Genetic Algorithm Based Solution for Large-Scale Topology Mapping

Nada Osman[1(✉)], Mustafa ElNainay[1,2], and Moustafa Youssef[1]

[1] Department of Computer and Systems Engineering, Alexandria University, Alexandria, Egypt
{nada_s_osman,ymustafa,moustafa}@alexu.edu.eg
[2] Islamic University of Madinah, Medina, Saudi Arabia
melnainay@iu.edu.sa

Abstract. Simulating large-scale network experiments requires powerful physical resources. However, partitioning could be used to reduce the required power of the resources and to reduce the simulation time. Topology mapping is a partitioning technique that maps the simulated nodes to different physical nodes based on a set of conditions. In this paper, genetic algorithm-based mapping is proposed to solve the topology mapping problem. The obtained results prove a high reduction in simulation time, in addition to high utilization of the used resources (The number of used resources is minimum).

Keywords: Network simulation · Topology mapping · Testbeds · Genetic algorithm

1 Introduction

Network simulation is an essential step in designing and validating new networking protocols. The availability of a wide range of network simulators makes it easy for researchers to run simulation experiments on their machines. Nevertheless, simulating large-scale network experiments on a single physical node requires powerful resources to assure time feasibility. Partitioning large-scale network simulation on multiple physical nodes will solve this scalability problem by reducing both the required resources of the used machines and the simulation time.

The partitioning can be done manually based on the number of accessible physical nodes. However, manual partitioning consumes time, and it may not reach an acceptable reduction in simulation time. Another option is the use of an automatic partitioner provided by some online accessible testbed, where the testbed automatically partitions the given experiment based on its available physical nodes.

One of the automatic partitioning techniques is topology mapping, where it maps each simulated node in an experiment to an available physical node. This mapping will partition the traffic of the topology into two parts:

H. Gao et al. (Eds.): TridentCom 2019, LNICST 309, pp. 62–77, 2020.
https://doi.org/10.1007/978-3-030-43215-7_5

- Simulated traffic: The inner traffic between simulated nodes on the same physical machine.
- Emulated traffic: The outer traffic between simulated nodes on different physical nodes, which passes through the real links between the physical nodes.

The main goal of topology mapping is to reduce the simulation time, taking into account the effect of the emulated traffic on the evaluation of the experiment.

Most of the existing topology mapping solutions fix the number of physical nodes and work on finding a load-balanced partitioning over them. But from a testbed viewpoint, it is essential to minimize both the number of occupied physical nodes and their occupation time. Therefore, other topology mapping solutions worked on reducing the used resources by penalizing them with an approximated cost.

In contrast to previous topology mapping solutions that focused on a single topology mapping goal, or needed prior step of cost estimation of the used resources, in this paper, we are proposing a genetic algorithm based solution for the topology mapping problem that achieves all of the following goals, depending only on the developed fitness function:

- Minimizing the number of physical nodes used in the simulation by maximizing their utilization.
- Minimizing the simulation time of the experiment by limiting the amount of simulated traffic on each physical node to the capacity of the node.
- Limiting the amount of emulated traffic passing through real links to the capacity of the links. This is to reduce the effect of using the real network on the evaluation of the experiment.

The evaluation results show that the proposed technique can find mapping solutions that reduce simulation time by more than 90% for different topology sizes while keeping the utilization above 95%, which minimizes the number of used physical nodes.

The paper is organized into six sections. Following the introduction, related work is surveyed in Sect. 2. In Sect. 3, the topology mapping problem is formulated, while the details of the proposed GA approach is explained in Sect. 4. The obtained evaluation results are presented in Sect. 5. Finally, in Sect. 6, conclusion and future works are discussed.

2 Related Work

In the topology mapping problem, the used algorithm needs to search the mapping space (between the simulated nodes of a given topology and the available physical nodes to use) for the optimal mapping that minimizes the simulation time while satisfying a set of constraints. Searching for that optimal mapping is an NP-hard problem [1].

A well-known solution for the topology mapping problem is through the use of graph partitioning techniques. The minimum graph cut algorithm is used to partition a given topology on the available workers. Where the load is balanced over the workers, and the traffic on real links is minimized to prevent the physical network from being a bottleneck.

Graph partitioning was used for topology mapping in MaxiNet [2], where Wette et al. used the METIS technique to parallelize the simulation of large software-defined networks [3]. Similarly, Liu et al. applied graph partitioning in [4] and used large realistic network architectures for evaluation. Furthermore, Yocum and Ken et al. compared the minimum graph cut technique to a random partitioning technique, concluding that the minimum graph cut outperforms random partitioning [5].

Other solutions use greedy optimization search to solve the topology mapping problem. In [6], Galvez et al. proposed a randomized mapping technique named GreedyMap. It iterates over tasks, and computes a partial cost of the current mapping state, then chooses the node that minimizes the cost. Another example is [7], where Hoefler et al. applied two greedy approaches: A greedy heuristic approach that worked on integrating heavy traffic on the same physical node. In addition to a graph similarity approach that used the similarity between the simulation experiment topology and the physical network topology to perform the mapping.

The mentioned techniques have two main drawbacks: First, they assume the number of physical nodes to use is fixed and known, without trying to minimize it. Second, they assume homogeneous workers with similar resources and evenly partition the experiment load over them. A generalization is needed to allow the use of heterogeneous physical nodes.

Ricci and Robert et al. proposed a more general solution named *assign* in [8] and [9] using simulated annealing algorithm [10], for the Emulab testbed [11]. The *assign* solution used a cost function that depends on the amount of traffic on the physical network. The cost differs based on the physical resources used, where each resource in the testbed has a fixed approximated cost. Moreover, to minimize the number of physical resources used, the solution penalizes the cost function with the number of physical nodes and physical links used in the mapping.

One drawback of the *assign* solution is the use of fixed approximated costs, which requires manual setting. Furthermore, the solution targets to minimize resources usage, without considering reaching the minimum simulation time.

The proposed genetic algorithm uses a fitness function that depends on analytically computed values of both the utilization of the used resources and the simulation time of the experiment after mapping, where the utilization part of the function controls the number of the used resources. Furthermore, the fitness function takes into account the capacity of each used resource, allowing the use of heterogeneous resources, without the need to fix approximated costs for them. The proposed technique in this paper overcomes the discussed drawbacks, with the privilege of reaching the minimum simulation time and the maximum utilization.

3 Topology Mapping Problem Formulation

The topology mapping problem, in this work, is defined as follows: Given a simulation topology of N simulated nodes, and a physical network of P physical nodes, the problem maps each simulated node n to a physical node p, such that, the simulation time of the experiment is min, and the utilization of the used resources is max.

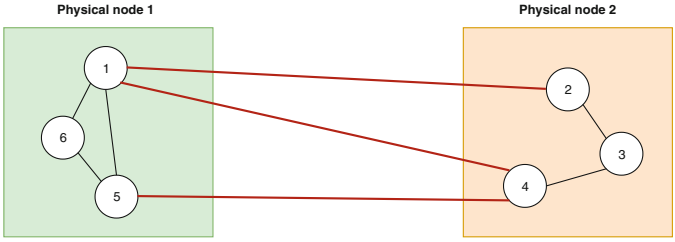


Fig. 1. Two physical nodes are used to simulate a topology of six simulated nodes, where the partitioning emulates three links using the topology on the real network interface connecting the two physical nodes.

In Fig. 1, a connected topology of six nodes is simulated on a physical testbed of two physical nodes ($N = 6$ and $P = 2$). As shown in the figure, the mapping partitions the topology links into simulated links inside the physical nodes, and emulated links using the physical network between the physical nodes (Table 1).

Next subsections formulate the topology mapping problem in details.

3.1 Simulation Time

As mentioned earlier, there are two types of traffic in the distributed experiment: The emulated traffic, and the simulated traffic. The simulation time differs based on the traffic type.

– **Emulated Traffic:**
Let L_r^R be a set of topology links emulated using the real link r. Equation (1) computes the total amount of traffic passing through r, and (2) estimates the emulation time of this traffic (in seconds).

$$T_r^R = \sum_{l \in L_r^R} T_l \tag{1}$$

$$\tau_r^R = \frac{T_r^R}{C^R} \tag{2}$$

Where C^R is the capacity of r.

Table 1. Table of symbols

Symbol	Definition
N	Number of simulated nodes/Number of genes
L	Number of simulated links
n	A simulated node/A gene, where $n = 1, 2, \ldots, N$
l	A simulated link
T	Traffic in Mbps
T_l	Traffic on the simulated link l, where $l = 1, 2, \ldots, L$
P	Number of all available physical nodes
p	A physical node, where $p = 1, 2, \ldots, P$
r	A real link
U_P	Number of used physical nodes
U_R	Number of used real links
C^S	Simulation capacity in MHz
C^R	Real traffic capacity in Mbps
C_p^S	The simulation capacity of the physical node p
L_p^S	Set of simulated links on the physical node p
L_r^R	Set of simulated links emulated on the real link r
T_p^S	Simulated traffic on the physical node p
T_r^R	Real traffic passing through the real link r
τ	Time in seconds
τ_p^S	Time taken to simulate the traffic on the physical node p
τ_r^R	Time taken by real traffic on the real link r
μ	Utilization (%)
μ_p	Utilization of the physical node p
η	Population size
i	An individual in the population, where $i = 1, 2, \ldots, \eta$
g_n^i	The value of the gene n in the individual i
M_I	Mutation probability of individuals
M_G	Mutation probability of genes
G_τ	Gain in time
D_μ	Drop in utilization
F	Fitness value
E	Above link capacity error
A, B, C	Utilization model Constants
α, β, γ	Scaling parameters

– **Simulated Traffic:**
Let L_p^S be a set of topology links simulated inside the physical node p. Similar to the emulated traffic, Eq. (3) computes the total amount of traffic running on p.

$$T_p^S = \sum_{l \in L_p^S} T_l \tag{3}$$

Assuming a discrete-event simulator such as NS3 [12], each bit in the simulated traffic consumes at least two CPU clock events, ignoring the processing time: A sending event, and a receiving event. Given that, Eq. (4) estimates the simulation time of this traffic (in seconds).

$$\tau_p^S = \frac{2 \times T_p^S}{C_p^S} \tag{4}$$

Where C_p^S is the simulation capacity of the physical node p.
Figure 2 compares the experimentally measured time and the analytically computed time using (4). Although the experimental time is longer than the analytical time, due to ignoring the processing time in the analytical formula, both times approximately follow the same behavior. Consequently, the analytical time is a valid estimation of the real-time needed to simulate a given amount of traffic.

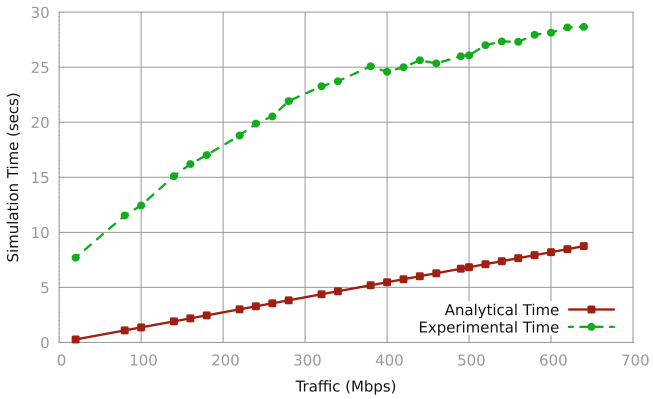


Fig. 2. The experimental simulation time measured VS the analytical simulation time computed using (4)

The total simulation time of an experiment is the maximum time needed by both the simulated traffic and the emulated traffic, given by (5).

$$\tau(U_P) = Max(\tau_1^S, \tau_2^S, \ldots, \tau_{U_P}^S, \tau_1^R, \tau_1^R, \ldots, \tau_{U_R}^R) \tag{5}$$

Where U_P is the number of physical nodes used in the mapping, and U_R is the number of real links used.

3.2 Utilization

Figure 3 shows the relation between the amount of traffic running on a physical node and the utilization of the node, which is described by (6). The Constants A, B, and C were estimated using Regression.

$$\mu_p = A - B \times e^{C \times T_p^S} \tag{6}$$

Where T_p^S is the simulated traffic using the physical node p.

The overall utilization is computed using (7), Where U_P is the number of physical machines used in the experiment.

$$\mu(U_P) = Min(\mu_1, \mu_2, \ldots, \mu_{U_P}) \tag{7}$$

Fig. 3. The experimentally measured utilization VS the analytically computed utilization.

4 Genetic Algorithm Approach

A genetic algorithm [13] uses the idea of evolution to enhance the searching path of reaching the goal. It starts with an initial population of individuals, where an individual is a chromosome that consists of a set of genes. Each iteration the best individuals in the current generation are elected to produce new ones, supposedly, closer to the goal. The algorithm uses a fitness function to measure how fit is an individual, where the higher the fitness of an individual, the closer it is to the goal and the higher its election probability.

The proposed chromosome in this work is of length N, where each gene in the chromosome represents a simulated node. The value of the gene represents the physical node to be used to simulate the corresponding simulated node. Each gene has a value in the range $[1, P]$, where P is the number of available physical nodes in the testbed.

4.1 Initial Population

Let the number of individuals in the population to be η. Initially, η individuals are generated by random. For each gene in an individual, a physical node is chosen by random using (8).

$$g_n^i = rand(1, P) \tag{8}$$

Where g_n^k is the gene represents the simulated node n of the individual i.

4.2 Crossover and Mutation

A new generation consists of η new children. To produce a new child, the parents are chosen randomly from the top half of the current population where the population is sorted based on the fitness of the individuals. A child takes half of its genes from one of the parents, and the second half from the other parent.

Genetic algorithm uses mutation to produce new random genes that might lead to better solutions, instead of only depending on inherited genes from parents. In this GA work, a percentage of $M_I\%$ of the produced children are mutated. For each mutated child, $M_G\%$ of its genes are rechosen by random from the set of available physical nodes.

4.3 Individual Evaluation and Fitness Function

To measure how fit is an individual (mapping solution), the algorithm needs to compute the following values: The number of physical machines used (U_P), the traffic to be simulated on each physical node p (T_p^S), and the traffic passing through each real link r (T_r^R). The fitness of an individual depends on how much reduction in simulation time it gives, compared to simulating the whole topology on a single physical node, taking into account the reduction in utilization caused by using more physical nodes.

Equation (9) gives the amount of reduction in simulation time, which represents the gain of a given mapping.

$$G_\tau(U_P) = 1 - \frac{\tau(U_P)}{\tau(1)} \tag{9}$$

Where $\tau(1)$ is the time taken to simulate the whole topology on a single physical node ($U_P = 1$).

Equation (10) gives the amount of drop in utilization, which represents the loss of a given mapping.

$$D_{\mu}(U_P) = 1 - \frac{\mu(U_P)}{\mu(1)} \tag{10}$$

Where $\mu(1)$ is the utilization of simulating the whole topology on a single physical node ($U_P = 1$).

The goal of the algorithm is to maximize the gain and minimize the loss. Many forms of the fitness function were tested, starting from dividing G_{τ} by D_{μ}, passing through many normalization problems. Until reaching the fitness function that gave the best performance in terms of simulation time and utilization. The final fitness function is defined in (11), where α is a scaling parameter that controls the minimization in simulation time against lowering the utilization. β is a function that represents a normalization factor between the gain and the loss.

$$F(U_P) = E \times [\alpha \times \beta(U_P) \times G_{\tau}(U_P) - (1 - \alpha) \times D_{\mu}(U_P)] \tag{11}$$

For individuals that do not match the maximum link capacity condition, instead of ignoring them, their fitness value is penalized by the amount of exceedance in traffic over the limited capacity as shown in (12), where the γ parameter represents the penalization factor. This way, the algorithm can benefit from the infeasible solutions if they can lead to a better feasible solution.

$$E = \gamma \times (1 - \frac{Max(T_1^R, T_2^R, \ldots, T_{U_R}^R) - C^R}{C^R}) \tag{12}$$

4.4 Parameters Setting

The value of some parameters such as η, M_I, M_G, and γ, is set using parameter tuning, where the values that achieve the best performance in terms of simulation time and utilization are the values used in the final model. The value of the scaling factor α depends on the following:

- The privileges that are given by the testbed to the user. As a privileged user can use more resources to reduce the simulation time with ($\alpha > 0.5$).
- The cost of the physical nodes used. As ($\alpha < 0.5$) for expensive resources.

For large-scale experiments, the simulation time in seconds (which has no upper limit) is much higher than the utilization (which has a maximum value of 100%). According to (9) and (10), the time gain could be much less than the utilization loss. Hence, the fitness function tends to maximize the utilization more than reducing the time. The normalization parameter β described in (13), is used to overcome the dominance of the utilization part.

$$\beta(U_P) = \frac{\tau(U_P)}{\mu(U_P)} \tag{13}$$

4.5 Minimum Graph Cut

A simulation topology is a weighted graph, where the traffic passes through a link represents the link's weight. Consequently, the proposed GA-based mapping could benefit from the minimum graph cut technique to enhance the initial population of the algorithm. If there are P available physical nodes, the number of used machines in the final solution U_p is in the range $[1, \ldots, P]$. The algorithm uses the minimum graph cut technique to find a minimum cut for each possible value of U_p, then the found solutions are added to the initial population of the GA.

Despite the high complexity of the minimum graph cut algorithm ($O(N^3)$ [14], it can be afforded. The running time of the minimum graph cut is ignored, compared to the high convergence time of the GA.

5 Evaluation

This section describes the evaluation procedure and the obtained results of the GA-based mapping compared to other techniques.

5.1 Tuning and Testing Procedures

Random topologies are generated for tuning and testing purposes as follows:

- The number of links in the generated topology is in the range $[N-1, \frac{N(N-1)}{20}]$, where N is the number of simulated nodes.
- The traffic passing through each link is in the range $[1, 1000]$ Mbps.

a- Tuning. The following procedure is used to tune the parameters:

- Tuning used three categories of topology size: Small topologies of 10 simulated nodes, medium topologies of 50 simulated nodes, and large topologies of 100 simulated nodes. Ten random topologies are generated in each category.
- A set of possible values is defined for each parameter to be tuned.
- For each combination of values of the tuned parameters, the GA-based mapping uses these values to find a solution for the random topologies generated for tuning purpose.
- The final GA-based mapping uses the combination that gives the highest average time gain while minimizing the average utilization loss.

b- Testing. For testing, ten random topologies with sizes $\{10, 20, 30, \ldots, 100\}$ simulated nodes, are generated using the same procedure described earlier.

5.2 Environment and Parameters Configuration

The proposed GA-based mapping is evaluated using the CRC testbed [15]. Table 2 defines all the used parameters based on the CRC testbed and based on the performed parameter tuning.

All the computations are performed on the CRC testbed server (Intel(R) Xeon(R) Silver 4114 CPU @ 2.20 GHz).

Table 2. Parameters configuration

Parameter	Value
P	25 physical nodes
C_p^S	4390 MHz
C^R	1000 Mbps
A	1
B	0.8899
C	0.004197
η	2000 individuals
M_G	0.5
M_I	0.05
α	0.5
γ	0.5

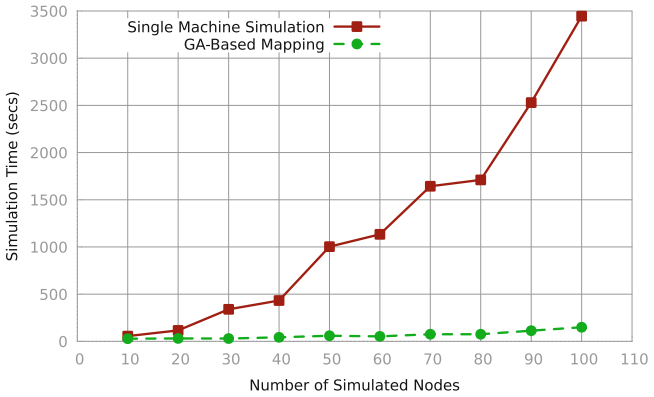
5.3 Comparing GA-Based Mapping to Single Machine Simulation

The effect of using the GA-based mapping is first compared to not using topology mapping, and simulating the whole topology on a single node. As shown in Fig. 4, applying the GA-based mapping reduces the simulation time by more than 90%, while keeping the utilization above 95%.

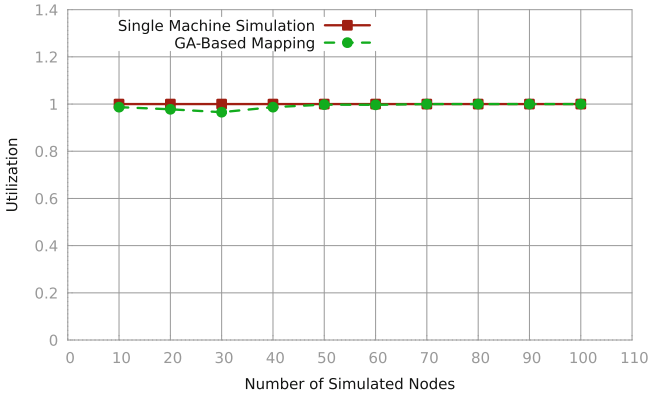
5.4 Comparing GA-Based Mapping to Other Mapping Techniques

Finally, to evaluate the GA-based mapping approach, the algorithm is compared to the following:

- The minimum graph cut technique described in Sect. 4.
- A random mapping approach that randomly maps simulated nodes to physical nodes. There are two possible random solutions: 1- The first feasible mapping found by the search. 2- The algorithm keeps searching randomly for a time equal to the time taken by GA-based mapping to converge, and the best mapping found is considered.



(a) Comparing the simulation time with and without applying the GA-based topology mapping



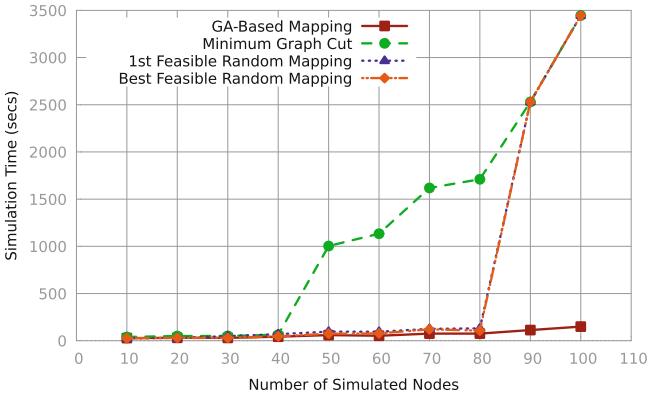
(b) Comparing the utilization with and without applying the GA-based topology mapping

Fig. 4. The performance of the proposed GA-based mapping compared to the single machine simulation

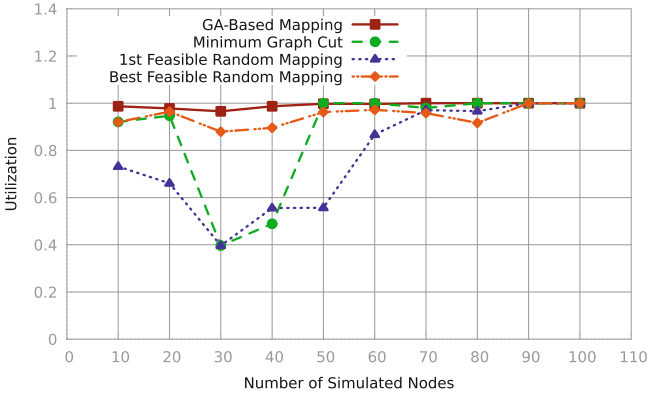
Figure 5 illustrates the comparison between the GA-based mapping and the mapping techniques mentioned in four aspects: The simulation time of the experiment, the utilization of the physical nodes, the number of used physical nodes, and the running time of the used technique.

a- Simulation Time

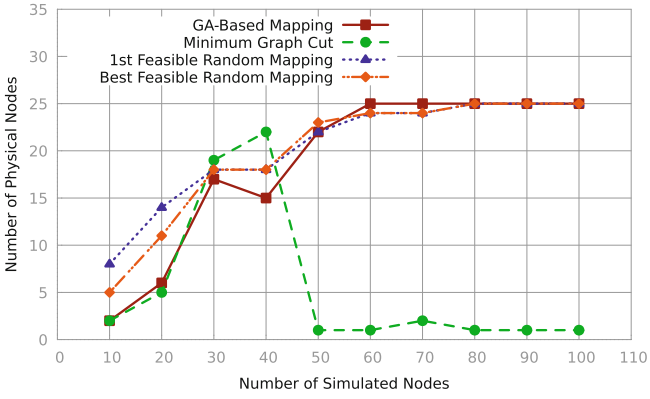
Figure 5a shows that the GA-based mapping achieves the lowest simulation time for all topology sizes. On the other hand, the minimum graph cut technique fails to find a feasible mapping for large topologies ($N > 40$), and therefore its simulation time is almost the same as simulating the whole topology on a single node. Thus the number of physical nodes used by the minimum graph cut for large topologies is fixed to one or two nodes only as shown in Fig. 5c.



(a) Simulation time comparison

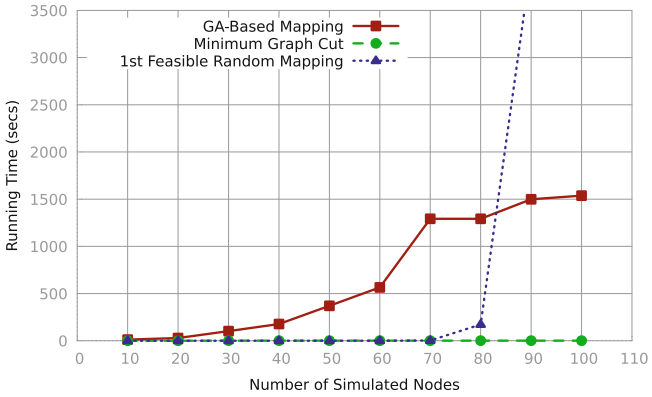


(b) Utilization comparison

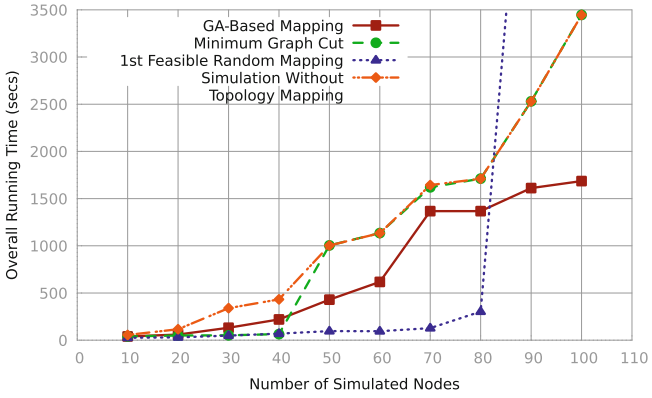


(c) Number of used psychical nodes comparison

Fig. 5. A Comparison between the different topology mapping techniques



(a) Time to find the mapping



(b) Overall time to run the experiment

Fig. 6. Running time comparison

b- Utilization

As shown in Fig. 5b, all of used topology mapping techniques gives high utilization for large topologies, and that is due to the limit of the available physical nodes, as they cannot use more nodes. For smaller topology sizes, the GA-based mapping gives the highest utilization.

c- Number of Physical Nodes

Ignoring the drop in the minimum graph cut curve caused by its failure to find feasible solutions, Fig. 5c shows that the GA-based mapping approximately uses the minimum number of physical nodes. For large topologies ($N > 70$), all of the techniques almost use all of the available physical nodes. However, the GA-based mapping is better in terms of simulation time and utilization, as shown in Fig. 5a and b.

d- Running Time
Figure 6 compares the running time of the three topology mapping techniques. As shown in Fig. 6a, the minimum graph cut technique takes the lowest time to find the mapping for all topology sizes, but running the overall experiment using minimum graph cut takes the longest time, almost the same as running the whole topology on a single physical node without partitioning, as shown in Fig. 6b.

The first feasible solution found by the random search takes low time for small and average topology sizes, but it takes a very long time, or it even might not converge to find a feasible mapping for large topologies.

For the GA-based mapping, it takes higher convergence time than both the minimum graph cut and the random search technique for small topologies, but it achieves the lowest overall time for large topologies. Furthermore, the GA-based mapping finds the mapping that gives the best utilization of the used physical nodes.

6 Conclusion and Future Works

This work proposes a genetic algorithm-based solution for the topology mapping problem and designs a fitness function that allows the GA-based technique to find the mapping that minimizes the simulation time and maximizes the utilization. Furthermore, Simulation time and utilization are estimated using new proposed analytical methods, without actually running the simulation.

As shown in the evaluation, GA-based mapping gives the best simulation time and the best utilization for all topology sizes. On the contrary, both the minimum graph cut and random search approach are not able to find a feasible solution for large topologies. Although the GA-based mapping takes a long time to converge for large topologies, the overall experiment time is less than the experiment simulation time on a single machine. Furthermore, Minimizing the simulation time is more important than minimizing the running time, as the testbed server is always available to run the topology mapping search, while the testbed nodes must be reserved to run the simulation. The running time of the GA-based mapping technique can be afforded given the simulation time reduction and the utilization benefits achieved.

Many future works could be done to enhance the proposed GA-based mapping. Improving the fitness function to include more testbed parameters will improve the found solutions; parameters such as the cost of the physical nodes, and the number of concurrent topology mapping requests. Furthermore, developing an advanced method to benefit from infeasible solutions could also improve the performance of the algorithm. Finally, enabling the algorithm to handle parallel topology mapping requests will allow for better time and resources management.

References

1. Garey, M.R., Johnson, D.S.: Computers and Intractability a Guide to the Theory of NP-Completeness. Freeman and Company, New York (1979)
2. Wette, P., Draxler, M., Schwabe, A.: MaxiNet: distributed emulation of software-defined networks. In: 2014 IFIP Networking Conference (2014)
3. Kreutz, D., Ramos, F.M.V., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. Proc. IEEE **103**(1), 14–76 (2015)
4. Liu, X., Chien, A.: Realistic large-scale online network simulation. In: Proceedings of the ACM/IEEE SC2004 Conference (2004)
5. Yocum, K., Eade, E., Degesys, J., Becker, D., Chase, J., Vahdat, A.: Toward scaling network emulation using topology partitioning. In: 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems, MASCOTS 2003 (2003)
6. Galvez, J.J., Jain, N., Kale, L.V.: Automatic topology mapping of diverse large-scale parallel applications. In: Proceedings of the International Conference on Supercomputing - ICS 2017 (2017)
7. Hoefler, T., Snir, M.: Generic topology mapping strategies for large-scale parallel architectures. In: Proceedings of the International Conference on Supercomputing - ICS 2011 (2011)
8. Ricci, R., Alfeld, C., Lepreau, J.: A solver for the network testbed mapping problem. ACM SIGCOMM Comput. Commun. Rev. **33**(2), 65 (2003)
9. Stoller, M.H.R.R.L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., Lepreau, J.: Large-scale virtualization in the emulab network testbed. In: USENIX Annual Technical Conference, Boston, MA (2008)
10. van Laarhoven, P.J.M., Aarts, E.H.L.: Simulated Annealing: Theory and Applications. D. Reidel, Dordrecht (1988)
11. White, B., et al.: An integrated experimental environment for distributed systems and networks. ACM SIGOPS Oper. Syst. Rev. **36**(SI), 255–270 (2002)
12. Riley, G.F., Henderson, T.R.: The ns-3 network simulator. In: Modeling and Tools for Network Simulation, pp. 15–34 (2010). https://doi.org/10.1007/978-3-642-12331-3_2
13. Davis, L.: Handbook of Genetic Algorithms. International Thomson Computer Press, London (1996)
14. Hagen, L., Kahng, A.: New spectral methods for ratio cut partitioning and clustering. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. **11**(9), 1074–1085 (1992)
15. Hanna, S.S., Guirguis, A., Mahdi, M.A., El-Nakieb, Y.A., Eldin, M.A., Saber, D.M.: CRC: collaborative research and teaching testbed for wireless communications and networks. In: Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization - WiNTECH 2016 (2016)