Computer Engineering and Intelligent Systems www.iiste.org
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol 2, No.5, 2011

Logical Topology Design Using Efficient Heuristics in Wavelength Routed Networks

Y. Rama Mohan

Department of Computer Science & Engineering, G.Pulla Reddy Engineering College

Kurnool-518007, Andhra Pradesh. India.

E-mail: yekkanti@gmail.com

V. Raghunatha Reddy

Department of Computer Science & Technology, Sri Krishnadevaraya University

Anantapur, Andhra Pradesh, India.

E-mail: vraghu9@gmail.com

Abstract

Wavelength Division Multiplexed (WDM) point to point networks play a vital role in the backbone transport networks. The set of light paths at optical layer forms a Logical Topology. This paper deals with the design of Logical Topology for wavelength routed WDM networks. This paper proposes new heuristics on fiber optic networks to develop efficient logical topology design and to examine the critical aspects of performance constraints like single hop traffic maximization, Average weighted hop count and number of wavelengths/Transceivers. Further two new heuristics LUMHSN and ILUMHSN are proposed, tested and compared the performances with the existing HLDA on 14-node NSFNET Model.

Keywords: wavelength routed WDM, Logical Topology, single hop traffic, Average weighted hop Count, LUMHSN, ILUMHSN.

1. Introduction

Wavelength-division multiplexing (WDM) networks are believed to be a promising candidate to meet the explosive increase of bandwidth demand in the Internet. In wavelength routed optical networks, a virtual topology is overlaid on the physical network that minimizes the electro optical conversion of the traffic flow. This paper deals with the logical topology design for wavelength routed WDM Networks proposed by Rajiv Ramaswami and Kumar N.Sivarajan(2002), B.Mukherjee, (1997) and C.siva Ram Murthy and Mohan Gurusamy,(2002) using efficient heuristics as proposed in the following sections. Several heuristic solutions for logical topology design problems are available as proposed by R.Ramaswami and A.Segall,(1996) in the literature.

The proposed heuristics are implemented in a standard 14-node NSFNET model. The various performance measures obtained over the network models for the different proposed heuristics: 1) Lightpath Utility Maximization Heuristic based on Source Nodes 2) Iterative Lightpath Utility Maximization Heuristic based on Source Nodes are compared with the existing Heuristic Logical Topology Design Algorithm. Some possible objective functions for the virtual topology design problem are given below.

(i) Minimizing average weighted number of hops**:** If t_{total} is the total amount of offered traffic onto the network, then the average weighted number of (virtual) hops, denoted by $h_{ave,}$ is computed as

$$h_{ave} = \frac{1}{t_{total}} \sum_{sd} h_{sd} X^{sd} \qquad (1)$$

The objective function is given by Minimize h_{ave}

(ii) Minimizing network congestion**.** The maximum flow on any lightpath in a network is called network congestion. It is denoted by f_{ave} is computed as

$$f_{\max} = \max_{i,j} f_{ij} \qquad (2)$$

The objective function is given by Minimize $f_{\max}$

(iii) Maximizing single-hop traffic**.** The amount of traffic that is carried in one (virtual) hop, denoted as $t_{single,}$ is computed as

$$t_{single} = \sum_{s,d} p_{sd} X^{sd} t \qquad (3)$$

The objective function is given by Maximize t_{single}

The heuristic logical topology design algorithm (HLDA) and the proposed Heuristics for the design of logical topology are presented in the following sections.

1.1 Heuristic Logical Topology Design Algorithm (HLDA):

The heuristic logical topology design algorithm (HLDA) heuristic proposed to minimize congestion in a given network. The traffic matrix and the physical topology are taken as the input. The number of transmitters and receivers available at every node is assumed to be given. Also, the number of available wavelengths per fiber is assumed to be fixed. This algorithm attempts to maximize single. (Virtual)-hop traffic flow.

The HLDA propsed by Bala Rajagopalan et al.,(2000) considers node pairs in non increasing order of their traffic. It selects the node pair (say, x) with the most nonzero traffic flow between them. A lightpath is established between this node pair, if permissible. A lightpath is permissible for node pair x is a physical route, a wavelength on the route, a transmitter at the source node of x, and a receiver at the destination node of x are all available. When a lightpath is established between pair x, the traffic associated with x is updated by subtracting from it the traffic associated with pair y. Here, node pair y has the highest traffic after pair x. If a lightpath cannot be established between node pair x, the traffic associated with it is set to zero. Now, the node pair which has the maximum amount of nonzero traffic is chosen and the above procedure is repeated. Note that the chosen node pair could be either x or y. When all the node pairs with nonzero traffic have been considered, the procedure stops. It may so happen that a few transmitters and receivers are available at some nodes when the procedure terminates. The HLDA creates lightpahts between such nodes to exhaust the available transmitters and receivers.

Pseudo code for HLDA:

```
HeuristicBasedMethod (ulTransmitterCount, ulReceiverCount)
{
/* Initialize the traffic values and the transmitter and reciever count at each node */
for each physical node (ucRow)
{for each physical node (ucColumn)
{ set ucLoc to (ucRow * MAX_PHY_NODE) + ucColumn
check if (ucRow is equal to ucColumn)
```

```
            { set pointer of (ppucTraffic + ucLoc) to 0 } else
            {set *(ppucTraffic + ucLoc) to aucVPathCost[ucRow][ucColumn]}}
             set aucTransmitters[ucRow] to ulTransmitterCount
             set aucReceivers[ucRow] to ulReceiverCount
        increment ucAvailTransmitter by aucTransmitters[ucRow]
        increment ucAvailReceiver by aucReceivers[ucRow] }
    while (1)
{ /* Break the loop if no transmitters or receivers are available */
check if ((ucAvailTransmitter <= ucAllocTransmitter) || (ucAvailReceiver <= ucAllocReceiver))
        {break}
 /* Get the maximum traffic pair from the 2d array */
call the API GetMaxTrafficPair (&MaxRow, &MaxCol, &SMaxRow, &SMaxCol) to get the first and
second max traffic value vertices
set ucLoc to (MaxRow * MAX_PHY_NODE) + MaxCol
/* Check if there are no transmitters on the sender node and no receivers on the receiving node */
if ((aucTransmitters[MaxRow] is equal to 0) || (aucReceivers[MaxCol] is equal to 0))
        {*(ppucTraffic + ucLoc) = 0;
         /* Check if any traffic node exists in the array */
        for each physical node (ucRow)
        { for each physical node (ucColumn)
        { set ucLoc to (ucRow * MAX_PHY_NODE) + ucColumn
        increment ulSum by *(ppucTraffic + ucLoc) }}
        check if (0 is equal to ulSum)
        { break} set ulSum to 0    continue }
decrement *(ppucTraffic + ucLoc) by *(ppucTraffic + ((SMaxRow * MAX_PHY_NODE) + SMaxCol))
        decrement aucTransmitters[MaxRow] by 1
        decrement aucReceivers[MaxCol] by 1
        increment ucAllocTransmitter by 1
        increment ucAllocReceiver by 1
call the API AddVirtualPath (ucColumn, ucRow) to add the virtual path informatio to the global list}
call the API DisplayVirtualPath (ulTransmitterCount) to display the virtual path information
call the API GetPrintHeavyTrafficLink () to display the heavily loaded link information
call the API GetPrintLowTrafficLink () to display the least loaded link information
    /* Release the allocated memory */
    while (NULL != pstVPHead)
    {   pstTmpNode = pstVPHead->pstNext;
        free (pstVPHead);
        pstVPHead = pstTmpNode; }
        return SUCCESS; }
```

2. Proposed Heuristics

2.1 Lightpath Utility Maximization Heuristic Based On Source Node (LUMHSN):

The Heuristic proposed here is a lightpath Utilization with respect to Source node. In this heuristic the selection of lightpath from source to destination is based on the total cost of source node and depends upon the number of transceivers placed at each node. For example, if there are 2-transceivers select only two maximum lightpaths with respect to the source node. The selection of lightpath with respect to the source node will be continued until the scope of transmitters and receivers exhausted or utilized at each node.

The input to the heuristic is traffic matrix (arbitrarily chosen) for given physical topology network model. The procedure of selection of virtual lightpath based on the criteria of arranging the total source cost with respect to each row in the traffic matrix in ascending order and then selecting the row with maximum cost. This procedure is iteratively repeated until the scope of transceivers is exhausted or utilized at each node.

Pseudo Code for LUMHSN:

```
Method1_SRCNode (ulTransmitterCount, ulReceiverCount)
{
    /* Initialize the traffic values and the transmitter and reciever count at each node */
    for each physical node (ucRow)
    { for each physical node (ucColumn)
            { set ucLoc to (ucRow * MAX_PHY_NODE) + ucColumn
            check if (ucRow is equal to ucColumn)
            {set pointer of (ppucTraffic + ucLoc) to 0 }else
            {set    pointer of (ppucTraffic + ucLoc) to aucVPathCost[ucRow][ucColumn] }}
    /* Initialize the transmitter and receiver info of the physical node */
        set aucTransmitters[ucRow] to ulTransmitterCount
        set aucReceivers[ucRow] to ulReceiverCount }
    for each physical node (ucRow)
    {for each physical node (ucColumn)
    {increment ulRowSum by aucVPathCost[ucRow][ucColumn] }
        set aulRowSum[ucRow] to ulRowSum
        set ulRowSum to 0
    } for each physical node (ucIter)
    { call the API GetMaxVal (aulRowSum) to get the max traffic sum row
    check if (0 is equal to aulRowSum[ucRow])
    {continue}for each physical node (ucIter1)
    { /* Check if transmitters of the row are non zero */
    check if (0 is equal to aucTransmitters[ucRow])
    { break }call the API GetMaxColVal (ppucTraffic, ucRow) to get the column (ucColumn)
    having    highest traffic value from the row (ucRow)
    set ucLoc to (ucRow * MAX_PHY_NODE) + ucColumn
    check    if (0 is equal to pointer of (ppucTraffic + ucLoc))
    {continue} set pointer of (ppucTraffic + ucLoc) to 0
    /* Check if receivers of column are non zero */
    check if (0 is equal to aucReceivers[ucColumn])
    {continue } /* Add virtual path */
    call the API AddVirtualPath (ucColumn, ucRow) to add the virtual path informatio to the global list
            decrement aucTransmitters[ucColumn] by 1
            decrement aucReceivers[ucRow] by 1
    }aulRowSum[ucRow] = 0; }
    call the API DisplayVirtualPath (ulTransmitterCount) to display the virtual path information
    call the API GetPrintHeavyTrafficLink () to display the heavily loaded link information
    call the API GetPrintLowTrafficLink () to display the least loaded link information
    /* Release the allocated memory */
    while (NULL != pstVPHead)
    {    pstTmpNode = pstVPHead->pstNext;
    free (pstVPHead);
    pstVPHead = pstTmpNode;
    } return SUCCESS ;}
```

2.2 Iterative Lightpath Utility Maximization Heuristic Based on Source Node (ILUMHSN)

The Heuristic proposed here is based on Iterative Lightpath Utilisation with respect to Source node. In this heuristic the selection of one lightpath is from source to destination is performed on the maximum total cost of source node and the next lightpath is from source to destination is based on the next total cost of source node, this will be continued to complete one cycle i.e., until all the source nodes total traffic cost are processed the same procedure is implemented repeatedly and it depends upon the number of transceivers fixed at each node. For example, if there are 2-transceivers, select only one maximum lightpath with respect to the source node and another maximum lightpath with respect to other source node are processed. This procedure is repeated until all the transceivers at all nodes are processed. However the iterative lightpath selection is subject to the condition that the source nodes will be processed until the scope of transmitters and receivers exhausted or utilized at each node.

The input to the heuristic is traffic matrix (arbitrarily chosen) for given physical topology network model. The procedure of selection of Virtual lightpath based on the criteria of arranging the total source cost with respect to each row in the traffic matrix in ascending order and then selecting the row with maximum cost. This procedure is iteratively repeated until the scope of transceivers is exhausted or utilized at each node

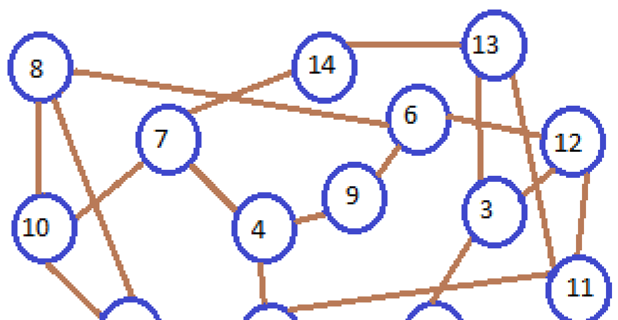
Pseudo Code for ILUMHSN:

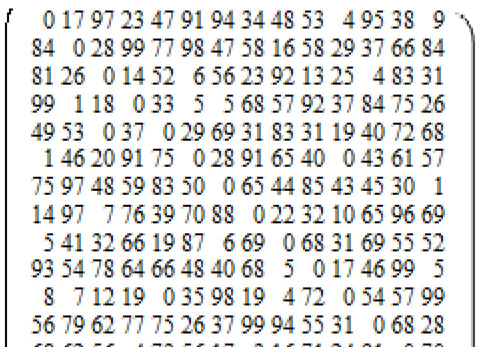
```
Method_IterativeSouce( ulTransmitterCount, ULONG ulReceiverCount)
{
    /* Initialize the traffic values and the transmitter and reciever count at each node */
    for each physical node (ucRow)
    {for each physical node (ucColumn)
    {set ucLoc to (ucRow * MAX_PHY_NODE) + ucColumn
            check if (ucRow is equal to ucColumn)
            { set *(ppucTraffic + ucLoc) to 0 } else
            { set *(ppucTraffic + ucLoc) to aucVPathCost[ucRow][ucColumn]}}
    /* Initialize the transmitter and receiver info of the physical node */
    set aucTransmitters[ucRow] to ulTransmitterCount
    set aucReceivers[ucRow] to ulReceiverCount
    increment ucAvailTransmitter by aucTransmitters[ucRow]
    increment ucAvailReceiver by aucReceivers[ucRow]    }
    for each physical node (ucRow)      { for each physical node (ucColumn)
    { increment ulRowSum by aucVPathCost[ucRow][ucColumn]       }
    set aulRowSum[ucRow] to ulRowSum
    set aucRowOrder[ucRow] to ucRow
    set ulRowSum to 0    }
    /* Sort the array and update the elements with the appropriate row number */
    call the API GetSortedElmOrder (aulRowSum, MAX_PHY_NODE, aucRowOrder) to sort the column
    sum values
    set ucIter to 0
    while (1){ /* Break the loop if no transmitters or receivers are available */
    check if ((ucAvailTransmitter <= ucAllocTransmitter) || (ucAvailReceiver <= ucAllocReceiver))
        { break }
        /* Check if every node is reacheable from every other node */
        set ulSum to 0
        /* Check if any traffic node exists in the array */
        for each physical node (ucIter1)
        { for each physical node (ucIter2){ set ucLoc to (ucIter1 * MAX_PHY_NODE) + ucIter2
                increment ulSum by   pointer (ppucTraffic + ucLoc) } }
        check if (0 is equal to ulSum)
        { break }
        set ucLoc to (ucIter % MAX_PHY_NODE)
        set ucRow to aucRowOrder[ucLoc]
        /* Check if transmitters of the row are non zero */
        check if (0 is equal to aucTransmitters[ucRow])
        {   increment ucIter by 1 continue    }
  call the API GetMaxColVal (ppucTraffic, ucRow) to get the column (ucColumn) having highest
  traffic matrix
```

```
            from the specified row (ucRow)
            set ucLoc to (ucRow * MAX_PHY_NODE) + ucColumn
            check if (0 is equal to *(ppucTraffic + ucLoc))
            { /* If there is no more traffic left from the current row, then mark that
            all the receivers and transmitters of the row node are engaged */
            increment ucAllocTransmitter by aucTransmitters[ucRow]
            increment ucAllocReceiver by aucReceivers[ucRow]
            increment ucIter by 1 continue    }
            set pointer of (ppucTraffic + ucLoc) to 0
            /* Check if receivers of column are non zero */
            check if (0 is equal to aucReceivers[ucColumn])
            {      increment ucIter by 1 continue    }
call the API AddVirtualPath (ucColumn, ucRow) to add the virtual path informatio to the global list
            decrement aucTransmitters[ucColumn] by 1
            decrement aucReceivers[ucRow] by 1
            increment ucAllocTransmitter by 1
            increment ucAllocReceiver by 1
            increment ucIter by 1    }
call the API DisplayVirtualPath (ulTransmitterCount) to display the virtual path information
call the API GetPrintHeavyTrafficLink () to display the heavily loaded link information
call the API GetPrintLowTrafficLink () to display the least loaded link information
      /* Release the allocated memory */
      while (NULL != pstVPHead)
      {      pstTmpNode = pstVPHead->pstNext;
            free (pstVPHead);
            pstVPHead = pstTmpNode;      }
      return SUCCESS; }
```

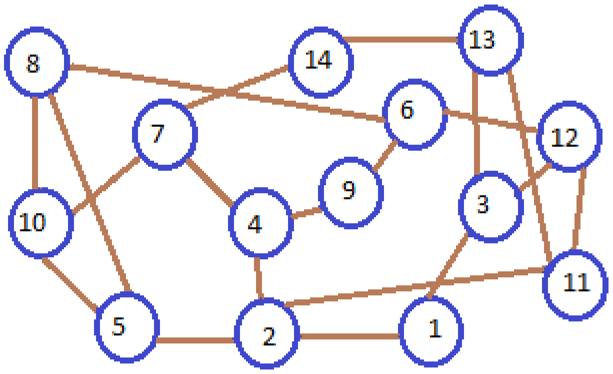
3. Results and Discussions

The Heuristics proposed are implemented on 14-node NSFNET MODEL. The above proposed heuristics LUMHSN, ILUMHSN are compared with existing HLDA. The critical comparison and performance analysis takes place with various Objective Functions like T-single, Lightpath, Wavelength, Average Weighted Hop Count, Maximum congestion and Minimum congestion.





$$\begin{pmatrix}
0 & 17 & 97 & 23 & 47 & 91 & 94 & 34 & 48 & 53 & 4 & 95 & 38 & 9 \\
84 & 0 & 28 & 99 & 77 & 98 & 47 & 58 & 16 & 58 & 29 & 37 & 66 & 84 \\
81 & 26 & 0 & 14 & 52 & 6 & 56 & 23 & 92 & 13 & 25 & 4 & 83 & 31 \\
99 & 1 & 18 & 0 & 33 & 5 & 5 & 68 & 57 & 92 & 37 & 84 & 75 & 26 \\
49 & 53 & 0 & 37 & 0 & 29 & 69 & 31 & 83 & 31 & 19 & 40 & 72 & 68 \\
1 & 46 & 20 & 91 & 75 & 0 & 28 & 91 & 65 & 40 & 0 & 43 & 61 & 57 \\
75 & 97 & 48 & 59 & 83 & 50 & 0 & 65 & 44 & 85 & 43 & 45 & 30 & 1 \\
14 & 97 & 7 & 76 & 39 & 70 & 88 & 0 & 22 & 32 & 10 & 65 & 96 & 69 \\
5 & 41 & 32 & 66 & 19 & 87 & 6 & 69 & 0 & 68 & 31 & 69 & 55 & 52 \\
93 & 54 & 78 & 64 & 66 & 48 & 40 & 68 & 5 & 0 & 17 & 46 & 99 & 5 \\
8 & 7 & 12 & 19 & 0 & 35 & 98 & 19 & 4 & 72 & 0 & 54 & 57 & 99 \\
56 & 79 & 62 & 77 & 75 & 26 & 37 & 99 & 94 & 55 & 31 & 0 & 68 & 28
\end{pmatrix}$$



$$\begin{pmatrix}
0 & 17 & 97 & 23 & 47 & 91 & 94 & 34 & 48 & 53 & 4 & 95 & 38 & 9 \\
84 & 0 & 28 & 99 & 77 & 98 & 47 & 58 & 16 & 58 & 29 & 37 & 66 & 84 \\
81 & 26 & 0 & 14 & 52 & 6 & 56 & 23 & 92 & 13 & 25 & 4 & 83 & 31 \\
99 & 1 & 18 & 0 & 33 & 5 & 5 & 68 & 57 & 92 & 37 & 84 & 75 & 26 \\
49 & 53 & 0 & 37 & 0 & 29 & 69 & 31 & 83 & 31 & 19 & 40 & 72 & 68 \\
1 & 46 & 20 & 91 & 75 & 0 & 28 & 91 & 65 & 40 & 0 & 43 & 61 & 57 \\
75 & 97 & 48 & 59 & 83 & 50 & 0 & 65 & 44 & 85 & 43 & 45 & 30 & 1 \\
14 & 97 & 7 & 76 & 39 & 70 & 88 & 0 & 22 & 32 & 10 & 65 & 96 & 69 \\
5 & 41 & 32 & 66 & 19 & 87 & 6 & 69 & 0 & 68 & 31 & 69 & 55 & 52 \\
93 & 54 & 78 & 64 & 66 & 48 & 40 & 68 & 5 & 0 & 17 & 46 & 99 & 5 \\
8 & 7 & 12 & 19 & 0 & 35 & 98 & 19 & 4 & 72 & 0 & 54 & 57 & 99 \\
56 & 79 & 62 & 77 & 75 & 26 & 37 & 99 & 94 & 55 & 31 & 0 & 68 & 28 \\
68 & 62 & 56 & 4 & 73 & 56 & 17 & 3 & 16 & 71 & 24 & 91 & 0 & 70 \\
84 & 1 & 33 & 69 & 13 & 58 & 39 & 3 & 67 & 21 & 24 & 75 & 44 & 0
\end{pmatrix}$$

Figure 1: 14-Node NSFNET Backbone Network with Traffic Matrix

Table 1: Heuristic Logical Topology Design Algorithm (HLDA) Results on 14-NSFNET

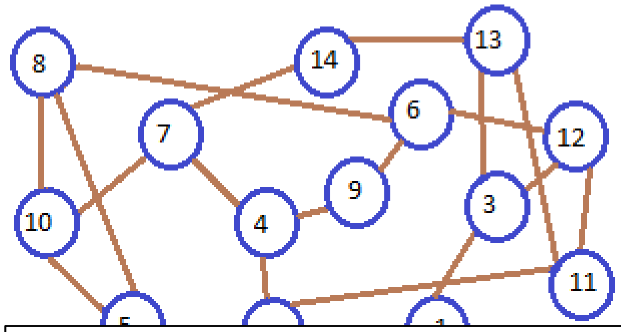
Nodes	T-Single	TRANS	Lightpath	Wavelength	Physical Hops	Hop Weight	Total Hop Weight	Average Weighted Hop Count h_{avg}	Maximum Congestion	Minimum Congestion
14	459	2	28	5	68	2339	5653	2.417	9->6(529)	11 ->13(82)
14	696	3	42	7	101	3407	8216	2.412	3->1(737)	12 ->13(70)
14	892	4	54	9	134	4308	10663	2.475	1->4(939)	9 ->7(68)
14	1132	5	69	11	168	5193	12692	2.444	1->4(1121)	9 ->7(68)
14	1543	6	82	11	188	5923	13894	2.346	1->4(1229)	9 ->4(66)
14	1585	7	96	13	228	6758	16185	2.395	1->4(1473)	9 ->4(66)
14	1676	8	108	15	256	7291	17377	2.383	1->4(1800)	9 ->4(66)
14	1795	9	126	17	298	7894	18805	2.382	1->4(1946)	9 ->4(66)
14	1913	10	139	17	321	8253	19429	2.354	1->4(2024)	9 ->4(66)

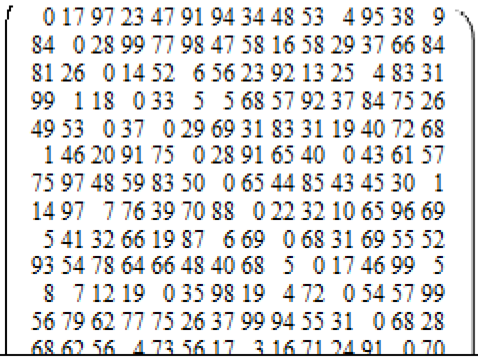
Table2: Lightpath utility Maximization Heuristic based on Source Node (LUMHSN) Results on 14-NSFNET

NOdes	T-Single	TRANS	Lightpath	Wavelength	Physical Hops	Hop Weight	Total Hop Weight	Average Weighted Hop Count h_{avg}	Maximum Congestion	Minimum Congestion
14	498	2	26	4	62	2132	5138	2.410	9->6(457)	11 ->13(32)
14	609	3	39	7	97	3181	7935	2.494	9->6(786)	13 ->11(81)
14	785	4	53	9	133	4031	10181	2.526	9->6(989)	9 ->7(68)
14	1117	5	66	9	157	4889	11712	2.396	6->3(937)	9 ->4(66)
14	1248	6	78	10	185	5510	13261	2.407	1->4(1131)	9 ->4(66)
14	1463	7	92	12	219	6303	15221	2.415	1->4(1333)	9 ->4(66)
14	1496	8	104	13	248	6812	16389	2.406	1->4(1441)	9 ->4(66)
14	1564	9	117	16	277	7419	17857	2.407	1->4(1790)	9 ->4(66)
14	1686	10	133	18	315	7784	18585	2.388	1->4(2125)	9 ->4(66)

Table3: Iterative Lightpath utility Maximisation Heuristic based on Source Node (ILUMHSN) Results on 14-NSFNET

Nodes	T-Single	TRANS	Lightpath	Wavelength	Physical Hops	Hop Weight	Total Hop Weight	Average Weighted Hop Count h_{avg}	Maximum Congestion	Minimum Congestion
14	459	2	28	5	68	2350	5705	2.428	10->1(639)	6 ->3(60)
14	534	3	41	7	105	3363	8618	2.563	9->6(787)	13 ->11(75)
14	974	4	54	9	132	4279	10402	2.431	10->1(977)	9 ->7(68)
14	1185	5	69	10	166	5181	12484	2.410	9->6(1098)	12 ->13(70)
14	1614	6	82	11	189	5998	14057	2.344	1->4(1246)	9 ->4(66)
14	1701	7	97	13	230	6660	15862	2.382	1->4(1581)	9 ->4(66)
14	1725	8	111	16	265	7267	17409	2.396	1->4(1936)	9 ->4(66)
14	1836	9	123	17	290	7759	18408	2.372	1->4(2061)	9 ->4(66)





0	17	97	23	47	91	94	34	48	53	4	95	38	9
84	0	28	99	77	98	47	58	16	58	29	37	66	84
81	26	0	14	52	6	56	23	92	13	25	4	83	31
99	1	18	0	33	5	5	68	57	92	37	84	75	26
49	53	0	37	0	29	69	31	83	31	19	40	72	68
1	46	20	91	75	0	28	91	65	40	0	43	61	57
75	97	48	59	83	50	0	65	44	85	43	45	30	1
14	97	7	76	39	70	88	0	22	32	10	65	96	69
5	41	32	66	19	87	6	69	0	68	31	69	55	52
93	54	78	64	66	48	40	68	5	0	17	46	99	5
8	7	12	19	0	35	98	19	4	72	0	54	57	99
56	79	62	77	75	26	37	99	94	55	31	0	68	28
68	62	56	4	73	56	17	3	16	71	24	91	0	70

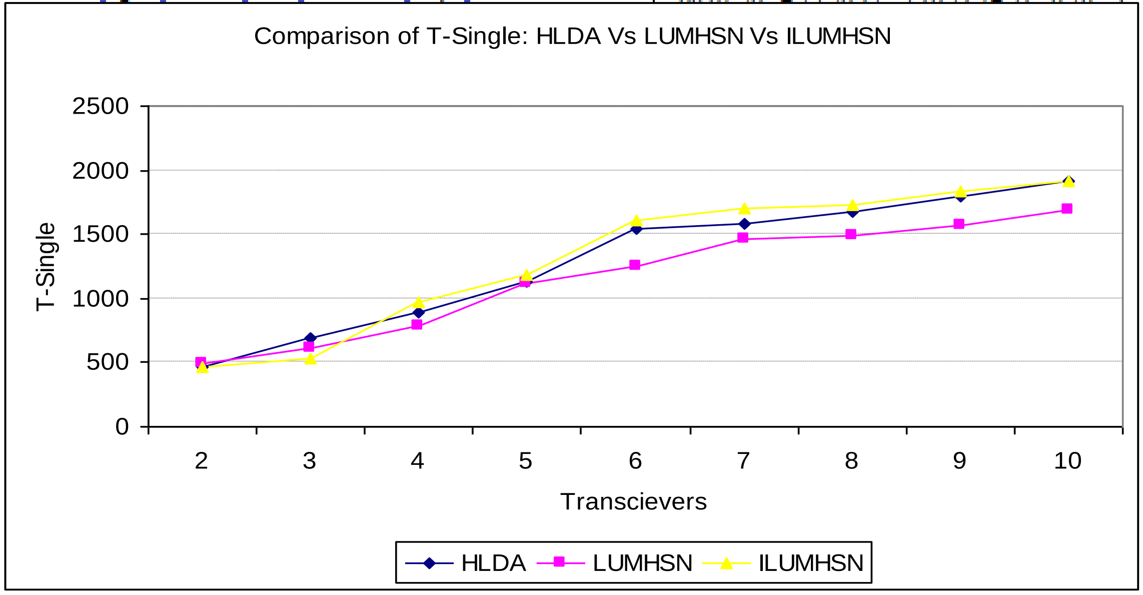


Figure 2: Comparison of T-Single of HLDA, LUMHSN, ILUMHSN

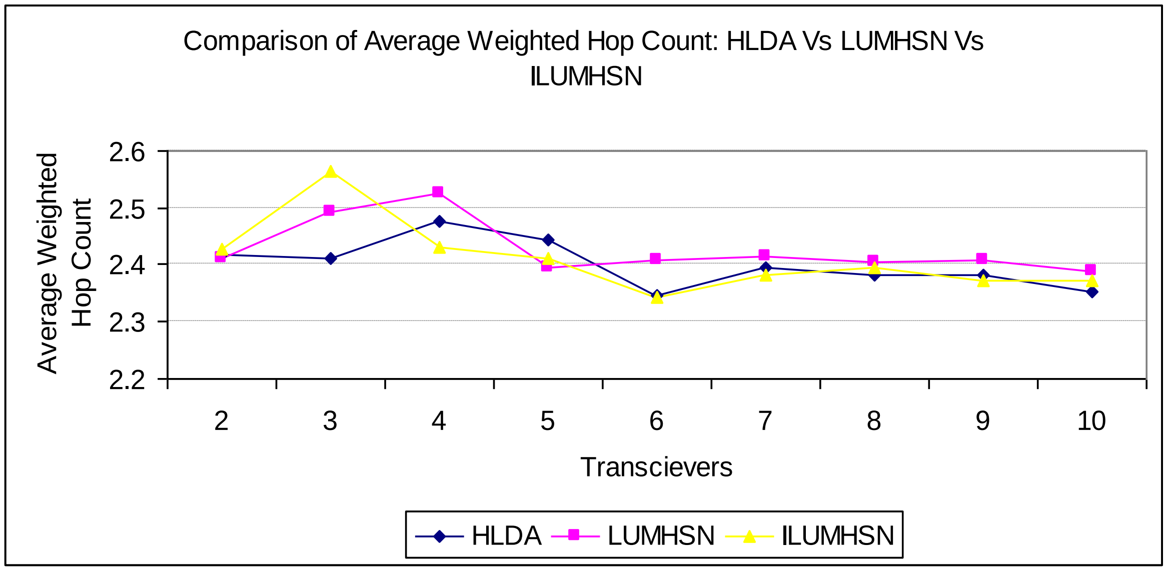


Figure 3: Comparison of Average Weighted Hop Count of HLDA, LUMHSN, ILUMHSN

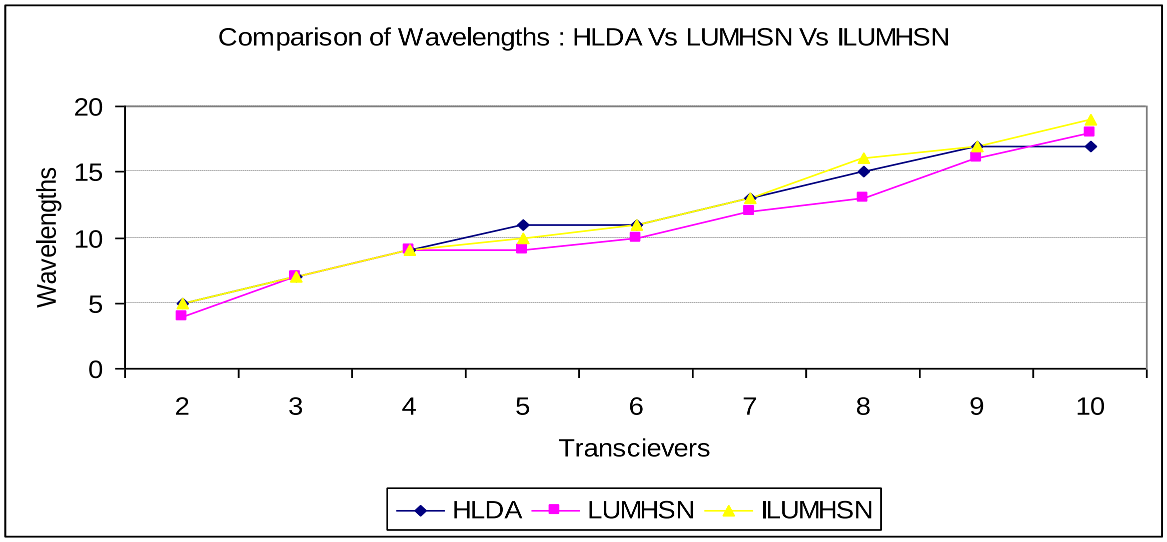


Figure 4: Comparison of Number of Wavelengths of HLDA, LUMHSN, ILUMHSN.

4. Conclusion

On comparison of the results obtained and presented in the tables 1, 2, and 3 the following observations on the existing heuristics i.e. HLDA and the proposed heuristics in this paper i.e. LUMHSN, ILUMHSN.

a) The algorithmic complexity is simple with respect to all the three heuristics.

b) The average weighted hop length is found to be slightly varying for different wavelengths with respect to all the three heuristics. However, the existing heuristic HLDA gives better results when compared to LUMHSN but ILUMHSN is better when the number of transceivers is less than four and all the three heuristics converges when the number of transceivers enhanced.

c) For the Single Hop Maximisation (T-single) ILUMHSN is performing better than both HLDA and LUMHSN.

d) In the case of number of wavelengths utilized it is observed that both the proposed heuristics LUMHSN and ILUMHSN are performing better than existing heuristics namely HLDA.

e) Hence it is concluded that the proposed heuristics namely Lightpath Utilization Maximum heuristics based of Source Node(LUMHSN) and Iterative Lightpath Utilization Maximum Heuristics based on source Node(ILUMHSN) are competent heuristic algorithms for the Virtual Topology design in WDM networks.

5. References

Rajiv Ramaswami and Kumar N.Sivarajan,(2002), "*Optical Networks – A Practical Perspective*", Second Edition, Moran Kaufmann Publishers, San Francisco.

B.Mukherjee, (1997), "Optical Communication Networks", McGraw Hill, New York .

C.siva Ram Murthy and Mohan Gurusamy, ,(2002), "WDM optical Networks: concepts Design, And Algorithms", Prentice Hall of India.

R.Ramaswami and A.Segall,(1996), " Distributed Network Control for Wavelength Routed Optical Networks, "*Proc. of IEEE INFOCOM'96*.

Bala Rajagopalan et al.,(2000), "IP Over Optical Networks: Architectural Aspects, "*IEEE Communications Magazine,* 38(9), 94-102.