Node Placement Optimization Techniques in Multihop Lightwave Based de Bruijn Graph Network

Tarun Kumar Ghosh[1], Debi Bera[2]

[1]Department of Computer Science & Engineering
Haldia Institute of Technology, West Bengal-721657, INDIA
[2] Department of Computer Science & Engineering
Haldia Institute of Technology, West Bengal-721657, INDIA

ABSTRACT
The de Bruijn graph being a regular topology and having structured node connectivity has a small nodal degree. Node placement problem in de Bruijn graph is a combinatorial optimization problem. To exploit the limitless capabilities of lightwave technology, we construct optimized regular multihop network based de Bruijn graph when the traffic flow among the network nodes is asymmetric. Given that the network nodes must be connected in a regular interconnection pattern and that the node positions in the regular network can be adjusted by properly tuning their (optical) transceivers, here we propose the best possible node placement in the given regular topology. We formulate four efficient heuristic algorithms to design optimized de Bruijn graph structures for given traffic matrices and compare these algorithms.

Keywords:
Lightwave networks, node placement problem, multihop, de Bruijn graph, optimized structures.

I. INTRODUCTION

Life in our increasingly information-dependent society requires multimedia services such as real-time voice, video, high-resolution graphics, distributed databases, distributed computing among high-performance systems, etc. These applications require fast delivery of high volume of traffic over a large area. To satisfy these demanding needs, fiber-optic medium, which offers a very high bandwidth-distance product, is commonly chosen as the transmission medium.

The huge (nearly 50 Tbps) and inexpensive bandwidth of the optical medium promise the potential for new services and capabilities. However, the ability of a user to access this huge bandwidth is constrained by the much slower electronic processing speed of its channel interface.

A lightwave network can be constructed by exploiting the capabilities of optical technology, viz., WDM and tunable optical transceivers (transmitters and receivers) as follows: the vast optical bandwidth of a fiber is carved up into smaller capacity channels, each of which can operate at peak electronic processing speeds (viz., over a small wavelength range) of, say a few Gbps. By tuning its transmitter(s) to one or more wavelength channels, a node can transmit into those channel(s) to receive from the appropriate channels. The system can be configured as a broadcast-and-select network in which all of the inputs from various nodes are combined in a WDM passive star coupler and the mixed optical information is broadcast to all outputs (see figure 1).

Thus, given any physical network topology, the fact is that the lasers (transmitters) and the filters (receivers) can be made tunable opens up a multitude of possible virtual network configurations. Thus, there are three main advantages for employing WDM. First, using WDM, a regular virtual structure can be realized over an arbitrary physical topology by superimposing on it a logical structure. Second, WDM technology allows parallel and concurrent transmissions, thus making available a much larger bandwidth at each station. Third, for better reuse of each WDM channel's bandwidth, the relative positions of the nodes can be changed dynamically based on the traffic fluctuations.

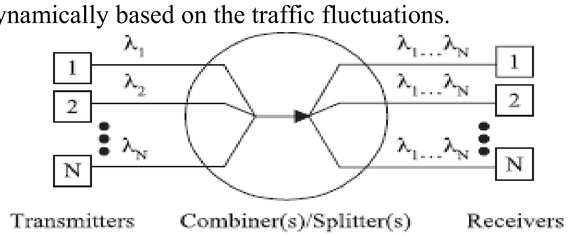


Figure1: Broadcast-and-select WDM network

Network designs using wavelength-division multiplexing (WDM) technologies can be grouped into two classes: single-hop networks and multihop networks [1]. In a single-hop WDM network, information is transmitted directly from the source to the destination without going through intermediate nodes. The main drawback of this design is the need to use fast tunable optical transceivers, which are not mature for mass production at the current stage. On the other hand, in a multihop WDM network only a small number of fixed optical transceivers are needed, but the information from a source node may

Manuscript received July 5, 2010
Manuscript revised July 20, 2010

need to go through a number of intermediate nodes before reaching its destination node.

On any underlying physical topology, one can impose a carefully selected connectivity pattern that provides dedicated connections between certain pairs of stations. Traffic destined to a station that is not directly receiving from the transmitting station must be routed through intermediate stations. This overlaid topology is referred to as the logical or virtual topology.

The logical topology of a multihop WDM network can be either regular (symmetrical), such as ShuffleNet, Toroid, DeBruijn graph, and Hypercube, or irregular (asymmetrical). A regular topology can use simple routing rules, which is crucial for high-speed networks. Two major design issues for regular topologies are: i) modular increase and decrease of network size and ii) adaptation to nonuniform traffic. Research on the first issue can be found, for example, in [2] and [3]. Research on the second issue can be grouped into two classes. The first class focuses on the optimal initial placement of nodes into an empty regular topology given the (average) traffic requirements between all pairs of nodes. This is known as the node placement problem. The commonly used objective function is to minimize the traffic weighted mean internodal distance of a network. If the network consists of equal length links, this is equivalent to minimize the mean network packet delay. Assume that all nodes have already been placed in the network; the second class of research for adapting to nonuniform traffic focuses on the outing/switching problem on a packet-by-packet or call-by-call basis [4], [5].

In general, arbitrary virtual topologies may provide better performance than regular topologies; however, regular topologies provide simpler routing mechanisms which are desirable for high-speed environments since they consume less processing time. Among the regular virtual topologies, the linear bus, ring, ShuffleNet, de Bruijn graph, toroid and hypercube are the more popular ones. The linear bus topology is the simplest; however, designing an optimal linear structure is an NP-hard problem, as shown in [6]. A study on optimized ring structures can be found in [7]. In bus and ring structured networks, the per-node throughput decreases linearly with increasing number of nodes. Thus, these networks cannot be scaled up over larger areas for supporting an increasing number of nodes [8]. Hence, mesh topologies (e.g., ShuffleNet, de Bruijn graph) are preferred since they use a smaller average fraction of the network's resources for transmitting information from a source to its destination. In general, mesh-connected networks have multiple paths between node-pairs; thus, they are more reliable and they can support more concurrent transmissions. All of these properties are very desirable for high-speed switching environments.

Due to the aforementioned properties of mesh-connected regular structures, we concentrate on one such structure in this class. In particular, we consider the de Bruijn graph. The de Bruijn Graphs can support much larger numbers of nodes than the same degree Shuffle nets by having the same average number of hops[5]. Moreover, it retains the simple addressing and routing properties of Shuffle nets.

Depending on the design goals, several optimality criteria may be considered. These include minimizing the flow-weighted average hop distance, minimizing the maximum load over any link, etc. In our present study, minimization of weighted average hop distance is investigated. In general, this optimization problem does not yield to polynomial time solutions (i.e., the search space for the optimum solution grows exponentially with the number of nodes). So, we investigate four efficient heuristic algorithms, namely GREEDY, LOCAL, GLOBAL and ITERATIVE for constructing optimized node arrangements in de Bruijn graph configurations.

II. DE BRUIJN GRAPH

A de Bruijn Graph can support much larger numbers of nodes than the same degree Shuffle net by having the same average number of hops. Moreover, it retains the simple addressing and routing properties of Shuffle nets.

For any positive integers $\Delta \geq 2$ and $D \geq 1$, de Bruijn graph G (Δ, D) [9] is a directed graph consisting of $N = \Delta^D$ nodes with the set of $\{0,1,2,\ldots \Delta\text{-}1\}^D$ nodes where there is an edge from node $(a_1, a_2, \ldots\ldots a_D)$ to node $(b_1, b_2, \ldots\ldots b_D)$ if and only if $b_i = a_{i+1}$, where $a_i, b_i \in \{0, 1, 2,\ldots \Delta\text{-}1\}$, $1 \leq i \leq D\text{-}1$.

D is the diameter of the de Bruijn graph with a deflection penalty D. Each node has in- and out- degree Δ, and Δ nodes (i.e. 000,111) have self-loops (self-loop exists in the logical graph but does not exist in the physical network configuration). The de Bruijn graph structure is inherently asymmetric due to the nodes with self-loops (see Figure 2) [5].

There is one-to-one correspondence between the connectivity of the nodes in the de Bruijn graph G (Δ, D) with all the possible states of a Δ shift register of length D. If state b can be reached from state a in one shift operation in the shift register then there is an edge from node a to b. Therefore, the de Bruijn graph can be seen as the state transition diagram of the shift register [9]. A node in the de Bruijn graph can be represented by a sequence of D digits as defined in the shift register analogy [10]. An edge from node A to node B can be represented by a string of (D + 1) digit. Consequently, any path in the graph of length k from source to destination nodes can be represented by a string D + k digits. The first D digits represent the source node and the last D digits represent the destination node.

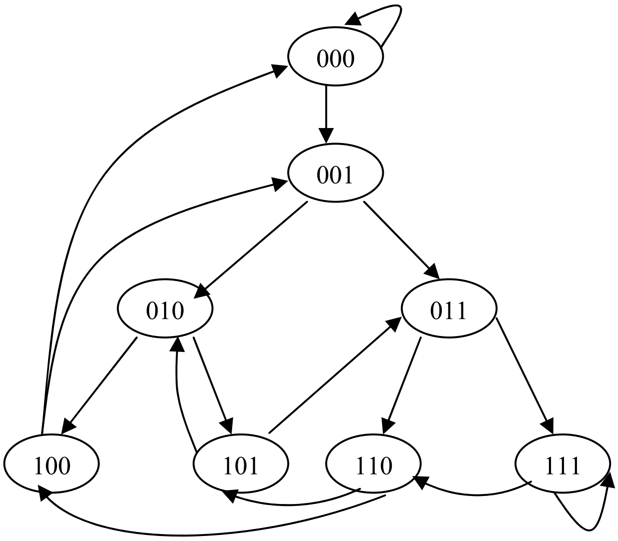


Figure 2: An 8 Node ($\Delta = 2$, $D = 3$) de Bruijn graph

Let **F** = $\{\lambda_{ij}\}$ be the traffic matrix where λ_{ij} denotes the traffic from the source Node i to the destination Node j ($\lambda_{ii} = 0$) and let $\Lambda = \Sigma_{i,j}\ \lambda_{ij}$ be the total load offered to the network.

The average weighted hop-distance under generalized traffic conditions can be computed as follows:
$\overline{H}_{general} = (1/\Lambda \quad \Sigma_{i\in V}\ \Sigma_{j\in V}$ (hop distance from Node i to Node j) x λ_{ij} ; where V is the set of N nodes.
A number of heuristic algorithms for minimizing $\overline{H}$ $_{general}$, i.e., the weighted number of hops, averaged over all source destination pairs, are described next.

III. OPTIMIZATION ALGORITHMS

(i) Algorithm GREEDY

This algorithm employs a greedy approach to maximize the one-hop traffic in a de Bruijn graph. First, the N^2 elements of the traffic matrix are sorted in non-decreasing order. Then, find two higher elements in the sorted list corresponding to the two directly connected distinct nodes having highest average flow value and these two are attached to two directly connected links of the $N\Delta$ links of the de Bruijn graph. This defines the placement of two nodes connected directly. In the successive steps, the next higher two elements of the sorted list are assigned to two of the directly connected available links. If no directly connected nodes found, next higher element from the list is placed in one of the available links. During this process, a flow assignment (λ_{ab}) might be invalid if (1) its two corresponding nodes (nodes a and b) are already placed, or (2) only node a is placed and all the p locations to which it directly connects are already assigned, or (3) only node b is placed and all the p locations from where it is directly connected are already assigned. In such cases, the current element is skipped and the next highest element is considered. An algorithmic description of this procedure is given below. For an illustrative example, see Figure 3.

Algorithm GREEDY:

let **G** ={0,1, ……N-1} be the set of nodes;
let **S** be the sorted list of the elements of the traffic matrix **F**;
while (all nodes are **not** placed) **do**
begin
find two higher elements in S corresponding to traffics from node a to node b (≠ a) and node b to node a, having highest average flow value;
if (links available for connecting node a to b and b to a) **then**
place nodes a and b at two unoccupied locations such that node a is directly connected to node b and node b is directly connected to node a;
discard the two current higher elements from S;
else
find the highest element in **S** corresponding to traffic from node a to node b;
if (link available for connecting node a to node b) **then**
place nodes a and b at two unoccupied locations such that
node a is directly connected to node b;
discard the current highest element from **S**;
end;

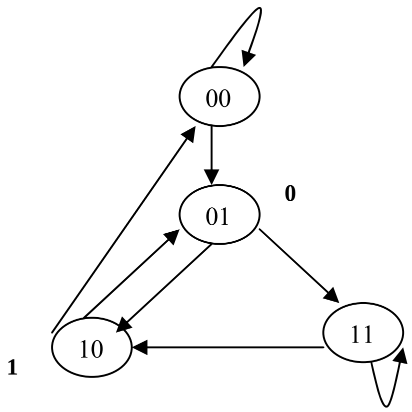
Source \ Destination	0	1	2	3
0	-	8	3	4
1	7	-	6	3
2	8	7	-	2
3	6	3	5	-

(a) Traffic Matrix F (Blank entries denote zero flow)

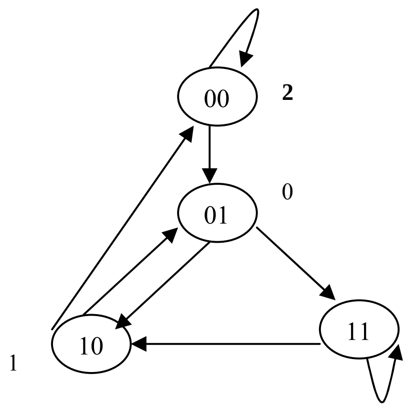
Flow	From	To
8	0	1
8	2	0
7	1	0
7	2	1
6	1	2
6	3	0
5	3	2
4	0	3

3	0	2
3	1	3
3	3	1
2	2	3

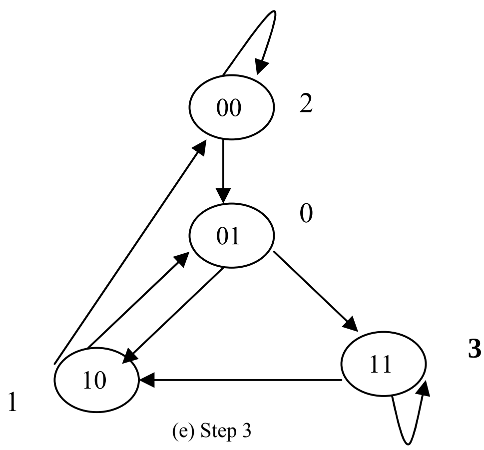
(b) Traffic matrix F sorted in descending order



(c) Step 1



(d) Step 2



(e) Step 3

Average weighted hop distance = 1.42

Figure 3: An illustrative example for Algorithm GREEDY

(ii) Algorithm LOCAL

In (Δ, D) de Bruijn graph, there are Δ^{D-1} groups each consisting of Δ no. of children and Δ no. of parents. However, these groups are not necessarily disjoint and self-loop nodes are counted twice.

This algorithm first places the nodes in a group where no node is self-looped, if possible. The procedure is as follows. From among the N nodes, two sets with Δ nodes in each set are chosen such that the traffic from one set (say set A) to the other (say set B) is heavy compared to the reverse-flowing traffic from set B to set A. Set A nodes are assigned as parents and each of these nodes has direct links to all the nodes in set B which are considered as children. Based on the same criterion, another 2Δ nodes are chosen from the remaining nodes. This process is repeated until all the nodes (parents and children) are assigned. An algorithmic description of this procedure is given below. For an illustrative example, using the above same traffic matrix, step-by-step node placement is shown in Figure 4.

Algorithm Local:

Let G= {0,1,2,….., N-1} be the set of nodes;

For r = 1 to Δ^{D-1} do

Begin

While ($n_{ai} \neq n_{bj}$) do

Begin

Choose 2Δ nodes (n_{a1} , n_{b1} , n_{a2} , n_{b2} , …., $n_{a\Delta}$, $n_{b\Delta}$) from G such that

$$\sum_{i-1}^{\Delta}\sum_{j=1}^{\Delta}(\lambda_{nai\,nbj} - \lambda_{nbj\,nai}) \text{ is maximized;}$$

For i = 1 to Δ do

Begin

Place nodes n_{ai} as the parent nodes;

Place nodes n_{bi} as children nodes;

End

G = G – { n_{a1} , n_{b1} , n_{a2} , n_{b2} , …., $n_{a\Delta}$, $n_{b\Delta}$ };

End

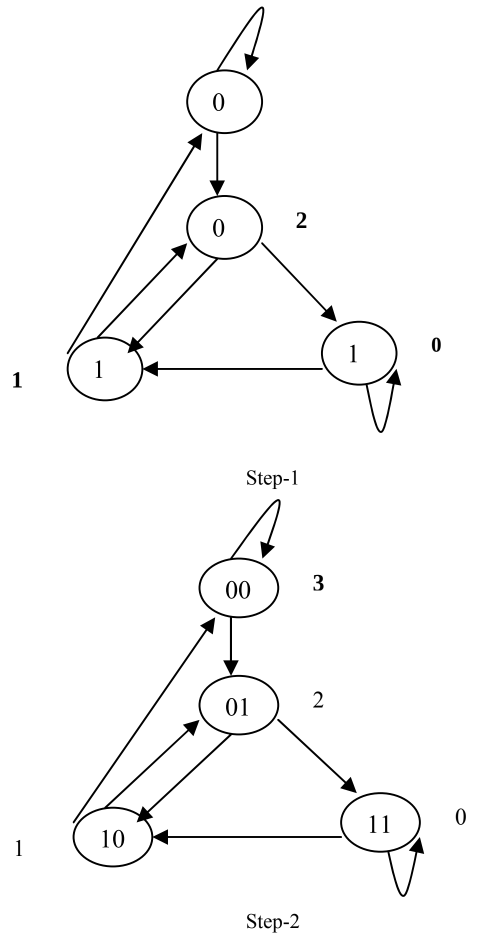
Else

Begin

Choose 2Δ nodes (n_{a1} , n_{b1} , n_{a2} , n_{b2} , …., $n_{a\Delta}$, $n_{b\Delta}$), where self-looped node is counted twice, from G such that

$$\sum_{i-1}^{\Delta}\sum_{j=1}^{\Delta}(\lambda_{nai\,nbj} - \lambda_{nbj\,nai}) \text{ is maximized;}$$

For i = 1 to Δ do
Begin
Place nodes n_{ai} as the parent nodes;
Place nodes n_{bi} as children nodes;
End
$G = G - \{ n_{a1}, n_{b1}, n_{a2}, n_{b2}, \ldots, n_{a\Delta}, n_{b\Delta} \}$;
End
End



Average weighted hop distance = 1.37

Figure 4: Illustration of Local Algorithm (Using the above traffic matrix)

(iii) Algorithm GLOBAL

This heuristic algorithm performs optimization based on global information. The algorithm takes into consideration the traffic to (and from) this node from (to) all the nodes that are already placed in the de Bruijn graph. First, node 0 is placed at location {0}D. Then, a penalty function, fP(0, i, c) for node i (i Є unplaced nodes) for location c (c Є unassigned locations) is evaluated. We have used the penalty function based on the overall flow-weighted hop-distance of a candidate node to and from all the other nodes that are already placed. Then, node i' is placed at location c', where fp(0,i' ,c') = min{fp(0, i, c)} for all unplaced nodes i and unassigned locations c. Penalty values, P (g, c) of all the unplaced nodes g for all the unassigned locations c are updated by incorporating the newly placed node i' into the computations. This process is repeated until all the nodes are placed. An illustrative example for this algorithm is given in Figure 5.

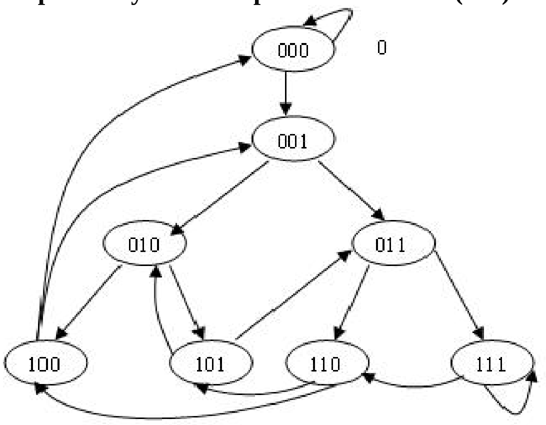
Algorithm Global:
let **G** = {0, 1, ..., N -1} be the set of nodes;
let **C** = $\{0,1, 2,\ldots N-1\}^D$, where D is the diameter of the de Bruijn graph
G = **G -** {0};
C = **C** - $\{0\}^D$;
compute penalty functions $f_p(0, g, c)$ **for** all g Є **G** and c Є **C** ;
P (g, c) = $f_P(0, g, c)$ **for** all g Є **G** and c Є **C** ;
while G ≠ φ **do**
begin
let $P_{min}(g', c')$ = **min**{ P (g, c) | g Є **G**; c Є **C**};
place node g' at position c';
G = **G** - { g'}; **C** = **C** - { c' };
compute penalty functions $f_P(g', g, c)$ **for** all g Є **G** and c Є **C**;
P (g, c) = P (g, c) + $f_P(g', g, c)$ for all g Є **G** and c Є **C**;
end;

function $f_P(g',g,c)$; {Penalty function}
begin
let $\lambda^{g'}_{max}$ and $\lambda^{g'}_{min}$ be the maximum and minimum elements
in $\{\lambda_{g'i} | i \in G - \{ g' \}\}$;
On a (traffic flow, hop distance) – space let L(g') be a straight line from
($\lambda^{g'}_{max}$, 1) to ($\lambda^{g'}_{min}$, D);
{Note that minimum and maximum possible hop-distances are 1 and D}
let traffic from node g' reaches position c in h' hops;
also, let traffic from position c reaches node g' in h' hops;
let d = distance of the point ($\lambda_{g'g}$, h) from L (g');
let d'= distance of the point ($\lambda_{gg'}$, h) from L (g);
$f_P(g', g, c) = (d + d')$;
end;

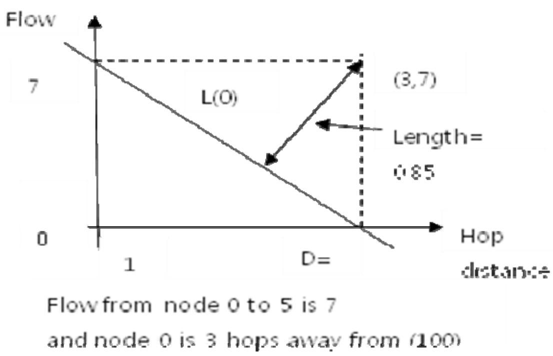
Destination / Source	0	1	2	3	4	5	6	7
0	-	6	0	2	0	7	7	5
1	5	-	8	6	4	8	1	9
2	2	0	-	2	1	3	7	9
3	1	0	5	-	4	9	2	1
4	9	6	1	1	-	5	3	4
5	0	9	5	1	2	-	9	8
6	6	4	2	3	6	4	-	9
7	1	7	0	5	2	8	6	-

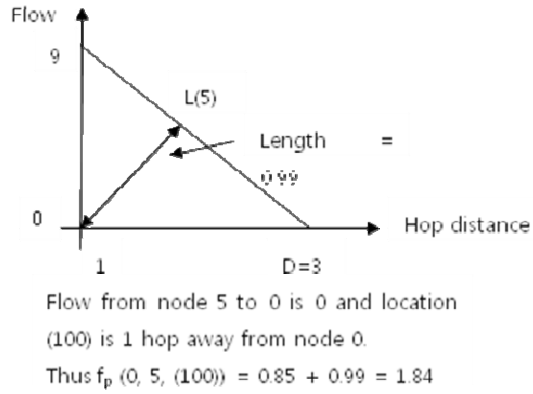
Traffic matrix F (blank entries denote zero flow)

Step-1: Only node 0 is placed at location (000)



Step-2:
Compute penalty function fp (0, 5, (100)) i.e., to compute penalty value calculation for placing node 5 at location (100), given node 0 was placed in the previous step.





Step -3:
We calculate penalty function for nodes 1, 2, 3, 4, 5, 6 and 7 for location 001, 010, 011,100, 101, 110, 111 and final node arrangement will be as follows:

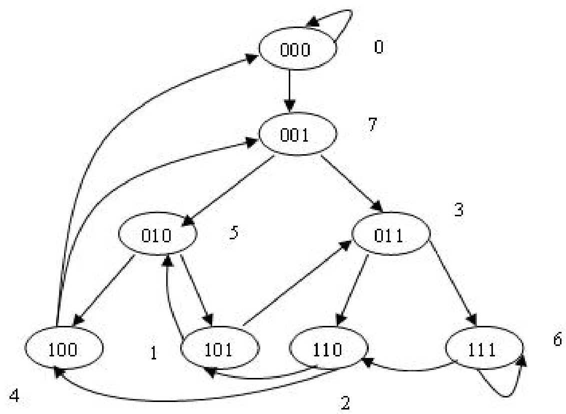


Figure 5: An illustrative example for algorithm GLOBAL

(iv) Algorithm ITERATIVE

Consider an N-dimensional surface composed of (N!) points representing the values of our cost function (i.e., average weighted hop-distance) for all different permutations of [0, 1, ..., N- 1] with node i placed at location whose decimal address value is i . Now, this surface will have several local minima and one (or more) global minimum. Our iterative algorithm starts by picking randomly one point (α_0 , α_1 ......, α_{N-1}) on this surface. Then, at each iteration, node α_u is inserted at the place of node α_g (g < u; g = 0, ... , N- 2; u = g + 1, ... , N-1) if the average hop distance in the new arrangement is less than that in the previous arrangement.

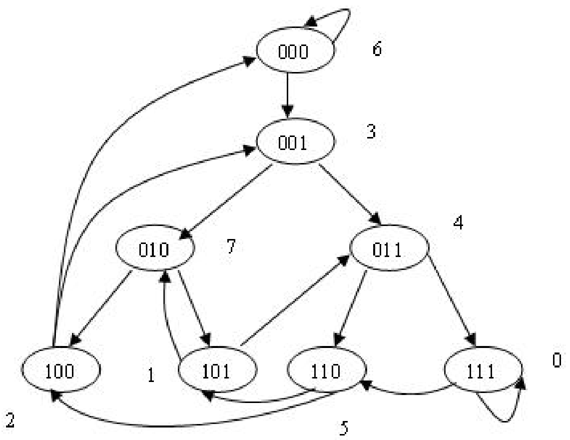
Algorithm Iterative:
let G = {0, 1, ..., N - 1} be the set of nodes;
NoOfIterations = $\log_2 N$;
{Could be performed for different number of iterations as well,

but this value appeared to perform quite well}
for q := 1 to NoOfIterations **do**
begin
pick a random permutation of [0, ... , $N-1$]
{or the output of an earlier algorithm}
as the initial node sequence;
let $\alpha = [\alpha_0,...,\alpha_{N-1}]$ denote this initial node sequence, where α_i is the node placed at location whose decimal address value is i
for g := 0 to N - 2 **do**
begin
for u := g + 1 to N - 1 **do**
begin
$\alpha' = [\alpha'_0,...,\alpha'_{N-1}]$
where $\alpha'_i = \alpha_i$ for $0 \le i \le g-1$ and $u+1 \le i < N-1$
$\alpha'_g = \alpha_u$ and $\alpha'_i = \alpha_{i-1}$ for $g+1 \le i \le u$
let $H(\alpha)$ = average weighted hop-distance in the de Bruijn Graph represented by α ;
if $H(\alpha') < H(\alpha)$
$\alpha \leftarrow \alpha'$;
end;
end;
end;

An illustrative example for this algorithm is given in Figure 6.

G = set of nodes {0,1, 2, 3, 4, 5, 6, 7};
NoOfIterations = $\log_2 N = \log_2 8 = 3$;
Consider, the node placement of GLOBAL algorithm is the input of ITERATIVE algorithm.
Thus from previous example, α = {0, 7, 5, 3, 4, 1, 2, 6}
Average weighted hop distance in GLOBAL algorithm $H(\alpha) = 2.11$

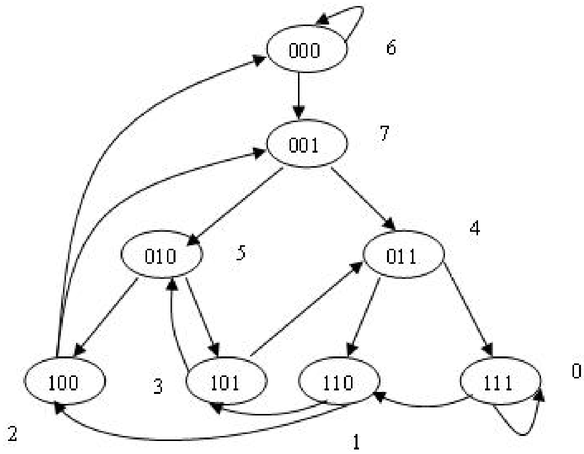
Step 1:
New node arrangement after first iteration is as follows:



At first iteration average weighted hop distance $H(\alpha)$=1.96

Step 2:
In the second iteration, there is no change in node arrangement.

Step 3:
Final node placement is as follows:



Average weighted hop distance $H(\alpha)$ = 1.93

Figure 6: Illustrative example of ITERATIVE algorithm

IV. COMPARISON AMONG ALGORITHMS

Algorithm GREEDY is the simplest and is also the fastest. It provides good solutions for small networks; however, its performance degrades as N increases. Algorithm GREEDY attempts to maximize only the one-hop traffic. However, this approach can lead to large hop-distances for the remaining traffic that are not routed in one hop. The numerical example in Fig. 3 points out this weakness of the algorithm. In Fig. 3, the traffic matrix is constructed in such a way that algorithm GREEDY routes the first four heaviest traffic in one hop. However, all the remaining traffic is routed via maximum number of hops.
Algorithm LOCAL works only on local traffic information. However, unlike the GREEDY algorithm, while considering a pair of nodes a and b, algorithm LOCAL considers the traffic from node a to node b as well as the traffic from node b to node a.
Algorithm GLOBAL, while choosing a node for placing in the de Bruijn graph, considers all the nodes that are already placed. Algorithm GLOBAL employs a penalty function (see Fig. 5) to reduce the difference between i and j. For example, in Fig. 5, first node 0 is placed at location

(000). Then, node 7 is placed at location (001) since node 0 and Node 7 have the minimum traffic among themselves and are maximum hop distances away from each other.
The algorithm ITERATIVE exhibits the general properties of iterative algorithms. For example, performance of this algorithm can be arbitrarily improved, at the cost of reduced speed by running the algorithm for a larger number of iterations. The computations performed in this algorithm are very regular and can be easily conducted in a pipelined fashion. Moreover, different iterations can be performed in parallel on different processors of a multiprocessor system where each processor uses its independent random number generator for constructing the initial random sequence.
The overall performance of the ITERATIVE algorithm is better than other algorithms. GREEDY algorithm is the fastest, but it does not perform as good as the other algorithms, especially for large networks.

V. NUMERICAL RESULTS

Each of the two algorithms is applied to the same traffic matrix. In order to reduce the bias of certain algorithms to certain traffic patterns, the algorithms are applied to a fixed set of 20 different, random traffic matrices. The weighted average hop-distances obtained from the experiments are then averaged over the set of 20 traffic matrices. These results are tabulated in Table I.

Table- I

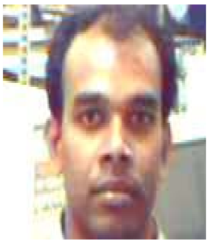
Degree	Diameter	No. of Nodes	Average weighted hop distance			
			Algo. GREEDY	Algo. LOCAL	Algo. GLOBAL	Algo. ITERATIVE
2	2	4	1.41	1.35	1.35	1.27
2	3	8	2.29	2.30	2.04	1.91
2	4	16	2.88	2.72	2.83	2.67
2	5	32	3.72	3.77	3.69	3.52
2	6	64	4.84	4.44	4.32	4.12
3	2	9	1.85	1.77	1.61	1.53
3	3	27	2.71	2.59	2.47	2.40
4	2	16	1.91	1.76	1.74	1.69
4	3	64	3.94	3.83	3.68	3.45

VI. CONCLUSION

Four heuristic algorithms based on GREEDY, LOCAL, GLOBAL and ITERATIVE approaches are proposed for constructing photonic implementations of optimized de Bruijn graph configurations. These logical structures are multihop in nature, and they can be superimposed on any physical topology by exploiting the broadcast-and-select property of WDM lightwave networks in which all of the inputs from various nodes are combined in a star coupler and the mixed optical information is broadcast to all outputs.
Assuming *static* traffic conditions, these heuristic algorithms optimize the weighted average hop distance in the network. A comparative study is made for these algorithms and their performance was demonstrated by employing numerical examples.
The algorithms corresponding to our discussions above are static. However, the virtual network topology might have to be reconfigured in order to respond to any change in the traffic pattern. Reconfigurable networks would require tunable transceivers, instead of fixed-tuned ones. Although, currently, wavelength-agile transceivers are quite expensive, we believe that this cost-performance tradeoff will favor reconfigurable networks when low-cost tunable transceivers become commercially available. Thus, a dynamic reconfiguration heuristic is to be studied next.

VII. REFERENCES

[1] A. S. Acampora and M. J. Karol, “An overview of lightwave packet networks,” IEEE Network Mag., vol. 3, pp. 29–41, Jan. 1989.
[2] N. F. Maxemchuk, “Regular mesh topologies in local and metropolitan area networks,” AT&T Tech. J., vol. 64, no. 7, pp. 1659–1685, Sept. 1985.
[3] P. P. To, T. S. Yum, and Y. W. Leung, “Multistar implementation of expandable shufflenets,” IEEE/ACM Trans. Networking, vol. 2, pp. 345–351, Aug. 1994.
[4] K. C. Lee and V. O. K. Li, “Routing for all-optical networks using wavelengths outside erbium-doped fiber amplifier bandwidth,” Proc. IEEE INFOCOM’94, vol. 2, Toronto, Ont., Canada, 1994, pp. 946–953.
[5] K. N. Sivarajan and R. Ramaswami, “Lightwave networks based on de bruijin graphs,” IEEE/ACM Trans. Networking, vol. 2, pp. 70–79, Feb. 1994.
[6] S. Banerjee, B. Mukherjee and D. Sarkar "Heuristic algorithms for constructing near-optimal structures of linear multihop lightwave networks," Proc., IEEE INFOCOM ’92, Florence, pp. 671-680, May 1992.
[7] S. Banerjee and B. Mukherjee, "The Photonic Ring Algorithms for near-optimal node arrangements," Journal of Fiber and Integrated Optics, vol. 12, pp. 133-171, April 1993. (special issue on networking with optical technology).
[8] N. F. Maxemchuk, "Regular mesh topologies in local and metropolitan area networks," AT&T Tech. J., vol. 64, pp. 1659-1686, Sept. 1985.
[9] B. Bollobas, “External Graph Theory with Emphasis on Probabilistic Methods”, American Mathematics Society, 1986.
[10] S. W. Golomb, “Shift register sequences”, Aegean Park Press, 1982.



Tarun Kumar Ghosh received the B.Tech. and M.Tech degrees in Computer Science & Engineering from the University of Calcuta, Kolkata, India in 1999 and the Bengal Engineering & Science University, Howrah, West Bengal, India in 2001 respectively.

He joined as a Lecturer in the Department of Computer Science & Engineering, Asansol Engineering College, West Bengal, India in February 2001 and worked there for about three and half years. Then he has been working as an Assistant Professor in the Department of Computer Science & Engineering, Haldia Institute of Technology, West Bengal, India for about six years. He published eight papers in various journal and conference proceedings. He has written a book titled “Computer Organization & Architecture” from Tata McGraw Hill publication. He is a member of ACM. His research interests include lightwave networks, high-speed computing and computer architecture.

Debi Bera received the M. Sc. Degree in Computer Science from Vidyasagar University, West Bengal, India in 2008 and will receive the M. Tech. degree in Computer Science & Engineering from West Bengal University of Technology, India in July 2010.

Her research interests include lightwave networks and computer architecture.