AIMS Mathematics, 6(3): 2094–2113.
DOI:10.3934/math.2021128
Received: 12 October 2020
Accepted: 04 December 2020
Published: 09 December 2020

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

Research article

A polynomial local optimality condition for the concave piecewise linear network flow problem

Zhibin Nie[1], **Shuning Wang**[1] **and Xiaolin Huang**[2,*]

[1] Department of Automation, Tsinghua University, Beijing, 100084, P. R. China
[2] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, P. R. China

* **Correspondence:** Email: xiaolinhuang@sjtu.edu.cn.

Abstract: This paper studies the local optimality condition for the widely applied concave piecewise linear network flow problem (CPLNFP). Traditionally, for CPLNFP the complexity of checking the local optimality condition is exponentially related to the number of active arcs (i.e., arcs in which the flow is at the breakpoints). When the scale of CPLNFP is large, even local optimality is unverifiable and the corresponding local algorithms are inefficient. To overcome this shortcoming, a new local optimality condition is given. This new condition is based on the concavity and piecewise linearity of the cost function and makes full use of the network structure. It is proven that the complexity of the new condition is polynomial. Therefore, using the new condition to verify the local optimality is far superior to using the traditional condition, especially when there are many active arcs.

Keywords: nonlinear programming; concave piecewise linear; network flow problem; local optimality condition; polynomial complexity
Mathematics Subject Classification: 90B06, 90C26, 90C46

1. Introduction

The network flow problem (NFP) is a fundamental and important task in network optimization. NFP has been widely applied in transportation, communication, construction project [1], manufacturing [2], network design [3], water resource management [4], and other fields. NFP is to minimize a cost function under linear constraints. Thus, its complexity strongly depends on the category of the cost function. Using a linear function as the cost is the simplest case. But in practice, the cost function is often concave due to the marginal utility. As described in [5], the concave cost is or can be well approximated by a concave piecewise linear function. Moreover, NFP with general nonlinear cost functions can be transformed into NFP with concave cost functions on an expanded network [6], so it

is of great significance to investigate the concave piecewise linear network flow problem (CPLNFP).

Let $G = (N, A)$ be a directed network with n nodes and m arcs, where N and A are the sets of nodes and arcs, respectively. Each arc $(i, j) \in A$ is associated with a flow x_{ij}, a capacity u_{ij}, and a cost function $f_{ij}(x_{ij})$. For the i-th node $i \in N$, the supply or the demand flow is denoted by b_i. If $b_i > 0$, then node i is a supply node; if $b_i < 0$, then node i is a demand node; otherwise, node i is a transshipment node. Due to the balance of network flow, the total supply flow must be equal to the total demand flow, which means $\sum_{i\in N} b_i = 0$. With those notations, a CPLNFP defined on network G can be formulated as follows:

$$\begin{aligned} \min & \sum_{(i,j)\in A} f_{ij}(x_{ij}) \\ \text{s.t.} & \sum_{\{j|(i,j)\in A\}} x_{ij} - \sum_{\{j|(j,i)\in A\}} x_{ji} = b_i, \forall i \in N, \\ & 0 \le x_{ij} \le u_{ij}, \forall (i,j) \in A, \end{aligned} \tag{1.1}$$

where $f_{ij}(x_{ij})$ is a continuous concave piecewise linear function that can be expressed as

$$f_{ij}(x_{ij}) = \begin{cases} c_{ij}^1 x_{ij} + 0, & x_{ij} \in [0, \lambda_{ij}^1], \\ c_{ij}^2 x_{ij} + d_{ij}^2, & x_{ij} \in [\lambda_{ij}^1, \lambda_{ij}^2], \\ \cdots & \cdots \\ c_{ij}^s x_{ij} + d_{ij}^s, & x_{ij} \in [\lambda_{ij}^{s-1}, \lambda_{ij}^s], \\ \cdots & \cdots \\ c_{ij}^{s_{ij}} x_{ij} + d_{ij}^{s_{ij}}, & x_{ij} \in [\lambda_{ij}^{s_{ij}-1}, u_{ij}]. \end{cases} \tag{1.2}$$

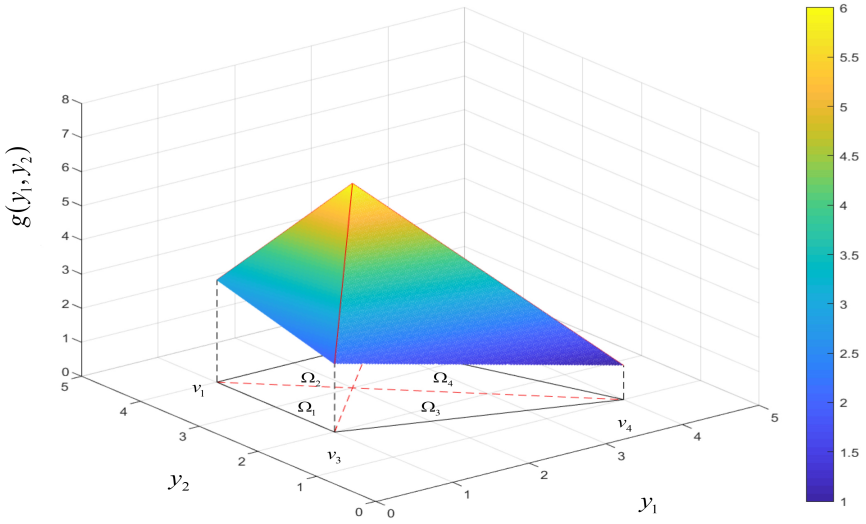
In (1.2), $f_{ij}(x_{ij})$ has s_{ij} segments, and the interval $[0, u_{ij}]$ is accordingly divided into s_{ij} subintervals with breakpoints $\lambda_{ij}^1, \lambda_{ij}^2, \cdots, \lambda_{ij}^{s_{ij}-1}$. To make notations consistent, we set $d_{ij}^1 = 0$, $\lambda_{ij}^0 = 0$ and $\lambda_{ij}^{s_{ij}} = u_{ij}$. Then, in each subinterval $[\lambda_{ij}^{s-1}, \lambda_{ij}^s]$, the objective function $f_{ij}(x_{ij})$ becomes a linear function with slope c_{ij}^s and intercept d_{ij}^s. Due to the continuity and concavity of $f_{ij}(x_{ij})$, we have $c_{ij}^s \lambda_{ij}^s + d_{ij}^s = c_{ij}^{s+1} \lambda_{ij}^s + d_{ij}^{s+1}$, $s = 1, 2, \cdots, s_{ij} - 1$, and $c_{ij}^1 > c_{ij}^2 > \cdots > c_{ij}^{s_{ij}}$.

CPLNFP has been widely applied in many areas, including facilities location [7] and inventory management [8]. Since CPLNFP is nonconvex, the global optimality cannot be guaranteed, and researchers focus on its local optimality, which also serves as the basis of many algorithms [9, 10]. The classical local algorithms include the dynamic slope-scaling procedure (DSSP) [11] and the dynamic cost updating procedure (DCUP) [12]. DSSP uses a linear function to approximate the concave cost function based on the iterative solution. DCUP relaxes the CPLNFP as a bilinear programming problem that can be transformed into a series of linear NFPs.

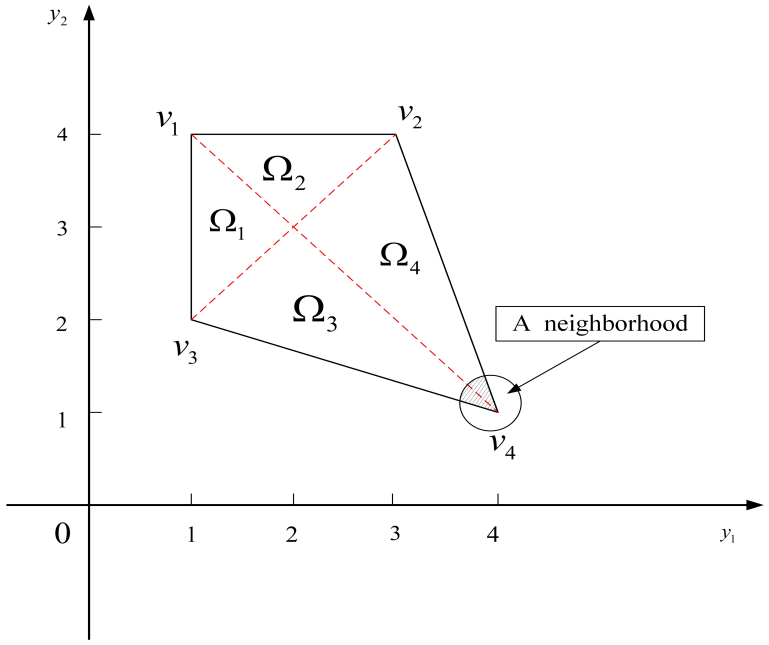
As a concave piecewise linear optimization over a polyhedron, the locally optimal solution of the CPLNFP can be achieved at some vertices of the feasible domain. Algebraically, the vertex refers to the basic feasible solution of the constraint system in (1.1). In many local algorithms, such as the DCUP and DSSP, the vertex is obtained by solving a linear NFP and then its local optimality is checked.

By definition, the feasible domain of a piecewise linear function can be divided into many regions in which the function stays linear. Each feasible point, including the vertex, belongs to one or more regions. Moreover, for each point, we can certainly find a neighbourhood whose intersection with the feasible domain is contained in the union of regions that this point belongs to. Figure 1 illustrates

these direct but important properties with a two-dimensional concave piecewise linear function. For example, it can be seen that the feasible domain can be divided into four regions and v_1 belongs to two regions: Ω_1 and Ω_2.



(a) The piecewise linear function $g(y_1, y_2)$.



(b) The feasible domain.

Figure 1. A piecewise linear function $g(y_1, y_2)$ defined on a two-dimensional polyhedron that can be divided into four regions.

From the above discussing, for concave piecewise linear minimization over a polyhedron, a vertex that is locally optimal is equivalent means that it is optimal in all the regions that it belongs to. This

local optimality condition is geometrically intuitive: If we can find a region in which this vertex is not optimal, there must be a feasible descent direction in this region, so this vertex is not locally optimal. For example, in Figure 1, v_4 is a locally optimal point if and only if v_4 is an optimal point in Ω_3 and Ω_4. For more detailed discussions about local optimality for piecewise linear programming, see [13], [14] and [15]. Following the idea of checking optimality for all regions, we come to the traditional local optimality condition for the CPLNFP.

A vertex v in the feasible domain is locally optimal for the CPLNFP if and only if v is optimal in all the regions that v belongs to.

Consider a vertex v in the feasible domain of the CPLNFP. Based on the characteristics of the cost function, the regions that v belongs to can be obtained by restricting the flow associated with arc (i, j) to a subinterval of $[0, u_{ij}]$ that v_{ij} belongs to. Let m_1 be the number of active arcs (i.e., arcs in which the flow is at the breakpoints), and then clearly the number of regions that v belongs to is 2^{m_1} since the flow in active arcs belongs to two subintervals at the same time. In each region, the concave piecewise linear cost function is a linear function, from which a linear NFP is defined with the constraints of the CPLNFP; whether v is optimal in this region is equivalent to whether v is optimal for this linear NFP. Therefore, to verify the local optimality, one needs to check the optimality of v for 2^{m_1} linear NFPs, which is obviously inefficient when the problem dimension, the number of segments of the cost function, and thus the number of active arcs increase. In [12], the number of active arcs (m_1) is assumed to be zero and based on this assumption, DCUP was designed to obtain a local optimum. However, this assumption exclude many real situations. The interested readers could find a numerical experiment in Appendix A, showing the estimated m_1 for different CPLNFPs. In Figure 2, we plot the results: m_1 is not always zero and becomes large and unacceptable as the problem dimension increases, from which it follows that using the traditional local optimality condition is inefficient.

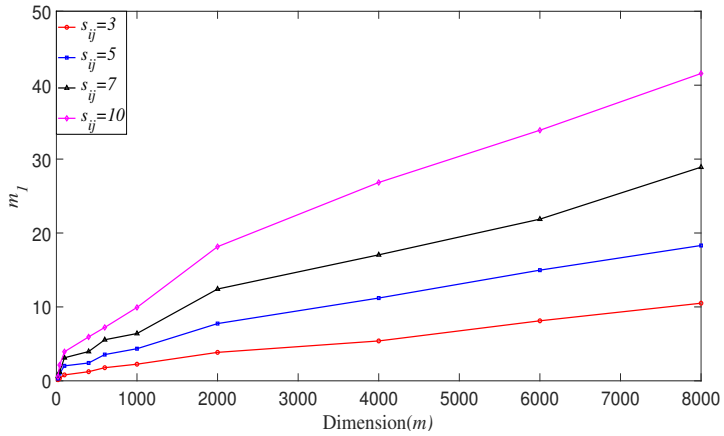


Figure 2. The number of linear NFPs that need to be verified is 2^{m_1}, and this figure plots the increasing trend of m_1 with respect to the problem dimension. For example, when $m = 8000, s_{ij} = 10, \forall (i, j) \in A$, m_1 is approximately equal to 42 and 2^{m_1} is more than 4×10^{12}, which is certainly unacceptable.

In this paper, we provide a new local optimality condition, which is based on the network structure and the concavity of $f_{ij}(x_{ij}), \forall (i, j) \in A$. The complexity of implementing the proposed condition is polynomial with the network parameters. After briefly introducing the background, the proposed

condition, the proof, and the strategy of implementation will be given in section 3.

2. Preliminaries

2.1. Basic concepts

To better represent the parameters associated with the arcs in A, we first define a tuple collection Θ. Each tuple in Θ has m elements, having one-to-one correspondence with the arcs in A. For any tuple $\gamma \in \Theta$, the element corresponding to arc $(i, j) \in A$ is a real number and denoted as γ_{ij}. In other words, the tuple γ can be expressed as

$$\gamma = (\gamma_{ij}, \forall (i, j) \in A).$$

Based on Θ, we can easily represent the parameters associated with the arcs in A. For example, we use $x \in \Theta$ to denote flows in all the arcs in A, and express the feasible domain of the CPLNFP as below,

$$\Omega = \left\{ x \in \Theta \left| \begin{array}{c} \sum\limits_{\{j|(i,j)\in A\}} x_{ij} - \sum\limits_{\{j|(j,i)\in A\}} x_{ji} = b_i, \forall i \in N \\ 0 \le x_{ij} \le u_{ij}, \forall (i, j) \in A \end{array} \right. \right\}.$$

By this way, the CPLNFP can be succinctly written as

$$\begin{aligned} \min \ \ & f(x) = \sum_{(i,j)\in A} f_{ij}(x_{ij}) \\ s.t. \ \ & x \in \Omega. \end{aligned} \tag{2.1}$$

As discussed, the feasible domain can be divided into a number of regions obtained by restricting $x_{ij}, \forall (i, j) \in A$ to one of the subintervals in $[0, u_{ij}]$. Evidently, the cost function stays linear in each region. Before introducing the new local optimality condition, we give the following definitions to describe the related regions and linear NFPs for a feasible point $\hat{x} \in \Omega$.

Definition 1. *For any $\hat{x} \in \Omega$ and $(i, j) \in A$, we define $\Psi_{ij}(\hat{x}_{ij})$ as follows:*

$$\begin{aligned} &\Psi_{ij}(\hat{x}_{ij}) = \{1\}, \textit{if } \hat{x}_{ij} = 0, \\ &\Psi_{ij}(\hat{x}_{ij}) = \{s\}, \textit{if } \lambda_{ij}^{s-1} < \hat{x}_{ij} < \lambda_{ij}^{s}, \\ &\Psi_{ij}(\hat{x}_{ij}) = \{s, s+1\}, \textit{if } \hat{x}_{ij} = \lambda_{ij}^{s}, s \neq s_{ij}, \\ &\Psi_{ij}(\hat{x}_{ij}) = \{s_{ij}\}, \textit{if } \hat{x}_{ij} = u_{ij}. \end{aligned}$$

$\Psi_{ij}(\hat{x}_{ij})$ denotes the index(es) of the subinterval(s) that $\hat{x}_{ij}$ belongs to. If $\hat{x}_{ij}$ is at the breakpoints, then $\hat{x}_{ij}$ belongs to two subintervals, and $\Psi_{ij}(\hat{x}_{ij})$ has two elements. For example, if $\hat{x}_{ij} = \lambda_{ij}^{2}$, then $\hat{x}_{ij}$ belongs to both the second and the third subinterval, and thus $\Psi_{ij}(\hat{x}_{ij}) = \{2, 3\}$.

Definition 2. *For any $\hat{x} \in \Omega$, the arc set A can be partitioned into two subsets as follows:*

$$\begin{cases} (i, j) \in A_1, & \textit{if } (i, j) \in A, \hat{x}_{ij} \in B_{ij}, \\ (i, j) \in A_2, & \textit{if } (i, j) \in A, \hat{x}_{ij} \notin B_{ij}, \end{cases} \tag{2.2}$$

where $B_{ij} = \{\lambda_{ij}^{1}, \lambda_{ij}^{2}, \cdots, \lambda_{ij}^{s_{ij}-1}\}$ consists of all breakpoints in $[0, u_{ij}]$. For any $(i, j) \in A_1$, $\Psi_{ij}(\hat{x}_{ij})$ has two elements. We call arcs in A_1 active arcs and use m_1 to represent the number of active arcs.

As discussed, we can randomly select an element from $\Psi_{ij}(\hat{x}_{ij}), \forall(i,j) \in A$ and restrict the flow x_{ij} to the corresponding subinterval to obtain a region that $\hat{x}$ belongs to. Considering all possible selections, a set collection $\Phi(\hat{x})$ can be used to indirectly index all the regions that $\hat{x}$ belongs to as follows.

$$\Phi(\hat{x}) = \{z \in \Theta | z_{ij} \in \Psi_{ij}(\hat{x}_{ij}), \forall(i,j) \in A\}.$$

In above definition, "$z \in \Theta$" is just used to show that any tuple $\varphi \in \Phi(\hat{x})$ has m elements that have a one-to-one correspondence with the arcs in A. In fact, φ_{ij} is an integer in $[1, s_{ij}]$ since $\varphi_{ij} \in \Psi_{ij}(\hat{x}_{ij})$. The region associated with φ is determined by selecting the φ_{ij}-th subinterval for any arc $(i,j) \in A$ and can be denoted as

$$\Omega_\varphi = \{x \in \Omega | \lambda_{ij}^{\varphi_{ij}-1} \leq x_{ij} \leq \lambda_{ij}^{\varphi_{ij}}, \forall(i,j) \in A\}.$$

Clearly, the number of regions that $\hat{x}$ belongs to is 2^{m_1} because $\Psi_{ij}(\hat{x}_{ij})$ has two elements for any $(i,j) \in A_1$. Since the feasible domain Ω is constrained by equality constraints, Ω_φ might have only one point $\hat{x}$. In this case, when checking the local optimality, we do not need to check the optimality of $\hat{x}$ in this region because $\hat{x}$ is the only point in this region and is naturally optimal. However, in practice, we do not know in advance which region has only one point, so we have to treat all the regions equally and check the optimality of $\hat{x}$ in this region. When Ω_φ has only one point, we can easily know that $\hat{x}$ is optimal in this region, and the process of checking the optimality of $\hat{x}$ is equal to verifying that Ω_φ has a unique point. Therefore, we only need to check the optimality of $\hat{x}$ in all the regions that $\hat{x}$ belongs to, and consequently, the region with only one point becomes irrelevant.

In Ω_φ, the cost function is linear and can be expressed as

$$f^\varphi(x) = \sum_{(i,j)\in A} (c_{ij}^{\varphi_{ij}} x_{ij} + d_{ij}^{\varphi_{ij}}).$$

Due to the concavity of $f(x)$, we can derive the following property that will be used in the proof of Theorem 1.

Property 1. *For any $\hat{x} \in \Omega$, we have $f(\hat{x}) \leq f^\varphi(\hat{x})$.*

Proof. This property can be obtained directly: $f(\hat{x})$ can be expressed as $f(\hat{x}) = \sum_{(i,j)\in A} \min\{c_{ij}^1 \hat{x}_{ij} + d_{ij}^1, c_{ij}^2 \hat{x}_{ij} + d_{ij}^2, \cdots, c_{ij}^{s_{ij}} \hat{x}_{ij} + d_{ij}^{s_{ij}}\}$. □

Definition 3. *The linear NFP associated with φ is defined as*

$$\begin{aligned} P(\varphi): \quad & \min \ f^\varphi(x) \\ & s.t. \quad x \in \Omega. \end{aligned} \tag{2.3}$$

Similarly, the linear NFPs associated with $\hat{x} \in \Omega$ are defined as $P(\varphi), \forall \varphi \in \Phi(\hat{x})$. The number of linear NFPs associated with $\hat{x}$ is 2^{m_1}.

Now we use a CPLNFP instance to illustrate the above concepts, including $\Psi_{ij}(\hat{x}_{ij})$, $\Phi(\hat{x})$, φ and $P(\varphi)$.

Example 1. *Consider a CPLNFP defined on a network with* 6 *nodes and* 8 *arcs, as shown in Figure 3. Following the formulation (1.1), this CPLNFP is formulated as*

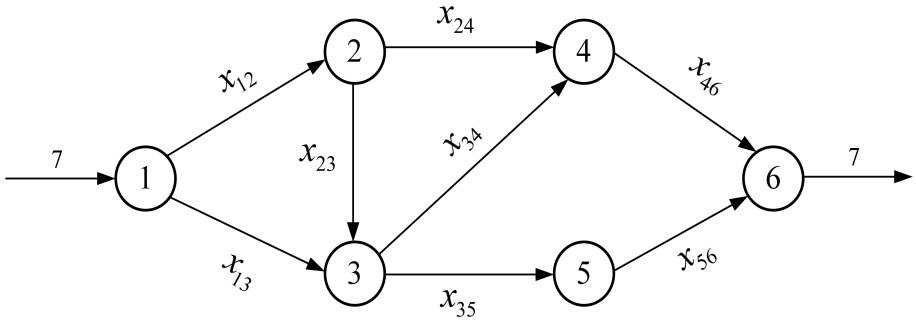


Figure 3. A practical CPLNFP.

$$\begin{aligned}
\min\ & f_{12}(x_{12}) + f_{13}(x_{13}) + f_{23}(x_{23}) + f_{24}(x_{24}) + f_{34}(x_{34}) \\
& + f_{35}(x_{35}) + f_{46}(x_{46}) + f_{56}(x_{56}) \\
\text{s.t. } & x_{12} + x_{13} = 7; x_{24} + x_{23} - x_{12} = 0 \\
& x_{34} + x_{35} - x_{13} - x_{23} = 0; x_{46} - x_{23} - x_{34} = 0 \\
& x_{56} - x_{35} = 0; -x_{46} - x_{56} = -7 \\
& 0 \le x_{12} \le 6; 0 \le x_{13} \le 4; 0 \le x_{23} \le 3 \\
& 0 \le x_{24} \le 4; 0 \le x_{34} \le 5; 0 \le x_{35} \le 8 \\
& 0 \le x_{46} \le 2; 0 \le x_{56} \le 7
\end{aligned}$$

where

$$f_{12}(x_{12}) = \begin{cases} 5x_{12}, & x_{12} \in [0,3] \\ 3x_{12} + 6, & x_{12} \in [3,6] \end{cases}; \quad f_{13}(x_{13}) = \begin{cases} 6x_{13}, & x_{13} \in [0,2] \\ 4x_{13} + 4, & x_{13} \in [2,4] \end{cases};$$

$$f_{23}(x_{23}) = \begin{cases} 3x_{23}, & x_{23} \in [0,1] \\ 2x_{23} + 1, & x_{23} \in [1,3] \end{cases}; \quad f_{24}(x_{24}) = \begin{cases} 6x_{24}, & x_{24} \in [0,2] \\ 3x_{24} + 6, & x_{24} \in [2,4] \end{cases};$$

$$f_{34}(x_{34}) = \begin{cases} 4x_{34}, & x_{34} \in [0,3] \\ 3x_{34} + 3, & x_{34} \in [3,5] \end{cases}; \quad f_{35}(x_{35}) = \begin{cases} 4x_{35}, & x_{35} \in [0,5] \\ 2x_{35} + 10, & x_{35} \in [5,8] \end{cases};$$

$$f_{46}(x_{46}) = \begin{cases} 6x_{46}, & x_{46} \in [0,1] \\ 3x_{46} + 3, & x_{46} \in [1,2] \end{cases}; \quad f_{56}(x_{56}) = \begin{cases} 6x_{56}, & x_{56} \in [0,5] \\ 4x_{56} + 10, & x_{56} \in [5,7] \end{cases}.$$

The corresponding tuple collection Θ *can be expressed as*

$$\Theta = \{(\gamma_{12}, \gamma_{13}, \gamma_{23}, \gamma_{24}, \gamma_{34}, \gamma_{35}, \gamma_{46}, \gamma_{56})\},$$

and the feasible domain can be expressed as $\Omega = \{x \in \Theta | x_{12} + x_{13} = 6, x_{24} + x_{23} - x_{12} = 0, x_{34} + x_{35} - x_{13} - x_{23} = 0, x_{46} - x_{23} - x_{34} = 0, x_{56} - x_{35} = 0, -x_{46} + x_{56} = -6, 0 \le x_{12} \le 6, 0 \le x_{13} \le 4, 0 \le x_{23} \le 3, 0 \le x_{24} \le 4, 0 \le x_{34} \le 5, 0 \le x_{35} \le 8, 0 \le x_{46} \le 2, 0 \le x_{56} \le 7\}$.

For a feasible solution $\hat{x} = (3, 4, 1, 2, 0, 5, 2, 5)$*, we know* $\Psi_{12}(\hat{x}_{12}) = \{1, 2\}$*,* $\Psi_{13}(\hat{x}_{13}) = \{2\}$*,* $\Psi_{23}(\hat{x}_{23}) = \{1, 2\}$*,* $\Psi_{24}(\hat{x}_{24}) = \{1, 2\}$*,* $\Psi_{34}(\hat{x}_{34}) = \{1\}$*,* $\Psi_{35}(\hat{x}_{35}) = \{1, 2\}$*,* $\Psi_{46}(\hat{x}_{46}) = \{2\}$*,* $\Psi_{56}(\hat{x}_{56}) = \{1, 2\}$*. Obviously, there are* 5 *active arcs:* $(1, 2)$*,* $(2, 3)$*,* $(2, 4)$*,* $(3, 5)$ *and* $(5, 6)$*. By definition, we have*

$\Phi(\hat{x}) = \{(z_{12}, z_{13}, z_{23}, z_{24}, z_{34}, z_{35}, z_{46}, z_{56}) | z_{12} \in \{1,2\}, z_{13} \in \{2\}, z_{23} \in \{1,2\}, z_{24} \in \{1,2\}, z_{34} \in \{1\}, z_{35} \in \{1,2\}, z_{46} \in \{2\}, z_{56} \in \{1,2\}\}$ *, which consists of* $2^5 = 32$ *tuples. For* $\varphi = (1,2,1,2,1,2,2,2) \in \Phi(\hat{x})$*, the linear NFP associated with* φ *can be expressed as*

$$\begin{aligned} P(\varphi): \min\ & 5x_{12} + 4x_{13} + 4 + 3x_{23} + 3x_{24} + 6 + 4x_{34} + 2x_{35} + 10 + 3x_{46} + 3 + 4x_{56} + 10 \\ \text{s.t.}\ & x \in \Omega \end{aligned}$$

Let V represent the set of vertices of Ω, or in other words, the set of the basic feasible solutions of the constraint system in (1.1). Based on Definition 3, we can derive the following local optimality condition.

Theorem 1. *A vertex* $v \in V$ *is locally optimal for the CPLNFP if and only if* v *is optimal for all the linear NFPs associated with it.*

Proof. $\Rightarrow$ If there is a $P(\varphi), \varphi \in \Phi(v)$ for which v is not optimal, then there must be a feasible descent direction $d \in \Theta$ satisfying $v + \varepsilon d \in \Omega$ and $f^{\varphi}(v + \varepsilon d) < f^{\varphi}(v)$ for a sufficiently small positive number ε. Since $f^{\varphi}(v) = f(v)$, we will have $f(v + \varepsilon d) \leq f^{\varphi}(v + \varepsilon d) < f(v)$, which contradicts that v is locally optimal for the CPLNFP.

$\Leftarrow$ For a sufficiently small $\delta \in R^m$, if $v + \delta \in \Omega$, then $v + \delta$ must belong to the union of regions that v belongs to. Therefore, there must be a region $\Omega_{\varphi'}, \varphi' \in \Phi(v)$ containing $v + \delta$, from which it follows that $f(v + \delta) = f^{\varphi'}(v + \delta)$. Since v is optimal for all the linear NFPs associated with it, we have $f(v) = f^{\varphi'}(v) \leq f^{\varphi'}(v + \delta) = f(v + \delta)$, which means v is locally optimal for the CPLNFP. □

In the worst case, checking the local optimality of v needs to consider the optimality of v for 2^{m_1} linear NFPs, when one relies on Theorem 1. It is noted that the feasible domain of $P(\varphi)$ is the same as that of the CPLNFP, not Ω_{φ}. Thus, one does not need to worry about the change of the feasible domain, but only the change of the objective function when checking the optimality of v for these linear NFPs.

2.2. Network simplex algorithm

The network simplex algorithm (NSA) (for detail see [16], [17] and[18]) is a classical and efficient algorithm to solve the linear NFP. When solving $P(\varphi)$ by the NSA, the vertex $v \in V$ as the basic feasible solution obtained in each iteration is associated with the following parameters:

- **Basis:** Mathematically, the basis is a triple (T, L, U), which is a partitioning of the arc set A as follows:

$$\begin{cases} \text{The arcs in } T \text{ constitute a spanning tree of network } G; \\ (i,j) \in L, \text{ if } (i,j) \notin T, v_{ij} = 0; \\ (i,j) \in U, \text{ if } (i,j) \notin T, v_{ij} = u_{ij}. \end{cases} \tag{2.4}$$

  The iteration of the NSA is essentially the iteration of the basis. Therefore, the basis is the core of iteration and others parameters are related to the vertex through the basis.
  For ease of expression, we set $\overline{T} = L \cup U$. An arc in T is called a basis arc and an arc in $\overline{T}$ is called a nonbasis arc. By definition, we have $A_1 \cap \overline{T} = \phi$. Similar to the partitioning of A, T can also be partitioned into T_1 and T_2 as follows:

$$\begin{cases} (i,j) \in T_1, \text{ if } (i,j) \in T, (i,j) \in A_1; \\ (i,j) \in T_2, \text{ if } (i,j) \in T, (i,j) \notin A_1. \end{cases} \tag{2.5}$$

By definition, we have $T_1 = A_1$.

- **Price:** The price p_i for node i is defined as

$$p_i = p_r - \sum_{(h,t)\in Path(i)_T^+} c_{ht}^{\varphi_{ht}} + \sum_{(h,t)\in Path(i)_T^-} c_{ht}^{\varphi_{ht}}, \tag{2.6}$$

where p_r is the price for the root node of the spanning tree, and $Path(i)_T^+$ and $Path(i)_T^-$ are the arc sets consisting of the forward arcs and backward arcs in the path of the spanning tree from the root node to node i, respectively. Generally, the root node of the spanning tree and its price p_r are specified in advance and will not change during iteration. In this paper, p_r is set as zero.

- **Reduced cost:** The reduced cost r_{ij} for arc $(i, j) \in A$ is defined as

$$r_{ij} = c_{ij}^{\varphi_{ij}} + p_j - p_i. \tag{2.7}$$

Note that, for any $(i, j) \in T$, the reduced cost r_{ij} is always zero by this definition.

- **Optimality condition:** A vertex $v \in V$ is optimal for $P(\varphi)$ if the reduced costs associated with v satisfy the following condition:

$$\begin{cases} r_{ij} \geq 0, \forall (i, j) \in L, \\ r_{ij} \leq 0, \forall (i, j) \in U. \end{cases} \tag{2.8}$$

3. The polynomial local optimality condition

In this section, we first derive the detailed local optimality condition by using the NSA to determine the optimality of the vertex for linear NFPs and then prove that the derived condition can be implemented with polynomial complexity in network parameters.

Let v^φ be the optimal vertex to $P(\varphi)$. For vertex v, we first randomly select a $\varphi \in \Phi(v)$ and then verify the optimality of v for $P(\varphi)$. If v is not optimal for $P(\varphi)$, we can immediately determine that v is not locally optimal for the CPLNFP. Therefore, we only need to focus on the case that $v = v^\varphi$ and propose a local optimality condition for v^φ.

3.1. Local optimality condition for v^φ

Traditionally, we need to verify the optimality of v^φ for the remaining $2^{m_1} - 1$ linear NFPs associated with it to determine the local optimality. The optimality of v^φ for $P(\varphi')$ can be verified through solving $P(\varphi')$ by the NSA. In this paper, it is assumed that v^φ is nondegenerate and is thus associated with only one spanning tree. Here, the statement that v^φ is nondegenerate means that the basic feasible solution corresponding to v^φ is nondegenerate. We say that v^φ is nondegenerate from a geometrically intuitive perspective. Similarly to the discussion on simplex algorithm, dealing with degeneracy is skilful and requires complicated representations. In this paper, we focus on nondegenerate case and leave the discussion on degeneracy for further study. When solving the remaining linear NFPs associated with v^φ, we can use v^φ as the initial solution, and thus the parameters associated with v^φ can be utilized. For example, the basis associated with v^φ will not change when solving the linear NFPs associated with v^φ.

For the sake of distinction, in the following discussion, we use r_{ij}^{φ} and $r_{ij}^{\varphi'}$ to represent the reduced cost associated with v^φ in solving $P(\varphi)$ and $P(\varphi')$, respectively. Based on the optimality condition (2.8) for the linear NFP, the local optimality condition for v^φ can be expressed as follows.

Theorem 2. *The vertex v^{φ} is locally optimal for CPLNFP if and only if $r_{ij}^{\varphi'}, \forall (i,j) \in \overline{T}$ satisfy condition (2.8) for any $\varphi' \in \Phi(v^{\varphi})$.*

Traditionally, for each of the linear NFPs associated with v^{φ}, we need to calculate the reduced costs to determine the optimality of v^{φ}. In other words, we have to calculate the reduced cost for each nonbasis arc 2^{m_1} times.

In this paper, we consider the optimality of v^{φ} for the linear NFPs associated with it from the point of view of each nonbasis arc rather than each linear NFP. According to Theorem 2, for each $(i,j) \in \overline{T}$, we actually only need to check whether the extremum of the reduced cost when solving the linear NFPs associated with v^{φ} satisfies condition (2.8). Here, the extremum refers to the minimum for $(i,j) \in L$ and the maximum for $(i,j) \in U$. If the extremum of the reduced cost for arc (i,j) satisfies (2.8), we can conclude that the reduced cost for arc (i,j) always satisfies (2.8) when solving all the linear NFPs associated with v^{φ}. Based on this idea, we actually only need to calculate the extremum of the reduced cost for each nonbasis arc, which means we only need to calculate the reduced cost for each nonbasis arc once. To better represent the extremum, we define $\psi(ij)$ as follows.

$$\psi(ij) = \begin{cases} \arg\min\limits_{\varphi'} \{r_{ij}^{\varphi'} | \varphi' \in \Phi(v^{\varphi})\}, \text{if } (i,j) \in L; \\ \arg\max\limits_{\varphi'} \{r_{ij}^{\varphi'} | \varphi' \in \Phi(v^{\varphi})\}, \text{if } (i,j) \in U. \end{cases} \tag{3.1}$$

With this definition, $r_{ij}^{\psi(ij)}$ represents the extremum of the reduced cost for arc (i,j) when solving the linear NFPs associated with v^{φ} and $P(\psi(ij))$ is the linear NFP that makes the reduced cost reach the extremum. Notice that although $\psi(ij)$ is defined on an exponential-sized set, but we do not need actually calculate it in that way. In the next subsection, we will provide a strategy to obtain $\psi(ij)$ and $r_{ij}^{\psi(ij)}$ in complexity $O(n)$.

Theorem 3. *The local optimality condition shown in Theorem 2 can be checked with polynomial complexity.*

Proof. Since $r_{ij}^{\psi(ij)}$ can be obtained with complexity $O(n)$ for any nonbasis arc and the number of nonbasis arcs is $m - n$, the total complexity for checking the condition shown in Theorem 2 is $O(n(m-n))$, which is polynomial in the network parameters. □

3.2. A simple strategy to obtain the extremum

One key procedure of checking local optimality is to obtain $\psi(ij)$ and $r_{ij}^{\psi(ij)}$ for any nonbasis arc (i,j) with complexity $O(n)$, which is represented as the following theorem.

Theorem 4. *For vertex v^{φ}, $\psi(ij)$ and $r_{ij}^{\psi(ij)}$ can be obtained with complexity $O(n)$ for any $(i,j) \in \overline{T}$.*

Proof. This is a constructive proof, in which a strategy to obtain $\psi(ij)$ for any nonbasis arc (i,j) with complexity $O(n)$ is given. This strategy is based on the difference in the reduced cost between solving $P(\varphi)$ and the remaining linear NFPs associated with v^{φ}.

For any of the remaining linear NFPs: $P(\varphi')$ ($\varphi' \in \Phi(v^{\varphi})$,$\varphi' \neq \varphi$), we need to calculate $r_{ij}^{\varphi'}$ to check the optimality of v^{φ}. Since r_{ij}^{φ} has been obtained when solving $P(\varphi)$, we only need to consider the difference between $r_{ij}^{\varphi'}$ and r_{ij}^{φ}. Moreover, for any nobasis arc (i,j), v_{ij}^{φ} equals 0 or u_{ij}, which means that

v_{ij}^{φ} belongs to only one subinterval (i.e., the first subinterval or the last subinterval), and thus we have $\varphi'_{ij} = \varphi_{ij}$ and $c_{ij}^{\varphi'_{ij}} = c_{ij}^{\varphi_{ij}}$. Therefore, based on the definition of the reduced cost, the difference between $r_{ij}^{\varphi'}$ and r_{ij}^{φ} can be expressed as

$$\underset{\varphi\to\varphi'}{\Delta} r_{ij} = \underset{\varphi\to\varphi'}{\Delta} p_j - \underset{\varphi\to\varphi'}{\Delta} p_i, \forall (i,j) \in \overline{T}, \tag{3.2}$$

where $\underset{\varphi\to\varphi'}{\Delta} p_i$ and $\underset{\varphi\to\varphi'}{\Delta} p_j$ are the increments of p_i and p_j when the linear NFP changes from $P(\varphi)$ to $P(\varphi')$, respectively.

By definition, the price is related to the arcs in the spanning tree. So we only need to consider the effect of basis arcs on $\underset{\varphi\to\varphi'}{\Delta} p_i$ and $\underset{\varphi\to\varphi'}{\Delta} p_j$. For ease of expression, we use $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij}$, $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_i$ and $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_j$ to represent the effects of arc $(h,t) \in T$ on $\underset{\varphi\to\varphi'}{\Delta} r_{ij}$, $\underset{\varphi\to\varphi'}{\Delta} p_i$ and $\underset{\varphi\to\varphi'}{\Delta} p_j$, respectively. Then, we have

$$\underset{\varphi\to\varphi'}{\Delta} r_{ij} = \sum_{(h,t)\in T} \underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij} = \sum_{(h,t)\in T} \left(\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_j - \underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_i \right) \tag{3.3}$$

Moreover, for any basis arc (h,t), if $(h,t) \notin T_1$, we have $\varphi_{ht} = \varphi'_{ht}$ according to the definition of T_1, which means arc (h,t) has no effect on $\underset{\varphi\to\varphi'}{\Delta} p_i$ and $\underset{\varphi\to\varphi'}{\Delta} p_j$. Therefore, when calculating $\underset{\varphi\to\varphi'}{\Delta} r_{ij}$, we need to focus on the arcs in T_1.

Definition 4. *For any $(i,j) \in \overline{T}$, the unique cycle formed by arc (i,j) and the spanning tree is called the derived cycle and is denoted as C_{ij}^T.*

By this definition, C_{ij}^T can be expressed as

$$C_{ij}^T = (i,j) \cup Path(i)_T \cup Path(j)_T.$$

where $Path(i)_T$ and $Path(j)_T$ are the arc sets consisting of the arcs in the path of the spanning tree from the root node to node i and j, respectively.

Lemma 1. *For any $(h,t) \in T_1$ and $\forall (i,j) \in \overline{T}$, if $(h,t) \notin C_{ij}^T$, then arc (h,t) has no effect on $\underset{\varphi\to\varphi'}{\Delta} r_{ij}$.*

Proof. If $(h,t) \notin C_{ij}^T$, then there are three cases for any $(h,t) \in T_1$.

- **Case 1:** $(h,t) \in Path(i)_T^+$, $(h,t) \in Path(j)_T^+$;
- **Case 2:** $(h,t) \in Path(i)_T^-$, $(h,t) \in Path(j)_T^-$;
- **Case 3:** $(h,t) \notin Path(i)_T^+ \cup Path(i)_T^-$, $(h,t) \notin Path(j)_T^+ \cup Path(j)_T^-$.

Clearly, without considering other arcs, we have $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_i = \underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_j$ for all three cases based on the definition of price in (2.6). Therefore, we have $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij} = 0$ which means that arc (h,t) has no effect on $\underset{\varphi\to\varphi'}{\Delta} r_{ij}$. □

Therefore, only $(h,t) \in T_1 \cap C_{ij}^T$ contributes to $\underset{\varphi\to\varphi'}{\Delta} r_{ij}$. Let $G_{ij}^T = T_1 \cap C_{ij}^T$; we have

$$\underset{\varphi\to\varphi'}{\Delta} r_{ij} = \sum_{(h,t)\in G_{ij}^T} \underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij} = \sum_{(h,t)\in G_{ij}^T} (\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_j - \underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_i) \tag{3.4}$$

According to the location and direction, the arcs in G_{ij}^T can be divided into the following four subsets:

- $G_{1\cdot ij}^T = \{(h,t) \in G_{ij}^T \mid (h,t)$ belongs to the path of the spanning tree from the root node to node j, and it has the reverse direction as (i,j) in the cycle $C_{ij}^T\}$.
- $G_{2\cdot ij}^T = \{(h,t) \in G_{ij}^T \mid (h,t)$ belongs to the path of the spanning tree from the root node to node j, and it has the same direction as (i,j) in the cycle $C_{ij}^T\}$.
- $G_{3\cdot ij}^T = \{(h,t) \in G_{ij}^T \mid (h,t)$ belongs to the path of the spanning tree from the root node to node i, and it has the same direction as (i,j) in the cycle $C_{ij}^T\}$.
- $G_{4\cdot ij}^T = \{(h,t) \in G_{ij}^T \mid (h,t)$ belongs to the path of the spanning tree from the root node to node i, and it has the reverse direction as (i,j) in the cycle $C_{ij}^T\}$.

Now, for any arc (h,t) in these four subsets, we use an example to illustrate the effect of arc (h,t) on $\underset{\varphi\to\varphi'}{\Delta} r_{ij}$.

Example 2. *Examples of these four subsets are shown in Figure 4. In the derived cycle C_{ij}^T, it is assumed that $(e,f),(g,f),(b,c),(d,c) \in T_1$; then, we have $(e,f) \in G_{1\cdot ij}^T$, $(g,f) \in G_{2\cdot ij}^T$, $(b,c) \in G_{3\cdot ij}^T$, and $(d,c) \in G_{4\cdot ij}^T$.*

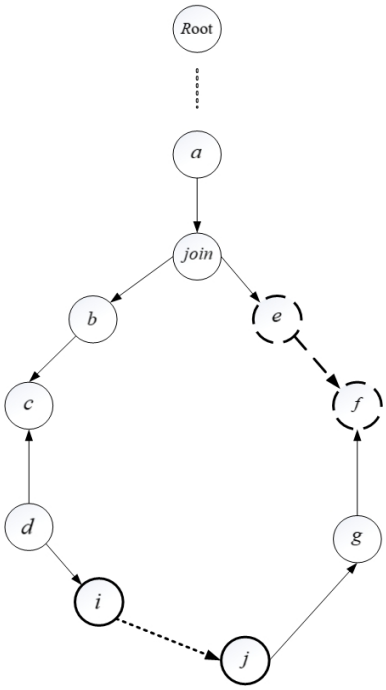
For any $(h,t) \in G_{ij}^T$, if $v_{ht}^{\varphi_{ht}} = \lambda_{ht}^k$, then $\varphi'_{ht} \in \{k, k+1\}$. Regardless of the influence of other arcs, $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij}$ can be calculated as follows.

(1) Case 1: As shown in Figure 4(a), $(h,t) = (e,f) \in G_{1\cdot ij}^T$. Based on (2.6), we have $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_i = 0$ and $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_j = c_{ht}^{\varphi_{ij}} - c_{ht}^{\varphi'_{ij}}$. Therefore, according to (3.2), $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij}$ can be calculated by $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij} = c_{ht}^{\varphi_{ij}} - c_{ht}^{\varphi'_{ij}}$.

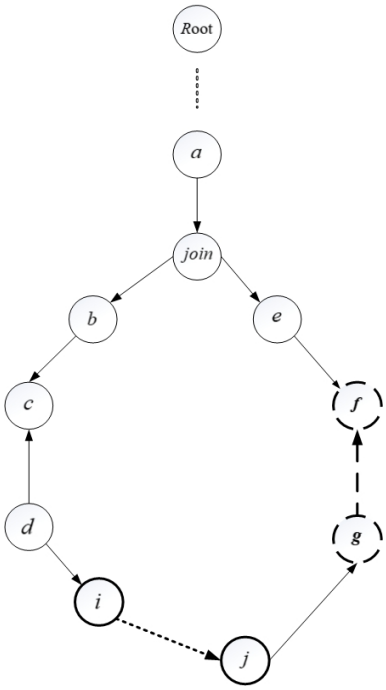
(2) Case 2: As shown in Figure 4(b), $(h,t) = (g,f) \in G_{2\cdot ij}^T$. Based on (2.6), we have $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_i = 0$ and $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_j = c_{ht}^{\varphi'_{ij}} - c_{ht}^{\varphi_{ij}}$. Therefore, according to (3.2), $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij}$ can be calculated by $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij} = c_{ht}^{\varphi'_{ij}} - c_{ht}^{\varphi_{ij}}$.

(3) Case 3: As shown in Figure 4(c), $(h,t) = (b,c) \in G_{3\cdot ij}^T$. Based on (2.6), we have $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_i = c_{ht}^{\varphi_{ij}} - c_{ht}^{\varphi'_{ij}}$ and $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_j = 0$. Therefore, according to (3.2), $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij}$ can be calculated by $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij} = c_{ht}^{\varphi'_{ij}} - c_{ht}^{\varphi_{ij}}$.

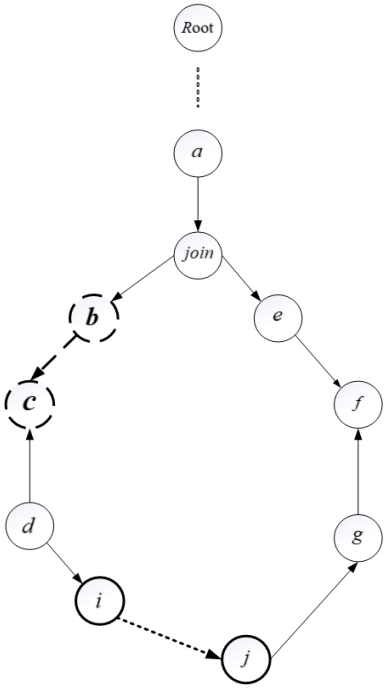
(4) Case 4: As shown in Figure 4(d), $(h,t) = (d,c) \in G_{4\cdot ij}^T$. Based on (2.6), we have $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_i = c_{ht}^{\varphi'_{ij}} - c_{ht}^{\varphi_{ij}}$ and $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} p_j = 0$. Therefore, according to (3.2), $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij}$ can be calculated by $\underset{\varphi_{ht}\to\varphi'_{ht}}{\Delta} r_{ij} = c_{ht}^{\varphi_{ij}} - c_{ht}^{\varphi'_{ij}}$.



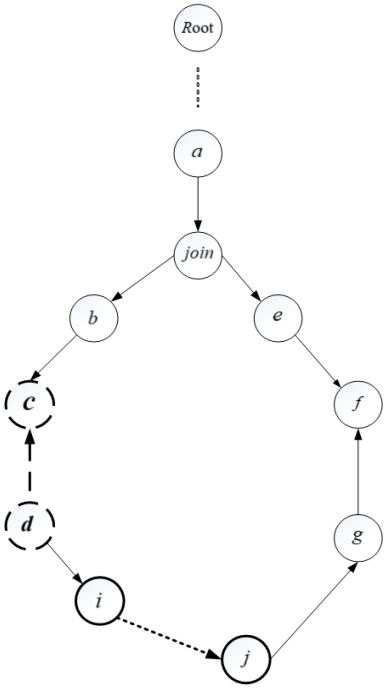
(a) Case 1: $(h,t) = (e,f) \in G^T_{1\cdot ij}$



(b) Case 2: $(h,t) = (g,f) \in G^T_{2\cdot ij}$



(c) Case 3: $(h,t) = (b,c) \in G^T_{3\cdot ij}$



(d) Case 4: $(h,t) = (d,c) \in G^T_{4\cdot ij}$

Figure 4. Four cases of arc $(h,t) \in G^T_{ij}$.

This example illustrates the effect of a single arc on $\underset{\varphi\to\varphi'}{\Delta} r_{ij}$. Considering the effect of all the arcs in G_{ij}^T, we can calculate $\underset{\varphi\to\varphi'}{\Delta} r_{ij}$ by

$$\begin{aligned}\underset{\varphi\to\varphi'}{\Delta} r_{ij} = &\sum_{(h,t)\in G_{1\cdot ij}^T}(c_{ht}^{\varphi_{ht}} - c_{ht}^{\varphi'_{ht}}) + \sum_{(h,t)\in G_{2\cdot ij}^T}(c_{ht}^{\varphi'_{ht}} - c_{ht}^{\varphi_{ht}}) \\ &+ \sum_{(h,t)\in G_{3\cdot ij}^T}(c_{ht}^{\varphi'_{ij}} - c_{ht}^{\varphi_{ht}}) + \sum_{(h,t)\in G_{4\cdot ij}^T}(c_{ht}^{\varphi_{ht}} - c_{ht}^{\varphi'_{ht}}).\end{aligned} \tag{3.5}$$

Based on the concavity of $f_{ij}(x_{ij})$, we have $c_{ij}^k > c_{ij}^{k+1}$. For any arc $(i, j) \in L$, since $\psi(ij) = \arg\min_{\varphi'}\{r_{ij}^{\varphi'} | \forall\varphi' \in \Phi(v^\varphi)\}$, the value of $\psi(ij)_{ht}$ for any $(h, t) \in A$ can be obtained as follows.

$$\psi(ij)_{ht} = \begin{cases} k, & \text{if } (h,t) \in G_{1\cdot ij}^T \cup G_{4\cdot ij}^T; \\ k+1, & \text{if } (h,t) \in G_{2\cdot ij}^T \cup G_{3\cdot ij}^T; \\ \varphi_{ht}, & \text{else.} \end{cases} \tag{3.6}$$

Accordingly, the difference between $r_{ij}^{\psi(ij)}$ and r_{ij}^{φ} can be calculated as

$$\underset{\varphi\to\psi(ij)}{\Delta} r_{ij} = \sum_{(h,t)\in G_{1\cdot ij}^T \cup G_{4\cdot ij}^T}(c_{ht}^{\varphi_{ht}} - c_{ht}^{k}) + \sum_{(h,t)\in G_{2\cdot ij}^T \cup G_{3\cdot ij}^T}(c_{ht}^{k+1} - c_{ht}^{\varphi_{ht}}). \tag{3.7}$$

Similarly, for any arc $(i, j) \in U$, since $\psi(ij) = argmax_{\varphi'}\{r_{ij}^{\varphi'} | \forall\varphi' \in \Phi(v^\varphi)\}$, the value of $\psi(ij)_{ht}$ for any $(h, t) \in A$ can be obtained as follows.

$$\psi(ij)_{ht} = \begin{cases} k+1, & \text{if } (h,t) \in G_{1\cdot ij}^T \cup G_{4\cdot ij}^T; \\ k, & \text{if } (h,t) \in G_{2\cdot ij}^T \cup G_{3\cdot ij}^T; \\ \varphi_{ht}, & \text{else.} \end{cases} \tag{3.8}$$

Accordingly, the difference between $r_{ij}^{\psi(ij)}$ and r_{ij}^{φ} can be calculated as

$$\underset{\varphi\to\psi(ij)}{\Delta} r_{ij} = \sum_{(h,t)\in G_{1\cdot ij}^T \cup G_{4\cdot ij}^T}(c_{ht}^{\varphi_{ht}} - c_{ht}^{k+1}) + \sum_{(h,t)\in G_{2\cdot ij}^T \cup G_{3\cdot ij}^T}(c_{ht}^{k} - c_{ht}^{\varphi_{ht}}). \tag{3.9}$$

In summary, for any nonbasis arc (i, j), if we have already obtained C_{ij}^T, then it is clear that the computational complexity of obtaining $\psi(ij)$ and $\underset{\varphi\to\psi(ij)}{\Delta} r_{ij}$ by (3.6) and (3.7) (or (3.8) and (3.9)) is $O(n)$. Based on the NSA, the computational complexity of searching for a derived cycle C_{ij}^T is also $O(n)$ through backtracking the spanning tree from node i and node j to the root node. Then, $r_{ij}^{\psi(ij)}$ can be obtained by $r_{ij}^{\psi(ij)} = r_{ij}^{\varphi} + \underset{\varphi\to\psi(ij)}{\Delta} r_{ij}$, which requires only an addition operation. Therefore, the total complexity of obtaining $\psi(ij)$ and $\Delta r_{ij}^{\psi(ij)}$ is still $O(n)$.

At this point, the constructive proof of Theorem 4 is completed. □

3.3. Check the local optimality

Using the strategy in the proof of Theorem 4 to obtain $\psi(ij)$, the procedure for checking the local optimality of a vertex v can be expressed as Algorithm 1.

Algorithm 1 Check the local optimality of v.

Require: Feasible domain Ω; $f_{ij}(x_{ij}), \forall (i,j) \in A$, v, $flag = 1$;

• Select an index tuple $\varphi \in \Phi(v)$;
• Solve the linear NFP $P(\varphi)$ and obtain r_{ij}^{φ};
if v is not optimal for $P(\varphi)$ **then**
    • $flag := 0$;
else
    for $\forall (i,j) \in \overline{T}$ **do**
        • Obtain $\psi(ij)$ by (3.6) or (3.8);
        • Calculate $\underset{\varphi \to \psi(ij)}{\Delta} r_{ij}$ by (3.7) or (3.9);
        • Calculate $r_{ij}^{\psi(ij)}$ by $r_{ij}^{\psi(ij)} = r_{ij}^{\varphi} + \underset{\varphi \to \psi(ij)}{\Delta} r_{ij}$;
        if $r_{ij}^{\psi(ij)}$ satisfies condition (2.8) **then**
            • Go to the next arc in $\overline{T}$.
        else
            $flag := 0$;
            • End the current for-loop.
        end if
    end for
end if
if flag == 1 **then**
    • v is locally optimal for CPLNFP;
else
    • v is not locally optimal for CPLNFP;
end if

Theorem 5. *The complexity of checking the local optimality of v by Algorithm 1 is polynomial in the network parameters.*

Proof. If v is not optimal for $P(\varphi)$, we determine that v is not locally optimal for CPLNFP by verifying the optimality of v for only one linear NFP. In this case, the computational complexity of verifying the optimality of v for $P(\varphi)$ is $O(m-n)$ according to the NSA. If v is optimal for $P(\varphi)$ (i.e., $v = v^{\varphi}$), we check the optimality of $v(v^{\varphi})$ by the strategy stated in the proof of Theorem 4 in complexity $O(n(m-n))$. Therefore, the total complexity of checking the local optimality of v by Algorithm 1 is still $O(n(m-n))$, which is polynomial in the network parameters. □

Traditionally, as mentioned earlier, for any vertex v, we need to verify the optimality of v for 2^{m_1} linear NFPs $P(\varphi), \forall \varphi \in \Phi(v)$. The total computational complexity is $O(2^{m_1}(m-n))$ since the complexity of checking v for a linear NFP is $O(m-n)$. It is clear that for large-scale CPLNFPs with $2^{m_1} >> n$, Algorithm 1 is far superior to the traditional implementation of the local optimality condition.

3.4. An example to illustrate Algorithm 1

Example 3. *For the CPLNFP instance in Example 1, we can easily obtain that* $\hat{x} = (3, 4, 1, 2, 0, 5, 2, 5)$ *is a nondegenerate vertex. Moreover,* $\hat{x}$ *is also an optimal solution to* $P(\varphi)$ *for* $\varphi = (1, 2, 1, 2, 1, 2, 2, 2)$*, which means that* $v^{\varphi} = \hat{x} = (3, 4, 1, 2, 0, 5, 2, 5)$*. When solving* $P(\varphi)$ *by the NSA, the basis associated with* v^{φ} *is*

$$\begin{cases} T = \{(1,2),(2,3),(2,4),(3,5),(5,6)\}, \\ L = \{(3,4)\}, \\ U = \{(1,3),(4,6)\}. \end{cases} \tag{3.10}$$

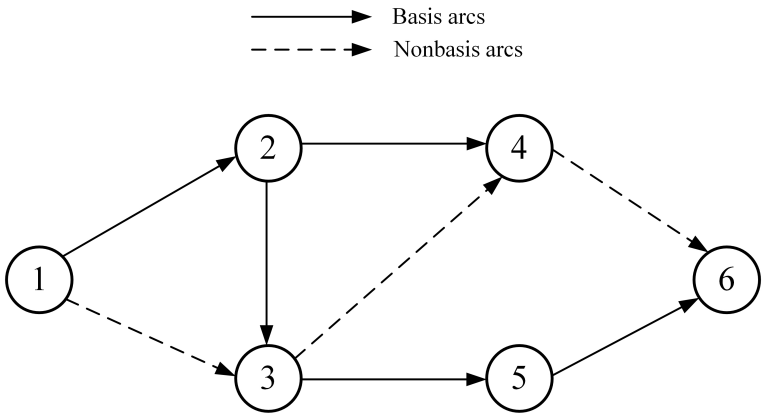


Figure 5. The basis associated with v^{φ}.

Figure 5 shows the basis, where the basis arcs are represented by solid lines and nonbasis arcs are represented by dashed lines. By definition, we have $\overline{T} = L \cup U = \{(1,3),(3,4),(4,6)\}$ *and* $T_1 = \{(1,2),(2,3),(2,4),(3,5),(5,6)\}$*. Let node* 1 *be the root of* T*. The prices associated with* v^{φ} *are* $p_1 = 0, p_2 = -5, p_3 = -8, p_4 = -8, p_5 = -10, p_6 = -14$ *when solving* $P(\varphi)$ *by the NSA. Accordingly, the reduced costs for nonbasis arcs are* $r^{\varphi}_{13} = 4 + (-8) - 0 = -4, r^{\varphi}_{34} = 4 + (-8) - (-8) = 4, r^{\varphi}_{46} = 3 + (-14) - (-8) = -3$*. It can be seen that* r^{φ}_{13}*,* r^{φ}_{34} *and* r^{φ}_{46} *satisfy the optimality condition (2.8), which in turn verifies the optimality of* $\hat{x}$ *for* $P(\varphi)$*.*

For arc $(1,3) \in U$*, the derived cycle is* $C^T_{13} = \{(1,2),(2,3),(1,3)\}$ *and the intersection of* C^T_{13} *and* T_1 *is* $G^T_{13} = \{(1,2),(2,3)\}$*. As discussed in section 3.2,* G^T_{13} *can be divided into four subsets:* $G^T_{1\cdot 13} = \{(1,2),(2,3)\}$*,* $G^T_{2\cdot 13} = \phi$*,* $G^T_{3\cdot 13} = \phi$ *and* $G^T_{4\cdot 13} = \phi$*. According to (3.6) and (3.7), we have* $\psi(13) = (2, 2, 2, 2, 1, 2, 2, 2)$ *and* $\underset{\varphi \to \psi(13)}{\Delta} r_{13} = (5-3) + (3-2) = 3$*. Then,* $r^{\psi(13)}_{13}$ *can be calculated as* $r^{\psi(13)}_{13} = r^{\varphi}_{13} + \underset{\varphi \to \psi(13)}{\Delta} r_{13} = -4 + 3 = -1 < 0$*. Clearly,* $r^{\psi(13)}_{13}$ *still satisfies condition (2.8).*

Similarly, for arc $(3,4) \in L$ *and arc* $(4,6) \in U$*, we can obtain* $\psi(34) = (1, 2, 2, 1, 1, 2, 2, 2)$*,* $\underset{\varphi \to \psi(34)}{\Delta} r_{34} = (2-3) + (3-6) = -4, r^{\psi(34)}_{34} = 4 - 4 = 0$ *and* $\psi(46) = (1, 2, 1, 2, 1, 1, 2, 1)$*,* $\underset{\varphi \to \psi(46)}{\Delta} r_{46} = (3-3) + (4-2) + (6-4) = 4, r^{\psi(46)}_{46} = -3 + 4 = 1 > 0$*. Obviously,* $r^{\psi(46)}_{46}$ *does not satisfy condition (2.8). Therefore,* v^{φ} *is not locally optimal for the CPLNFP instance in Example 1. In practice, when* $r^{\psi(46)}_{46}$ *is obtained, one can determine that* v^{φ} *is not locally optimal and obtain a better solution by solving* $P(\psi(46))$*. In the process of checking the local optimality of* v^{φ}*, we calculate the reduced cost for each nonbasis arc only once.*

Traditionally, we need to check the optimality of v^{φ} for 32 linear NFPs. Moreover, we need to calculate the reduced cost for each nonbasis arc every time we solve one of these 32 linear NFPs. For example, when solving $P(\varphi')$ where $\varphi' = (1,2,2,2,1,1,2,2) \in \Phi(\hat{x})$, we need to recalculate prices as: $p_1 = 0, p_2 = -5, p_3 = -7, p_4 = -8, p_5 = -11, p_6 = -15$, and then check the optimality of v^{φ} by calculating the reduced costs for nonbasis arcs as $r_{13}^{\varphi'} = 4 + (-7) - 0 = -3, r_{34}^{\varphi'} = 4 + (-8) - (-7) = 3, r_{46}^{\varphi'} = 3 + (-15) - (-8) = -4$. Therefore, if we want to check the optimality of v^{φ} for 32 linear NFPs, we need to calculate the reduced cost for each nonbasis arc 32 times.

4. Conclusion

The motivation of our study comes from the demand for the local optimal solution to the CPLNFP. The efficient implementation of the local optimality condition is the key to verify and obtain a local optimal solution. In this paper, we propose a polynomial local optimality condition for the nondegenerate vertex based on the mathematical properties of linear NFPs associated with this vertex. The proposed local optimality condition makes use of the network characteristics of the CPLNFP and is suitable for large-scale CPLNFPs.

Acknowledgments

This project was supported by the National Natural Science Foundation of China (61473165,61977046).

Conflict of interest

All authors declare no conflicts of interest in this paper.

References

1. J. Xu, Y. Tu, Z. Zeng, A nonlinear multiobjective bilevel model for minimum cost network flow problem in a large-scale construction project, *Math. Probl. Eng.*, **2012** (2012), 115–123.
2. V. Prahalad, K. Mathur, R. H. Ballou, An efficient generalized network-simplex-based algorithm for manufacturing network flows, *J. Comb. Optim.*, **15** (2008), 315–341.
3. F. S. Salman, R. Ravi, J. N. Hooker, Solving the Capacitated Local Access Network Design Problem, *Informs J. Comput.,* **20** (2008), 243–254.
4. C. D'Ambrosioa, A. Lodi, S. Wiese, et al., Mathematical programming techniques in water network optimization, *Eur. J. Oper. Res.*, **243** (2015), 774–788.
5. T. L. Magnanti, D. Stratila, Separable concave optimization approximately equals piecewise linear optimization, In: *Integer Programming and Combinatorial Optimization,* Springer, Berlin, 2004.
6. B. W. Lamar, A method for solving network flow problems with general non-linear arc costs, In: *Network Optimization Problems: Algorithms Applications and Complexity*, World Scientific, Singapore, 1993.

7. A. Saif, S. Elhedhli, A Lagrangian heuristic for concave cost facility location problems: the plant location and technology acquisition problem, *Optim. Lett.*, **10** (2016), 1087–1100.
8. Y. Perlman, I. Levner, Perishable Inventory Management in Healthcare, *Journal of Service Science and Management*, **7** (2014), 11–17.
9. S. Yan, Y. L. Shih, W. T. Lee, A particle swarm optimization-based hybrid algorithm for minimum concave cost network flow problems, *J. Global Optim.,* **49** (2011), 539–559.
10. Z. Xu, K. Liu, X. Xi, S. Wang, Method of hill tunneling via weighted simplex centroid for continuous piecewise linear programming, *2015 Conference on Decision and Control*, **24** (2015), 301–316.
11. D. Kim, P. M. Pardalos, Dynamic slope scaling and trust interval techniques for solving concave piecewise linear network flow problems, *Networks*, **35** (2000), 216–222.
12. A. Nahapetyan, P. M. Pardalos, A bilinear relaxation based algorithm for Concave Piecewise Linear Networks Flow Problems, *J. Ind. Manag. Optim.*, **3** (2007), 71–85.
13. A. R. Conn, M. Mongeau, Discontinuous piecewise linear optimization, *Math. Program.*, **80** (1998), 315–380.
14. L. T. H. An, P. D. Tao, The DC (Difference of Convex Functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems, *Ann. Oper. Res.*, **133** (2005), 23–46.
15. X. Huang, J. Xu, X. Mu, S. Wang, The hill detouring method for minimizing hinging hyperplanes functions, *Comput. Oper. Res.*, **39** (2012), 1763–1770.
16. G. B. Dantzig, *Linear Programming and Extensions*, Journal of Industrial and Management Optimization, Princeton University Press, NJ, 1963.
17. P. Kovacs, Minimum-cost flow algorithms: an experimental evaluation, *Optim. Methods Softw.*, **30** (2015), 94–127.
18. M. Holzhauser, S. O. Krumke, C. Thielen, A Network Simplex Method for the Budget-Constrained Minimum Cost Flow Problem, *Eur. J. Oper. Res.*, **259** (2017), 864–872.
19. D. Klingman, A. Napier, J. Stutz, NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems, *Manage. Sci.*, **20** (1973), 814–821.

Appendix

A. The growth trend of the number of active arcs

This appendix estimates the growth trend of the number of active arcs (m_1) in the CPLNFPs defined on benchmark networks.

A.1. Test problems

Test problems used to estimate the growth trend of m_1 are generated randomly. Each test problem consists of two parts: the network and the concave piecewise linear cost function. The network is

generated randomly with parameters n and m by the benchmark network generator NETGEN [19]. The total supply flow is equal to the total demand flow and is set as $10 \times n$. The capacity u_{ij} for arc (i, j) is generated along with G, as stated in [19]. Note that parameters n and m are correlated; thus, we use n/m to denote these two parameters. Without loss of generality, it is assumed that s_{ij} remains the same over all arcs and is expressed as $\hat{s}$. The concave piecewise linear cost function $f_{ij}(x_{ij})$ for each arc $(i, j) \in A$ is generated according to the following three steps.

- **Step 1:** The breakpoints $(\lambda_{ij}^1, \lambda_{ij}^2, \cdots, \lambda_{ij}^{\hat{s}-1})$ are randomly generated in $(0, u_{ij})$, which satisfies $\lambda_{ij}^1 < \lambda_{ij}^2 < \cdots < \lambda_{ij}^{\hat{s}-1}$.
- **Step 2:** The slopes $(c_{ij}^1, c_{ij}^2, \cdots, c_{ij}^{\hat{s}})$ are randomly generated in $[c_{min}, c_{max}]$, which satisfies $c_{ij}^1 > c_{ij}^2 > \cdots > c_{ij}^{\hat{s}}$. In this paper, we set $c_{min} = 5$ and $c_{max} = 5\hat{s}$.
- **Step 3:** The intercepts $(d_{ij}^1, d_{ij}^2, \cdots, d_{ij}^{\hat{s}})$ are generated iteratively based on the continuity of $f_{ij}(x_{ij})$ as follows:
  - $d_{ij}^1 = 0$;
  - **For** $s = 2$ to $\hat{s}$ **do**
    - $d_{ij}^s = c_{ij}^{s-1}\lambda_{ij}^{s-1} + d_{ij}^{s-1} - c_{ij}^s\lambda_{ij}^{s-1}$.

Considering that various practical problems require various network dimension parameters n/m and different piecewise linear cost function parameters $\hat{s}$, we set ten different values for parameter n/m and four different values for parameter $\hat{s}$, which can be expressed as as Ξ={10/20 20/40 40/100 60/400 80/600 100/1000 200/2000 300/3000 400/6000 500/8000 } and $S = \{3, 5, 7, 10\}$, respectively. Problems with the same n/m and $\hat{s}$ form a problem set, which is denoted as $P_{n,m,\hat{s}}$. In each problem set $P_{n,m,\hat{s}}$, there are 20 randomly generated test problems.

In this paper, the problem group that consists of all test problems is denoted as P. According to parameter $\hat{s}$, P is divided into four problem subgroups P_1, P_2, P_3 and P_4.

$$P_1 = \{P_{n,m,3} | n/m \in \Xi\}, P_2 = \{P_{n,m,5} | n/m \in \Xi\},$$
$$P_3 = \{P_{n,m,7} | n/m \in \Xi\}, P_4 = \{P_{n,m,10} | n/m \in \Xi\}.$$

Clearly, there are 200 test problems in $P_k, k = 1, 2, 3, 4$ and a total of 800 test problems in P.

A.2. Estimate the growth trend of m_1

To study the relationship between m_1 and the parameters n, m and $\hat{s}$, we estimate the value of m_1 in problem groups P_1, P_2, P_3 and P_4. The test problems in each problem set $P_{n,m,\hat{s}}$ have the same parameters n, m and $\hat{s}$; thus, they are estimated as a whole. The average of the estimated values of m_1 for all 20 problems is used to estimate the overall level of m_1 for $P_{n,m,\hat{s}}$. For a test problem, each vertex in the feasible region corresponds to a value of m_1. Since the number of vertices is combinatorial, it is not realistic to find values of m_1 for all vertices. Here, we propose a reasonable sampling method to estimate the value of m_1.

For ease of expression, we use the concept of $P(\varphi)$ that is a linear NFP defined from one of the regions in the feasible domain. The formal definition of $P(\varphi)$ will be given in (2.3). The linear NFPs defined from all the possible regions in the feasible domain can be expressed as

$$\hat{P} = \{P(\varphi) | \varphi \in \hat{\Phi}\},$$

where $\hat{\Phi}$ is defined as

$$\hat{\Phi} = \{\varphi \in \Theta | \varphi_{ij} \in \{1, 2, \cdots, \hat{s}\}, \forall (i, j) \in A\}.$$

The definition of Θ can be seen in section 2.1.

For a vertex v in the feasible domain, if v is not optimal for $P(\varphi), \forall \varphi \in \hat{P}$, then we can determine v is not locally optimal for the CPLNFP by solving any one of these linear NFPs. Therefore, we do not have to explore the value of m_1 for this vertex. We should focus on vertices that are at least optimal for one linear NFP in $\hat{P}$. It is appropriate to sample vertices from the optimal solutions to problems in $\hat{P}$.

For a test problem $p \in P$ with the feasible region Ω, the process of sampling n_s vertices is shown as follows.

For $k = 1 : n_s$ **do**

- Randomly select φ satisfying $\varphi \in \hat{\Phi}$;
- Solve $P(\varphi)$ by the network simplex algorithm ([16]) and obtain the optimal vertex v^{φ};
- Calculate the value of m_1 associated with v^{φ};
- $\gamma_k = m_1$;
- $\hat{\Phi} := \hat{\Phi} \backslash \varphi$;

End

The average of $\gamma_1, \gamma_2, \cdots, \gamma_{n_s}$ is used to estimate the value of m_1 in this problem.

$$\eta_p = \frac{\gamma_1 + \gamma_2 + \cdots + \gamma_{n_s}}{n_s}. \tag{A.1}$$

In our experiments, we set $n_s = n$. The overall level of m_1 for $P_{n,m,\hat{s}}$ is estimated by

$$\zeta = \frac{\sum_{p \in P_{n,m,\hat{s}}} \eta_p}{20}. \tag{A.2}$$

Table 1. The overall level of the value of m_1.

Problem	n	10	20	40	60	80	100	200	300	400	500
Group	m	20	40	100	400	600	1000	2000	3000	6000	8000
$P_1(\hat{s} = 3)$	ζ	0.15	0.34	0.80	1.24	1.78	2.25	3.85	5.40	8.12	10.50
$P_2(\hat{s} = 5)$	ζ	0.32	0.98	2.03	2.42	3.55	4.35	7.75	11.20	14.98	18.31
$P_3(\hat{s} = 7)$	ζ	0.60	1.20	3.12	3.96	5.56	6.40	12.42	17.05	21.88	28.91
$P_4(\hat{s} = 10)$	ζ	0.87	2.14	3.94	5.94	7.22	9.93	18.15	26.84	33.91	41.58

It can be seen in Table 1, as the scale of the problem increases, the value of ζ gradually increases and the value of 2^{ζ} becomes very large. For example, we have $2^{\zeta} \approx 1.71 \times 10^{47} >> 500$ for test problems in $P_{500,8000,10}$. Moreover, for the same problem scale, the value of ζ also increases as the value of $\hat{s}$ increases. The specific growth trend of ζ can be seen in Figure 2 in Introduction.