Computational Optimization and Applications, 35, 375–398, 2006

DOI: 10.1007/s10589-006-8720-6

On the Smoothing of the Square-Root Exact Penalty Function for Inequality Constrained Optimization

ZHIQING MENG mengzhiqing@zjut.edu.cn
College of Business and Administration, Zhejiang University of Technology, Hangzhou, Zhejiang 310032, China

CHUANGYIN DANG mecdang@cityu.edu.hk
Department of Manufacturing Engineering & Engineering Management, City University of Hong Kong, Kowloon, Hong Kong

XIAOQI YANG mayangxq@polyu.edu.hk
Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

Received June 17, 2004; Revised July 18, 2005

Published online: 7 June 2006

Abstract. In this paper we propose two methods for smoothing a nonsmooth square-root exact penalty function for inequality constrained optimization. Error estimations are obtained among the optimal objective function values of the smoothed penalty problem, of the nonsmooth penalty problem and of the original optimization problem. We develop an algorithm for solving the optimization problem based on the smoothed penalty function and prove the convergence of the algorithm. The efficiency of the smoothed penalty function is illustrated with some numerical examples, which show that the algorithm seems efficient.

Keywords: constrained optimization, penalty function, exact penalty function, smoothing method, ϵ-feasible solution, optimal solution

1. Introduction

Consider

$$
\begin{aligned}
(\mathrm{P}):\ \min\ & f(x) \\
\text{s.t.}\ & g_i(x) \le 0, \quad i = 1, 2, \ldots, m,
\end{aligned}
$$

where $f,\ g_i : R^n \to R,\ i \in I = \{1, 2, \ldots, m\}$. Let

$$X_0 = \{x \in R^n \mid g_i(x) \le 0,\ i = 1, 2, \ldots, m\}.$$

To solve (P), the penalty function methods have been proposed in the literature. One of the popular penalty functions is given by

$$F_2(x, \rho) = f(x) + \rho \sum_{i=1}^{m} \max\{g_i(x), 0\}^2. \tag{1}$$

Note that (1) is not an exact penalty function. In Zangwill [1], an exact penalty function was defined as follows:

$$F_1(x, \rho) = f(x) + \rho \sum_{i=1}^{m} \max\{g_i(x), 0\}. \tag{2}$$

After Zangwill's development, extensive research of exact penalty function methods has been carried out in the literature (e.g, [2–7]). However, (2) is not a smooth function and causes some numerical instability problems in its implementation when the value of the penalty parameter ρ becomes larger. In practice, we only need to obtain an approximately optimal solution to (P). In fact, we can only get an approximate solution because of the finite precision of a computer. In order to improve the smoothness of an exact penalty function, Zenios et al. [9] and Pinar and Zenios [10] give a smooth exact penalty function for convex constrained optimization problems, which can be applied to obtain a good approximately optimal solution to (P). Chen and Mangasarian [11] obtains by integrating the sigmoid function $1/(1 + e^{-\alpha t})$ a smooth function to approximate $\max\{0, t\}$. Yang et al. [12] develop a method for smoothing an exact penalty function.

In this paper, we propose two new methods for smoothing the exact penalty function of (P) given by

$$F(x, \rho) = f(x) + \rho \sum_{i=1}^{m} \sqrt{\max\{g_i(x), 0\}}. \tag{3}$$

The corresponding unconstrained optimization problem to (3) is given by

$$(\mathrm{P}_\rho): \quad \min F(x, \rho) \quad s.t. \quad x \in R^n.$$

The penalty function $F(x, \rho)$ is exact but not smooth. We smooth the penalty function (3) so that it can been applied to solve the problem (P) via a gradient-type or Newton-type method. Numerical results show that the smoothed penalty function is numerically more efficient and stable than the penalty functions (1) and (2) for computing an approximate solution to (P).

The rest of this paper is organized as follows. In Section 2, we propose a method for smoothing the penalty function (3) in terms of first-order differentiability, which yields a first-order continuously differentiable penalty function. We prove some error estimates among the optimal objective function values of the smoothed penalty problem, of the nonsmooth penalty problem and of the original constrained optimization problem. We present an algorithm to compute an approximate solution to (P) based on the smooth penalty function and show the convergence of the algorithm. In Section 3, we propose another method for smoothing the penalty function (3) in terms of second-order differentiability, which yields a second-order continuously differentiable penalty function. We prove some error estimates and give an algorithm to compute an approximate solution to (P) based on the smooth penalty function. In Section 4, we give numerical results and conclude the paper with some remarks.

2. A first-order differentiable smoothing method

We define a function $p_\epsilon(t)$ by

$$p_\epsilon(t) = \begin{cases} 0 & \text{if} \quad t < 0, \\ \dfrac{1}{3\epsilon} t^{\frac{3}{2}} & \text{if} \quad 0 \le t < \epsilon, \\ t^{\frac{1}{2}} - \dfrac{2}{3}\epsilon^{\frac{1}{2}} & \text{if} \quad \epsilon \le t. \end{cases}$$

Let $p(t) = \sqrt{\max\{t, 0\}}$. Then,

$$\lim_{\epsilon \to 0} p_\epsilon(t) = p(t).$$

Assume that f and g_i, $i \in I$ are first-order continuously differentiable. Let $g_i^+(x) = \max\{0, g_i(x)\}$ for any $i \in I$. Consider the penalty function for (P) given by

$$F(x, \rho, \epsilon) = f(x) + \rho \sum_{i \in I} p_\epsilon(g_i(x)), \tag{4}$$

where $\rho > 0$. Clearly, $F(x, \rho, \epsilon)$ is first-order continuously differentiable at any $x \in R^n$. Applying $F(x, \rho, \epsilon)$, we obtain the following penalty problem,

$$(PI_\rho): \qquad \min F(x, \rho, \epsilon) \quad s.t. \quad x \in R^n.$$

Since $\lim_{\epsilon \to 0} F(x, \rho, \epsilon) = F(x, \rho)$ for any given ρ, we will first study the relationships between (P_ρ) and (PI_ρ).

Lemma 2.1. *For any $x \in R^n$ and $\epsilon > 0$,*

$$0 \le F(x, \rho) - F(x, \rho, \epsilon) \le \frac{2}{3} m \rho \epsilon^{\frac{1}{2}}. \tag{5}$$

Proof: From the definition of $p_\epsilon(t)$, we obtain

$$0 \le p(t) - p_\epsilon(t) \le \frac{2}{3}\epsilon^{\frac{1}{2}}.$$

Then, for any $x \in X$,

$$0 \leq p(g_i(x)) - p_\epsilon(g_i(x)) \leq \frac{2}{3}\epsilon^{\frac{1}{2}}, \quad \forall i \in I.$$

Thus,

$$0 \leq \sum_{i \in I} p(g_i(x)) - \sum_{i \in I} p_\epsilon(g_i(x)) \leq \frac{2}{3}m\epsilon^{\frac{1}{2}}.$$

Therefore,

$$0 \leq F(x, \rho) - F(x, \rho, \epsilon) \leq \frac{2}{3}m\rho\epsilon^{\frac{1}{2}}$$

since $\rho > 0$. □

As a direct result of Lemma 2.1, we obtain the following two theorems.

Theorem 2.1. *Let $\{\varepsilon_j\} \to 0$ be a sequence of positive numbers and assume that x^j is a solution to $\min_{x \in R^n} F(x, \rho, \varepsilon_j)$ for some given $\rho > 0$. Let $\bar{x}$ be an accumulating point of the sequence $\{x^j\}$. Then $\bar{x}$ is an optimal solution to $\min_{x \in R^n} F(x, \rho)$.*

Theorem 2.2. *Let x^* be an optimal solution of (P_ρ) and $\bar{x} \in X$ an optimal solution of (PI_ρ). Then,*

$$0 \leq F(x^*, \rho) - F(\bar{x}, \rho, \epsilon) \leq \frac{2}{3}m\rho\epsilon^{\frac{1}{2}}. \tag{6}$$

Definition 2.1. A point $x_\epsilon \in X$ is an ϵ-feasible solution or an ϵ-solution if

$$g_i(x_\varepsilon) \leq \epsilon, \quad \forall i \in I.$$

Theorem 2.3. *Let x^* be an optimal solution of (P_ρ) and $\bar{x} \in R^n$ an optimal solution of (PI_ρ). Furthermore, let x^* be feasible to (P) and $\bar{x}$ ϵ-feasible to (P). Then*

$$0 \leq f(x^*) - f(\bar{x}) \leq \frac{4}{3}m\rho\epsilon^{\frac{1}{2}}. \tag{7}$$

Proof: Since $\bar{x}$ is ϵ-feasible to (P), hence,

$$\sum_{i \in I} p_\epsilon(g_i(\bar{x})) \leq \frac{2}{3}m\epsilon^{\frac{1}{2}}.$$

As x^* is an optimal solution to (P), we have

$$\sum_{i \in I} p(g_i(x^*)) = 0.$$

Then, by Theorem 2.2, we get

$$0 \leq (f(x^*) + \rho \sum_{i \in I} p(g_i(x^*))) - \left(f(\bar{x}) + \rho \sum_{i \in I} p_\epsilon(g_i(\bar{x})) \right) \leq \frac{2}{3} m\rho\epsilon^{\frac{1}{2}}.$$

Thus,

$$\rho \sum_{i \in I} p_\epsilon(g_i(\bar{x})) \leq f(x^*) - f(\bar{x}) \leq \rho \sum_{i \in I} p_\epsilon(g_i(\bar{x})) + \frac{2}{3} m\rho\epsilon^{\frac{1}{2}}.$$

Therefore, $0 \leq f(x^*) - f(\bar{x}) \leq \frac{4}{3} m\rho\epsilon^{\frac{1}{2}}$. □

Definition 2.2. For $x^* \in R^n$, we call $y^* \in R^m$ a Lagrange multiplier vector corresponding to x^* if x^* and y^* satisfy

$$\nabla f(x^*) = -\sum_{i \in I} y_i^* \nabla g_i(x^*), \tag{8}$$

$$y_i^* g_i(x^*) = 0, \quad y_i^* \geq 0, \quad g_i(x^*) \leq 0, \quad i = 1, 2, \ldots, m. \tag{9}$$

Theorem 2.4. *Let f and g_i, $i = 1, 2, \ldots, m$, be convex. Let x^* be an optimal solution of (P) and $y^* \in R^m$ a Lagrange multiplier vector corresponding to x^*. Then*

$$F(x^*, \rho) - F(x, \rho, \epsilon) \leq \frac{2}{3} m\rho\epsilon^{\frac{1}{2}}, \quad \forall x \in R^n, \tag{10}$$

provided that $\rho \geq m\lambda\sqrt{b(x)}$, where $\lambda = \max\{y_i^*, i = 1, 2, \ldots, m\}$ and $b(x) =$ $\max\{g_i^+(x),\ i = 1, 2, \ldots, m\}$.

Proof: By the convexity of f and g_i, $i = 1, 2, \ldots, m$, we have

$$\begin{aligned} f(x) &\geq f(x^*) + \nabla f(x^*)^T (x - x^*), \quad \forall x \in R^n, \\ g_i(x) &\geq g_i(x^*) + \nabla g_i(x^*)^T (x - x^*), \quad \forall x \in R^n. \end{aligned} \tag{11}$$

Since x^* is an optimal solution of (P) and y^* a Lagrange multiplier vector corresponding to x^*, hence, (8) and (9) follows. Applying (8), (9) and (11), we obtain

$$\begin{aligned} f(x) &\geq f(x^*) + \nabla f(x^*)^T (x - x^*) \\ &= f(x^*) - \sum_{i \in I} y_i^* \nabla g_i(x^*)^T (x - x^*) \end{aligned}$$

$$\geq f(x^*) - \sum_{i \in I} y_i^*(g_i(x) - g_i(x^*))$$
$$= f(x^*) - \sum_{i \in I} y_i^* g_i(x).$$

Let $I^+(x) = \{i \in I \mid g_i(x) > 0\}$. Then,

$$f(x) \geq f(x^*) - \sum_{i \in I^+(x)} y_i^* g_i(x). \tag{12}$$

Let

$$\lambda = \max\{y_i^*, i = 1, 2 \ldots, m\} \text{ and } b(x) = \max\{g_i^+(x), i = 1, 2, \ldots, m\}. \tag{13}$$

Then, $-y_i^* g_i(x) \geq -\lambda b(x)$ for any $i \in I^+(x)$. Thus,

$$f(x) \geq f(x^*) - \sum_{i \in I^+(x)} y_i^* g_i(x) \geq f(x^*) - \lambda m b(x). \tag{14}$$

Therefore,

$$\begin{aligned} F(x, \rho) &= f(x) + \sum_{i \in I^+(x)} \rho \sqrt{g_i^+(x)} \\ &\geq f(x^*) - \lambda m b(x) + \sum_{i \in I^+(x)} \rho \sqrt{g_i^+(x)} \\ &\geq f(x^*) - \lambda m b(x) + \rho \sqrt{b(x)}. \end{aligned}$$

When $\rho \geq \lambda m \sqrt{b(x)}$, we obtain $F(x, \rho) \geq f(x^*)$. So, when $\rho \geq \lambda m \sqrt{b(x)}$, we always have $f(x^*) - F(x, \rho) \leq 0$. By Lemma 2.1, we obtain (10). □

As a corollary of Theorem 2.4, we have

Corollary 2.1. *Let f and g_i, $i = 1, 2, \ldots, m$, be convex, x^* an optimal solution of (P), and $y^* \in R^m$ a Lagrange multiplier vector corresponding to x^*. If x_ρ^* is an optimal solution of (P_ρ), then $f(x^*) = F(x_\rho^*, \rho)$ when $\rho \geq \lambda m \sqrt{b^*}$, where $\lambda = \max\{y_i^*,\ i = 1, \ldots, m\}$, $b^* = \max\{g_i^+(x_\rho^*),\ i = 1, 2, \ldots, m\}$.*

Theorems 2.1 and 2.2 mean that an approximate solution to (PI_ρ) is also an approximate solution to (P_ρ) when the error ε is sufficiently small. Furthermore, by Theorem 2.3, an approximately optimal solution to (PI_ρ) becomes an approximately optimal solution to (P) if the solution to (PI_ρ) is $\epsilon-$feasible. Especially, by Theorem 2.4, under some conditions, an approximately optimal solution to (PI_ρ) is an approximately optimal solution to (P). Therefore, we may obtain an approximately optimal solution to (P) by computing an approximately optimal solution to (PI_ρ).

As follows, we give a penalty function algorithm for the problem (P). In order to solve (P), we attempt to solve its smoothed penalty problem given by

$$\min_{x\in R^n} F(x,\rho,\epsilon).$$

For $x \in R^n$, we define

$$\begin{aligned}
I^0(x) &= \{i \mid g_i(x) = 0, i \in I\},\\
I_\epsilon^+(x) &= \{i \mid g_i(x) \geq \epsilon, i \in I\},\\
I_\epsilon^-(x) &= \{i \mid g_i(x) < \epsilon, i \in I\}.
\end{aligned}$$

We propose the following algorithm to solve (P).

Algorithm I.

Step 1: *Given* x^0, $\epsilon > 0$, $\epsilon_0 > 0$, $\rho_0 > 0$, $0 < \eta < 1$, *and* $N > 1$, *let* $j = 0$ *and go to Step 2.*

Step 2: *Use* x^j *as the starting point to solve* $\min_{x\in R^n} F(x,\rho_j,\epsilon_j)$. *Let* x^{j+1} *be the optimal solution obtained.*

Step 3: *If* x^{j+1} *is* ϵ*-feasible to (P), then stop and we have obtained an approximate solution* x^{j+1} *of (P). Otherwise, let* $\rho_{j+1} = N\rho_j$, $\epsilon_{j+1} = \eta\epsilon_j$, *and* $j = j+1$, *and go to Step 2.*

Remark. Since $0 < \eta < 1$ and $N > 1$, hence, as $j \to +\infty$, the sequence $\{\epsilon_j\}$ is decreasing to 0 and the sequence $\{\rho_j\}$ is increasing to $+\infty$.

Theorem 2.5. *Assume that* $\lim_{\|x\|\to\infty} f(x) = +\infty$. *Let* $\{x^j\}$ *be the sequence generated by Algorithm I. Suppose that the sequence* $\{F(x^j,\rho_j,\epsilon_j)\}$ *is bounded. Then* $\{x^j\}$ *is bounded and any limit point* x^* *of* $\{x^j\}$ *belongs to* X_0 *and satisfies*

$$\lambda\nabla f(x^*) + \sum_{i\in I^0(x^*)} \mu_i \nabla g_i(x^*) = 0, \tag{15}$$

where $\lambda \geq 0$ *and* $\mu_i \geq 0$, $i = 1,2,\ldots,m$.

Proof: By the assumptions, there is some number L such that

$$L > F(x^j,\rho_j,\epsilon_j), \quad j = 0,1,2,\ldots. \tag{16}$$

Suppose to the contrary that $\{x^j\}$ is unbounded. Assume without loss of generality that $\|x^j\| \to \infty$ as $j \to +\infty$. Then, from (16), we get

$$L > f(x^j), \quad j = 0,1,2,\ldots,$$

which results in a contradiction since $\lim_{\|x\|\to\infty} f(x) = +\infty$.

We show next that any limit point of $\{x^j\}$ belongs to X_0. Without loss of generality, we assume $\lim_{j\to\infty} x^j = x^*$. Suppose to the contrary that $x^* \notin X_0$. Then there exists some i such that $p(g_i(x^*)) > 0$. As g_i $(i \in I)$ are continuous, so are $F(x^j, \rho_j, \epsilon_j)$ $(j = 1, 2, \ldots)$. Note that

$$\begin{aligned} F(x^j, \rho_j, \epsilon_j) = \; & f(x^j) + \rho_j \sum_{i\in I^+_{\epsilon_j}(x^j)} \left(g_i^+(x^j)^{1/2} - \frac{2}{3}\epsilon_j^{1/2} \right) \\ & + \rho_j \sum_{i\in I^-_{\epsilon_j}(x^j)} \frac{1}{3}\epsilon_j^{-1} g_i^+(x^j)^{3/2}. \end{aligned} \tag{17}$$

Then, as $j \to \infty$, $F(x^j, \rho_j, \epsilon_j) \to \infty$, which contradicts the assumption.

Finally, we show that (15) holds. By Step 2, $\nabla F(x^j, \rho_j, \epsilon_j) = 0$, that is

$$\begin{aligned} \nabla f(x^j) + \; & \rho_j \sum_{i\in I^+_{\epsilon_j}(x^j)} \frac{1}{2} g_i^+(x^j)^{-1/2} \nabla g_i(x^j) \\ + \; & \rho_j \sum_{i\in I^-_{\epsilon_j}(x^j)} \frac{1}{2}\epsilon_j^{-1} g_i^+(x^j)^{1/2} \nabla g_i(x^j) = 0. \end{aligned} \tag{18}$$

For $j = 1, 2, \ldots,$ let

$$\gamma_j = 1 + \sum_{i\in I^+_{\epsilon_j}(x^j)} \rho_j \frac{1}{2} g_i^+(x^j)^{-1/2} + \sum_{i\in I^-_{\epsilon_j}(x^j)} \frac{1}{2}\rho_j \epsilon_j^{-1} g_i^+(x^j)^{1/2}.$$

Then $\gamma_j > 0$. From (18), we have

$$\begin{aligned} \frac{1}{\gamma_j}\nabla f_0(x^j) \; & + \sum_{i\in I^+_{\epsilon_j}(x^j)} \frac{\rho_j \frac{1}{2} g_i^+(x^j)^{-1/2}}{\gamma_j} \nabla g_i(x^j) \\ & + \sum_{i\in I^-_{\epsilon_j}(x^j)} \frac{\frac{1}{2}\rho_j \epsilon_j^{-1} g_i^+(x^j)^{1/2}}{\gamma_j} \nabla g_i(x^j) = 0. \end{aligned} \tag{19}$$

Let

$$\begin{aligned} \lambda^j &= \frac{1}{\gamma_j}, \\ \mu_i^j &= \frac{\rho_j \frac{1}{2}\epsilon_j^{-1} g_i^+(x^j)^{-1/2}}{\gamma_j}, \quad i \in I^+_{\epsilon_j}(x^j), \\ \mu_i^j &= \frac{\frac{1}{2}\rho_j \epsilon_j^{-1} g_i^+(x^j)^{1/2}}{\gamma_j}, \quad i \in I^-_{\epsilon_j}(x^j), \\ \mu_i^j &= 0, \quad i \in I \Big\backslash \left(I^+_{\epsilon_j}(x^j) \bigcup I^-_{\epsilon_j}(x^j) \right). \end{aligned}$$

Then,

$$\begin{aligned} \lambda^j &+ \sum_{i\in I} \mu_i^j = 1, \quad \forall j, \\ \mu_i^j &\geq 0,\ i \in I, \quad \forall j. \end{aligned} \tag{20}$$

Clearly, as $j \to \infty$, $\lambda^j \to \lambda \geq 0$, $\mu_i^j \to \mu_i \geq 0$, $\forall i \in I$. By (19) and (20), as $j \to +\infty$, we have

$$\begin{aligned} \lambda \nabla f_0(x^*) &+ \sum_{i\in I} \mu_i \nabla g_i(x^*) = 0, \\ \lambda &+ \sum_{i\in I} \mu_i = 1. \end{aligned}$$

For $i \in I^-(x^*)$, as $j \to +\infty$, we get $\mu_i^j \to 0$. Therefore, $\mu_i = 0$, $\forall i \in I^-(x^*)$. So, (15) holds. □

Theorem 2.5 points out that the sequence $\{x^j\}$ generated by Algorithm I may converge to a K-T point to (P) under some conditions. The speed of convergence of Algorithm I depends on the speed of convergence of the algorithm employed in Step 2 to solve the unconstrained optimization problem $\min_{x\in R^n} F(x, \rho_j, \epsilon_j)$. Note that $F(x, \rho_j, \epsilon_j)$ is only first-order differentiable. If we want to use a Newton-type method, a smooth penalty function must be second-order differentiable. In the next section, we present a method for smoothing the penalty function (3) to obtain a twice continuously differentiable penalty function.

3. A second-order differentiable smoothing method

In this section, we propose a method for smoothing the penalty function (3) to obtain a twice continuously differentiable penalty function. We define $q_\epsilon(t)$ by

$$q_\epsilon(t) = \begin{cases} 0 & \text{if} \quad t \leq 0, \\ \dfrac{1}{15\epsilon^2} t^{\frac{5}{2}} & \text{if} \quad 0 \leq t < \epsilon, \\ t^{\frac{1}{2}} + \dfrac{2}{3}\epsilon t^{-\frac{1}{2}} - \dfrac{8}{5}\epsilon^{\frac{1}{2}} & \text{if} \quad \epsilon \leq t. \end{cases}$$

It is easy to see that $q_\epsilon(t)$ is twice continuously differentiable and that $\lim_{\epsilon\to 0} q_\epsilon(t) = p(t)$.

Assume that f and g_i, $i \in I$, are twice continuously differentiable. Consider

$$G(x, \rho, \epsilon) = f(x) + \rho \sum_{i\in I} q_\epsilon(g_i(x)), \tag{21}$$

where $\rho > 0$. Clearly, $G(x, \rho, \epsilon)$ is twice continuously differentiable at any $x \in R^n$. Applying $G(x, \rho, \epsilon)$, we obtain the following penalty problem

$$(\mathrm{PII}_\rho): \qquad \min G(x, \rho, \epsilon) \quad s.t. \quad x \in R^n.$$

Since $\lim_{\epsilon \to 0} G(x, \rho, \epsilon) = F(x, \rho)$, we will first consider the relationships between (P_ρ) and (PII_ρ).

Lemma 3.1. *For any $x \in R^n$ and $\epsilon > 0$, we have*

$$0 \le F(x, \rho) - G(x, \rho, \epsilon) \le \frac{8}{5} m\rho\epsilon^{\frac{1}{2}}. \tag{22}$$

Proof: From the definition of $q_\epsilon(t)$, we obtain

$$0 \le p(t) - q_\epsilon(t) \le \frac{8}{5}\epsilon^{\frac{1}{2}}.$$

Then,

$$0 \le p(g_i(x)) - q_\epsilon(g_i(x)) \le \frac{8}{5}\epsilon^{\frac{1}{2}}, \quad \forall x \in R^n,\ i = 1, 2, \ldots, m.$$

Thus,

$$0 \le \sum_{i \in I} p(g_i(x)) - \sum_{i \in I} q_\epsilon(g_i(x)) \le \frac{8}{5} m\epsilon^{\frac{1}{2}}.$$

Therefore,

$$0 \le F(x, \rho) - G(x, \rho, \epsilon) \le \frac{8}{5} m\rho\epsilon^{\frac{1}{2}}.$$

□

As a direct result of Lemma 3.1, we have the following two theorems.

Theorem 3.1. *Let $\{\varepsilon_j\} \to 0$ be a sequence of positive numbers and assume that x^j is a solution to $\min_{x \in R^n} G(x, \rho, \varepsilon_j)$ for some $\rho > 0$. Let $\bar{x}$ be an accumulating point of the sequence $\{x^j\}$. Then $\bar{x}$ is an optimal solution to $\min_{x \in R^n} F(x, \rho)$.*

Theorem 3.2. *Let x^* be an optimal solution of (P_ρ) and $\bar{x} \in R^n$ an optimal solution of (PII_ρ). Then,*

$$0 \le F(x^*, \rho) - G(\bar{x}, \rho, \epsilon) \le \frac{8}{5} m\rho\epsilon^{\frac{1}{2}}. \tag{23}$$

Theorem 3.3. *Let x^* be an optimal solution of (P_ρ) and $\bar{x} \in R^n$ an optimal solution of (PII_ρ). Furthermore, let x^* be feasible to (P) and $\bar{x}$ ϵ-feasible to (P). Then,*

$$0 \le f(x^*) - f(\bar{x}) \le \frac{16}{5} m\rho\epsilon^{\frac{1}{2}}. \tag{24}$$

Proof: Since $\bar{x}$ is ϵ-feasible to (P), hence,

$$\sum_{i\in I} q_\epsilon(g_i(\bar{x})) \leq \frac{8}{5} m\epsilon^{\frac{1}{2}}.$$

As x^* is an optimal solution to (P), we have

$$\sum_{i\in I} p(g_i(x^*)) = 0.$$

By Lemma 3.1, we get

$$0 \leq f(x^*) + \rho \sum_{i\in I} p(g_i(x^*)) - \left(f(\bar{x}) + \rho \sum_{i\in I} q_\epsilon(g_i(\bar{x})) \right) \leq \frac{8}{5} m\rho\epsilon^{\frac{1}{2}},$$

which implies $0 \leq f(x^*) - f(\bar{x}) \leq \frac{16}{5} m\rho\epsilon^{\frac{1}{2}}$. □

Theorem 3.4. *Let f and g_i, $i = 1, 2, \ldots, m$, be convex. Let x^* be an optimal solution of (P) and $y^* \in R^m$ a Lagrange multiplier vector corresponding to x^*. Then,*

$$F(x^*, \rho) - G(x, \rho, \epsilon) \leq \frac{8}{5} m\rho\epsilon^{\frac{1}{2}}, \qquad \forall x \in R^n, \tag{25}$$

provided that $\rho \geq m\lambda\sqrt{b(x)}$, where $\lambda = \max\{y_i^*, i = 1, 2, \ldots, m\}$ and $b(x) = \max\{g_i^+(x), i = 1, 2, \ldots, m\}$.

Proof: The proof is the same as that of Theorem 2.4. □

Now, we present a penalty function algorithm to solve (P). In order to solve (P), we attempt to solve its smoothed penalty problem given by $\min_{x\in R^n} G(x, \rho, \epsilon)$ with

$$\begin{aligned} G(x, \rho, \epsilon) = f(x) + \rho \sum_{i\in I_\epsilon^-(x)} \frac{1}{15\epsilon^2} g_i(x)^{\frac{5}{2}} \\ + \rho \sum_{i\in I_\epsilon^+(x)} \left(g_i(x)^{\frac{1}{2}} + \frac{2}{3}\epsilon g_i(x)^{-\frac{1}{2}} - \frac{8}{5}\epsilon^{\frac{1}{2}} \right). \end{aligned} \tag{26}$$

We propose the following algorithm.

Algorithm II.

Step 1: *Given* $x^0,\ \epsilon > 0,\ \epsilon_0 > 0,\ \rho_0 > 0,\ 0 < \eta < 1$ *and* $N > 1$, *let* $j = 0$ *and go to Step 1.*

Step 2: *Use* x^j *as the starting point to solve* $\min_{x \in R^n} G(x, \rho_j, \epsilon_j)$. *Let* x^{j+1} *be the optimal solution obtained.*

Step 3: *If* x^{j+1} *is* ϵ*-feasible to (P), then stop and the algorithm has generated an approximate solution* x^{j+1} *of (P). Otherwise, let* $\rho_{j+1} = N\rho_j$, $\epsilon_{j+1} = \eta\epsilon_j$, *and* $j = j + 1$, *and go to Step 2.*

Theorem 3.5. *Assume that* $\lim_{\|x\| \to \infty} f(x) = +\infty$. *Let* $\{x^j\}$ *be the sequence generated by Algorithm II. Suppose that the sequence* $\{G(x^j, \rho_j, \epsilon_j)\}$ *is bounded and* $\|\nabla G(x^j, \rho_j, \epsilon_j)\| = 0,\ j = 1, 2, \ldots$. *Then,* $\{x^j\}$ *is bounded, and any limit point* x^* *of* $\{x^j\}$ *belongs to* X_0 *and satisfies*

$$\lambda \nabla f(x^*) + \sum_{i \in I^0(x^*)} \mu_i \nabla g_i(x^*) = 0, \tag{27}$$

where $\lambda \geq 0$ *and* $\mu_i \geq 0, i = 1, 2, \ldots, m$.

Proof: The proof is similar to that of Theorem 2.5. □

Theorems 3.2, 3.3 and 3.4 mean that we may get an approximate solution to (P) by solving (PIIρ). In the next section, we use Algorithms I and II to compute an approximate solution to (P).

4. Numerical results

In this section, we solve some constrained optimization problems with Algorithms I and II on Matlab. Numerical results show that Algorithms I and II yield some approximate solutions that have a better objective function value in comparison with some other algorithms. For a larger value of the penalty parameter ρ, Algorithms I and II have better stability and convergence near a solution.

In order to compare the efficiency of Algorithm I and Algorithm II with the algorithms based on the penalty function (1) and the exact penalty function (2), we use all of them to solve the same examples.

The algorithms based on the penalty function (1) and the exact penalty function (2) are described as follows:

Algorithm III (or IV).

Step 1: *Given* $x^0,\ \epsilon > 0,\ \rho_0 > 0$, *and* $N > 1$, *let* $j = 0$ *and go to Step 2.*

Step 2: *Use* x^j *as the starting point to solve*

$$\min_{x \in R^n} F_1(x, \rho_j) = f(x) + \rho_j \sum_{i=1}^{m} \max\{g_i(x), 0\}$$

Table 1. Results of Algorithm I, II, III and IV with $\rho_0 = 1$ and $N = 2$.

Penalty function	No. iterations	Penalty parameter ρ_j	Objective value	ϵ-solution (x_1, x_2)
Algorithm I	1	1	0.000000	(0.000120, 0.000392)
$F(x, \rho, \epsilon)$	2	2	0.000000	(0.000000, 0.000000)
Algorithm II	1	1	0.000000	(0.000000, 0.000000)
$G(x, \rho, \epsilon)$	2	2	0.000000	(0.000000, 0.000000)
Algorithm III	1	1	0.000000	(0.000028, 0.000023)
$F_1(x, \rho)$	2	2	0.000000	(0.000000, 0.000000)
Algorithm IV	1	1	0.000000	(0.000004, 0.000012)
$F_2(x, \rho)$	2	2	0.000000	(0.000004, 0.000004)
	3	4	0.000000	(0.000000, 0.000000)

(or $\min_{x\in X} F_2(x, \rho_j) = f(x) + \rho_j \sum_{i=1}^m \max\{g_i(x), 0\}^2$*). Let* x^{j+1} *be the optimal solution obtained.*

Step 3: *If* x^{j+1} *is* ϵ*-feasible to (P), then stop and the algorithm has generated an approximate solution* x^{j+1} *of (P). Otherwise, let* $\rho_{j+1} = N\rho_j$ *and* $j = j + 1$*, and go to Step 2.*

For the j'th iteration of the algorithm, we define a constraint error $e_j = e(j)$ by

$$e(j) = \sum_{i=1}^{m} \max\{g(x^j), 0\}.$$

It is clear that x^j is ϵ-feasible to (P) when $e(j) < \epsilon$.

Example 4.1. Consider

$$(P4.1) \qquad \begin{aligned} \min\ & x_1^2 + x_2^2 \\ \text{s.t.}\ & x_1^2 - x_2 \le 0, \quad -x_1 \le 0. \end{aligned}$$

The optimal solution to (P4.1) is given by $(x_1, x_2) = (0, 0)$. Let $x^0 = (4, 4)$, $\epsilon_0 = 1$, $\rho_0 = 1$, $N = 2$, $\eta = 0.5$, $\rho_{j+1} = 2\rho_j$, and $\epsilon_{j+1} = 0.5\epsilon_j$. We choose $\epsilon = 0.0000001$ for ϵ-feasibility. Numerical results for (P4.1) are given in Table 1. The results show that the convergence of four penalty functions is more or less the same and that the convergence of the penalty function $F_2(x, \rho)$ is not good.

Example 4.2. Consider the Rosen-Suzki problem given in [4],

$$(P4.2) \quad \begin{aligned} \min\ & f(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4 \\ \text{s.t.}\ & g_1(x) = 2x_1^2 + x_2^2 + x_3^2 + 2x_1 + x_2 + x_4 - 5 \le 0, \\ & g_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \le 0, \\ & g_3(x) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10 \le 0. \end{aligned}$$

Table 2. Results of Algorithm I.

j	ρ_j	ϵ_j	$f(x^j)$	(x_1, x_2, x_3, x_4)	$g_1(x^j)$	$g_2(x^j)$	$g_3(x^j)$
1	1	1	−77.662039	(1.779734, 1.862917, 4.630400, −2.778546)	28.889815	35.124604	37.988430
2	10	0.5	−44.315201	(0.169629, 0.834436, 2.012455, −0.971824)	0.005678	0.038947	−1.837607
3	100	0.25	−44.234019	(0.168545, 0.836585, 2.008670, −0.965062)	0.000057	0.000074	−1.877881
4	1000	0.125	−44.233582	(0.169234, 0.835656, 2.008690, −0.964901)	−0.000341	0.000000	−1.88214

Let $x^0 = (0, 0, 0, 0)$, $\epsilon = 10^{-6}$, $\epsilon_0 = 1$, $\rho_0 = 1$, $\eta = 0.5$, and $N = 10$. We use Algorithm I to solve (P4.2). Numerical results are given in Table 2.

Therefore, we get an approximate solution

$$x^4 = (0.169234, 0.835656, 2.008690, -0.964901)$$

at the fourth iteration. One can easily check that x^4 is a feasible solution since the constraints of $(P4.2)$ at x^4 are as follows:

$$\begin{aligned}
g_1(x4) &= 2 * 0.028640146756 + 0.698320950336 + 4.0348355161 \\
&\quad + + 2 * 0.169234 + 0.835656 - 0.964901 - 5 \\
&= 4.790436759948 + 1.174124 - 5.964901 = -0.000340240052, \\
g_2(x^4) &= 0.028640146756 + 0.698320950336 + 4.0348355161 \\
&\quad + 0.931033939801 + 0.169234 - 0.835656 + 2.008690 \\
&\quad + 0.964901 - 8 \\
&= 5.692830552993 + 3.142825 - 8.835656 = -0.000000447007 \\
g_3(x^4) &= 0.028640146756 + 2 * 0.698320950336 + 4.0348355161 \\
&\quad + 2 * 0.931033939801 - 0.169234 + 0.964901 - 10 \\
&= 7.32218544313 + 0.964901 - 10.169234 = -1.88214755687.
\end{aligned}$$

The objective function value is given by $f(x^4) = -44.233582$ that is better than the objective function value $f(x') = -44$ at the solution to (P4.2) $x' = (0, 1, 2, -1)$ obtained in [4].

With starting points $x^0 = (5, 5, 5, 5)$, $x^0 = (10, 10, 10, 10)$ and $x^0 = (20, 20, 20, 20)$, numerical results are given in Table 2(a), Table 2(b) and Table 2(c) respectively. One can see that the numerical results in Table 2(a) and Table 2(a)–(c) are almost the same. This means that Algorithm I does not completely depend on how to choose a starting point in this example. Although we choose a starting point $x^0 = (0, 0, 0, 0)$ that is a feasible point, after the first iteration, the algorithm generates $x^1 = (1.779734, 1.862917, 4.630400, -2.778546)$, which is not feasible. Therefore, one can choose any starting point for Algorithm I in this example.

Table 2a. Results of Algorithm I with a starting point $x^0 = (5, 5, 5, 5)$.

j	ρ_j	ϵ_j	$f(x^j)$	(x_1, x_2, x_3, x_4)	$g_1(x^j)$	$g_2(x^j)$	$g_3(x^j)$
1	1	1	−77.662064	(1.779716, 1.862945, 4.630412, −2.778538)	28.889888	35.124655	37.988592
2	10	0.5	−44.315203	(0.169631, 0.834438, 2.012454, −0.971826)	0.005679	0.038948	−1.837597
3	100	0.25	−44.234024	(0.169578, 0.835831, 2.008481, −0.965091)	0.000020	0.000087	−1.881706
4	1000	0.125	−44.233296	(0.169553, 0.835787, 2.008382, −0.965252)	−0.000722	=0.000000	−1.881450

Table 2b. Results of Algorithm I with a starting point $x^0 = (10, 10, 10, 10)$.

j	ρ_j	ϵ_j	$f(x^j)$	(x_1, x_2, x_3, x_4)	$g_1(x^j)$	$g_2(x^j)$	$g_3(x^j)$
1	1	1	−77.662264	(1.779701, 1.862915, 4.630476, −2.778605)	28.890150	35.125618	37.989756
2	10	0.5	−44.315211	(0.169631, 0.834435, 2.012454, −0.971827)	0.005675	0.038953	−1.837598
3	100	0.25	−44.234027	(0.169278, 0.835588, 2.008767, −0.964795)	0.000014	0.000091	−1.882607
4	1000	0.125	−44.233211	(0.169193, 0.835487, 2.008712, −0.964922)	−0.000836	0.000000	=−1.882500

Table 2c. Results of Algorithm I with a starting point $x^0 = (20, 20, 20, 20)$.

j	ρ_j	ϵ_j	$f(x^j)$	(x_1, x_2, x_3, x_4)	$g_1(x^j)$	$g_2(x^j)$	$g_3(x^j)$
1	1	1	−77.662043	(1.779713, 1.862946, 4.630400, −2.778545)	28.889751	35.124574	37.988578
2	10	0.5	−44.315201	(0.169629, 0.834437, 2.012455, −0.971824)	0.005678	0.038947	−1.837603
3	100	0.25	−44.234049	(0.169183, 0.835757, 2.008735, −0.964859)	0.000017	0.000101	−1.881802
4	1000	0.125	−44.233836	(0.169244, 0.835900, 2.008620, −0.964960)	0.000000	0.000000	−1.881331

Let $x^0 = (0, 0, 0, 0)$, $\epsilon = 10^{-6}$, $\epsilon_0 = 1$, $\rho_0 = 1$, $\eta = 0.5$, and $N = 2$. We use Algorithms I, II, III and IV to solve (P4.2). Numerical results are given in Table 3. From the table, one can see that Algorithm I converges faster than Algorithm II. Algorithm IV is the slowest one.

When ρ_0 and N become $\rho_0 = 10$ and $N = 2$, numerical results are given in Table 4. From Table 4, one can see that Algorithms I and II converge faster and have better numerical stability. Especially, the point x^{12} is also feasible to (P4.2) in Table 4. Numerical results in Table 4 show that the exact penalty function $F_1(x, \rho)$ is not numerically stable

Table 3. Results of Algorithm I, II, III and IV with $\rho_0 = 1$ and $N = 2$.

Penalty function	No. iter.	ρ_k	Cons. error $e(k)$	Objective value	ϵ-Solution (x_1, x_2, x_3, x_4)
Algorithm I	1	1	102.002849	−77.662039	(1.779734, 1.862917, 4.630400, −2.778546)
$F(x, \rho, \epsilon)$	2	2	52.440428	−70.152233	(1.138369, 1.271270, 3.818482, −1.996516)
	3	4	0.069425	−44.360100	(0.169559, 0.833940, 2.014588, −0.975638)
	4	8	0.004396	−44.241880	(0.169504, 0.835277, 2.009119, −0.965442)
	7	64	0.000000	−44.233835	(0.170126, 0.835578, 2.008301, −0.965178)
Algorithm II	1	1	103.026548	−77.748562	(1.793572, 1.874673, 4.643953, −2.791637)
$G(x, \rho, \epsilon)$	2	2	54.266418	−70.580637	(1.163216, 1.294403, 3.855829, −2.028104)
	3	4	2.023972	−47.472827	(0.188840, 0.734762, 2.176659, −1.251934)
	4	8	0.123484	−44.425671	(0.172309, 0.837356, 2.018416, −0.974029)
	13	4096	0.000000	−44.233837	(0.169401, 0.835571, 2.008701, −0.964825)
Algorithm III	1	1	3.082668	−48.629509	(0.339654, 0.677748, 2.240736, −1.231420)
$F_1(x, \rho)$	2	2	0.000004	−44.233744	(0.171993, 0.831487, 2.009344, −0.963467)
	3	4	0.000000	−44.233741	(0.171993, 0.831486, 2.009344, −0.963467)
	4	8	0.000000	−44.233741	(0.171993, 0.831486, 2.009344, −0.963467)
Algorithm IV	1	1	1.278325	−46.204979	(0.192648, 0.844548, 2.108774, −1.076979)
$F_2(x, \rho)$	2	2	0.659897	−45.281613	(0.180725, 0.838664, 2.061378, −1.026197)
	3	4	0.335606	−44.775924	(0.174983, 0.836616, 2.035780, −0.997182)
	4	8	0.169287	−44.509867	(0.172223, 0.835930, 2.022415, −0.981500)
	23	4194304	0.000000	−44.233837	(0.169555, 0.835503, 2.008651, −0.964856)

Table 4. Results of Algorithm I, II, III and IV with $\rho_0 = 10$ and $N = 2$.

Penalty function	No. iter.	ρ_k	Cons. error $e(k)$	Objective value	Solution (x_1, x_2, x_3, x_4)
Algorithm I	1	10	0.174189	−44.547243	(0.169737, 0.831692, 2.023480, −0.991311)
$F(x, \rho, \epsilon)$	2	20	0.011168	−44.254312	(0.169496, 0.835219, 2.009648, −0.966585)
	3	40	0.000704	−44.235125	(0.169408, 0.835431, 2.008820, −0.964860)
	4	80	0.000030	−44.233882	(0.169162, 0.835257, 2.008994, −0.964512)
	6	320	0.000000	−44.233835	(0.169174, 0.835217, 2.009005, −0.964490)
Algorithm II	1	10	1.624681	−46.624520	(0.207426, 0.861010, 2.133251, −1.075119)
$G(x, \rho, \epsilon)$	2	20	0.421837	−44.881344	(0.179257, 0.841571, 2.041896, −0.995292)
	3	40	0.106477	−44.399382	(0.171953, 0.836970, 2.017124, −0.972714)
	4	80	0.026691	−44.275467	(0.170095, 0.835921, 2.010785, −0.966842)
	12	20480	0.000000	−44.233834	(0.169568, 0.834758, 2.009021, −0.964387)
Algorithm III	1	10	0.042087	−43.810366	(0.447942, 0.726313, 1.874833, −1.066028)
$F_1(x, \rho)$	2	20	0.003425	−44.061588	(0.338032, 0.787990, 1.926692, −1.026223)
	3	40	Inf	NaN	(Inf, Inf, Inf, Inf)
Algorithm IV	1	10	0.135674	−44.455497	(0.171726, 0.835805, 2.019679, −0.978264)
$F_2(x, \rho)$	2	20	0.068081	−44.345510	(0.170614, 0.835655, 2.014196, −0.971650)
	3	40	0.034102	−44.289889	(0.170085, 0.835586, 2.011424, −0.968284)
	4	80	0.017067	−44.261918	(0.169821, 0.835560, 2.010031, −0.966587)
	19	2621440	0.000000	−44.233837	(0.169537, 0.835486, 2.008670, −0.964836)

Table 5. Results of Algorithm I, II, III and IV with $\rho_0 = 100$ and $N = 10$.

Penalty function	No. iter.	ρ_k	Cons. error $e(k)$	Objective value	Solution (x_1, x_2, x_3, x_4)
Algorithm I	1	100	0.002760	944.210629	(2.500212, 4.217963, 0.979881)
$F(x, \rho, \epsilon)$	4	100000	0.000000	944.215662	(2.500000, 4.220720, 0.967224)
Algorithm II	1	100	0.296528	943.655185	(2.521627, 4.237162, 0.971728)
$G(x, \rho, \epsilon)$	7	100000000	0.000000	944.215654	(2.500000, 4.221049, 0.965786)
Algorithm III	1	100	0.026443	947.775367	(2.501592, 3.469350, 2.593613)
$F_1(x, \rho)$	5	1000000	0.000000	947.541190	(2.500000, 3.514028, 2.530140)
Algorithm IV	1	100	0.011477	944.197815	(2.501028, 4.206208, 1.031285)
$F_2(x, \rho)$	6	10000000	0.000000	944.215652	(2.500000, 4.221305, 0.964666)

when the penalty parameter ρ becomes larger. Algorithm IV still converges very slow.

Example 4.3. Consider (Example 2 in [4])

$$
\begin{aligned}
(P4.3)\qquad \min\ f(x) &= 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \\
\text{s.t.}\ g_1(x) &= x_1^2 + x_2^2 + x_3^2 - 25 = 0, \\
g_2(x) &= (x_1 - 5)^2 + x_2^2 + x_3^2 - 25 = 0, \\
g_3(x) &= (x_1 - 5)^2 + (x_2 - 5)^2 + (x_3 - 5)^2 - 25 \le 0.
\end{aligned}
$$

Let $x^0 = (2, 2, 2)$, $\epsilon = 10^{-6}$, $\epsilon_0 = 1$, $\rho_0 = 100$, $\eta = 0.5$, and $N = 10$. We use Algorithms I, II, III and IV to solve (P4.3). Numerical results are given in Table 5. The fourth ϵ-solution generated by Algorithm I yields a better objective function value than that obtained in [4]. In the same number of iterations, the objective function value at the solution generated by Algorithm III is worse than the objective function value at the solution generated by Algorithms I and II.

In this example, our algorithm has obtained an ϵ-feasible point that has a better objective function value. In many practical applications, an approximate solution is good enough. An ϵ-feasible point may be infeasible, however, when ϵ is sufficiently small, the point is acceptable in many situations.

Example 4.4. Consider (Example 1 in [10])

$$
\begin{aligned}
(P4.4)\qquad \min\ f(x) &= 100x_1 + 120x_2 + 90x_3 + 80x_4 + 70x_5 + 140x_6 \\
&\quad + 40x_7 + 20x_8 + 30x_9 + 20x_{10} + 40x_{11} + 10x_{12} \\
\text{s.t.}\ g_1(x) &= x_1 + x_2 + x_3 - 25 = 0, \\
g_2(x) &= x_4 + x_5 + x_6 - 15 = 0, \\
g_3(x) &= x_1 + x_4 - 20 = 0, \\
g_4(x) &= x_2 + x_5 - 10 = 0, \\
g_5(x) &= x_3 + x_6 - 10 = 0,
\end{aligned}
$$

$$g_6(x) = x_7 + x_8 + x_9 - 50 = 0,$$
$$g_7(x) = x_{10} + x_{11} + x_{12} - 30 = 0,$$
$$g_8(x) = x_7 + x_{10} - 20 = 0,$$
$$g_9(x) = x_8 + x_{11} - 40 = 0,$$
$$g_{10}(x) = x_9 + x_{12} - 20 = 0,$$
$$g_{11}(x) = x_1 + x_7 - 30 \leq 0,$$
$$g_{12}(x) = x_3 + x_9 - 30 \leq 0,$$
$$0 \leq x_i \leq 75, \; i = 1, 2, \ldots, 12.$$

Let $x^0 = (0, 0, \ldots, 0)$, $\epsilon = 10^{-6}$, $\epsilon_0 = 0.1$, $\rho_0 = 1000$, $\eta = 0.01$ and $N = 2$. We use Algorithms I–IV to solve (P4.4). Numerical results are given in Table 6. The solution generated by Algorithm I at the second iteration (the objective function value $f(x^*) = 5100.001160$) is much better than that obtained in [10] (the objective function value $f(x^*) = 7100$).

Example 4.5. Consider (Example 2 in [10])

$$(P4.5) \quad \min \; f(x) = 10x_2 + 2x_3 + x_4 + 3x_5 + 4x_6$$
$$\text{s.t.} \; g_1(x) = x_1 + x_2 - 10 = 0,$$
$$g_2(x) = -x_1 + x_3 + x_4 + x_5 = 0,$$
$$g_3(x) = -x_2 - x_3 + x_5 + x_6 = 0,$$
$$g_4(x) = 10x_1 - 2x_3 + 3x_4 - 2x_5 - 16 \leq 0,$$
$$g_5(x) = x_1 + 4x_3 + x_5 - 10 \leq 0,$$
$$0 \leq x_1 \leq 12,$$
$$0 \leq x_2 \leq 18,$$
$$0 \leq x_3 \leq 5,$$
$$0 \leq x_4 \leq 12,$$
$$0 \leq x_5 \leq 1,$$
$$0 \leq x_6 \leq 16.$$

Let $x^0 = (0, 0, \ldots, 0)$, $\epsilon = 10^{-6}$, $\epsilon_0 = 0.1$, $\rho_0 = 1000$, $\eta = 0.01$ and $N = 2$. We use Algorithm I, II, III and IV to solve (P4.5). Numerical results are given in Table 7. The solution generated by Algorithm I at the second iteration (the objective function value $f(x^*) = 117.038781$) is much better than that obtained in [10] (the objective function value $f(x^*) = 124$). From Table 7, one can see that the penalty functions $F(x, \rho, \epsilon)$ and $G(x, \rho, \epsilon)$ yield a better convergence result than $F_1(x, \rho)$ and $F_2(x, \rho)$.

Table 6. Results of Algorithm I, II, III and IV with $\rho_0 = 1000$ and $N = 2$.

Penalty funct.	No. it.	ρ_k	$e(k)$	Obj. value	ϵ-solution $(x_1, x_2, \ldots, x_{12})$
Algorithm I	1	1000	0.004428	5099.962388	(14.999825, −0.000359, 9.999590, 4.999557, 10.000391, 0.000321, 4.965522, 39.999817, 5.034382, 15.034300, 0.000753, 14.965827)
$F(x, \rho, \epsilon)$	2	2000	0.000000	5100.001160	(14.999961, 0.000039, 10.000000, 5.000039, 9.999961, 0.000000, 4.965876, 40.000000, 5.034124, 15.034124, 0.000000, 14.965876)
Algorithm II	1	1000	0.179645	5092.793195	(14.977732, −0.014815, 10.012552, 5.005587, 10.001325, −0.026010, 4.981189, 40.020414, 4.986428, 15.008250, −0.017924, 15.007315)
$G(x, \rho, \epsilon)$	4	8000	0.000000	5100.000794	(15.000007, 0.000002, 9.999991, 4.999993, 9.999998, 0.000009, 6.621689, 39.999998, 3.378312, 13.378311, 0.000002, 16.621687)
Algorithm III	1	1000	0.000251	5117.691391	(14.415970, 0.583947, 10.000084, 5.584016, 9.415987, 0.000003, 4.936768, 39.995673, 5.067566, 15.063255, 0.004296, 14.932438)
$F_1(x, \rho)$	3	4000	0.000000	5100.000531	(14.999996, 0.000008, 9.999997, 5.000004, 9.999992, 0.000004, 4.951794, 39.999999, 5.048207, 15.048206, 0.000001, 14.951793)
Algorithm IV	1	1000	0.000251	5117.691391	(14.415970, 0.583947, 10.000084, 5.584016, 9.415987, 0.000003, 4.936768, 39.995673, 5.067566, 15.063255, 0.004296, 14.932438)
$F_2(x, \rho)$	3	4000	0.000000	5100.000531	(14.999996, 0.000008, 9.999997, 5.000004, 9.999992, 0.000004, 4.951794, 39.999999, 5.048207, 15.048206, 0.000001, 14.951793)

Example 4.6. Consider (Page 110 in [13])

$$(P4.6) \quad \min\ f(x) = \sum_{j=1}^{n} x_j \ln x_j$$

$$\text{s.t.} \ \sum_{j=1}^{n} |\sin(jt)| x_j \geq nt, \ \forall t \in T = [0, 1],$$

$$x_j \geq 0, \ j = 1, 2, \ldots, n.$$

(P4.6) is an entropy optimization problem with infinitely many linear constraints given in [13]. We also use the same simple discretization method as that in [13] to solve

Table 7. Results of Algorithm I, II, III and IV with $\rho_0 = 1000$ and $N = 2$.

Penalty funct.	No. it.	ρ_k	$e(k)$	Obj. value	ϵ-solution $(x_1, x_2, x_3, x_4, x_5, x_6)$
Algorithm I	1	1000	7.098851	18.038351	(1.692653, 1.234077, 0.201055, 0.493765, 1.003608, 0.447721)
$F(x, \rho, \epsilon)$	2	2000	0.000000	117.038781	(1.847052, 8.152948, 0.607878, 0.244707, 0.994467, 7.766359)
Algorithm II	1	1000	0.024180	116.802598	(1.907158, 8.083882, 0.750435, 0.147413, 1.005545, 7.824715)
$G(x, \rho, \epsilon)$	3	4000	0.000000	117.010399	(1.805996, 8.194004, 0.497669, 0.308327, 1.000000, 7.691673)
Algorithm III	1	1000	6.812417	25.950527	(1.560720, 1.999444, −0.171976, 0.702805, 1.029784, 0.626970)
$F_1(x, \rho)$	3	4000	0.000000	123.918322	(1.608741, 8.391259, 0.971062, 0.626000, 0.011679, 9.350643)
Algorithm IV	1	1000	6.812417	25.950527	(1.560720, 1.999444, −0.171976, 0.702805, 1.029784, 0.626970)
$F_2(x, \rho)$	3	4000	0.000000	123.918322	(1.608741, 8.391259, 0.971062, 0.626000, 0.011679, 9.350643)

(P4.6). For each problem, we discretize $T = [0, 1]$ into m equal parts, and a constraint is obtained at $t = i/m,\ i =, 1, 2, \ldots, m$. We solve the following problem by Algorithm I.

$$
\begin{aligned}
(P4.6)' \quad & \min\ f(x) = \sum_{j=1}^{n} x_j \ln x_j \\
& \text{s.t. } g_i(x) = (i/m)n - \sum_{j=1}^{n} |\sin(ji/m)| x_j \le 0,\ i = 1, 2, \ldots, m, \\
& \quad x_j \ge 0,\ j = 1, 2, \ldots, n.
\end{aligned}
$$

Let $x^0 = (2, 2, \ldots, 2),\ \epsilon = 10^{-6},\ \rho_0 = 10,\ \eta = 0.5$ and $N = 2$. For several different starting values of ϵ_0, numerical results for Algorithm I are given in Table 8 with $n = 30$ and $m = 30$. For (P4.6)′, we have found that the convergence of the objective function value is slower when the initial value of ϵ_0 is too big or too small. When $\epsilon_0 \in [10, 20]$, the algorithm converges at a good speed. But the convergence of the objective function value is not influenced when the value of the penalty parameter becomes bigger, which is shown in Table 9. Therefore, when a suitable initial value of ϵ_0 is chosen, the algorithm will converge faster to a better $\epsilon-$feasible solution.

Let $x^0 = (2, 2, \ldots, 2),\ \epsilon = 10^{-6},\ \rho_0 = 2,\ \eta = 0.5$ and $N = 2$. For sveral different initial values of ϵ_0, numerical results of Algorithm I are given in Table 10. The results obtained by Algorithm I are almost the same as those obtained in [13], but slightly better than those obtained by Algorithm III.

Table 8. Results of Algorithm I when $n = 30$ and $m = 30$.

No. iter.	ρ_j	ϵ_0	e_j	Objective value
9	2560	40.000000	0.156250	15.283552
10	5120	30.000000	0.058594	15.284887
8	1280	20.000000	0.156250	15.284095
9	2560	15.000000	0.058594	15.283094
8	1280	10.000000	0.078125	15.287858
7	640	5.000000	0.078125	15.283920
4	80	1.000000	0.125000	15.416194
5	160	0.500000	0.031250	15.417025

Table 9. Results of Algorithm I when $\rho = 10$ and $\rho_0 = 10$.

No. iter.	ρ_j	ϵ_0	e_j	Objective value
8	100000000	40.000000	0.312500	17.205912
8	100000000	30.000000	0.234375	15.381366
8	100000000	20.000000	0.156250	15.297681
8	100000000	15.000000	0.117188	15.335761
8	100000000	10.000000	0.078125	15.422016
8	100000000	1.000000	0.007813	15.413923

Table 10. Results of Algorithm I and III when $\rho = 10$ and $\rho_0 = 2$.

n	m	No. iter.	ρ_k	ϵ_0	Objective value of Algorithm I	Objective value of Algorithm III	Objective value in [13]
10	10	8	1280	5	4.718230	4.721694	4.720
10	30	8	1280	5	4.718230	4.720791	4.721
10	100	8	1280	5	4.719137	4.720168	4.715
10	300	8	1280	5	4.721301	4.721330	4.727
10	1000	8	1280	5	4.721945	4.722012	4.708
30	10	8	1280	15	13.954221	14.104101	13.950
30	30	9	2560	15	15.283094	15.328519	15.280
30	100	9	2560	15	15.389501	15.398350	15.393
30	300	8	1280	15	15.443326	15.451099	15.454
30	1000	8	1280	15	15.474361	15.479682	15.467
100	10	14	81920	40	48.341221	48.865000	48.310
100	30	14	163840	40	53.403638	55.205700	53.352
100	100	14	163840	40	58.633121	61.209947	58.453
100	300	14	163840	40	58.553288	61.175242	58.544
100	1000	14	163840	40	58.698932	61.299541	58.696

Table 11. Results of Algorithm I, II and III when m increases.

			Algorithm I			Algorithm II			Algorithm III
m	No. iter.	ρ_k	Objective value	No. iter.	ρ_k	Objective value	No. iter.	ρ_k	Objective value
1	2	200	5.334687	5	1600	5.334721	2	200	5.334688
10	2	200	5.334765	4	800	5.334720	2	200	5.334690
100	2	200	5.334687	5	1600	5.334714	2	200	5.336118
1000	2	200	5.335877	5	1600	5.334716	4	800	7.900587
2000	1	100	5.336072	5	1600	5.334709	5	1600	7.396654

Example 4.7. Consider (an example in [14])

$$(P4.7) \quad \min\ f(x) = x_1^2 + x_2^2 + x_3^2$$
$$\text{s.t. } g(x) = x_1 + x_2 e^{x_3 t} - 2\sin(4t) \le 0,\ t \in [0, 1].$$

In [14], an approximate solution $(-0.2133, -1.3615, 1.8535)$ is given with the objective function value 5.3347. We discretize $T = [0, 1]$ into m equal parts and a constraint is obtained at $t = i/m,\ i =, 1, 2, \ldots, m$.

$$(P4.7)' \quad \min\ f(x) = x_1^2 + x_2^2 + x_3^2$$
$$\text{s.t. } g_i(x) = x_1 + x_2 e^{x_3 \frac{i}{m}} - 2\sin\left(4\frac{i}{m}\right) \le 0,\ i = 1, 2, \ldots, m.$$

Let $x^0 = (1, 1, 1)$, $\epsilon = 10^{-6}$, $\epsilon_0 = 0.1$, $\rho_0 = 100$, $\eta = 0.1$ and $N = 2$. We use Algorithms I, II, and III to solve (P4.7)$'$. Numerical results are given in Table 11. From Table 11, one can see that the penalty functions $F(x, \rho, \epsilon)$ and $G(x, \rho, \epsilon)$ yield some better convergence results than the exact penalty function $F_1(x, \rho)$ when m increases.

Example 4.8. Consider (an example in [15])

$$(P4.8) \quad \min\ f(x) = x_1^2 + (x_2 - 3)^2$$
$$\text{s.t. } g(x) = x_2 - 2 + x_1 \sin\left(\frac{t}{x_2 - 0.5}\right) \le 0,\ t \in [0, 10].$$

In [15], an approximate solution (0,2) is obtained with the objective function value 1. We discretize $T = [0, 10]$ into m equal parts and a constraint is obtained at $t = i/m,\ i =, 1, 2, \ldots, m$.

$$(P4.8)' \quad \min\ f(x) = x_1^2 + (x_2 - 3)^2$$
$$\text{s.t. } g_i(x) = x_2 - 2 + x_1 \sin\left(\frac{10i}{m(x_2 - 0.5)}\right) \le 0,\ i = 1, 2, \ldots, m.$$

Table 12. Results of Algorithm I, II and III when m increases.

	Algorithm I			Algorithm II			Algorithm III		
m	No. iter.	ρ_k	Objective value	No. iter.	ρ_k	Objective value	No. iter.	ρ_k	Objective value
1	1	10000	0.954338	4	80000	0.768697	1	10000	0.768697
10	1	10000	1.000024	4	80000	0.995607	1	10000	0.995607
100	1	10000	1.000057	4	80000	0.999956	1	10000	0.999956
1000	1	10000	1.000057	4	80000	1.000001	1	10000	1.000000
2000	1	10000	1.000057	4	80000	1.000000	1	10000	1.000000

Let $x^0 = (1, 1, 1)$, $\epsilon = 10^{-6}$, $\epsilon_0 = 0.1$, $\rho_0 = 10000$, $\eta = 0.1$ and $N = 2$. We use Algorithms I, II, and III to solve (P4.7)$'$. Numerical results are given in Table 12. From Table 12, one can see that the penalty functions $F(x, \rho, \epsilon)$ and $G(x, \rho, \epsilon)$ yield the same convergence results as the exact penalty function $F_1(x, \rho)$ when m increases.

According to the numerical results given above, one may draw the following conclusions on Algorithm I and Algorithm II: In general, the smoothing penalty functions $F(x, \rho, \epsilon)$ and $G(x, \rho, \epsilon)$ yield some better convergence and stability results for computing an approximate solution to (P) than $F_1(x, \rho)$ and $F_2(x, \rho)$.

Finally, we give some guidances on how to choose parameter in our algorithm. The important parameters in our algorithm are the penalty parameter ρ and the error ϵ. According to our experience, initially ρ_0 may be 1, 2, 5,10,100 or 1000, $N = 2, 5, 10$ or 100, and the iteration formula $\rho := N\rho$. When we choose a big initial value of the penalty parameter ρ_0, the number of iterations may be fewer. It is satisfactory when the initial value ρ_0 is not taken too big. The initial value of the error ϵ_0 may be 10,1,0.5,0.2 or 0.1, $\eta = 0.5, 0.1, 0.05$ or 0.01, and the iteration formula $\epsilon := \eta\epsilon$.

Acknowledgments

The authors would like to thank two anonymous referees for their helpful comments and suggestions. This research was carried out when the first author was visiting City University of Hong Kong, and partially supported by CERG: CityU 101003 of HKSAR.

References

1. W.I. Zangwill, "Nonlinear programming via penalty function," Manangement Science, vol. 13, pp. 334–358, 1967.
2. S.P. Han and O.L. Mangasrian, "Exact penalty function in nonlinear programming," Mathematical Programming, vol. 17, pp. 251–269, 1979.
3. E. Rosenberg, "Globally convergent algorithms for convex programming," Mathematics of Operational Rresearch, vol. 6, pp. 437–443, 1981.
4. J.B. Lasserre, "A globally convergent algorithm for exact penalty functions," European Journal of Opterational Research, vol. 7, pp. 389–395, 1981.
5. F.H. Clarke, Optimization and Nonsmooth Analysis, Wiley, New York, 1982.
6. E. Rosenberg, "Exact penalty functions and stability in locally Lipschitz programming," Mathematical Programming, vol. 30, pp. 340–356, 1984.

7. G. Di Pillo and L. Grippo, “An exact penalty function method with global conergence properties for nonlinear programming problems,” Mathemathical Programming, vol. 36, pp. 1–18, 1986.
8. G. Di Pillo and L. Grippo, “On the exactness of a class of nondifferentiable penalty function,” Journal of Optimization Theory and Applications, vol. 57, pp. 385–406, 1988.
9. S.A. Zenios, M.C. Pinar, and R.S. Dembo, “A smooth penalty function algorithm for network-structured problems,” European Journal of Operational Research, vol. 64, pp. 258–277, 1993.
10. M.C. Pinar and S.A. Zenios, “On smoothing exact penalty functions for convex constarained optimization,” SIAM Journal on Optimization, vol. 4, pp. 486–511, 1994.
11. C. Chen and O.L. Mangasarian, “Smoothing methods for convex inequalities and linear complementarity problems,” Mathematical Programming, vol. 71, pp. 51–69, 1995.
12. X.Q. Yang, Z.Q. Meng, X.X. Huang, and G.T.Y. Pong, “Smoothing nonlinear penalty functions for constrained optimization,” Numerical Functional Analysis and Optimization, vol. 24, pp. 351–364, 2003.
13. S.C. Fang, J.R. Rajasekera, and H.S.J. Tsao, Entropy Optimization and Mathematical Proggramming, Kluwer, 1997.
14. L. Qi, S.Y. Wu, and G. Zhou, “Semismooth newton methods for solving semi-infinite programming problems,” Journal of Global Optimization, vol. 27, pp. 215–232, 2003.
15. K.L. Teo, X.Q. Yang, and L.S. Jennings, “Computational discretization algorithms for functional inequality constrained optimization,” Annals of Operations Research, vol. 98, pp. 215–234, 2000.