AIMS Mathematics, 9(3): 6528–6554.
DOI: 10.3934/math.2024318
Received: 30 October 2023
Revised: 27 December 2023
Accepted: 29 December 2023
Published: 06 February 2024

http://www.aimspress.com/journal/Math

Research article

A nonmonton active interior point trust region algorithm based on CHKS smoothing function for solving nonlinear bilevel programming problems

B. El-Sobky[1,*], Y. Abo-Elnaga[2], G. Ashry[1] and M. Zidan[3]

[1] Department of Mathematics, Faculty of Science, Alexandria University, Alexandria, Egypt
[2] Department of basic science, Tenth of Ramadan City, Higher Technological Institute, Egypt
[3] Department of Physics and Engineering Mathematics, Faculty of Engineering-Tanta University, Egypt

* **Correspondence:** Email: bothina-elsobky@alexu.edu.eg.

Abstract: In this paper, an approach is suggested to solve nonlinear bilevel programming (NBLP) problems. In the suggested method, we convert the NBLP problem into a standard nonlinear programming problem with complementary constraints by applying the Karush-Kuhn-Tucker condition to the lower-level problem. By using the Chen-Harker-Kanzow-Smale (CHKS) smoothing function, the nonlinear programming problem is successively smoothed. A nonmonton active interior-point trust-region algorithm is introduced to solve the smoothed nonlinear programming problem to obtain an approximately optimal solution to the NBLP problem. Results from simulations on several benchmark problems and a real-world case about a watershed trading decision-making problem show how the effectiveness of the suggested approach in NBLP solution development.

Keywords: nonlinear bilevel problem; nonmonton trust-region; CHKS smoothing function; active-set; interior-point
Mathematics Subject Classification: 49N35, 49N10, 93D52, 93D22, 65K05

1. Introduction

A nested optimization problem with two levels in a hierarchy, i.e the upper-level and lower-level decision-making, is known as the bilevel programming problem. Each level has distinct constraints and objective functions. Both of them have their objective functions and constraints. The decision-maker at the upper level always takes the lead, followed by those at the lower level. The objective function and constraint of the upper-level programming depend on their decision variables and the optimum solution of the lower-level programming. The decision maker at the lower level must maximize its objective function by using the variables provided by the decision maker at the upper level, who in

turn chooses the variables after having full knowledge of the lower level's potential responses. Bilevel mathematical programming stresses the system's non-cooperative nature, in contrast to many objective mathematical programming methodologies. This hierarchical model has several applications, including resource allocation, decentralized control, and network design issues. The successful application of this hierarchical model depends on how well it is solved when handling realistic complications, etc. [3]. Bilevel mathematical programming has attracted a lot of attention, and many effective algorithms have been presented. There are currently several methods for solving NBLP problems and they can be divided into four categories: the Karush-Kuhn-Tucker condition approach [1,15,17,19,21], the penalty function approach [24, 29], the descent approach [22, 36], and the evolutionary approach [41].

In this paper, a class of NBLP problems is reduced to a traditional nonlinear programming problem with complementary constraints. The lower-level problem is then substituted by its Karush-Kuhn-Tucker optimality conditions. The CHKS smoothing function is then used to smooth it.

The smoothed nonlinear programming problem (SNLP) is solved by using the nonmonton active interior-point trust-region technique to obtain an approximately optimal solution to the nonlinear bilevel programming problem. In the nonmonton active interior-point trust-region technique, the smoothed nonlinear programming problem is transformed into an equality-constrained optimization problem with bounded variables by using an active-set approach and the penalty method; for more details see [12–15, 18, 19]. To solve the equality- constrained optimization problem with bounded variables, Newton's interior-point approach [6] is used. Because Newton's interior-point method is a local method, it might not converge if the starting point is far from a stationary point. A trust-region strategy is used to treat this problem and ensure convergence to the stationary point from any starting point. A trust-region technique can induce strong global convergence and it is a very important method for solving unconstrained and constrained optimization problems; see [10–14, 16–19]. One advantage of the trust-region technique is that it does not require the models objective function to be convex. However, in the traditional trust-region strategy, we must use some criteria to determine whether the trial step is acceptable after solving a trust-region subproblem. Otherwise, the trust-region radius needs to be reduced. A method for calculating the trust-region radius Δ_k at each iteration is an important part of trust-region techniques. The standard trust-region strategy is predicated on the objective function and the model agreement. The trust region's radius is updated by paying attention to the ratio $t_k = \frac{Ared_k}{Pred_k}$ where $Ared_k$ refers to the actual reduction and $Pred_k$ refers to the predicted reduction. It can be deduced that whenever t_k is close to 1, there will be a good agreement between the model and the objective function over a current trust region. It is well-known that the standard trust-region radius Δ_k is independent of the gradient and Hessian of the objective function, so we are not able to know if the radius Δ_k is convenient for the whole implementation. This condition might lead to an increase in the number of subproblems that must be resolved in the method's inner phases which reduces the method's efficiency.

To overcome this problem many authors proposed various nonmonotone trust-region methods, for example, see [9, 31, 37, 38, 43, 44]. Motivated by the nonmonotone trust-region strategy in [31], we use it in our method. It is generally promising and efficient, and it can overcome the aforementioned shortcomings.

Furthermore, the usefulness of the CHKS smoothing function with the nonmonton active interior-point trust-region algorithm to solve the NBLP problems was examined by using several benchmark problems and a real-world case about a watershed trading decision-making problem.

Numerical experiments show that the suggested method surpasses rival algorithms in terms of efficacy.

This paper is organized as follows: Section 2 introduces the mathematical formulation of the NBLP problem, basic definitions of the CHKS smoothing functions , and the smoothing method for the nonlinear complementarity problem to obtain the SNLP problem. Section 3 introduces the nonmonton active interior-point trust region algorithm for solving the SNLP problem. Results from simulations on several benchmark problems and a real-world case about a watershed trading decision-making problem are reported in Section 4. The conclusion is given in Section 5.

2. Mathematical model for SNLP problem

In this paper, we will consider the following NBLP problem

$$\begin{array}{ll} \min\limits_{a_u \le y \le b_u} & f_u(y,z) \\ s.t. & c_u(y,z) \le 0, \\ \quad \min\limits_{a_l \le z \le b_l} & f_l(y,z), \\ \quad s.t. & c_l(y,z) \le 0, \end{array} \tag{2.1}$$

where $y \in \mathfrak{R}^{n_1}$ and $z \in \mathfrak{R}^{n_2}$. The functions $f_u : \mathfrak{R}^{n_1+n_2} \to \mathfrak{R}$, $f_l : \mathfrak{R}^{n_1+n_2} \to \mathfrak{R}$, $c_u : \mathfrak{R}^{n_1+n_2} \to \mathfrak{R}^{m_1}$, and $c_l : \mathfrak{R}^{n_1+n_2} \to \mathfrak{R}^{m_2}$ are assumed to be at least twice continuously differentiable function in our method.

The NBLP problem (2.1) was reduced to the following one-objective optimization problem using Karush-Kuhn-Tucker optimality assumptions for the lower level problem:

$$\begin{array}{ll} \min\limits_{y,z} & f_u(y,z) \\ s.t. & c_u(y,z) \le 0, \\ & \nabla_z f_l(y,z) + \nabla_z c_l(y,z)\lambda = 0, \\ & c_l(y,z) \le 0, \\ & \lambda_j c_{l_j}(y,z) = 0, \quad j = 1,...,m_2, \\ & \lambda_j \ge 0, \quad j = 1,...,m_2, \\ & a_u \le y \le b_u, \\ & a_l \le z \le b_l, \end{array} \tag{2.2}$$

where $\lambda \in \mathfrak{R}^{m_2}$ is a multiplier vector associated with the inequality constraint $c_l(y,z)$. Problem (2.2) with the nonlinear complementarity condition is non-convex and non-differentiable; moreover, the regularity assumptions required to handle smooth optimization problems are never satisfied and it is not good to use our approach to solve problem (2.2). Due to this, we use the CHKS smoothing function to overcome this problem, see [5, 26, 35].

Definition 2.1. *The Chen-Mangasarian smoothing function is represented by the notation* $\hat{\phi}(g,h) : \mathfrak{R}^2 \to \mathfrak{R}$ *and defined by the expression* $\hat{\phi}(g,h) = g + h - \sqrt{(g-h)^2}$. *By introducing a smoothing parameter* $\tilde{\epsilon} \in R$ *into the the smoothing function* $\hat{\phi}(g,h)$, *we obtain the CHKS smoothing function*

$$\hat{\phi}(g,h,\tilde{\epsilon}) = g + h - \sqrt{(g-h)^2 + 4\tilde{\epsilon}^2}. \tag{2.3}$$

The Chen-Mangasarian smoothing function has the property $\hat{\phi}(g,h)=0$ if and only if $g\geq 0$, $h\geq 0$, $gh=0$ but it is non-differentiable at $g=h=0$. But, the CHKS smoothing function has the property $\hat{\phi}(g,h,\tilde{\epsilon})=0$ if and only if $g\geq 0$, $h\geq 0$, and $gh=\frac{\tilde{\epsilon}}{2}$ for $\tilde{\epsilon}\geq 0$, and the function is smoothing for g,h, and $\tilde{\epsilon}\geq 0$.

Consequently, by using the CHKS smoothing function (2.3), the problem (2.2) can be approximated as follows:

$$\begin{array}{ll}\min\limits_{y,z} & f_u(y,z)\\ s.t. & c_u(y,z)\leq 0,\\ & \nabla_z f_l(y,z)+\nabla_z c_l(y,z)\lambda=0,\\ & \lambda_j-c_{l_j}-\sqrt{(\lambda_j+c_{l_j})^2+4\tilde{\epsilon}^2}=0,\quad j=1,...,m_2,\\ & a_u\leq y\leq b_u,\\ & a_l\leq z\leq b_l.\end{array}\tag{2.4}$$

The above smoothed nonlinear programming problem can be summarized as follows

$$\begin{array}{ll}\min\limits_{x} & f_u(x)\\ s.t. & c_u(x)\leq 0,\\ & c_e(x)=0,\\ & \beta_a\leq x\leq\beta_b,\end{array}\tag{2.5}$$

where $x=(y,z,\lambda)^T\in\Re^n$ where $n=n_1+n_2+m_2$, $c_e(x)=[\nabla_z f_l(y,z)+\nabla_z c_l(y,z)\lambda, \lambda_j-c_{l_j}-\sqrt{(\lambda_j+c_{l_j})^2+4\tilde{\epsilon}^2}]$, $j=1,...,m_2$. The functions $f_u(x):\Re^n\to\Re$, $c_u(x):\Re^n\to\Re^{m_1}$, and $c_e(x):\Re^n\to\Re^{n_2+m_2}$ are twice continuously differentiable and $m_1<n$. We denote the feasible set $\boldsymbol{E}=\{x:\beta_a\leq x\leq\beta_b\}$ and the strict interior feasible set $int(\boldsymbol{E})=\{x:\beta_a<x<\beta_b\}$ where $\beta_a\in\{\Re\bigcup\{-\infty\}\}^n$, $\beta_b\in\{\Re\bigcup\{\infty\}\}^n$, and $\beta_a<\beta_b$.

Several methods that have been suggested to solve the smoothed nonlinear programming problem (2.5), see [11, 12, 16, 17, 19]. The nonmonton active interior-point trust-region algorithm is proposed in this paper to solve problem (2.5) and a detailed description of this algorithm is clarified in the following section.

3. Nonmontone active-set interior-point trust-region algorithm

In this section, firstly, we will offer a detailed description of the active-set strategy with the penalty method to convert problem (2.5) to an equality-constrained optimization problem with bounded variables. Second, the basic steps for using Newton's interior-point method to solve the equality-constrained optimization problem are presented clearly. Thirdly, the main steps for the nonmontone trust-region algorithm are presented. Finally, the main steps for solving problem 2.1 are introduced.

3.1. An active-set strategy

Motivated by the active-set strategy proposed in [8] and used in [10–14], we define a 0-1 diagonal matrix $D(x)\in\Re^{m_1\times m_1}$, whose diagonal entries are defined as follows:

$$d_i(x)=\begin{cases}1 & \text{if } c_{u_i}(x)\geq 0,\\ 0 & \text{if } c_{u_i}(x)<0.\end{cases}\tag{3.1}$$

Problem (2.5) is transformed into the following equality-constrained optimization problem using the previous matrix.

$$\begin{array}{cc} \min\limits_{x} & f_u(x) \\ s.t. & c_e(x) = 0, \\ & c_u(x)^T D(x) c_u(x) = 0, \\ & \beta_a \leq x \leq \beta_b. \end{array}$$

The previous problem is transformed into the following problem by using a penalty method

$$\begin{array}{cc} \min\limits_{x} & f_u(x) + \frac{\sigma_u}{2}\|D(x)c_u(x)\|^2 \\ s.t. & c_e(x) = 0, \\ & \beta_a \leq x \leq \beta_b, \end{array} \tag{3.2}$$

where σ_u is a positive parameter.

3.2. Newton's interior-point method

Motivated by the interior point method in [6], we let $L(x, \mu_e, \lambda_a, \lambda_b)$ be a Lagrangian function associated with problem (3.2) and it is defined as follows

$$L(x, \mu_e, \lambda_a, \lambda_b) = \ell(x, \mu_e; \sigma_u) - \lambda_a^T(x - \beta_a) - \lambda_b^T(\beta_b - x), \tag{3.3}$$

where

$$\ell(x, \mu_e) = f_u(x) + \mu_e^T c_e(x), \tag{3.4}$$

and

$$\ell(x, \mu_e; \sigma_u) = \ell(x, \mu_e) + \frac{\sigma_u}{2}\|D(x)c_u(x)\|^2, \tag{3.5}$$

such that μ_e, λ_a, and λ_b represent the Lagrange multiplier vectors associated with the equality constraint $c_e(x)$ and inequality constraints $(x - \beta_a)$ and $(\beta_b - x)$ respectively. A point $x_* \in \boldsymbol{E}$ will be a local minimizer of problem (3.2) if there exists multiplier vectors $\mu_{e_*} \in \Re^{m_1}$, $\lambda_{a_*} \in \Re_+^n$, and $\lambda_{b_*} \in \Re_+^n$ such that $(x_*, \mu_{e_*}, \lambda_{a_*}, \lambda_{b_*})$ satisfies the following Karush-Kuhn-Tucker conditions,

$$\nabla_x \ell(x_*, \mu_{e_*}; \sigma_{e_*}) - \lambda_{a_*} + \lambda_{b_*} = 0, \tag{3.6}$$

$$c_e(x_*) = 0, \tag{3.7}$$

$$\lambda_{a_*}{}^{(j)}(x_*^{(j)} - \beta_a{}^{(j)}) = 0, \tag{3.8}$$

$$\lambda_{b_*}{}^{(j)}(\beta_b{}^{(j)} - x_*^{(j)}) = 0, \tag{3.9}$$

where

$$\nabla_x \ell(x_*, \mu_{e_*}; \sigma_{u_*}) = \nabla_x \ell(x_*, \mu_{e_*}) + \sigma_{u_*} \nabla c_u(x_*) D(x_*) c_u(x_*), \tag{3.10}$$

and

$$\nabla_x \ell(x_*, \mu_{e_*}) = \nabla f_u(x_*) + \nabla c_e(x_*) \mu_{e_*}. \tag{3.11}$$

Let $V(x)$ be a diagonal matrix whose diagonal elements are as follows:

$$v^{(j)}(x) = \begin{cases} \sqrt{(x^{(j)} - \beta_a{}^{(j)})}, & \text{if } (\nabla_x \ell(x_*, \mu_{e_*}; \sigma_{u_*}))^{(j)} \geq 0 \text{ and } \beta_a{}^{(j)} > -\infty, \\ \sqrt{(\beta_b{}^{(j)} - x^{(j)})}, & \text{if } (\nabla_x \ell(x_*, \mu_{e_*}; \sigma_{u_*}))^{(j)} < 0 \text{ and } \beta_b{}^{(j)} < +\infty\ , \\ 1, & \text{otherwise.} \end{cases} \tag{3.12}$$

For more details see [11, 12, 20]. Using the scaling matrix $V(x)$, conditions (3.6)–(3.9) are equivalent to the following nonlinear system,

$$V^2(x)\nabla_x \ell(x_*, \mu_{e_*}; \sigma_{u_*}) = 0, \tag{3.13}$$

$$c_e(x) = 0. \tag{3.14}$$

For the following reasons, the nonlinear systems (3.13) and (3.14) is continuous but not everywhere differentiable.

- It may be non-differentiable when $v^{(j)} = 0$. To overcome this problem, restricting $x \in int(\boldsymbol{E})$.
- It may be non-differentiable when $v^{(j)}$ has an infinite upper bound and a finite lower bound, and $(\nabla_x \ell(x_*, \mu_{e_*}; \sigma_{u_*}))^{(j)} = 0$. To overcome this problem, we define a vector $\theta(x)$ whose components $\theta^{(j)}(x) = \frac{\partial((v^{(j)})^2)}{\partial x^{(j)}}$, $j = 1, ..., n$ are defined as follows

$$\theta^{(j)}(x) = \begin{cases} 1, & \text{if } (\nabla_x \ell(x, \mu_e; \sigma_u))^{(j)} \geq 0 \text{ and } {\beta_a}^{(j)} > -\infty, \\ -1, & \text{if } (\nabla_x \ell(x, \mu_e; \sigma_u))^{(j)} < 0 \text{ and } {\beta_b}^{(j)} < +\infty, \\ 0, & \text{otherwise.} \end{cases} \tag{3.15}$$

If we use Newton's method to solve the nonlinear systems (3.13) and (3.14), we get

$$[V^2(x)\nabla_x^2 \ell(x, \mu_e; \sigma_u) + \text{diag}(\nabla_x \ell(x, \mu_e; \sigma_u))\text{diag}(\theta(x))]\Delta x + V^2(x)\nabla c_e(x)\Delta\mu_e = -V^2(x)\nabla_x \ell(x, \mu_e; \sigma_u), \tag{3.16}$$

$$\nabla c_e(x)^T \Delta x = -c_e(x), \tag{3.17}$$

where

$$\nabla_x^2 \ell(x, \mu_e; \sigma_u) = H + \sigma_u \nabla c_u(x) D(x) \nabla c_u(x)^T, \tag{3.18}$$

and H is the Hessian of the Lagrangian function (3.4) or an approximation to it.

Restricting $x \in int(\boldsymbol{E})$ is necessary to ensure that the matrix $V(x)$ is nonsingular. Therefore, putting $\Delta x = V(x)s$ in both Eqs (3.16) and (3.17) and multiplying both sides of equation (3.16) by $V^{-1}(x)$, we get

$$[V(x)\nabla_x^2 \ell(x, \mu_e; \sigma_u)V(x) + \text{diag}(\nabla_x \ell(x, \mu_e; \sigma_u))\text{diag}(\theta(x))]s + V(x)\nabla c_e(x)\Delta\mu_e = -V(x)\nabla_x \ell(x, \mu_e; \sigma_u), \tag{3.19}$$

$$(V(x)\nabla c_e(x))^T s = -c_e(x). \tag{3.20}$$

It should be noted that the step s_k produced by the systems (3.19) and (3.20) is equivalent to the step produced by resolving the following quadratic programming subproblem,

$$\begin{aligned} \min_x \quad & \ell(x, \mu_e; \sigma_u) + V(x)\nabla_x \ell(x, \mu_e; \sigma_u)^T s + \tfrac{1}{2} s^T B s \\ s.t. \quad & c_e(x) + V(x)\nabla c_e(x)^T s = 0, \end{aligned} \tag{3.21}$$

where

$$B = G(x) + \sigma_u V(x)\nabla c_u(x) D(x) \nabla c_u(x)^T V(x), \tag{3.22}$$

and

$$G(x) \quad = \quad V(x)H(x)V(x) + \text{diag}(\nabla_x \ell(x, \mu_e; \sigma_u))\text{diag}(\theta(x)). \tag{3.23}$$

While Newton's approach has the advantage of being quadratically convergent under reasonable assumptions, it also has the disadvantage of requiring that the starting point be close to the solution. The nonmonotone trust-region globalization approach is used to ensure convergence from any starting point. The nonmonotone trust-region globalization strategy is a crucial method for solving a smooth nonlinear unconstrained or constrained optimization problem that can produce substantial global convergence. In the following section, we present the main steps of the nonmontone trust-region algorithm to solve quadratic subproblem (3.21).

The main steps of the process to apply the nonmontone trust-region algorithm to solve the quadratic subproblem (3.21) are described in the section that follows.

3.3. The nonmontone trust region algorithm

The trust-region subproblem associated with the problem (3.21) is given by

$$\begin{aligned} \min_x \quad & \ell(x, \mu_e; \sigma_u) + V(x)\nabla_x \ell(x, \mu_e; \sigma_u)^T s + \tfrac{1}{2} s^T B s \\ s.t. \quad & c_e(x) + V(x)\nabla c_e(x)^T s = 0, \\ & \|s\| \leq \delta_k, \end{aligned} \tag{3.24}$$

where δ_k is the radius of the trust-region.

Due to the possibility of no intersecting points existing between the hyperplane of the linearized constraints $c_e(x) + V(x)\nabla c_e(x)^T s = 0$ and the constraint $\|s\| \leq \delta_k$, subproblem (3.26) may be infeasible. There is no guarantee that this will remain true even if they do intersect if δ_k is reduced; see [7]. To solve this problem, we used a reduced Hessian strategy. This strategy was suggested in [2, 32] and subsequently implemented in [12, 13, 16–18, 20]. This strategy divides the step s_k into two orthogonal components: the normal component s_k^n for improving feasibility and the tangential component s_k^t for improving optimality. That is, $s_k = s_k^n + s_k^t$ and $s_k^t = \tilde{Z}_k \bar{s}_k^t$ where $\tilde{Z}_k$ is a matrix whose columns form a basis for the null space of $(V_k \nabla c_{e_k})^T$.

- **To compute** s_k^n

Obtaining the normal component s_k^n by solving the following trust-region subproblem

$$\min_x \tfrac{1}{2}\|c_{e_k} + V_k \nabla c_{e_k}{}^T s^n\|^2 \qquad s.t. \|s^n\| \leq \zeta \delta_k, \tag{3.25}$$

for some $\zeta \in (0, 1)$.

A conjugate gradient method [34] which is very cheap if the problem is large-scale and the Hessian is indefinite, is used to compute the normal component s_k^n. The main steps involved in applying the conjugate gradient method to solve subproblem (3.25) are presented in the following algorithm.

Algorithm 3.1. *: (A conjugate gradient method to calculate s_k^n)*
Step 1. *Set $s_0^n = 0$, $r_{n_0} = -V_k \nabla c_{e_k} c_{e_k}$ and $p_{n_0} = r_{n_0}$; pick $\epsilon > 0$.*
Step 2. *For i=0,1,.... do*

Compute $\gamma_{n_i} = \frac{{r_{n_i}}^T r_{n_i}}{{p_{n_i}}^T V_k \nabla c_{e_k} \nabla c_{e_k}^T V_k p_{n_i}}$.
Compute τ_{n_i} *such that* $\|s_i^n + \tau_n p_{n_i}\| = \delta_k$.
If $\gamma_{n_i} \leq 0$, *or if* $\gamma_{n_i} > \tau_{n_i}$, *then set* $s_k^n = s_i^n + \tau_{n_i} p_{n_i}$ *and stop.*
Otherwise set $s_{i+1}^n = s_i^n + \gamma_{n_i} p_{n_i}$ *and*
$r_{i+1}^n = r_{n_i} - \gamma_{n_i} V_k \nabla c_{e_k} \nabla c_{e_k}^T V_k p_{n_i}$.
If $\frac{\|r_{n_{i+1}}\|}{\|r_{n_0}\|} \leq \epsilon_0$,
set $s_k^n = s_{i+1}^n$ *and stop.*
Compute $\hat{\beta}_i = \frac{{r_{n_{i+1}}}^T r_{n_{i+1}}}{{r_{n_i}}^T r_{n_i}}$, *and the new direction* $p_{n_{i+1}} = r_{n_{i+1}} + \hat{\beta}_i p_{n_i}$.

Given s_k^n, let $q(V_k s_k)$ be the quadratic form of the function (3.5) and let it be defined as follows

$$q(V_k s_k) = \ell(x_k, \mu_{e_k}; \sigma_{u_k}) + V \nabla_x \ell(x_k, \mu_{e_k}; \sigma_{u_k})^T s + \frac{1}{2} s_k^T B_k s_k. \tag{3.26}$$

- **To compute** s_k^t

To compute the tangential component $s_k^t = \tilde{Z}_k \bar{s}_k^t$, solve the following trust-region subproblem

$$\begin{array}{ll} \min\limits_{x} & [\tilde{Z}_k^T \nabla q_k(V_k s_k^n) + B_k s_k^n]^T \bar{s}^t + \frac{1}{2} \bar{s}^{t^T} \tilde{Z}_k^T B_k \tilde{Z}_k \bar{s}^t \\ s.t. & \|\tilde{Z}_k \bar{s}^t\| \leq \Delta_k, \end{array} \tag{3.27}$$

where $\nabla q_k(V_k s_k^n) = V_k \nabla_x \ell(x_k, \mu_{e_k}; \sigma_{u_k}) + B_k s_k^n$ and $\Delta_k = \sqrt{\delta_k^2 - \|s_k^n\|^2}$. The main steps involved in applying the conjugate gradient method to solve subproblem (3.27) are presented in the following algorithm.

Algorithm 3.2. *(A conjugate gradient method to compute* s_k^t*)*
Step 1. *Set* $\bar{s}_0^t = s_k^n$, $r_{t_0} = -\tilde{Z}_k^T \nabla q_k(V_k s_k^n) + B_k s_k^n$, $p_{t_0} = r_{t_0}$.
Step 2. *For i=0,1,.... do*

Compute $\gamma_{t_i} = \frac{r_{t_i}^T r_{t_i}}{p_{t_i}^T \tilde{Z}_k^T B_k \tilde{Z}_k p_{t_i}}$.
Compute τ_{t_i} *such that* $\|\bar{s}_i^t + \tau_t \, p_{t_i}\| = \Delta_k$.
If $\gamma_{t_i} \leq 0$, *or if* $\gamma_{t_i} > \tau_{t_i}$, *then set* $\bar{s}_k^t = \bar{s}_i^t + \tau_{t_i} p_{t_i}$ *and stop.*
Otherwise set $\bar{s}_{i+1}^t = \bar{s}_i^t + \gamma_{t_i} p_{t_i}$ *and*
$r_{t_{i+1}} = r_{t_i} - \gamma_{t_i} \tilde{Z}_k^T B_k \tilde{Z}_k p_{t_i}$.
If $\frac{\|r_{t_{i+1}}\|}{\|r_{t_0}\|} \leq \epsilon$,
set $\bar{s}_k^t = \bar{s}_{i+1}^t$ *and stop.*
Compute $\bar{\beta}_i = \frac{r_{t_{i+1}}^T r_{t_{i+1}}}{r_{t_i}^T r_{t_i}}$, *and* $p_{t_{i+1}} = r_{t_{i+1}} + \bar{\beta}_i p_{t_i}$.

After computing s_k, we set $x_{k+1} = x_k + V_k s_k$. To ensure that the matrix V_k is nonsingular, we need to guarantee that $x_{k+1} \in int\boldsymbol{E}$. So, the damping parameter φ_k is required at every iteration k.

- **To obtain the damping parameter** φ_k

The following algorithm clarifies the fundamental steps required to obtain the damping parameter φ_k.

Algorithm 3.3. *(Damping parameter φ_k)*
Step 1. *Compute the parameter ω_k as follows*

$$\omega_k^{(i)} = \begin{cases} \frac{\beta_a^{(i)} - x_k^{(i)}}{V_k^{(i)} s_k^{(i)}}, & \text{if } \beta_a^{(i)} > -\infty \text{ and } V_k^{(i)} s_k^{(i)} < 0 \\ 1, & \text{otherwise.} \end{cases}$$

Step 2. *Compute the parameter υ_k as follows*

$$\upsilon_k^{(i)} = \begin{cases} \frac{\beta_b^{(i)} - x_k^{(i)}}{V_k^{(i)} s_k^{(i)}}, & \text{if } \beta_b^{(i)} < \infty \text{ and } V_k^{(i)} s_k^{(i)} > 0 \\ 1, & \text{otherwise.} \end{cases}$$

Step 3. *Compute the damping parameter φ_k as follows*

$$\varphi_k = \min\{1, \{\min_i\{\omega_k^{(i)}, \upsilon_k^{(i)}\}\}. \tag{3.28}$$

Step 4. *Set $x_{k+1} = x_k + \varphi_k V_k s_k$.*

To determine whether the scaled step $\varphi_k V_k s_k$ will be accepted or no, we need a merit function that connects the objective function and the constraints such that progress in the merit function equates to progress in solving the problem. The augmented Lagrangian function that follows is used as a merit function,

$$\Phi(x, \mu_e; \sigma_u; \sigma_e) = f_u(x) + \mu_e^T c_e(x) + \frac{\sigma_u}{2}\|D(x)c_u(x)\|^2 + \sigma_e\|c_e(x)\|^2, \tag{3.29}$$

where σ_e is the penalty parameter.

To test the scaled step, we use the following scheme to determine the Lagrange multiplier vector $\mu_{e_{k+1}}$,

$$\min \quad \|\nabla f_{u_{k+1}} + \nabla c_{e_{k+1}}\mu_e + \sigma_{u_k}\nabla c_{u_{k+1}} D_{k+1} c_{u_{k+1}}\|^2. \tag{3.30}$$

The following actual reduction and the predicted reduction must be defined to determine if the point $(x_{k+1}, \mu_{e_{k+1}})$ will be accepted in the next iteration or no. The actual reduction in the merit function in moving from (x_k, μ_{e_k}) to $(x_k + \varphi_k V_k s_k, \mu_{e_{k+1}})$ is defined as

$$Ared_k = \Phi(x_k, \mu_{e_k}; \sigma_{u_k}; \sigma_{e_k}) - \Phi(x_k + \varphi_k V_k s_k, \mu_{e_{k+1}}; \sigma_{u_k}; \sigma_{e_k}).$$

The predicted reduction $Pred_k$ in the merit function (3.29) is defined as follows

$$Pred_k \quad = \quad q_k(0) - q_k(\varphi_k V_k s_k) + \sigma_{e_k}[\|c_{e_k}\|^2 - \|c_{e_k} + \nabla c_{e_k}^T \varphi_k V_k s_k\|^2], \tag{3.31}$$

where

$$q_k(V_k\varphi_k s_k) = \ell(x_k, \mu_{e_k}) + \nabla_x \ell(x_k, \mu_{e_k})^T \varphi_k V_k s_k + \frac{1}{2}\varphi_k^2 s_k^T G_k s_k + \frac{\sigma_{u_k}}{2}\|D_k(c_{u_k} + \nabla c_{u_k}^T \varphi_k V_k s_k)\|^2. \tag{3.32}$$

To ensure that $Pred_k \geq 0$, the penalty parameter σ_{e_k} must be updated.

- **To update the penalty parameter σ_{e_k}**

To ensure that $Pred_k \geq 0$, we need to update the penalty parameter σ_{e_k} by using the following algorithm.

Algorithm 3.4. *(Process to update σ_{e_k})*
If

$$Pred_k \leq \frac{\sigma_{e_k}}{2}[\|c_{e_k}\|^2 - \|c_{e_k} + \nabla c_{e_k}^T \varphi_k V_k s_k\|^2], \tag{3.33}$$

set

$$\sigma_{e_k} = \frac{2[q_k(\varphi_k V_k s_k) - q_k(0)]}{\|c_{e_k}\|^2 - \|c_{e_k} + \nabla c_{e_k}^T \varphi_k V_k s_k\|^2} + \tilde{b}_0, \tag{3.34}$$

where $\tilde{b}_0 > 0$ is a small fixed constant.
Else, set $\sigma_{e_{k+1}} = \sigma_{e_k}$.
End if.

For more details, see [12].

- **To test $\varphi_k V_k s_k$ and update δ_k**

Motivated by the nonmonotone trust-region strategy in [31], we define

$$t_k = \frac{C_k - \Phi(x_k + \varphi_k V_k s_k, \mu_{e_{k+1}}; \sigma_{u_k}; \sigma_{e_k})}{Pred_k} \tag{3.35}$$

where

$$C_k = \begin{cases} \Phi(x_k; \mu_{e_k}; \sigma_{u_k}; \sigma_{e_k}), & \text{if } k = 0, \\ \frac{\xi_{k-1} Q_{k-1} C_{k-1} + \Phi(x_k + \varphi_k V_k s_k, \mu_{e_{k+1}}; \sigma_{u_k}; \sigma_{e_k})}{Q_k}, & \text{if } k \geq 1, \end{cases} \tag{3.36}$$

and

$$Q_k = \begin{cases} 1, & \text{if } k = 0, \\ \xi_{k-1} Q_{k-1} + 1, & \text{if } k \geq 1, \end{cases} \tag{3.37}$$

such that $0 \leq \xi_{\min} \leq \xi_{k-1} \leq \xi_{\max} \leq 1$ and

$$\xi_k = \begin{cases} 0.5\xi_0, & \text{if } k = 1, \\ 0.5(\xi_{k-1} + \xi_{k-2}), & \text{if } k \geq 2. \end{cases} \tag{3.38}$$

The procedure below introduces the trial step $\varphi_k V_k s_k$ for testing and updates the radius δ_k.

Algorithm 3.5. *(Test $\varphi_k V_k s_k$ and update δ_k)*
Step 0. *Choose $0 < \alpha_1 < \alpha_2 \leq 1$, $\delta_{max} > \delta_{min}$, and $0 < \zeta_1 < 1 < \zeta_2$.*
Step 1. *(Compute t_k)*

Compute ξ_k by using (3.38).
Compute Q_k as defined in (3.37).
Compute C_k as defined in (3.36).
Compute $t_k = \frac{C_k - \Phi(x_k + \varphi_k V_k s_k, \mu_{e_{k+1}}; \sigma_{u_k}; \sigma_{e_k})}{Pred_k}$.

Step 2. *(Update the trial step s_k)*

While $t_k < \alpha_1$, or $Pred_k \leq 0$,
set $\delta_k = \zeta_1 \|s_k\|$.
To evaluate a new step s_k, go to Algorithms (3.1) and (3.2).

Step 3. *(Update δ)*

If $\alpha_1 \leq t_k < \alpha_2$, then
set $x_{k+1} = x_k + \varphi_k V_k s_k$.
$\delta_{k+1} = \max(\delta_{min}, \delta_k)$.
End if.
If $t_k \geq \alpha_2$, then
set $x_{k+1} = x_k + \varphi_k V_k s_k$.
$\delta_{k+1} = \min\{\max\{\delta_{min}, \zeta_2 \delta_k\}, \delta_{max}\}$.
End if.

- **To update the parameter σ_{u_k}**

To update σ_{u_k}, we use a scheme proposed in [39]. Let a tangential predicted decrease $Tpred$ which is obtained by the tangential component s_k^t be given by

$$Tpred_k(\bar{s}_k^t) = q_k(V_k s_k^n) - q_k(V_k(s_k^n + \tilde{Z}_k \bar{s}_k^t)), \tag{3.39}$$

the main steps used to update the parameter σ_{u_k} is clarified by the following algorithm.

Algorithm 3.6. *(Updating σ_{u_k})*
Step 1. *Set $\sigma_{u_0} = 1$.*
Step 2. *If*

$$Tpred_k \geq \|V_k \nabla c_{u_k} D_k c_{u_k}\| \min\{\|V_k \nabla c_{u_k} D_k c_{u_k}\|, \delta_k\}, \tag{3.40}$$

then $\sigma_{u_{k+1}} = \sigma_{u_k}$,
else, $\sigma_{u_{k+1}} = 2\sigma_{u_k}$.
End if

Finally, the nonmontone trust-region algorithm is terminated when either

$$\|\tilde{Z}_k^T V_k \nabla_x \ell(x_k, \mu_{e_k})\| + \|V_k \nabla c_{u_k} D_k c_{u_k}\| + \|c_{e_k}\| \leq \varepsilon_1$$

or $\|s_k\| \leq \varepsilon_2$, for some $\varepsilon_1, \varepsilon_2 > 0$.

- **Nonmonotone trust-region algorithm**

In the algorithm that follows, we will outline the main steps of the nonmonotone trust-region algorithm in order to solve subproblem (3.21).

Algorithm 3.7. *(The nonmonotone trust-region algorithm)*
Step 0*. Start with $x_0 \in int\boldsymbol{E}$. Compute D_0, V_0, μ_{e_0}, and θ_0. Set $\sigma_{u_0} = 1$, $\sigma_{e_0} = 1$, and $\tilde{b}_0 = 0.1$.*
Choose $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$. Choose δ_{min}, δ_{max}, and δ_0 such that $\delta_{min} \leq \delta_0 \leq \delta_{max}$.
Choose α_1, α_2, ζ_1, and ζ_2 such that $0 < \zeta_1 < 1 < \zeta_2$, and $0 < \alpha_1 < \alpha_2 < 1$. Set $k = 0$.
Step 1. *(Termination)*
If $\|\tilde{Z}_k^T V_k \nabla_x \ell(x_k, \mu_{e_k})\| + \|V_k \nabla c_{u_k} D_k c_{u_k}\| + \|c_{e_k}\| \leq \varepsilon_1$, then stop the algorithm.
Step 2. *(Computing step $V_k \varphi_k s_k$)*

Evaluate the normal component step s_k^n by using Algorithm (3.1).
Evaluate the tangential step $\bar{s}_k^t$ by using Algorithm (3.2).
Set $s_k = s_k^n + \tilde{Z}_k \bar{s}_k^t$.
If $\|s_k\| \leq \varepsilon_2$, then stop the algorithm.
Else, compute the parameter φ_k by using (3.28).
Set $x_{k+1} = x_k + V_k \varphi_k s_k$.
End if.

Step 3. *Evaluate D_{k+1} which is defined by (3.1).*
Step 4. *(Updating the multiplier vector $\mu_{e_{k+1}}$)*
Computing the Lagrange multiplier vector $\mu_{e_{k+1}}$ by using (3.30).
Step 5. *(Updating the penalty parameter σ_e)*
Using Algorithm (3.4) to update the penalty parameter σ_{e_k} .
Step 6. *Compute t_k as defined in (3.35) by using both Q_k as defined in (3.37) and C_k as defined in (3.36).*
Using Algorithm (3.5) to test the scaling step $\varphi_k V_k s_k$ and update the radius of the trust-region.
Step 7. *Updating the parameter σ_{u_k} using Algorithm (3.6).*
Step 8. *Utilize (3.12) to evaluate the matrix V_{k+1}.*
Step 9. *Set $k = k + 1$ and go to Step 1.*

The global convergence theory for Algorithm (3.7) is similar to the global convergence theory of the algorithm presented in ([12]) when used to solve a general nonlinear programming problem.

3.4. CHKS nonmontone active-set interior-point trust-region algorithm

We will outline the main steps of the nonmontone active-set interior-point trust-region algorithm based on the CHKS smoothing function to resolve problem (2.1) in the following algorithm.

Algorithm 3.8. *(CHKS nonmontone active-set interior-point trust-region algorithm)*
Step 1. *Reduce the nonlinear bilevel programming problem (2.1) to the one-objective optimization problem (2.2) using Karush-Kuhn-Tucker optimality assumptions for the lower level problem.*
Step 2. *Use the CHKS smoothing function to convert non-deferential problem (2.2) to smoothing problem (2.4).*
Step 3. *Utilize an active-set strategy to convert the nonlinearly constrained optimization problem (2.4) to the equality constrained optimization problem with bounded variables (3.2).*
Step 4. *Utilize an interior-point method with the diagonal scaling matrix $V(x)$ given in (3.12) to obtain the nonlinear system [(3.13) - (3.14)].*
Step 5. *Utilize Newton's method with diagonal matrix θ as defined in (3.15), to solve the nonlinear system [(3.13)-(3.14)] and obtain the equivalent subproblem (3.21).*
Step 6. *Solve subproblem (3.21) by using nonmonton trust-region given by Algorithm (3.7).*

4. Computational tests and comparisons

In this section, we introduce an extensive variety of possible numeric NBLP problems to illustrate the validity of the proposed Algorithm (3.8) to solve problem (2.1).

4.1. Numerical examples

We shall introduced the MATLAB numerical results for Algorithm (3.8) with a starting point $x_0 \in int(\boldsymbol{E})$. The parameter setting that follows was utilized: $\alpha_1 = 10^{-2}$, $\alpha_2 = 0.75$, $\zeta_1 = 0.5$, $\zeta_2 = 2$, $\delta_{min} = 10^{-3}$, $\delta_0 = max(\|s_0^{cp}\|, \delta_{min})$, $\delta_{max} = 10^3\delta_0$, $\varepsilon_1 = 10^{-10}$, and $\varepsilon_2 = 10^{-12}$.

To demonstrate the efficacy of the suggested Algorithm (3.8) in obtaining the solution to NBLP problem (2.1), we offer a wide variety of possible numeric NBLP problems in this section. Fifteen benchmark instances from [4, 22, 27, 30, 33] were used to test the proposed Algorithm (3.8).

To check the consistency of the results, 10 independent runs using different initial starting points were carried out for each test example. Table 1 summarizes the statistical data for all examples and demonstrates that the proposed Algorithm (3.8) results are approximate or equal to those of the compared algorithms in the method proposed in [18] and the literature.

Table 1. Comparison of the results of Algorithm (3.8) with those of various existing methods(ref).

Problem name	(y_*, z_*) Algorithm (3.8)	f_u^* f_l^* Algorithm (3.8)	(y_*, z_*) Method [18]	f_u^* f_l^* Method [18]	(y_*, z_*) Ref.	f_u^* f_l^* Ref.
TP1	(0.8438, 0.7657, 0)	-2.0769 -0.5863	(.8465,0.7695,0)	-2.0772 -0.5919	(0.8438, 0.7657, 0)	-2.0769 -0.5863
TP2	(0.6111, 0.3890, 0, 0, 1.8339)	0.64013 1.6816	(0.6111,0.3890,0, 0, 1.8339)	0.64013 1.6816	(0.609, 0.391, 0, 0, 1.828)	0.6426 1.6708
TP3	(0.97, 3.14, 2.6, 1.8)	-8.92 -6.05	(0.97, 3.14 2.6, 1.8)	-8.92 -6.05	(0.97, 3.14, 2.6, 1.8)	-8.92 -6.05
TP4	(.5,.5,.5,.5)	-1 0	(0.5, 0.5, 0.5, 0.5)	-1 0	(0.5, 0.5, 0.5, 0.5)	-1 0
TP5	(9.9998, 9.9998)	99.9996 3.7160e-07	(9.9953,9.9955)	99.907 1.8628e-04	(10.03, 9.969)	100.58 0.001
TP6	(1.8884, 0.89041, 0)	-1.2067 7.6062	(1.8889,0.88889, 6.8157e-06)	-1.4074 7.6172	NA	3.57 2.4
TP7	(1, 0)	17 1	(1,0)	17 1	(1, 0)	17 1
TP8	(0.75,0.75, 0.75, 0.75)	-2.25 0	(0.7513,0.7513, 0.752,0.752)	-2.2480 0	($\sqrt{3}/2$, $\sqrt{3}/2$, $\sqrt{3}/2$, $\sqrt{3}/2$)	-2.1962 0
TP9	(11.25, 5)	2250 197.753	(11.25,5)	2250 197.753	(11.25,5)	2250 197.753
TP10	(1,0,0)	0 0	(1,0,1)	1 -1	(1,0,1)	1 -1
TP11	(25, 30, 4.9996, 10)	5.0024 1.5000e-06	(25, 30,5, 10)	5 0	(25,30,5,10)	5 0
TP12	(3,5)	9 0	(3,5)	9 0	(3,5)	9 0
TP13	(0, 2, 1.875, 0.90625)	-18.679 -1.0156	(0,2,1.875,0.9063)	-12.68 -1.016	(0,2,1.875,0.9063)	-12.68 -1.016
TP14	(10.016, 0.81967)	81.328 -0.33593	(10,0.011)	8.1978e+01 0	(10.04,0.1429)	82.44 0.271
TP15	(0,0.9,0,0.6,0.4)	-29.2 3.2	(0,0.9,0,0.6,0.4)	-29.2 3.2	(0,0.9,0,0.6,0.4)	-29.2 3.2

For purposes of comparison, in Table 2 we have provided the corresponding results of the average number of iterations (iter), the average number of function evaluations (nfunc), and the average amount of CPU time (CPUs) in seconds as obtained via the method in [18]. The results in Table 2

demonstrate that the proposed Algorithm (3.8) results are approximative or the best of those obtained via the comparative algorithms in the literature. Results show that our proposed method is capable of handling the NBLP problem (2.1) whether the upper or lower levels are convex or not, and the calculated results converge to the optimal solution that is approximate or equal to the optimal solution stated in the literature. Finally, it is evident from a comparison of the solutions obtained via the proposed Algorithm (3.8) and those found in the literature that the proposed Algorithm (3.8) is capable of finding the best solution to some problems under conditions of a small amount of time, fewer function evaluations, and fewer iterations.

Table 2. Comparison of the results of Algorithm (3.8) with those of the method in [18], with respect to the number of iterations, the number of function evaluations, and time.

Problem name	iter Algorithm (3.8)	nfunc Algorithm (3.8)	CPU(s) Algorithm (3.8)	iter Method [18]	nfunc Method [18]	CPUs Method [18]
TP1	10	11	1.54	10	13	1.62
TP2	8	9	1.78	9	12	1.87
TP3	6	8	2.1	7	8	2.52
TP4	10	12	1.76	12	13	1.92
TP5	6	8	1.65	6	7	1.523
TP6	7	9	3.5	8	10	3.95
TP7	11	13	1.8	11	12	1.652
TP8	4	5	1.569	11	12	0.953
TP9	9	11	2.23	8	10	1.87
TP10	4	7	2.887	5	6	3.31
TP11	9	11	3.542	10	13	3.632
TP12	4	5	1.12	7	9	1.33
TP13	5	7	2.1	5	8	1.998
TP14	5	6	1.87	5	6	1.97
TP15	5	6	20.212	6	7	20.125

Test Problem 1 [28]:

$$\begin{aligned} \min_{y_1} \quad & f_u = z_1^2 + z_2^2 + y_1^2 - 4y_1 \\ s.t. \quad & 0 \le y_1 \le 2, \\ \min_{z_1,z_2} \quad & f_l = z_1^2 + 0.5z_2^2 + z_1z_2 + \\ & (1-3y_1)z_1 + (1+y_1)z_2, \\ s.t. \quad & 2z_1 + z_2 - 2y_1 \le 1, \\ & z_1 \ge 0, \quad z_2 \ge 0. \end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 1 is converted to the following SNLP problem

$$\begin{aligned} \min_{y_1,z_1,z_2,\lambda} \quad & f_u = z_1^2 + z_2^2 + y_1^2 - 4y_1 \\ s.t. \quad & 2z_1 + z_2 + (1-3y_1) + (1+y_1) + 2\lambda = 0, \\ & z_1 + z_2 + (1+y_1) + \lambda = 0, \\ & \lambda - (2z_1 + z_2 - 2y_1 - 1) - \sqrt{(\lambda + 2z_1 + z_2 - 2y_1 - 1)^2 + 4\tilde{\epsilon}^2} = 0, \\ & 0 \le y_1 \le 2, \\ & z_1 \ge 0, \quad z_2 \ge 0. \end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, z_1^*, z_2^*) = (0.8438, 0.7657, 0)$, $f_u = -2.0769$, and $f_l = -0.5863$.

Test Problem 2 [28]:

$$\begin{aligned}
\min_{y_1,y_2} \quad & f_u = z_1^2 + z_3^2 - z_1z_3 - 4z_2 - 7y_1 + 4y_2 \\
s.t. \quad & y_1 + y_2 \leq 1, \\
& y_1 \geq 0, \quad y_2 \geq 0 \\
\min_{z_1,z_2,z_3} \quad & f_l = z_1^2 + 0.5z_2^2 + 0.5z_3^2 + z_1z_2 + \\
& (1 - 3y_1)z_1 + (1 + y_2)z_2, \\
s.t. \quad & 2z_1 + z_2 - z_3 + y_1 - 2y_2 + 2 \leq 0, \\
& z_1 \geq 0; \quad z_2 \geq 0 \quad z_3 \geq 0.
\end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 2 is converted to the following SNLP problem

$$\begin{aligned}
\min_{y_1,y_2,z_1,z_2,z_3,\lambda} \quad & f_u = z_1^2 + z_3^2 - z_1z_3 - 4z_2 - 7y_1 + 4y_2 \\
s.t. \quad & 2z_1 + z_2 + (1 - 3y_1) + 2\lambda = 0, \\
& z_1 + z_2 + (1 + y_2) + \lambda = 0, \\
& z_3 - \lambda = 0, \\
& \lambda - (2z_1 + z_2 - z_3 + y_1 - 2y_2 + 2) - \sqrt{(\lambda + 2z_1 + z_2 - z_3 + y_1 - 2y_2 + 2)^2 + 4\tilde{\epsilon}^2} = 0, \\
& y_1 + y_2 \leq 1, \\
& y_1 \geq 0; \quad y_2 \geq 0 \quad z_1 \geq 0; \quad z_2 \geq 0 \quad z_3 \geq 0.
\end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*) = (0.6111, 0.3890, 0, 0, 1.8339)$, $f_u = 0.64013$, and $f_l = 1.6816$.

Test Problem 3 [28]:

$$\begin{aligned}
\min_{y_1,y_2} \quad & f_u = 0.1(y_1^2 + y_2^2) - 3z_1 - 4z_2 + 0.5(z_1^2 + z_2^2) \\
s.t. \quad & \\
\min_{z_1,z_2} \quad & f_l = 0.5(z_1^2 + 5z_2^2) - 2z_1z_2 - y_1z_1 - y_2z_2, \\
s.t. \quad & -0.333z_1 + z_2 - 2 \leq 0, \\
& z_1 - 0.333z_2 - 2 \leq 0, \\
& z_1 \geq 0, \quad z_2 \geq 0.
\end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 3 is converted to the following SNLP problem

$$\begin{aligned}
\min_{y_1,y_2,z_1,z_2,\lambda_1,\lambda_2} \quad & f_u = 0.1(y_1^2 + y_2^2) - 3z_1 - 4z_2 + 0.5(z_1^2 + z_2^2) \\
s.t. \quad & z_1 - 2z_2 - y_1 - 0.333\lambda_1 + \lambda_2 = 0, \\
& 5z_2 - 2z_1 - y_2 + \lambda_1 - 0.333\lambda_2 = 0, \\
& \lambda_1 - (-0.333z_1 + z_2 - 2) - \sqrt{(\lambda_1 - 0.333z_1 + z_2 - 2)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_2 - (z_1 - 0.333z_2 - 2) - \sqrt{(\lambda_2 + z_1 - 0.333z_2 - 2)^2 + 4\tilde{\epsilon}^2} = 0, \\
& y_1 \geq 0, \quad y_2 \geq 0, \quad z_1 \geq 0, \quad z_2 \geq 0.
\end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, y_2^*, z_1^*, z_2^*) = (0.97, 3.14, 2.6, 1.8)$, $f_u = -8.92$, and $f_l = -6.05$.

Test Problem 4 [28]:

$$\begin{aligned} \min_{y_1,y_2} \quad & f_u = y_1^2 - 2y_1 + y_2^2 - 2y_2 + z_1^2 + z_2^2 \\ S.T \quad & y_1 \geq 0, \quad y_2 \geq 0 \\ \min_{z_1,z_2} \quad & f_l = (z_1 - y_1)^2 + (z_2 - y_2)^2, \\ s.t. \quad & 0.5 \leq z_1 \leq 1.5, \\ & 0.5 \leq z_2 \leq 1.5. \end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 4 is converted to the following SNLP problem

$$\begin{aligned} \min_{y_1,y_2,z_1,z_2} \quad & f_u = y_1^2 - 2y_1 + y_2^2 - 2y_2 + z_1^2 + z_2^2 \\ s.t \quad & 2(z_1 - y_1) = 0, \\ & 2(z_2 - y_2) = 0, \\ & 0.5 \leq z_1 \leq 1.5, \\ & 0.5 \leq z_2 \leq 1.5, \\ & y_1 \geq 0, \quad y_2 \geq 0. \end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, y_2^*, z_1^*, z_2^*) = (0.5, 0.5, 0.5, 0.5)$, $f_u = -1$, and $f_l = 0$.

Test Problem 5 [28]:

$$\begin{aligned} \min_{y} \quad & f_u = y^2 + (z - 10)^2 \\ s.t. \quad & -y + z \leq 0, \\ & 0 \leq y \leq 15, \\ \min_{z} \quad & f_l = (y + 2z - 30)^2, \\ s.t. \quad & y + z \leq 20, \\ & 0 \leq z \leq 20. \end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 5 is converted to the following SNLP problem

$$\begin{aligned} \min_{y,z,\lambda} \quad & f_u = y^2 + (z - 10)^2 \\ s.t \quad & 4(y + 2z - 30) + \lambda = 0, \\ & \lambda - (y + z - 20) - \sqrt{(\lambda + y + z - 20)^2 + 4\tilde{\epsilon}^2} = 0, \\ & -y + z \leq 0, \\ & 0 \leq y \leq 15, \\ & 0 \leq z \leq 20. \end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y^*, z^*) = (9.9998, 9.9998)$, $f_u = 99.9996$, and $f_l = 3.7160e - 07$.

Test Problem 6 [28]:

$$\begin{aligned}
\min_{y_1} \quad & f_u = (y_1 - 1)^2 + 2z_1^2 - 2y_1 \\
s.t. \quad & y_1 \geq 0, \\
\min_{z_1, z_2} \quad & f_l = (2z_1 - 4)^2 + (2z_2 - 1)^2 + y_1 z_1, \\
s.t. \quad & 4y_1 + 5z_1 + 4z_2 \leq 12, \\
& -4y_1 - 5z_1 + 4z_2 \leq -4, \\
& 4y_1 - 4z_1 + 5z_2 \leq 4, \\
& -4y_1 + 4z_1 + 5z_2 \leq 4, \\
& z_1 \geq 0, \quad z_2 \geq 0.
\end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 6 is converted to the following SNLP problem

$$\begin{aligned}
\min_{y_1, z_1, z_2, \lambda_1, \lambda_2, \lambda_3, \lambda_4} \quad & f_u = (y_1 - 1)^2 + 2z_1^2 - 2y_1 \\
s.t. \quad & 4(2z_1 - 4) + y_1 + 5\lambda_1 - 5\lambda_2 - 4\lambda_3 + 4\lambda_4 = 0, \\
& 4(2z_2 - 1) + 4\lambda_1 + 4\lambda_2 + 5\lambda_3 + 5\lambda_4 = 0, \\
& \lambda_1 - (4y_1 + 5z_1 + 4z_2 - 12) - \sqrt{(\lambda_1 + 4y_1 + 5z_1 + 4z_2 - 12)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_2 - (-4y_1 - 5z_1 + 4z_2 + 4) - \sqrt{(\lambda_2 - 4y_1 - 5z_1 + 4z_2 + 4)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_3 - (4y_1 - 4z_1 + 5z_2 - 4) - \sqrt{(\lambda_3 + 4y_1 - 4z_1 + 5z_2 - 4)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_4 - (-4y_1 + 4z_1 + 5z_2 - 4) - \sqrt{(\lambda_4 - 4y_1 + 4z_1 + 5z_2 - 4)^2 + 4\tilde{\epsilon}^2} = 0, \\
& y_1 \geq 0, \quad z_1 \geq 0, \quad z_2 \geq 0.
\end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, z_1^*, z_2^*) = (1.8884, 0.89041, 0)$, $f_u = -1.2067$, and $f_l = 7.6062$.

Test Problem 7 [28]:

$$\begin{aligned}
\min_{y} \quad & f_u = (y - 5)^2 + (2z + 1)^2 \\
s.t. \quad & y \geq 0, \\
\min_{z} \quad & f_l = (2z - 1)^2 - 1.5yz, \\
s.t. \quad & -3y + z \leq -3, \\
& y - 0.5z \leq 4, \\
& y + z \leq 7, \\
& z \geq 0.
\end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 7 is converted to the following SNLP problem

$$\begin{aligned}
\min_{y, z, \lambda_1, \lambda_2, \lambda_3} \quad & f_u = (y - 5)^2 + (2z + 1)^2 \\
s.t. \quad & 4(2z - 1) - 1.5y + \lambda_1 - 0.5\lambda_2 + \lambda_3 = 0, \\
& \lambda_1 - (-3y + z + 3) - \sqrt{(\lambda_1 - 3y + z + 3)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_2 - (y - 0.5z - 4) - \sqrt{(\lambda_2 + y - 0.5z - 4)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_3 - (y + z - 7) - \sqrt{(\lambda_2 + y + z - 7)^2 + 4\tilde{\epsilon}^2} = 0, \\
& y \geq 0, \quad z \geq 0.
\end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y^*, z^*) = (1, 0)$, $f_u = 17$, and $f_l = 1$.

Test Problem 8 [28]:

$$\begin{aligned} \min_{y_1,y_2} \quad & f_u = y_1^2 - 3y_1 + y_2^2 - 3y_2 + z_1^2 + z_2^2 \\ s.t. \quad & y_1 \geq 0, \quad y_2 \geq 0, \\ \min_{z_1,z_2} \quad & f_l = (z_1 - y_1)^2 + (z_2 - y_2)^2, \\ s.t. \quad & 0.5 \leq z_1 \leq 1.5, \\ & 0.5 \leq z_2 \leq 1.5. \end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 8 is converted to the following SNLP problem

$$\begin{aligned} \min_{y_1,y_2,z_1,z_2} \quad & f_u = y_1^2 - 3y_1 + y_2^2 - 3y_2 + z_1^2 + z_2^2 \\ s.t. \quad & 2(z_1 - y_1) = 0, \\ & 2(z_2 - y_2) = 0, \\ & 0.5 \leq z_1 \leq 1.5, \\ & 0.5 \leq z_2 \leq 1.5, \\ & y_1 \geq 0, \quad y_2 \geq 0. \end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, y_2^*, z_1^*, z_2^*) = (0.75; 0.75; 0.75; 0.75)$, $f_u = -2.25$, and $f_l = 0$.

Test Problem 9 [23]:

$$\begin{aligned} \min_{y} \quad & f_u = 16y^2 + 9z^2 \\ s.t. \quad & -4y + z \leq 0, \\ & y \geq 0, \\ \min_{z} \quad & f_l = (y + z - 20)^4, \\ s.t. \quad & 4y + z - 50 \leq 0, \\ & z \geq 0. \end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 9 is converted to the following SNLP problem

$$\begin{aligned} \min_{y,z,\lambda} \quad & f_u = 16y^2 + 9z^2 \\ s.t \quad & 4(y + z - 20)^3 + \lambda = 0, \\ & \lambda - (4y + z - 50) - \sqrt{(\lambda + 4y + z - 50)^2 + 4\tilde{\epsilon}^2} = 0, \\ & -4y + z \leq 0, \\ & y \geq 0, \quad z \geq 0. \end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y^*, z^*) = (11.25, 5)$, $f_u = 2250$, and $f_l = 197.753$.

Test Problem 10 [23]:

$$\begin{aligned} \min_{y_1} \quad & f_u = y_1^3 z_1 + z_2 \\ s.t. \quad & 0 \le y_1 \le 1, \\ \min_{z_1, z_2} \quad & f_l = -z_2 \\ s.t. \quad & y_1 z_1 \le 10, \\ & z_1^2 + y_1 z_2 \le 1, \\ & z_2 \ge 0. \end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 10 is converted to the following SNLP problem

$$\begin{aligned} \min_{y_1, z_1, z_2, \lambda_1, \lambda_2} \quad & f_u = y_1^3 z_1 + z_2 \\ s.t. \quad & \lambda_1 y_1 + 2\lambda_2 z_1 = 0, \\ & -1 + \lambda_2 y_1 = 0, \\ & \lambda_1 - (y_1 z_1 - 10) - \sqrt{(\lambda_1 + y_1 z_1 - 10)^2 + 4\tilde{\epsilon}^2} = 0, \\ & \lambda_2 - (z_1^2 + y_1 z_2 - 1) - \sqrt{(\lambda_2 + z_1^2 + y_1 z_2 - 1)^2 + 4\tilde{\epsilon}^2} = 0, \\ & 0 \le y_1 \le 1, \\ & z_2 \ge 0. \end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, z_1^*, z_2^*) = (1, 0, 0)$, $f_u = 0$, and $f_l = 0$.

Test Problem 11 [40]:

$$\begin{aligned} \min_{y_1, y_2} \quad & f_u = 2y_1 + 2y_2 - 3z_1 - 3z_2 - 60 \\ s.t. \quad & y_1 + y_2 + z_1 - 2z_2 \le 40, \\ & 0 \le y_1 \le 50, \\ & 0 \le y_2 \le 50, \\ \min_{z_1, z_2} \quad & f_l = (z_1 - y_1 + 20)^2 + (z_2 - y_2 + 20)^2, \\ s.t. \quad & y_1 - 2z_1 \ge 10, \\ & y_2 - 2z_2 \ge 10, \\ & -10 \le z_1 \le 20, \\ & -10 \le z_2 \le 20. \end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 11 is converted to the following smooth nonlinear programming problem

$$
\begin{aligned}
\min_{y_1,y_2,z_1,z_2,\lambda_1,\lambda_2} \quad & f_u = 2y_1 + 2y_2 - 3z_1 - 3z_2 - 60 \\
s.t. \quad & 2(z_1 - y_1 + 20) + 2\lambda_1 = 0, \\
& 2(z_2 - y_2 + 20) + 2\lambda_2 = 0, \\
& \lambda_1 - (10 - y_1 + 2z_1) - \sqrt{(\lambda_1 + 10 - y_1 + 2z_1)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_2 - (10 - y_2 + 2z_2) - \sqrt{(\lambda_2 + 10 - y_2 + 2z_2)^2 + 4\tilde{\epsilon}^2} = 0, \\
& y_1 + y_2 + z_1 - 2z_2 \leq 40, \\
& 0 \leq y_1 \leq 50, \\
& 0 \leq y_2 \leq 50, \\
& -10 \leq z_1 \leq 20, \\
& -10 \leq z_2 \leq 20.
\end{aligned}
$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, y_2^*, z_1^*, z_2^*) = (25, 30, 4.9996, 10)$, $f_u = 5.0024$, and $f_l = 1.5000e - 06$.

Test Problem 12 [23]:

$$
\begin{aligned}
\min_{y} \quad & f_u = (y-3)^2 + (z-2)^2 \\
s.t. \quad & -2y + z - 1 \leq 0, \\
& y - 2z + 2 \leq 0, \\
& y + 2z - 14 \leq 0, \\
& 0 \leq y \leq 8, \\
\min_{z} \quad & f_l = (z-5)^2 \\
s.t. \quad & z \geq 0.
\end{aligned}
$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 12 is converted to the following SNLP problem

$$
\begin{aligned}
\min_{y,z} \quad & f_u = (y-3)^2 + (z-2)^2 \\
s.t. \quad & 2(z-5) = 0, \\
& -2y + z - 1 \leq 0, \\
& y - 2z + 2 \leq 0, \\
& y + 2z - 14 \leq 0, \\
& 0 \leq y \leq 8, \\
& z \geq 0.
\end{aligned}
$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y^*, z^*) = (3, 5)$, $f_u = 9$, and $f_l = 0$.

Test Problem 13 [40]:

$$
\begin{aligned}
\min_{y_1,y_2} \quad & f_u = -y_1^2 - 3y_2^2 - 4z_1 + z_2^2 \\
s.t. \quad & y_1^2 + 2y_2 \leq 4, \\
& y_1 \geq 0, \quad y_2 \geq 0, \\
\min_{z_1,z_2} \quad & f_l = 2y_1^2 + z_1^2 - 5z_2, \\
s.t. \quad & y_1^2 - 2y_1 + 2y_2^2 - 2z_1 + z_2 \geq -3, \\
& y_2 + 3z_1 - 4z_2 \geq 4, \\
& z_1 \geq 0, \quad z_2 \geq 0.
\end{aligned}
$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 13 is converted to the following SNLP problem.

$$\begin{aligned}
\min_{y_1,y_2,z_1,z_2,\lambda_1,\lambda_2} \quad & f_u = -y_1^2 - 3y_2^2 - 4z_1 + z_2^2 \\
s.t. \quad & 2z_1 + 2\lambda_1 - 3\lambda_2 = 0, \\
& -5 - \lambda_1 + 4\lambda_2 = 0, \\
& \lambda_1 - (-y_1^2 + 2y_1 - 2y_2^2 + 2z_1 - z_2 - 3) - \sqrt{(\lambda_1 - y_1^2 + 2y_1 - 2y_2^2 + 2z_1 - z_2 - 3)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_2 - (-y_2 - 3z_1 + 4z_2 - 4) - \sqrt{(\lambda_2 - y_2 - 3z_1 + 4z_2 - 4)^2 + 4\tilde{\epsilon}^2} = 0, \\
& y_1^2 + 2y_2 \leq 4, \\
& y_1 \geq 0, \quad y_2 \geq 0, \\
& z_1 \geq 0, \quad z_2 \geq 0.
\end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, y_2^*, z_1^*, z_2^*) = (0, 2, 1.875, 0.90625)$, $f_u = -18.679$, and $f_l = -1.0156$.

Test Problem 14 [40]:

$$\begin{aligned}
\min_{y} \quad & f_u = (y-1)^2 + (z-1)^2 \\
s.t. \quad & y \geq 0, \\
\min_{z} \quad & f_l = 0.5z^2 + 500z - 50yz \\
s.t. \quad & z \geq 0.
\end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 14 is converted to the following SNLP problem.

$$\begin{aligned}
\min_{y,z} \quad & f_u = (y-1)^2 + (z-1)^2 \\
s.t. \quad & z - 50y + 500 = 0 \\
& y \geq 0, \quad z \geq 0.
\end{aligned}$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y^*, z^*) = (10.016, 0.81967)$, $f_u = 81.328$, and $f_l = -0.33593$.

Test Problem 15 [40]:

$$\begin{aligned}
\min_{y_1,y_2} \quad & f_u = -8y_1 - 4y_2 + 4z_1 - 40z_2 - 4z_3 \\
s.t. \quad & y_1 \geq 0, \quad y_2 \geq 0 \\
\min_{z} \quad & f_l = y_1 + 2y_2 + z_1 + z_2 + 2z_3, \\
s.t. \quad & z_2 + z_3 - z_1 \leq 1, \\
& 2y_1 - z_1 + 2z_2 - 0.5z_3 \leq 1, \\
& 2y_2 + 2z_1 - z_2 - 0.5z_3 \leq 1, \\
& z_i \geq 0, \quad i = 1, 2, 3.
\end{aligned}$$

Applying the Karush-Kuhn-Tucker condition to the lower level problem and using the CHKS smoothing function, Test Problem 15 is converted to the following SNLP problem,

$$
\begin{aligned}
\min_{y_1,y_2,z_1,z_2,z_3,\lambda_1,\lambda_2,\lambda_3} \quad & f_u = -8y_1 - 4y_2 + 4z_1 - 40z_2 - 4z_3 \\
s.t. \quad & 1 - \lambda_1 - \lambda_2 + 2\lambda_3 = 0, \\
& 1 + \lambda_1 + 2\lambda_2 - \lambda_3 = 0, \\
& 2 + \lambda_1 - 0.5\lambda_2 - 0.5\lambda_3 = 0, \\
& \lambda_1 - (z_2 + z_3 - z_1 - 1) - \sqrt{(\lambda_1 + z_2 + z_3 - z_1 - 1)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_2 - (2y_1 - z_1 + 2z_2 - 0.5z_3 - 1) - \sqrt{(\lambda_2 + 2y_1 - z_1 + 2z_2 - 0.5z_3 - 1)^2 + 4\tilde{\epsilon}^2} = 0, \\
& \lambda_3 - (2y_2 + 2z_1 - z_2 - 0.5z_3 - 1) - \sqrt{(\lambda_3 + 2y_2 + 2z_1 - z_2 - 0.5z_3 - 1)^2 + 4\tilde{\epsilon}^2} = 0, \\
& y_1 \geq 0, \quad y_2 \geq 0, \quad z_i \geq 0, \quad i = 1, 2, 3.
\end{aligned}
$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have that $(y_1^*, y_2^*, z_1^*, z_2^*, z_3^*) = (0, 0.9, 0, 0.6, 0.4)$, $f_u = -29.2$, and $f_l = 3.2$.

4.2. Practical example

In this section, the efficacy of the proposed Algorithm (3.8) was tested by using a watershed water trading decision-making model based on bilevel programming [42]. The upper decision-maker is the watershed management agency which acts as the planning, controlling and coordinating center of a watershed, and each user is the lower decision-maker. The mathematical formulation for the watershed water trading decision-making model is formulated as follows:

$$
\begin{aligned}
\max_{w,t,r_1,g_1,r_2,g_2} \quad & F = 0.4W + t(q_1 + q_2) + f_1 + f_2, \\
s.t. \quad & r_1 + r_2 + w = 90, \\
& q_1 + q_2 + w \leq 90, \\
& g_1 + g_2 = 20, \\
& r_1 \geq 38,\ r_2 \geq 42,\ g_1 \geq 7,\ g_2 \geq 8,\ w \geq 6,\ 0.3 \leq t \leq 2.0, \\
\max_{q_1,l_1} \quad & f_1 = 0.7q_1 - q_1 t - 0.3(45 - q_1)^2 + (r_1 - q_1)[0.9 - 0.01(r_1 + r_2 - q_1 - q_2)], \\
& -0.2(0.2q_1 - l_1)^2 + (g_1 - l_1)[0.8 - 0.01(g_1 + g_2 - l_1 - l_2)], \\
\max_{q_2,l_2} \quad & f_2 = 0.8q_2 - q_2 t - 0.2(47 - q_2)^2 + (r_2 - q_2)[0.9 - 0.01(r_1 + r_2 - q_1 - q_2)], \\
& -0.1(0.3q_2 - l_2)^2 + (g_2 - l_2)[0.8 - 0.01(g_1 + g_2 - l_1 - l_2)], \\
s.t. \quad & l_1 + l_2 \leq 20, \\
& q_1,\ l_1 \geq 0, \\
& q_2,\ l_2 \geq 0
\end{aligned}
$$

where q_1 and q_2 are the actual water intake volumes of water consumer A and water consumer B respectively. l_1 and l_2 are the wastewater discharge volumes of two users respectively. r_1 and r_2 are the water rights of the two users respectively. g_1 and g_2 are the emission rights of two users respectively. w is the ecological water requirement of the watershed. t represents the water resource fees. Applying the Karush-Kuhn-Tucker condition to the lower-level problem and using the CHKS smoothing function, the watershed water trading decision-making model is converted to the following SNLP problem,

$$
\begin{aligned}
\max_{w,t,r_1,g_1,r_2,g_2} \quad & F = 0.4W + t(q_1 + q_2) + f_1 + f_2, \\
s.t. \quad & r_1 + r_2 + w = 90, \\
& g_1 + g_2 = 20, \\
& 0.8 - t + 0.4(47 - q_2) + 0.01(r_2 - q_2) - (0.9 - 0.01(r_1 + r_2 - q_1 - q_2)) \\
& -0.06(0.3q_2 - l_2) = 0, \\
& 0.2(0.3q_2 - l_2) + 0.01(g_2 - l_2) - (0.8 - 0.01(g_1 + g_2 - l_1 - l_2)) - \lambda_1 = 0, \\
& \lambda_1 - (l_1 + l_2 - 20) - \sqrt{(\lambda_1 + l_1 + l_2 - 20)^2 + 4\tilde{\epsilon}^2} = 0, \\
& 0.7 - t + 0.6(45 - q_1) + 0.01(r_1 - q_1) - (0.9 - 0.01(r_1 + r_2 - q_1 - q_2)) \\
& -0.8(0.2q_1 - l_1) - 0.01\lambda_2 = 0, \\
& 0.4(0.2q_1 - l_1) + 0.01(g_1 - l_1) - (0.8 - 0.01(g_1 + g_2 - l_1 - l_2)) - 0.01\lambda_3 \\
& +\lambda_4(-1 - \frac{\lambda_1 + l_1 + l_2 - 20}{\sqrt{(\lambda_1 + l_1 + l_2 - 20)^2 + 4\tilde{\epsilon}^2}}) = 0, \\
& q_1 + q_2 + w \leq 90, \\
& g_1 \geq 7, g_2 \geq 8, r_1 \geq 38, r_2 \geq 42, w \geq 6, 0.3 \leq t \leq 2.0, \\
& q_1,\ l_1 \geq 0, \\
& q_2,\ l_2 \geq 0.
\end{aligned}
$$

Applying Algorithm (3.7) to the above nonlinear programming problem we have the following data at $\tilde{\epsilon} = 0.001$. $g_1 = 9$, $g_2 = 10.450$, $l_1 = 6.8755$, $l_2 = 9.0751$, $q_1 = 41.497$, $q_2 = 43$, $r_1 = 39.332$, $r_2 = 44.269$, $t = 0.3$, $w = 6.2338$, $f_1 = 12.17$, $f_2 = 19.043$, and $F = 59.055$.

Table 3 compares the obtained results with method [25] and those of various existing methods(Ref). From Table 3, we can see that the solutions obtained via Algorithm (3.8) are better than those given in [25] and those of various existing methods(Ref). Therefore, Algorithm (3.8) can also be applied to the practical problem.

Table 3. Comparison of the results of Algorithm (3.8) with method [25] and with those of various existing methods(ref) for a watershed water application.

Parameter	Method in [25]	Algorithm (3.8)	Results in Ref.
q_1	42	41.497	41.2016
q_2	41.9680	43	42.5388
l_1	6.9984	6.8755	6.4772
l_2	9.1751	9.0751	9.1611
r_1	39.9679	39.332	39.4861
r_2	44	44.269	44.2542
g_1	9	9	9.0015
g_2	11	10.45	10.9985
w	6.0321	6.2338	6.2596
t	0.3000	0.3	0.3226
F	58.97837	59.055	58.4482
f_1	13.40286	12.17	10.964
f_2	17.97226	19.043	17.964

5. Conclusions

In this paper, Algorithm (3.8) has been presented to solve the NBLP problem (2.1). A Karush-Kuhn-Tucker condition is used with the CHKS smoothing function to convert the NBLP problem (2.1) into a standard SNLP problem. The SNLP problem is solved by using the proposed nonmonton active interior-point trust-region technique to obtain an approximately optimal solution to the NBLP problem. In the proposed nonmonton active interior-point trust-region technique, the smoothed nonlinear programming problem is transformed into an equality-constrained optimization problem with bounded variables by using an active-set approach and the penalty method. To solve the equality-constrained optimization problem with bounded variables, Newton's interior-point method is used. But it is a local method so it might not converge if the starting point is far from a stationary point. To overcome this problem, the nonmonotone trust-region strategy is used. It is generally promising and efficient, and it can overcome the aforementioned shortcomings.

Applications to mathematical programs with equilibrium constraints are given to clarify the effectiveness of the proposed approach. Numerical results reflect the good behavior of the proposed technique and the computed results converge to the optimal solutions. It is clear from the comparison between the solutions obtained by using Algorithm (3.8) and the algorithm of [18], that Algorithm (3.8) is able to find the optimal solution of some problems with fewer iterations, fewer function evaluations, and less time.

Furthermore, the usefulness of the CHKS smoothing function with the nonmonton active interior-point trust-region algorithm in efforts to solve NBLP problems was examined by using a real-world case about a watershed trading decision-making problem. Numerical experiments show that the suggested method surpasses rival algorithms in terms of efficacy.

Use of AI tools declaration

The authors declare that they have not used artificial intelligence tools in the creation of this article.

Conflict of interest

The authors declare that they have no competing interests.

References

1. M. A. Amouzegar, A global optimization method for nonlinear bilevel programming problems, *IEEE Trans. Syst. Men Cybernet.*, **29** (1999), 771–777. https://doi.org/10.1109/3477.809031
2. R. Byrd, Omojokun, Robust trust-region methods for nonlinearly constrained optimization, In: *Second SIAM Conference on Optimization, Houston*, 1987.
3. J. F. Bard, Coordination of a multidivisional organization through two levels of management, *Omega*, **11** (1983), 457–468. https://doi.org/10.1016/0305-0483(83)90038-5
4. J. F. Bard, Convex two-level optimization, *Math. Program.*, **40** (1988), 15–27. https://doi.org/10.1007/BF01580720

5. B. Chen, P. T. Harker, A non-interior-point continuation method for linear complementarity problem, *SIAM J. Matrix Anal. Appl.*, **14** (1993), 1168–1190. https://doi.org/10.1137/0614081
6. I. Das, An interior point algorithm for the general nonlinear programming problem with trust region globlization, In: *Technical Report*, 1996.
7. J. Dennis, M. Heinkenschloss, L. Vicente, Trust-region interior-point SQP algorithms for a class of nonlinear programming problems, *SIAM J. Control Optim.*, **36** (1998), 1750–1794. https://doi.org/10.1137/S0036012995279031
8. J. Dennis, M. El-Alem, K. Williamson, A trust-region approach to nonlinear systems of equalities and inequalities, *SIAM J. Optim.*, **9** (1999), 291–315. https://doi.org/10.1137/S1052623494276208
9. N. Y. Deng, Y. Xiao, F. J. Zhou, Nonmonotonic trust region algorithm, *J. Optim. Theory Appl.*, **76** (1993), 259–285. https://doi.org/10.1007/BF00939608
10. B. El-Sobky, A multiplier active trust-region algorithm for solving general nonlinear programming problem, *Appl. Math. Comput.*, **219** (2012), 928–946.
11. B. El-Sobky, An interior-point penalty active-set trust-region algorithm, *J. Egypt. Math. Soc.*, **24** (2016), 672–680. https://doi.org/10.1016/j.joems.2016.04.003
12. B. El-Sobky, An active-set interior-point trust-region algorithm, *Pacific J. Optim.*, **14** (2018), 125–159.
13. B. El-Sobky, A. Abotahoun, An active-set algorithm and a trust-region approach in constrained minimax problem, *Comp. Appl. Math.*, **37** (2018), 2605–2631. https://doi.org/10.1007/s40314-017-0468-3
14. B. El-Sobky, A. Abotahoun, A trust-region algorithm for solving mini-max problem, *J. Comput. Math.*, **36** (2018), 881–902.
15. B. El-Sobky, Y. Abouel-Naga, A penalty method with trust-region mechanism for nonlinear bilevel optimization problem, *J. Comput. Appl. Math.*, **340** (2018), 360–374. https://doi.org/10.1016/j.cam.2018.03.004
16. B. El-Sobky, Y.Abo-Elnaga, A. Mousa, A. El-Shorbagy, Trust-region based penalty barrier algorithm for constrained nonlinear programming problems: an application of design of minimum cost canal sections, *Mathematics*, **9** (2021), 1551. https://doi.org/10.3390/math9131551
17. B. El-Sobky, G. Ashry, An interior-point trust-region algorithm to solve a nonlinear bilevel programming problem, *AIMS Math.*, **7** (2022), 5534–5562. http://dx.doi.org/10.3934/math.2022307
18. B. El-Sobky, G. Ashry, An Active-set Fischer-Burmeister trust-region algorithm to solve a nonlinear bilevel optimization problem, *Fractal Fract.*, **6** (2022), 412. https://doi.org/10.3390/fractalfract6080412
19. B. El-Sobky, G. Ashry, Y. Abo-Elnaga, An active-set with barrier method and trust-region mechanism to solve a nonlinear bilevel programming problem, *AIMS Math.*, **7** (2022), 16112–16146. http://dx.doi.org/10.3934/math.2022882
20. B. El-Sobky, M. F. Zidan, A trust-region based an active-set interior-point algorithm for fuzzy continuous Static Games, *AIMS Math.*, **8** (2023), 13706–13724. http://dx.doi.org/10.3934/math.2023696

21. J. B. E. Etoa, Solving quadratic convex bilevel programming problems using a smoothing method, *Appl. Math. Comput.*, **217** (2011), 6680–6690. https://doi.org/10.1016/j.amc.2011.01.066

22. J. E. Falk, J. M. Liu, On bilevel programming, Part I: general nonlinear cases, *Math. Program.*, **70** (1995), 47–72. https://doi.org/10.1007/BF01585928

23. H. Gumus, A. Flouda, Global optimization of nonlinear bilevel programming problems, *J. Global Optim.*, **20** (2001), 1–31. https://doi.org/10.1023/A:1011268113791

24. Y. Ishizuka, E. Aiyoshi, Double penalty method for bilevel optimization problems, *Ann. Oper. Res.*, **34** (1992), 73–88. https://doi.org/10.1007/BF02098173

25. Y. Jiang, X. Li, C. Huang, X. Wu, Application of particle swarm optimization based on CHKS smoothing function for solving nonlinear bilevel programming problem, *Appl. Math. Comput.*, **219** (2013), 4332–4339. https://doi.org/10.1016/j.amc.2012.10.010

26. C. Kanzow, Some noninterior continuation methods for linear complementarity problems, *SIAM J. Matrix Anal. Appl.*, **17** (1996), 851–868. https://doi.org/10.1137/S0895479894273134

27. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, *Science*, **220** (1983), 671–680. https://doi.org/10.1126/science.220.4598.671

28. H. Li, Y. Jiao, L. Zhang, Orthogonal genetic algorithm for solving quadratic bilevel programming problems, *J. Syst. Eng. Elect.*, **21** (2010), 763–770. https://doi.org/10.3969/j.issn.1004-4132.2010.05.008

29. Y. B. Lv, T. S. Hu, G. M. Wang, Z. P. Wan, A penalty function method based on Kuhn-Tucker condition for solving linear bilevel programming, *Appl. Math. Comput.*, **188** (2007) 808–813. https://doi.org/10.1016/j.amc.2006.10.045

30. D. Muu, N. Quy, A global optimization method for solving convex quadratic bilevel programming problems, *J. Global Optim.*, **26** (2003), 199–219. https://doi.org/10.1023/A:1023047900333

31. J. Mo, C. Liu, S. Yan, A nonmonotone trust region method based on nonincreasing technique of weighted average of the successive function value, *J. Comput. Appl. Math.*, **209** (2007), 97–108. https://doi.org/10.1016/j.cam.2006.10.070

32. E. Omojokun, Trust-region strategies for optimization with nonlinear equality and inequality constraints, PhD thesis, *Department of Computer Science, University of Colorado, Boulder, Colorado*, 1989.

33. V. Oduguwa, R. Roy, Bi-level optimization using genetic algorithm, In: *Proceedings 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002)*, 2002, 123–128. https://doi.org/10.1109/ICAIS.2002.1048121

34. T. Steihaug, The conjugate gradient method and trust-region in large scale optimization, *SIAM J. Numer. Anal.*, **20** (1983), 0720042. https://doi.org/10.1137/0720042

35. S. Smale, Algorithms for solving equations, In: *Proceeding of International Congress of Mathematicians, American Mathematics Society, Rhode Island*, 1987, 72–195.

36. G. Savard, J. Gauvin, The steepest descent direction for the nonlinear bilevel programming problem, *Oper. Res. Lett.*, **15** (1994), 265–272. https://doi.org/10.1016/0167-6377(94)90086-8

37. Z. J. Shi, J. H. Guo, A new trust region methods for unconstrained optimization, *J. Comput. Appl. Math.*, **213** (2008), 509–520. https://doi.org/10.1016/j.cam.2007.01.027

38. P. L. Toint, Non-monotone trust-region algorithm for nonlinear optimization subject to convex constraints, *Math. Program.*, **77** (1997), 69–94. https://doi.org/10.1007/BF02614518

39. Y. Yuan, On the convergence of a new trust region algorithm, *Numer. Math.*, **70** (1995), 515–539. https://doi.org/10.1007/s002110050133

40. Y. Wang, Y. Jiao, H. Li, An evolutionary algorithm for solving nonlinear bilevel programming based on a new constraint-Handling scheme, *IEEE T. Syst. Man Cy. C*, **35** (2005), 221–232. https://doi.org/10.1109/TSMCC.2004.841908

41. G. M. Wang, X. J. Wang, Z. P. Wan, Y. B. Lv, A globally convergent algorithm for a class of bilevel nonlinear programming problem, *Appl. Math. Comput.*, **188** (2007), 166–172. https://doi.org/10.1016/j.amc.2006.09.130

42. C. Y. Wu, Y. Z. Zhao, Watershed water trading decision-making model based on bilevel programming, *Oper. Res. Manage. Sci.*, in Chinese, **20** (2011), 30–37.

43. X. S. Zhang, J. L. Zhang, L. Z. Liao, A nonmonotone adaptive trust region method and its convergence, *Comput. Math. Appl.*, **45** (2003), 1469–1477. https://doi.org/10.1016/S0898-1221(03)00130-5

44. H. C. Zhang, W. W. Hager, A nonmonotone line search technique for unconstrained optimization, *SIAM J. Optim.*, **14** (2004), 1043–1056.

AIMS Press

© 2024 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)