Optimizing the Distributed Network Monitoring Model with Bounded Bandwidth and Delay Constraints by Genetic Algorithm

Xianghui Liu[1], Jianping Yin[1], Zhiping Cai[1], Xueyuan Huang[2], and Shiming Chen[2]

[1] School of Computer Science, National University of Defense Technology, Changsha City, Hunan Province, 410073, PRC
LiuXH@tom.com
[2] Ende Technology, Changsha City, Hunan Province, 410073, PRC

Abstract. Designing optimal measurement infrastructure is a key step for network management. In this work the goal of the optimization is to identify a minimum aggregating nodes set subject to bandwidth and delay constraints on the aggregating procedure. The problem is NP-hard. In this paper, we describe the way of using Genetic Algorithm for finding aggregating nodes set. The simulation indicates that Genetic Algorithm can produce much better result than the current method of randomly picking aggregating nodes.

1 Introduction

The explosive growth of Internet has emerged a massive need for monitoring technology that will support this growth by providing IP network managers with effective tools for monitoring network utilization and performance[1][2]. Monitoring of the network-wide state is usually achieved through the use of the Simple Network Management Protocol (SNMP) with two kinds of entities: one management center and some monitoring nodes. The management center sends SNMP commands to the monitoring nodes to obtain information about the network and this function is performed by a centralized component responsible for aggregating all monitoring nodes [3]. Yet such processing queries have some inherent weaknesses. Firstly it can adversely impact router performance and result in significant volumes of additional network traffic. Secondly aggregating procedure is its time dependency. The support of knowledge of the up-to-date performance information requires the establishment of reliable, low delay and low cost aggregating routes [4] [5].

In above traditional centralized monitoring system, although the center provides a network-wide view but has some inherent weaknesses as being pointed out and not suitable for large scale network. Taking into account the issues of scalability and network-wide view for large service provider networks, an ideal monitoring architecture is a hierarchical system which implied that there is a management center but the resource intensive nodes such as polling are distributed. Between the management center and the monitoring nodes, there exists a set of aggregating nodes.

L. Wang, K. Chen, and Y.S. Ong (Eds.): ICNC 2005, LNCS 3611, pp. 913 – 921, 2005.

The aggregating nodes are distributed and each node is responsible for an aggregating domain consisting of a subset of the network nodes. Information gathered from the individual monitoring nodes is then aggregated. Such a hierarchical architecture overcomes the weaknesses while still maintaining a network-wide view [4] [5].

In particular, the most recently works addresses the problem of minimizing the number of aggregating nodes while keeping the aggregating bandwidth or delay within predefined limits individually [4] [5]. And all these problems are NP-Hard with solutions to this problem by using heuristics based on the aggregating load and the maximum assignment of monitoring nodes. The difficulties of using heuristics for optimal distributed network monitoring model is that after a possible aggregating node is picked, the algorithm tries to assign the maximum number of un-assigned monitoring nodes to the it without violating bandwidth and delay constraints. Unfortunately the general problem that assigns the maximum number of un-assigned monitoring nodes without violating constraints is also NP-Hard and all the heuristics only consider some special situation now[4] [5].

As the idea of using Genetic Algorithm to provide solutions to difficult NP-Hard optimization problems has been pursued for over a decade and have some significant results. There are no polynomial-time algorithms (yet) that solve NP-Complete problems, finding approximate solutions for these problems is usually made more efficient when we use the GA concept. (Although a main drawback is that we are not guaranteed to be given an optimal solution, even if we spend a large amount of time running this genetic process.)

1. GA provides approximate solutions to several problems.
2. GA is a valid approach, since we are often times willing to settle for approximate solutions.
3. GA allows one to spend as much time as is allowed to find a solution, while providing the "best" solution so far, if terminated.

In this paper, we consider optimizing distributed monitoring modal with bounded bandwidth and delay constraints problem by Genetic Algorithm [6] [7] [8].

2 Problem Formulation

We represent the whole monitoring domain of our model as an undirected graph $G(V,E)$, where $V=\{v_1,v_2,\cdots v_n\}$ is the set of all nodes or routers that are in the monitoring domain and. $E\subseteq V\times V$ represents the set of edges. The node set $S_m(S_m\subseteq V\wedge S_m\neq\Phi)$ represents the monitoring nodes in the monitoring domain. Each node $v(v\in S_m)$ generates an aggregating traffic of w_i *bps*. This aggregating traffic is destined to the relative aggregating node which has been assigned to. We define function $L:E\to R^+$ and $B:E\to R^+$ which assign a non-negative weight to each link in the network and represent the actual aggregating bandwidth used and the amount of link bandwidth allocated for aggregating traffic for each of the edges. And we also define edge-delay function $D:E\to R^+$ which assigns a non-negative weight to each of the edges. The value $D(e)$ associated with edge $e\in E$ is a measure (estimate) of

total delay that packets experience on the link. Let the set $E(Path(u,v))=\{e_1,e_2,\cdots,e_m\}$ represents the links in the path between node *u* and *v*.

The optimal aggregating node location and monitoring node assignment problem can therefore be stated as follows: Given a network $G(V,E)$, determine (1) a minimum subset of nodes $S_a(S_a \subseteq V)$ on which to place aggregating node such that the bandwidth constraint on each and every link $L(e) \le B(e)$ is satisfied (where $B(e)$ is the maximum bandwidth that can be used for aggregating on link *e*) and the delay constraint on every node $v(v \in S_m)$ satisfy $Delay(Path(v,w)) \le \delta$ (where δ is the maximum delay that can be used for aggregating by a node defined as w). (2) A mapping λ which maps a monitoring node to its aggregating node. That is, for any node $v(v \in S_m)$, if $\lambda(v)=w$, then node v is assigned to the aggregating node w. Note in some situation, we can use additional constraints to decide whether the monitoring node v can be aggregated by itself.

Now we define some variable to describe the integer program formulation about the problem. The binary variable x_{ij} indicates whether monitoring node v_i is aggregated by node v_j, where $v_i \in S_m$ and $v_j \in V$. The binary variable b_e^{ij} indicates whether edge e belongs to the $Path(v_i,v_j)$ between node v_i and v_j. The binary variable y_j indicates whether node v_j is an aggregating node or not .The problem minimizing the number of aggregating nodes in a given network subject to delay constraints can naturally expressed as an integer programming formulation:

The objective is: $Minimize \sum_{j=1}^{|V|} y_j$ and the constraints are below:

$$\sum_{j=1}^{|V|} x_{ij} = 1(\forall v_i \in S_m) \tag{1}$$

$$x_{ij} \le y_j(\forall v_i \in S_m, \forall v_j \in V) \tag{2}$$

$$\sum_i \sum_j b_e^{ij} L(v_i) x_{ij} \le B(e)(\forall v_i \in S_m, \forall v_j \in V, e \in E) \tag{3}$$

$$\sum_{e \in E} b_e^{ij} D(e) x_{ij} \le \delta(\forall v_i \in S_m, \forall v_j \in V) \tag{4}$$

$$x_{ij} \in \{0,1\}(\forall v_i \in S_m, \forall v_j \in V) \tag{5}$$

$$y_j \in \{0,1\}(\forall v_j \in V) \tag{6}$$

The first constraint makes sure that each monitoring node v_i is aggregated by exactly one aggregating node. The second constraint guarantees that a node v_j must be an aggregating node if some other monitoring node v_i is assigned to (aggregated by) it. The third constraint ensures the aggregating traffic on every link e not exceed the predefined bandwidth limits $B(e)$. The fourth constraint ensures that delay during aggregating procedure not exceeds the delay constraint on the path between each monitoring node and its aggregating node.

It is well-known that the integer programming formulation has an exponential running time in the worst case. In the previous work the greedy algorithm normally

consists of two steps. In the first step the algorithm calculate out the maximum number of monitoring nodes satisfying the bandwidth or delay constraint when they are assigned to an aggregating node, and the set of these monitoring nodes is called candidate monitoring set of the relative node. In the second step algorithm greedily repeatedly picks an additional aggregating node (based on the greedy selection criteria) if there are any monitoring nodes still present in the network that does not have an aggregating node assigned to it. After an aggregating node is picked, the algorithm assigns candidate monitoring set to it without violating bandwidth or delay constraint. The repeat will interrupt when all monitoring nodes have been assigned, and the approximate aggregating node set includes all pickup additional aggregating nodes. Unfortunately the general problem that assigns the maximum number of un-assigned monitoring nodes without violating constraints is also NP-Hard and all the heuristic algorithm only consider some special situation. So the below we consider using Genetic Algorithm to solve the problem.

3 Evolution and Genetic Algorithm

Evolutionary algorithms are optimisation and search procedures inspired by genetics and the process of natural selection. This form of search evolves throughout generations improving the features of potential solutions by means of biologically inspired operations. On the ground of the structures undergoing optimisation the reproduction strategies, the genetic operators' adopted, evolutionary algorithms can be grouped in: evolutionary programming, evolution strategies, classifier systems, genetic algorithms and genetic programming.

The genetic algorithms behave much like biological genetics [8]. The genetic algorithms are an attractive class of computational models that mimic natural evaluation to solve problems in a wide variety of domains [9] [10]. A genetic algorithm comprises a set of individual elements (the population size) and a set of biologically inspired operators defined over the population itself etc. Genetic algorithms manipulate a population of potential solutions to an optimisation (or search) problem and use probabilistic transition rules. According to evolutionary theories, only the most suited elements in a population are likely to survive and generate offspring thus transmitting their biological heredity to new generations. A genetic algorithm maps a problem onto a set of strings (the chromosomes) and each string representing a potential solution. The three most important aspects of using genetic algorithms are: (1) definition of the objective function, (2) definition and implementation of the genetic representation, and (3) definition and implementation of the genetic operators.

There are a lot of list heuristic methods which are used to scheduling nodes onto parallel processors. Most of them give a good solution problem. Example, each node graph is assigned a priority, and then added to a list of waiting nodes in order of decreasing priority. As processors become available the node with the highest priority is selected from the list and assigned to the most suited processor. If more than one node has the same priority a node is selected randomly.

Initialisation - an initial population of the search nodes is randomly generated. The strings encoding mechanism should map each solution to a unique string. The

encoding mechanism depends on the nature of the problem variables and it use for representing the optimisation problem's variables. The representation is unique. In some cases the variables assume continuous values, while in other cases the variables are binary. It can be integer parameters, real-valued parameters, vectors of parameters, Gray code, dynamic parameter encoding etc. The fitness values of each node are calculated according to the fitness function (objective function). The fitness function provides the mechanism for evaluating each chromosome in the problem domain. It is always positive. Three operators are needed to achieve this selection, crossover and mutation. The selection criterion is that string with higher fitness value should have a higher chance of surviving to the next generation. A quality measure for the solutions (fitness function) of the problem is known. Fitter solutions survive, while weaker ones perish. There are many different models of selection. The most popular selection in genetic algorithms is fitness proportionate selection, rank selection, tournament selection and elitist selection. After selection comes crossover.

The crossover operator takes two chromosomes (parents) and swaps part of their genetic information to produce new chromosomes (child). The offspring (child) keep some of the characteristics of the parents. One point crossover involves cutting the chromosomes of the parents at a randomly chosen common point and exchanging the right - hand – side sub-chromosomes. In two – point crossover chromosomes are thought of as rings with the last and the first gene connected. The rings are cut in two sites and the resulting sub-parts are exchanged. In uniform crossover each gene of the offspring is selected randomly from the corresponding genes of the parents. Crossover is applied to the individuals of a population with a constant probability, usually from 0.5 to 0.95.

Mutation consists of making (usually small) alterations to the values of one or more genes in a chromosome. In genetic algorithms, mutation is considered a method to recover lost genetic material. When we have the network topology graph, we need use topological sorting. Topological sorting consists of finding some global aggregating node set with these local constraints.

4 Proposed Algorithm

Now, we will consider initialisation, crossover, and mutation and evaluation algorithm.

With initialisation we will make population of solutions. Let be N population size and Z will be number of node in network topology graph. Randomly we choose the one of processors from set of [1,P] where P is total number of processors and then add the node from the list of node sortie by indexes on increasing order.

The chromosome is consisting from P sorted arrays. Example, let be P=3 and Z=10. One example of string looking as:

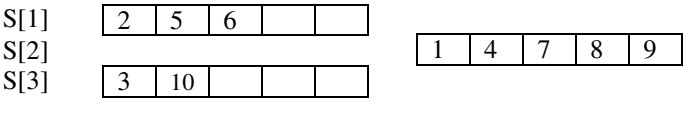


Fig. 1. Example of string (work with indexes (sorted by width) of nodes)

This is only chromosome (string) but not scheduling. On that way and with number of iterations we have defined algorithm of initialisation.

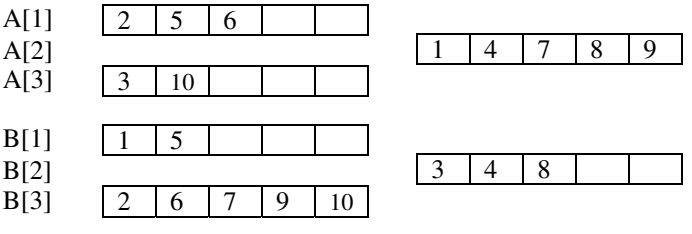
The crossover operator use two strings randomly choose (choose one at random node) T_i from one of two sets and put on the new string. Precedence relation must be kept and whole time we work with indexes of nodes (getting by topological sorting). If T_i element the same set at both parents then T_i is coping on the same place for new string (child). If we randomly choose two same parents (strings A=B) and if one of parents e.g. A the best string we use operator mutation on the second B string. It is elitism. Else, we mutate the first set and child generate randomly.

This algorithm performs the crossover operation on two strings (A and B) and generates new string.

```
Crossover (A; B) {
  If (A==B) {//elimination duplicate
    If (A is the best string) mutation (B);
    Else mutation (A);
    Random generate child;
  Return
}
  For (i=1;i<=Z;i++){
    If (Ti∈P on both parents)
      Ti copy on the same place on the child
    Else
      Ti below one from sets of parents (randomly);
  }
}
```

Increasing order indexes of the nodes must be kept.

Before crossover operation:



After crossover operation (child):

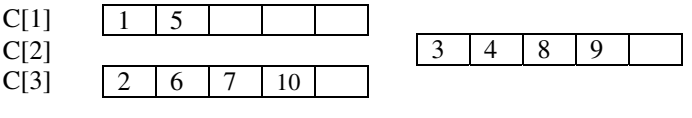


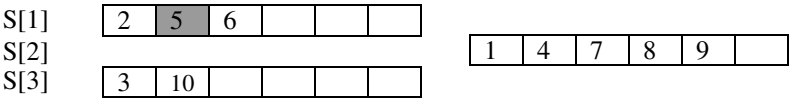
Fig. 2. Example of crossover operator

Next, what we must to do is define the mutation operator. We first generate two randomly chosen numbers *r* i *q* from the sets *[1,P]*. The condition for that is that: a) *r* ≠ *q*, and b) set *r* aren't empty.

After that from set *r,* choose one node at random and remove him in the set *q*. We must take in the account that the node which we move, must be put on the place that indexes of node be ordered by increasing.

Let be r=1 and q=2 and randomly choose the node has the index 5.

Before mutation:



After mutation:

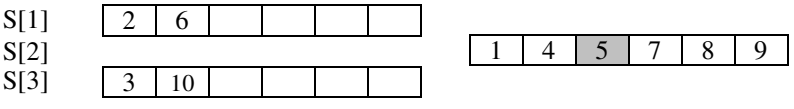


Fig. 3. Example of mutation operator

The below is the evaluation algorithm.

```
Evaluation () {
  For (i=1;i<=P;i++) FTP[i]=0;// reset FTP for all
processors
  For (i=1;i<=Z;i++){
// Ti∈{sets of nodes processor p},p∈[1,P]
    FTP[p] += duration (Ti);
    For (j=1;j<=P;j++){
      If (j == p) continue;
      // the precedence relations are maintained in
this line
      If (((Tx∈j)<Ti ) && (FTP[j]>pom))
      /* x is the biggest index of nodes set p (nodes
which execution onto processor p), and that content the
condition x<j.*/
          FTP[p] = FTP[j] + Ti;
    }
  }
  FT = max i∈[1,P]{FTP[i]};
}
```

The fitness function for the multiprocessor scheduling problem in our genetic algorithms is finishing time a besides it can be also throughput and processor utilization. Finishing time of a schedule is defined as follows: $FT=max_{i\in[1,P]}\{FTP[i]\}$ where $FTP[i]$ is the finishing time for the last node in processor i.

5 Simulation

In this section, we evaluate the performance of proposed algorithm with the heuristic algorithm on several different topologies and parameter settings. For simplicity, we make the reasonable assumption of shortest path routing.

The network graph used in this study is the Waxman model. We generate different network topologies by varying the Waxman parameter β for a fixed parameter $\alpha = 0.2$ .The varying β gives topologies with degrees of connectivity ranging from 4 to 8 for a given number of nodes in the network. Each link allocates a certain fraction of their bandwidth for aggregating traffic based on the capacity constraint imposed for the link. In our simulation, the fraction is the same for all links and the value is 5% and 10% respectively. The delay for every link changes from 0.1 to 2.0 with a fix delay tolerance parameter $\delta = 10$. Simulation results presented here are averaged over 5 different topologies. Our performance metrics are (1) total number of aggregating nodes required, and (2) fraction of total bandwidth consumed for aggregating.

Random Picking of Aggregator: the heuristic is to pick a possible aggregating node from V randomly. This heuristic serves as a base-line comparison for the neural network algorithm proposed in the paper. Once we select an aggregating node, we need to determine the set of monitoring nodes to be assigned to that it. Ideally, we would like to assign maximum number of unassigned monitoring nodes to a new aggregating node. The algorithm is present in paper [4].

The simulation shows the Hopfield network have more better result than the randomly method.

Nodes	**Algorithm**	**Link Fraction**	β			
			0.05	0.10	0.15	0.20
200	Random Picking	5%	13.973%	9.257%	7.193%	5.067%
		10%	17.157%	14.928%	13.016%	9.942%
	GA	5%	11.265%	8.364%	6.327%	3.985%
		10%	12.943%	11.406%	9.158%	9.043%
400	Random Picking	5%	18.278%	14.487%	11.975%	10.013%
		10%	21.345 %	19.857%	15.475%	13.976%
	GA	5%	15.433 %	12.587%	10.247%	8.083%
		10%	19.164 %	16.169%	14.724%	12.867%

Fig. 4. The result for simulation

6 Conclusion

The primary motivation for work to design good measurement infrastructure it is necessary to have a scalable system at a reduced cost of deployment. As the model is NP-Hard and the current heuristics algorithm is that after a possible aggregating node is picked, the algorithm tries to assign the maximum number of un-assigned monitoring nodes to it without violating bandwidth and delay constraints. Unfortunately the general problem that assigns the maximum number of un-assigned monitoring nodes without violating constraints is also NP-Hard.

In this paper, we have demonstrated that Genetic Algorithms techniques can compete effectively with more traditional heuristic solutions to practical combinatorial optimization problems, but they are not guaranteed to perfectly solve a problem (esp. in polynomial time).

References

[1] A. Asgari, P. Trimintzios, M. Irons, G. Pavlou, and S. V. den BergheR. Egan.: A Scalable Real-Time Monitoring System for Supporting Traffic Engineering. In: Proceedings of IEEE Workshop on IP Operations and Management, IEEE, New York (2002)

[2] Y. Breitbart, C. Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz.: Efficiently Monitoring Bandwidth and Latency in IP Networks. In: Proceedings of IEEE Infocom 2002, IEEE, New York (2002)

[3] D. Breitgand, D. Raz, and Y. Shavitt.: SNMP GetPrev: An efficient way to access data in large MIB tables. In: IEEE Journal of Selected Areas in Communication, Volume 20, No 4, IEEE, New York , (2002), 656–667,

[4] L. Li, M. Thottan, B. Yao, S. Paul.: Distributed Network Monitoring with Bounded Link Utilization in IP Networks. In: Proceedings. of IEEE Infocom 2003, IEEE, San Francisco (2003)

[5] Liu, Xiang-Hui; Yin, Jian-Ping; Lu, Xi-Cheng; Cai, Zhi-Ping; Zhao, Jian-Min.: Distributed network monitoring model with bounded delay constraints. In: Wuhan University Journal of Natural Sciences, Volume 9,No 4, (2004) 429-434

[6] Ricardo C. Correa, Afonso Ferreira, Pascal Rebreyend.: Scheduling Multiprocessor Node with Genetic Algorithm. In: IEEE Transactions on Parallel and Distributed systems. Volume 10, No 8, IEEE, New York (1999)

[7] Albert Y. Zomaya, Chris Ward, Ben Macey.: Genetic Scheduling for Parallel Processor Systems: Comparative studies and Performance Issues. In: IEEE Transactions on Parallel and Distributed systems. Volume 10, No 8, IEEE, New York (1999)

[8] G.N. Srinivasa Prasanna , B.R. Musicus.: Generalized Multiprocessor Scheduling and Applications to Matrix Computations. In: IEEE Transactions on Parallel and Distributed systems, Volume 7, No 6, IEEE, New York (1996)

[9] C.W. Ahn and R. S. Ramakrishna.: A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations. IEEE Transactions on Evolutionary Computation, Volume 6, Issue 6, IEEE, New York (2002), 566–579

[10] D.E. Goldberg. Genetic Algorithms in Search.: *Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, (1989)