*IMA Journal of Management Mathematics* Page 1 of 26
doi:10.1093/imaman/dpq020

# Non-linear programs with max-linear constraints: a heuristic approach

A. AMINU*
*Department of Mathematical Sciences, Kano University of Science and Technology, Wudil, P.M.B 3244, Kano, Nigeria*
*Corresponding author: abdulaamin77@yahoo.com

AND

P. BUTKOVIČ
*School of Mathematics, University of Birmingham, Edgbaston, Birmingham B15 2TT, UK*

[Received on 21 June 2009; accepted on 3 December 2010]

In multiprocessor interactive systems, a number of products are prepared using simultaneously working processors, each contributing to the completion of every product. The completion times of products can be expressed in terms of the starting times of individual processors as max-linear functions. It is not difficult to find the starting times once the completion times are given. However, if two such systems work in parallel, it may be necessary to synchronize their work, i.e. to find starting times so that the products are completed at the same time. This task leads to the problem of solving two-sided max-linear systems, a problem of undecided computational complexity. If the solution set is non-trivial, then the task of finding solutions optimal with respect to a certain criterion arises. In this paper, we propose and examine a number of heuristics for solving non-linear programs with two-sided max-linear constraints and compare their performance.

*Keywords*: max-linear systems; max-linear programming; neighbourhood local search.

## 1. Introduction

Consider the following 'multiprocessor interactive system': products $P_1, \ldots, P_m$ are prepared using $n$ processors, every processor contributing to the completion of each product by producing a partial product. It is assumed that every processor can work on all products simultaneously and that all these actions on a processor start as soon as the processor starts to work. Let $a_{ij}$ be the duration of the work of the $j$th processor needed to complete the partial product for $P_i$ $(i = 1, \ldots, m;\ j = 1, \ldots, n)$. Let us denote by $x_j$ the starting time of the $j$th processor $(j = 1, \ldots, n)$. Then all partial products for $P_i$ $(i = 1, \ldots, m)$ will be ready at time $\max(x_1 + a_{i1}, \ldots, x_n + a_{in})$. Now suppose that independently, $k$ other processors prepare partial products for products $Q_1, \ldots, Q_m$ and the duration and starting times are $b_{ij}$ and $y_j$, respectively. Then the 'synchronization problem' is to find starting times of all $n + k$ processors so that each pair $(P_i, Q_i)$ $(i = 1, \ldots, m)$ is completed at the same time. This task is equivalent to solving the system of equations

$$\max(x_1 + a_{i1}, \ldots, x_n + a_{in}) = \max(y_1 + b_{i1}, \ldots, y_k + b_{ik}) \quad (i = 1, \ldots, m).$$

It may also be required that $P_i$ is not completed before a particular time $c_i$ and similarly $Q_i$ not completed before time $d_i$. Then the equations are

$$\max(x_1 + a_{i1}, \ldots, x_n + a_{in}, c_i) = \max(y_1 + b_{i1}, \ldots, y_k + b_{ik}, d_i) \quad (i = 1, \ldots, m). \tag{1}$$

If we denote $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$ for $a, b \in \mathbb{R}$, then this system gets the form

$$\sum_{j=1,\ldots,n}{}^{\oplus} a_{ij} \otimes x_j \oplus c_i = \sum_{j=1,\ldots,k}{}^{\oplus} b_{ij} \otimes y_j \ \oplus d_i \quad (i = 1, \ldots, m). \tag{2}$$

Therefore, (1) (and also (2)) is called a 'two-sided system of max-linear equations' (or briefly a 'two-sided max-linear system' or just 'max-linear system'). If the solution set to a two-sided max-linear system is non-trivial, then the task of finding solutions optimal with respect to a certain criterion arises. This motivates us to introduce the following notation.

Let $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$ for $a, b \in \mathbb{R}$, and extend the pair of operations to matrices and vectors in the same way as in linear algebra. That is, if $A = (a_{ij})$, $B = (b_{ij})$ and $C = (c_{ij})$ are matrices of compatible sizes with entries from $\mathbb{R}$, we write $C = A \oplus B$ if $c_{ij} = a_{ij} \oplus b_{ij}$ for all $i$, $j$ and $C = A \otimes B$ if $c_{ij} = \sum_k{}^{\oplus} a_{ik} \otimes b_{kj} = \max_k(a_{ik} + b_{kj})$ for all $i$, $j$. Also, if $\alpha \in \mathbb{R}$, then $\alpha \otimes A = (\alpha \otimes a_{ij})$. Max-algebra has been studied by many authors and the reader is referred to Cuninghame-Green (1979, 1995), Heidergott *et al.* (2005), Baccelli *et al.* (1992), Butkovič (2003), Hogben *et al.* (2006).

The aim of this paper is for finding an $x \in \mathbb{R}$ that minimizes the non-linear function $f(x)$ subject to

$$A \otimes x \oplus c = B \otimes x \oplus d, \tag{3}$$

where $A = (a_{ij})$, $B = (b_{ij}) \in \mathbb{R}^{m \times n}$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}$, $d = (d_1, \ldots, d_m)^{\mathrm{T}}$ are given matrices and vectors. Optimization problems of this type will be called 'non-linear programs with two-sided max-linear constraints' or just 'non-linear programs with max-linear constraints (NLPMs)'.

Systems of max-linear equations were investigated already in the first publications dealing with the algebraic structure called max-algebra (sometimes also extremal algebra, path algebra or tropical algebra). In these publications, systems of equations with all variables on one side were considered (Cuninghame-Green, 1979; Butkovič, 2003; Vorobyov, 1967; Zimmermann, 1976;). Other systems with a special structure were studied in the context of solving eigenvalue problems in the corresponding algebraic structures or synchronization in discrete-event systems (Baccelli *et al.*, 1992). Using the $(\oplus, \otimes)$-notation, the studied systems had one of the following forms: $A \otimes x = b$, $A \otimes x = x$ or $A \otimes x = x \oplus b$, where $A$ is a given matrix and $b$ is a given vector. Infinite-dimensional generalizations can be found, e.g. in Akian *et al.* (2005).

General two-sided max-linear systems have also been studied (Butkovič & Hegedüs, 1984; Cuninghame-Green & Butkovič, 2003; Heidergott *et al.*, 2005; Walkup & Boriello, 1988). A general solution method was presented in Walkup & Boriello (1988), however, no complexity bound was given. In Cuninghame-Green & Butkovič (2003), a pseudopolynomial algorithm, called the alternating method, has been developed. In Butkovič & Hegedüs (1984), it was shown that the solution set is generated by a finite number of vectors and an elimination method was suggested. A general iterative approach suggested in Cuninghame-Green & Zimmermann (2001) assumes that finite upper and lower bounds for all variables are given.

Solution methods for max-linear programs with two-sided constraints (MLPs) for both minimization and maximization problems have been presented in Butkovič & Aminu (2009) also see Aminu (2009). It was proved that the methods are pseudopolynomial if all entries are integers. The methods are based on the alternating method Cuninghame-Green & Butkovič (1991). Note that MLP is also NLPM with the objective function max-linear. NLPM has a number of applications for more information, see Butkovič & Aminu (2009) and a comprehensive monograph (Butkovič, 2010) with many other applications and various different approaches.

## 2. The one-sided case

One-sided systems of max-linear equations have been studied for many years and they are very well understood, see Cuninghame-Green (1979), Zimmermann (1976) and Butkovič (2003). These problems have the following form:

$$A \otimes x = b, \tag{4}$$

where $A = (a_{ij}) \in \mathbb{R}^{m\times n}$ and $b = (b_1, \ldots, b_m)^{\mathrm{T}} \in \mathbb{R}^m$.

A method for minimizing max-linear function $f(x) = f^{\mathrm{T}} \otimes x = \max(f_1 + x_1, \ldots, f_n + x_n)$ subject to (4) is presented in Butkovič & Aminu (2009). Note that the problem of minimizing the non-linear function $f(x) = 2^{x_1} + 2^{x_2} + \cdots + 2^{x_n}$ is NP-complete Butkovič & Aminu (2009). Since a one-sided system is a special case of a two-sided system (3), where $a_{ij} > b_{ij}$ and $c_i < d_i$ for every $i$ and $j$, NLPM is also *NP*-complete.

## 3. Definitions and problem formulation

Since NLPM is NP-complete for a general non-linear function, an exact solution method is unlikely to be efficient. Therefore, we will develop heuristic methods for solving this problem. That is we develop heuristic methods for minimizing non-linear function $f(x)$ subject to two-sided max-linear system

$$A \otimes x \oplus c = B \otimes x \oplus d, \tag{5}$$

where $A = (a_{ij})$, $B = (b_{ij}) \in \mathbb{R}^{m\times n}$ and $c = (c_1, \ldots, c_m)^{\mathrm{T}}$, $d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$ are given matrices and vectors. A practical motivation for this research is that no polynomial method is known for solving MLPs. We denote the set of solutions for (5) by $S$. Note that we may consider any matrix $A \in \mathbb{R}^{m\times n}$ to be made up of $n$ column vectors with $m$ entries in each vector, i.e.

$$A = (A_1, A_2, \ldots, A_n), \quad A_j = (a_{1j}, a_{2j}, \ldots, a_{mj})^{\top}.$$

A function $f(x)$: $\mathbb{R}^n \to \mathbb{R}$ is said to be 'isotone' if for every $x, y \in \mathbb{R}^n$, $x \leqslant y$, we have $f(x) \leqslant f(y)$. In what follows, we assume that the objective function $f(x)$ is non-linear, isotone and computable in polynomial time.

DEFINITION 3.1 Let $x = (x_1, \ldots, x_n) \in S$ and $r$ be a positive integer, $r \leqslant n$. The $r$-neighbourhood $N_r(x)$ of $x$ is the set of feasible solutions obtainable by changing at most $r$ components of $x$ (and fixing the remaining variables).

DEFINITION 3.2 A solution $x \in S$ is called a 'local optimum', if there is no $z \in N_r(x)$ such that $f(z) < f(x)$.

Note that $r$ will be omitted if the value of $r$ is known and thus instead of $r$-neighbourhood, we will just say neighbourhood.

The aim here is to develop local search methods, based on $r$-optimum neighbourhood for $r = 1$ and $r = 2$, to minimize non-linear isotone function $f(x)$ subject to

$$A \otimes x \oplus c = B \otimes x \oplus d,$$

where $A = (a_{ij})$, $B = (b_{ij}) \in \mathbb{R}^{m\times n}$ and $c = (c_1, \ldots, c_m)^{\mathrm{T}}$, $d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$ are given matrices and vectors.

## 4. Max-linear programs: some special cases

MLPs (minimization) are the following:

$$\begin{aligned} & f(x) = f^{\mathrm{T}} \otimes x \longrightarrow \min \\ & \text{subject to} \\ & A \otimes x \oplus c = B \otimes x \oplus d, \end{aligned} \tag{6}$$

where $f = (f_1, \ldots, f_n)^{\mathrm{T}} \in \mathbb{R}^n$, $c = (c_1, \ldots, c_n)^{\mathrm{T}}, d = (d_1, \ldots, d_n)^{\mathrm{T}} \in \mathbb{R}^m$, $A = (a_{ij})$ and $B = (b_{ij}) \in \mathbb{R}^{m \times n}$. This problem is denoted by $\mathrm{MLP}^{\min}$. Solution methods for (6) are developed in Butkovič & Aminu (2009). We denote the optimal solution set for $\mathrm{MLP}^{\min}$ by $S^{\min}$.

### 4.1 *Basic properties*

Before we develop methods for solving max-linear programs for special cases, we summarize some basic properties. To do so we will denote $\inf_{x \in S} f(x)$ by $f^{\min}$ and $M^{>} = \{i \in M; c_i > d_i\}$. Also, if $\alpha \in \mathbb{R}$, then the symbol $\alpha^{-1}$ stands for $-\alpha$.

THEOREM 4.1 (Butkovič & Aminu, 2009) *$f^{\min} = -\infty$ if and only if $c = d$.*

LEMMA 4.1 (Butkovič & Aminu, 2009) Let $c \geqslant d$. If $x \in S$ and $(A \otimes x)_i \geqslant c_i$ for all $i \in M$, then $x' = \alpha \otimes x \in S$ and $(A \otimes x)_i = c_i$ for some $i \in M$, where

$$\alpha = \max_{i \in M} (c_i \otimes (A \otimes x)_i^{-1}). \tag{7}$$

For $j \in N$, we denote by

$$h'_j = \min(\min_{r \in M} a_{rj}^{-1} \otimes c_r, \min_{r \in M} b_{rj}^{-1} \otimes d_r), \tag{8}$$

and $h' = (h'_1, \ldots, h'_n)^{\mathrm{T}}$.

REMARK 4.1 Each component $h'_j$ for $j \in N$ can be determined in $O(m)$ time. Hence, the vector $h'$ is $O(mn)$.

PROPOSITION 4.1 Let $c \geqslant d$ and $c \neq d$. If $S \neq \emptyset$, then for every $x \in S^{\min}$ there is an $i \in M^{>}$ and $j \in N$ such that $(B \otimes x)_i = c_i \geqslant (A \otimes x)_i$ and $x_j = c_i \otimes b_{ij}^{-1}$.

*Proof.* Suppose $c \geqslant d$ and $c \neq d$. Let $x \in S^{\min}$ such that $\exists i \in M^{>}, j \in N, x_j < c_i \otimes b_{ij}^{-1}$. Then we have

$$(B \otimes x)_i < c_i \leqslant (A \otimes x)_i \oplus c_i,$$

which contradicts the assumption that $x \in S$. Hence, $x_j \geqslant c_i \otimes b_{ij}^{-1}, \forall i \in M^{>}$. If $x_j = c_i \otimes b_{ij}^{-1}$ for some $i \in M^{>}$, then the statement follows. Suppose $\forall i \in M^{>}, j \in N, x_j > c_i \otimes b_{ij}^{-1}$. Since $x \in S$, we have

$$(A \otimes x)_i \oplus c_i = (B \otimes x)_i, \quad i \in M^{>}, \tag{9}$$

$$(A \otimes x)_r \oplus c_r = (B \otimes x)_r \oplus c_r, \quad r \notin M^{>}. \tag{10}$$

Now, $x_j > c_i \otimes b_{ij}^{-1}, \forall i \in M^{>}$, $j \in N$ and $x \in S$, therefore from (9), we have

$$(A \otimes x)_i = (B \otimes x)_i > c_i, \quad \forall\, i \in M^{>}, \tag{11}$$

and it follows from Lemma 4.1 that $\alpha \otimes x = x^{'} \in S$, where in this case $\alpha$ as defined in (7) is

$$\alpha = \max_{i \in M^{>}} (c_i \otimes (A \otimes x)_i^{-1}) < 0.$$

Clearly, $f(x^{'}) < f(x)$, a contradiction with optimality of $x$. □

### 4.2 *The m-by-one case*

Given $f \in \mathbb{R}$ and $a, b, c, d \in \mathbb{R}^{m\times 1}$, the problem of minimizing the function $f(x) = f \otimes x$ (of one variable) subject to

$$\begin{aligned} a_i \otimes x \oplus c_i &= b_i \otimes x, \quad i \in M^{>}, \\ a_i \otimes x \oplus c_i &= b_i \otimes x \oplus d_i, \quad i \notin M^{>} \end{aligned} \tag{12}$$

is denoted $\text{MLP}_{m1}^{\min}$. We define by $M^{=} = \{i \in M; c_i = d_i\}$ and $\overline{M^{=}} = \{i \in M^{=}; a_i \neq b_i\}$. We denote by $S_{m1}$ the solution set of (12) and $S_{m1}^{\min}$ the set of minimizers of (12).

PROPOSITION 4.2 Let $c \geqslant d$ and $c \neq d$. Then
(i) If $\exists i \in M^{>}$ such that $a_i > b_i$, then $S_{m1} = \emptyset$.
(ii) If $\exists i \in M^{>}$ such that $a_i < b_i$, then $S_{m1} \neq \emptyset$ implies that $S_{m1} = \{c_i \otimes b_i^{-1}\}$.

*Proof.* (i) Suppose there is an equation $i \in M^{>}$ such that $a_i > b_i$. Then it follows from (12) that

$$b_i \otimes x < a_i \otimes x \leqslant a_i \otimes x \oplus c_i.$$

Thus, $S_{m1} = \emptyset$.
(ii) Suppose there is an equation $i \in M^{>}$ such that $a_i < b_i$. Then from (12), we have

$$\begin{aligned} a_i \otimes x \oplus c_i &= b_i \otimes x \\ \Longrightarrow\ c_i &= b_i \otimes x \\ \Longrightarrow\ x &= c_i \otimes b_i^{-1}. \end{aligned}$$

Therefore, it follows that if $S_{m1} \neq \emptyset$ then $x = c_i \otimes b_i^{-1} \in S_{m1}$ and it is unique. □

Due to Proposition 4.2, we recognize that there is at most one feasible solution to $\text{MLP}_{m1}^{\min}$ if $\exists i \in M^{>}$ such that $a_i > b_i$ or $a_i < b_i$. For that reason, we may assume that without loss of generality in $\text{MLP}_{m1}^{\min}$, we have $a_i = b_i$ for all $i \in M^{>}$.

Let us define by

$$\tilde{x} = \max_{i \in M^{>}} (c_i \otimes b_i^{-1}). \tag{13}$$

PROPOSITION 4.3 Let $a_i = b_i$ for all $i \in M^{>}$. If $S_{m1} \neq \emptyset$, then $\tilde{x} \in S_{m1}$.

*Proof.* Let $a_i = b_i$ for all $i \in M^{>}$. Suppose $\exists x \in S_{m1}$. It follows from (12) that

$$b_i \otimes x = a_i \otimes x \oplus c_i, \quad i \in M^{>} \tag{14}$$

and

$$b_i \otimes x \oplus d_i = a_i \otimes x \oplus c_i, \quad i \in M^{=}. \tag{15}$$

From (14), we have

$$\begin{aligned} & b_i \otimes x \geqslant c_i, \quad \forall\, i \in M^{>} \\ \Longrightarrow\ & x \geqslant c_i \otimes b_i^{-1}, \quad \forall\, i \in M^{>} \\ \Longrightarrow\ & x \geqslant \max_{i \in M^{>}} (c_i \otimes b_i^{-1}) = \tilde{x}. \end{aligned}$$

Now, if $x > \tilde{x}$, then $b_i \otimes x = a_i \otimes x$, $i \in M^{>}$. It follows from Lemma 4.1 that we can scale down $x$ to the level, where $b_i \otimes x = c_i$ for some $i \in M^{>}$ . Let $\alpha = \tilde{x} \otimes x^{-1}$. Then we have $\alpha \otimes x = \tilde{x} \in S_{m1}$. Hence the proof. □

ALGORITHM 4.1 ONEVARIABLE (Non-homogeneous max-linear systems with one variable)
**Input**: $c = (c_i), d = (d_i), A = (a_i), B = (b_i) \in \mathbb{R}^{m \times 1}$.
**Output**: $x \in S_{m1}$ or an indication that $S_{m1} = \emptyset$.

1. If $c = d$ then stop ($x = h^{'}$ where $h^{'}$ is defined in (8)) and $x \in S_{m1}$
2. for all $i \in M^{>}$ do
   begin
   if $a_i > b_i$ then stop ('$S_{m1} = \emptyset$')
   if $a_i < b_i$ then
   begin
   $x := c_i \otimes b_i^{-1}$
   if $x \in S_{m1}$ then stop ($x \in S_{m1}$)
   else stop ("$S_{m1} = \emptyset$")
   end
   end
3. $x := \max\{c_i - b_i; i \in M^{>}\}$
   if $x \in S_{m1}$ then stop ($x \in S_{m1}$)
   else stop ('$S_{m1} = \emptyset$')

THEOREM 4.2 *Algorithm ONEVARIABLE is correct and its computational complexity is* O($m$).

It follows from Proposition 4.3 that the existence of an optimal solution for $\text{MLP}_{m1}^{\min}$ is reduced to the checking whether $\tilde{x} \in S_{m1}$.

PROPOSITION 4.4 If $a_i = b_i$ for all $i \in M^{>}$ and $S_{m1} \neq \emptyset$ then $\tilde{x} \in S_{m1}^{\min}$.

*Proof.* Suppose $a_i = b_i$ for all $i \in M^{>}$ and $x \in S_{m1}$ such that $x < \tilde{x} = \max_{i \in M^{>}} (c_i \otimes b_i^{-1})$. Therefore, we have

$$\begin{aligned} & x < c_k \otimes b_k^{-1} \text{ for some } k \in M^{>} \\ \Longrightarrow\ & b_k \otimes x < c_k, \end{aligned}$$

which contradicts the assumption that $x \in S_{m1}$. Therefore, $\tilde{x} \in S_{m1}^{\min}$. □

Based on Propositions 4.2, 4.3 and 4.4, we will formulate the following algorithm for finding a solution to $\text{MLP}^{\min}_{m1}$ or to indicate that $S_{m1} = \emptyset$. We denote $\inf_{x \in S_{m1}} f(x)$ by $f^{\min}$.

The following algorithm solves the $\text{MLP}^{\min}_{m1}$ in polynomial time or finds out that $S_{m1} = \emptyset$ or the objective function is unbounded.

ALGORITHM 4.2 MAXLINMINm1 (Max-linear minimization for $m \times 1$ matrices with $c \geqslant d$)
**Input**: $f \in \mathbb{R}$, $c = (c_i)$, $d = (d_i)$, $A = (a_i)$, $B = (b_i) \in \mathbb{R}^{m \times 1}$.
**Output**: $x \in S^{\min}_{m1}$, an indication that $S_{m1} = \emptyset$ or $f^{\min} = -\infty$.

1. If $c = d$ then stop ('$f^{\min} = -\infty$')
2. Use the Algorithm ONEVARIABLE to find an $x \in S_{m1}$
   if it exists then $x \in S^{\min}_{m1}$
   else $S_{m1} = \emptyset$

THEOREM 4.3 *Algorithm MAXLINMINm1 is correct and its computational complexity is* $O(m)$.

### 4.3 *The m-by-two case*

In this section, we will study and develop polynomial solution methods for max-linear programs with two-sided constraints (MLP) with $m \times 2$ entry matrices for minimization problems. That is,

$$\begin{aligned} & f^{\mathrm{T}} \otimes x \longrightarrow \min \\ & \text{subject to} \\ & A \otimes x \oplus c = B \otimes x \oplus d, \end{aligned} \tag{16}$$

where $A = (a_{ij})$, $B = (b_{ij}) \in \mathbb{R}^{m \times 2}$, $f = (f_1, f_2)^{\mathrm{T}} \in \mathbb{R}^2$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}$ and $d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$.

An optimization problem of this type will be denoted by $\text{MLP}^{\min}_{m2}$. The set of feasible solutions for $\text{MLP}^{\min}_{m2}$ will be denoted by $S_{m2}$, the set of optimal solutions by $S^{\min}_{m2}$.

PROPOSITION 4.5 Solving $\text{MLP}^{\min}_{m2}$ can be converted to a repeated solving of problem $\text{MLP}^{\min}_{m1}$.

*Proof.* It follows from Proposition 4.1 that at least one of the decision variables of $\text{MLP}^{\min}_{m2}$ can be fixed to $x_j = c_i \otimes b_{ij}^{-1}$ for some $i \in M^{>}$ and $j \in \{1, 2\}$. If this value is substituted on to $\text{MLP}^{\min}_{m2}$, then this problem will have one variable to be determined and the statement follows. □

Again, it follows from Proposition 4.1 that finding a feasible solution to $\text{MLP}^{\min}_{m2}$ is equivalent to checking whether a feasible solution can be found by fixing one of the two decision variables to $x_j = c_i \otimes b_{ij}^{-1}$, for some $i \in M^{>}$ and $j \in \{1, 2\}$. Due to Proposition 4.5, we can determine the optimal value of the free variable by solving the corresponding $\text{MLP}^{\min}_{m1}$. Therefore, we may assume that a feasible solution exists. The use of Algorithm MAXLINMINm1 is applied to the corresponding $\text{MLP}^{\min}_{m1}$ for finding the free variable in order to minimize $f(x)$. The algorithm will first fix $x_1$ at a number of rows and determine $x_2$ using Algorithm MAXLINMINm1, then fix $x_2$ and determine $x_1$ and stop (when all these are done) with an output of either $x \in S^{\min}_{m2}$, an indication that $S_{m2} = \emptyset$ or $f^{\min} = -\infty$, where $f^{\min} = \inf_{x \in S_{m2}} f(x)$. We assume that without loss of generality $c \geqslant d$ otherwise we swap the sides of equations appropriately.

ALGORITHM 4.3 MAXLINMINm2 (Max-linear minimization for $m \times 2$ matrices with $c \geqslant d$)
**Input**: $f = (f_1, f_2)^{\mathrm{T}} \in \mathbb{R}^2, c = (c_1, \ldots, c_m)^{\mathrm{T}}, d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, $A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{m\times 2}$.
**Output**: $x \in S_{m2}^{\min}$, an indication that $S_{m2} = \emptyset$ or $f^{\min} = -\infty$.

1. If $c = d$ then stop ('$f^{\min} = -\infty$')
2. Set $x^0 := (+\infty, +\infty)^{\mathrm{T}}$, $x := x^0$
3. for $j = 1, 2$ do
   begin
    for all $i \in M^{>}$ do
     begin
      $y_j := c_i \otimes b_{ij}^{-1}$
      $k := 3 - j$
      for all $r \in M$ do
       begin
        $A_r^{'} := a_{rk}$
        $B_r^{'} := b_{rk}$
        $c_r^{'} := \max(c_r, a_{rj} + y_j)$
        $d_r^{'} := \max(d_r, b_{rj} + y_j)$
        if $c_r^{'} < d_r^{'}$ then swap the sides in the $r$th equation
        $M_{\mathrm{new}}^{>} := \{r \in M; c_r^{'} > d_r^{'}\}$
       end
      if $M_{\mathrm{new}}^{>} = \emptyset$ then $y_k := h_k^{'}$ where $h_k^{'}$ is defined in (8)
      else find $y_k$ by applying Algorithm MAXLINMINm1 to
      $A^{'}, B^{'}, c^{'}$ and $d^{'}$
      if $A \otimes y \oplus c = B \otimes y \oplus d$ and $f(y) < f(x)$ then $x := y$
     end
   end
  If $x = x^0$ then stop ($S_{m2} = \emptyset$)
  else stop ($x \in S_{m2}^{\min}$)

THEOREM 4.4 *Algorithm MAXLINMINm2 is correct and its computational complexity is* $\mathrm{O}(m)$.

*Proof.* The correctness of MAXLINMINm2 follows from Propositions 4.1 and 4.5. In Step 3, there are three loops. The first loop runs twice, the second in $\mathrm{O}(m)$ time and the third in $\mathrm{O}(m)$ time. The non-fixed variable can be determined in $\mathrm{O}(m)$ and checking whether $y \in S_{m2}$ or not can be done in $\mathrm{O}(m)$ time. Thus, the computational complexity of Algorithm MAXLINMINm2 is $\mathrm{O}(m^2) + \mathrm{O}(m) = \mathrm{O}(m^2)$. □

## 5. The local search methods

In this section, we propose $r$-optimum local search methods for non-linear programs with two-sided max-linear constraints (NLPMs) for $r = 1$ and $r = 2$. The methods will repeatedly use Algorithm MAXLINMINm1 if $r = 1$ and Algorithm MAXLINMINm2 if $r = 2$. More on heuristic methods can be found in Michalewicz & Fogel (2004).

### 5.1 *Steepest descent method with improvement*

The steepest descent method with improvement (SDMI) is a method that will consider all candidates in the neighbourhood of a current feasible solution $x^c$, and one that gives the best objective function value is chosen to compete with the current feasible solution. This chosen solution is denoted as $x$. If $f(x^c)$ is worse than $f(x)$, then $x$ becomes the current feasible solution. Otherwise the method will produce $x^c$ as the output.

5.1.1 *1-optimum local search.* The SDMI will start with an initial feasible solution $x$. This solution can be found in pseudopolynomial time using (say) the alternating method Cuninghame-Green & Butkovič (1991) and is called current feasible solution. Then fix all but one component $x_r$, of the solution $x$. These values for the fixed variables are then substituted in the problem which would yield a one variable problem with column matrices $A' = A_r$ and $B' = B_r$, $r \in N$ and column vectors $c_i' = \max_{j \neq r}(c_i, a_{ij} + x_j)$ and $d_i' = \max_{j \neq r}(d_i, b_{ij} + x_j)$, $i \in M$. Algorithm MAXLINMINm1 is then applied on to $A'$, $B'$, $c'$ and $d'$ to find the best value for $x_r$ in O($m$) time (Theorem 4.3). After finding the best value for $x_r$, the feasible solution $x$ is denoted by $y$ and $y \in N_1(x)$. This process will be repeated for every $r$. Choose among the feasible solutions in the neighbourhood $N_1(x)$ one with smallest objective function value and compare it with the current feasible solution. If the current feasible solution returns worse objective function value, then this chosen solution becomes current feasible solution and the process will be repeated again. Otherwise, the procedure stops and produces the current feasible solution as the output. The following algorithm sums up this method.

ALGORITHM 5.1 ONEOPT-SDMI (One-optimum steepest descent method with improvement for the NLPM)
**Input**: $A = (a_{ij})$, $B = (b_{ij}) \in \mathbb{R}^{m \times n}$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}$, $d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, a non-linear isotone function $f$ and a maximum number of iterations $U$.
**Output**: An $x \in S$.

1. Find $x^0 \in S$ (for example, using the Alternating Method)
2. Set $x := x^0$, $T = (+\infty, \ldots, +\infty)^{\mathrm{T}}$, flag := true and iteration := 0
3. Repeat until flag = false or iteration = $U$
   begin
   for $r = 1$ to $n$ do
   begin
   for $j = 1$ to $n$ do
   begin
   $y_j := x_j$ if $j \neq r$
   end
   $A' := A_r$
   $B' := B_r$
   for all $i \in M$ do
   begin
   $c_i' := \max_{j \neq r}(c_i, a_{ij} + y_j)$
   $d_i' := \max_{j \neq r}(d_i, b_{ij} + y_j)$
   if $c_i' < d_i'$ then swap the sides of the $i$th equation
   $M^{>}_{\text{new}} := \{i \in M; c_i' > d_i'\}$

end
Apply Algorithm MAXLINMINm1 to $A^{'}$, $B^{'}$, $c^{'}$ and $d^{'}$ to find $y_r$
if $f(y) < f(T)$ then $T := y$
end
if $f(T) < f(x)$ then $x := T$
if $x = x^0$ then flag := false ($x$ is the best feasible solution found)
else $x^0 := x$
iteration := iteration + 1
end

5.1.2 *2-optimum local search.* The 2-optimum SDMI starts with an initial feasible solution $x$ which can again be found in pseudopolynomial time using say alternating method Cuninghame-Green & Butkovič (1991). This solution is called a current feasible solution. Repeat the following process for all $r, s = 1, \ldots, n, r \neq s$. Fix all but two components $x_r$ and $x_s$ of the current feasible solution $x$. The values for the fixed variables are then substituted in the problem. This would produce a two variables problem whose matrices and vectors are $A^{'} = (A_r, A_s) \in \mathbb{R}^{m\times 2}$ and $B^{'} = (B_r, B_s) \in \mathbb{R}^{m\times 2}$ and

$$c_i^{'} = \max_{\substack{j\neq r\\ j\neq s}}(c_i, a_{ij} + x_j), \quad i \in M$$

and

$$d_i^{'} = \max_{\substack{j\neq r\\ j\neq s}}(d_i, b_{ij} + x_j), \quad i \in M,$$

where $i \in M$ and $j \in N$. Algorithm MAXLINMINm2 is then applied on to $A^{'}$, $B^{'}$, $c^{'}$ and $d^{'}$ to find the best value of the free variables, $x_r$ and $x_s$ in $O(m^2)$ time (Theorem 4.4). After finding the best value for $x_r$ and $x_s$, the feasible solution $x$ is denoted by $y$ and $y \in N_2(x)$. Select in the neighbourhood $N_2(x)$, a solution with the smallest objective function value and compare it with the current feasible solution. If the current feasible solution returns worse objective function value, then the chosen solution becomes the current feasible solution and the process will be repeated again. Otherwise stop and declare current feasible solution the best feasible solution found. This procedure can be formulated in the following algorithm.

ALGORITHM 5.2 TWOOPT-SDMI (Two-optimum steepest descent method with improvement for the NLPM)
**Input**: $A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{m\times n}$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}, d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, a non-linear isotone function $f$ and a maximum number of iterations $U$.
**Output**: An $x \in S$.

1. Find $x^0 \in S$ (for example using the Alternating Method)
2. Set $x := x^0$, $T = (+\infty, \ldots, +\infty)^{\mathrm{T}}$, flag := true and iteration := 0
3. Repeat until flag = false or iteration = $U$
   begin
   for $r = 1$ to $n$ do
   begin
   for $s = r + 1$ to $n$ do

begin
for $j = 1$ to $n$ do
begin
$y_j := x_j$ if $j \neq r$ and $j \neq s$
end
$A^{'} := (A_r, A_s)$
$B^{'} := (B_r, B_s)$
for all $i \in M$ do
begin
$c_i^{'} := \max(c_i, a_{ij} + y_j)$, $j \neq r$, $j \neq s$
$d_i^{'} := \max(d_i, b_{ij} + y_j)$, $j \neq r$, $j \neq s$
if $c_i^{'} < d_i^{'}$ then swap the sides of the $i$th equation
$M_{\text{new}}^{>} := \{r \in M; c_i^{'} > d_i^{'}\}$
end
Apply Algorithm MAXLINMINm2 to $A^{'}$, $B^{'}$, $c^{'}$ and $d^{'}$
to find $y_r$ and $y_s$
if $f(y) < f(T)$ then $T := y$
end
end
if $f(T) < f(x)$ then $x := T$
if $x = x^0$ then flag := false ($x$ is best feasible solution found)
else $x^0 := x$
iteration := iteration + 1
end

### 5.2 *Steepest descent method without improvement*

In the SDMI, we have seen that a best solution in the neighbourhood of a current feasible solution $x$ is chosen to compete with the current feasible solution. If this solution has a better objective function value than the current feasible solution, then it will become current, and the process will start again. Otherwise, the current feasible solution is the best feasible solution found and the process stops. However, this method may not provide us with the best solution. As a consequence, we may get stuck in a local optimum. To overcome this, we made some modifications to the SDMI so that we allow moves to a best solution $z$ in the neighbourhood of the current solution regardless of the fact that its objective function value may be worse than that of the current feasible solution $x$. This method is called SDMI, for brevity steepest descent method without improvement (SDMW). This is motivated by the fact that it may be possible to find a solution in the neighbourhood of $z$ which provides a better objective function value than that of $f(z)$ and indeed even than in the current solution.

5.2.1 *One-optimum local search.* This second version of the steepest descent method will consider all candidates in the neighbourhood $N_1(x)$ of a current feasible solution $x$ and chooses the one with the best objective function value among these solutions. The method is formulated in the following algorithm.

ALGORITHM 5.3 ONEOPT-SDMW (One-optimum steepest descent method without improvement for the NLPM)

**Input**:$A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{m\times n}$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}, d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, a non-linear isotone function $f$ and a maximum number of iterations $U$.
**Output**: An $x \in S$.

1. Find $x^0 \in S$ (for example using the Alternating Method)
2. Set $x := x^0$, $T = (+\infty, \ldots, +\infty)^{\mathrm{T}}$ and Iteration := 0
3. Repeat until Iteration $= U$
   begin
   for $r = 1$ to $n$ do
   begin
   for $j = 1$ to $n$ do
   begin
   $y_j := x_j$ if $j \neq r$
   end
   $A^{'} := A_r$
   $B^{'} := B_r$
   for all $i \in M$ do
   begin
   $c_i^{'} := \max_{j\neq r}(c_i, a_{ij} + y_j)$
   $d_i^{'} := \max_{j\neq r}(d_i, b_{ij} + y_j)$
   if $c_i^{'} < d_i^{'}$ then swap the sides of the $i$th equation
   $M^{>}_{\text{new}} := \{i \in M; c_i^{'} > d_i^{'}\}$
   end
   Apply Algorithm MAXLINMINm1 on to $A^{'}$, $B^{'}$, $c^{'}$ and $d^{'}$ to find $y_r$
   if $f(y) < f(T)$ then $T := y$
   end
   $x := T$
   $x^0 := x$
   Iteration := Iteration + 1
   end

5.2.2 *Two-optimum local search.* Similarly as for 1-optimum SDMW, the 2-optimum SDMW will begin by finding an initial feasible (current) solution $x$ and chooses in the neighbourhood $N_2(x)$ a feasible solution with the smallest objective function value regardless of the fact that its objective function value may be worse than the current feasible solution $x$. The chosen solution is now current feasible solution and the process starts again. The method stops if the maximum number of iterations (or time) is reached.

ALGORITHM 5.4 TWOOPT-SDMW (Two-optimum steepest descent method without improvement for the NLPM)
**Input**:$A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{m\times n}$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}, d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, a non-linear isotone function $f$ and a maximum number of iteration $U$.
**Output**: An $x \in S$.

1. Find $x^0 \in S$ (for example using the Alternating Method)
2. Set $x := x^0$, $T := (+\infty, \ldots, +\infty)^{\mathrm{T}}$, iteration := 0

3. Repeat until iteration $= U$
begin
for $r = 1$ to $n$ do
begin
for $s = r + 1$ to $n$ do
begin
for $j = 1$ to $n$ do
begin
$y_j := x_j$ if $j \neq r$ and $j \neq s$
end
$A^{'} := (A_r, A_s)$
$B^{'} := (B_r, B_s)$
for all $i \in M$ do
begin
$c_i^{'} := \max(c_i, a_{ij} + y_j)$, $j \neq r$, $j \neq s$
$d_i^{'} := \max(d_i, b_{ij} + y_j)$, $j \neq r$, $j \neq s$
if $c_i^{'} < d_i^{'}$ then swap the sides of the $i^{\text{th}}$ equation
$M_{\text{new}}^{>} := \{r \in M; c_i^{'} > d_i^{'}\}$
end
Apply Algorithm MAXLINMINm2 to $A^{'}$, $B^{'}$, $c^{'}$ and $d^{'}$
to find $y_r$ and $y_s$
if $f(y) < f(T)$ then $T := y$
end
end
$x := T$
$x^0 := x$
iteration $=$ iteration $+ 1$
end

### 5.3 *Hill-descending method*

In both methods described previously, we do not have the guarantee of escaping from local optima. In an attempt to overcoming this problem, we consider modifying those methods so that an immediate solution in the neighbourhood of the current solution with a better objective function value is chosen. This method is called 'hill descending method' (HDM).

5.3.1 *One-optimum local search.* The method will start with a feasible solution $x$, then chooses an immediate solution in the neighbourhood, $N_1(x)$, of $x$ with better objective function value than the current solution and repeats the process until no further improvement is possible or the maximum number of time/iterations are reached. The method is formulated in the following algorithm.

ALGORITHM 5.5 ONEOPT-HDM (One-optimum hill-descending method for the NLPM)
**Input**: $A = (a_{ij})$, $B = (b_{ij}) \in \mathbb{R}^{m \times n}$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}$, $d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, a non-linear isotone function $f$ and a maximum number of iterations $U$.
**Output**: An $x \in S$.

1. Find $x^0 \in S$ (for example using the Alternating Method)
2. Set $x := x^0$ and flag := true, iteration := 0
3. Repeat until flag = false or iteration = $U$

```
begin
 for r = 1 to n do
  begin
   for j = 1 to n do
    begin
     y_j := x_j if r ≠ j
    end
    A' := A_r
    B' := B_r
    for all i ∈ M do
     begin
      c_i' := max(c_i, a_ij + y_j), j ≠ r
      d_i' := max(d_i, b_ij + y_j), j ≠ r
      if c_i' < d_i' then swap sides of the ith equation
      M>_new := {i ∈ M; c_i' > d_i'}
     end
    Apply Algorithm MAXLINMINm1 to A', B', c' and d'
    to find y_r
    if f(y) < f(x) then
     begin
      x := y
      break
     end
  end
 if x = x^0 then flag := false (x is the best solution found)
 else
  begin
   x^0 := x
   iteration := iteration + 1
  end
end
```

5.3.2 *Two-optimum local search.* The HDM will consider the neighbourhood $N_2(x)$ of a current solution $x$. Then chooses immediately the first feasible solution in the neighbourhood with smaller objective function value than the current feasible solution. The chosen solution will become the current feasible and the procedure starts again. If there is no solution in the neighbourhood with better objective function than the current feasible solution the method stops.

ALGORITHM 5.6 TWOOPT-HDM (Two-optimum hill-descending method for the NLPM)
**Input**: $A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{m\times n}$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}, d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, a non-linear isotone function $f$ and a maximum number of iterations $U$.
**Output**: An $x \in S$.

1. Find $x^0 \in S$ (for example using the Alternating Method)
2. Set $x := x^0$, flag := true, $T := 0$, iteration := 0
3. Repeat until flag = false or iteration = $U$

```
begin
 for r = 1 to n do
  begin
   for s = r + 1 to n do
    begin
     for j = 1 to n do
      begin
       y_j := x_j if j ≠ r and j ≠ s
      end
      A' := (A_r, A_s)
      B' := (B_r, B_s)
      for all i ∈ M do
      begin
       c'_i := max(c_i, a_ij + y_j), j ≠ r, j ≠ s
       d'_i := max(d_i, b_ij + y_j), j ≠ r, j ≠ s
       if c'_i < d'_i then swap the sides
       M^>_new := {r ∈ M; c'_i > d'_i}
      end
      Apply Algorithm MAXLINMINm2 to A', B', c' and d'
      to find y_r and y_s
      if f(y) < f(x) then
       begin
        x := y
        T := 1
        break
       end
    end
   if T = 1 then T := 0 and break
  end
 if x = x^0 then flag := false (x is the best solution found)
 else
  begin
   x^0 := x
   iteration := iteration + 1
  end
end
```

### 5.4 *The multi-start heuristic*

The multi-start heuristic method will consider all the one-optimum and two-optimum local search methods proposed in the previous sections and repeat each method for a certain number of times.

Since the method (alternating method) we are using to find the initial feasible solution depends on the starting vector, we consider a positive integer say $R > 1$ and repeat the one-optimum local search

methods $R$ times. For each run, we generate a new starting vector which may in return give a different feasible solution to the problem and then apply the one-optimum or two-optimum local search methods. After repeating the method $R$ times, we find among the $R$ solutions the best solution to our problem. This method is summed up in the following algorithms.

ALGORITHM 5.7 MULTI-START (Multi-start heuristic for the NLPM)
**Input**: $A = (a_{ij}), B = (b_{ij}) \in \mathbb{R}^{m\times n}$, $c = (c_1, \ldots, c_m)^{\mathrm{T}}, d = (d_1, \ldots, d_m)^{\mathrm{T}} \in \mathbb{R}^m$, a non-linear isotone function $f$ and a maximum number of iterations $R$.
**Output**: A solution $x^{\text{best}} \in S$.

1. Set run $:= 0$ and $x^{\text{best}} := (+\infty, \ldots, +\infty)^{\mathrm{T}}$
2. Randomly find a starting vector $x(\text{run})$
3. Use $x(\text{run})$ and apply the Alternating Method to find a solution $x^0$ to $A \otimes x \oplus c = B \otimes x \oplus d$
4. Apply a 1−optimum (or 2-optimum) local search method starting from $x^0$ to find a solution $x(\text{run})$
5. if $f(x(\text{run})) < f(x^{\text{best}})$ then $x^{\text{best}} := x(\text{run})$
6. if run $= R$ then stop ($x^{\text{best}}$ is the best feasible solution found)
   else run $:=$ run $+ 1$ and go to 3.

## 6. Computational results

In this section, we will compare the 1-optimum and 2-optimum local search methods, developed in the previous sections. We divide this section into two subsections: in the first subsection, we consider the objective function to be max-linear and present results obtained when comparing exact method for solving NLPM and $r$-optimum local search methods where $r = 1, 2$. In the second subsection, the objective function is $f(x) = 2^{x_1} + 2^{x_2} + \ldots + 2^{x_n}$ and we compare the 1-optimum and 2-optimum local search methods.

All the algorithms developed for both heuristics and exact methods were implemented in Matlab 7.2. Matrices and vectors are generated randomly. The experiments were carried out on a PC with Intel Centrino Duo, 1.66.GHz of CPU and 0.99 GB of RAM.

### 6.1 *Instances when the objective function is max-linear*

In this section, we consider NLPM when the objective function is max-linear. First, we show the performance of multi-start heuristics, then compare the exact method and $r$-optimum local search methods where $r = 1, 2$. We begin by finding a feasible solution to the given problem (using the alternating method). Then each of the methods will use this feasible solution in order to find an optimal or a best feasible to the given problem. For each method, we report the computing time spent up to the point where its optimal solution or best feasible solution is found for the first time. In the 1-optimum and 2-optimum local search methods, we impose a maximum number of iterations as five (5) for the SDMW.

Recall that $f^{min} = \inf_{x\in S} f(x)$ denotes the optimal value of the objective function. The size of matrices and range of entries for the problems are given at the end of the tables. Entries in each table have the following meaning:
FV: Feasible objective function value;
OFV: Optimal objective function value;
Sol: The best feasible objective function value found;

# times best: How many times the method produces the best solution among the heuristics;
# times fast: How many times the method produces solution faster than all other methods in the table;
# times exact: How many times the heuristic method finds an exact solution;
RE: The relative error determined as follows

$$\mathrm{RE} = \left| \frac{\mathrm{Sol} - \mathrm{OFV}}{\mathrm{OFV}} \right|. \tag{17}$$

Note that RE is used only for instances whose optimal objective function value is positive.

Tables 1 and 2 show that the multi-start heuristic is not improving the result obtained by the local search methods.

Table 3 reports the result on comparison of exact, 1-optimum and 2-optimum local search methods. The objective function is max-linear and the input matrices have 10 rows and 20 columns. All the methods were able to produce an exact solution for six (6) instances except HDM for 2-optimum local search which produces four (4). Each of the five methods has average RE = 0.04 and HDM for 2-optimum has average RE = 0.34. The fastest among all the methods is the HDM for 1-optimum local search with average computing time of 0.04 s. The exact method is slow with the average computing time of 216.3 s.

Table 4 gives the result on the comparison of exact, 1-optimum and 2-optimum local search methods. The matrices have 10 rows and fifty columns. SDMI, HDM for 1-optimum and SDMI for 2-optimum local were able to find exact solution in three instances. Each of these methods have average RE = 0.78. SDMW for 1-optimum has average RE = 2.90. SDMW and HDM for 2-optimum has average RE = 1.57 and RE = 2.56, respectively. The fastest method is SDMW for 1-optimum with average computing time of 0.1 s. The exact method is once more very slow with average computing time of 218.3 s.

Table 5 reports the output on the comparison of exact, 1-optimum and 2-optimum local search methods. The matrices have 10 rows and 100 columns. In the 1-optimum local search SDMI and HDM were able to find exact solution twice. While in the 2-optimum only SDMI is able to find exact solution in two instances. The average RE for SDMI, HDM in 1-optimum and SDMI in 2-optimum is 0.7. The fastest method is the 1-optimum SDMW with average computing time of 1.3 s. The exact method is slow with average computing time of 131.7 s.

Table 6 gives the output on the comparison of exact, 1-optimum and 2-optimum local search methods. The matrices have 40 rows and 100 columns. All the methods produced the same solution with average RE = 1.16. 1-optimum SDMI is the fastest method with average computing time of 0.5 s. The exact method is slow with average computing time of 6958.7 s.

We conclude that the 1-optimum SDMI, HDM and 2-optimum SDMI are the best solution methods that can produce a solution with minimum objective function value. The method that can produce solution with minimum computing time is 1-optimum SDMI.

### 6.2 *Instances when the objective function is not max-linear*

In this section, we will consider the objective function is $f(x) = 2^{x_1} + 2^{x_2} + \ldots + 2^{x_n}$. We compare performance of all the local search methods presented in Section 5. For each problem, we find a feasible solution using the alternating method. Each of the heuristic methods will use this feasible solution to find a best feasible solution. We report computing time each method used to find a best feasible solution for the first time. As with the other experiments, the maximum number of iterations for SDMW for both 1-optimum and two-optimum local search methods is fixed as five (5).

TABLE 1 *Results on the comparison of performance of exact method, 1-optimum local search methods and multi-start heuristics on NLPM with max-linear objective function. The matrices have* 20 *rows and up to* 300 *columns*

| | | | Exact integer maxlinmin | | 1-optimum local search | | | | | | Multi-start(10) 1-optimum local search | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | SDMI | | SDMW | | HDM | | SDMI | | SDMW | | HDM | |
| $(m, n)$ | Range | FV | OFV | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time |
| (20, 40) | $(0, 10^5)$ | 1065 | 336 | 3.2 | 336 | 1.1 | 336 | 2.3 | 336 | 1.2 | – | – | – | – | – | – |
| (20, 60) | $(0, 10^5)$ | 3348 | −3320 | 541.4 | −2802 | 2.9 | −1613 | 5.1 | −2804 | 1.7 | −2804 | 6.2 | −2804 | 12.7 | −2804 | 5.6 |
| (20, 80) | $(0, 10^5)$ | 11510 | 4242 | 680.0 | 4242 | 1.0 | 4242 | 2.0 | 4242 | 1.3 | – | – | – | – | – | – |
| (20, 100) | $(0, 10^5)$ | 5092 | −1853 | 5.6 | −1254 | 2.9 | 1597 | 1.8 | 1255 | 1.7 | −1255 | 9.0 | −578 | 10.6 | −1256 | 6.2 |
| (20, 120) | $(0, 10^5)$ | 9348 | 1744 | 183.9 | 3867 | 2.2 | 6613 | 1.7 | 3867 | 2.8 | 3865 | 17.5 | 5264 | 14.5 | 3866 | 14.0 |
| (20, 140) | $(0, 10^5)$ | 10439 | 1012 | 289.9 | 7092 | 1.2 | 7092 | 1.8 | 7092 | 1.2 | 7091 | 5.9 | 7091 | 13.6 | 7090 | 4.0 |
| (20, 160) | $(0, 10^5)$ | 9726 | −2408 | 48.9 | −1493 | 7.1 | 3842 | 2.2 | −1493 | 7.0 | −1495 | 67.3 | 2640 | 15.4 | −1495 | 36.4 |
| (20, 180) | $(0, 10^5)$ | 8775 | −2217 | 39.2 | −977 | 8.7 | 6230 | 2.2 | −977 | 8.8 | −978 | 37.2 | 3937 | 11.3 | −978 | 13.4 |
| (20, 200) | $(0, 10^5)$ | 11022 | −3733 | 12.9 | −3371 | 23.3 | 4912 | 2.6 | −3371 | 23.4 | −3372 | 78.5 | −1543 | 11.4 | −3371 | 81.6 |
| (20, 220) | $(0, 10^5)$ | 9539 | 1346 | 180.2 | 2599 | 10.1 | 7040 | 4.0 | 5681 | 22.6 | 2599 | 40.9 | 6123 | 32.3 | 3529 | 70.9 |
| (20, 240) | $(0, 10^5)$ | 8743 | −3379 | 30.1 | −1972 | 26.6 | 5756 | 3.1 | −1972 | 27.7 | −2560 | 220 | 1210 | 30.1 | −2661 | 107,7 |
| (20, 260) | $(0, 10^5)$ | 14298 | 159 | 47.1 | 5064 | 4.4 | 9158 | 3.2 | 3137 | 6.6 | 3137 | 54.8 | 5949 | 29.9 | 3137 | 32.1 |
| (20, 280) | $(0, 10^5)$ | 10975 | −1977 | 548.8 | 2207 | 10.9 | 7501 | 4.1 | 2193 | 12.0 | 926 | 112.1 | 6050 | 30.0 | −58 | 108.8 |
| (20, 300) | $(0, 10^5)$ | 8949 | −2676 | 81.1 | −2323 | 34.9 | 5778 | 6.2 | −2323 | 25.3 | −2324 | 162.2 | 4557 | 39.2 | −2607 | 133.9 |

TABLE 2 *Results on the comparison of performance of exact method, 2-optimum local search methods and multi-start heuristics on NLPM with max-linear objective function. The matrices have* 20 *rows and up to* 160 *columns*

| | | | Exact integer maxlinmin | | 2-optimum local search | | | | | | Multi-start(10) 2-optimum local search | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | SDMI | | SDMW | | HDM | | SDMI | | SDMW | | HDM | |
| $(m, n)$ | Range | FV | OFV | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time |
| $(20, 40)$ | $(0, 10^5)$ | 9702 | 8857 | 277.3 | 9423 | 2.2 | 9423 | 2.2 | 9423 | 0.6 | – | – | – | – | – | – |
| $(20, 60)$ | $(0, 10^5)$ | 9346 | 1625 | 231.7 | 5937 | 11.6 | 5937 | 25.0 | 5937 | 6.2 | 5928 | 108.2 | 5926 | 236.3 | 5924 | 49.3 |
| $(20, 80)$ | $(0, 10^5)$ | 14835 | 4936 | 1176.3 | 6492 | 50.1 | 6810 | 34.1 | 6492 | 29.7 | 6492 | 506.6 | 6800 | 323.5 | 6492 | 97.5 |
| $(20, 100)$ | $(0, 10^5)$ | 12268 | 1478 | 378.4 | 4422 | 55.2 | 5193 | 36.9 | 6653 | 9.1 | 4416 | 525.9 | 5181 | 317.5 | 6642 | 11.1 |
| $(20, 120)$ | $(0, 10^5)$ | 12466 | 1304 | 407.2 | 8482 | 15.6 | 8482 | 34.8 | 8482 | 11.5 | 8470 | 516.4 | 8466 | 361.5 | 8467 | 72.7 |
| $(20, 140)$ | $(0, 10^5)$ | 12270 | 2072 | 700.9 | 6084 | 68.7 | 6106 | 50.4 | 6084 | 17.1 | 6079 | 677.6 | 6091 | 481.4 | 6071 | 149.4 |
| $(20, 160)$ | $(0, 10^5)$ | 12542 | 1624 | 641.6 | 7329 | 61.2 | 7329 | 61.8 | 7329 | 17.0 | 7311 | 586.1 | 7312 | 600.2 | 7319 | 137.8 |

TABLE 3 *Results on the comparison of performance of exact method and r-optimum local search methods,* $r = 1, 2$. *The matrices have* 10 *rows and* 20 *columns. Range of entries is* $(0, 10^5)$

| | Exact integer maxlinmin | | 1-optimum local search | | | | | | | | | 2-Optimum local search | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SDMI | | | SDMW | | | HDM | | | SDMI | | | SDMW | | | HDM | | |
| FV | OFV | Time | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE |
| 13643 | 12207 | 100.2 | 13569 | 6.5 | 0.11 | 13569 | 3.0 | 0.11 | 13569 | 1.0 | 0.11 | 13569 | 57.7 | 0.11 | 13569 | 24.0 | 0.11 | 13643 | 1.6 | 0.11 |
| 2710 | −9180 | 1.9 | −7427 | 1.9 | | −7427 | 4.0 | | −7427 | 1.1 | | −7427 | 5.2 | | −7427 | 8.2 | | 2710 | 2.6 | |
| 16236 | 15361 | 861.3 | 16236 | 1.6 | 0.06 | 16236 | 2.7 | 0.06 | 16236 | 1.1 | 0.06 | 16236 | 8.4 | 0.06 | 16236 | 7.5 | 0.06 | 16236 | 2.1 | 0.06 |
| 20926 | 16318 | 367.8 | 20926 | 0.0 | 0.28 | 20926 | 0.1 | 0.28 | 20926 | 0.1 | 0.28 | 20926 | 2.4 | 0.28 | 20926 | 3.5 | 0.28 | 20926 | 1.8 | 0.28 |
| 6381 | 468 | 12.6 | 468 | 0.0 | 0.0 | 468 | 0.1 | 0.0 | 468 | 0.1 | 0.0 | 468 | 1.0 | 0.0 | 468 | 2.5 | 0.0 | 468 | 0.5 | 0.0 |
| 13009 | 10892 | 207.0 | 10892 | 0.1 | 0.0 | 10892 | 0.1 | 0.0 | 10892 | 0.0 | 0.0 | 10892 | 1.1 | 0.0 | 10892 | 2.8 | 0.0 | 13009 | 0.6 | 0.19 |
| 6691 | 3342 | 18.1 | 3342 | 0.1 | 0.0 | 3342 | 0.1 | 0.0 | 3342 | 0.1 | 0.0 | 3342 | 1.5 | 0.0 | 3342 | 2.6 | 0.0 | 5967 | 0.5 | 0.79 |
| 12374 | 11221 | 418.1 | 11221 | 0.0 | 0.0 | 11221 | 0.1 | 0.0 | 11221 | 0.0 | 0.0 | 11221 | 0.9 | 0.0 | 11221 | 2.3 | 0.0 | 12374 | 0.5 | 0.0 |
| 6334 | −3899 | 1.0 | −3899 | 0.1 | | −3899 | 0.1 | | −3899 | 0.1 | | −3899 | 2.1 | | −3899 | 2.7 | | −674 | 0.5 | |
| 11086 | 8590 | 174.7 | 8590 | 0.0 | 0.0 | 8590 | 0.1 | 0.0 | 8590 | 0.1 | 0.0 | 8590 | 0.9 | 0.0 | 8590 | 2.3 | 0.0 | 8590 | 0.5 | 0.0 |
| Average | 6532 | 216.3 | 7391.8 | 1.0 | 0.04 | 7391.8 | 1.0 | 0.04 | 7391.8 | 0.4 | 0.04 | 7391.8 | 8.1 | 0.04 | 7391.8 | 5.8 | 0.04 | 9774.9 | 1.1 | 0.34 |
| # times best | | | 10 | | | 10 | | | 10 | | | 10 | | | 10 | | | 3 | | |
| # times fastest | | | | 6 | | | 2 | | | 7 | | | 0 | | | 0 | | | 0 | |
| # times exact | | | 6 | | | 6 | | | 6 | | | 6 | | | 6 | | | 4 | | |

TABLE 4 *Results on the comparison of performance of exact method and r-optimum local search methods,* $r = 1, 2$. *The matrices have* 10 *rows and* 50 *columns. Range of entries is* $(0, 10^5)$

| | Exact integer maxlinmin | | 1-optimum local search | | | | | | | | | 2-optimum local search | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SDMI | | | SDMW | | | HDM | | | SDMI | | | SDMW | | | HDM | | |
| FV | OFV | Time | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE |
| 120913 | 1249 | 155.3 | 1249 | 0.4 | 0.0 | 12594 | 0.1 | 9.08 | 1249 | 0.5 | 0.0 | 1249 | 14.3 | 0.0 | 7112 | 12.6 | 4.69 | 12030 | 3.7 | 8.63 |
| 8450 | 1100 | 11 .9 | 4138 | 0.1 | 2.76 | 7486 | 0.1 | 5.81 | 4137 | 0.1 | 2.76 | 4138 | 4.0 | 2.76 | 4138 | 3.2 | 2.76 | 4138 | 1.5 | 2.76 |
| 24764 | 3619 | 518.6 | 5414 | 0.2 | 0.50 | 11434 | 0.1 | 2.16 | 5414 | 0.3 | 0.50 | 5414 | 8.8 | 0.50 | 7681 | 3.3 | 1.11 | 11434 | 1.0 | 2.16 |
| 16437 | 2224 | 181.3 | 3829 | 0.2 | 0.72 | 8775 | 0.1 | 2.95 | 3829 | 0.2 | 0.75 | 3829 | 7.0 | 0.72 | 5311 | 4.3 | 1.39 | 16437 | 0.7 | 6.39 |
| 15434 | 2116 | 267.5 | 5701 | 0.2 | 1.69 | 10109 | 0.1 | 3.78 | 5701 | 0.2 | 1.69 | 5701 | 5.6 | 1.69 | 8344 | 3.6 | 2.94 | 8635 | 1.3 | 3.08 |
| 27314 | 2932 | 253.7 | 4209 | 0.2 | 0.43 | 8898 | 0.1 | 2.03 | 4209 | 0.2 | 0.43 | 4209 | 8.0 | 0.43 | 4903 | 4.0 | 0.67 | 4209 | 1.6 | 0.43 |
| −4000 | −15041 | 1.1 | −15041 | 0.4 | | −8289 | 0.0 | | −15041 | 0.4 | | −15041 | 16.7 | | −9954 | 3.4 | | −12243 | 1.9 | |
| 13562 | 7132 | 744.8 | 13239 | 0.0 | 0.86 | 13239 | 0.1 | 0.86 | 13239 | 0.0 | 0.86 | 13239 | 1.4 | 0.86 | 13239 | 3.5 | 0.86 | 13239 | 1.0 | 0.86 |
| 20453 | 4707 | 46.2 | 8581 | 0.1 | 0.82 | 11513 | 0.1 | 1.45 | 8581 | 0.2 | 0.82 | 8581 | 4.5 | 0.82 | 8581 | 3.6 | 0.82 | 8581 | 1.5 | 0.82 |
| −137 | −7470 | 2.7 | −7470 | 0.18 | | −4617 | 0.1 | | −7470 | 0.2 | | −7470 | 7.8 | | −6440 | 4.5 | | −5199 | 1.3 | |
| Average | 256.8 | 218.3 | 238.49 | 0.2 | 0.78 | 7114.2 | 0.1 | 2.90 | 2384.8 | 0.2 | 0.78 | 2384.8 | 7.8 | 0.78 | 4291.5 | 4.6 | 1.57 | 6126.1 | 1.6 | 2.56 |
| # times best | | – | 9 | | | 1 | | | 0 | | | 9 | | | 3 | | | 4 | | |
| # times fastest | | | | 3 | | | 9 | | | 2 | | | 0 | | | 0 | | | 0 | |
| # times exact | | | 3 | | | 0 | | | 3 | | | 3 | | | 0 | | | 0 | | |

TABLE 5 *Results on the comparison of performance of exact method and r-optimum local search methods,* $r = 1, 2$. *The matrices have* 10 *rows and* 100 *columns. Range of entries is* $(0, 10^5)$

| | Exact integer maxlinmin | | 1-optimum local search | | | | | | | | | 2-optimum local search | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SDMI | | | SDMW | | | HDM | | | SDMI | | | SDMW | | | HDM | | |
| FV | OFV | Time | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE |
| 17999 | 536 | 36.4 | 536 | 1.8 | 0.0 | 8399 | 0.2 | 14.67 | 536 | 2.1 | 0.0 | 536 | 100.5 | 0.0 | 7155 | 9.1 | 12.35 | 2134 | 5.1 | 2.98 |
| 25215 | 3945 | 71.3 | 4309 | 1.5 | 0.02 | 15011 | 0.2 | 2.81 | 4309 | 1.7 | 0.02 | 4309 | 83.6 | 0.02 | 12224 | 9.5 | 2.10 | 9053 | 3.2 | 1.30 |
| 22343 | 2403 | 42.6 | 5933 | 1.0 | 1.47 | 14004 | 0.2 | 4.82 | 5536 | 1.1 | 1.30 | 5536 | 61.6 | 1.30 | 12539 | 9.5 | 4.21 | 5933 | 4.8 | 1.47 |
| 16702 | −1978 | 2.1 | −1978 | 2.0 | | 10702 | 0.2 | | −1978 | 2.1 | | −1978 | 110.0 | | 7715 | 9.7 | | 211 | 5.6 | |
| 17392 | −3941 | 25.6 | −2509 | 2.2 | | 10629 | 0.2 | | −2509 | 2.3 | | −2509 | 117.0 | | 7816 | 9.0 | | 3839 | 3.6 | |
| 18000 | 94 | 10.2 | 103 | 1.7 | 0.10 | 1211 | 0.2 | 11.8 | 103 | 1.9 | 0.10 | 103 | 87.0 | 0.10 | 893 | 8.9 | 8.50 | 3191 | 4.4 | 2.39 |
| 14302 | 1013 | 209.6 | 3945 | 0.9 | 2.89 | 10239 | 0.2 | 9.11 | 3943 | 0.8 | 2.89 | 3945 | 44.7 | 2.89 | 9004 | 9.0 | 7.89 | 3945 | 5.2 | 2.89 |
| 18591 | 1514 | 27.5 | 3601 | 1.2 | 1.38 | 10111 | 0.2 | 5.68 | 3600 | 1.4 | 1.38 | 3601 | 65.5 | 1.38 | 8617 | 8.9 | 4.89 | 5567 | 4.1 | 2.68 |
| 19596 | −4989 | 115.6 | −2962 | 1.5 | | 7747 | 0.2 | | −2962 | 1.7 | | −2962 | 81.2 | | 3756 | 8.7 | | 632 | 3.5 | |
| 20311 | 3041 | 775.8 | 4203 | 1.3 | 0.38 | 13804 | 0.7 | 3.30 | 4202 | 1.2 | 0.38 | 4203 | 63.3 | 0.38 | 11502 | 9.3 | 0.78 | 4446 | 5.0 | 0.46 |
| Average | 163.8 | 131.7 | 1491.1 | 1.5 | 0.7 | 9095.4 | 0.3 | 6.49 | 1490.7 | 1.6 | 0.68 | 1491.1 | 75.3 | 0.68 | 8926.2 | 9.2 | 5.05 | 3607.9 | 4.5 | 1.84 |
| # times best | | | 6 | | | 0 | | | 10 | | | 6 | | | 0 | | | 0 | | |
| # times fastest | | | | 0 | | | 10 | | | 0 | | | 0 | | | 0 | | | 0 | |
| # times exact | | | 2 | | | 0 | | | 2 | | | 2 | | | 0 | | | 0 | | |

TABLE 6 *Results on the comparison of performance of exact method and r-optimum local search methods,* $r = 1, 2$. *The matrices have 40 rows and* 100 *columns. Range of entries is* $(0, 10^5)$

| | Exact integer maxlinmin | | 1-optimum local search | | | | | | | | | 2-optimum local search | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SDMI | | | SDMW | | | HDM | | | SDMI | | | SDMW | | | HDM | | |
| FV | OFV | Time | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE | Sol | Time | RE |
| 7557 | 4349 | 1151.8 | 7463 | 0.3 | 0.71 | 7463 | 0.9 | 0.71 | 7461 | 0.5 | 0.71 | 7463 | 80.9 | 0.71 | 7463 | 204.0 | 0.71 | 7463 | 40.6 | 0.71 |
| 8256 | 5614 | 1097.3 | 8256 | 0.2 | 0.47 | 8256 | 1.0 | 0.47 | 8255 | 1.1 | 0.47 | 8256 | 42.5 | 0.47 | 8256 | 211.9 | 0.47 | 8256 | 42.9 | 0.47 |
| 4655 | 1228 | 356.0 | 3616 | 0.5 | 1.57 | 3616 | 1.0 | 1.57 | 3615 | 1.0 | 1.57 | 3616 | 85.5 | 1.57 | 3616 | 217.4 | 1.57 | 3616 | 45.6 | 1.57 |
| 7924 | 3880 | 1233.7 | 7633 | 1.1 | 0.97 | 7633 | 1.5 | 0.97 | 7632 | 1.0 | 0.97 | 7633 | 94.0 | 0.97 | 7633 | 238.2 | 0.97 | 7633 | 35.6 | 0.97 |
| 6883 | 3182 | 932.6 | 6021 | 0.5 | 0.89 | 6021 | 1.0 | 0.89 | 6020 | 1.1 | 0.89 | 6021 | 83.4 | 0.89 | 6021 | 210.6 | 0.89 | 6021 | 49.5 | 0.89 |
| 8005 | 4009 | 1173.5 | 6404 | 1.5 | 0.60 | 6404 | 1.4 | 0.60 | 6404 | 1.7 | 0.60 | 6404 | 90.8 | 0.60 | 6404 | 230.8 | 0.60 | 6404 | 45.5 | 0.60 |
| 6534 | 3033 | 74.2 | 5477 | 0.3 | 0.81 | 5477 | 1.0 | 0.81 | 5477 | 0.8 | 0.81 | 5477 | 84.4 | 0.81 | 5477 | 212.2 | 0.81 | 5477 | 38.4 | 0.81 |
| 5495 | 4064 | 510.7 | 4697 | 0.4 | 0.22 | 4697 | 1.1 | 0.22 | 4696 | 0.7 | 0.22 | 4697 | 95.5 | 0.22 | 4697 | 240.0 | 0.22 | 4697 | 45.5 | 0.22 |
| 4831 | 504 | 107.9 | 2766 | 0.3 | 4.49 | 2766 | 1.0 | 4.49 | 2766 | 0.5 | 4.49 | 2766 | 90.4 | 4.49 | 2766 | 232.0 | 4.49 | 2766 | 46.0 | 4.49 |
| 5682 | 1428 | 321.0 | 2617 | 1.1 | 0.83 | 2617 | 1.5 | 0.83 | 2617 | 1.9 | 0.83 | 2617 | 144.7 | 0.83 | 2617 | 315.4 | 0.83 | 3840 | 91.3 | 0.83 |
| Average | 3121.1 | 6958.7 | 5495 | 0.5 | 1.16 | 5495 | 1.1 | 1.16 | 5494.3 | 1.0 | 1.16 | 5495 | 89.2 | 1.16 | 5495 | 231.3 | 1.16 | 5495 | 48.1 | 1.16 |
| # times best | | | 4 | | | 4 | | | 0 | | | 4 | | | 4 | | | 4 | | |
| # times fastest | | | | 8 | | | 1 | | | 1 | | | 7 | | | 0 | | | 0 | |
| # times exact | | | 0 | | | 0 | | | 0 | | | 0 | | | 0 | | | 0 | | |

TABLE 7 *Results on the comparison of 1-optimum and 2-optimum local search methods. The objective function is* $f(x) = 2^{x_1} + 2^{x_2} + \cdots + 2^{x_n}$. *Input matrices have dimensions* $10 \times 50$ *and* $10 \times 100$

| | | | 1-optimum local search | | | | | | 2-optimum local search | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SDMI | | SDMW | | HDM | | SDMI | | SDMW | | HDM | |
| $(m, n)$ | Range | FV | Sol | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time |
| (10, 50) | (0, 100) | 268.0898 | 0.5625 | 0.6 | 28.0898 | 0.1 | 74.25 | 0.5 | 0.4922 | 25.1 | 8.0898 | 4.1 | 92.5 | 2.6 |
| (10, 50) | (0, 100) | 65678 | 0.5318 | 1.1 | 2.6270 | 0.3 | 9.6315 | 1.0 | 0.5318 | 23.1 | 0.8770 | 3.2 | 9.6334 | 3.1 |
| (10, 50) | (0, 100) | 37871 | 43.0166 | 0.5 | 366.8916 | 0.1 | 138.75 | 0.4 | 27.0156 | 21.3 | 174.8916 | 3.6 | 169.75 | 3.3 |
| (10, 50) | (0, 100) | 331.7051 | 1.4141 | 0.6 | 35.7051 | 0.1 | 73.5703 | 0.6 | 1.1328 | 26.2 | 17.7051 | 4.1 | 74.5752 | 2.5 |
| (10, 50) | (0, 100) | 152.5098 | 0.0225 | 0.7 | 1.5098 | 0.0 | 130.0327 | 0.5 | 0.0225 | 20.4 | 0.1660 | 3.4 | 130.0342 | 3.2 |
| (10, 50) | (0, 100) | 2030.2 | 0.9873 | 0.6 | 46.21 | 0.1 | 258.8613 | 0.5 | 0.8457 | 24.8 | 5.21 | 3.6 | 259.0723 | 3.2 |
| (10, 50) | (0, 100) | 87721 | 525.6270 | 0.6 | 1193 | 0.1 | 78376 | 0.5 | 525.6270 | 22.3 | 713.0195 | 4.3 | 78381 | 3.3 |
| (10, 50) | (0, 100) | 33687000 | 23.4141 | 0.6 | 258.6719 | 0.1 | 44.4141 | 0.5 | 23.4141 | 22.3 | 66.6719 | 3.9 | 44.6641 | 2.7 |
| (10, 50) | (0, 100) | 2238400 | 35.3125 | 0.5 | 5.0 | 0.1 | 1054500 | 0.5 | 34.25 | 21.8 | 963.3125 | 3.2 | 2100000 | 2.7 |
| (10, 50) | (0, 100) | 1153978 | 2658.5 | 0.6 | 11194 | 0.1 | 89066 | 0.5 | 2648.5 | 25.1 | 4538 | 4.1 | 93802 | 2.4 |
| (10, 100) | (0, 100) | 1928535 | 121 | 3.1 | 224599 | 0.1 | 87200 | 3.0 | 104 | 164.5 | 60759 | 11.0 | 209208 | 10.6 |
| (10, 100) | (0, 100) | 2959.3 | 0.332 | 3.2 | 143.2539 | 0.1 | 7.4688 | 3.1 | 0.3320 | 167.8 | 47.2539 | 10.5 | 14.1563 | 10.7 |
| (10, 100) | (0, 100) | 4256000 | 4.1250 | 2.8 | 6382.7 | 0.1 | 1250.8 | 2.9 | 4.1250 | 140.8 | 622.6875 | 9.9 | 1284.8 | 10.3 |
| (10, 100) | (0, 100) | 49524 | 3.6250 | 3.1 | 4467.6 | 0.1 | 10301 | 2.9 | 3.2500 | 165.7 | 755.6250 | 9.8 | 10465 | 10.1 |
| (10, 100) | (0, 100) | 4696935 | 57 | 3.0 | 27495 | 0.1 | 3900 | 3.3 | 49 | 167.1 | 9063 | 10.4 | 4198972 | 11.1 |
| (10, 100) | (0, 100) | 17829000 | 0.2168 | 3.2 | 114.9 | 0.3 | 4194800 | 3.4 | 0.1660 | 179.2 | 18.9648 | 11.0 | 16779000 | 10.9 |
| (10, 100) | (0, 100) | 79136 | 2.8750 | 3.0 | 2336.3 | 0.1 | 33001 | 3.3 | 2.8750 | 153.4 | 672.25 | 10.0 | 33516 | 10.2 |
| (10, 100) | (0, 100) | 4289500 | 0.6641 | 3.0 | 3076.8 | 0.1 | 201.5625 | 2.9 | 0.6250 | 161.4 | 516.8359 | 10.4 | 8262.6 | 10.5 |
| (10, 100) | (0, 100) | 1617600 | 3.5938 | 2.9 | 7822.6 | 0.1 | 3603.9 | 2.8 | 2.2500 | 166.7 | 1166.6 | 9.9 | 3603.9 | 11.9 |
| (10, 100) | (0, 100) | 6454021 | 31.7500 | 2.9 | 6917 | 0.1 | 2097819 | 3.0 | 21.7 | 156.0 | 1797 | 9.9 | 4195100 | 10.6 |

TABLE 8 *Results on the comparison of 1-optimum and 2-optimum local search methods. The objective function is* $f(x) = 2^{x_1} + 2^{x_2} + \cdots + 2^{x_n}$. *Input matrices have dimensions* $20 \times 50$ *and* $20 \times 100$

| | | | 1-optimum local search | | | | | | 2-optimum local search | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | SDMI | | SDMW | | HDM | | SDMI | | SDMW | | HDM | |
| $(m, n)$ | Range | FV | Sol | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time | Sol | Time |
| (20, 50) | (0, 100) | 485.4375 | 348.4375 | 2.6 | 365.4375 | 0.5 | 364.8125 | 2.2 | 212.75 | 182.4 | 293.4375 | 47.8 | 435.0625 | 16.2 |
| (20, 50) | (0, 100) | 45.4844 | 6.625 | 2.4 | 8.9844 | 0.5 | 38.0625 | 2.0 | 6.125 | 158.0 | 7.4844 | 51.9 | 6.9688 | 20.2 |
| (20, 50) | (0, 100) | 1054.1 | 7.4063 | 2.7 | 19.0703 | 0.4 | 14.9688 | 2.3 | 7.4063 | 174.9 | 14.0703 | 38.2 | 15.5313 | 20.9 |
| (20, 50) | (0, 100) | 163.6328 | 21.5078 | 2.3 | 27.6328 | 0.5 | 21.25 | 2.0 | 21.5078 | 141.5 | 23.6328 | 48.5 | 23.8125 | 16.7 |
| (20, 50) | (0, 100) | 46205 | 1998.3 | 2.7 | 2684.8 | 0.5 | 2.0 | 2.9 | 1998.3 | 182.7 | 2204.8 | 54.4 | 2662.3 | 18.0 |
| (20, 50) | (0, 100) | 836.5 | 227.25 | 2.3 | 388.5 | 0.5 | 502 | 2.5 | 217.25 | 164.2 | 276.5 | 50.5 | 550 | 17.2 |
| (20, 50) | (0, 100) | 12.1974 | 2.1467 | 2.8 | 2.5724 | 0.5 | 1.1062 | 3.0 | 1.7092 | 194.7 | 2.0099 | 53.4 | 2.1927 | 19.0 |
| (20, 50) | (0, 100) | 84.2407 | 2.7959 | 2.8 | 2.9907 | 0.5 | 3.4990 | 2.6 | 0.7949 | 202.7 | 0.8647 | 57.4 | 3.9759 | 15.4 |
| (20, 50) | (0, 100) | 3033.3 | 2617 | 2.7 | 2681.3 | 0.5 | 1514.9 | 2.2 | 2617 | 166.9 | 2641.3 | 55.9 | 2929.4 | 16.5 |
| (20, 50) | (0, 100) | 18920 | 17073 | 2.7 | 17640 | 0.5 | 10012 | 3.0 | 1.7 | 170.6 | 17320 | 51.2 | 17541 | 18.4 |
| (20, 100) | (0, 100) | 12432 | 62 | 11.7 | 784 | 0.7 | 271 | 12.5 | 53 | 940.4 | 368 | 66.7 | 344.5 | 58.2 |
| (20, 100) | (0, 100) | 3929 | 44.25 | 14.2 | 665 | 0.8 | 759 | 15.1 | 34.7 | 1106.5 | 377 | 84.7 | 699.5 | 59.1 |
| (20, 100) | (0, 100) | 5511.1 | 39.75 | 12.4 | 199.125 | 0.7 | 85.75 | 11.3 | 27.625 | 971.6 | 151.125 | 58.2 | 88.25 | 67.0 |
| (20, 100) | (0, 100) | 231.5313 | 7.6641 | 5.3 | 55.5313 | 0.3 | 25.1641 | 5.0 | 6.4141 | 425.8 | 37.5313 | 29.5 | 42.6641 | 29.5 |
| (20, 100) | (0, 100) | 4391 | 125 | 5.1 | 1831 | 0.3 | 641.75 | 5.4 | 109 | 416.8 | 807 | 28.1 | 1715.5 | 28.2 |
| (20, 100) | (0, 100) | 7050.4 | 59.625 | 4.9 | 778.375 | 0.3 | 4922.5 | 4.9 | 50.625 | 384.4 | 426.375 | 23.8 | 5895.5 | 25.8 |
| (20, 100) | (0, 100) | 415.5938 | 20.8438 | 5.3 | 127.5938 | 0.3 | 54.0625 | 5.1 | 20.1563 | 405.1 | 91.5938 | 27.5 | 64.0625 | 27.2 |
| (20, 100) | (0, 100) | 18.3359 | 1.1289 | 5.3 | 4.8359 | 0.3 | 10.0313 | 5.2 | 0.8984 | 414.7 | 3.3359 | 30.8 | 12.668 | 24.3 |
| (20, 100) | (0, 100) | 1290.7 | 55.9063 | 5.1 | 298.7188 | 0.3 | 247.125 | 4.9 | 20.1563 | 411.8 | 200.7188 | 28.7 | 247.875 | 28.4 |
| 1(20, 100) | (0, 100) | 2049.5 | 759 | 5.5 | 1217.5 | 0.3 | 810.375 | 5.0 | 684.25 | 419.0 | 985.5 | 27.3 | 1323.4 | 24.7 |

It can easily be verified from Tables 7 and 8 that 2-optimum SDMI is the best method that can produce a solution with a minimal objective function value. 1-optimum SDMW is the fastest method that can produce solution in a very short period of time.

Finally, we concluded that when the objective function is max-linear 1-optimum local search methods are more efficient. When the objective function is not max-linear (that is $f(x) = 2^{x_1}+2^{x_2}+\cdots+2^{x_n}$) 2-optimum local search methods are more appropriate.

**Funding**

Engineering and Physical Sciences Research Council (RRAH12809 to PB).

## References

AKIAN, M., GAUBERT, S. & KOLOKOLTSOV, V. (2005) Set coverings and invertibility of functionalGalois connections. *Idempotent Mathematics and Mathematical Physics* (G. L. Litvinov & V. P. Maslov eds). Contemporary Mathematics, vol. 377. Providence, RI: American Mathematical Society, pp. 19–51.

AMINU, A. (2009) Max-algebraic linear systems and programs. *Ph.D. Thesis*, University of Birmingham, Birmingham, UK.

BACCELLI, F. L., COHEN, G., OLSDER, G. J. & QUADRAT, J. P. (1992) *Synchronization and Linearity*. Chichester, NY: John Wiley.

BUTKOVIČ, P. (2003) Max-algebra: the linear algebra of combinatorics? *Linear Algebra Appl.*, **367**, 313–335.

BUTKOVIČ, P. (2010) *Max-linear Systems: Theory and Algorithms*. Springer Monographs in Mathematics. Berlin, Germany: Springer.

BUTKOVIČ, P. & AMINU, A. (2009) Introduction to max-linear programming. *IMA J. Manage. Math.*, **20**, 233–249.

BUTKOVIČ, P. & HEGEDUS, G. (1984) An elimination method for finding all solutions of the system of linear equationsover an extremal algebra. *Ekonom. Mat. Obzor.*, **20**, 203–215.

CUNINGHAME-GREEN, R. A. (1979) *Minimax Algebra*. Lecture Notes in Economics and Mathematical Systems, vol. 166. Berlin: Springer.

CUNINGHAME-GREEN, R. A. (1995) Minimax algebra and applications. *Advances in Imaging and Electron Physics*, vol. 90. New York: Academic Press, pp. 1–121.

CUNINGHAME-GREEN, R. A. & BUTKOVIČ, P. (2003) The equation Ax = By over (max,+). *Theoretical Comput. Sci.*, **293**, 3–12.

CUNINGHAME-GREEN, R. A. & ZIMMERMANN, K. (2001) Equation with residual functions. *Comment. Math. Univ. Carolin.*, **42**, 729–740.

HEIDERGOTT, B., OLSDER, G.-J. & VAN DER WOUDE, J. (2005) *Max Plus at Work: Modeling and Analysis of Synchronized Systems, A Course on Max-Plus Algebra*. Woodstock: Princeton University Press.

HOGBEN, L., BRUALDI, R., GREENBAUM, A. & MATHIAS, R. (eds) (2006) *Handbook of Linear Algebra*. Discrete Mathematics and Its Applications, vol. 39. Baton Rouge, LA: Chapman and Hall.

MICHALEWICZ, Z. & FOGEL, D. B. (2004) *How to Solve It: Modern Heuristics*. Berlin, Germany: Springer.

VOROBYOV, N. N. (1967) Extremal algebra of positive matrices. *Elektron. Informa- tionsverarbeit. Kybernetik*, **3**, 39–71 (in Russian).

WALKUP, E. A. & BORIELLO, G. (1988) A general linear max-plus solution technique. *Idempotency* (J. Gunawardena ed.). Cambridge: Cambridge University Press, pp. 406–415.

ZIMMERMANN, K. (1976) Extremální algebra. *Výzkumná publikace Ekonomicko-matem- atické laboratoře při Ekonomickém ústavě ČSAV*, **46**, Praha (in Czech).