

# Design of 2-level Hierarchical Ring Networks

Kishore Kshirsagar, Kesav Kaza and Krishnan Rajan  
International Institute of Information Technology, Hyderabad, India

**Abstract**—Hierarchical Ring Networks are widely used in communication systems, logistics and shared multiprocessors. It is hence imperative that efficient design of these structures has to be carried out, to reduce costs of their installation and help increase the ease of management.

In this paper, we look at the problem of designing efficient 2-level Hierarchical Ring Networks, in the context of link costs optimization. 2-level HRNs only have two kinds of rings - local rings to connect disjoint sets of nodes and a global ring to interconnect all the local rings. We present an algorithm that is based on Simulated Annealing. We present results for scenarios containing upto 300 nodes. The results obtained from the algorithm are most encouraging.

## I. INTRODUCTION

Self-healing rings (SHRs) are widely used in communication systems to increase the resilience of networks at a very small cost and to improve the recovery time. SDH/SONET, WDM systems and now Resilient Packet Rings are configured as SHRs. SHRs belong to a class of survivable network architectures that implement physical protection [14]. When networks tend to become very large, they are arranged in interconnected groups. A hierarchical network is one possible way to arrange the network. One such hierarchical network, the features of which SHRs can make use of, is the Hierarchical Ring Network (HRN). Hierarchical networks are used widely, from logistics to processor design and communications. One main advantage of these structures is that they provide greater resilience compared to simple structures like ring, bus and star topologies. Also, breaking down a large network into smaller structures helps in managing the network elements efficiently. Apart from HRNs, such structures have also been modeled as “Tree of Rings” ([16], [15]). For example, a 2-level HRN can also be represented as a weighted tree of rings, where all the weights are equal to 1.

An example of a HRN is given in Fig. 1. We call the rings connecting the nodes in each group as “local rings” and the ring connecting all these local rings together as the “global ring”. Also, we refer to the hierarchical nodes that constitute the global ring as “Main Control Nodes” or MCNs. A variation in such a topology is one in which each local ring has two nodes on the global ring. These are called as “dual-homing networks”. Another variation is having multiple levels, where the local rings are the global rings for lower level rings and so on.

The 2-level Hierarchical Ring Network Design problem is one in which, given the locations of a set of nodes, the aim is to find a partition of this set such that, the sum of the costs of the links connecting all the nodes of each group into rings

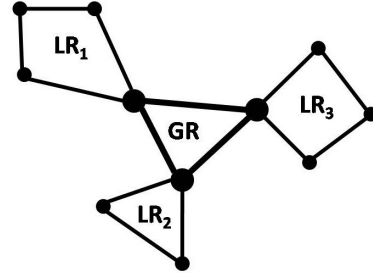


Fig. 1. An example of a 2-level HRN.  $LR_i$ s are the local rings and  $GR$  is the global ring

and the costs of the links connecting each of these groups at a single point constituting another ring is minimum.

Due to its wide-spread utility (for example, [18], [17], [12]), a lot of work has focussed on trying to efficiently design these networks, sometimes with a few variations. Traditionally, this problem has been solved in a series of chronological steps (for example, [10] and [9]), with few changes. These are

- 1) Identification of clusters of nodes
- 2) Construction of local rings
- 3) Selection of MCNs
- 4) Construction of a global ring

Depending on the problem, a further step that has been included is the Routing of the network to minimize the capacity usage. Though the problem does become simpler by breaking it down into sub-problems, one main issue is that by solving them sequentially, we may end up with sub-optimal solutions [10]. For example, let us consider the construction of the local and the global rings as two separate steps. The objective of local rings’ cost minimization inherently makes us select a partition such that the nodes in each local ring are close to each other and far away from the nodes of other rings. Hence, there is a conflict between the objectives of local and global rings’ cost optimization. We need an algorithm that does not work in sequential steps and optimizes multiple objectives at the same time.

This is the endeavour we take up in this work. We present a heuristic for the 2-level Hierarchical Ring Design problem that can be extended to multiple levels. However, this algorithm only works for “single-homing networks” like the one shown in Fig. 1. The algorithm is based on the Simulated Annealing (SA) meta-heuristic. The main advantage of the algorithm is the capability of giving good solutions in reasonable amount of time. Also, while previous works have given solutions for scenarios containing only upto 120 nodes, we present solutions

for upto 300 nodes which can be obtained in a small amount of time.

This paper is organized as follows: in Section II, we present related work and discuss the solutions to similar problems achieved in the past. In Section III, we present the model of the problem, the variables, the constraints etc. In Section IV, we present our heuristic - the initial solution construction phase and the improvement phase that is based on SA. In Section V, we present our results. We compare our solutions with ones obtained from the TOMLAB/CPLEX [22] optimizer and analyze the achieved results. Finally, in Section VI, some concluding remarks are presented.

## II. RELATED WORK

The HRN problem and its variations have received a good amount of attention in the past. In [4], a multiple-level algorithm is presented for a modified Hierarchical Self Healing Rings design problem, where apart from the link lengths, the demand at each node is considered in designing the network. The greedy heuristic presented is capable of handling large networks (nearly 150 nodes), but the quality of solution is not established. Also, there is a requirement of having equal number of nodes in each ring. An enumerative approach for an exact solution that works for small networks is presented.

In [3], the problem statement is slightly different. There is an assumption that the MCNs have already been identified. Hence the global ring problem is reduced to the Travelling Salesman Problem (TSP) over these pre-specified MCNs. For the local rings, a parallel savings heuristic is suggested, that performs with a worst-performance guarantee.

In [5], a heuristic is suggested for a related problem - ring topology construction in a sparse network where adjacent rings need to have atleast two nodes in common. Such a structure is called as Dual Ring Interconnection (DRI). The problem is formulated as the classic set covering problem and an additional heuristic is applied to ensure that the DRI condition is satisfied.

In [6], the problem considered was a slight variation of the HRN problem. While the backbone network consisting of the MCN-like digital hubs has to constitute a ring topology, the local access network could be radial. A tabu search heuristic is employed and scenarios with up to 300 nodes are considered.

In [7] a partition, construct and perturb method is presented for the construction of near-optimal hierarchical networks. The problem is modified to include the intra-ring traffic along with the link costs. Their method is compared with an integer-linear programming formulation and the method converges to the optimal solution for small network sizes only.

In [8], the network structure is more relaxed and the groups of nodes are not required to be disjoint and so, they can be part of multiple rings. Also, there is no hierarchy in the nodes. Here once again, a tabu search heuristic is used to solve the problem of ring interactions. However, scenarios with only upto 48 nodes are considered. Also, the quality of the solutions was not properly assessed and the comparison of their solutions was done against greedy heuristics.

In [9] a similar problem, the SONET ring assignment problem (SRAP) is considered. Here, the aim is to minimize the total cost, subject to the capacity limit of the ring, which is determined by the total traffic anticipated between nodes. The authors present an exact method, based on integer-programming techniques to cluster the nodes, into the smallest number of rings possible. However, the solution is restricted to only those sets of nodes, that will be on the same SONET ring. The construction and costs of the global and the local rings are not considered.

In [10], once again, a modified version of the problem is considered. Apart from the network design problem, the routing problem is also considered and the aim is to design the HRN which satisfies the communication demands as cheaply as possible. The approach employed is the Branch-and-Price algorithm, which performs with optimality for a scenario with a small number of nodes. However, the computational costs involved are quite prohibitive and a scenario with 20 nodes requires a lot of computation time.

In [11], a similar problem - the design of Balanced Disjoint Rings (BDR) was considered. In this problem, the set of nodes is divided into groups and the nodes belonging to each group are connected in a ring topology (local rings). The local rings in such a network are not connected to each other, i.e. a global ring is missing. The authors tried to solve the problem using a Cyclic Exchange heuristic. A solution to the HRD problem was also offered: the problem of interconnecting the rings could be considered as a General Travelling Salesman Problem (GTSP) and a goodness measure that included the cost of this GTSP tour, apart from the local rings' costs could be the criterion for evaluating a solution. There are two problems with their approach: firstly, the initial solution construction phase involves selection of the MCNs such that they are as far away from each other as possible. The initial construction heuristic plays a very important role in how the final solution takes shape. Hence, while this is definitely a good step if one wants to solve the BDR problem, it cannot be used for the HRN problem, where even the cost of the global ring is to be considered. Secondly, the cost of the GTSP is not taken into consideration in each step of the algorithm, but only in the final evaluation, after the local rings have been finalized.

We present an algorithm that improvises on the current standards. Our algorithm can provide solutions for upto 300 nodes in a small amount of time.

## III. PROBLEM FORMULATION

In this section, we provide a formulation of the HRN problem. While mathematical modeling has been done in literature ([6]-[11]), ring constraints have usually been left out, either, for the local or the global rings. Here we take up this problem and provide efficient constraints to obtain the HRN structure.

Let  $G = (V, E)$  be an undirected complete graph, where  $V$  is the set of vertices and  $E$  is the set of edges. The number of nodes in  $G$ ,  $|V|$ , is equal to  $n$ . Let us denote each vertex as  $v_i$ , where  $i = 1 \dots n$ . Let the edge between  $v_i$  and  $v_j$ , be

denoted as  $e_{ij}$  and let its corresponding weight be given by  $w_{ij}$ . Also, let the number of desired rings be equal to  $R$ . We model the problem as dividing the set of vertices into  $R + 1$  groups. Here the  $(R+1)^{th}$  group refers to the global ring. It is understood that the vertices in this group are also repeated in the local groups. We define some variables before we proceed to the formulation.

- $N_{ir}$  is a binary constraint that is equal to 1 if the vertex  $v_i$  is present in the ring  $Ring_r$
- $x_{ijr}$  is once again a binary constraint that is equal to 1 if the edge  $e_{ij}$  is present in ring  $Ring_r$
- $L_1, L_2$  are the lower and upper bounds on the local ring sizes respectively

Then the Hierarchical Ring Network Design problem can be stated as, the minimization of

$$C = \sum_{r=1}^{R+1} \sum_{j=1}^n \sum_{i=1}^n x_{ijr} w_{ij} \quad (1)$$

subject to the constraints

$$L_1 \leq \sum_{i=1}^n N_{ir} \leq L_2 \quad r = 1, \dots, R \quad (2)$$

$$\sum_{i=1}^n N_{ir} = R \quad r = R + 1 \quad (3)$$

$$\sum_{r=1}^R N_{ir} = 1 \quad i = 1, \dots, n \quad (4)$$

$$\sum_{i=1}^n N_{ir} N_{i(R+1)} = 1 \quad r = 1, \dots, R \quad (5)$$

$$x_{ijr} \leq N_{ir}, \quad x_{ijr} \leq N_{jr} \quad \forall i, j, r \quad (6)$$

$$\sum_{j=1}^n x_{ijr} = 2N_{ir} \quad i = 1, \dots, n, \quad r = 1, \dots, R + 1 \quad (7)$$

$$\sum_{i \in V^*} \sum_{j \in V^*, j > i} x_{ijr} \leq |V^*| - N_{tr} \\ \forall V^* \subset V, V^* \neq \emptyset, \quad \forall t \in V \setminus V^*, \quad \forall r \quad (8)$$

$$x_{ijr}, N_{ir} \in \{0, 1\} \quad i, j = 1, \dots, n, \quad r = 1, \dots, R \quad (9)$$

Eq. 1 represents the cost function that is to be minimized. Eq. 2 - Eq. 7 are the design constraints. The Eq. 2 is the ring cardinality constraint. It ensures that the size of each ring (except the global ring) is between  $L_1$  and  $L_2$ . The size of the global ring is controlled by Eq. 3. The Eq. 4 ensures that all the vertices are part of only a single local ring. The Eq. 5 ensures that only one vertex from each ring is part of the global ring. The constraints in Eq. 6 make sure that an edge is selected in a ring, only if its adjacent vertices are also present

in that ring. Eq. 7 is the vertex degree constraint. It makes sure that the degree of each vertex present in a ring (both local and global) is exactly 2. The Eq. 8 ([11]) is the subtour elimination constraint. It ensures that the graph of each ring is connected. Finally Eq. 9 is the binary constraint on the variables.

## IV. OUR SOLUTION

### A. Simulated Annealing

Simulated Annealing (SA) is a probabilistic method for finding the global optima of a cost function, proposed in [1] and [2]. It is basically an imitation of the annealing process in which a liquid freezes so that when the structure settles down, it has a minimum energy configuration. We give a short introduction to the basics of SA in this section. A complete mathematical analysis can be found in [19].

The main advantage of SA is that it avoids local minima (or optima) by ‘‘jumping’’ from the current solution to a point in the neighbourhood. The probability of this jump depends on the value of the cost function at the current solution and the neighbour and also on the temperature value. The temperature function is to be selected such that at the initial stages, these jumps are to be relatively frequent compared to the later stages, when the structure cools down.

Let  $f$  be the real-valued cost function to be minimized, defined on a set  $S$ . Let  $T(t)$  be the temperature of the structure at time  $t$ . We start with an initial feasible solution,  $s(0)$  and an initial temperature  $T_0$ . We proceed towards the optimum as follows:

- 1) From the current solution,  $s$ , select a random point,  $s^*$  in  $S$ , in the neighbourhood of the solution
- 2) Calculate the value of  $f(s^*)$
- 3) If  $f(s^*) \leq f(s)$ ,  $s = s^*$
- 4) Else, select  $s^*$  with probability  $\exp\left(-\frac{(f(s^*)-f(s))}{T(t)}\right)$
- 5) If  $T(t) \geq T_{limit}$ , make  $t = t + 1$  and go to step 1
- 6) Otherwise, stop annealing

### B. The Algorithm

In this section, we present our SA based heuristic. The algorithm can be broadly broken down into two 2 steps - an initial solution construction step and the improvement step. The former is discussed in Section. IV-B1 and the latter in Section. IV-B2

1) *Initial Solution Construction Heuristic*: The initial solution plays a very important role in how well the algorithm performs overall. In this section we present two methods for producing an initial solution. While the first is based on k-means clustering, the second is based on the construction of a Minimum Spanning Tree (MST).

a) *Clustering based heuristic*: In this construction method is based on clustering of nodes in the network, followed by identification of the shortest path connecting the nodes in each group to form a ring. The method is as follows:

Let us assume  $H_{ij}$  is the graph of the solution. At the initial stage, it is an empty set as no edges have been selected at the initial stage.  $N_{ir}$  is also empty as nodes have not been assigned to the rings.

- 1) Apply k-means clustering on the set of nodes with  $k = R$
- 2) For every ring,  $r$ , solve the TSP
- 3) For the global ring, solve as General Travelling Salesman Problem
- 4) Update  $N_{ir}$  and  $H_{ij}$

Here, after the application of k-means on the set  $N$ , we solve the TSP for each ring  $r$ . For this purpose, we have tried using the Christofides' 3/2-approximation heuristic [13] and a SA based algorithm. The Christofides' algorithm gives a solution in less time, but the SA based algorithm gives a better solution. However, the time taken by the latter is more. As the construction is done just once, it is not prohibitive if we use it in this stage for each ring. Following this, the identification of MCNs and the construction of the global ring is modeled as the GTSP. For solving the GTSP, we have used the method proposed in [21]. The matrices  $N_{ir}$  and  $H_{ij}$  are updated depending upon the MCNs and the edges that are selected to be part of the initial solution.

The reason k-means is applied is that it leads to rings containing nodes that are very close to each other. However, as the basic premise of k-means is the increase in inter-class variance, applying it here leads to an initial solution that is low on the cost of local rings, but has a high global ring cost. Also, this method is computationally expensive as we have to solve the GTSP once, and the TSP multiple times. The main advantage of using it is that it leads to solutions with very few or no overlapping edges. This is unlike the MST based heuristic which is explained next, where we have many overlapping edges. It is possible that we may end up with a solution in which some rings have just 2 or 3 nodes, but the improvement heuristic normalizes the ring sizes so that their cardinality is within the bounds.

*b) MST based heuristic:* This is the second construction heuristic and it can be further broken down into two steps - the MCN identification step and the node assignment step. For the MCN identification, we proceed as follows:

- 1) Construct the Minimum Spanning Tree  $M$ , of the graph  $G$
- 2) Compute the degree of all the nodes in  $M$
- 3) Select the  $k * R$  nodes with the highest degree. Let us call this set  $S_{top}$
- 4) Randomly select  $R$  nodes from  $S_{top}$ . Call this set  $MCN$
- 5) Assign each node from  $MCN$  to a separate ring
- 6) Update  $N_{ir}$

As we are trying to minimize the distances between the MCNs and the distances between the MCNs and other nodes in the local rings, it is imperative that the MCNs should be central nodes in the graph. However, to locate such nodes, so that both costs are minimized would be very difficult for a construction heuristic. We have employed the MST approach because although it does not say anything about distances between the MCNs, it does minimize the local rings' lengths.

Also, we select the top  $k * R$  nodes in Step 3 because we have observed that most of the top nodes in  $M$  have a

degree of 3, and very few have degree 4. The random selection brings about a sense of fairness in the MCN selection. We thus have alternate starting positions instead of fixed ones. In our experiments, we used a value of  $k = 1.5$ . MST based construction was considered in [11], but was discarded there because of poor results. However, there the farthest points in the MST were selected as seed nodes for the BDR problem. Hence, it is highly possible that outlying points are selected as seed nodes, which could lead to the poor results reported. We do not have such a problem because outlying nodes are usually of degree 1 or 2 and hence are less likely to be selected as MCNs.

In the next step in the MST based construction heuristic, we assign the rest of the nodes to each ring. We follow a simple best-insertion heuristic as follows:

As we have assigned no edges yet,  $H_{ij} = 0$ . Let  $Ring_r$  represent set of nodes in ring  $r$ .

- 1) For every  $r$ , assign a free node  $n_i$  to  $r$  such that cost of  $Ring_r \cup \{i\} < \text{cost of } Ring_r \cup \{j\} \forall j \neq i$
- 2) Update  $N_{ir}$  and  $H_{ij}$
- 3) Repeat steps 1 and 2 until no nodes are left

Hence what we are doing is that for each ring, we assign that node, which on insertion in the ring would lead to the least increase in the cost of its edges. Once again, the insertion of each node in a ring is between those 2 nodes where the increase in cost is minimum. This is basically a greedy heuristic, which is trying to maintain the uniformity of ring sizes. While this construction method is fast, there are a large number of overlaps between edges belonging to different rings. This is because, even though a node may be closest to one ring, it is assigned to another, because it happens to be the closest node to that ring. However, these overlaps are removed to a large extent in the improvement heuristic, at the cost of convergence time.

*2) Improvement Heuristic:* In this section, we present an algorithm for improving the solution obtained either from the MST-based heuristic or the Clustering-based heuristic. We also present a method for randomly selecting neighbours and explain our choice of annealing function.

The improvement heuristic is given as follows.

- 1) Start annealing
- 2) Randomly select a node,  $v_i$ . Let us assume it belongs to ring,  $Ring_{r,1}$
- 3) If cardinality of  $Ring_{r,1}$ ,  $|Ring_{r,1}| \leq L_1$  go to step 2
- 4) Randomly select a target ring,  $Ring_{r,2}$
- 5) If  $|Ring_{r,2}| \geq L_2$  go to step 4
- 6)  $S_{new} = \text{Move}(v_i, Ring_{r,2})$
- 7) If  $S_{new}$  is a valid annealing move, update  $H_{ij}$  and  $N_{ir}$
- 8) If  $T(t) > T_{limit}$ , make  $t = t + 1$  and go to step 2
- 9) Otherwise, stop annealing

Simulated Annealing depends on a number of parameters and strategies to converge to the optimal solution. Usually these parameters change with the task at hand. We have worked on some of these to suit our problem. Those are

- Generation Probability Function - determines with what distribution the neighbours are randomly selected
- Probability of Acceptance - determines with what probability an inferior solution is accepted
- Annealing Function - determines how fast or slow the cooling takes place

a) *Generation Probability Function*: The standard practice while using SA is that, a neighbourhood is constructed and a neighbour is randomly chosen following a uniform distribution. However, this strategy may not always yield good results or may perhaps delay the convergence. For example, in our case, the neighbourhood is the set of nodes that can be moved to another ring. It is possible to select a node that is in its optimal ring and the cost of its movement may be small. Then, once the movement is complete, it is not possible at all to move towards the optimum, as it was earlier in the right ring, unless we make a random move to the previous state.

Hence, we have introduced the concept of a Precedence list when the nodes are to be selected for moving. For every node, we maintain a metric that signifies, how “comfortable” or “uncomfortable” a node is, in its current ring. That is, we see how ready a node is, to move to another ring. The metric we have used is the ratio of the sum of the costs of the ring to the sum of the costs of the ring, when that particular node is removed from it.

If node  $v_i$  belongs to ring,  $Ring_r$ , then,

$$Precedence(i) = \frac{\text{cost of } Ring_r}{\text{cost of } Ring_r - \{i\}} \quad (10)$$

The greater the precedence order, the more uncomfortable a node is in the current ring, and greater the probability of it being selected for movement. This list is updated every time a valid move is carried out. We have also conducted experiments without the Precedence list and present both results for comparison. The results are promising and highlight the importance of the Precedence list in selecting the right neighbours.

b) *Cooling Schedule*: The cooling schedule is a most important aspect of SA. It determines the rate of saturation of the process. That is, it says how fast or slow the algorithm proceeds. It also plays a major factor in determining whether a particular inferior solution has to be accepted or rejected. A number of strategies are presented on how to select a temperature function. The most commonly used function is a geometric progression. That is, the cooling is of the form

$$T(t) = \alpha^t T(0) \quad \alpha < 1 \quad (11)$$

We have however used the  $\log()$  function, which is of the form

$$T(t) = c/\log(t) \quad (12)$$

This is because, the  $\log$  function is known to converge with probability 1 ([20]) and has a smooth decay curve. The problem with the geometric progression is that the function decays too quickly.

c) *Acceptance Probability*: This parameter determines how easily an inferior solution - one that has a higher cost than the current solution is accepted in the annealing process. It depends on the cost function values of the current and the new solution and the temperature function. Usually the function that is used is

$$\exp\left(-\frac{f_{new} - f_{old}}{T(t)}\right) \quad (13)$$

We use a slightly modified version of the function:

$$\exp\left(-\frac{(f_{new} - f_{old})/f_0}{T(t)}\right) \quad (14)$$

where  $f_0$  is the value of the cost function evaluated at the initial state  $s(0)$ , i.e. we are normalizing the cost function. The reason being that the normalization of the difference of the cost function makes the acceptance function rely more on the degree of change brought about by the new solution instead of just the quantum of change. This gives us more control over which moves to make. For example, if we want to make a jump to an inferior solution even if the increase in the cost compared to the initial solution is 10%, with a probability, say 0.5 at the start of the experiment, we can tune the temperature function appropriately.

In fact, in our experiments, the temperature function is set to satisfy this above condition.

d) *Move() routine*: This routine ensures that the random moves that are generated always lead to feasible solutions, that is, they follow the constraints given in Section III. This function is called with the index of the node that has to be moved and the target ring index. Move() first removes the node from its current ring by deleting the 2 edges with its neighbours in that ring. This is followed by the addition of the edge between those two neighbours in the ring. If the node happens to be an MCN, then the 2 adjacent edges from the global ring are deleted and just like in the case of the local ring, the neighbouring MCNs are joined. In the next step, the node has to be inserted in the target ring. This is done as follows: the node is inserted in between two nodes in the target ring such that the increase in the link cost is minimum. After the best location has been found, the edge is broken and links are formed between the new neighbours and the node. Following this, a check is made if the shifting of the MCN of the target ring from the current to this new node would bring about a decrease in the cost. If yes, the edges with the neighboring MCNs are deleted and new edges are formed with this node.

e) *Stopping criteria*: The stopping criteria for SA can be an upper limit on the number of iterations, a lower limit on the temperature function or a lower limit on the cost function. We have used a mixture of the first two. We stop the annealing if the number of successive iterations without any improvement is equal to some  $\lambda$  or if the temperature function reaches a limit,  $T_{limit}$ . The annealing stops when the first of these two criteria is met.

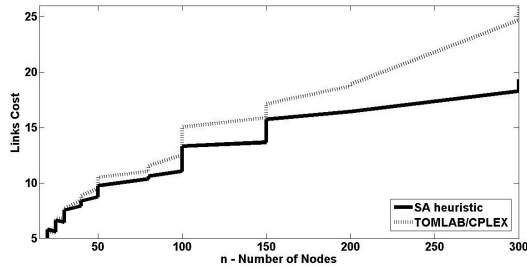


Fig. 2. Cost of Links of the SA heuristic compared with TOMLAB/CPLEX. Here we have used Precedence List and construction by Clustering for the SA heuristic

## V. RESULTS

In this section we present computational results of the performance of our algorithm. We compare our algorithm with the TOMLAB/CPLEX integer programming tool. All programs are run on a system with a memory of 2 GB and a processor speed of 2.93 GHz.

To test the algorithms, we have 10 classes of data depending on the number of nodes,  $n$ , in the scenario (each scenario being a randomly generated instance). The value of  $n$  ranges from 20 to 300. Each of these classes is further divided into 2 sub-classes, based on the number of rings in the topology. For every class and sub-class, we generate 5 random scenarios and average the cost and the time taken by the algorithm.

The Table I shows the comparison of our algorithm with the performance of TOMLAB/CPLEX. Here, we run our algorithm, measure the CPU run-time and set a time limit for the integer programming tool. This can be controlled using the `cpxControl.TILIM` parameter in TOMLAB. Hence we only allow the tool to run as long as the time taken by our algorithm. In this table, the  $C_{A,1}$  refers to the average value of the cost function achieved by our algorithm,  $T_{A,1}$  is the average time taken by our algorithm over the 5 scenarios to either reach the  $T_{limit}$  or to saturate.  $C_{A,2}$  is the average cost of the solutions achieved by TOMLAB and  $T_{A,2}$  is the average time it was allowed to run. Sometimes, the tool returns a zero value, implying that none of the variables were set. We ignore such conditions.  $I$  represents the percentage improvement of our algorithm over the TOMLAB tool. In Fig. 2, we plot the cost of the network obtained from SA heuristic and that obtained from TOMLAB/CPLEX.

It can be seen that for small instances, the performance of TOMLAB is superior. However, for  $n > 25$ , our algorithm starts performing better and for  $n = 300$ , the performance gap is almost 26%. This is because, for smaller instances, the number of variables to be set by interger programming are very small. Once the number of variables starts to increase, the performance of the tool starts deteriorating.

In Table II, we compare the two construction heuristics, Clustering-based and MST-based. In both the implementations, we make use of precedence list.  $C_{A,1}$  is the average cost function value of MST-based heuristic,  $T_{A,1}$  is its average time. Similarly,  $C_{A,2}$  and  $T_{A,2}$  are the average cost and average

TABLE I  
COMPARISON OF OUR ALGORITHM (CONSTRUCTION BASED ON CLUSTERING) WITH INTERGER PROGRAMMING METHODS

$n$	$r$	$C_{A,1}$	$T_{A,1}$	$C_{A,2}$	$T_{A,2}$	$I$
20	3	5.07	6.13	4.95	6.20	-2
	4	5.80	5.44	5.57	5.50	-4
25	4	5.60	6.87	5.55	6.90	-0.9
	5	6.64	6.09	6.80	6.10	2
30	5	6.44	6.99	6.80	7.00	5
	6	7.55	6.46	7.67	6.50	1.5
40	6	7.92	7.10	8.46	7.10	6.3
	7	8.37	6.45	8.84	6.50	5.3
50	6	8.74	8.15	9.50	8.20	8
	8	9.73	7.63	10.51	7.70	7.4
80	7	10.34	9.97	11.04	10.00	6.3
	9	10.61	9.51	11.56	9.50	8.2
100	8	11.06	9.42	12.55	9.50	11.8
	10	13.32	7.62	15.03	7.70	11
150	12	13.66	12.31	15.87	12.30	13.9
	15	15.73	10.07	17.15	10.10	8.2
200	14	16.43	17.95	18.72	18.00	12.2
	18	16.43	14.99	18.90	15.00	13.06
300	15	18.27	106.85	24.68	110.00	25.9
	20	19.33	86.10	25.99	90.00	25.6

TABLE II  
COMPARISON OF MST BASED HEURISTIC WITH CLUSTERING BASED HEURISTIC. HERE, WITH USE OF PRECEDENCE LIST

$n$	$r$	$C_{A,1}$	$T_{A,1}$	$C_{A,2}$	$T_{A,2}$
20	3	5.24	6.58	5.07	6.13
	4	5.43	6.11	5.80	5.44
25	4	5.25	6.97	5.60	6.87
	5	5.27	6.86	6.64	6.09
30	5	7.20	7.06	6.44	6.99
	6	7.23	6.99	7.55	6.46
40	6	7.85	7.32	7.92	7.10
	7	8.92	7.22	8.37	6.45
50	6	8.51	8.08	8.74	8.15
	8	9.22	7.71	9.73	7.63
80	7	10.54	10.24	10.34	9.97
	9	12.92	9.46	10.61	9.51
100	8	14.03	11.53	11.06	9.42
	10	15.75	10.98	13.32	7.62
150	12	18.54	16.68	13.66	12.31
	15	19.09	15.20	15.73	10.07
200	14	18.79	23.60	16.43	17.95
	18	21.88	22.60	16.43	14.99
300	15	24.66	89.10	18.27	106.85
	20	28.32	107.91	19.33	86.10

time taken by the Clustering-based heuristic. We can see that for small instances ( $n < 100$ ), the performance of both the algorithms is similar. However, for  $n > 100$ , there is a clear distinction and Clustering based heuristic performs better than the MST based one. This is probably because, as the number of nodes increases, the number of overlaps in the MST based heuristic also increase. Hence, it would take a much longer time for it to remove all the overlaps. Hence, the difference in the performance.

In Table III, we try and understand the importance of the Precedence list in the algorithm. We compare the performance

TABLE III  
COMPARISON OF THE PERFORMANCE OF AN MST-BASED HEURISTIC IN  
THE PRESENCE AND ABSENCE OF A PRECEDENCE LIST

$n$	$r$	$C_{A,1}$	$T_{A,1}$	$C_{A,2}$	$T_{A,2}$
20	3	5.30	5.80	5.24	6.58
	4	5.60	5.78	5.43	6.11
25	4	5.45	5.91	5.25	6.97
	5	5.96	6.11	5.27	6.86
30	5	7.94	5.97	7.20	7.06
	6	7.77	5.97	7.23	6.99
40	6	9.49	6.04	7.85	7.32
	7	10.49	5.98	8.92	7.22
50	6	11.13	6.20	8.51	8.08
	8	11.94	6.09	9.22	7.71
80	7	15.26	6.52	10.54	10.24
	9	17.74	6.41	12.92	9.46
100	8	17.84	6.63	14.03	11.53
	10	18.56	6.66	15.75	10.98
150	12	22.91	7.47	18.54	16.68
	15	26.95	3.45	19.09	15.20
200	14	22.57	2.14	18.79	23.60
	18	32.91	8.23	21.88	22.60
300	15	29.89	36.30	24.66	89.10
	20	40.92	35.55	28.32	107.91

of the MST-based heuristic when the list is used and when the list is not used. We choose MST because, as the MST method is prone to overlaps, the concept of “comfort” of a node would make more sense here. Hence, the use of the list ensures that the number of overlaps are reduced. This can be verified from the results. In the table,  $C_{A,1}$  and  $T_{A,1}$  are the cost and time-to-solution, respectively, of the heuristic without the list and  $C_{A,2}$  and  $T_{A,2}$  are the same, when the algorithm is used with the list. A clear difference is seen in the performance of the two. Using the precedence list, the performance is much greater and for larger instances, the difference is more than 30%. Thus, the utility of the list has been established.

## VI. CONCLUSION

We have presented an algorithm for the efficient and fast design of 2-level hierarchical ring networks. This algorithm was based on Simulated Annealing. We have also presented two construction heuristics for this problem. While one was based on k-means clustering of the set of nodes, the other was based on the construction of the Minimum Spanning Tree. The performance of both of them was found to be equal for small instances. However, for large scenarios, the performance of the former was better. We also compared the performance of this algorithm with an integer-programming method. Our algorithm performed significantly better than the tool. We also introduced the concept of a Precedence list to randomly select nodes in the neighbourhood for movement in SA algorithm and this lead to significant performance improvement.

In the future, we wish to improve on the mathematical models defining the problem. The number of variables in use is large and this is detrimental to the performance of the algorithms. Also, we wish to develop a mathematical model and devise efficient algorithms for the k-level hierarchical ring networks. Another direction for future work would be to come up with a capacity constrained model along with the link costs model for k-level hierarchical ring networks.

## REFERENCES

- [1] Kirkpatrick, S., C. D. Gelatt Jr., M. P. Vecchi, Optimization by Simulated Annealing, Science, 220, 4598, 671-680 (1983)
- [2] Cerny, V., Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm, J. Opt. Theory Appl., 45, 1, 41-51 (1985)
- [3] Altinkemer, K., Topological Design of Ring Networks., Comput. Oper. Res. 21, 421431 (1994)
- [4] Shi, J., Fonseca, J. P., Hierarchical Self-Healing Rings, IEEE/ACM Trans. Netw., 3(6), 690-697 (1995)
- [5] Luss, H., Rosenwein, M.B., Wong, R.T., Topological Network Design for SONET Ring Structure, IEEE Trans. Syst. Man Cybern.-Part A: Syst. Hum. 28(6), 780790 (1998)
- [6] Xu, J., Chiu, S.Y., Glover, F., Optimizing a Ring-Based Private Line Telecommunication Network Using Tabu Search., Manag. Sci. 45(3), 330345 (1999)
- [7] Proestaki, A. and Sinclair, M. C., Design and Dimensioning of Dual-Homing Hierarchical Multi-Ring Networks, IEEE Proceedings-Communications 147(2), 96-104 (2000)
- [8] Fortz, B., Soriano, P. and Wynants, C., A Tabu Search Algorithm for Self-Healing Ring Network Design, Eur. J. Oper. Res. 151(2), 280-295 (2003)
- [9] Goldschmidt, O., Laugier, A. and Olinick, E. V., SONET/SDH Ring Assignment with Capacity Constraints, Discrete Applied Mathematics 129(1), 99-128 (2003)
- [10] Thomadsen, T., Stidsen, T., Hierarchical Ring Network Design Using Branch-and-Price, Telecomm. Syst., 29(1), 61-76 (2005)
- [11] Üster, H., Kumar, S. K. S., Algorithms for the Design of Network Topologies with Balanced Disjoint Rings, J. Heuristics, Springer, 16, 37-63 (2010)
- [12] Kshirsagar, K., Kaza, K. and Rajan, K., Hierarchical Ring Topology for E-UTRAN, to appear in Proc. 11th Intern. Conf. on Telecomm. (2011)
- [13] Christofides, N., Worst-case Analysis for a New Heuristic for the Travelling Salesman Problem, Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University (1976)
- [14] Kang, D., et al, Design of Local Networks Using USHRs, Telecommunication Systems, 14, 197-217 (2000)
- [15] Deng, X., Li, G., Zang, W. and Zhou, Y., A 2-Approximation Algorithm for Path Coloring on a Restricted Class of Trees of Rings, J Algorithms, 47, 1-13 (2003)
- [16] Beauquier, B., Perennes, S., Toth, D., All-to-all routing and coloring in weighted trees of rings, in Proc. SPAA (1999)
- [17] Ravindran, G. and Stumm, M., A Performance Comparison of Hierarchical Ring- and Mesh-connected Multiprocessor Networks, Proc. Intl. Symp. on High Performance Computer Architecture (1997)
- [18] Hamacher, C. and Jiang, H., Hierarchical Ring Network Configuration and Performance Modeling, IEEE Trans. Comput., 50(1), 1-12 (2001)
- [19] Otten, R. H. J. M and van Ginneken, L. P. P., The Annealing Algorithm, Kluwer Academic Publishers (1989)
- [20] Hajek, B., Cooling Schedules for Optimal Annealing, Math. Oper. Res., 13, 311-329 (1988)
- [21] Noon, C. E., The Generalized Travelling Salesman Problem, PhD Thesis, Dept. Ind. Oper. Res., Univ. Tennessee (1988)
- [22] TOMLAB: <http://tomopt.com/tomlab/>