Least-Delay Widest-Bandwidth Routing Algorithm*

JIA-QING HUANG, ZONG-KAI YANG, XU DU
Department of Electronics and Information Engineering
Huazhong University of Science and Technology
Wuhan, 430074, Hubei, P.R.China

Abstract: - In this paper, available bandwidth instead of cost is taken as the prime metric of the routing algorithms in that available bandwidth is the appropriate representative of the real network. On account of concave bandwidth, this paper first presents a novel $k$-widest paths algorithm. Then, a novel end-to-end least-delay widest-available bandwidth path routing algorithm based on $k$-widest paths is proposed. The correctness and looplessness of the novel algorithm are proved as well as its polynomial complexity.

Key-Words: - Group multicast, Routing algorithm, Available bandwidth, Delay, Cost, $K$-widest paths

1 Introduction
With the increasing multimedia applications throughout the whole Internet, the study on multicast in the sparse environment becomes one of the hot spots. In general, Protocol Independent Multicast-Sparse Mode (PIMSM) is one of the satisfied solutions.

Nevertheless, there are two problems: (1) PIMSM adopts shared tree, which can easily lead to congestion at the Rendezvous Point (RP); (2) PIMSM in group multicast (i.e. multiple-session multicast) is likely to result in congestion as well. The reasons are as follows. First, PIMSM makes use of underlying unicast routing table. While most unicast routing algorithms adopt shortest path tree (SPT) algorithm and simply adopt hop as the metric. Thus many multicast sessions will try to occupy the same critical link of the path with the shortest hops. That will certainly give rise to congestion and packet loss. In fact, there are many links with big available bandwidth that can be used, although those links are not along the path with the shortest hops.

On the other hand, bandwidth and delay are the most significant representatives of the real network. Furthermore, bandwidth is more critical than delay [1]. Whereas the additive cost, adopted by plenty of multicast routing algorithms, cannot be directly transformed to the concave bandwidth [2]. That is, the cost-optimal multicast routing algorithms cannot directly guarantee the optimal usage of the bandwidth resources.

Therefore, we take available bandwidth as the prime metric instead of cost to build a modified PIMSM especially for group multicast in our project*. The main contents involve finding the appropriate RP and connecting a source node with the RP using a least-delay widest-available bandwidth path. This paper focuses on the latter.

In order to find the least-delay widest available bandwidth path, this paper proposes a novel loopless $k$-widest paths (KWP) algorithm as well.

Section 1 is the introduction to the paper. Section 2, the Related Work, is the summary of the previous research. Section 3, the Notations and Definitions, gives the basic knowledge. Section 4 introduces the detailed procedures of KWP and provides an illustration. Section 5, the Least-Delay Widest-Available Bandwidth Path (LDWABP) algorithm is proposed in the same section. Finally, the conclusions are drawn in Section 6.

2 Related Work
As for the two-metrics QoS multicast routing, the classical constrained Steiner Minimum Tree (SMT) problem algorithms with total cost and delay, such as KPP, CAO, BSMA, CDKS and MSC, were discussed in detail in [3]. Mokbel presented a delay-bounded end-to-end shortest path algorithm DCSP that used DAG [4]. Rouskas proposed a delay- and delay variation- based multicast routing algorithm DVMA that adopted the $k$-shortest paths algorithm [5]. However, they did not consider the metric of bandwidth.

With regard to the bandwidth-based multicast routing algorithm, the centralized (and distributed) widest bandwidth path can be derived from
modifying the operations of minimization and addition of Dijkstra (and Bellman-Ford) into maximization and minimization, respectively [1,6].

Literature presented a centralized bandwidth-delay multicast routing algorithm [1]. It eliminates any links with their available bandwidth less than requirements. Thus the remaining work is to construct a least delay tree (LDT), which can be computed by Dijkstra. It avoids the essential consideration of the metric of bandwidth.

In the same literature, a loopless distributed shortest-widest path (SWP) algorithm was proposed [1]. SWP is based on distributed widest bandwidth path algorithm using modified Bellman-Ford mentioned above. If multiple paths exist, the path with the minimum delay is selected [1]. The complexity is $O(n^3)$. In essence, it is the algorithm of the single metric of bandwidth.

Correspondingly, a widest-shortest path (WSP) algorithm based on Dijkstra was addressed in [6]. In the WSP, if multiple paths exist, the path with maximum bandwidth is chosen. The complexity is $O(n^3)$. In fact, the two metrics are reduced to one metric of delay.

The author presented a source-based multicast routing algorithm BDVMA with the widest available bandwidth paths and the constraints of delay and delay variation [7]. It takes available bandwidth as the prime metric and introduces available bandwidth into a selection function.

With respect to the k-shortest paths (KSP) problem, it has been well-studied [8-12]. Literature chose four prime KSP algorithms to compare their performance from about seventy literatures. In general, the KSP can be divided into two categories [8-11]: unconstrained KSP and constrained KSP. In the latter category, all the paths should satisfy some conditions, for instance, to be loopless.

Among all those KSP algorithms, the double sweep algorithm (DSA) is one of the effective solutions to KSP problem as long as there is no negative cycle in the graph [11]. Furthermore, the KSP using DSA can be modified to solve the k-widest paths problem [12]. That is, modify generalized minimization and generalized addition of KSP into generalized maximization and generalized minimization, respectively. Nevertheless, the k-widest paths recording cannot be attained by simply modified KSP due to the distinct difference between the additive cost and the concave bandwidth. In addition, the simply modified KSP using DSA will have loops. Consequently, this paper provides two novel definitions of path operations to record the k-widest paths. At the same time, the definitions guarantee the looplessness of the presented algorithm.

3 Notations and Definitions
Let $P^a$ be the path vector sets of all the paths from node $s$ to $t$: \{p_{s1}, p_{s2}, \ldots, p_{sm}\}, where each loopless path $p_{si}(1 \leq i \leq m)$ that is from node $s$ to $t$ is composed of no duplicate nodes, and the longest path has no more than $n$ nodes. Moreover, all $p_{si}$ are different from each other and are arranged in non-decrease order of nodes number. For instance, a path vector from node 1 to 5 is \{(1,2,5), (1,4,5), (1,2,3,4,5)\} $\in P^d$.

**Definition 1:** Let $P_X=$\{$p_{x1}, p_{x2}, \ldots, p_{xmn}$\} $\in P^x$ and $P_Y=$\{$p_{y1}, p_{y2}, \ldots, p_{yn}$\} $\in P^y$. Use $\cup$ to represent the operation of path combination:

$$P_X \cup P_Y = \text{Check} \{ \text{Comb} \{ p_{x1}, p_{y1} \} : 1 \leq i \leq m_x, 1 \leq j \leq m_y \}$$

Where Comb\{ $p_{x1}$, $p_{y1}$ \} represents to keep both paths $p_{x1}$ and $p_{y1}$ if they are unequal, otherwise, either of them is kept. Check\{ $p_{mn}$ \} will check whether each path $p_{mn}$ either has duplicate nodes or has more than $n$ nodes. If so, it will eliminate the specific path $p_{mn}$. Thus, $P_X \cup P_Y \in P^g$. If all the paths $p_{mn}$ are deleted, denote $P_X \cup P_Y$ as $\phi$. It is known that Comb\{} requires no more than (n-2) comparisons. Therefore, the operation of path combination needs $(m_x, m_y, (n-2))$ comparisons.

**Definition 2:** Let $l_t$ be the link that is from the node $t$ to its adjacent node $r$. Use $\&$ to represent the operation of path stitching:

$$P_X \& l_t = \text{Check} \{ \text{Stit} \{ p_{x1}, l_t \} : 1 \leq i \leq m_x \}$$

Where Stit\{ $p_{x1}$, $l_t$ \} creates a new path $p_{x1}$ by extending the path $p_{x1}$ from node $s$ to $t$ with the link $l_t$. Check\{ $p_{x1}$ \} will check whether each path $p_{x1}$ either has duplicate nodes or has more than $n$ nodes. If so, it will eliminate the specific path $p_{x1}$. Thus, $P_X \& l_t \in P^g$. If all the paths $p_{x1}$ are deleted, denote $P_X \& l_t$ as $\phi$. It is known that Check\{ $p_{x1}$ \} with Stit\{} requires no more than (n-3) comparisons to check whether it has duplicated nodes. Therefore, the operation of path stitching needs $(m_x, (n-3))$ comparisons.

For instance, let $P_X=$\{(1,2,5),(1,3,5),(1,2,3,4,5)\} $\in P^d$. $P_Y=$\{(1,5),(1,2,5)\} $\in P^d$, then $P_X \cup P_Y=$\{(1,5), (1,2,5), (1,3,5), (1,2,3,4,5)\} $\in P^d$. Let $l_t=(5,3)$, then $P_X \& l_t=$\{(1,2,5,3)\} $\in P^d$. $P_Y \& l_t=$\{(1,5,3), (1,2,5,3)\} $\in P^d$. Note, the paths with duplicated nodes or with node number more than $n$ have been eliminated.

**Definition 3:** Let $B^d$ be the sets of all value vectors of (b1,b2,···, bk). Each value vector has the characteristics of $b_1>b_2>\cdots>b_k$. If there are not
enough \( k \) values, supplement them with negative infinity. For example, \((8,2,1,\infty)\in B_4^I\). Furthermore, every \( b_i(1\leq i\leq k) \) has its corresponding path vector \( \{ p_{i1}, p_{i2}, \cdots, p_{im} \} \in P^m \). That is, each path \( p_{ij}^k(1\leq j \leq m) \) has equal value of \( b_i \). Note, \(-\infty\) has no corresponding path vector.

**Definition 4:** Let \( X_{ai}=(x_{ai}, x_{a2}, \cdots, x_{ak}) \in B^I \), where each \( x_{ai}(1\leq i\leq k) \) has its corresponding path vector \( p_{ai}^i = \{ p_{x_{ai}1}, p_{x_{ai}2}, \cdots, p_{x_{ai}m} \} \), \( (m_i \) is positive integer\). Let \( Y_{ai}=(y_{ai1}, y_{ai2}, \cdots, y_{aij}) \in B^I \), where each \( y_{aij} \) \((1\leq j \leq k) \) has its corresponding path vector \( p_{yi}^j = \{ p_{y_{ai1}j}, p_{y_{ai2}j}, \cdots, p_{y_{aij}j} \} \), \( (m_j \) is positive integer\). Denote \(+\) as the generalized maximization:

\[
X_{ai} + Y_{ai} = \max_{\{ x_{aij}, y_{aij} : i, j = 1,2,\cdots, k \}}
\]

Where \( \max_{\{ \}} \) computes the previous \( k \) different values in the sequence of decrease, thus \( X_{ai} + Y_{ai} \in B^I \). Furthermore, if \( x_{ai} \) and \( y_{aij} \) are identical, perform the operation of path combination \( p_{ai}^i \cup p_{yi}^j \). If there are not enough \( k \) values, supplement them with negative infinity. Note, \(-\infty\) has no corresponding paths. Due to the sequence of strict decrease, the generalized maximization of equation (3) requires no more than \( k \) comparisons and \( k \) corresponding operations of path combination.

**Definition 5:** Let \( L_{ai}=(z_{ai}, -\infty, \cdots, -\infty) \in B^I \), where \( z_{ai} \) has its corresponding link \( l_{ai} \) that is from the node \( i \) to its adjacent node \( r \). Denote \( \times \) as the generalized minimization:

\[
X_{ai} \times L_{ai} = \max_{\{ x_{aij}, z_{aij} : i=1,2,\cdots, k \}}
\]

Where \( \min_{\{ \}} \) obtains the minimum value between \( x_{aij} \) and \( z_{aij} \). Furthermore, perform the operation of path stitching \( p_{ai}^i \& l_{ai} \). If the result of path stitching is \( \Phi \), delete its corresponding \( x_{ai} \). The function of \( \max_{\{ \}} \) is the same as explained in Definition 4. Thus, \( X_{ai} \times L_{ai} \in B^I \). It is easy to know that the generalized minimization of equation (4) requires no more than \( k \) comparisons and \( k \) corresponding operations of path stitching.

For instance, Let \( X_{14}=(4,2,1,\infty) \in B^I_4 \), and the corresponding path vectors of 4, 2, and 1 are \{(1,3,4)\}, \{(1,3,2,4)\} \in P^I, \{(1,2,4)\} \in P^I \) and \{(1,2,3,4)\} \in P^I \); respectively; Let \( Y_{14}=(5,2,1,\infty) \in B^I_4 \), the corresponding path vectors of 5, 2, and 1 are \{(1,4)\} \in P^I_1, \{(1,3,2,4)\} \in P^I \) and \{(1,2,3,4)\} \in P^I \), respectively.

Then, \( X_{14} + Y_{14}=(5,4,2,1) \in B^I_4 \), and the corresponding path vectors of 5, 4, 2, and 1 are \{(4,1)\} \in P^I_1, \{(3,1,4)\} \in P^I_2, \{(1,3,2,4)\} \in P^I_3 \) and \{(1,2,3,4)\} \in P^I_4 \), respectively.

Let \( L_{42}=(3, -\infty, -\infty, -\infty) \in B^I_4 \), and the corresponding link of 3 is \( L_{42} \). Then, \( X_{44} \times L_{42}=(3,2,1, -\infty) \in B^I_4 \), the corresponding path vectors of 3, 2, and 1 are \{(1,3,4,2)\} \in P^I_1, \Phi \) and \( \Phi \) respectively. According to the Definition 5, eliminate the values of 2 and 1 with path vectors \( \Phi \). Consequently, \( X_{44} \times L_{42}=(3, -\infty, -\infty, -\infty) \in B^I_4 \), the corresponding path vector of 3 is \{(1,3,4,2)\} \in P^I_1 \).

In the following, the symbol definitions are extended.

**Definition 6:** Let \( b_{ij} = (b_{ij1}, b_{ij2}, \cdots, b_{ijk}) \in B^k \) be the \( k \) different bandwidth values of all the links from node \( i \) to its adjacent node \( j \). In consideration of the simple graph, \( b_{ij} \) represents the bandwidth value of the link from \( i \) to its adjacent node \( j \), whereas the residual bandwidth values are denoted as \(-\infty\). That is, \( b_{ij} = (b_{ij1}, -\infty, -\infty, -\infty) \in B^k_4 \). If \( i=j \), let \( b_{ij} = (-\infty, -\infty, -\infty, -\infty) \in B^k_4 \), because there is no link from node \( i \) to itself in the simple graph. Denote \( B^k_4 \) as the matrix whose element \( (i,j) = b_{ij} \); denote \( b_{ij} \) as \( (b_{i1}, b_{i2}, \cdots, b_{ik}) \).

**Definition 7:** Let \( b_{ij}^* = (b_{ij1}^*, b_{ij2}^*, \cdots, b_{ijk}^*) \in B^k \) be the \( k \) different bandwidth values of all the paths from node \( i \) to \( j \). Furthermore, they are called as optimal values. Similarly, denote \( B^k \) as the matrix whose whose element \( (i,j) = b_{ij}^* \); denote \( b_{ij}^* \) as \( (b_{i1}^*, b_{i2}^*, \cdots, b_{ik}^*) \).

**Definition 8:** The path bandwidth is equal to the least bandwidth of the links along the path. Define that bandwidth as the bottleneck bandwidth of the path.

### 4 Detailed Algorithm

There are two steps in this section. First, the novel loopless \( k \)-widest paths (KWP) algorithm is explained. Then, the novel least-delay widest-available bandwidth path (LDWABP) routing algorithm is presented.

#### 4.1 Loopless KWP Algorithm

The loopless \( k \)-widest paths (KWP) algorithm consists of two parts: (1) Compute the bandwidth values of the \( k \)-widest paths; (2) Record the paths of the \( k \)-widest paths. The first part can be derived from the modified \( k \)-shortest paths (KSP) algorithm with double sweep algorithm (DSA) [12]. That is, modify generalized minimization and generalized addition of KSP into generalized maximization and generalized minimization, respectively. Nevertheless, the second part...
part cannot be attained by the modified KSP due to the distinct difference between the additive cost and the concave bandwidth. Therefore, this paper presents two novel path operations to compute and record the k-widest paths. Furthermore, KWP is always loop-free by virtue of the path operations.

The KWP algorithm takes advantage of double sweep algorithm (DSA). The first estimates \((b_{i1}^{(0)}, \ldots, b_{in}^{(0)})\) are given from the network graph, that is, a row of matrix \(B^0\). Then gradually compute the increasing estimates of \((b_{i1}^{(h)}, \ldots, b_{in}^{(h)})\) and corresponding path vectors. For \(h \geq 1\), the KWP algorithm using DSA will terminate if the successive estimates \((b_{i1}^{(h)}, b_{i2}^{(h)}, \ldots, b_{in}^{(h)})\) and \((b_{i1}^{(h+1)}, b_{i2}^{(h+1)}, \ldots, b_{in}^{(h+1)})\) as well as their corresponding path vectors are identical. The last estimates equal to the optimal values. Note, all elements of \(b_{i1}^{(0)}\) are always \(-\infty\) because there is no need to compute the path from node 1 to itself in the case of the simple graph.

The detailed procedures of KWP can be described as follows. (Assume \(s=1\))

1. Let the first estimates \(b_{i1}^{(0)}\) be \((b_{i1}^{(0)}, b_{i2}^{(0)}, \ldots, b_{in}^{(0)})\) that is the first line of matrix \(B^0\), its corresponding path vectors are the links from node 1 to other adjacent nodes. Note, all elements of \(b_{i1}^{(0)} (\in B^0)\) are \(-\infty\).

2. For one estimate \(b_{i1}^{(2r)}\) of \(b_1^s\), the new estimates \(b_{i1}^{(2r+1)}\) and \(b_{i1}^{(2r+2)}\) can be computed by the backward sweep and the forward sweep, respectively. \((r=0,1,2,3, \ldots)\)

**Backward Sweep:**

\[
b_{i1}^{(2r+1)} = b_{i1}^{(2r)} + b_{i1}^{(2r+1)} \times L
\]

Where \(L=\{ b_{ij}^{(0)} : i \geq j \}\) is the bottom triangle of the matrix \(B^0\). Note, there is no need to compute \(b_{i1}^{(2r+1)}\).

**Forward Sweep:**

\[
b_{i1}^{(2r+2)} = b_{i1}^{(2r+1)} + b_{i1}^{(2r+2)} \times U
\]

Where \(U=\{ b_{ij}^{(0)} : i < j \}\) is the upper triangle of the matrix \(B^0\). Similarly, there is no need to compute \(b_{i1}^{(2r+2)}\). Furthermore, “+” is the generalized maximization and its corresponding operations of path combination according to Definition 4; “\(\times\)” is the generalized minimization and its corresponding operations of path stitching according to Definition 5.

3. If the successive estimates of \(b_1^h\) and \(b_1^{(h+1)} (h>0)\) and their corresponding path vectors are identical, KWP terminates; otherwise, goto 2. \(\square\)

### 4.1.1 An Illustration of KWP

An example using KWP to compute \(k\)-widest paths from node 1 to other nodes in a network (See Fig. 1) is offered in this section. Since the successive estimates (including values and corresponding path vectors) of the 3rd and 4th sweep are identical, they are the final optimal results (See Table 1). Where \(h=1, 3\) are backward sweeps and \(h=2,4\) are forward sweeps.

The matrix \(B^0\) is as follows, where \(k=4\).

\[
\begin{align*}
(1, -\infty, -\infty, -\infty) & (1, -\infty, -\infty, -\infty) & (4, -\infty, -\infty, -\infty) & (5, -\infty, -\infty, -\infty) \\
(1, -\infty, -\infty, -\infty) & (2, -\infty, -\infty, -\infty) & (3, -\infty, -\infty, -\infty) & (4, -\infty, -\infty, -\infty) \\
(1, -\infty, -\infty, -\infty) & (2, -\infty, -\infty, -\infty) & (3, -\infty, -\infty, -\infty) & (6, -\infty, -\infty, -\infty) \\
(1, -\infty, -\infty, -\infty) & (2, -\infty, -\infty, -\infty) & (3, -\infty, -\infty, -\infty) & (6, -\infty, -\infty, -\infty)
\end{align*}
\]

In Table 1, it shows the whole process and results of KWP. The row of \(h=0\) is attained directly from the \(B^0\). Next, the backward sweep starts from the \(b_{14}^{(1)}\). Then compute from \(b_{14}^{(1)}\) to \(b_{14}^{(2)}\). After that, the forward sweep begins from \(b_{12}^{(2)}\) to \(b_{12}^{(2)}\). Repeat above process till \(h=4\). Note: there is no need to compute \(b_{11}^{(h)}\).

**Table 1 Results of KWP Algorithm (k=4)**

<table>
<thead>
<tr>
<th></th>
<th>(b_{12}^{(h)})</th>
<th>(b_{13}^{(h)})</th>
<th>(b_{14}^{(h)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>h=0</td>
<td>(1, -\infty, -\infty)</td>
<td>(4, -\infty, -\infty)</td>
<td>(5, -\infty, -\infty)</td>
</tr>
<tr>
<td>path</td>
<td>142</td>
<td>143</td>
<td>144</td>
</tr>
<tr>
<td>h=1</td>
<td>(3, 2, 1, -\infty)</td>
<td>(5, 4, -\infty)</td>
<td>(5, -\infty, -\infty)</td>
</tr>
<tr>
<td>path</td>
<td>142</td>
<td>143</td>
<td>144</td>
</tr>
<tr>
<td>h=2</td>
<td>(3, 2, 1, -\infty)</td>
<td>(5, 4, 2, 1)</td>
<td>(5, 4, 2, 1)</td>
</tr>
<tr>
<td>path</td>
<td>142</td>
<td>143</td>
<td>144</td>
</tr>
<tr>
<td>h=3</td>
<td>(3, 2, 1, -\infty)</td>
<td>(5, 4, 2, 1)</td>
<td>(5, 4, 2, 1)</td>
</tr>
<tr>
<td>path</td>
<td>142</td>
<td>143</td>
<td>144</td>
</tr>
<tr>
<td>h=4</td>
<td>(3, 2, 1, -\infty)</td>
<td>(5, 4, 2, 1)</td>
<td>(5, 4, 2, 1)</td>
</tr>
</tbody>
</table>

### 4.2 LDWABP Algorithm

The least-delay widest available bandwidth path (LDWABP) algorithm is to find the least-delay path among \(k\)-widest paths from the source node to the RP. The procedures can be described as follows.

1. Initiate the source node \(s\) and RP. Execute the KWP algorithm to find the \(k\)-widest paths;
Let the delay of 1-widest paths be MIN;
If the delay of k-widest paths less than MIN, let MIN be the delay of k-widest paths;
Execute step 3 to find the least-delay path till traversing all the k-widest paths. □

5 Correctness and Complexity

Lemma 1: The optimal values $b_1^*$ of KWP algorithm can only be chosen from the bandwidth value sets of all links in a network.

Proof: By contradiction. Suppose that the optimal values $b_1^*$ may be chosen out of the bandwidth value sets of all links in a network. Thus, there exists at least a bandwidth value set of the $m^{th}$ path, from node 1 to \( t=(t=2,3,\cdots,n) \), that is chosen out of the bandwidth value set of all links in the network. Denote this link with bottleneck bandwidth as $l_{ij}$, the link from node $i$ to $j$. Then, the bandwidth of the link $l_{ij}$ will be chosen out of the bandwidth value sets of all links in the network.

However, $l_{ij}$ is one link of the network graph. Thereby it leads to contradiction. The proof is completed. □

Theorem 1: KWP algorithm can always find the optimal values and their corresponding path vectors in the finite steps. Furthermore, it totally needs $n$ single sweeps.

Proof: From equation (5), it is known to compute the generalized maximization between $b_{ij}^{(2^t)}$ and $b_{ij}^{(2^t+1)} \times b_{ij}^{(0)} (t=n-1,n-2,\cdots,2; i=t+1,t+2,\cdots,n)$; Similarly, equation (6) are to find the generalized maximization between $b_{ij}^{(2^t)}$ and $b_{ij}^{(2^t+1)} \times b_{ij}^{(0)} (t=3,4,\cdots,n; i=2,3,\cdots,t-1)$. Thereby, there is no value can be replaced by a smaller value in the next iterative steps. That is, the sequences \{ $b_{ij}^{(0)}, b_{ij}^{(1)}, \cdots, b_{ij}^{(h)}$ \} are non-decrease. Furthermore, according to Lemma 1, the optimal values are chosen from the finite bandwidth value sets of all links in the network, KWP algorithm will always converge to the optimal values and their corresponding path vectors in finite steps.

In addition, it is shown from equation (4) that every operation of generalized minimization will operate the path stitching. That is, the generalized minimization either creates the new bandwidth values and corresponding path vectors, or only creates new paths. Then those bandwidth values and their path vectors are combined by equation (3). As to a network graph with $n$ nodes, there are $c=(1+P_{n-2}^1+P_{n-2}^2+\cdots+P_{n-2}^n)$ total paths from node 1 to node $t$, where $P_{n-2}^m$ is the permutation of $m$ nodes chosen from $(n-2)$ nodes. In essence, the process of finding $b_{ij}^{(h)}$ using double sweep algorithm is to traverse all the paths mentioned above.

On the other hand, due to the specific directions of forward sweep and backward sweep, if $n$ is odd, $b_{ij}^{(n-1)}$ is the last one that finishes the traversing (See Fig. 2(a)); if $n$ is even, $b_{ij}^{(n-1)}$ is the last one that finishes the traversing (See Fig. 2(b)). In the case of completed simple graph, it needs $(n-1)$ single sweeps to complete traversing all the paths, i.e. (n-2) permutation. On account of the step 3 of KWP, there needs one more sweep to terminate the algorithm. Therefore, the KWP totally needs $(n-1)+1=n$ single sweeps to find the optimal values and corresponding path vectors. The proof is completed. □

Fig. 2 Double Sweeps in KWP algorithm
(Note: there is no need to compute $b_{ij}^{(0)}$)

Theorem 2: KWP algorithm is always loop-free.

Proof: According to the path definitions of equations (1) and (2), if there are duplicate nodes in a path, the path will be deleted by Check[] due to the specific directions of forward sweep and backward sweep. Due to equations (3) and (4), both generalized maximization and generalized minimization have path operation of Check[]. Thus, there are no duplicate nodes in any paths. Therefore, the KWP algorithm is always loop-free. The proof is completed. □

Theorem 3: The time complexity of KWP is $O(n^3)$.

Proof: Since there is no need to compute $b_{ij}^{(h)}$, it is known from equation (5) that the one backward sweep needs $O((n-1)(n-2)/2)$ generalized maximization and corresponding operations of path combination, and $O((n-1)(n-2)/2)$ generalized minimization and corresponding operations of path stitching. Similarly, it is shown from equation (6) that one forward sweep needs $O((n-1)(n-2)/2)$
generalized maximization and corresponding operations of path combination, and $o(n(n-1)(n-2)/2)$ generalized minimization and corresponding operations of path stitching.

In addition, the generalized maximization needs no more than $k$ comparisons and $k$ operations of path combination according to equation (3); the generalized minimization needs no more than $k$ minimization comparisons and $k$ operations of path stitching according to equation (4). By virtue of Theorem 1, there need $n$ single sweeps to traverse all the paths. Consequently, KWP algorithm has time complexity of $o(n(n-1)(n-2)/2 + k(n(n-1)(n-2)/2)) = o(kn(n-1)(n-2))$, where $k$ represents the number of the $k$-widest paths. The proof is completed. □

**Theorem 4:** The time complexity of LDWABP algorithm is $o(kn^3)$.

**Proof:** The LDWABP algorithm is to find the least-delay path among the $k$-widest paths. While the maximum number of the paths between two nodes is mentioned as the constant $c=(1 + P_{n-2}^1 + P_{n-2}^2 + \cdots + P_{n-2}^{n-2})$. In addition to Theorem 3, the time complexity of LDWABP is $o(kn^3)$. The proof is completed. □

**Theorem 5:** LDWABP is always loop-free.

**Proof:** LDWABP is to find the least-delay path among the $k$-widest paths. According to Theorem 2, the LDWABP is always loop-free. The proof is completed. □

6 Conclusions

This paper first presents the novel loopless $k$-widest paths (KWP) algorithm that has essential difference with $k$-shortest paths (KSP) algorithm with regard to the path recording. The KWP takes advantage of the two newly defined path operations to record the $k$-widest paths during the double sweep processing. To the best knowledge of the author, it is the first time to propose the detailed KWP algorithm.

Based on the KWP algorithm, a novel least-delay widest-available bandwidth path (LDWABP) algorithm is proposed to connect the source node and the RP in our project of modified PIMSM in group multicast. Next, the detailed procedures of the algorithm are provided as well. Finally, the correctness and looplessness are proved as well as its polynomial time complexity.

The future work focuses on the RP placement of PIMSM in the group multicast.

References: