Computational Optimization and Applications
https://doi.org/10.1007/s10589-021-00285-4

An effective logarithmic formulation for piecewise linearization requiring no inequality constraint

F. J. Hwang[1] · Yao-Huei Huang[2]

Received: 22 August 2020 / Accepted: 19 May 2021

Abstract
One of the commonly used techniques for tackling the nonconvex optimization problems in which all the nonlinear terms are univariate is the piecewise linear approximation by which the nonlinear terms are reformulated. The performance of the linearization technique primarily depends on the quantities of variables and constraints required in the formulation of a piecewise linear function. The state-of-the-art linearization method introduces $2\lceil \log_2 m \rceil$ inequality constraints, where m is the number of line segments in the constructed piecewise linear function. This study proposes an effective alternative logarithmic scheme by which no inequality constraint is incurred. The price that more continuous variables are needed in the proposed scheme than in the state-of-the-art method is less than offset by the simultaneous inclusion of a system of equality constraints satisfying the canonical form and the absence of any inequality constraint. Our numerical experiments demonstrate that the developed scheme has the computational superiority, the degree of which increases with m.

Keywords Nonlinear programming · Nonconvex optimization · Piecewise linearization · Logarithmic method · Inequality constraint

✉ Yao-Huei Huang
yaohuei.huang@gmail.com

F. J. Hwang
feng-jang.hwang@uts.edu.au

1 School of Mathematical and Physical Sciences, Transport Research Centre, University of Technology Sydney, Ultimo 2007, Australia

2 Department of Information Management, Fu Jen Catholic University, New Taipei City 24205, Taiwan

Published online: 01 June 2021

1 Introduction

The piecewise linear approximation has been widely applied in various nonconvex optimization problems such as the supply chain management problems [40], network flow problems [1, 2, 7–9, 11], network loading problems [6, 13, 15, 33], facility location problems [10, 17, 18], packing and assortment problems [20, 24, 25, 41, 42], electronic circuit design problems [14], and engineering optimization problems [30, 31]. The nonconvex optimization problems considered herein are the nonlinear programming (NLP) problems in which all the nonlinear terms are univariate. Numerous piecewise linearization techniques for the nonconvex NLP problems have been proposed to yield an approximate global optimal solution [3, 12, 16, 23, 28, 29, 32, 34, 38, 44]. One popular category of the methodologies is to utilize the convex combination reformulation by introducing extra continuous variables, binary variables, and linear constraints. The key to the development of piecewise linearization using the convex combination formulation is the design of variables composing special ordered sets of type 2 (SOS2), where (1) at most two variables can be nonzero, and (2) if two variables are nonzero, they must be adjacent in the ordering [4]. Different convex combination formulations having distinct SOS2 designs would bring about different quantities of extra variables as well as constraints in the derived mixed integer linear programming (MILP) model, and thus retain dissimilar computational capabilities. Comparing the competing formulation approaches to univariate piecewise linearization, this study aims at an effective alternative convex combination formulation attaining enhanced computational efficiency in the solving of the formulated MILP model.

Regarding the convex combination formulation of piecewise linear functions, the conventional approach [12] in the seminal mathematical programming textbooks [3, 16] constructs the SOS2 variables by introducing extra m binary variables and m inequality constraints, where m is the number of line segments in the constructed piecewise linear function. Although the conventional formulation is straightforward and effective, considerable numbers of extra inequality constraints and binary variables could impose a heavy computational burden on the formulated MILP solving. To improve the computational performance, several reformulation methods such as the multiple-choice formulation [21] and disaggregated formulation [36] had been presented. Readers are referred to Sridhar et al. [37] and Rebennack [38] for reviews on the popular models, including the incremental formulation [34, 35], which does not belong to the convex combination category, and this paper concerns the logarithmic-type convex combination formulation, which is one of the most recent research focuses. Li et al. [28] developed a logarithmic procedure which reduces the required numbers of binary variables and constraints to the numbers logarithmic in m. Then Vielma et al. [43] showed that the piecewise linearization modeling of Li et al. [28] could yield a poor MILP formulation, which is even computationally inferior to that using the conventional method. Vielma and Nemhauser [44] later proposed an advanced logarithmic procedure which introduces far fewer variables and constraints than the conventional formulation or the method of Li et al. [28]. The state-of-the-art logarithmic method of Vielma and Nemhauser [44] (hereafter referred to as the logarithmic method for brevity), though outperformed by the incremental formulation in some instances [38], was shown to have generally the computational advantage over other existing models. Hence, it would be well worth investigating the possibility that the required number of extra variables or constraints can be further reduced.

This study, which is derived from our unpublished research work [19], attempts to develop an alternative reformulation technique that involves fewer extra inequality constraints than and is computationally superior to the reference methods, including the conventional formulation, incremental formulation, and logarithmic method. The advantages of the proposed technique are listed as follows.

1. The proposed formulation, in contrast to existing models, incurs no inequality constraint in piecewise linearization, and the binary variables required are as few as those in the logarithmic method.
2. Although more continuous variables are introduced in the proposed scheme than in the logarithmic method, the price is less than offset by the simultaneous inclusion of a system of equality constraints satisfying the canonical form, where each equality constraint has an isolated variable, and the absence of any inequality constraint.
3. Demonstrated by the numerical experiments, the developed scheme has the computational superiority, the degree of which increases with m.

The remainder of the paper is organized as follows. In Sect. 2, the three foregoing reference piecewise linearization models are described. The proposed linearization technique is introduced in Sect. 3. Numerical experiments are presented in Sect. 4, and Sect. 5 provides the concluding remarks.

2 Reference models

This section introduces three reference models, including the conventional formulation, incremental formulation, and logarithmic method for approximating a univariate nonconvex function using a piecewise linear function. Being considered as a reference model, the conventional formulation is used as a baseline for the theoretical and computational comparison. Furthermore, the incremental formulation is included to demonstrate the difference between the linearization in the convex combination category and that of incremental type. Consider a nonconvex function $f(x)$ of a single variable $x \in [\underline{x}, \overline{x}] \subset \mathbb{R}$. Assume that the domain $[\underline{x}, \overline{x}]$ is divided into m intervals by setting $m+1$ argument values, i.e. breaking points, α_l, $\forall l \in \{0, 1, \dots, m\}$ such that $\alpha_0 = \underline{x} < \alpha_1 < \cdots < \alpha_m = \overline{x}$. Denote by $L(f(x))$ a piecewise linear function obtained from linearizing $f(x)$.

(a) *Conventional formulation* [3, 12, 16]

The variable x and function $L(f(x))$ are expressed respectively as

$$x = \sum_{l=0}^{m} \alpha_l u_l, \tag{1}$$

$$L(f(x)) = \sum_{l=0}^{m} f(\alpha_l) u_l, \tag{2}$$

where continuous variables $u_l \geq 0,\ \forall l \in \{0, 1, \ldots, m\}$ satisfy

$$\sum_{l=0}^{m} u_l = 1, \tag{3}$$

and auxiliary binary variables as well as linear constraints to make variables $u_l,\ \forall l \in \{0, 1, \ldots, m\}$ constitute SOS2 are required.

Denote $M = \{1, 2, \ldots, m\}$. By introducing a set of m binary variables $w_l \in \{0, 1\},\ \forall l \in M$, the conventional formulation to govern the SOS2 construction for variables $u_l,\ \forall l \in M \cup \{0\}$ is shown as follows:

$$\begin{aligned} & u_{l-1} + u_l \geq w_l, \quad \forall l \in M, \\ & \sum_{l=1}^{m} w_l = 1. \end{aligned}$$

(b) *Incremental formulation* [34, 35]

Employing m continuous variables u_l satisfying $0 \leq u_l \leq \alpha_l - \alpha_{l-1},\ \forall l \in M$ and $m-1$ binary variables $w_l \in \{0, 1\},\ \forall l \in M \backslash \{m\}$, the incremental formulation constructs the piecewise linearization as follows:

$$\begin{aligned} & x = \alpha_0 + \sum_{l=1}^{m} u_l, \\ & L(f(x)) = f(\alpha_0) + \sum_{l=1}^{m} \frac{f(\alpha_l) - f(\alpha_{l-1})}{\alpha_l - \alpha_{l-1}} u_l, \\ & u_l \geq (\alpha_l - \alpha_{l-1}) w_l, \quad \forall l \in M \backslash \{m\}, \\ & u_{l+1} \leq (\alpha_{l+1} - \alpha_l) w_l, \quad \forall l \in M \backslash \{m\}. \end{aligned}$$

(c) *Logarithmic method* [44]

Define an injective function $\boldsymbol{\theta}: M \rightarrow \{0, 1\}^{\lceil \log_2 m \rceil}$, where the vectors $\boldsymbol{\theta}(l)$ and $\boldsymbol{\theta}(l+1)$ differ in exactly one element for all $l \in M \backslash \{m\}$. Denote also $G = \{1, 2, \ldots, \lceil \log_2 m \rceil\}$. Let $\boldsymbol{\theta}(l) = (\theta_1^l, \theta_2^l, \ldots, \theta_{\lceil \log_2 m \rceil}^l)$, where

$\theta_k^l \in \{0,1\}$, $\forall k \in G$, $\forall l \in M \cup \{0\}$ and $\boldsymbol{\theta}(0) = \boldsymbol{\theta}(1)$. Then the two sets $S^+(k)$ and $S^-(k)$ are defined as follows:

1. $S^+(k) = \left\{l\colon l \in M\backslash\{m\} \text{ and } \theta_k^l = \theta_k^{l+1} = 1\right\} \cup \left\{l\colon l \in \{0,m\} \text{ and } \theta_k^l = 1\right\}$;
2. $S^-(k) = \left\{l\colon l \in M\backslash\{m\} \text{ and } \theta_k^l = \theta_k^{l+1} = 0\right\} \cup \left\{l\colon l \in \{0,m\} \text{ and } \theta_k^l = 0\right\}$.

Utilizing a set of $\lceil \log_2 m \rceil$ binary variables $v_k \in \{0,1\}$, $\forall k \in G$, the logarithmic method formulates SOS2 for variables u_l, $\forall l \in M \cup \{0\}$ in Eqs. (1)–(3) by virtue of the following linear inequalities:

$$\sum_{l \in S^+(k)} u_l \le v_k, \quad \forall k \in G, \tag{4}$$

$$\sum_{l \in S^-(k)} u_l \le 1 - v_k, \quad \forall k \in G. \tag{5}$$

Then Eqs. (1)–(5) together form the piecewise linearization with the logarithmic method.

We note that the conventional formulation and incremental model use extra m binary variables coupled with m inequality constraints and extra $m-1$ binary variables coupled with $2(m-1)$ inequality constraints, respectively, while the logarithmic method utilizes extra $\lceil \log_2 m \rceil$ binary variables coupled with $2\lceil \log_2 m \rceil$ inequality constraints only. More detailed comparisons will be provided in the next section.

3 Proposed linearization method

In this section, we present an alternative reformulation technique requiring no extra inequality constraint for achieving the SOS2 construction in the piecewise linearization. Our method is inspired by the following design proposed by Li et al. [27] for yielding the SOS1 construction.

Remark 1 (Analog of Theorem 1 in Li et al. [27]) Assume that binary numbers $b_{lk} \in \{0,1\}, \forall l \in M, \forall k \in G$ satisfy

$$\sum_{k=1}^{\lceil \log_2 m \rceil} 2^{k-1} b_{lk} = l - 1, \quad \forall l \in M. \tag{6}$$

An m-dimensional nonnegative vector $\mathbf{u} = (u_1, u_2, \dots, u_m)$ satisfying

$$\sum_{l=1}^{m} u_l = 1$$

is a binary vector if there exists a $\lceil \log_2 m \rceil$-dimensional binary vector $\boldsymbol{v} = (v_1, v_2, \dots, v_{\lceil \log_2 m \rceil}) \in \{0,1\}^{\lceil \log_2 m \rceil}$ satisfying

$$\sum_{l=1}^{m} u_l b_{lk} = v_k, \quad \forall k \in G.$$

The proof follows Theorem 1 in Li et al. [27].

Example 1 Given $m = 5$ (and thus $\lceil \log_2 m \rceil = 3$), we have $(b_{1,1}, b_{1,2}, b_{1,3}) = (0, 0, 0)$, $(b_{2,1}, b_{2,2}, b_{2,3}) = (1, 0, 0)$, $(b_{3,1}, b_{3,2}, b_{3,3}) = (0, 1, 0)$, $(b_{4,1}, b_{4,2}, b_{4,3}) = (1, 1, 0)$, and $(b_{5,1}, b_{5,2}, b_{5,3}) = (0, 0, 1)$ as per Eq. (6). According to Remark 1, we consider a non-negative vector $\mathbf{u} = (u_1, u_2, u_3, u_4, u_5)$ satisfying

$$u_1 + u_2 + u_3 + u_4 + u_5 = 1,$$

and a binary vector $\boldsymbol{v} = (v_1, v_2, v_3) \in \{0, 1\}^3$ satisfying

$$\begin{aligned} u_2 + u_4 &= v_1, \\ u_3 + u_4 &= v_2, \text{ and} \\ u_5 &= v_3. \end{aligned}$$

It is thus obvious that all the five feasible states for $\boldsymbol{v}$, viz. (0, 0, 0), (1, 0, 0), (0, 1, 0), (1, 1, 0), and (0, 0, 1), yield the five states (1, 0, 0, 0, 0), (0, 1, 0, 0, 0), (0, 0, 1, 0, 0), (0, 0, 0, 1, 0), and (0, 0, 0, 0, 1), respectively, for $\mathbf{u}$.

Then the following proposition can be derived from Remark 1.

Proposition 1 *Assume that binary numbers $b_{lk} \in \{0, 1\}, \forall l \in M, \forall k \in G$ satisfy* Eq. (6). *Consider two m-dimensional nonnegative vectors $\mathbf{u}^{(h)} = (u_1^{(h)}, u_2^{(h)}, \ldots, u_m^{(h)})$, $h = 1, 2$, and a $\lceil \log_2 m \rceil$-dimensional binary vector $\boldsymbol{v} = (v_1, v_2, \ldots, v_{\lceil \log_2 m \rceil}) \in \{0, 1\}^{\lceil \log_2 m \rceil}$. If the following system of linear equations*:

$$\sum_{l=1}^{m} \sum_{h=1}^{2} u_l^{(h)} = 1, \tag{7}$$

$$\sum_{l=1}^{m} \left(b_{lk} \sum_{h=1}^{2} u_l^{(h)} \right) = v_k, \quad \forall k \in G, \tag{8}$$

is satisfied, then

1. *there exists exactly one index $l' \in M$ such that $\sum_{h=1}^{2} u_{l'}^{(h)} = 1$, i.e. the vector $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ is a binary unit vector*;
2. *each of the m states of vector $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ corresponds to a unique state of vector $\boldsymbol{v}$.*

Proof Denote by $\mathbf{B} = \left(b_{lk}\right)_{m\times\lceil\log_2 m\rceil}$ the constant binary matrix constructed with b_{lk}, and by $\mathbf{B}_l = (b_{l,1}, b_{l,2}, \dots, b_{l,\lceil\log_2 m\rceil})$, $\forall l \in M$, the lth row of matrix $\mathbf{B}$. Then Eq. (8) indicates

$$(\mathbf{u}^{(1)} + \mathbf{u}^{(2)})\mathbf{B} = \boldsymbol{v}. \tag{9}$$

1. Given an arbitrary vector for $\boldsymbol{v}$, say $\boldsymbol{v} = \boldsymbol{v}'$, satisfying Eq. (8), we have the following two possible cases:
   1. $\boldsymbol{v}' = (0, \dots, 0)$
      Since $\mathbf{B}_1 = (0, \dots, 0)$ and $\mathbf{B}_l \neq (0, \dots, 0)$, $\forall l \in M\backslash\{1\}$, Eq. (7) implies $\sum_{h=1}^{2} u_1^{(h)} = 1$.
   2. $\boldsymbol{v}' \neq (0, \dots, 0)$
      Assume that in this case the set of the subscript indexes of the nonzero components in the binary vector $\boldsymbol{v}'$ is $H \subseteq G$, where $|H| \geq 1$, i.e. $v_k' = 1, \forall k \in H$, and $v_k' = 0, \forall k \in G\backslash H$. Thus we have

$$\sum_{l=1}^{m}\left(b_{lk}\sum_{h=1}^{2} u_l^{(h)}\right) = 1, \quad \forall k \in H, \tag{10}$$

$$\sum_{l=1}^{m}\left(b_{lk}\sum_{h=1}^{2} u_l^{(h)}\right) = 0, \quad \forall k \in G\backslash H, \tag{11}$$

   for $\boldsymbol{v}'$ as per Eq. (8). Suppose that vector $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ has more than one nonzero component and the set of the sequential indexes of the nonzero components in $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ is $J \subseteq M$, where $|J| > 1$, i.e. $\sum_{h=1}^{2} u_l^{(h)} > 0$, $\forall l \in J$. Then Eq. (7) implies $\sum_{l\in J}\sum_{h=1}^{2} u_l^{(h)} = 1$, and we have $\sum_{h=1}^{2} u_l^{(h)} = 0$, $\forall l \in M\backslash J$. To keep Eqs. (10) and (11) satisfied, we must have respectively $b_{lk} = 1$, $\forall l \in J$, $\forall k \in H$ and $b_{lk} = 0$, $\forall l \in J$, $\forall k \in G\backslash H$, which together imply the $|J|$ identical rows in matrix $\mathbf{B}$, viz. $\mathbf{B}_l$, $\forall l \in J$, and contradict the definition of matrix $\mathbf{B}$.

   Thus, the inference that $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ must have no more than one nonzero component, together with Eq. (7), implies that $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ must have exactly one nonzero component, the value of which is one. Since $\sum_{h=1}^{2} u_1^{(h)} = 1$ does not satisfy $\boldsymbol{v}' \neq (0, \dots, 0)$, we can conclude that there exists exactly one index $l' \in M\backslash\{1\}$ such that $\sum_{h=1}^{2} u_{l'}^{(h)} = 1$.
2. Since $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ is a binary unit vector, Eq. (9) shows that the vector $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ whose lth component equals 1 corresponds to the vector $\boldsymbol{v} = \mathbf{B}_l$, where $l \in M$.

□

Example 2 Considering $m = 5$, we have two nonnegative vectors $\mathbf{u}^{(h)} = (u_1^{(h)}, u_2^{(h)}, u_3^{(h)}, u_4^{(h)}, u_5^{(h)})$, $h = 1, 2$, a binary vector $\boldsymbol{v} = (v_1, v_2, v_3) \in \{0, 1\}^3$, and the following linear equations:

$$\sum_{h=1}^{2} u_1^{(h)} + \sum_{h=1}^{2} u_2^{(h)} + \sum_{h=1}^{2} u_3^{(h)} + \sum_{h=1}^{2} u_4^{(h)} + \sum_{h=1}^{2} u_5^{(h)} = 1$$

$$\sum_{h=1}^{2} u_2^{(h)} + \sum_{h=1}^{2} u_4^{(h)} = v_1$$

$$\sum_{h=1}^{2} u_3^{(h)} + \sum_{h=1}^{2} u_4^{(h)} = v_2$$

$$\sum_{h=1}^{2} u_5^{(h)} = v_3$$

in Proposition 1. Then it can be observed that the five states of $\boldsymbol{v}$ satisfying the above four equations are (0, 0, 0), (1, 0, 0), (0, 1, 0), (1, 1, 0), and (0, 0, 1) which yield the five states (1, 0, 0, 0, 0), (0, 1, 0, 0, 0), (0, 0, 1, 0, 0), (0, 0, 0, 1, 0), and (0, 0, 0, 0, 1), respectively, for vector $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$.

Proposition 1 leads to the following theorem.

Theorem 1 *Given two m-dimensional nonnegative vectors* $\mathbf{u}^{(h)} = (u_1^{(h)}, u_2^{(h)}, \dots, u_m^{(h)})$, $h = 1, 2$, *and a* $\lceil \log_2 m \rceil$*-dimensional binary vector* $\boldsymbol{v} = (v_1, v_2, \dots, v_{\lceil \log_2 m \rceil}) \in \{0, 1\}^{\lceil \log_2 m \rceil}$ *satisfying* Eqs. (7) *and* (8), *the variable x and function L(f(x)) can be formulated as follows*:

$$x = \sum_{l=1}^{m} \left(\alpha_{l-1} u_l^{(1)} + \alpha_l u_l^{(2)} \right),$$

$$L(f(x)) = \sum_{l=1}^{m} \left(f(\alpha_{l-1}) u_l^{(1)} + f(\alpha_l) u_l^{(2)} \right).$$

Proof Since Proposition 1 indicates that the vector $\mathbf{u}^{(1)} + \mathbf{u}^{(2)}$ is a binary unit vector and each of the its m states can be yielded with a unique state of vector $\boldsymbol{v}$, the valid SOS2 construction for formulating a piecewise linear function is achieved. □

The numbers of variables and constraints required by the four linearization schemes, viz. the conventional formulation, the incremental formulation, the logarithmic method, and the proposed method, are listed in Table 1.

The comparisons are summarized as follows:

1. Among the compared models, the proposed method, as well as the logarithmic formulation, introduces the fewest binary variables, the quantity of which is $\lceil \log_2 m \rceil$.

Table 1 Quantities of variables and constraints required by the four linearization schemes

Items of different types	Conventional formulation	Incremental formulation	Logarithmic method	Proposed method
Variables				
Continuous	$m+1$	m	$m+1$	$2m$
Binary	m	$m-1$	$\lceil \log_2 m \rceil$	$\lceil \log_2 m \rceil$
Constraints				
Equality	4	2	3	$\lceil \log_2 m \rceil + 3$
Inequality	m	$2(m-1)$	$2\lceil \log_2 m \rceil$	0

2. No inequality constraint is necessary in the proposed method, while $2\lceil \log_2 m \rceil$ or more inequality constraints are required by the logarithmic method or the other two formulations.
3. Although the proposed method incurs $m-1$ continuous variables more than the logarithmic method, it simultaneously introduces a system of $\lceil \log_2 m \rceil$ equalities satisfying the canonical form. Since each equation in the canonical form reduces the number of dimensions of the linear programming (LP) solution space by one, the LP relaxation solution space constructed by the proposed method has $m - \lceil \log_2 m \rceil - 1$ dimensions higher than that using the logarithmic method.

It is argued that the difficulty of the MILP solving would be mainly determined by the quantities of binary variables as well as inequality constraints and the number of continuous variables plays a relatively minor role in general [5, 26, 39]. Kettani and Oral [22] indicate that the inequality constraints may deteriorate the solving of integer programs due to being inactive. Our preliminary studies also imply that it is relatively computationally efficient for the general MILP solvers, e.g. CPLEX and Gurobi, to cope with a model with relatively few inequality constraints even if the number of continuous variables is increased. It is thus reasonable to expect that the proposed method has a higher computational efficiency than the reference models. This claim will be validated with the computational experiments in Sect. 4.

4 Numerical experiments

The numerical experiments comprising three sets of test instances were conducted to compare the performances of each reference model and the proposed method. The MILP models formulated by the four compared linearization schemes were solved using the Gurobi MILP solver, and all the experiments were run on a PC equipped with the Intel Core i5-4210 CPU, 8 GB RAM, and Windows 10 64-bit operating system. The CPU time limit for each experiment was set to be 7200 s while all the other default settings in Gurobi were kept.

4.1 Experiment instances

The considered experiment instances were set by referring to the NLP models in the literature.

(a) *Instance 1*

The following NLP problem from Li et al. [28] was employed as Instance 1:

$$\text{Min} \quad x^{0.4} - y^2 \tag{12}$$

$$\text{s.t.} \quad x^{0.8} - 6x + y^2 \leq -7, \tag{13}$$

$$x + y \leq 8, \tag{14}$$

where $1 \leq x \leq 7.4$ and $1 \leq y \leq 7.4$. Recall that m is the number of intervals obtained from dividing the domain of the function to be linearized. The setting that the domain is evenly split was considered. Since both variables x and y share the bounds, the breaking points for either of them were given as

$$\alpha_l = 1 + 6.4\frac{l}{m}, \quad \forall l \in M \cup \{0\}. \tag{15}$$

For each of the distinct nonlinear terms in the objective function (12) and constraints (13)–(14), i.e. $x^{0.4}$, y^2, and $x^{0.8}$, an individual linearized function was constructed for formulating an MILP model.

(b) *Instance 2*

The NLP model [28] shown below was used as Instance 2:

$$\begin{aligned}
\text{Min} \quad & x_1^3 - 1.8x_1^{2.8} + 0.8x_2^{2.2} - x_2^{2.1} + x_3^{0.5} - 3.5x_4^{0.8} - 0.3x_5^{1.1} \\
\text{s.t.} \quad & x_1^{1.2} + x_2^{0.8} \leq 8, \\
& x_1^{1.2} - x_3^{1.7} \leq 2, \\
& x_2^{2.1} - x_4^{1.7} \geq 4.5, \\
& x_4^{0.8} - x_5^{0.96} \geq -3, \\
& x_2^{2.2} - x_5^{1.1} \geq -0.1,
\end{aligned}$$

where $1 \leq x_i \leq 7.4$, $\forall i \in \{1, 2, \ldots, 5\}$. The break points for each variable x_i, $\forall i \in \{1, 2, \ldots, 5\}$ were again given as Eq. (15). Note that 12 linearized functions were constructed in the MILP formulation since there are 12 distinct nonlinear terms in the objective function and constraints.

(c) *Instance set 3*

Instance set 3 was derived from a two-dimensional rectangular packing problem [42]. The considered NLP model comprises a nonconvex objective function and a system of linear constraints as shown below:

$$\text{Min} \quad \ln(x) + \ln(y) \tag{16}$$

$$\text{s.t.} \quad x_i + p_i s_i + q_i(1 - s_i) \le x_j + \bar{x}(1 - \lambda_{ij} + \mu_{ij}), \quad \forall i,j \in \{1, 2, \dots, n\}, \quad i < j, \tag{17}$$

$$x_j + p_j s_j + q_j(1 - s_j) \le x_i + \bar{x}(\lambda_{ij} + \mu_{ij}), \quad \forall i,j \in \{1, 2, \dots, n\},\ i < j, \tag{18}$$

$$y_i + q_i s_i + p_i(1 - s_i) \le y_j + \bar{y}(2 - \lambda_{ij} - \mu_{ij}), \quad \forall i,j \in \{1, 2, \dots, n\},\ i < j, \tag{19}$$

$$y_j + q_j s_j + p_j(1 - s_j) \le y_i + \bar{y}(1 + \lambda_{ij} - \mu_{ij}), \quad \forall i,j \in \{1, 2, \dots, n\},\ i < j, \tag{20}$$

$$x_i + p_i s_i + q_i(1 - s_i) \le x, \quad \forall i \in \{1, 2, \dots, n\}, \tag{21}$$

$$y_i + q_i s_i + p_i(1 - s_i) \le y, \quad \forall i \in \{1, 2, \dots, n\}, \tag{22}$$

$$\underline{x} \le x \le \bar{x}, \tag{23}$$

$$\underline{y} \le y \le \bar{y}, \tag{24}$$

where $x_i, y_i \ge 0, s_i \in \{0, 1\},\ \forall i \in \{1, 2, \dots, n\}$, and $\lambda_{ij}, \mu_{ij} \in \{0, 1\},\ \forall i,j \in \{1, 2, \dots, n\},\ i < j$. Note that n, $\underline{x}$, $\bar{x}$, $\underline{y}$, $\bar{y}$, and p_i, q_i, $\forall i \in \{1, 2, \dots, n\}$ are all the given parameters, whose values are listed in Table 2 for five instances.

To develop the piecewise linearization for the two nonlinear terms $\ln(x)$ and $\ln(y)$ in the objective function (16), we generated two sets of breaking points respectively for variables x and y, the domains of which are respectively given by constraints (23) and (24), as follows:

$$\alpha_l = \underline{x} + (\bar{x} - \underline{x})\frac{l}{m}, \quad \forall l \in M \cup \{0\}, \tag{25}$$

$$\beta_l = \underline{y} + (\bar{y} - \underline{y})\frac{l}{m}, \quad \forall l \in M \cup \{0\}. \tag{26}$$

The MILP model formulated by our proposed method is shown as follows:

Table 2 Given parameters for the five instances in Instance set 3

Instance	n	$(\underline{x}, \bar{x})$	$(\underline{y}, \bar{y})$	(p_i, q_i)
3A	6	(50, 100)	(35, 100)	(50, 35), (22, 13), (31, 17), (15, 10), (11, 9), (20, 6)
3B	7	(45, 100)	(19, 100)	(45, 19), (20, 16), (30, 17), (25, 10), (21, 5), (20, 8), (13, 13)
3C	8	(40, 100)	(12, 100)	(40, 12), (20, 10), (20, 5), (10, 6), (15, 7), (10, 9), (9, 8), (37, 5)
3D	9	(50, 100)	(10, 100)	(10, 3), (15, 7), (15, 10), (20, 8), (20, 6), (50, 7), (50, 11), (20, 10), (30, 5)
3E	10	(60, 100)	(20, 100)	(5, 3), (10, 7), (10, 8), (15, 10), (20, 15), (25, 10), (60, 10), (50, 20), (30, 5), (30, 10)

$$\begin{aligned}
\text{Min} \quad & \sum_{l=1}^{m} \left(\ln(\alpha_{l-1}) u_l^{(1)} + \ln(\alpha_l) u_l^{(2)} \right) + \sum_{l=1}^{m} \left(\ln(\beta_{l-1}) \hat{u}_l^{(1)} + \ln(\beta_l) \hat{u}_l^{(2)} \right) \\
\text{s.t.} \quad & (17)\text{–}(22), \\
& x = \sum_{l=1}^{m} \left(\alpha_{l-1} u_l^{(1)} + \alpha_l u_l^{(2)} \right), \\
& y = \sum_{l=1}^{m} \left(\beta_{l-1} \hat{u}_l^{(1)} + \beta_l \hat{u}_l^{(2)} \right), \\
& (7), (8), \\
& \sum_{l=1}^{m} \sum_{h=1}^{2} \hat{u}_l^{(h)} = 1, \\
& \sum_{l=1}^{m} \left(b_{lk} \sum_{h=1}^{2} \hat{u}_l^{(h)} \right) = \hat{v}_k, \quad \forall k \in G,
\end{aligned}$$

where (25), (26), $u_l^{(h)}, \hat{u}_l^{(h)} \geq 0,\ \forall l \in M,\ \forall h \in \{1, 2\}$, and $v_k, \hat{v}_k \in \{0, 1\}, \forall k \in G$.

4.2 Experiment results

For each of the foregoing test instances, the five cases $m = 50$, 100, 500, 1000, and 2000 were considered in the computational experiments.

4.2.1 Experiment 1

In Experiment 1, the solving of the MILP models formulated by the four compared linearization schemes for Instance 1, which can be regarded as a small-sized numerical instance, was conducted. In each of the five cases, all the four formulated MILP models yielded the same solution and thus objective value, which are shown in Table 10. The computational results, including the number of Simplex iterations (abbreviated as #ITER) and CPU time, of the MILP solving for all the five cases are shown in Table 3. The numbers of continuous variables (i.e. #CVAR), binary variables (i.e. #BVAR), equality constraints (i.e. #ECONS), and inequality constraints (i.e. #ICONS) in the formulated MILP models are also listed. Notice that for each of the five cases the two inequality constraints existing in the MILP model formulated by the proposed method exactly stem from constraints (13) and (14). It is shown that the solving of the MILP model built by the proposed method required the fewest iterations and shortest CPU time for the cases 1–3 (i.e. $m = 500$), 1–4 (i.e. $m = 1000$), and 1–5 (i.e. $m = 2000$) while all the four models needed less than 0.2 s for the cases 1–1 (i.e. $m = 50$) and 1–2 (i.e. $m = 100$). The comparisons of CPU times required by the proposed method and each reference formulation are illustrated in Fig. 1.

Table 3 Computational results of experiment 1

Case (m)	MILP items	Conventional formulation	Incremental formulation	Logarithmic method	Proposed method
	#CVAR	104	102	104	202
	#BVAR	100	98	12	12
1–1	#ECONS	8	4	6	18
(50)	#ICONS	102	198	26	2
	#ITER	469	133	248	215
	CPU time (s)	0.17	0.03	0.19	0.13
	#CVAR	204	202	204	402
	#BVAR	200	198	14	14
1–2	#ECONS	8	4	6	20
(100)	#ICONS	202	398	30	2
	#ITER	6826	260	376	319
	CPU time (s)	0.18	0.04	0.19	0.13
	#CVAR	1004	1002	1004	2002
	#BVAR	1000	998	18	18
1–3	#ECONS	8	4	6	24
(500)	#ICONS	1002	1598	38	2
	#ITER	34,200	1327	982	816
	CPU time (s)	1.75	1.21	0.29	0.27
	#CVAR	2004	2002	2004	4002
	#BVAR	2000	1998	20	20
1–4	#ECONS	8	4	6	26
(1000)	#ICONS	2002	3998	42	2
	#ITER	88,415	2617	1269	1202
	CPU time (s)	12.90	4.23	0.46	0.36
	#CVAR	4004	4002	4004	8002
	#BVAR	4000	3998	22	22
1–5	#ECONS	8	4	6	28
(2000)	#ICONS	4002	7998	46	2
	#ITER	1,444,551	8163	1280	1035
	CPU time (s)	68.21	13.30	0.80	0.68

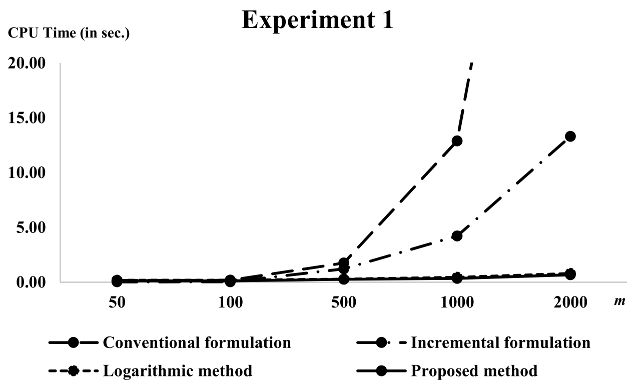

Fig. 1 Trends in the required CPU time in Experiment 1

4.2.2 Experiment 2

The MILP solving for Instance 2, which can be considered as a medium-sized instance, was conducted in Experiment 2. All the four compared methods produced the identical solution for the cases 2–1, 2–2, and 2–3, as shown in Table 11, while the recorded solutions for the cases 2–4 and 2–5 were obtained from the methods other than the conventional formulation. The computational results for all the five cases are shown in Table 4. The proposed method outperformed all the reference methods by demonstrating the fewest iterations and shortest CPU time for each case. More than 25% of Simplex iterations and 16% of CPU time on average were saved by utilizing the proposed scheme instead of the logarithmic method. The CPU-time comparison between the proposed method and each reference formulation can be found in Fig. 2.

4.2.3 Experiment 3

Experiment 3 was designed to perform the MILP solving for Instance set 3, including Instances 3A–3E. As Experiments 1 and 2, the compared methods, if the formulated MILP was solved to optimality within the CPU time limit, generated the identical solution, which is recorded for each case in Tables 12, 13, 14, 15 and 16.

Table 4 Computational results of Experiment 2

Case (m)	MILP items	Conventional formulation	Incremental formulation	Logarithmic method	Proposed method
	#CVAR	260	255	260	505
	#BVAR	250	245	30	30
2–1	#ECONS	20	10	15	45
(50)	#ICONS	255	495	65	5
	#ITER	27,912	2854	2514	2047
	CPU time (s)	0.60	0.23	0.37	0.17
	#CVAR	510	505	510	1005
	#BVAR	500	495	35	35
2–2	#ECONS	20	10	15	50
(100)	#ICONS	505	995	75	5
	#ITER	89,797	2834	2840	2469
	CPU time (s)	1.34	0.33	0.37	0.15
	#CVAR	2510	2505	2510	5005
	#BVAR	2500	2495	45	45
2-3	#ECONS	20	10	15	60
(500)	#ICONS	2505	4995	95	5
	#ITER	10,417,072	14,368	9955	8447
	CPU time (s)	398.66	3.67	2.09	1.76
	#CVAR	5010	5005	5010	10,005
	#BVAR	5000	4995	50	50
2–4	#ECONS	20	10	15	65
(1000)	#ICONS	5005	9995	105	5
	#ITER	–	25,221	15,729	12,336
	CPU time (s)	–	17.32	3.19	2.40
	#CVAR	10,010	10,005	10,010	20,005
	#BVAR	10,000	9995	55	55
2–5	#ECONS	20	10	15	70
(2000)	#ICONS	10,005	19,995	115	5
	#ITER	–	30,810	25,709	17,214
	CPU time (s)	–	44.27	5.2	4.91

"–" not applicable due to exceeding the computational time threshold (7200 s)

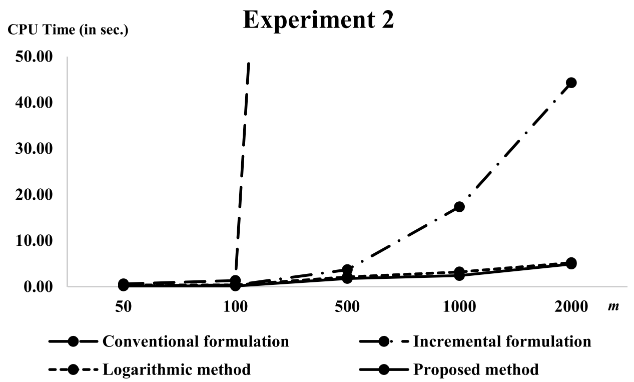

Fig. 2 Trends in the required CPU time in Experiment 2

(a) *Experiment 3A*

The computational results of Instance 3A for the five cases are provided in Table 5. The proposed scheme was again the most advantageous in computation among the compared methods for all the cases. More than 26% of Simplex iterations and 30% of CPU time on average were reduced by employing the proposed method instead of the logarithmic formulation. The CPU-time comparison between the proposed method and each reference formulation is illustrated in Fig. 3.

Table 5 Computational results of Experiment 3A

Case (m)	MILP items	Conventional formulation	Incremental formulation	Logarithmic method	Proposed method
	#CVAR	116	114	116	214
	#BVAR	136	134	48	48
3A-1	#ECONS	8	4	6	18
(50)	#ICONS	172	268	96	72
	#ITER	71,739	63,620	30,275	29,397
	CPU time (s)	1.89	1.17	0.48	0.41
	#CVAR	216	214	216	414
	#BVAR	236	234	50	50
3A-2	#ECONS	8	4	6	20
(100)	#ICONS	272	468	100	72
	#ITER	205,038	48,781	66,227	40,918
	CPU time (s)	1.97	0.53	0.58	0.42
	#CVAR	1016	1014	1016	2014
	#BVAR	1036	1034	54	54
3A-3	#ECONS	8	4	6	24
(500)	#ICONS	1072	2068	108	72
	#ITER	2,152,703	105,804	62,817	57,573
	CPU time (s)	46.59	7.29	1.94	1.07
	#CVAR	2016	2014	2016	4014
	#BVAR	2036	2034	56	56
3A-4	#ECONS	8	4	6	26
(1000)	#ICONS	2072	4068	112	72
	#ITER	9,411,212	149,225	60,890	46,442
	CPU time (s)	477.64	12.46	2.50	1.90
	#CVAR	4016	4014	4016	8014
	#BVAR	4036	4034	58	58
3A-5	#ECONS	8	4	6	28
(2000)	#ICONS	4072	8068	116	72
	#ITER	–	428,472	79,746	46,234
	CPU time (s)	–	187.33	5.84	4.03

"–" not applicable due to exceeding the computational time threshold (7200 s)

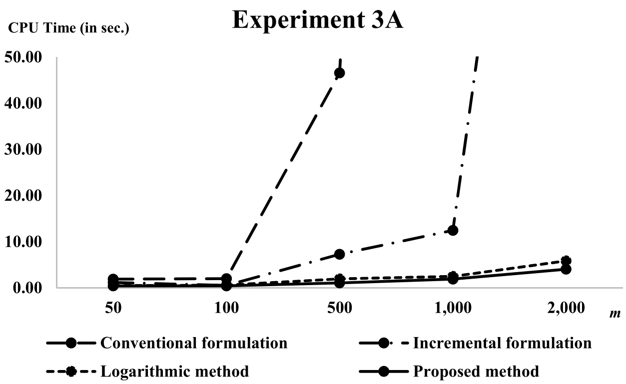

Fig. 3 Trends in the required CPU time in Experiment 3A

(b) *Experiment 3B*

Table 6 demonstrates the computational results of Instance 3B. The proposed scheme again outperformed the reference methods for all the cases. The trends in the required CPU time in Experiment 3B are shown in Fig. 4. It is illustrated that the computational superiority of the proposed scheme becomes obvious as the value of m increases.

Table 6 Computational results of Experiment 3B

Case (m)	MILP items	Conventional formulation	Incremental formulation	Logarithmic method	Proposed method
	#CVAR	118	116	118	216
	#BVAR	149	147	61	61
3B-1	#ECONS	8	4	6	18
(50)	#ICONS	198	294	122	98
	#ITER	674,585	219,885	238,600	165,924
	CPU time (s)	11.44	3.04	3.29	3.02
	#CVAR	218	216	218	416
	#BVAR	249	247	63	63
3B-2	#ECONS	8	4	6	20
(100)	#ICONS	298	494	126	98
	#ITER	1,111,838	241,212	387,810	342,042
	CPU time (s)	252.93	7.95	4.99	4.17
	#CVAR	1018	1016	1018	2016
	#BVAR	1049	1047	67	67
3B-3	#ECONS	8	4	6	24
(500)	#ICONS	1098	2094	134	98
	#ITER	–	10,660,858	383,696	308,336
	CPU time (s)	–	814.10	11.67	9.61
	#CVAR	2018	2016	2018	4016
	#BVAR	2049	2047	69	69
3B-4	#ECONS	8	4	6	26
(1000)	#ICONS	2098	4094	138	98
	#ITER	–	1,098,779	541,961	462,368
	CPU time (s)	–	321.30	40.33	37.67
	#CVAR	4018	4016	4018	8016
	#BVAR	4049	4047	71	71
3B-5	#ECONS	8	4	6	28
(2000)	#ICONS	4098	8094	142	98
	#ITER	–	4,508,429	526,690	479,777
	CPU time (s)	–	2831.18	49.34	37.92

"–" not applicable due to exceeding the computational time threshold (7200 s)

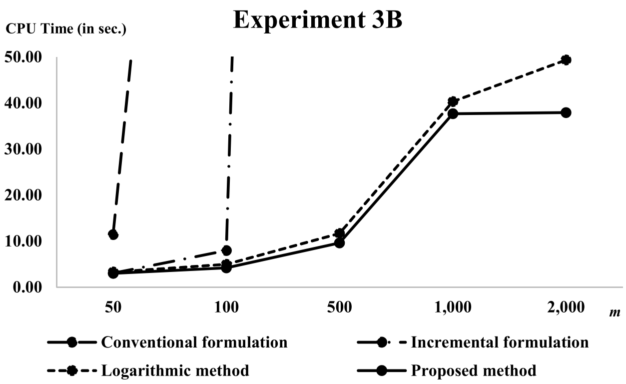

Fig. 4 Trends in the required CPU time in Experiment 3B

(c) *Experiment 3C*

Table 7 shows the numerical results of Instance 3C. The proposed scheme was still the most efficient in terms of both Simplex iterations and computational time among all the four formulations. The comparisons of the required CPU times in Experiment 3C are shown in Fig. 5.

Table 7 Computational results of Experiment 3C

Case (m)	MILP items	Conventional formulation	Incremental formulation	Logarithmic method	Proposed method
	#CVAR	120	118	120	218
	#BVAR	164	162	76	76
3C-1	#ECONS	8	4	6	18
(50)	#ICONS	228	324	152	128
	#ITER	1,396,676	434,945	404,062	340,738
	CPU time (s)	22.86	8.05	5.58	4.62
	#CVAR	220	218	220	418
	#BVAR	264	262	78	78
3C-2	#ECONS	8	4	6	20
(100)	#ICONS	328	524	156	128
	#ITER	2,727,282	676,868	601,268	582,025
	CPU time (s)	61.80	6.94	5.94	4.45
	#CVAR	1020	1018	1020	2018
	#BVAR	1064	1062	82	82
3C-3	#ECONS	8	4	6	24
(500)	#ICONS	1128	2124	164	128
	#ITER	–	3,562,135	757,785	754,611
	CPU time (s)	–	125.17	13.53	11.24
	#CVAR	2020	2018	2020	4018
	#BVAR	2064	2062	84	84
3C-4	#ECONS	8	4	6	26
(1000)	#ICONS	2128	4124	168	128
	#ITER	–	28,546,133	1,336,540	1,259,797
	CPU time (s)	–	1857.36	40.33	33.53
	#CVAR	4020	4018	4020	8018
	#BVAR	4064	4062	86	86
3C-5	#ECONS	8	4	6	28
(2000)	#ICONS	4128	8124	172	128
	#ITER	–	–	1,252,284	1,067,126
	CPU time (s)	–	–	84.15	74.32

"–" not applicable due to exceeding the computational time threshold (7200 s)

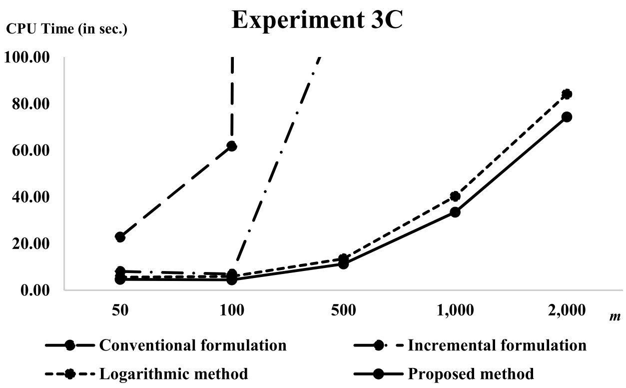

Fig. 5 Trends in the required CPU time in Experiment 3C

(d) *Experiment 3D*

The numerical results of Instance 3D are shown in Table 8, where once again the proposed method demonstrates the competitiveness. The CPU-time comparisons in Experiment 3D can be found in Fig. 6.

Table 8 Computational results of Experiment 3D

Case (m)	MILP items	Conventional formulation	Incremental formulation	Logarithmic method	Proposed method
	#CVAR	122	120	122	220
	#BVAR	181	179	93	93
3D-1	#ECONS	8	4	6	18
(50)	#ICONS	262	358	186	162
	#ITER	13,848,625	2,340,763	3,705,627	1,925,302
	CPU time (s)	365.53	25.28	18.96	17.34
	#CVAR	222	220	222	420
	#BVAR	281	279	95	95
3D-2	#ECONS	8	4	6	20
(100)	#ICONS	362	558	190	162
	#ITER	–	2,269,431	1,960,072	1,709,865
	CPU time (s)	–	38.93	42.08	25.54
	#CVAR	1022	1020	1022	2020
	#BVAR	1081	1079	99	99
3D-3	#ECONS	8	4	6	24
(500)	#ICONS	1162	2158	198	162
	#ITER	–	21,590,658	1,889,079	1,502,082
	CPU time (s)	–	791.51	42.62	31.36
	#CVAR	2022	2020	2022	4020
	#BVAR	2081	2079	101	101
3D-4	#ECONS	8	4	6	26
(1000)	#ICONS	2162	4158	202	162
	#ITER	–	7,928,953	4,869,733	3,014,976
	CPU time (s)	–	955.60	167.29	126.80
	#CVAR	4022	4020	4022	8020
	#BVAR	4081	4079	103	103
3D-5	#ECONS	8	4	6	28
(2000)	#ICONS	4162	8158	206	162
	#ITER	–	–	5,290,824	5,043,709
	CPU time (s)	–	–	462.17	432.52

"–" not applicable due to exceeding the computational time threshold (7200 s)

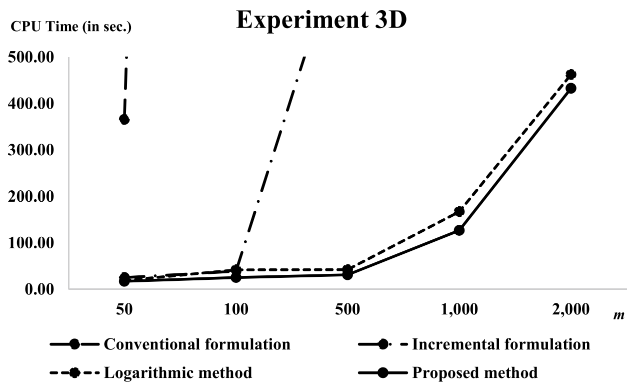

Fig. 6 Trends in the required CPU time in Experiment 3D

(e) *Experiment 3E*

The computational results of Instance 3E, which is the large-sized experiment instance, are reported in Table 9, and the CPU-time comparisons in Experiment 3E are illustrated in Fig. 7. In Experiment 3E, the growing computational dominance of the proposed scheme is evident in the widening performance gap between the proposed scheme and each reference formulation. More than 58% of Simplex iterations and 30% of CPU time on average were saved by adopting the proposed scheme instead of the logarithmic method.

Table 9 Computational results of Experiment 3E

Case (m)	MILP items	Conventional formulation	Incremental formulation	Logarithmic method	Proposed method
	#CVAR	124	122	124	222
	#BVAR	200	198	112	112
3E-1	#ECONS	8	4	6	18
(50)	#ICONS	300	396	224	200
	#ITER	41,813,539	2,186,162	4,992,450	4,071,559
	CPU time (s)	1352.77	76.09	72.73	62.74
	#CVAR	224	222	224	422
	#BVAR	300	298	114	114
3E-2	#ECONS	8	4	6	20
(100)	#ICONS	400	596	228	200
	#ITER	–	33,076,374	5,923,315	4,685,074
	CPU time (s)	–	222.21	76.42	65.94
	#CVAR	1024	1022	1024	2022
	#BVAR	1100	1098	118	118
3E-3	#ECONS	8	4	6	24
(500)	#ICONS	1200	2196	236	200
	#ITER	–	95,092,253	71,210,754	12,289,396
	CPU time (s)	–	1290.43	608.79	190.45
	#CVAR	2024	2022	2024	4022
	#BVAR	2100	2098	120	120
3E-4	#ECONS	8	4	6	26
(1000)	#ICONS	2200	4196	240	200
	#ITER	–	167,291,533	23,942,846	15,176,977
	CPU time (s)	–	1438.23	630.62	487.78
	#CVAR	4024	4022	4024	8022
	#BVAR	4100	4098	122	122
3E-5	#ECONS	8	4	6	28
(2000)	#ICONS	4200	8196	244	200
	#ITER	–	–	26,676,412	19,416,014
	CPU time (s)	–	–	2272.35	1732.12

"–" not applicable due to exceeding the computational time threshold (7200 s)

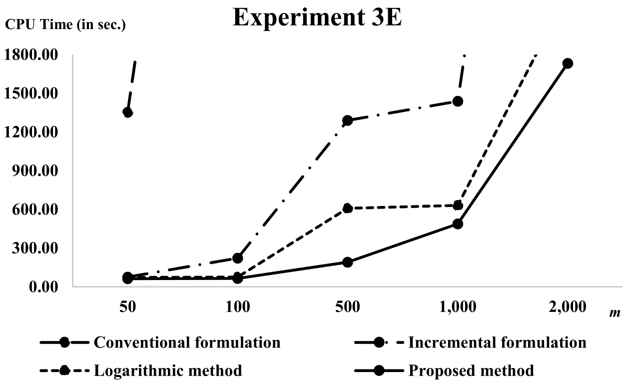

Fig. 7 Trends in the required CPU time in Experiment 3E

5 Concluding remarks

This paper considers the nonconvex optimization problems in which all the nonlinear terms are univariate. An effective and efficient piecewise linear approximation scheme for these problems has been developed. The performance of the linearization technique primarily depends on the numbers of variables and inequality constraints required in constructing the SOS2 formulation for the piecewise linear function. While the state-of-the-art logarithmic method requires $2\lceil \log_2 m \rceil$ inequality constraints, where m is the number of line segments in the constructed piecewise linear function, none is incurred by our proposed scheme. The price that more continuous variables are introduced in the proposed scheme than in the state-of-the-art method is less than offset by the simultaneous inclusion of a system of equality constraints satisfying the canonical form and the absence of any inequality constraint. The conducted computational experiments have demonstrated that the presented scheme retains the computational superiority, the degree of which increases with m.

Further study could be conducted by investigating the possibility of reducing the number of binary or continuous variables for the proposed linearization scheme. It is also worth developing another alternative method which requires fewer binary variables or inequality constraints without incurring more continuous variables than the logarithmic method. Another direction for future research is to generalize our proposed method to the piecewise linearization formulation for the multivariate function.

Appendix

See Tables 10, 11, 12, 13, 14, 15 and 16.

Table 10 Solutions yielded in Experiment 1

Case (m)	Solution (x, y)	Objective value
1–1 (50)	(4.153624, 3.846376)	− 13.030076
1–2 (100)	(4.153479, 3.846521)	− 13.029238
1–3 (500)	(4.153404, 3.846596)	− 13.028829
1–4 (1000)	(4.153402, 3.846598)	− 13.028815
1–5 (2000)	(4.153401, 3.846598)	− 13.028813

Table 11 Solutions yielded in Experiment 2

Case (m)	Solution $(x_1, x_2, \ldots, x_5)$	Objective value
2–1 (50)	(3.671195, 4.343845, 1.816988, 5.359112, 7.4)	− 35.565041
2–2 (100)	(3.671174, 4.343953, 1.817564, 5.359103, 7.4)	− 35.562564
2–3 (500)	(3.671159, 4.344030, 1.817677, 5.359097, 7.4)	− 35.560999
2–4 (1000)	(3.671159, 4.344031, 1.817679, 5.359096, 7.4)	− 35.560954
2–5 (2000)	(3.671159, 4.344032, 1.817679, 5.359096, 7.4)	− 35.560937

Table 12 Solutions yielded in Experiment 3-A

Case (m)	Solution (x, y)	Objective value
3A-1 (50)	(62, 50)	8.039073
3A-2 (100)	(62, 50)	8.039143
3A-3 (500)	(62, 50)	8.039157
3A-4 (1000)	(62, 50)	8.039157
3A-5 (2000)	(62, 50)	8.039157

Table 13 Solutions yielded in Experiment 3-B

Case (m)	Solution (x, y)	Objective value
3B-1 (50)	(62, 40)	7.815945
3B-2 (100)	(62, 40)	7.815996
3B-3 (500)	(62, 40)	7.816012
3B-4 (1000)	(62, 40)	7.816013
3B-5 (2000)	(62, 40)	7.816014

Table 14 Solutions yielded in Experiment 3-C

Case (m)	Solution (x,y)	Objective value
3C-1 (50)	(60, 22)	7.184634
3C-2 (100)	(60, 22)	7.185192
3C-3 (500)	(60, 22)	7.185382
3C-4 (1000)	(60, 22)	7.185385
3C-5 (2000)	(60, 22)	7.185387

Table 15 Solutions yielded in Experiment 3-D

Case (m)	Solution (x, y)	Objective value
3D-1 (50)	(56, 33)	7.521597
3D-2 (100)	(56, 33)	7.521767
3D-3 (500)	(56, 33)	7.521857
3D-4 (1000)	(56, 33)	7.521858
3D-5 (2000)	(56, 33)	7.521859

Table 16 Solutions yielded in Experiment 3-E

Case (m)	Solution (x, y)	Objective value
3E-1 (50)	(98, 30)	7.985894
3E-2 (100)	(98, 30)	7.986076
3E-3 (500)	(98, 30)	7.986161
3E-4 (1000)	(98, 30)	7.986165
3E-5 (2000)	(98, 30)	7.98616

Acknowledgements The authors would like to thank sincerely the Editor and anonymous reviewers for their thoughtful and valuable comments which have significantly improved the quality of this paper. F. J. Hwang was supported by the 2017 University of Technology Sydney Professional Experience Program grant. Y.-H. Huang was partially supported by the Ministry of Science and Technology of Taiwan under the Grant MOST 109-2410-H-030-037-MY3.

References

1. Aghezzaf, E.H., Wolsey, L.A.: Modelling piecewise linear concave costs in a tree partitioning problem. Discrete Appl. Math. **50**(2), 101–109 (1994)
2. Balakrishnan, A., Graves, S.: A composite algorithm for a concave-cost network flow problem. Networks **19**(2), 175–202 (1989)
3. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming Theory and Algorithms, 2nd edn. Wiley, New York (1993)
4. Beale, E.M.L., Tomlin, J.A.: Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. In: Lawrence, J. (ed.) Proceedings of the Fifth International Conference on Operational Research, pp. 447–454. Tavistock Publications, London (1970)

5. Bertsimas, D., Tsitsiklis, J.N.: Introduction to Linear Optimization. Athena Scientific Belmont, Massachusetts (1997)
6. Bienstock, D., Günlük, O.: Capacitated network design polyhedral structure and computation. INFORMS J. Comput. **8**(3), 243–259 (1996)
7. Chan, L.M.A., Muriel, A., Shen, Z.J., Simchi-Levi, D.: On the effectiveness of zero-inventory-ordering policies for the economic lot-sizing model with a class of piecewise linear cost structures. Oper. Res. **50**(6), 1058–1067 (2002)
8. Chan, L.M.A., Muriel, A., Shen, Z.J., Simchi-Levi, D., Teo, C.P.: Effective zero-inventory-ordering policies for the single-warehouse multiretailer problem with piecewise linear cost structures. Manag. Sci. **48**(11), 1446–1460 (2002)
9. Croxton, K.L.: Modeling and Solving Network Flow Problems with Piecewise Linear Costs, with Applications in Supply Chain Management, Ph.D. thesis. Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts (1999)
10. Croxton, K.L., Gendron, B., Magnanti, T.L.: Models and methods for merge-in-transit operations. Transp. Sci. **37**(1), 1–22 (2003)
11. Croxton, K.L., Gendron, B., Magnanti, T.L.: Variable disaggregation in network flow problems with piecewise linear costs. Oper. Res. **55**(1), 146–157 (2007)
12. Dantzig, G.B.: On the significance of solving linear-programming problems with some integer variables. Econometrica **28**(1), 30–44 (1960)
13. Gabrel, V., Knippel, A., Minoux, M.: Exact solution of multicommodity network optimization problems with general step cost functions. Oper. Res. Lett. **25**(1), 15–23 (1999)
14. Graf, T., Van Hentenryck, P., Pradelles-Lasserre, C., Zimmer, L.: Simulation of hybrid circuits in constraint logic programming. Comput. Math. Appl. **20**(9–10), 45–56 (1990)
15. Günlük, O.: A branch-and-cut algorithm for capacitated network design problems. Math. Program. **86**(1), 17–39 (1999)
16. Hillier, F.S., Lieberman, G.J.: Introduction to Operations Research, 6th edn. McGraw-Hill, New York (1995)
17. Holmberg, K.: Solving the staircase cost facility location problem with decomposition and piecewise linearization. Eur. J. Oper. Res. **75**(1), 41–61 (1994)
18. Holmberg, K., Ling, J.: A Lagrangean heuristic for the facility location problem with staircase costs. Eur. J. Oper. Res. **97**(1), 63–74 (1997)
19. Huang, Y.H., Li, H.L.: A note on logarithmic method for non-separable function. NCTU Research Report OPTL-D-10-00325: 1–17 (2010)
20. Huang, Y.H., Hwang, F.J.: Global optimization for the three-dimensional open-dimension rectangular packing problem. Eng. Optim. **50**(10), 1789–1809 (2018)
21. Jeroslow, R.G., Lowe, J.K.: Modelling with integer variables. In: Mathematical Programming at Oberwolfach II: Mathematical Programming Studies, vol. 22. Springer, Berlin (1984)
22. Kettani, O., Oral, M.: Equivalent formulations of nonlinear integer problems for efficient optimization. Manag. Sci. **36**(1), 115–119 (1990)
23. Li, H.L.: An efficient method for solving linear goal programming problems. J. Optim. Theory Appl. **90**(2), 465–469 (1996)
24. Li, H.L., Chang, C.T.: An approximately global optimization method for assortment problems. Eur. J. Oper. Res. **105**(3), 604–612 (1998)
25. Li, H.L., Chang, C.T., Tsai, J.F.: Approximately global optimization for assortment problems using piecewise linearization techniques. Eur. J. Oper. Res. **140**(3), 584–589 (2002)
26. Li, H.L., Fang, S.C., Huang, Y.H., Nie, T.: An enhanced logarithmic method for signomial programming with discrete variables. Eur. J. Oper. Res. **255**(3), 922–934 (2016)
27. Li, H.L., Huang, Y.H., Fang, S.C.: A logarithmic method for reducing binary variables and inequality constraints in solving task assignment problems. INFORMS J. Comput. **25**(4), 643–653 (2013)
28. Li, H.L., Lu, H.C., Huang, C.H., Hu, N.Z.: A superior representation method for piecewise linear functions. INFORMS J. Comput. **21**(2), 314–321 (2009)
29. Li, H.L., Yu, C.S.: Global optimization method for nonconvex separable programming problems. Eur. J. Oper. Res. **117**(2), 275–292 (1999)
30. Lin, M.H., Tsai, J.F.: A deterministic global approach for mixed-discrete structural optimization. Eng. Optim. **46**(7), 863–879 (2014)
31. Lin, M.H., Tsai, J.F., Wang, P.C.: Solving engineering optimization problems by a deterministic global optimization approach. Appl. Math. Inf. Sci. **6**(3), 1101–1107 (2012)

32. Lundell, A., Westerlund, J., Westerlund, T.: Some transformation techniques with applications in global optimization. J. Glob. Optim. **43**(2), 391–405 (2009)
33. Magnanti, T.L., Mirchandani, P., Vachani, R.: Modeling and solving the two-facility capacitated network loading problem. Oper. Res. **43**(1), 142–157 (1995)
34. Markowitz, H.M., Manne, A.S.: On the solution of discrete programming problems. Econometrica **25**, 84–110 (1957)
35. Padberg, M.: Approximating separable nonlinear functions via mixed zero-one programs. Oper. Res. Lett. **27**(1), 1–5 (2000)
36. Sherali, H.D.: On mixed-integer zero-one representations for separable lower-semicontinuous piecewise-linear functions. Oper. Res. Lett. **28**(4), 155–160 (2001)
37. Sridhar, S., Linderoth, J., Luedtke, J.: Locally ideal formulations for piecewise linear functions with indicator variables. Oper. Res. Lett. **41**(6), 627–632 (2013)
38. Rebennack, S.: Computing tight bounds via piecewise linear functions through the example of circle cutting problems. Math. Methods Oper. Res. **84**(1), 3–57 (2016)
39. Till, J., Engell, S., Panek, S., Stursberg, O.: Applied hybrid system optimization: an empirical investigation of complexity. Control Eng. Pract. **12**(10), 1291–1303 (2004)
40. Tsai, J.F.: An optimization approach for supply chain management models with quantity discount policy. Eur. J. Oper. Res. **177**(2), 982–994 (2007)
41. Tsai, J.F., Li, H.L.: A global optimization method for packing problems. Eng. Optim. **38**(6), 687–700 (2006)
42. Tsai, J.F., Wang, P.C., Lin, M.H.: An efficient deterministic optimization approach for rectangular packing problems. Optimization **62**(7), 989–1002 (2013)
43. Vielma, J.P., Ahmed, S., Nemhauser, G.: A note on “a superior representation method for piecewise linear functions’’. INFORMS J. Comput. **22**(3), 493–497 (2010)
44. Vielma, J.P., Nemhauser, G.L.: Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. Math. Program. **128**(1), 49–72 (2011)