"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XVIII – 2015 – Issue 2
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania // The journal is indexed in:
PROQUEST SciTech Journals, PROQUEST Engineering Journals, PROQUEST Illustrata: Technology, PROQUEST Technology Journals, PROQUEST Military Collection PROQUEST Advanced Technologies & Aerospace

GENETIC TECHNIQUES APPLIED IN ACTUAL NETWORK MANAGEMENT

Rares MANIU[1]
[1] Eng., Ph.D. (c), Military Technical Academy, Bucharest, Romania

Abstract: *Actual IT medium and future trends in this domain encourages the development of large, high-evolutive networks, with a high heterogeneity, who must meet a number of requirements in terms of maximum throughput, minimum delay, loss, delay-variation. The idea of mobility for everyone, using a large number of devices is becoming normality and change the way to manage networks. All of these characteristics, combined with anarchic scalability and event-driven architecture, transform network management in a multicriteria optimization problem. This article proposes a genetic algorithms way to realize a adaptive, autonomous management of such networks, with a efficient resources allocation, a efficient utilization of links and QoS guarantee.*

INTRODUCTION

With the number of hosts up to 1.200.000.000 in 2013, INTERNET is the largest network infrastructure to date. One of the most important ability is that this network exists and adapt in a context that is difficult to control or predict. The only certitude is that it had exponential growth even if the development follows an anarchic scalability. Over this infrastructure, a multitude of services exchange a huge amount of data, with a lot of requirements of quality of services.

New trends in network evolution will sustain the exponential growth of networks.

Because of the big number of electronic devices, most being mobile, who have computational power and communication capabilities, the concept of "mobility for everyone" became familiar and forces almost all real networks to be dynamic and large in size.

In 1999, Kevin Ashton launched the idea of "Internet of Things". IoT is expected to offer connectivity between devices, systems, services, covering a large number of protocols, applications and domains of activity. The connectivity between these devices must be automatically. A thing in IoT can be any natural or man-made object that can be an IP address and have the ability to communicate over the network.

The huge address space provided by IPv6 is an important factor that sustains the development of Internet of Things. According to Steve Leibson, "we can assign an IPv6 address to every atom on the surface of the earth, and still have enough addresses left to do another 100+ earths" that means we can assign an IP address to every "thing" on the planet.

Another new concept that require the automatic exchange data over a network is M2M concept.

The basic way to define Machine-to-Machine concept is that its role is to allow a device to exchange information with an application, over a communication network and that device and that application can act as the basis for this communication. The main characteristics of "smart objects" that support M2M communication are [1]:

- multitude - is generally agreed that the number of devices connected in M2M relationship will soon largely exceed the sum of all those that directly interact with humans, resulting in more pressure on application architectures, as well as on network load;
- variety – M2M devices are extremely diverse requirements in terms of data exchange rate, form factor, computing or communication capabilities, resulting a big heterogeneity, a major challenge to interoperability;
- invisibility – is an important requirement, the devices most deliver their services without or with minimum human control;
- criticality – some devices are life-savers, or are important elements in life-critical infrastructures and have particular requirements upon latency or reliability for communication;
- intrusiveness – the privacy may present a obstacle in the deployment M2M systems;
- limited in functionality – most M2M devices have low computational capabilities, because of the nature of exchanged information and performable action;
- low-powered – because of possibilities to connect to power supplies, the frequency and quantity of information exchanged will be limited;
- embedded – because of operating conditions;
- here to stay.

All of these directions of evolution, define real networks as evolving networks, that change as a function of time, either by adding or removing nodes or links or modify its parameters (delay, bandwidth, loss of packets, etc.).

The who-is-near-whom network (network with mobile devices with wireless capabilities) is an example of evolving network. Its components evolve every time users change its position and the communication services will deeply rely on the mobility and on the characteristics of the underlying network [2].

Because of these characteristics, the permanent task to generate and modify of network's configuration to sustain all services with respect of QoS is a multi-criteria optimization problem. In large and dynamic networks, where a solution

must respect all constraints and must be provided quickly such a problem can be solved using genetic techniques.

Related Work

Network viability relies on a combination of protocols and routing strategies to transport information between two nodes. Routing is the process that searches the path between source and destination and protocols assure handshaking activities for error checking and receiver acknowledgements. Because of the dynamicity of network's parameters, a solution who is sufficient in normal network condition may well be inefficient under other conditions produced by the network (changes in terms of load, topologies, etc.).

For networks with high grade of dynamicity, classical algorithms experiences difficulties to find a solution in a timely manner with respect of QoS criteria. Since classical algorithms can find the solution for short-path problems in polynomial time, they will be a good choice in fixed or low dynamical networks. But they can give a solution in real time, when rapidly changing occurs in network topologies.

Artificial intelligence techniques using genetic algorithms or neural networks are alternatives for solving the routing problem in evolving networks, even if they involve a large number of iteration till they present a solution. But this problem can be approached using high-speed processors and very fast hardware implementation. Even if neural network cannot be implemented in hardware because physical limitation (the size of communication network is variable in time), the hardware implementation of genetic algorithms follows well the modifications in network structure.

In [3], authors have applied genetic algorithms to solve shortest path problem in dynamic networks, to find a set of optimal routes to send the traffic between source and destination. Here, the chromosome is a path between source and destination. The selection of parents for next generation is based on "roulette wheel selection".

To eliminate the cycle, the authors uses multipoint crossover techniques, in fact Partially Mapped Crossover. Generated offspring should be validated, by checking with all possible routes. If offspring belongs to all possible routes, then its fitness is calculated and sent to next operation. If the offspring does not belong to all possible routes, it is dropped.

Because crossover may produce degenerate population and the result of algorithm will be a local solution, not a global solution, to avoid this mutation operation is performed. The paper uses insertion method; a node is inserted at random position in the spring, because a node along the optimal path may be eliminated through crossover. The offspring generated by mutation have to be validated. The authors said nothing about number of mutations in a iteration of algorithm.

In this algorithm, no change in population fitness and stall generation are considered as algorithm stop condition. A second stopping criterion is until some chromosome reaches a specified fitness level.

In [4], T.R. Gopalakrishnan, K. Sooda and R. Selvarani propose solution for route discovery, using a genetic algorithm approach for a dense network. The network is represented as a connected non-loop free graph with N nodes. The matric for optimization is the bandwidth available between the nodes. The chromosomes represent the path in network.

In this implementation, the selection process is carried out by roulette wheel method, the individuals being chosen based to the relative fitness with its competitors.

Crossover operator combines subparts of two chromosomes, producing offspring that contain parts of both parents. Here, for crossover, was used a partially mapped crossover, a kind of multipoint crossover mechanism.

To avoid a degenerate population produced by crossover, authors chosen the insertion method as mutation operation.

After a number of generation, the algorithm will stop and the values obtained in the last generation determines which among the chromosome will get selected as optimal path.

In [5], is presented a genetic algorithm approach to the shortest path routing problem.

As genetic representation, a chromosome of the proposed GA consists a sequence of numbers that represent the ID of nodes in routing path, and the locus represents the order of a node. First locus is the source. The length is variable, but it should not exceed the total number of nodes in network.

In this paper, random initialization is effected. By random initialization, nodes from the topological information database are chosen in a random manner, during the encoding process.

Fitness function evaluates the chromosomes as a solution generated by algorithm, and has a bigger value when the fitness characteristic of the chromosome is better than others. And, in addition, it introduces a criterion for selection.

In this algorithm, the function that involves computational efficiency and accuracy is:

$$f_i = \frac{1}{\sum_{j=1}^{li-1} c_{g_{i(j)},g_{i(j+1)}}} \qquad (1)$$

where f_i is the fitness value of the i-th chromosome, l_i is the length of i-th chromosome, $g_i(j)$ is the node of the j-th locus in the i-th chromosome and C is the link cost between nodes.

For selection, the pair wise tournament selection without replacement is chosen for proposed genetic algorithm. Two chromosomes are picked ant the one that is fitter is selected. The same chromosome should not be picked twice as parent.

Crossover combines the current solutions to find the better one. In this implementation, two chromosomes chosen for crossover should have at least one common gene, excepting source and destination. There are no requirements about locus of these common genes. The one-point crossover will be defined by the locus of these common nodes in paths.

To avoid finding a local solution for given problem, the mutation ensure completion of the entire search space. In order to perform the mutation, a gene is randomly selected first from the chosen chromosome and will be mutation point. One of the nodes, connected directly to the mutation point is randomly chosen as the first node of the alternative partial-route. The mutation probability is set to 5% from chromosomes in population.

The algorithm will finish when all the chromosomes have converged to the same solution.

These are some examples of genetic algorithms implemented to solve the routing problem in networks. All found solutions in a timely manner, with respect of imposed conditions.

GENETIC ALGORITHM

Genetic algorithm is a stochastic search algorithm, inspired from biological evolution. It works on a search space named population, each element from this population being a chromosome (in fact a solution). In the first step, the algorithm starts with a randomly set of chromosomes from population. Each chromosome is evaluated by a fitness function, this value being a quality of solution. Using operators like crossover, selection and mutation, from initial generation the new offspring are generated and the most fit chromosomes are selected for next generation. All this process repeats until the chromosomes have the best solution from given problem.

A general schema for a genetic algorithm can be illustrated as:

procedure GENETIC-ALGORITHM
Generate initial population P0;
Evaluate population P0;
Generation counter g=0;
While fitness function not satisfied repeat
Select chromosomes from Pg to copy into Pg+1;
Crossover chromosomes from Pg and put into Pg+1;
Mutate chromosomes from Pg and put into Pg+1;
Evaluate some elements of Pg and put into Pg+1;
Generation counter g = g+1;
End while
End procedure

Using a constant selection based on fitness, the average fitness from next population increases on every iteration, GA being an adaptive heuristic search technique that finds a set of best solution from population.

Because GA works with multi-objective optimization problem, there does not exist a single solution that simultaneously optimizes each variable, so, there exists a number of solutions named Pareto optimal solutions. A solution is called nondominated or optimal Pareto, if none of the objective functions can be improved in value without modifying some of the other objective values. All Pareto optimal solutions are considered good and the final solution will be selected using some additional criteria.

The most important characteristic of a genetic algorithm is that it is parallel. Because of random initializations and because of the genetic operators, it explores the solution space in multiple directions in one iteration. Because of this, it is used for solving problems where exists a huge solution space and it is difficult to search using exhaustively techniques.

The genetic algorithm can search for solutions without previous knowledge. Using a search in entire solution space and the fitness function that validate solutions, GA no need initial knowledge to search for optimum results.

GA can produce solutions, even if the problems have complex fitness and even if parameters are changing over time. It can solve every optimization problem which can be described with the chromosome encoding.

Genetically techniques can solve problems with multiple solutions, generating a set of Pareto optimal solutions and is a efficient for optimizations. Can provide good results in multi-objective, non-linear optimizations, with non-linear restrictions, for large seeking spaces, with a high number of variables.

Is a method easy to understand, to implement and to reconfigure. Because of its parallel nature, it can be implemented for parallel computing.

As disadvantages, some optimization problems cannot be solved using genetic algorithms due to poorly known fitness functions that generate bad chromosome.

There is no assurance that a genetic algorithm will find a global optimum. It generates solutions that solve the problem, but it can be a local, not global solution if the genetic operators can't search in entire solution space.

Like other artificial intelligence techniques, the genetic algorithm cannot work with a constant optimization response times and the difference between the shortest and the longest time is

larger than with conventional gradient methods. It is high time consumer due to the much iteration needed. Even with these drawbacks, the genetic algorithms can be implemented in real time applications, using parallel computing, high-speed processors or hardware acceleration.

Genetic algorithm applications in controls, which are working in real time, are limited because of random solutions and convergence. The entire population is improving, but this could not be said for an specific individual within population. The implementation must be tested first on simulation model if genetic algorithms will be used for online operation in real systems.

ROUTE SELECTION - THE IMPLEMENTATION OF GENETIC SEARCH

The network in consideration can be represented as a connected graph G=(nodes, links). Each link have a cost $C_{i,j}$,

where *i and j* are nodes.

The routing problem can be formulated as a minimization of function

$$F = \sum_{i=S}^{D} \sum_{j=S}^{C} C_{ij} * I_{ij} \qquad (2)$$

where S=source, D=Destination, $C_{i,j}$ is the cost between nodes I and j and

$$I_{ij} = \begin{cases} 1, if\ the\ link\ i-j\ exists\ in\ routing\ path \\ 0, othetwise\ or\ if\ i=j \end{cases}$$

a. Genetic representation

As in [6], the chromosome defines the route. Its genes are intermediate nodes between source and destination. Chromosome length is equal to the number of nodes in the network. It is obvious that the length of different routes will be not equal, so, the genes who don't exists because the difference between length of chromosome and the length of route will have value 0.

A chromosome encodes the problem by listing the node IDs from source to destination. This information is provided by routing protocols (OSPF, RIP, etc.), in real-time and is easy to obtain and manage. The main idea is that the topological information about network can be constructed easily, using information provided by routing protocols.

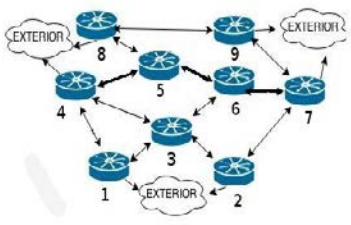

Fig.1. Network and chromosome representation

As in Fig.1., the chromosome (4,5,6,7,0,0,0,0,0) represents the definition of the route 4->5->6->7, between source node nr.4 and destination node nr.7.

b. Population initialization

The population size and structure affects the performance of genetic algorithm. In population initialization, exist two main aspects that must be considered: the initial population size and the procedure for initialization.

The population size is proportionally with the complexity of problem and is important to generate good solutions. But a large population size is reflected in high costs in terms of memory and computational time because the number of generations till convergence will be higher. If mutations occur for large population sizes, more generations are needed to assimilate information provided by mutated chromosomes. So, choosing an adequate population size is important for efficiency.

The initial population can be generated using heuristic or random techniques. Heuristic techniques can help the convergence, providing chromosomes well fitted, but it possible to find just a local solution, not a global solution, because of lack of diversity. Random initialization will provide chromosomes distributed in all search space, but needs more generation-time to convergence.

In this implementation, the initial population is generated random.

c. Fitness function

Fitness function is the core of GA and measure the quality of chromosome in the population and introduces a criterion for selection of chromosomes. It is an important point in the genetic algorithm definition, because a good implementation of this function improves the convergence (and the speed) of the genetic algorithm.

In this implementation, the fitness function has a lower value when the fitness value of chromosome is better. It is the sum of the costs of links from o route:

$$fitness_i = \sum_{j=0}^{l_i-1} C_{g_i(j),g_i(j+1)} \quad (3)$$

where $fitness_i$ is the fitness value for chromosome nr. I, l_i is the length of the chromosome (for this implementation is the number of nodes in network) and C is the link cost between 2 adjacent nodes in route.

Because it is a multi-criteria optimization, the value of C is a function of link parameter, like delay, bandwidth, stability, jitter, loss:

$C_{g_i(j),g_i(j+1)} =$
$f(\text{delay}, \text{bandwidth}, \text{stability}, \text{jitter}, \text{loss})$.

Because of particular demands required by services in a network, the impact of network parameters in link cost will be variable. For example, for FTP transfer, bandwidth will be more important then jitter. But for voice over IP, the jitter value will have a important impact in link cost.

d. Selection

The selection improves the average quality of population. It ensures promotion to the next generation of well-adjusted individuals. The selection pressure is the probability of selection of the best chromosome in the population relative to a chromosome with a medium fitness value. A high selection pressure results a quickly equilibrium of population but have a loss of diversity.

In genetic search exist two kinds of selection: proportionate and ordinal-based selection.

Proportionate selection evaluate chromosome based on their fitness value relative to the fitness of other chromosomes in population. Ordinal-based selection select chromosomes based their rank within population and this rank is based on their fitness.

For the proposed genetic algorithm, two chromosomes are evaluated according fitness function and the one that is fitter is selected. The better chromosome is promoted to be in next generation.

e. Crossover

As in [6], the crossover operator produces the mating between two chromosomes, being the most important genetic operator. There are two main types of conventional crossover: with one-cut point and with multi-cut points.

For crossover in one-cut point, a cutting point is chosen randomly and, after procedure, each resulting chromosome will contain information from each parent as in Fig. 3.

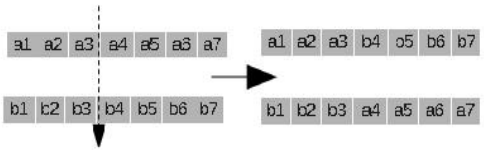

Fig. 2. One point chromosome crossover

For crossover in multi-cut points, more than one cutting point are chosen randomly and, after procedure, each resulting chromosome will contain information from each parent.

In this implementation, it is used a crossover in one-cut point. This point is chosen to be the middle node in route. After exchanging partial route of two parents, because is possible that loops are formatting during crossover, the offspring follows a procedure to eliminate loops and, after, they must be evaluated if represent a valid route. If not, is chosen another pair of parents for crossover, and the procedure are repeated till are obtained a full population of offspring.

f. Mutation

The mutation operator ensures that all of the solution space will be searched and avoids local optimum fitness function. It produces random changes in population and replaces the gene lost because of selection process or provide the gene that were not present in first population. If the mutation probability is too low, many genes that can be useful for optimal solution will never be tried and, if the mutation probability is too high, the difference between parents and offspring will be too high and algorithm will not be able to learn from the history of search. A well-chosen percent of mutation will maintenance diversity.

In this implementation, the mutation operator will replace randomly, a number of chromosomes from new population.

g. Termination criteria

The termination criterion is in fact the convergence of algorithm. No changes in population fitness and a stall generation are considered as convergence of algorithm. A second stopping criterion is if a chromosome reaches a specific fitness level. Because of the evolving network, there may be changes in topology. Some nodes can join the network, some links changes its costs, some nodes may leave the network or some nodes may fail. Because of these, the old optimal path may no more be the shortest and the algorithm has to be reloaded at every t seconds, or if exists a trigger who detect changes and start the algorithm.

RESULTS

The genetic algorithm from this implementation use a one point crossover operator. About selection method, two chromosomes are evaluated according fitness function and the one that is fitter is selected. The number of elements from parents generation, offspring generation and next generation are equal. The termination criterion is when the population from next generation become stable and presents a minimum as fitness function or a maximum time.

a. *For a high connected network* (with 50 nodes, 1000 links and mutation percent 5%). The minimum cost between start node and stop node was 3 for 1 route and 4 for another route. Another links have costs 5 and bigger.
   - Using a number of 100 chromosomes as parents, 100 chromosomes as offspring, 100 chromosomes in next generation, the time per iteration become very high. In 10 runs of the algorithm, for the same time interval, the algorithm found the route with cost 4 as minimum 3 times. 7 times,

the returned route had 5 as minimum cost.

- Using a number of 50 chromosomes as parents, 50 chromosomes as offspring, 50 chromosomes in next generation, the time per iteration was decreased. In 10 runs of the algorithm, for the same time interval, the algorithm found the route with cost 3 as minimum 2 times. 5 times, the returned route had 4 as minimum cost and 2 times 5 as minimum cost.
- Using a number of 30 chromosomes as parents, 30 chromosomes as offspring, 30 chromosomes in next generation, the time per iteration was low. In 10 runs of the algorithm, for the same time interval, the algorithm found the route with cost 3 as minimum, 5 times. 4 times, the returned route had 4 as minimum cost and 1 time 5 as minimum cost.
- Using a number of 20 chromosomes as parents, 20 chromosomes as offspring, 20 chromosomes in next generation, the algorithm runs quickly. In 10 runs of the algorithm, for the same time interval, the algorithm found the route with cost 3 as minimum, 5 times and 5 times, the returned route had 4 as minimum cost.

In all cases, a route with cost 5 is generated quickly (after about 10% from timeframe).

b. *For a low connected network* (with 50 nodes, 200 links and mutation percent 5%). The minimum cost between start node and stop node was 4 for 1 route, 5 for another route and 6 for another. Next links have costs 8 and bigger.
   - using a number of 100 chromosomes as parents, 100 chromosomes as offspring, 100 chromosomes in next generation, the time per iteration was big. In 10 runs of the algorithm, for the same time interval, the algorithm found 1 time the route with cost 6 as minimum, and 9 times, the algorithm returned route with 8 as minimum cost.
   - using a number of 50 chromosomes as parents, 50 chromosomes as offspring, 50 chromosomes in next generation, the algorithm runs quickly. In 10 runs of the algorithm, for the same time interval, the algorithm found 1 time the route with cost 6 as minimum, and 5 times, the algorithm returned route with 8 as minimum cost.
   - using a number of 30 chromosomes as parents, 30 chromosomes as offspring, 30 chromosomes in next generation, the algorithm runs quickly. In 10 runs of the algorithm, for the same time interval, the algorithm found 1 time the route with cost 5 as minimum, and 9 times, the algorithm returned route with 8 as minimum cost.
   - using a number of 20 chromosomes as parents, 20 chromosomes as offspring, 20 chromosomes in next generation, the algorithm runs quickly. In 10 runs of the algorithm, for the same time interval, the algorithm found 1 time the route with cost 6 as minimum, and 9 times, the algorithm returned route with 8 as minimum cost.

   In all cases, a route with cost 8 is generated quickly (after about 10-15% from timeframe).

CONCLUSIONS

This implementation, as it can be seen in results, is a good choice when the network can be represented as a high connected graph. In a dense network, the results are very good, the optimum route being detected in a short time.

A big number of chromosomes in a population increase the processing time per iteration and, even if it means a high diversity, the algorithm need a long time to converge. From tests, for 100 chromosomes per generation, in the timeframe, the algorithm didn't found the minimum. On the other way, a short number of chromosomes in population produce the increasing of processing speed. The simulation shows that, with 20 and 30 chromosomes per generation, the number of detection for optimum route is high.

For a low connected network, the results are different. In the same timeframe, the number of optimal route detection was lower then in a high-connected network, and, in this case, classical algorithms for route detection being the best choice.

For both types of network, the algorithm generated quickly a route with a cost closer then minimum (5 for high connected network, where minimum was 3 and 8 for low connected network, where minimum was 4). It means that this algorithm can generate in time manner an acceptable solution for problem, but not necessary the best.

BIBLIOGRAPHY:

[1] Emmanuel Darmois, Omar Elloumi "Introduction to M2M" Alcatel-Lucent, Velizy, France
[2] P. Borgnat, E.Fleury, J.L. Guillaume, C. Magnien, C. Robardet, A. Scherrer "Evolving Networks", http://liris.cnrs.fr/Documents/Liris-3669.pdf
[3] R. Kumar, M. Kumar "Exploring Genetic Algorithms for Shortest Path Optimization in Data Networks", Global Journal of Computer Science and Technology, Octomber 2010, vol.10, pag.8
[4] T.R. Gopalakrishnan, K. Sooda, R. Selvarani "A QoS based Routing Approach using Genetic Algorithms for Bandwidth Maximization in Networks", http://arxiv.org/ftp/arxiv/papers/1408/1408.1770.pdf
[5] C.W. Ahn, R.S. Ramakrishna "A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Population", IEEE Transactions on Evolutionary Computation, vol.6, no.6, December 2002
[6] R. Maniu, L.A. Dumitru, "Self-adaptive Networks with History Extrapolation, Evolutionary Selection and Realtime Response", OPTIM 2014