Step 2 of the **Main** algorithm was tried and, in general, the number of outer iterations decreased by one or two units and the time spent was smaller in 71% of the problems.

5 Conclusions

In this paper we propose an algorithm for finding an ε-feasible solution relatively to the constraints of a linear program. The first-order version of the algorithm attempts to minimize a linear approximation of an exponential penalty function using feasible directions and a constant stepsize. The second-order implementation uses the software LOQO to minimize the exponential function.

Our preliminary computational experiments show that the first-order method does not perform as expected when solving set-partitioning problems. In particular, the algorithm converges very slowly and does not reach an ε-approximate solution, for $\varepsilon < 10^{-4}$, for all tested problems but for sppnw08.

We may also conclude that the second-order implementation of the algorithm works quite well, specially for problems that are not large-scale, finding an ε-feasible solution in (1) satisfying $cx < z$, for a small and adequate ε value. Future developments will focus on improving convergence of the first-order algorithm.

References

1. Barahona, F., Anbil, R.: The volume algorithm: Producing primal solutions with a subgradient method. Math. Prog. **87** (2000) 385–399
2. Bienstock, D.: Potential Function Methods for approximately solving linear programming problems: theory and practice. Kluwer Academic Publishers (2002)
3. Grigoriadis, M.D., Khachiyan, L.G.:Fast approximation schemes for convex programs with many blocks and coupling constraints. SIAM J. Optim. **4** 1994 86–107

4. Karger, D., Plotkin, S.: Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows. Proceedings of the 27th Annual ACM Symposium on Theory of Computing (1995) 18 – 25
5. Plotkin, S., Shmoys, D.B., Tardos, E.: Fast approximation algorithms for fractional packing and covering problems. Math. Oper. Res. **20** (1995) 495–504
6. Rocha, A.M.: Fast and stable algorithms based on the Lagrangian relaxation. PhD Thesis (in portuguese), Universidade do Minho, Portugal (2004) (forthcoming)