Topological Optimization of Reliable Networks using A-Teams

Benjamín Barán
bbaran@cnc.una.py

Fabián Laufer[*]
flaufer@conexion.com.py

National Computer Center
National University of Asuncion
P.O.BOX 1439
University Campus of San Lorenzo – Paraguay
Tel./Fax: (595)(21) 585-619 and 585-550
http://www.cnc.una.py

ABSTRACT

This paper presents an *Asynchronous Team Algorithms* (A-Team) implementation, in a parallel heterogeneous asynchronous environment, to optimize the design of reliable communication networks given the set of nodes an possible links.
The proposed Team combines parallel *genetics algorithms (GA)*, with different reliability calculation approaches in a network of personal computers, proving good experimental results with considerable speedup.

Key Words: A-Teams, genetic algorithm, Monte Carlo simulation, network design, network reliability.

1. INTRODUCTION

Network planning is concerned with the design of sufficiently reliable networks at reasonable cost, to deliver high capacity and speed [1]. Because of the lack of good network designing tools, engineers have been using their experience and intuition to design networks in many fields as telecommunications, electricity distribution, gas pipeline and computer networks. But the old *trial & error* approach is not useful anymore for very large-scale networks, as the ones being designed nowadays. Clearly, a computational automatic tool is required.

For the present work, a reliable network design problem is stated as *all-terminal* network reliability (also known as *uniform* or *overall* reliability). In this approach, every pair of nodes needs a communication path to each other [2,3]; that is, the network forms at least a *spanning tree.* Thus, the primary design problem is to choose enough links to interconnect a given set of nodes with a minimal cost, given a minimum network reliability to be attained. This minimization design problem is NP-hard [4], and as a further complication because the calculation of all-terminal reliability is also NP-hard.

Although several papers have been published on this problem or similar ones, no known method is efficient enough to deal with real large networks. In this context, Jan et al. [5] developed a branch and bound search to design fully connected networks, reporting results up to 12 nodes. Ventetsanopoulos and Singh [6] proposed an algorithm to create an initial population for applying branch and bound. Atiqullah and Rao [7] used a deterministic simulated annealing with exact calculation for very small networks. Another simulated annealing technique was introduced by Pier et al. [8]. There are some approaches using Tabu search, like the ones by Glover et al. [9], Beltran & Skorkin-Kapov [10] and, Koh & Lee [11].

Several GA approaches have already been used for different network design problems. Series and parallel systems have been treated in [12-14]. Kumar et al. [15] calculated the *all-terminal* reliability with maximum network diameter constraint, and expanded their work in [16]. Other related problems can be found in Davis et al. [17] and in the work by Abouali et al. [18-19]. Walter and Smith [20] used GA to address optimal design of a pipe network. Deeter and Smith [21] presented a GA approach to minimize cost of a 5-node network with all-terminal reliability constraint. Dengiz et al. [22] addressed the same problem using a fairly standard GA implementation. Lately, Dengiz et al. [23] have introduced a GA that scale-up the size of tractable networks, but they solve complete networks no larger than 11 nodes, a small number for real size problems.

Considering the complexity of designing reliable networks, and the amount of different published method, this problem seems to be a good candidate for Team algorithms (TA). TA is a technique, which combines distinct algorithms interacting in the solution of the same global problem [24]. Team Algorithms can be naturally implemented in parallel assigning different sub-problems to each processor of an asynchronous distributed system, like a computer network. This parallel asynchronous combination of different algorithms is known as A-Team (Asynchronous Team) [25-27]. Baran et al. [28] have achieved excellent results applying this technique for hydroelectric optimization, a different engineering goal with a similar mathematical framework. Therefore, this paper proposes the implementation of an A-Team, for the topological optimization of telecommunication networks, subject to reliability constraints.
The paper is organized in the following way: Section 2 states the problem. The proposed method is introduced in section 3, with section 4 presents experimental results. The conclusion is left for section 5.

[*] Also at the "Universidad Católica *Nuestra Señora de la Asunción*" – Paraguay.

2. STATEMENT OF THE PROBLEM

A network is modeled by a probabilistic undirected graph **G**=(N, L, p), in which N represents the set of nodes, L a given set of possible links, and p the reliability of each link. It is assumed one bi-directional link between each pair of nodes; that is, there is no redundancy between nodes.

The optimization problem may be stated as:

$$Minimize \quad Z = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} x_{ij} \qquad (1)$$

$$Subject\,to: \quad R(x) \geq R_0$$

where x_{ij} is a decision variable {0, 1}, c_{ij} is the cost of a link (i, j), *R(x)* is the network reliability, and R_0 is the minimum reliability requirement.

To solve the problem, the following assumptions are made:

- The N nodes are perfectly reliable. A problem with a node may be simulated by the failure of its incident links.
- The cost c_{ij} and the reliability p_{ij} of each link (i,j) are known.
- The links have two states: either operational (x_{ij} = 1) or failed (x_{ij} = 0).
- The links failures are independent.
- No repair is considered.
- Two-connectivity is required.

3. PROPOSED A-TEAM

The main algorithm consists of two kinds of processes, a Coordinator and the PGAs (Parallel Genetic Algorithms). There is only one coordinator, which is responsible of creating the PGA processes, collect their results and take note of the global statistics. The PGAs do the real work. Once the coordinator initializes all the processes, each PGA computes its solutions, broadcasts its partial results to the others, and receives what its peers have sent to it.

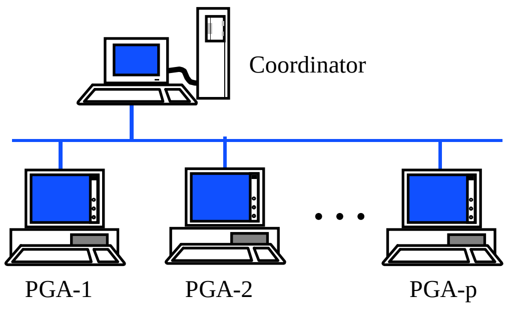

Figure 1: Asynchronous Team (A-Team) implementation.

3.1. The Coordinator

First, the coordinator spawns *P* slaves processes, where *P* depends on the number of available processors, and the relative performance between them. Work balance is required for an efficient implementation. There are several ways to balance the computational load using parallel genetic algorithms (PGA), as example, with different population size in each processor. For the present proposal, load balance is achieved by running a different number of similar processes, depending upon the relative performance of the processor.

The second task of the coordinator is to gather partial results sent by the PGAs, to check if a received solution is better than the current one, and if this is the case, to replace it. Also, it is the coordinator responsibility to take the required global statistics. For instance, the global time is measured here. At the same time, the coordinator does the entire user interface work.

Finally, the coordinator checks the finishing criteria. For the present implementation, it stops after being informed that all the PGAs have already satisfied a given finishing condition.

```
START
SPAWN P PGAs
stop_counter=0
DO WHILE( stop_condition < P )
        RECIVE candidate network $N_r$ and flag
        IF ($N_r$ is better than $N_{best}$ ) $N_{best}$ = $N_r$
        IF ( flag=STOP_CONDITION_REACHED)
                stop_counter = stop_counter+1
        ENDIF
ENDDO
KILL P PGAs
END
```

Pseudocode 1: Coordinator process.

3.2.Genetic Algorithm

The proposed GA is based on a previous work by Dengiz et al. [2] with the following main features:

- All the network solutions keep the two connectivity constraints; that is, there should be at least two links incident to each node.
- A specialized initialization is used to enhance the efficiency of the search.
- The crossover and mutation operations are specialized. They are local search operators based on set operations (union, intersection, and subtraction).
- A repair algorithm is utilized to keep the networks under the two-connectivity constraint.
- There are two different approaches to calculate network reliability. First, an upper bound of all the network candidates included in the population is efficiently calculated. Then, a computational *expensive* Monte Carlo simulation is used to get a real good approximation of the actual all-terminal reliability of the best candidates. That way, a trade off between computational effort and accuracy is achieved.

Encoding

In the present proposal, a network is encoded by a string of bits, where each bit represents the operability of a link. The position of the bit in the string matches the label of the link (see Figure 2). The coding is implemented at bit level, in this way, a network with up to 16 links can be saved as an integer of 16 bits, where the highest order bit is labeled 1, and the bit of lowest order is labeled 16. For Instance: the network of

Figure 2 is encoded by (reading from left to right): 0101100000000000. Thus, the former network is represented by the integer $2^{14} + 2^{12} + 2^{11} = 22528$.

In case of networks with more than 16 links, an array of integer is used instead. As example: if a network has 20 links, an array of two integers is used. The first integer represents the first 16 links. The highest order bit of the second integer represents link 17, and so on.

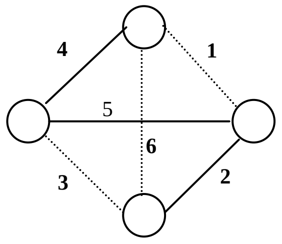

Figure 2: The dotted solid lines show where the lines are up, and the dotted line are the failure ones.

Parallel Implementation

There are several ways to implement a GA in a parallel asynchronous environment. For the present work, identical processes, called PGAs (*Parallel Genetic Algorithms*) are assigned to different processors. Each PGA has its own population, randomly generated by each independent process. The population size is maintained constant in each PGA process, even though it may receive any number of network candidates from the peer processes. For that porpoise, a genetic *selection* operator is used after asynchronously receiving any number of candidates, to choose between the own population and the incoming candidates (see Pseudocode 2). That way, the interaction between processes is performed, interchanging good candidate networks. Every *g* generations, or when the PGA find a new best network, a PGA broadcast his best network to share it with its peers. If *g* is too large, there will be little interaction, but if it is too small, there may be large overhead with premature convergence.

PGA genetic operators

A PGA uses basically the same mutation and crossover operators presented in [23]. These are specialized operators to perform local search and preserve two-connectivity constraints using a repair algorithm. Both, mutation and crossover, use set operators: union, intersection and subtraction. These operators are efficiently implemented using bit operators (OR, AND, NOT).

On the other hand, the proposed selection operator is not identical to the one presented in [23]. It has been extended to deal with variable-size population, as explained above. There are several PGA running in parallel, and interchanging partial results with their peers. So, when a PGA does a selection, it has to select a constant number of candidates, among the last population plus the recently arrived candidates, in order to preserve population size.

The PGA objective function (OF)

A penalization is used in the objective function to allow infeasible networks in the population, but preventing them to be chosen as the best one. This is done because two unfeasible networks can breed feasible solutions. A constant *k* has been introduced to the penalization function proposed in [23] to increase this restriction. Initial studies indicate that the objective function formulae can be further improved using a variable penalization, to get an adaptive penalization that would resemble a simulated annealing. For the present work, the first variation is preferred to maintain compatibility with Dengiz et al [23] results.

$$Z(x) = \sum\sum c_{ij} x_{ij} + \delta(c_{max} k(R(x) - R_0))^2 \qquad (2)$$

$$\delta = \begin{cases} 0, & if\ R(x) \geq R_0 \\ 10, & if\ R(x) < R_0 \end{cases}$$

$c_{max} = the\ maximum\ value\ of\ c_{ij}$

$k \quad a\ constant$

PGA stop-criteria

Each PGA runs independently of their peers, informing the coordinator when it reaches a stop-criteria. There are two finishing criteria:

- convergence of the population to a homogenous population;
- maximum number of generations.

A PGA does not stop when its finishing criterion is satisfied, because it may receive new network candidates from its peers. It just informs the coordinator instead. The coordinator is the one in charge of testing general convergence and killing the PGA processes.

At every generation, each member of the population is compared to the best candidate. If a given network has no more than *l* links that are not in the best, this network is considered similar to the best. When the average of similar networks is more than a given percentage, the population is considered homogenous.

The PGA Algorithm

A PGA process (see Pseudocode 2), has five main tasks:

- to generate the initial population. A specialized method is used to assure random and feasible network candidates;
- to receive candidate networks that have been transmitted by the peers, and to apply selection over the whole population (old population plus the arrived ones). If an imported candidate is better than the best local network, the later is replaced by the new arrived. That way, the penalization parameters are improved;
- to perform the regular GA operations (crossover and mutation);
- to broadcast the best network it finds. That way, a good candidate may be exported to other processes several times. If the network is the global best at this time, all the PGA will adopt it and export it, until a new best network appears. Each PGA transmits its best network every *g* generations or when a new best appears;
- to inform the coordinator when a given finishing criteria is satisfied.

4. EXPERIMENTAL RESULTS

The experimental experiences have been performed over a 10 Mbps Ethernet network, with three personal computers with Intel 80486 processors of different configurations. The programs were written in C and the parallel implementation was done using PVM (Parallel Virtual Machine). The processes have been assigned regarding the relative performance to have a reasonable work balance. The sequential GA, used as reference, processes a population size P times larger than the one on the PGAs, where P is the number of spawned PGA processes.

```
START
POPULATION pop[], pop_old[], imported[]
pop_size=s                                        /*population size*/
gen=0                                /*Generation number*/
exp_freq                             /*Export every exp_freq
generations*/
GENERATE_NEW_POPULATION ( pop_old )
FOR (i=1 to s)
    UPPER_BOUND(pop_old[i])  /*Reliability calculation*/
END FOR
ORDER_POP_BY_OF(pop_old)/*By objective function*/
$x_{best}$ = pop_old[1]
MONTE_CARLO (pop_old[1])
DO WHILE( TRUE )
     IMPORT_NETWORKS       /*From other processes*/
     FOR (i=1 TO pop_size/2)
          SELECT_TWO_NETWORKS
          IF (RANDOM<crossover_rate)
                    GET_CHILDREN_BY_CROSSOVER
          ELSE
                    CLONE_CHILDREN_FROM_PARENT
S
          ENDIF
          IF (STOP_CRITERIA) INFORM_COORDINATOR
          IF (RANDOM< mutation_rate) MUTATE_CHILD1
          IF (RANDOM< mutation_rate) MUTATE_CHILD2
          INSERT_CHILDREN(pop)
     END FOR
     ORDER(pop)
     i=1
     DO WHILE (pop[i]) better than $x_{best}$ AND i<pop_size)
          MONTE CARLO (pop [i])
          IF (pop[i]) better than $x_{best}$)
                    pop[i]=$x_{best}$
                    EXPORT($X_{best}$)
          END IF
          i=i+1
     END DO
     IF (gen MOD exp_freq = 0) EXPORT($X_{best}$)
```

Pseudocode 2: Parallel Genetic Algorithm – PGA.

For the present work, the reliability constraint is relaxed to allow performance comparisons between the sequential GA and the A-Team. As discussed before, the penalization function is not sufficient to prevent almost reliable networks from being chosen as the best solution, especially considering that Monte Carlo simulation only gives a good approximation of a given network reliability. Therefore, unfeasible solutions with low cost *all-terminal* reliability approximation that differs in no more than 2%, are accepted as good solutions.

Figure 3 shows a typical running of the GA and the implemented A-Team, plotting the temporary best solution as a function of time. It can be seen that the A-Team converges much faster than the GA. In fact, the GA satisfies a finishing criterion (by population homogenization) more than 400 s before the GA finds a similar quality solution.

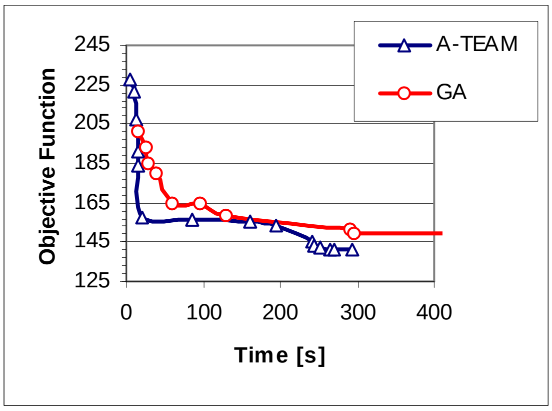

Figure 3: Typical running showing a fast convergence of an A-Team.

Table 1 presents results over 10 runs when designing a 10 nodes network that may be fully interconnected, i.e. there are 45 possible links. Each link has a reliability of 90% and an *all-terminal* reliability requirement of 95%. The testing network design problem is extracted from a test-set provided in [23]. It can be noted that the A-TEAM not only gets good solutions, but also a more predictable running time when compared to sequential GA.

	A-TEAM			Sequential GA		
Run	Best Cost	Reliability	Time [s]	Best Cost	Reliability	Time [s]
1	140	0.9412	339	147	0.9416	230
2	143	0.9374	305	149	0.9470	719
3	153	0.9458	158	140	0.9382	341
4	139	0.9360	374	135	0.9323	983
5	139	0.9375	192	140	0.9446	1480
6	142	0.9444	155	142	0.9364	334
7	142	0.9371	177	150	0.9381	640
8	141	0.9464	294	142	0.9370	1755
9	154	0.9467	351	139	0.9340	652
10	140	0.9379	198	139	0.9388	669
Average	143.3	0.94104	254.3	142.3	0.9388	780.3
Standard Deviation	5.54	0.0044	86.38	4.85	0.0043	497.55

Table 1: Experimental results over 10 runs.

Table 2 shows the average over three runs of a network design problem with 50 nodes fully interconnected. As can be seen, speedup scales very well indicating that the proposed A-Team can be used with advantage in the design of larger reliable networks.

	Best Cost	Reliability	Time [s]
GA	1536.5	0.8646	21481.5
A-TEAM	1383.3	0.8285	9302.3

Table 2: Average over 3 runs for a 50-nodes fully connected network design.

5. CONCLUSIONS AND FUTURE WORK

The paper presented a network design problem subject to reliability constrain that is especially complex because not only the design itself is NP-hard, but also the exact reliability calculation. For this reason, several different methods have been published but none of them is efficient enough to solve network size of nowadays.

The complexity of the design and the variety of completely different algorithms, each one with its own strength, make of this problem an ideal field for testing Asynchronous Team Algorithms, a technique to exploit available computer power of asynchronous distributed systems, as computer networks. The idea behind this proposal is to combine different algorithms, as the two implemented in the reliability calculation, to get the best of each one in what each is good for. In this context, it was proposed a parallel (-specialized) Genetic Algorithm for the design problem, while an upper bound calculation and Monte Carlo simulation were used for network reliability estimation. By combining all those algorithms in a network of available personal computers, good results are found, with considerable speedup that scales very well with the size of the problem.

Considering that most modern organizations today have access to a good number of computers interconnected through a network, the presented technique gives an ideal approach to solve very large and complex problems as the design of reliable networks. Furthermore, using the same ideas with relatively little change in the presented implementation, the authors are studying a variety of similar design problems, as the following ones:

- Maximize reliability of a network, given a maximum budget.
- Design of reliable networks with different kind of links as: optical fiber, wireless communication, telephone lines, etc., (with or without redundancy).
- Design of reliable networks with other constrains as: maximum number of hops between any two nodes, minimum total throughput, maximum delay, etc.

There is work to do, but A-Teams promise to be a very good tool for treating large and complex problems in modern internetworked environments.

ACKNOWLEGMENT

To Professor Alice E. Smith, for her collaboration.

REFERENCES

[1] Colbourn C.J., "*Reliability Issues in Telecommunication Network Planning*, University of Vermont. http://www.emba.uvm.edu/~colbourn.

[2] Colbourn C.J., "*The Combinatorics of Network Reliability.*" Oxford Univ. Press, 1987.

[3] R.H. Jan. "*Design of reliable networks.*" Comput. Oper. Res. Vol 20, pp 25-34, 1993.

[4] M.R. Garey and D.S. Johnson. "*Computers and Intractability: A Guide to the Theory of NP-Completeness.*" San Francisco, CA: Freeman, 1979.

[5] R.H. Jan, F J. Hwang, and S. T. Cheng, "*Topological optimization of a communication network subject to a reliability constraint.* " IEEE Trans. Reliability, vol. 42, pp. 63-70, 1993.

[6] A. N. Vetetsanopoulos and I. Singh, "*Topological optimization of communication networks subject to reliability constraint.*" Problem of Contr. Inform. Theory, vol. 15 pp. 63-78, 1986.

[7] M. M. Atiqullah and S. S. Rao, "Reliability optimization of comunication network using simulated annealing," *Microelectronics and Reliability*, vol. 33, pp. 1303-1319, 1993.

[8] S. Pierre, M. A. Hyppolite, J. M. Bourjolly and O. Dioume, "Topological desing of computer communication network using simulated annealing" *Eng. Applicat. Artificial Intell.*, vol. 8, pp. 61-69, 1995.

[9] F. Glover, M. Lee and J. Ryan, "Least-cost network topology design for a new service: An application of a tabu search," *Ann. Oper. Res.*, vol. 33, pp. 351-362, 1991.

[10] H. F. Beltran and D. Skorin-Kapov, "On minimum cost isolated failure immune network," *Telecommun. Syst.*, vol. 3, pp. 183-200, 1994.

[11] S. J. Koh and C. Y. Lee, "A tabu search for the survivable fiber optic communication network design," *Comput. Ind. Eng.*, vol. 28, pp. 689-700, 1995.

[12] D. W. Coit and A. E. Smith, "Reliability optimization of series-parallel systems using a genetic algorithm," *IEEE Trans. Reliability*, vol. 45, pp. 254-260, 1996.

[13] K. Ida, M. Gen and T. Yokota, "System reliability optimization of series-parallel systems using a genetic algorithm," *in Proc. 16th Int. Conf. Computers and Industrial Engineering*, 1994, pp. 349-352.

[14] L. Painton and J. Campbell, "Genetic algorithms in optimization of system reliability," *IEEE Trans. Reliability*, vol. 44, pp. 172-178, 1995.

[15] A. Kumar, R. M. Pathak, Y. P. Gupta and H. R. Parsaei, "A genetic algorithm for distributed system topology design," *Comp. Ind. Eng.*,vol. 28, pp. 659-670, 1995.

[16] A. Kumar, R. M. Pathak and Y. P. Gupta, "A genetic algorithm based reliability optimization for computer network expansion," *IEEE Trans. Reliability*, vol. 44, pp. 63-72, 1995.

[17] L. Davis, D. Orvosh, A. Cox and Y. Qui, "A genetic algorithm for survival network design," *in Proc. 5th Int Conf. Genetic Algorithms*, 1993, pp. 408-415.

[18] F. N. Abuali, D. A. Schoenefeld and R. L. Wainwright, "Terminal assignment in communication network using genetic algorithm," *in Proc. ACM Computer Science Conf.*, 1994, pp. 74-81.

[19] F. N. Abuali, D. A. Schoenefeld and R. L. Wainwright, "Designing telecommunications networks using genetic algorithm and probabilistic minimum spannig trees," *in Proc. 1994 ACM Symp. Applied Computing*, 1994, pp. 242-246.

[20] G. A. Walters and D. K. Smith, "Evolutionary design algorithm for optimal layout of tree networks," *Eng. Optim.*, vol. 24, pp. 261-281, 1995.

[21] D. L. Deeter and A. E. Smith, "Heuristic optimization of network design considering all-terminal reliability," *in Proc. Reliability and Maintaninnability Symp.*, 1997, pp. 194-199.

[23] Dengiz B., Altiparmak F., & Smith A.E. "*Local Search Genetic Algorithm for Optimal Design of Reliable Networks*," IEEE Transactions on Evolutionary Computation, Vol. 1, No. 3, 1997, pp. 179-188.

[24] Barán B., Kaskurewicz E., Bhaya A., "*Parallel Asynchronous Team Algorithms: Convergence and*

Performance Analiys," IEEE Transactions on Paralell and Distribuited Systems, Vol. 7, NO. 7, 1996.

[25] Souza P. S. y Talukdar S. N. "*Genetics Algorithms in Asynchronous Teams,"* ICGA- 91, pp. 392-397, San Diego – California, 1991.

[26] "*A-Teams Project Home Page"*. Carnegie Mellon University. URL: http://www.cs.cmu.edu/afs/cs/project/edrc-22/project/ateams/WWW/home_page.html

[27] Souza P. S. and Favilla J. R., "Asynchronous Teams for Steel Industry: Mini Mills," ISAS´96. Orlando- Florida, USA, 1996.

[28] Barán B., Chaparro E. and Cáceres N., "*A-Teams en la optimización del caudal turbinado de una represa hidroeléctrica.*" IBERAMIA´98 pp.187-199, 1998.

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

[30] M. S. Yeh, J. S. Lin, and W. C. Yeh, "*New Monte Carlo method for estimating network reliability.*" In Proc. 16th Int. Conf. Computers & Industrial Engineering, 1994, pp. 723-726.