Josai Mathematical Monographs
vol. 9 (2016), pp. 21-29

Bland's rule for the Network Simplex Algorithm

Sennosuke WATANABE and Yoshihide WATANABE

Abstract. In the present paper, we give the rigorous proof of the result that Bland's rule for network simplex algorithm prevent the cycling.

1. Introduction

In the present paper, we focus on the minimum cost flow problem which is one of the well-known optimization problem on flow-networks. The minimum cost flow problem is the problem for finding the minimum cost flow which satisfies the given capacity conditions and demand conditions. The most efficient algorithm for solving the minimum cost flow problem is the network simplex algorithm. This algorithm gives an optimal solution to the minimum cost flow problem by updating the so called tree structures, but it does not necessarily terminate in a finite number of steps, even for small networks. This infinite iteration is called the cycling. It is known that the cycling in the network simplex algorithm can be prevented by the rule of the last blocking edge. On the other hand, the problem of the cycling also arise in the simplex algorithm for linear programming (LP) problems. One of the famous rules for preventing the cycling in the simplex algorithm for LP is Bland's rule. We can easily expect that Bland's rule is applicable to the network simplex algorithm and prevent the cycling, because of the similarity of the both algorithms. However, we do not obtain a rigorous proof. The purpose of the present paper is to give the rigorous proof of the result that Bland's rule for network simplex algorithm prevent the cycling.

2. Minimum Cost Flow Problem

Let $G = (V, E)$ be a digraph with n vertices and m edges. We assign positive integers $b(e)$ and $c(e)$ to each edge $e \in E$ satisfying the inequality $0 \leq b(e) \leq c(e)$; $b(e)$ is called the lower capacity of e and $c(e)$ is called the upper capacity of e. We also assign a positive real number $\gamma(e)$ to each edge $e \in E$; $\gamma(e)$ is called the cost of e. Moreover, we assign the integer $d(v)$ to each vertex $v \in V$ with $\sum_{v \in V} d(v) = 0$; $d(v)$ is called the demand of v. The quadruple $\mathcal{N} = (G, b, c, d)$ is called a network on G. A flow on the network $\mathcal{N}$ is the function f on E satisfying the following conditions (i) and (ii):

(i) The capacity constraint at each edge:

$$b(e) \leq f(e) \leq c(e) \quad \text{for all } e \in E \ .$$

(ii) The demand condition at each vertex:

$$\sum_{\partial^+(e)=v} f(e) - \sum_{\partial^-(e)=v} f(e) = d(v) \quad \text{for all } v \in V \ .$$

Here, we introduce the maps $\partial^- : E \to V$ and $\partial^+ : E \to V$ by $\partial^-(e) = u$ and $\partial^+(e) = v$ for $e = (u, v)$ respectively. The minimum cost flow problem (MCFP) is the problem for finding the flow f whose cost $\gamma(f) = \sum_{e \in E} \gamma(e) f(e)$ is minimal. That is, the MCFP is formulated as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{e \in E} \gamma(e) f(e) \\ \text{subject to} \quad & \sum_{\partial^+(e)=v} f(e) - \sum_{\partial^-(e)=v} f(e) = d(v) \\ & b(e) \leq f(e) \leq c(e) \ . \end{aligned} \tag{1}$$

3. Network Simplex Algorithm

3.1. Tree Structure

We consider the MCFP on a digraph G. Let T be a spanning tree of G. We often identify the spanning tree T and its edges set $E(T)$ and use the same symbol T for $E(T)$. We divide the edge set $E \setminus T$ into the disjoint union as $E \setminus T = L \cup U$. We note that subsets $L, U \subset E$ are allowed to be empty sets. Fix the triplet (T, L, U), we define the function f on the edge set E as follows: First define the function f on L and U by $f(e) = c(e)$ $(e \in L)$ and $f(e) = b(e)$ $(e \in U)$ respectively. Then it is well-known that the values of the function f on the spanning tree T are uniquely determined by the demand condition. Thus we see that the function f on E is uniquely determined by the triplet (T, L, U). Note that the function f does not always become the flow on the network $\mathcal{N}$ because it does not always satisfies the capacity constraints on T. If the function f uniquely determined by the triplet (T, L, U) become the flow on the network, the flow f is called the tree solution associated with the feasible tree structure (T, L, U). We call an edge e free with respect to the flow on the network $\mathcal{N}$ provided that $c(e) < f(e) < b(e)$ holds. It follows directly from the definition of the tree solution that the free edge with respect to the tree solution must be contained in T. However all edges of T need not be free with respect to the tree solution. A feasible tree structure is called optimal if the unique tree solution associated with the tree structure is optimal. It is well-known that if the MCFP has an optimal solution then there exists an

optimal tree solution. The network simplex algorithm is an algorithm for solving MCFP by updating the feasible tree structure.

3.2. Network Simplex Algorithm

The network simplex algorithm consists of the following 3 steps:

0. Find the initial feasible tree structure.
1. Decide whether the given tree structure is optimal or not.
2. If the tree structure is not optimal, then update the tree structure to improve the cost of the tree solution.

We are concerned only with the degeneracy and cycling of the network simplex algorithm in the present paper, we have to describe the updating process of the tree structure (step 1 and step 2) in detail. So we will not refer to step 0 of the algorithm and cite [**3**] for the algorithm and omit the description.

3.2.1 Optimality Condition

Consider the MCFP on the network $\mathcal{N}$ defined on a digraph G and let f be the tree solution associated with a feasible tree structure (T, L, U). We determine the potential $\pi : V \to \mathbb{R}$ by the following procedure: Fix a vertex $x \in V$, and set $\pi(x) = 0$. For each vertex $v \in V \setminus \{x\}$, there exists the unique undirected path P_v from x to v in the spanning tree T. We define the direction of the path P_v from x to v. Then all the edges in P_v is divided into forward edges and backward edges with respect to the direction of P_v: the set of forward edges is denoted by P_v^+ and the set of backward edges by P_v^- respectively. The potential π is defined by

$$\pi(v) = \sum_{e \in P_v^+} \gamma(e) - \sum_{e \in P_v^-} \gamma(e) \quad \text{for all } v \in V \setminus \{x\} \ .$$

Further, we define the reduced cost $\gamma_\pi : E \to \mathbb{R}$ in terms of the potential π by

$$\gamma_\pi(e) = \gamma(e) + \pi(v_i) - \pi(v_j) \quad \text{for all } e = (v_i, v_j) \in E \ .$$

Then we have

$$\begin{aligned}
\gamma_\pi(f) &= \sum_{e \in E} f(e)\gamma_\pi(e) \\
&= \sum_{e \in E} f(e)(\gamma(e) + \pi(v_i) - \pi(v_j)) \\
&= \sum_{e \in E} f(e)\gamma(e) + \sum_{v_i \in V} \sum_{\partial^-(e) = v_i} f(e)\pi(v_i) - \sum_{v_j \in V} \sum_{\partial^+(e) = v_i} f(e)\pi(v_j)
\end{aligned}$$

$$= \gamma(f) + \sum_{v \in V} \pi(v)(\sum_{\partial^-(e)=v} f(e) - \sum_{\partial^+(e)=v} f(e))$$
$$= \gamma(f) - \sum_{v \in V} \pi(v) d(v) \ .$$

So the cost $\gamma(f)$ of the tree solution f and the reduced cost $\gamma_\pi(f)$ of f differ only by a constant which is independent of f. So it is easy to see that a tree structure (T, L, U) is optimal if and only if the reduced cost γ_π on (T, L, U) satisfies the following condition:

$$\gamma_\pi(e) \begin{cases} = 0 \ \text{ for all } \ e \in T \ , \\ \geq 0 \ \text{ for all } \ e \in L \ , \\ \leq 0 \ \text{ for all } \ e \in U \ . \end{cases} \tag{2}$$

3.2.2 The algorithm

We consider the digraph $G = (V, E)$ with the set of n edges $E = \{1, 2, \ldots, n\}$ and m vertices. We express the flow value on the edge i by f_i We will give the expression of the cost function (objective function) and constraints in terms of the tree solution f associated with some feasible tress structure (T, L, U). First we note that the cost $\gamma(f)$ can be written as $\gamma(f) = \gamma_\pi(f) - \sum_{v \in V} \pi(v) d(v)$ with the reduced cost $\gamma_\pi(f)$. If we use the fact that $\gamma_\pi(j) = 0$ for $j \in T$, and setting $\Phi = -\sum_{v \in V} \pi(v) d(v)$, we have

$$\gamma(f) = \Phi + \sum_{\nu \in L} \gamma_\pi(\nu) f_\nu + \sum_{\mu \in U} \gamma_\pi(\mu) f_\mu \ . \tag{3}$$

Eq.(3) express the cost γ using the reduced cost γ_π with respect to a feasible tree structure (T, L, L) We note the flow value f_ν of $\nu \in L$ is equal to the lower capacity $b(\nu)$ and the flow value f_μ of $\mu \in U$ is equal to the upper capacity $c(\mu)$.

Next we consider the constraints of the MCFP. Since the flow satisfies the demand condition, if a tree structure (T, L, U) is given, then the flow value f_j for $j \in T$ is uniquely determined by the flow values $f_\nu = b(\nu)$ $(\nu \in L)$ and by the flow values $f_\mu = c(\mu), (\mu \in L)$ as follows:

$$f_j + \sum_{\nu \in L} \alpha_{j\nu} f_\nu + \sum_{\mu \in U} \alpha_{j\mu} f_\mu = \beta_j \tag{4}$$

Here, coefficients $\alpha_{j\nu}$ and $\alpha_{j\mu}$ express whether the flow value f_j of $j \in T$ depends on the flow value f_ν of $\nu \in L$ and the flow value f_μ of $\mu \in U$, respectively. The constants β_j is determined by the demand conditions. Then, we have the expression

of the MCFP in terms of the tree structure as follows:

$$\begin{aligned} \text{minimize} \quad & \Phi + \sum_{\nu \in L} \gamma_\pi(\nu) f_\nu + \sum_{\mu \in U} \gamma_\pi(\mu) f_\mu \\ \text{subject to} \quad & f_j + \sum_{\nu \in L} \alpha_{j\nu} f_\nu + \sum_{\mu \in U} \alpha_{j\mu} f_\mu = \beta_j \ . \end{aligned} \tag{5}$$

Under the above preparations, we describe procedures of the network simplex algorithm.

Algorithm 3.1 (Network simplex algorithm). *Consider the MCFP on a digraph G. Let (T, L, U) be a tree structure associated with the tree solution f. The optimal solution of the MCFP is computed by the following procedures:*

1. *If the reduced cost γ_π on (T, L, U) satisfies the optimal condition (2), then (T, L, U) is an optimal tree structure and associated tree solution f is an optimal solution of the MCFP. Else go to the next step.*

2. *Chose some $\tau \in L \cup U$ satisfying (i) $\tau \in L$ and $\gamma_\pi(\tau) < 0$ or (ii) $\tau \in U$ and $\gamma_\pi(\tau) > 0$, and add the edge τ to T: $T \cup \{\tau\}$. We call the edge τ the entering edge. The set of edges $T \cup \{\tau\}$ contains the unique circuit C.*

3. *Consider C as oriented in the direction of τ if $\tau \in L$, and as oriented in the direction opposite to that of τ if $\tau \in U$. Augment f by an amount of δ by cancelling C, so that at least one edge of C reaches either its upper or lower capacity bound. In the case of the entering edge $\tau \in L$, the coefficient α in (4) is decided by:*

$$\alpha_{j\tau} = \begin{cases} 0 & \text{if } j \in T \text{ and } j \notin C \\ -1 & \text{if } j \in T \text{ and } j \in C \text{ and} \\ & \quad j \text{ is forward edge with respect to the direction of } C \\ 1 & \text{if } j \in T \text{ and } j \in C \text{ and} \\ & \quad j \text{ is backward edge with respect to the direction of } C \end{cases}$$

The increase δ of f is computed by

$$\delta = min\{min_j\{c(j) - f_j | \alpha_{j\tau} < 0\}, min_j\{f_j - b(j) | \alpha_{j\tau} > 0\}\} \ . \tag{6}$$

In the case of the entering edge $\tau \in U$, the coefficient α in (4) is decided by:

$$\alpha_{j\tau} = \begin{cases} 0 & \text{if } j \in T \text{ and } j \notin C \\ 1 & \text{if } j \in T \text{ and } j \in C \text{ and} \\ & \quad j \text{ is forward edge with respect to the direction of } C \\ -1 & \text{if } j \in T \text{ and } j \in C \text{ and} \\ & \quad j \text{ is backward edge with respect to the direction of } C \end{cases}$$

The increase δ of f is computed by

$$\delta = min\{min_j\{f_j - b(j)|\alpha_{j\tau} < 0\}, min_j\{c(j) - f_j|\alpha_{j\tau} > 0\}\} \ . \qquad (7)$$

Chose an edge i whose value of the flow reaches either its upper or lower capacity bound. We call the edge i the leaving edge.

4. *Update the new tree structure (T', L', U') as follows:*

$$T' := (T \setminus \{i\}) \cup \{\tau\},$$
$$L' := \begin{cases} (L \setminus \{\tau\}) \cup \{i\} & \text{if } i \text{ reaches its lower capacity bound} \ , \\ L \setminus \{\tau\} & \text{if } i \text{ reaches its upper capacity bound}, \end{cases}$$
$$U' := E \setminus (T \cup L) \ .$$

Compute the potential π of (T', L', U') and the reduced cost γ_π. Go to Step 1.

4. Network Simplex Algorithm with Bland's Rule

The network simplex algorithm 3.1 does not necessarily terminate in a finite number of iterations. This inconvenience is caused by the degeneracy of the tree structure. If the spanning tree T in the tree structure (T, L, U) does not consist of free edges only, then the tree structure (T, L, U) is called degenerate. In this case, a proper augmentation along the circuit determined in Step 2 of the algorithm 3.1 may be impossible since we may have $\delta = 0$ in Step 3. In such a case only the feasible tree structure is updated and the tree solution associated with the tree structure is not updated. In this case we may reach the same tree structure as the first one after a number of iterations without updating the tree solution. This phenomenon is called the cycling. It is known that the cycling can be prevented by choosing the leaving edge appropriately: for example the so called rule of the last blocking edge or the rule of the first blocking edge [**3, 5**]. In the present paper, we present the new rule for preventing the cycling: the analogue of famous Bland's rule for Linear Programming. We determine the order of all edges by $1 < 2 < \cdots < n$. We choose the entering edge with the smallest number if we have some candidates in Step 2 and choose the leaving edge with the smallest number if we have some candidates in Step 3 of the algorithm. We simply call this rule Bland's rule.

THEOREM 4.1. *Consider the MCFP on the digraph G with the set of edges $E = \{1, 2, \ldots, n\}$ and with the set of m vertices V. We determine the order of edges by $1 < 2 < \cdots < n$. The network simplex algorithm with Bland's rule terminates in a finite number of iterations.*

PROOF. Let $J_1 = (T_1, L_1, U_1), J_2 = (T_2, L_2, U_2), \ldots$ be the sequence of the tree

structures which is obtained by the procedure in the network simplex algorithm 3.1. If the cycling occurs then we have $J_i = J_j$ $(i < j)$ for some i, j. We note that there is no modification of flow from the tree structure J_i to the tree structure J_j. From the tree structure J_i to that of J_j, the set of edges entering the spanning tree coincides with the set of edges leaving the spanning tree. We denote by Q the set of all entering (leaving) edges and let q be the maximum edge in Q with respect to the edge ordering. We consider the updating process of tree structures illustrated by the following diagram:

$$J_1 \to \cdots \to J_i \to \cdots \to J \overset{\overset{s}{\downarrow}}{\underset{\underset{q}{\downarrow}}{\to}} J_a \to \cdots \to J^* \overset{\overset{q}{\downarrow}}{\underset{\underset{p}{\downarrow}}{\to}} J_b \to \cdots \to J_j \to \cdots$$

The updating from $J = (T, L, U)$ to $J_a = (T_a, L_a, U_a)$ we have

$$(T \cup \{s\}) \setminus \{q\} = T_a$$

and the updating from $J^* = (T^*, L^*, U^*)$ to $J_b = (T_b, L_b, U_b)$ we have

$$(T^* \cup \{q\}) \setminus \{p\} = T_b \ .$$

The expression of the cost and the constraints with respect to J is written as

$$\Gamma = \Phi + \sum_{\nu \in L} \gamma_\nu f_\nu + \sum_{\mu \in U} \gamma_\mu f_\mu \tag{8}$$

$$f_j + \sum_{\nu \in L} \alpha_{j\nu} f_\nu + \sum_{\mu \in U} \alpha_{j\mu} f_\mu = \beta_j \tag{9}$$

and the expression of the cost and the constraints with respect to J^* is writen as

$$\Gamma^* = \Phi^* + \sum_{\nu \in L^*} \gamma^*_\nu f_\nu + \sum_{\mu \in U^*} \gamma^*_\mu f_\mu \tag{10}$$

$$f_j + \sum_{\nu \in L^*} \alpha^*_{j\nu} f_\nu + \sum_{\mu \in U^*} \alpha^*_{j\mu} f_\mu = \beta^*_j \ . \tag{11}$$

In the updating from J to J_a, the entering edge s is the minimal edge in $\tau \in E$ which satisfies (i) $\tau \in L$ and $\gamma_\tau < 0$ or (ii) $\tau \in U$ and $\gamma_\tau > 0$. So we have to consider these two cases.

Case (i). $s \in L$ and $\gamma_s < 0$

Consider the following solution of the equation (9):

$$\begin{aligned} &f_s = b(s) + \varepsilon, \ f_i = b(i) \text{ for } i \in L \setminus \{s\}, f_i = c(i) \text{ for } i \in U \setminus \{s\} \\ &f_j = \beta_j - \alpha_{js}\varepsilon \text{ for } j \in T \ . \end{aligned} \tag{12}$$

We note that there is no modifications of flow while the cycling occurs. So the value of Γ in (8) is equal to the value of Γ^* in (10). And the equation (9) is equivalent to the equation (11), the solution (12) of (9) also become the solution of (11). Substituting (12) into (10), and using $\Gamma = \Gamma^*$, we have

$$\Phi + \gamma_s \varepsilon = \Phi^* + \sum_{\nu \in T \cap L^*} \gamma_\nu^* (\beta_\nu - \alpha_{\nu s} \varepsilon) + \gamma_s^* \varepsilon \ .$$

Thus we have

$$(\gamma_s - \gamma_s^* + \sum_{\nu \in T \cap L^*} \gamma_\nu^* \alpha_{\nu s}) \varepsilon = \sum_{\nu \in T \cap L^*} \gamma_\nu^* \beta_\nu + \Phi^* - \Phi \ .$$

Since ε is arbitrary, then we have

$$\gamma_s - \gamma_s^* + \sum_{\nu \in T \cap L^*} \gamma_\nu^* \alpha_{\nu s} = 0 \ .$$

First we prove $\gamma_s^* \geq 0$. Assume on the contrary $\gamma_s^* < 0$, then the edge $s \in L^*$(There is no modification of flow in the cycling process so that $s \notin U^*$ but $s \in L^*$.) and s becomes a candidate of the entering edge in the process $J^* \to J_b$. This contradicts the assumption that we does not chose the edge s but choose the edge q of the maximum order as the entering edge in the updating from J^* to J_b. By the assumption $\gamma_s < 0$, we have $\gamma_s - \gamma_s^* < 0$. Then there is an edge $t \in T \cap L^*$ whose cost γ_t^* in J^* satisfies $\gamma_t^* \alpha_{ts} > 0$. Since the edge t is contained in T and L^*, t is contained in Q and satisfies $t \leq q$. We will prove $t \neq q$. The edge q is the entering edge in the updating from J^* to J_b. So q satisfies (a) $q \in L^*$ and $\gamma_q^* < 0$ or (b) $q \in U^*$ and $\gamma_q^* > 0$.

Case (i)-(a). $q \in L^*$ and $\gamma_q^* < 0$

Since the cost γ_q^* of the edge q satisfies $\gamma_q^* \alpha_{qs} > 0$ and $\gamma_q^* < 0$, we have $\alpha_{qs} < 0$. The flow value f_q of q in J is equal to the lower bound $b(q)$ in the cycling process. This implies that the flow is updated in the updating from J to J_a, which contradicts the assumption.

Case (i)-(b). $q \in U^*$ and $\gamma_q^* > 0$

Since the cost γ_q^* of the edge q satisfies $\gamma_q^* \alpha_{qs} > 0$ and $\gamma_q^* > 0$, we have $\alpha_{qs} > 0$. The flow value f_q of q in J coincides with the upper bound $c(q)$ in the cycling occurring. This implies that the flow is updated in the updating from J to J_a, which contradicts the cycling occurring.

Then we can proved that $t \neq q$ and $t < q$. Since the edge q is the entering edge in the updating from J^* to J_b, the cost γ_t^* of the edge $t \in L^*$ has to satisfy $\gamma_t^* \geq 0$. The cost γ_t^* also satisfies $\gamma_t^* \alpha_{ts} > 0$. So we have $a_{ts} > 0$. The flow value f_t of t in J coincides with the lower bound $b(t)$ in the cycling occurring. This implies that

the edge t becomes a candidate of the leaving edge in the updating from J to J_a. This contradicts the assumption that the maximum edge q is the leaving edge in this updating. So we have proved the assertion in the case (i).

We can lead the contradiction to the assertion in the case (ii) by the similar manner as the case (i). So the we have proved that the cycling does not occur in the network simplex algorithm with Bland's rule. □

5. Conclusion

In the present paper, we prove that Bland's rele for the network simplex algorithm prevent the cycling. However, we do not know the practical efficiency of our result. So we will remain such kind of investigation for the future study.

References

[1] K. Ahuja, L. Magnanti and B. Orlin: *NETWORK FLOWS*, Prentice-Hall, New Jersey, 1993.

[2] A. Bachem and W. Kern: *Linear Programming Duality*, Springer Verlag, Berlin Heidelberg, 1992.

[3] D. Jungnickel: *Graphs, Networks and Algorithm*, Second Edition, Springer Verlag, Berlin Heidelberg, 2005.

[4] B. Korte and J. Vygen: *Combinatorial Optimization*, Second Edition, Springer Verlag, 2001.

[5] S. Yoneda, S. Watanabe and Y. Watanabe: *A New Rule for Preventing the Cycling in the Network Simplex Algorithm*, The Science and Engineering Review of Doshisha University, Science and Engineering Research Institute of Doshisha University, Vol.53 No.1, pp.54-57, 2012 (in Japanese).

Sennnosuke Watanabe

Organization for Reserch Initiatives and Development, Doshisha University
1-3 Tatara Miyakodani, Kyotanabe, Kyoto, 610–394, Japan
sewatana@mail.doshisha.ac.jp

Yoshihide Watanabe

Faculty of Science and Engineering Department of Mathematical Sciences, Doshisha University
1-3 Tatara Miyakodani, Kyotanabe, Kyoto, 610–394, Japan
yowatana@mail.doshisha.ac.jp