IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 13, 2021, accepted August 8, 2021, date of publication August 17, 2021, date of current version August 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3104794

A Novel Genetic Algorithm-Based Methodology for Large-Scale Fixed Charge Plus Routing Network Design Problem With Efficient Operators

SAMIRA DOOSTIE, (Graduate Student Member, IEEE), TETSUHEI NAKASHIMA-PANIAGUA, AND JOHN DOUCETTE

Donadeo Innovation Centre for Engineering, Department of Mechanical Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada

Corresponding author: John Doucette (john.doucette@ualberta.ca)

This work was supported in part by the Natural Science and Engineering Research Council of Canada (NSERC) under Discovery Grant 2017-06198.

ABSTRACT In this paper, we present a novel approach that addresses the problem of large-scale network topology design and routing. There are research works that used exact methodologies based on Integer Linear Programming (ILP) models to develop potential solutions for this problem. However, this problem is computationally NP-hard, thus solving it is hugely demanding on computational power for large-scale networks, and in many cases, it is not even possible to generate a solution with a reasonable optimality gap. This paper presents a hybrid algorithm based on the Genetic Algorithm with efficiently designed genetic operators. This algorithm aims to design the topology of large-scale networks and generate a routing configuration for a set of predefined traffic demands on the networks while keeping the total cost of design and routing at a minimum. The results have been compared to an exact ILP model, a relaxed ILP model, and a customized GA as benchmarks for validation purposes. These comparisons showed that the proposed algorithm significantly outperforms the ILP solutions in all of the large-scale network configurations that were used as case studies.

INDEX TERMS Genetic algorithm, large-scale network, optimal topology, routing.

I. INTRODUCTION

Optical transport networks have become widespread, and they have a pervasive role in most typical interactions of modern society. When designing and building out such networks, designers often assume static and known traffic demands and a known and static topology. However, over time, changes in traffic demands, or perhaps maintenance activities or upgrades in some parts of the networks, may drive inevitable topological changes that need to be implemented. To apply these changes, the network's topology needs to be restructured or redesigned.

The network design problem, in its general form, may include (holistically or as several sub-problems) topology design, demand routing and working capacity placement, and survivability routing and spare capacity placement. The topology design problem seeks the optimal topology structure for the networks considering various factors such as cost or certain constraints on the connections among the nodes. The traffic demand routing problem aims to route all of the demands on the network by finding the optimal path for each demand to minimize the total cost of the routing. These two sub-problems can be modeled together in a single ILP as the *fixed charge plus routing* (FCR) problem [1]–[4]. Given a fixed set of nodes, the objective of the FCR problem is to minimize the total cost of the network topology design, demand routing, and working capacity placement. The FCR problem is described in detail in Section I.C.

In the event of some failure, the survivability routing and spare capacity placement problem provides alternative routing on which the network can reroute the affected traffic and ensures sufficient spare capacity to accommodate that routing. There are several survivability mechanisms developed for the restoration process of networks, including *automatic*

The associate editor coordinating the review of this manuscript and approving it for publication was Hualong Yu.

protection switching (APS), *survivable rings*, *span restoration*, *path restoration*, *p-cycles*, and *shared backup path protection*, to name a few [5].

A number of works in the literature considered the basics of network topology design, routing, and survivability problems using various ILP models [6]–[11]. In addition, several works incorporated heuristics-based approaches in combination with ILPs that also consider the survivable network design problem [12]–[19]. Many of those ILP models or heuristic approaches can find an optimal or near-optimal solution to the specific problems they address, but challenges often arise associated with their processing and/or solution time, especially for large-scale networks [1] and [13].

In work presented in [1], the authors introduced a three-step ILP-based heuristic approach for the mesh-restorable network topology problem, determining the optimal topology, working traffic and restoration routing, and working and spare capacity allocation. This near-optimal approach provided solutions with minimal optimality gaps in most cases and is much faster than addressing the complete *mesh topology routing and sparing* (MTRS) problem in a single step [1]. There are additional works that explored the routing and survivability problem for networks with known topologies. In [20], the authors presented a new heuristic routing method (based on space reduction) as an improvement to the *mixed-integer linear programming* (MILP) runtime for large-scale optical network design. That approach is useful for moderate-sized networks with topologies that are known in advance. Another example of work that deals with known topologies is the work of [21], which developed a heuristic method for transporting demands on a shared-mesh protection network in the event of multiple failures. The work discussed in [22] introduced a two-level evolutionary method for a survivable network topology design problem. The results obtained showed 3% to 7% improvements in runtime relative to those discussed in [1], but only networks with less than 30 nodes were tested.

In the network reconfiguration problem, traffic demands may frequently change [23]–[25]. In [26], the authors presented an arc-chain formulation for the routing problem, given the network topology for medium-size networks. They employed column generation and a generalized upper bounding structure to improve the ILP efficiency.

A. GENETIC ALGORITHM

Employing evolutionary approaches has been an effective way to increase the efficiency of network design problems [27]. Specifically, heuristics and evolutionary algorithms have been proposed as alternative solution approaches for the FCR problem [28]–[30] to improve the runtime. *Genetic algorithms* (GAs) are a meta-heuristic that has received a lot of attention in recent years and have been employed in solving a broad cross-section of mathematically complex problems in diverse areas, including operational research [31], healthcare management [32], agriculture [33], path planning [34], and robotics [35], to name a few.

A number of works in the literature initially employed GA-based approaches for network design problems [36]–[39]. They employed basic genetic operators such as single-point crossover and random binary flip mutation, which can result in generating infeasibilities in the offspring. In general, their results were not compared against any deterministic benchmark to verify the optimality gap. In [40], a GA-based approach for digital-data-network design was presented. They encoded a binary string as their chromosome structure and employed uniform crossover and random swap mutation as their genetic operators. Their GA starts with a random initial population, and they have considered only the fixed cost of topology design and did not consider the cost of working traffic flow assignment in their model. The comparison between the presented GA-based approach and a tabu search in [40] showed that the results are almost the same with regard to the network design cost and the processing time. The work in [41] and [42] employed genetic algorithms to design minimum-cost networks that are restorable against single-span failures. They considered a fixed network topology and defined a set of pre-enumerated routes for transferring the working traffics (i.e., working routes) and restoring every working route (i.e., backup routes) in the event of single-span failures. The work in [41] presented a GA-based algorithm for minimizing the network cost while protecting it against single failures for a network with a predefined topology. Their chromosome structure consists of a string of working and backup paths for every traffic demand. They employed single-point crossover and random replacement mutation. Their results showed smaller costs and much longer processing runtimes in comparison to tabu search. In [42], the authors presented a GA-based algorithm for capacity and routing assignment on a network with fixed (known) topology and subject to single failures. Their aim was to minimize the total working and backup capacities on the network. Their chromosome structure consists of a set of working and backup routes for every traffic demand and they employed single-point crossover and random replacement of paths as their mutation operator.

GAs have also been used as an interim step of hybrid solution approaches, for example, to limit the search space for an ILP's initial solution. In [43], a hybrid ILP-GA approach was used to find a minimum-cost set of preselected cycles on a fixed-topology network design problem with 200 nodes. They defined their chromosome's structure as a set of pre-enumerated cycles within the network topology and employed single-point crossover and random swap mutation as their genetic operators. The authors in [44] presented an approach to design large-scale optical networks using a customized GA (CGA) to find a near-optimal design. They considered a binary string as their GA chromosome structure to represent the network's topology. In their chromosome structure, 1 represents the existence of a span in the network and 0 otherwise. They used a single-point crossover and a random binary flip mutation as their genetic operators. As mentioned by the authors in [44], the utilized random

operators can easily generate infeasible solutions (i.e., disconnected network topologies). Thus, they employed a repair operator that detected the infeasibilities and repaired them.

B. RESEARCH GAPS AND CONTRIBUTIONS

We have identified several shortcomings that commonly arise in the existing literature and developed an Improved Genetic Algorithm (IGA) in a manner that seeks to overcome the following common weaknesses:

1) Weakness: Generating a high-merit initial population is time-consuming, due to the stochastic nature of the process and subsequent repair actions that are often required to transform infeasible solutions into feasible solutions.

Response: In our work herein, we designed our GA to ensure that the solutions generated for the initial population are connected network topologies; hence, they are inherently feasible (Section II.B.1).

2) Weakness: Genetic operators (typically crossover and mutation) are often inefficient if great care is not given when designing them, specifically for the chromosome structure that frequently results in offspring in need of repair.

Response: In our work herein, we designed crossover and mutation functions that seek to satisfy the problem's constraints, thus avoiding the need for a computationally expensive repair process (Sections II.B.1.1 and II.B.1.2, II.B.2.2, and II.B.2.3).

3) Weakness: The complete parameter space of the control parameters (e.g., crossover and mutation rates) is vast, and although the precise settings for those parameters can have a very significant effect on the convergence rate of the GA and the subsequent processing time, many GAs in the literature do not specify any particular efforts to tune those parameters.

Response: In our work herein, we have carefully analyzed and tuned these parameters for the GA-based approach (Section II.B.3).

4) Weakness: The validation of the results of the non-deterministic approaches is an important step for analyzing their effectiveness. Many GA's in the literature were only compared against other GAs or other non-deterministic approaches.

Response: In our work herein, we have validated our results using an ILP model as a benchmark, which can find the optimal solution for cases where the complexity of the problem allows an ILP model to be used (Sections III.A and III.B).

C. THE FIXED CHARGE PLUS ROUTING (FCR) PROBLEM

The network design problem that we are addressing in this work is the FCR problem [1], where we seek to jointly optimize the topology and working routing (and subsequent working capacity allocation) to serve all traffic demands. Although the FCR problem does not originate with us, we will present the full ILP herein, for completion.

In the FCR problem, by optimizing the topology, we are aiming to establish the number of the required spans with the least cost possible and by the working routing optimization, we are seeking the cheapest working routes between the end nodes of every traffic demand for transferring the traffic. Two cost components define the total cost of a designed network:

1) The fixed cost (F_{ij}) is the cost of establishing the span between nodes i and j.

2) The capacity cost (C_{ij}) is the cost of transferring one unit of the traffic on the span between nodes i and j. For simplicity, hereafter, we refer to the span between nodes i and j as span ij. Based on the defined cost components, the total cost of a network with a set of established spans and routed traffic demands over those spans is calculated using (1).

$$Cost = \sum_{ij \in A} (F_{ij} \times \delta_{ij} + C_{ij} \times w_{ij}) \tag{1}$$

Here, A is the set of all the possible spans connecting the nodes, δ_{ij} is a binary decision variable which has the value of 1 if span ij exists in the topology and 0 otherwise, and w_{ij} is the total number of the traffic units on span ij. The goal is to minimize the total cost mentioned above subject to the traffic demand constraints, as described in (2) through (6) [1]. The ILP model of the FCR problem, which aims to minimize the value of (1), contains (1) to (6).

$$\sum_{nj \in A} w^r_{nj} = d^r \quad \forall r \in D, n = O[r] \tag{2}$$

$$\sum_{jn \in A} w^r_{jn} = d^r \quad \forall r \in D, n = T[r] \tag{3}$$

$$\sum_{in \in A} w^r_{in} - \sum_{nj \in A} w^r_{nj} = 0 \quad \forall r \in D,\ \forall n \notin \{O[r], T[r]\} \tag{4}$$

$$w_{ij} = \sum_{r \in D} w^r_{ij} \quad \forall ij \in A \tag{5}$$

$$w_{ij} \leq w_\infty \cdot \delta_{ij}; \quad \delta_{ij} \in \{0, 1\};\ \forall\, ij \in A | i < j \tag{6}$$

Here, w^r_{ij} represents the number of working traffic flows for demand r on the span ij, and d^r represents the number of flow units of demand r. Also, $O[r]$ and $T[r]$ represent the origin and destination nodes of demand r, respectively. Equations (2) and (3) ensure that the total flow from the demand's origin node to the destination node is precisely equal to the demand units. Equation set (4) makes sure the demand units pass entirely through the intermediate nodes. Equation set (5) places enough working capacity on each span to carry the flows of overlapping demands. Equation set (6) makes our decision variable maintain their structure throughout the solution, whether they are binary or integer.

The FCR problem has been shown to be NP-hard [1]. Although the problem can be solved efficiently for relatively small networks by an appropriate commercial ILP solver (e.g., Gurobi™, [45]), it is generally too computationally complex for large-scale networks [1]. Solving the FCR problem on a suite of test case networks as represented in Table 1 ranging in size from 40 nodes to 150 nodes, we observe the runtime behavior documented in Table 2.

II. PROPOSED METHODOLOGY

In solving the Fixed Charge plus Routing problem, the size of the set of candidate spans for the network's topology is considerable and selecting an optimal subset of those candidate spans is a massive combinatorial problem; as stated above, FCR is an NP-hard problem. Herein, we employ GA to solve the Fixed Charge plus Routing problem for large-scale network design.

A. GENETIC ALGORITHM BUILDING BLOCKS

In GA, a *population* of candidate solutions, or *individuals*, is evolved into subsequent *generations* in an iterative fashion until some termination criterion is met. The first step in a GA is the generation of an *initial population* of individuals. This is often done via some random or pseudo-random process. In the context of the FCR problem, this could be done via random selection of eligible spans to achieve full connectivity, for example, upon which we would then apply shortest path routing for working capacity allocation. Each individual is then evaluated with respect to some measure, typically the objective function of the optimization problem; this is the individual's *fitness*.

A *selection* process is then used to select some subset of the population, called *parents*, upon which *genetic operators* will act. The two most common genetic operators are *crossover* and *mutation*. In a crossover, the properties or characteristics of two or more individuals are combined in some fashion to produce one or more new individuals, called *offspring*. In mutation, one or more properties or characteristics of an individual are changed in some way to create offspring.

The properties of an individual are represented by a *chromosome*, which consists of a set of *genes*. In general, a chromosome is a string of data that fully encodes the properties of an individual solution to the problem. For the FCR problem, the chromosomes used would typically include genes that represent whether the various eligible spans are present in the solution. As an example, the network shown in Fig. 1 (where dark lines represent spans that are present in the solution and the light blue colored spans are not present) could be represented by the chromosome shown.

A general schematic of an individual chromosome with four genes and a population is depicted in Fig. 2 and Fig. 3, respectively. The individuals in the initial population can be evaluated using their fitness values. For GA to evolve the chromosomes and generate future populations with better fitness, the breeding process should be utilized. The breeding process in GA is facilitated by a set of genetic operators and results in the generation of a new set of chromosomes which will be parts of the next population.

The two classes of genetic operators are named *crossover* and *mutation*, which are responsible for deriving new chromosomes from the already existing ones in the population. A crossover operation takes the chromosomes of two or more individuals from the population and combines their genes in some fashion to generate two or more new solutions. In this process, the selected individuals are called *parents,* and the newly generated individuals are called *offspring*. The offspring resulted from a crossover operator inherit genes from both parents. On the other hand, mutation usually takes one individual's chromosome and applies a set of operations on its genomes to make some alterations and create a new offspring. The new offspring resulted from a mutation operator may

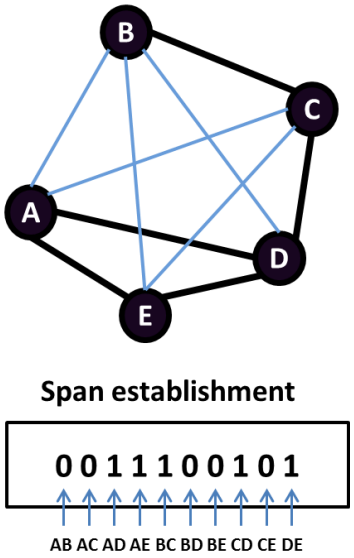

FIGURE 1. A GA chromosome representation of a network, including two genes as the established spans and their working capacities.

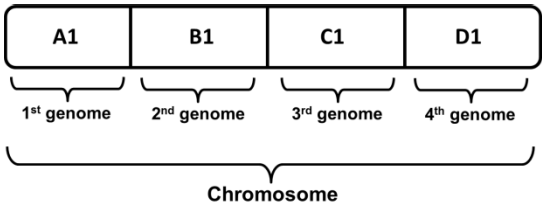

FIGURE 2. The general structure of an individual chromosome with its four constituent genes that are named A1, B1, C1, and D1.

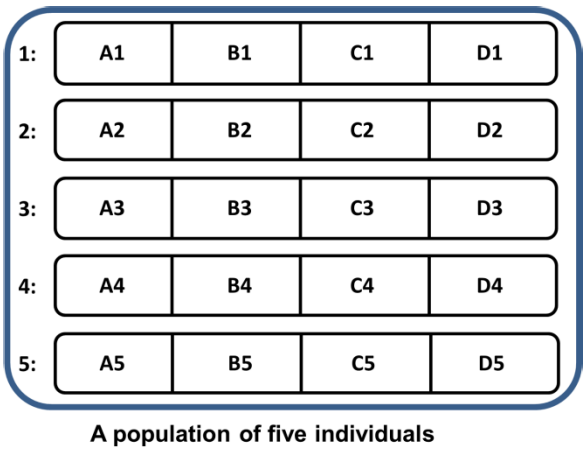

FIGURE 3. The general structure of a population including five individuals.

have some of its ancestor's features and information in its new genes. Therefore, by applying the genetic operators to the current members of the population, the current population members will breed among each other and generate a new population. If the chromosome structure and the genetic operators are well designed, we will observe an improvement in the populations' fitness in every iteration. This iterative algorithm will stop when it meets termination criteria such as a specific individual's fitness, passing a certain number of iterations, or a specific runtime. Fig. 4 shows a crossover operation applied on two chromosomes (chromosomes 5 and 3 from the population from Fig. 3) by swapping two genes between them. The resulted offspring inherited two genes from each parent. Fig. 5 demonstrates a general mutation process on a selected individual. In this mutation, the second and fourth genes of the chromosome are exposed to some alterations, and as a result, their old structure as B2 and D2 has been replaced by B'2 and D'2, respectively.

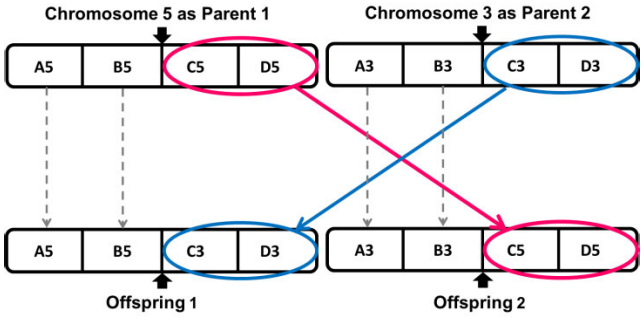

FIGURE 4. The general procedure of a basic crossover between two individual's chromosomes.

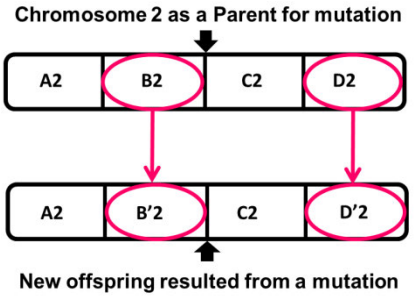

FIGURE 5. The general procedure of a fundamental mutation on two genes of an individual's chromosome.

To illustrate the general GA procedure, a high-level pseudo code of GA has been described in Fig. 6.

B. IMPROVED GA OPTIMIZATION ALGORITHM

In our work, we developed a GA-based optimization approach that consists of two well-designed GAs. The first GA is referred to as the *Initial-Population-Generator*, or *IPG-GA*, and is based upon the concept of the *Travelling Salesman Problem* (TSP) [46]. In TSP, the goal is to find

```
1. Generate the initial population
2. Evaluate the members of the population (their fitness);
3. While not meeting the termination criteria:
        4. Select some members of the population as
        crossover parents (pc);
        5. Apply crossover and obtain the resulted
        offspring;
        6. Evaluate the offspring. If "is infeasible":
                Go to "Repair operator";
           else:
                Add the offspring to the population;
        7. Select some members of the population as
        mutation parents (pm);
        8. Apply mutation and get the resulted offspring;
        9. Evaluate the offspring. If "is infeasible":
                Go to "Repair operator";
           else:
                Add the offspring to the population;
        10. Sort the population based on their objective
        function value;
        11. Truncate the population to maintain the same
        number of members;
```

FIGURE 6. Basic GA pseudocode.

the shortest cycle that passes through all nodes once and only once, i.e., the shortest *Hamiltonian cycle* [47]. Taking inspiration from the TSP, IPG-GA generates a set of Hamiltonian cycles as feasible but sub-optimal network topologies. The generated network topologies are considered as the set of initial feasible solutions to the network topology design and routing problem. The structure of the IPG-GA chromosome is depicted in Fig. 7. Here, the genes store the order of the selected nodes, which consequently represent the spans that form a Hamiltonian cycle as the network topology. Based on the installation cost of every selected span, the total topology cost of the network is calculated as the fitness function value of the individual. The details of the IPG-GA are discussed below, in Section II.B.1.

Once a sub-optimal solution for the IPG-GA is achieved, a routing function is called to route all of the demands on the current network. The second GA, which we call the *Master-GA*, is responsible for simultaneously improving the topology of the network and routing the demands. The structure of the chromosome for the Master-GA is depicted in Fig. 8. The genes encode the spans selected in the solution as binary values (1 if the span is selected, and 0 otherwise). A routing function routes all of the demands and assigns the related working capacities on the spans encoded in the genes, and determines the total working capacity that arises on each span. As with the IPG-GA, the fitness function is the total cost of the individual, though that cost now also includes capacity costs. The details of the Master-GA are discussed below, in Section II.B.2. The high-level steps of the IGA, including both IPG-GA and Master-GA, are presented in Fig. 9.

The proposed IGA algorithm includes two genetic algorithms: IPG-GA and Master-GA. The IPG-GA generates a set of sub-optimal Hamiltonian cycles as bi-connected network topologies. In other words, IPG-GA provides the Master-GA with a set of bi-connected network topologies that are of relatively good fitness. However, as the cyclic topology of

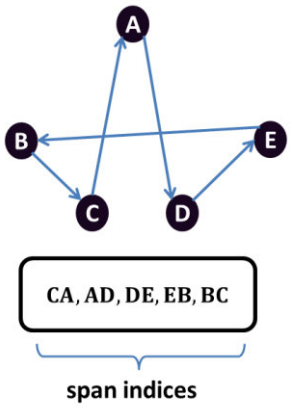

FIGURE 7. Structure of the chromosomes in IPG-GA that demonstrates a genetically coded Hamiltonian cycle as the network topology.

networks does not necessarily represent the optimum network topology in terms of the network topology and routing cost, we implemented the Master-GA to evolve the network topologies further. The detailed steps of the proposed IGA algorithm and its operators have been described in the following sections. Following a convention in some of the literature ([36], [39]–[42], [44]), the setting of the parameters was made according to our pre-assessment analyses.

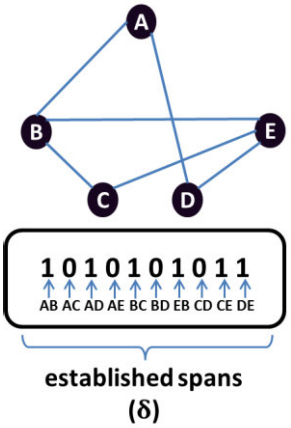

FIGURE 8. Structure of the chromosome in Master-GA that demonstrates the complete topology of the network and assigned working capacities to the established spans in the network.

1) IPG-GA

As mentioned above, the well-known TSP seeks to find the shortest path in a network by visiting all nodes once and only once; in network design and routing problems, the shortest path is often the lowest cost one. As we seek to minimize the total cost of network design, the optimal solution would be the cheapest network topology. In the discussion above, we introduced the IPG-GA, which is responsible for generating a set of sub-optimal Hamiltonian cycles where optimality refers to the cost of the cycles. The IPG-GA starts with developing a set of Hamiltonian cycles among the nodes by generating 60 random ordered sets of all nodes resulting in 60 cycles in each generation. To generate new members for the next generation from the current one, IPG-GA benefits from two genetic operators: crossover and mutation. In each generation, 65% of the population undergoes mutation, and 65% undergoes crossover. The IPG-GA proceeds for 60 iterations, and as it proceeds iteration by iteration, the total cost of the cycles in the upcoming generations either will be the same as the current generation or will be less. Thus, the sub-optimal set of Hamiltonian cycles will be achieved in the last generation of the IPG-GA. The set of developed sub-optimal cycles represent the topology of the networks. However, the traffic demands have not been routed over the network, and thus, the working capacities have not been assigned to the networks' spans yet. For every cycle in the final generation of IPG-GA, we used an assignment algorithm that routes all of the traffic demands between their origin and destination nodes. For every traffic demand whose end nodes are directly connected by a single span in the network, a number of working traffic units equal to the traffic demand are assigned to that span. The other traffic demands' end nodes are on the cycle, but they are not directly connected by a single span in the network. In other words, these end nodes partition the cycle into two incomplete cycles, either of which can be considered as a route for the demand. The traffic should be routed over the spans on either side of the cycle. To decide over which side of the cycle a demand should be routed, we calculate the total capacity cost of traffic routing on the spans on either side and choose the one with the least total cost. Once for every cycle from the last generation of the IPG-GA, all of the traffic demands have been routed, we obtained an initial population to be fed into the Master-GA for further improvement in the topology design and routing cost. The pseudocode of IPG-GA is shown in Fig. 9.

a: IPG-GA CROSSOVER

In IPG-GA, the chromosome structure is based on a random order of all of the nodes. Every two consecutive nodes represent a span. The set of all of the spans associated with the set of the random order of the nodes is stored as the genes of the chromosome. We employed two crossover functions to be applied to 65% of the current population in each iteration. First, we selected crossover parents using uniform random selection. Based on our pre-assessment analysis, both two-point and single-point crossovers generated diversity in the population. However, the two-point crossover was more effective compared to the single-point crossover. Thus, for every pair of selected parents, we applied either single-point crossover with 0.25 probability or two-point crossover with 0.75 probability.

In single-point crossover (depicted in Fig. 10(a)), a nodal index was selected using a uniform random process to be the crossover point, where we partition the chromosomes of both parents into two parts. Two new offspring are generated by combining the first part of one parent with the second part of the other parent and vice versa.

1. Initialize the population
2. Evaluate the members using the objective function
3. While not meeting the termination criteria:
    4. Select crossover parents
    5. Obtain the resulted offspring
    6. Evaluate the offspring
    7. Select mutation parents
    8. Obtain the resulted offspring
    9. Append the offspring to the current population
    10. Sort the population
    11. Truncate the population (maintain the same population size)
12. Route the demands on the last population of solutions
13. While not meeting the termination criteria (reaching 20 iterations):
    14. Select crossover parents
    15. Apply the designed crossovers and obtain the offspring;
    16. Evaluate the offspring
    17. Select mutation parents
    18. Apply the designed mutations and obtain the offspring;
    19. Evaluate the offspring
    20. Append the offspring to the current population
    21. Sort the population;
    22. Truncate the population to maintain the same number of members

IPG-GA

Master-GA

FIGURE 9. Proposed IGA Pseudocode including the IPG-GA and Master-GA.

Similarly, in a two-point crossover (depicted in Fig. 10(b)), we select two crossover points (using the same approach as for single-point crossover) and partition the chromosomes of both parents into three parts. Two new offspring are subsequently generated by exchanging the middle part of one parent with the middle part of the other, and vice versa. In some cases, the crossover process creates an infeasible solution, where some nodes appear twice in a chromosome, and other nodes are absent. A repair mechanism follows the crossover to check for such incidents and swaps duplicate nodes with absent nodes. A schematic of this process is shown in Fig. 10. The first time a duplicate member appears in the offspring, it is replaced by a duplicate member of the other offspring. The one-by-one correspondence for this replacement is based on the indices of the duplicate members in the second part on the parents. As an example, let us consider the two parents represented in Fig. 10(a), assuming the partitioning point is after the fifth member. The common members between the second part of the second parent and the first part of the first parent are 5 and 2. Similarly, the common members between the second part of the first parent and the first part of the second parent are 1 and 4. Considering the found duplicate members in the offspring, the first time members 5 and 2 appear in the first offspring, they are replaced by 1 and 4, respectively. Similarly, the first time members 4 and 1 appear in the second offspring, they are replaced by 2 and 5, respectively.

b: IPG-GA MUTATION

We utilized three different mutation functions: (1) two genes chosen by a uniform random process are swapped, (2) two genes are chosen by a uniform random process, and the entire sequence of genes between them (inclusive) is reversed, and (3) one gene chosen by a uniform random process is removed and inserted in a new location chosen by a uniform random process. Parent selection was made using the same approach described above for crossover. For each parent selected for mutation, we use the first function with 0.25 probability, the second with 0.50 probability, and the third with 0.25 probability. The three mutation functions are illustrated in Figs. 11(a), 11(b), and 11(c), respectively.

2) MASTER-GA

The output of IPG-GA (i.e., its last generation) is a set of high fitness Hamiltonian cycles. Each of these cycles represents a feasible network topology that would satisfy the basic connectivity constraints of the FCR problem. However, these cycles will not necessarily include the optimal topology. Thus, the Master-GA is employed to improve this set of topologies and converge to a near-optimal solution. We chose the top 65% of the chromosomes from the last iteration of the IPG-GA to be considered as the initial population for the Master-GA. The pseudocode of Master-GA is shown in lines 13-22 of Fig. 9.

a: MASTER-GA-ASSIGNMENT FUNCTION

In Master-GA, first, the topology of the network evolves using the genetic operators. Then using the *assignment* function, all of the demands are routed over the network's topology, and the required working capacities are allocated on the spans. We designed an improved Dijkstra's routing and assignment function to route the traffic demands and assign working capacities to spans based on the proposed routing operator in [44].

Given a set of traffic demands to be routed between their origin and destination nodes and the cost associated with routing the demands over every span in a network, Dijkstra's algorithm [48] finds the minimum-cost route for every demand between its origin and destination nodes. The *assignment* function employs an improved Dijkstra's algorithm to find the minimum-cost routes for every demand on the network. The cost of every route is determined based on the fixed cost of span establishment for every span along the route, plus the cost of transferring the traffic units on the spans along the route. Therefore, the cost matrix is comprised of both the span establishment and per-unit traffic routing costs, and once the demand is routed on the network, the fixed cost

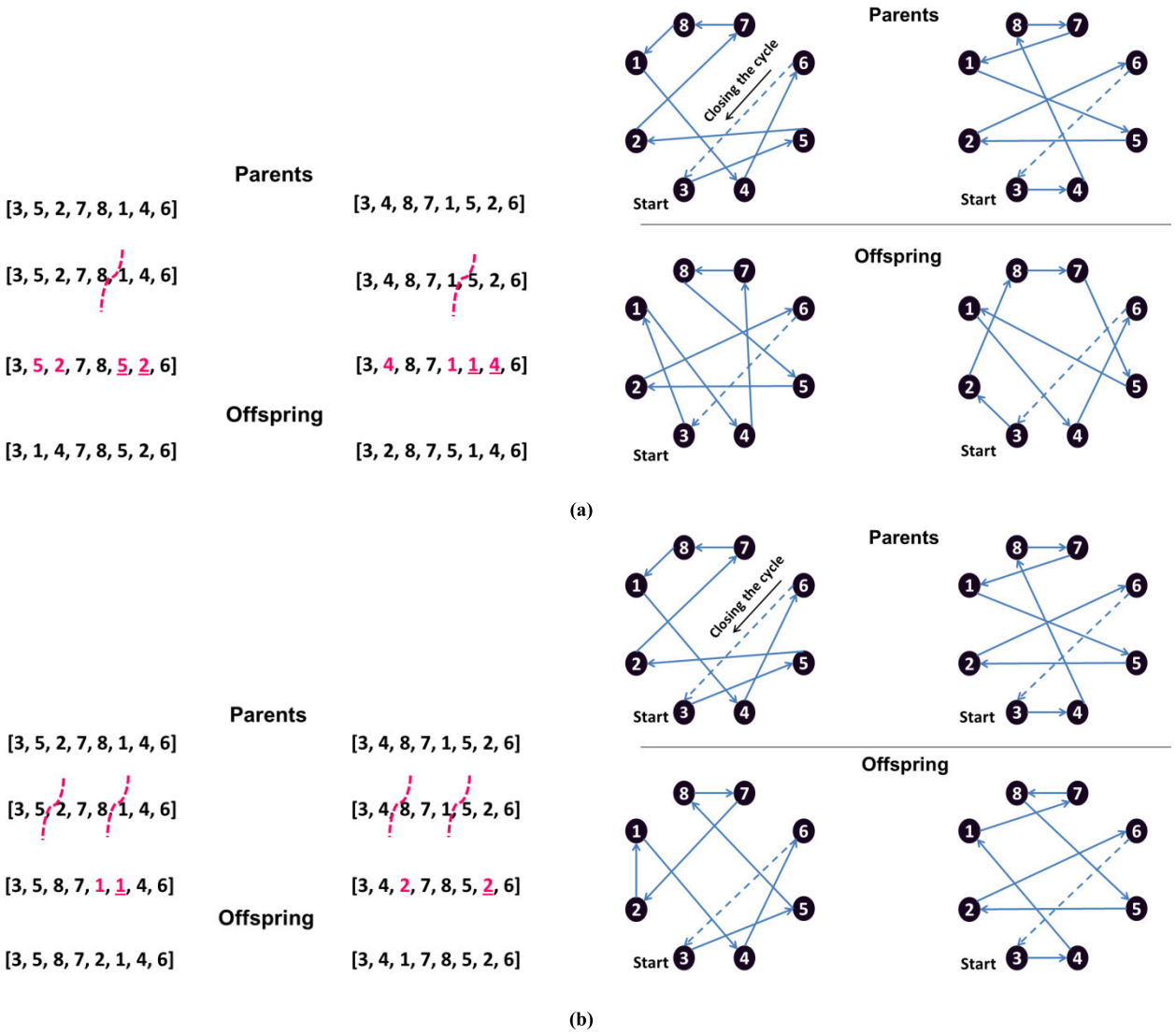

FIGURE 10. IPG-GA crossover operators; (a) Single-point crossover, (b) two-point crossover.

of the spans that form the route will be set to zero for the remaining demands; for already established spans, the cost matrix in Dijkstra's algorithm will only include the per-unit cost of routing the traffic.

In addition to the dynamic cost matrix for routing, the order of routing the demands have also been adjusted for routing. Demands have three components: 1) origin node, 2) destination node, and 3) the units of the traffic demand (d^r). Here, they are sorted in descending order of their demand unit values; demand relations with the most demand units (i.e., with maximum d^r) are routed before those with fewer demand units.

b: MASTER-GA-CROSSOVER

We employed two crossover functions to be applied to 55% of the current population in every iteration. The percentage of 55% has been selected based on a comprehensive set of analyses that showed us the best crossover rate is between 50% and 60%. A detailed description of our analysis has been provided in Section II.B.3. For a crossover operation, we needed two parents to be selected from the population. For every parent to be selected, we performed a tournament selection with a tournament size of 10. According to our pre-assessment analysis, tournament selection resulted in better objective function values compared to random selection. In tournament selection, the best solution among a set of randomly selected solutions will be selected for crossover and mutation, and having both the randomness and elitism mechanisms can lead to a better result as opposed to only random selection. We repeated this process until 55% of the population was selected as crossover parents. For every pair of selected parents, we applied *master-crossover1* with 0.1 probability or *master-crossover2* with 0.9 probability. The master-crossover1 function takes two members of the

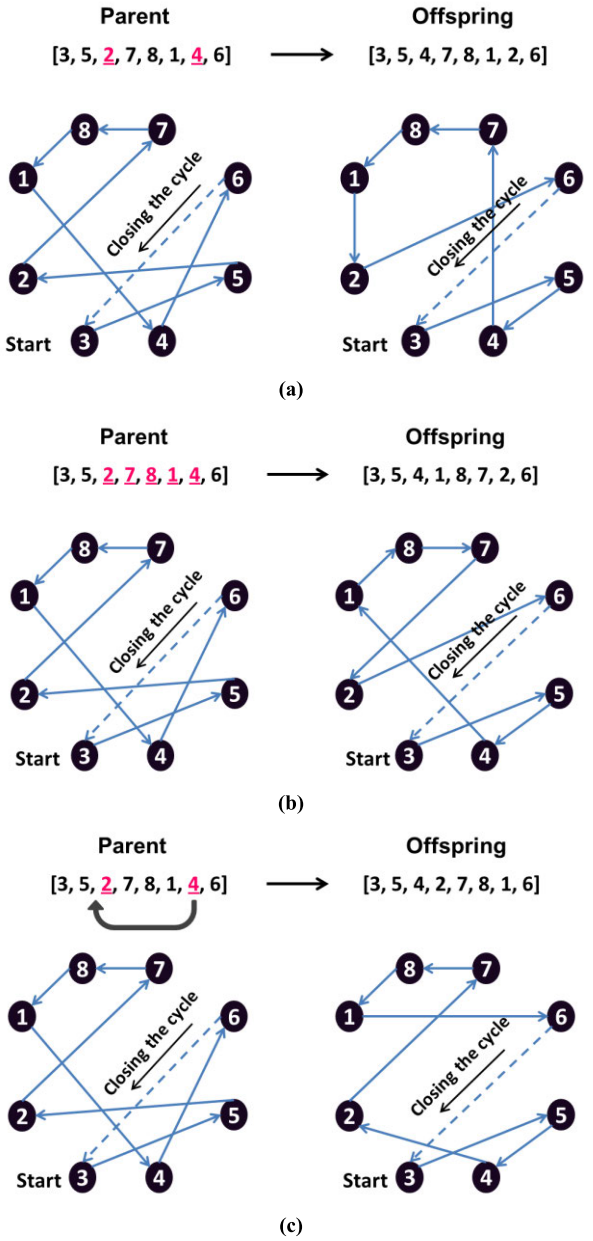

FIGURE 11. IPG-GA mutation operators; (a) swapping elements, (b) reversing a part of the chromosome, (c) moving an element from its position to a new position in the chromosome.

population as parents and randomly removes 20% of the spans from both of them using uniform random selection. For each removed span, one span from the set of existing spans in the second parent was appended to the first parent such that the newly-appended span has one node in common with the removed span from the first parent. The procedure of the master-crossover1 function has been illustrated in Fig. 12. This crossover generates new offspring using new combinations within the already installed spans. Thus, it increases the diversity of the population within the visited areas of the search space. In other words, it promotes the exploitation ability of the search algorithm.

The master-crossover2 function takes two selected parents from the population and forms a set of spans that are absent in both of the parents (i.e., the set of all of the eligible spans except those in at least one of the parents). Then, using uniform random selection, master-crossover2 removes a random percentage of the spans between 50% and 90% of the parent's spans individually. We selected this range (50 and 90) based on our pre-assessment analysis, during which we found span removal percentages greater than 50 and less than 90 resulted in higher convergence rates. For each parent, from the set of absent spans, the master-crossover2 selects 10% of spans to be appended to that parent using uniform random selection. An illustration of master-crossover2 for one attempt of span removal and span appending process has been shown in Fig. 13. In this example, the span between nodes 5 and 6 (i.e., span 56) and span 25 have been removed from parents 1 and 2, respectively. Then, from the set of absent spans, one span has been selected using uniform random selection to be appended to each offspring. As this crossover generates new offspring using absent spans (spans that were not installed in the selected parents yet), it increases the diversity of the population by exploring new areas of the search space.

c: MASTER-GA-MUTATION

We utilized four different mutation functions:

1) The most expensive span in the selected parent was replaced with another span with the least cost such that it has one end node in common with the replaced one. From the ascending ordered set of spans (i.e., sorted from the minimum to maximum cost), the first span that has an end node in common with the removed span will be added to the network.

2) Using uniform random selection, 40% of the spans in the network were selected. For every selected span, if the nodal degree of each end node of the span was greater than 2, that span was removed.

3) From the set of existing spans in the network, a randomly selected percentage of the spans between 20% and 50% of the spans were removed using a uniform random selection process.

4) From the set of absent spans in a selected parent, 20% of them were selected using uniform random selection and appended to the offspring.

Parent selection was done using the same approach described above for crossover. For each parent selected for mutation, we use the first function with 0.2 probability, the second with 0.3 probability, the third with 0.25 probability, and the fourth with 0.25 probability. Based on our pre-assessment analysis, we found the effects of the four mutation functions on the convergence rate of the algorithm almost the same with observing small improvement from the second mutation function compared to the first one. The first two mutation functions are illustrated in Figs. 14(a) and 14(b), respectively.

Although the mutation functions increase the exploration ability of the algorithm, there is a chance they would generate partitioned network topologies and hence, infeasible

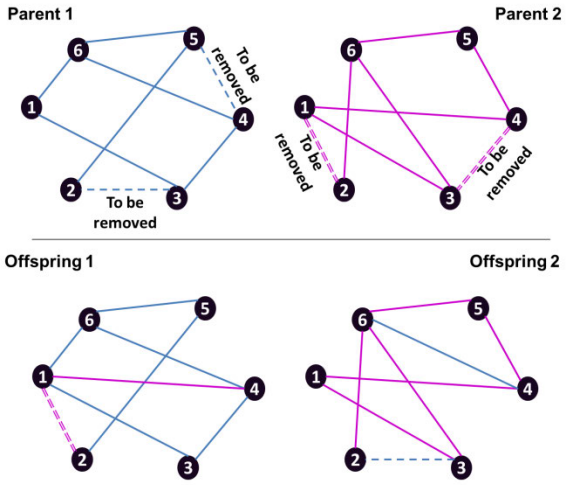

FIGURE 12. Master-GA, master-crossover1 function; Dashed spans in the parents are to be removed while the dashed spans in the offspring are inherited from the parents.

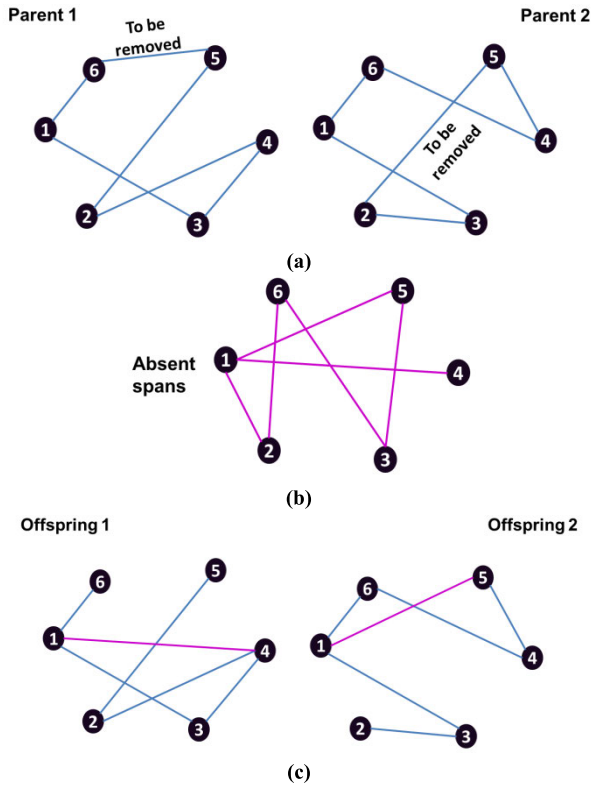

FIGURE 13. Master-GA, master-crossover2 function; (a) two parents with indicated spans to be removed, (b) the set of absent spans in both of the parents, and (c) the generated offspring.

solutions. As most of the designed genetic operators try to keep the offspring feasible, the small chance for generating infeasibilities is associated with the randomness of selections. To avoid an unnecessary computational cost, the infeasible chromosomes did not go through the fitness evaluation process and were removed from the population. Moreover, as we initiated the Master-GA using a fully feasible population

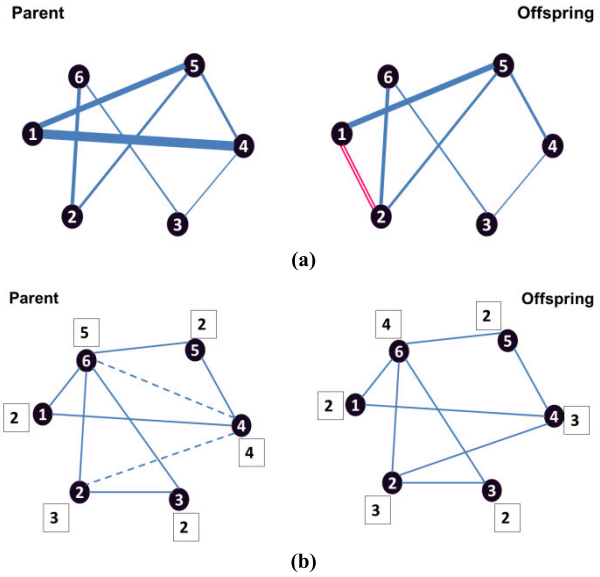

FIGURE 14. Master-GA mutations; (a) Cost-based mutation; the double-lined span is the newly installed span in the offspring, (b) Nodal degree-based mutation; the total nodal degree has been indicated next to every node, and the dashed-spans are the ones with highest nodal-degree end-nodes.

from IPG-GA, in the worst-case scenario, even if all of the offspring are infeasible, we would still have a sub-optimal feasible solution to our problem.

3) CONTROL PARAMETERS

There is a set of input parameters that directly control the efficiency of the algorithm. These parameters directly affect the processing time and the convergence rate of the algorithm. Setting up these parameters below a threshold can weaken their effects on the efficiency of the algorithm for converging toward the optimum solution while setting them up above that threshold can cause elongated processing times, which degrades the computational efficiency of the algorithm.

We performed a comprehensive set of analyses for tuning these control parameters for the presented IPG-GA and Master-GA separately. The parameters have been analyzed over 1440 separate runs (i.e., 20 runs per scenario). We analyzed a variety of crossover and mutation rates between 0.3 and 0.8. To select the best crossover and mutation rates, there must be a trade-off between the fitness value improvement and runtime increase. Our pre-assessment analysis showed that for every 10% increase in crossover and mutation rates of the IPG-GA, the average runtime increased by 2.6% and 2.1%, respectively. For example, by increasing the crossover rate of the IPG-GA from 0.6 to 0.7 when the mutation rate was fixed at 0.5, the runtime increased by 2.2%. Also, for every 10% increase in crossover and mutation rates of the Master-GA, the average runtime increased by 8.6% and 13.9%, respectively. Thus, we have to select the crossover and mutation rates that result in the best fitness value improvement while avoiding elongated runtimes.

The average fitness values for every combination of crossover and mutation rates of the IPG-GA are depicted in Fig. 15 and Fig. 16. For fixed values of mutation rate, by increasing the crossover rate of the IPG-GA from 0.5 to 0.6, the average fitness value improves by 1.5%, while by increasing the crossover rate beyond 0.6, according to Fig. 15, the fitness value improvements almost plateau (less than 0.5%). Moreover, for fixed values of crossover rate, by increasing the mutation rate of the IPG-GA from 0.5 to 0.6, the average fitness value improves by 0.3%, while by increasing the mutation rate above this value, the average fitness value improvement is less than 0.2%. Thus, based on the trade-off between the average fitness value improvement and runtime increase, the best threshold for the crossover and mutation rates in the IPG-GA were found between 0.6 and 0.7.

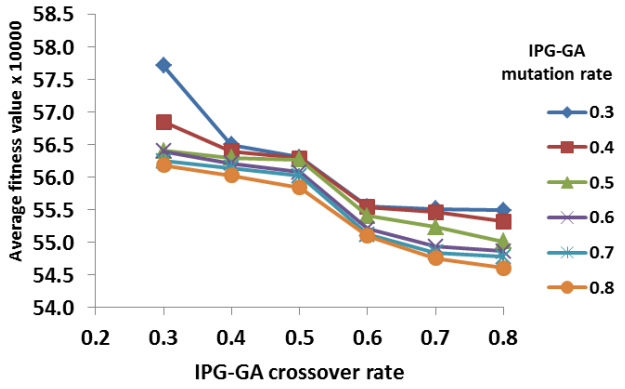

FIGURE 15. Average fitness value improvement with respect to various crossover rates of the IPG-GA, while the mutation rates are fixed.

Similarly, the average fitness values for every combination of crossover and mutation rates of the Master-GA are depicted in Fig. 17 and Fig. 18. For fixed values of the mutation rate, by increasing the crossover rate of the Master-GA from 0.4 to 0.5, the average fitness value improves by 0.2%, while by every 10% increase in the crossover rate beyond 0.5, the average fitness value improvement is less than 0.1%. Moreover, for fixed values of the crossover rate, by increasing the mutation rate of the Master-GA from 0.5 to 0.6, the average fitness value improves by 1.8%, while by increasing the mutation rate to higher values, the average fitness value improvement is less than 0.3%. Thus, for the Master-GA, the best threshold for the crossover rate was found between 0.5 and 06, while the best threshold for the mutation rate was found between 0.6 and 0.7.

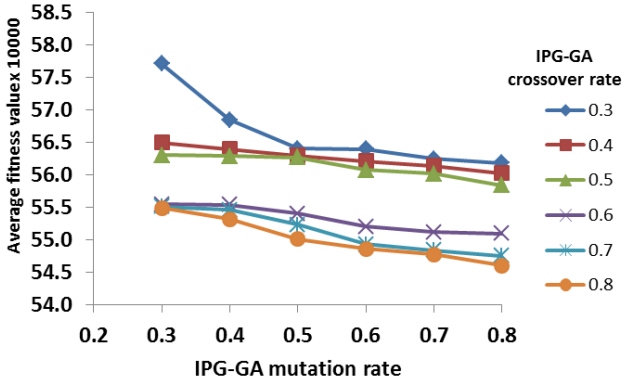

FIGURE 16. Average fitness value improvement with respect to various mutation rates of the IPG-GA, while the crossover rates are fixed.

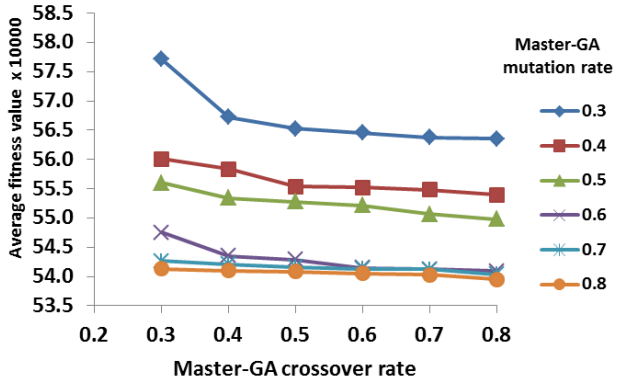

FIGURE 17. Average fitness value improvement with respect to various crossover rates of the Master-GA, while the mutation rates are fixed.

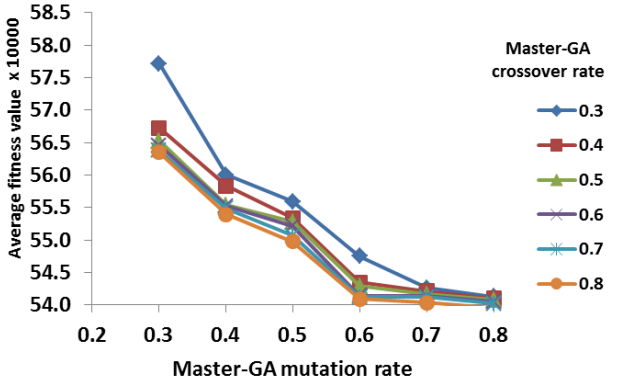

FIGURE 18. Average fitness value improvement with respect to various mutation rates of the Master-GA, while the crossover rates are fixed.

III. EXPERIMENTAL RESULTS AND DISCUSSION

A. EXPERIMENTAL SETUP

We now compare the results from the proposed IGA against those from the benchmark ILP model described by (1) through (6), a relaxed form of that ILP, and a customized GA (CGA) based on [44] for the network topology design and routing problem. We used a Microsoft Windows Server 2012 R2 Standard x64-based PC, Intel(R) Xeon(R) CPU E5-2650 v3 running at 2.30 GHz with 128 GB RAM to solve the IGA in python 3.7 [49] and the ILP models using Gurobi 8.1.0 [45].

In this study, we employed 22 different test case networks ranging in size from 40 nodes to 150 nodes [40], [43], [50], [51]. Following some sources in the literature, we refer to the networks with 80 and 100 nodes as ''medium-sized''

and 120 and 150 nodes as ''large-scale'' networks [40]. The unit cost of transferring the traffic over the spans (C) for test cases 1 through 4 was calculated based on the Euclidean distance between the end nodes of every span and the span establishment cost (F) is a factor of C. However, in the network design process, the cost of span establishment and the cost of transferring one unit of traffic on a span might not have such a direct relationship. For example, if a portion of a distance between two nodes is located undersea, the span establishment cost between those two nodes may not be the exact factor of their Euclidean distance. Therefore, based on the various environmental and financial circumstances, the cost factors can change. As we wanted to perform a comprehensive analysis of the presented algorithm, we have considered the second set of test cases with more general cost factors. For the test cases 5 to 22, the cost factors (i.e., the span establishment cost and the unit cost of transferring the traffic over the spans), the number of the traffic demands, and the magnitude of every traffic has been selected randomly using uniform random selection process. The upper bound and lower bound values of the unit cost (C) were selected between 10 and 100. Similarly, the values of the span establishment cost (F) were selected between 100 and 1000. Moreover, the demand magnitudes (d^r) were selected randomly with a magnitude between 1 and 100. The details of the test case configurations such as number of nodes, number of possible spans, and number of traffic demands for every test case have been tabulated in Table 1.

B. VALIDATION OF THE RESULTS

We employed both the ILP model and the presented IGA for solving the network topology design and routing problem on various test cases. The IGA results such as runtimes and fitness values are obtained as the average of twenty sets of runs for each network, individually. The results obtained are tabulated in Table 2 where the normalized costs are calculated by dividing the IGA fitness (i.e., objective function) value by the objective function values of the ILP and relaxed ILP models, respectively. The normalized costs (IGA/ILP) represents the optimality of the IGA's solution, where the optimal (minimum) network design cost is obtained from the FCR problem's ILP model presented in Section I.C.

It can be seen from Table 2, as the number of nodes in a network increases, the ILP runtime becomes progressively increased such that by doubling the number of nodes of the test cases from 40 to 80, the average ILP runtime increases by almost 30 times. The ILP runtime for larger networks (e.g., test cases 14 to 22 with more than 100 nodes) is quite prohibitive. For instance, the ILP solver could not improve the found solution for test case 14 after more than 100 hours of running. However, for the same test case, the IGA can get to a reasonable optimality gap within a significantly shorter time (i.e., up to 40% better objective function value compared to ILP, within less than 5 hours). If a designer needs to solve the problem just once, then long runtimes might be of little consequence on a build that takes years. However, when network designers analyze various demand matrices and look at many multiple scenarios to determine which design will work best for every scenario, they can run the network design problem a great many times to determine the good configuration. As the design process of large-scale networks can take weeks or months for large scenarios, having a sub-optimal solution within reasonable processing time might be a good trade-off.

Based on the obtained results, it can be seen that the proposed methodology can outperform the ILP both in terms of runtime and objective function value, specifically for large-scale networks. The IGA's CPU process times vary between 0.1 hours and 10.7 hours. For networks with more than 80 nodes, the performance of IGA in terms of the processing time and the objective function value is 100% of the time more efficient than the ILP.

TABLE 1. Test case configurations.

Test Case	Number of nodes |N|	$\|A\|$	$\|D\|$
1	40	780	780
2	40	780	780
3	60	1770	1770
4	60	1770	1770
5	40	780	400
6	40	780	288
7	40	780	353
8	60	1770	600
9	60	1770	448
10	60	1770	548
11	80	3160	800
12	80	3160	814
13	80	3160	758
14	100	4950	1117
15	100	4950	1042
16	100	4950	953
17	120	7140	1266
18	120	7140	1152
19	120	7140	1151
20	150	11175	1170
21	150	11175	1160
22	150	11175	1168

For large networks, as the ILP has prohibitive runtime, we solved a relaxed version of the ILP as well in order to get a lower performance bound for the network design and routing cost. In the relaxed version of the ILP model (R-ILP), the integrality constraints on the decision variables are removed. In other words, the number of working traffic flows on every span that primarily needed to be an integer value in the original ILP model can take any real value in the relaxed ILP model. Although relaxing the integrality constraints, reduces the computational complexity of the

problem, the found solution does not necessarily represent the optimal feasible solution to the original problem. The optimal solution of the relaxed ILP model only represents a lower performance bound on the network design and routing cost. Meaning that in a minimization problem such as ours, the optimal solution to the original ILP model will not be smaller than the optimal solution provided by the relaxed ILP model.

Considering this trade-off between the computational complexity and the performance bound, we compared our results against the relaxation of the original ILP, as well. The relative objective function values that represent the ratio of the IGA objective function value to the relaxed ILP solution, have been reported in Table 2. Where we were able to solve the original ILP to an optimal solution, the gaps between the original ILP and the relaxed ILP were 19%, 17%, and 20% on average for 40-node, 60-node, and 80-node networks, respectively (average optimality gaps were calculated among all test cases for each size).

As a third benchmark, we compared our IGA against the CGA [44]. The crossover and mutation operators were implemented according to [44], and their rates were set to the suggested values in [44]; we refer the interested reader to reference [44] for more details. For every test case, we ran the CGA twenty times and reported the average runtimes and normalized costs in the 4th and 8th columns of Table 2.

Due to the randomness of the operators, the objective function values are comparable to or worse than our IGA, while the runtimes are up to four times longer. Specifically, for test cases 1 and 2 (40-node networks), the normalized costs (IGA/CGA) are 1; however, CGA runtimes are almost four times longer than IGA runtimes. Moreover, as the test case sizes increase (moving from 40-node networks toward 150-node networks), the general trend of normalized costs (IGA/CGA) is descending. This comparison shows that IGA outperforms the CGA for larger networks.

The comparison between IGA and CGA shows that the employed basic genetic operators such as a single-point crossover and a random flip mutation cannot improve the objective function value as efficiently as the problem-specific operators introduced in this paper. Specifically, as the network size grows, the complexity of the search space (e.g., the number of decision variables and constraints) increases drastically. Thus, the effectiveness of such basic operators would decrease. Moreover, the basic genetic operators can easily produce infeasible solutions. Therefore, a time-consuming repair operator should be implemented to correct the infeasible solutions. Thus, CGA runtimes are longer than the proposed IGA.

Fig. 19 to Fig. 26 in the appendix illustrate the performance of the proposed IGA for all of the test cases. The obtained solutions from the IPG-GA are assigned the required working traffic to accommodate all the demands. The completed network topologies with routed demands are then utilized as the initial population for the Master-GA. Fig. 19 to Fig. 26 represent the progress rate of the Master-GA as the complete solution to the network design and routing problem.

TABLE 2. Comparison between the proposed IGA and benchmarks (objective function value and processing times) for various network sizes. R-ILP shows the relaxed ILP and CGA shows customized GA based on [44].

Test case	ILP runtime (hours)	R-ILP runtime (hours)	CGA runtime (hours)	IGA runtime (hours)	Normalized Costs		
					IGA / ILP	IGA / R-ILP	IGA / CGA
1	1.0	0.01	2.3	0.6	1.13	1.24	1.00
2	0.4	0.01	2.3	0.5	1.09	1.16	1.00
3	12.1	0.1	20.6	5.9	1.16	1.24	0.98
4	24.2	0.1	19.5	5.5	1.09	1.14	0.99
5	0.9	0.003	0.4	0.2	1.22	1.78	1.03
6	0.1	0.003	0.3	0.1	1.22	1.40	0.97
7	0.1	0.003	0.5	0.1	1.24	1.43	0.98
8	4.0	0.01	1.4	0.7	1.20	1.68	0.98
9	1.4	0.01	1.2	0.4	1.24	1.43	0.97
10	2.0	0.01	1.7	0.5	1.25	1.44	0.97
11	15.9	0.03	3.7	1.6	1.15	1.49	0.97
12	17.9	0.04	5.3	1.7	1.25	1.44	0.97
13	12.2	0.03	4.6	1.4	1.40	1.62	0.97
14	>40[1]	0.1	10.1	4.3	0.72	1.43	0.95
15	>40	0.1	11.2	3.9	0.74	1.45	0.96
16	>40	0.1	9.2	3.3	0.72	1.43	0.95
17	>40	0.1	16.4	7.6	0.72	1.43	0.94
18	>40	0.1	16.4	6.5	0.73	1.46	0.97
19	>40	0.1	16.3	6.4	0.72	1.43	0.94
20	>40	0.3	20.7	10.5	0.73	1.45	0.95
21	>40	0.3	18.8	10.5	0.72	1.44	0.94
22	>40	0.3	21.6	10.7	0.73	1.43	0.94

The costs are normalized to the best cost achieved from the benchmark ILP model. Note that in the larger test cases, the first iteration of the IGA already outperforms the benchmark, though we continue iterating to achieve better results. In Fig. 19 through Fig. 26, the solid data points represent the average of the normalized network design costs over twenty sets of IGA runs. The error bars represent one standard deviation among the twenty sets of runs.

IV. CONCLUSION

In this paper, we have presented an Improved GA-based algorithm for the large-scale network design and routing problem. In the presented algorithm, the concept of TSP has been employed to create a set of initial feasible and bi-connected

[1]The ILP solver was not able to improve the found solution even after 100 hours of running.

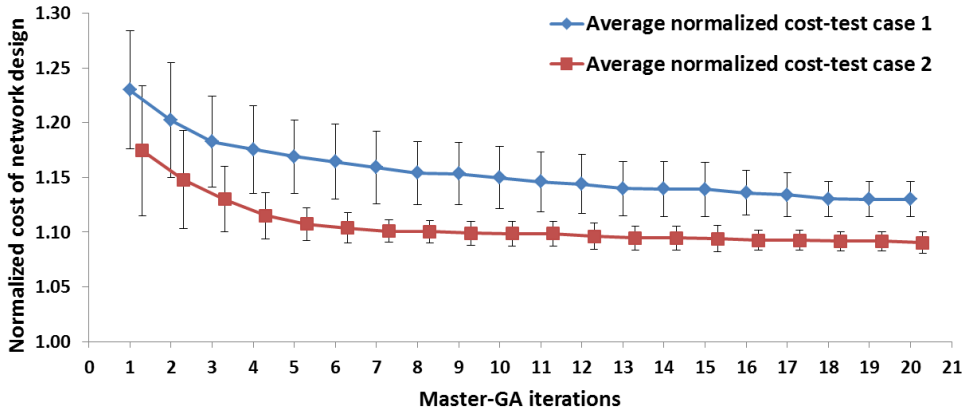

FIGURE 19. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations of Master-GA for test cases 1 and 2.

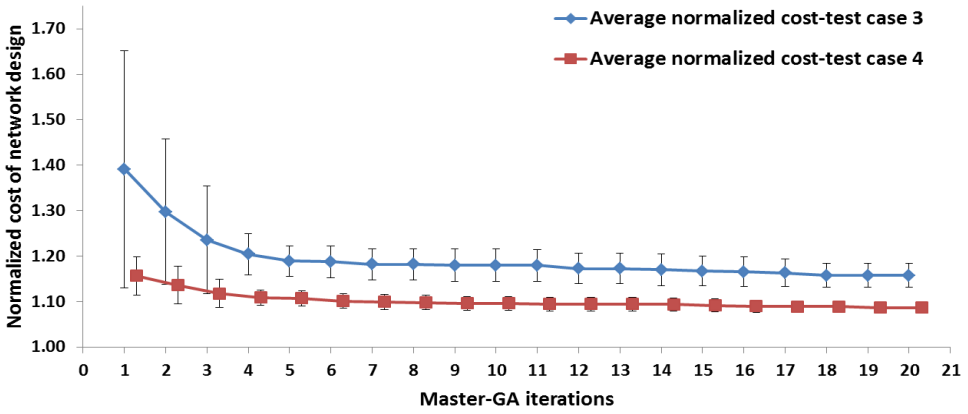

FIGURE 20. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations of Master-GA for test cases 3 and 4.

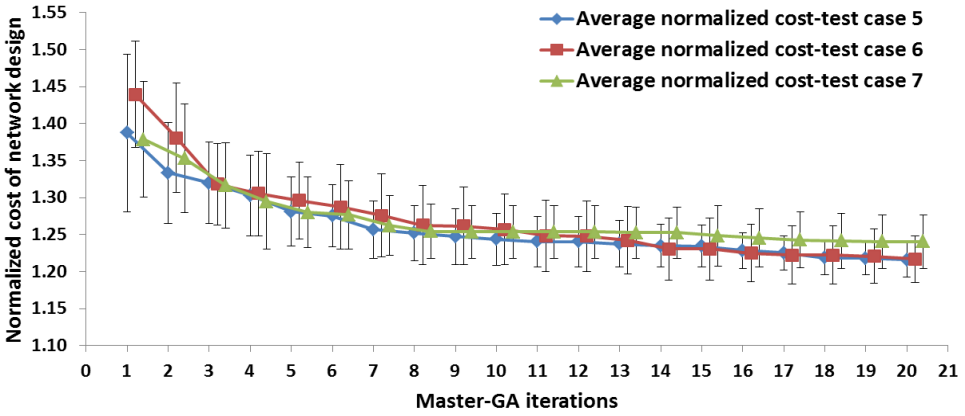

FIGURE 21. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations of Master-GA for test cases 5-7.

solutions (i.e., network topologies). The set of sub-optimal initial feasible solutions have been created using the IPG-GA algorithm. Then, a routing function routes all of the traffic demands on the network topologies that have been obtained from the last iteration of IPG-GA. The obtained population of network topologies is fed to the Master-GA for further topology and routing improvements in an iterative fashion. We have developed two sets of well-designed genetic

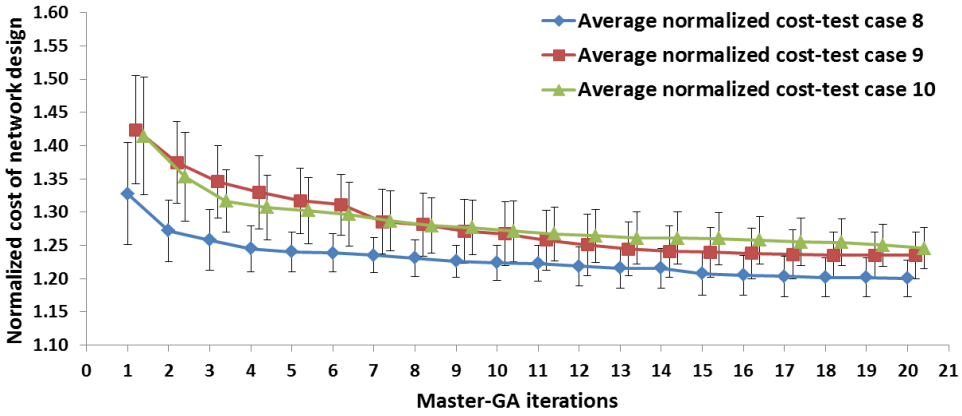

FIGURE 22. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations of Master-GA for test cases 8-10.

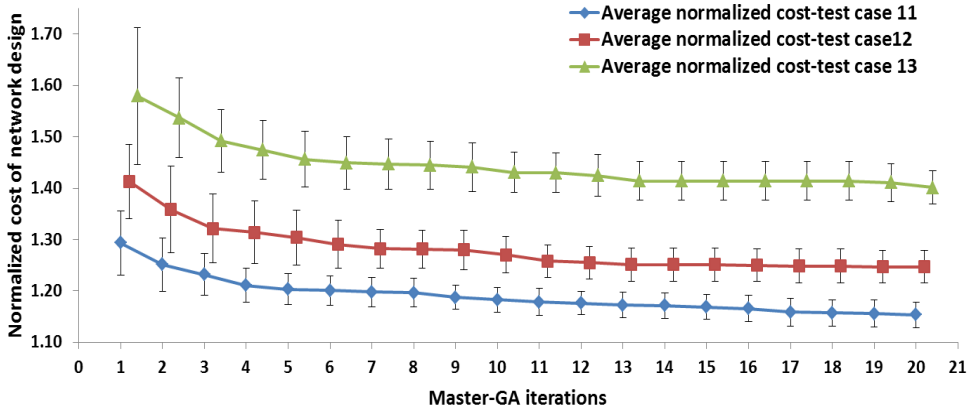

FIGURE 23. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations of Master-GA for test cases 11-13.

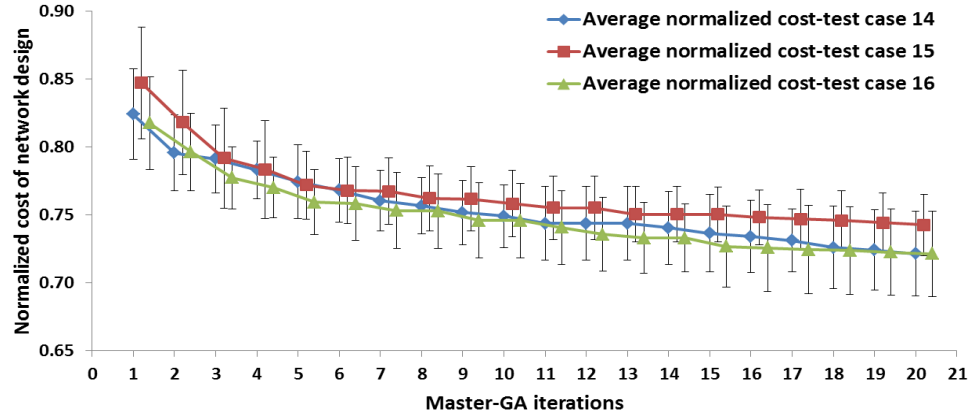

FIGURE 24. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations of Master-GA for test cases 14-16.

operators (crossovers and mutations) for breeding new solutions from the initial ones. These operators try to maintain the feasibility and diversity of the solutions.

One of the main contributions of this research work is that the proposed IGA starts with a set of feasible solutions. Thus, even in the very first iteration of the algorithm, we have

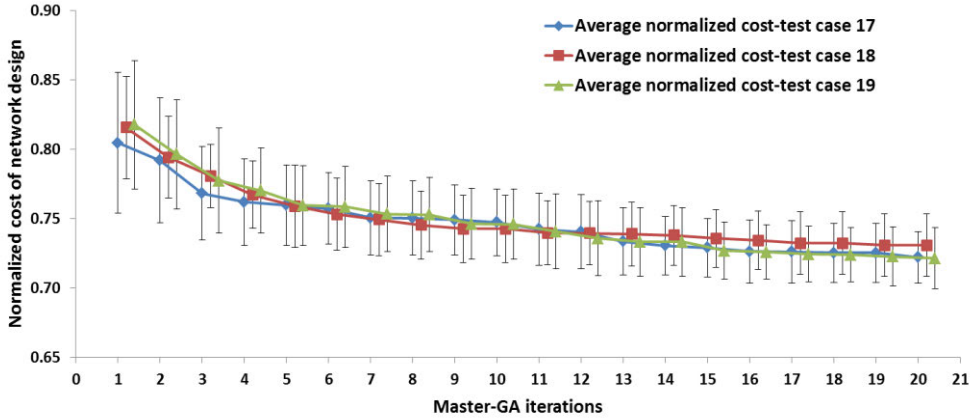

FIGURE 25. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations of Master-GA for test cases 17-19.

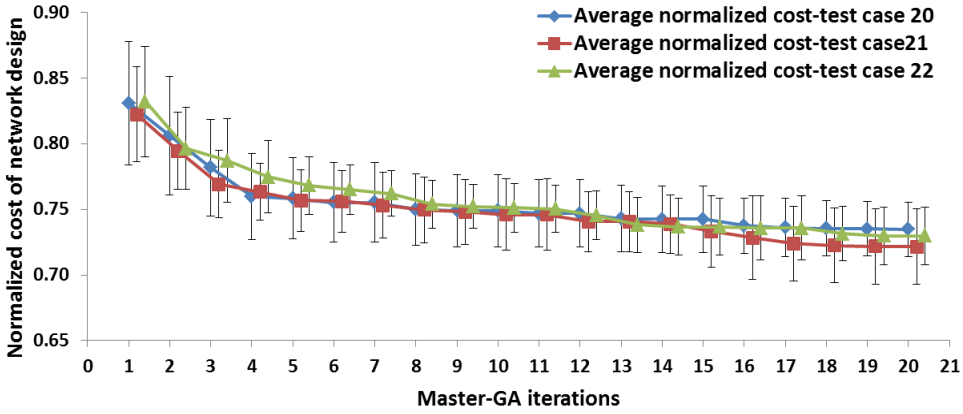

FIGURE 26. Performance of proposed IGA in terms of the normalized total cost of network design vs. the number of iterations of Master-GA for test cases 20-22.

network topologies that are bi-connected. The bi-connectivity of the initial solutions is the result of their cyclic structure. Using TSP, a set of Hamiltonian cycles has been generated for the initial population of the IPG-GA. Although the initial solutions were feasible for this problem, they were not optimal. Therefore, using the IPG-GA, the set of initial feasible solutions was improved iteration by iteration to get a set of optimal cycles that can be used as sub-optimal network topologies for the Master-GA. As the Master-GA starts with a set of initial feasible solutions, the convergence rate of the algorithm will be more than the case the initial population is generated randomly. Moreover, as the initial feasible solution in this research is bi-connected, the constraints for employing the restoration mechanisms can be incorporated in the algorithm too.

We presented well-designed crossover and mutation operators that try to keep the solutions feasible and decrease the total network design cost while increasing the diversity of the populations. Moreover, we conducted a comprehensive set of experiments to tune the genetic operators' rates. The higher the genetic operators' rates, the higher the diversity of the population and the higher the runtime. Thus, to reach a trade-off between the population diversity and runtime, a suitable rate for crossover and mutation rates in both the IPG-GA and Master-GA was obtained.

The proposed algorithm resulted in higher fitness levels and runtime for all test cases with 100 nodes and more. Moreover, for 9 test cases of moderate size, there have been optimality gaps of almost 18%, with runtime improvements as high as 75% compared to the ILP.

For larger test cases where the ILP solution encountered prohibitive runtimes, in order to provide a better analysis of the IGA's performance, we performed a relaxation of the original ILP (R-ILP) to obtain a lower bound on the network design and routing cost. We then compared the IGA results against the obtained solutions from R-ILP.

APPENDIX

See Figures 19–26.

REFERENCES

[1] W. D. Grover and J. Doucette, "Topological design of survivable mesh-based transport networks," *Ann. Oper. Res.*, vol. 106, nos. 1–4, pp. 79–125, 2001.

[2] B. Gendron, T. G. Crainic, and A. Frangioni, "Multicommodity capacitated network design," in *Telecommunications Network Planning* (Centre for Research on Transportation), B. Sansò and P. Soriano, Eds. Boston, MA, USA: Springer, 1999, pp. 1–19.

[3] B. Thiongane, J. Cordeau, and B. Gendron, "Formulations for the nonbifurcated hop-constrained multicommodity capacitated fixed-charge network design problem," *Comput. Oper. Res.*, vol. 53, pp. 1–8, Jan. 2015.

[4] L.-M. Munguía, S. Ahmed, D. A. Bader, G. L. Nemhauser, V. Goel, and Y. Shao, "A parallel local search framework for the fixed-charge multicommodity network flow problem," *Comput. Oper. Res.*, vol. 77, pp. 44–57, Jan. 2017.

[5] W. D. Grover, *Mesh-Based Survivable Networks, Options and Strategies for Optical, MPLS, SONET, and ATM Networking*, B. Goodwin, Ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2004.

[6] D. A. Schupke and R. G. Prinz, "Capacity efficiency and restorability of path protection and rerouting in WDM networks subject to dual failures," *Photon. Netw. Commun.*, vol. 8, no. 2, pp. 191–207, Sep. 2004.

[7] D. A. Schupke, W. D. Grover, and M. Clouqueur, "Strategies for enhanced dual failure restorability with static or reconfigurable P-cycle networks," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2004, pp. 1628–1633.

[8] A. Jaekel, A. Bari, Q. Rahman, Y. Chen, S. Bandyopadhyay, and Y. Aneja, "Resource efficient network design and traffic grooming strategy with guaranteed survivability," *Opt. Switching Netw.*, vol. 9, no. 4, pp. 271–285, Nov. 2012.

[9] A. Fumagalli, I. Cerutti, and M. Tacca, "Optimal design of survivable mesh networks based on line switched WDM self-healing rings," *IEEE/ACM Trans. Netw.*, vol. 11, no. 3, pp. 501–512, Jun. 2003.

[10] Y. Liu and D. Tipper, "Spare capacity allocation using shared backup path protection for dual link failures," *Comput. Commun.*, vol. 36, no. 6, pp. 666–677, Sep. 2012.

[11] X. Dong, T. El-gorashi, and J. M. H. Elmirghani, "On the energy efficiency of physical topology design for IP over WDM networks," *IEEE/OSA J. Lightw. Technol.*, vol. 30, no. 12, pp. 1931–1942, Feb. 2012.

[12] N. Bahria El Asghar, A. Ben Fradj, M. Frikha, and S. Tabbane, "A heuristic approach for solving the virtual topology design problem in next generation networks," in *Proc. 14th Int. Symp. Wireless Pers. Multimedia Commun., Commun., Netw. Appl. Internet Things*, 2011, pp. 1–5.

[13] W. D. Grover and J. Doucette, "A novel heuristic for topology planning and evolution of optical mesh networks," in *Proc. IEEE Global Telecommun. Conf.*, Nov. 2001, pp. 2169–2173.

[14] L. W. Clarke and G. Anandalingam, "A bootstrap heuristic for designing minimum cost survivable networks," *Comput. Oper. Res.*, vol. 22, no. 9, pp. 921–934, Nov. 1995.

[15] B. Jaumard and H. A. Hoang, "Design and dimensioning of logical survivable topologies against multiple failures," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 5, no. 1, pp. 23–36, Jan. 2013.

[16] B. Gendron, S. Hanafi, and R. Todosijević, "An efficient matheuristic for the multicommodity fixed-charge network design problem," *IFAC-PapersOnLine*, vol. 49, no. 12, pp. 117–120, 2016.

[17] H. Üster and S. K. S. Kumar, "Algorithms for the design of network topologies with balanced disjoint rings," *J. Heuristics*, vol. 16, no. 1, pp. 37–63, Feb. 2010.

[18] D. Wang and J. Y. McNair, "Topological optimization for spare-sharing-based wavelength-routed all-optical networks," *Opt. Switching Netw.*, vol. 9, no. 4, pp. 297–313, Nov. 2012.

[19] Y. Liu, D. Tipper, and P. Siripongwutikorn, "Approximating optimal spare capacity allocation by successive survivable routing," *IEEE/ACM Trans. Netw.*, vol. 13, no. 1, pp. 198–211, Feb. 2005.

[20] T. Mikoshi, T. Takenaka, R. Sugiyama, A. Masuda, and K. Shiomoto, "Multi-layer network topology design for large-scale network," in *Proc. 23rd Int. Teletraffic Congr.*, 2011, pp. 306–307.

[21] D. Pereira and M. C. Penna, "A new algorithm for dimensioning resilient optical networks for shared-mesh protection against multiple link failures," *Opt. Switching Netw.*, vol. 13, pp. 158–172, Jul. 2014.

[22] J. Sun, Q. Zhang, and J. Li, "Two-level evolutionary approach to the survivable mesh-based transport network topological design," *J. Heuristics*, vol. 16, no. 5, pp. 723–744, Oct. 2010.

[23] K. Inoue, S. Arakawa, and M. Murata, "A biological approach to physical topology design for plasticity in optical networks," *Opt. Switching Netw.*, vol. 25, pp. 124–132, Jul. 2017.

[24] Y. Xin, M. Shayman, R. J. La, and S. I. Marcus, "Reconfiguration of survivable IP over WDM networks," *Opt. Switching Netw.*, vol. 21, pp. 93–100, Jul. 2016.

[25] D.-R. Din and Y.-S. Chiu, "A genetic algorithm for solving virtual topology reconfiguration problem in survivable WDM networks with reconfiguration constraint," *Comput. Commun.*, vol. 31, no. 10, pp. 2520–2533, Jun. 2008.

[26] Y. Aneja, S. Bandyopadhyay, and A. Jaekel, "On routing in large WDM networks," *Opt. Switching Netw.*, vol. 3, nos. 3–4, pp. 219–232, Dec. 2006.

[27] S. K. Goudos, M. Deruyck, D. Plets, L. Martens, K. E. Psannis, P. Sarigiannidis, and W. Joseph, "A novel design approach for 5G massive MIMO and NB-IoT green networks using a hybrid jaya-differential evolution algorithm," *IEEE Access*, vol. 7, pp. 105687–105700, 2019.

[28] T. G. Crainic, Y. Li, and M. Toulouse, "A first multilevel cooperative algorithm for capacitated multicommodity network design," *Comput. Oper. Res.*, vol. 33, no. 9, pp. 2602–2622, Sep. 2006.

[29] A. Olivera, F. R. Amoza, and C. E. Testuri, "A GRASP algorithm for a capacitated, fixed charge, multicommodity network flow problem with uncertain demand and survivability constraints," *Int. Trans. Oper. Res.*, vol. 17, no. 6, pp. 765–776, Nov. 2010.

[30] B. Gendron, S. Hanafi, and R. Todosijević, "Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design," *Eur. J. Oper. Res.*, vol. 268, no. 1, pp. 70–81, Jul. 2018.

[31] K. Guo, M. Yang, and H. Zhu, "Application research of improved genetic algorithm based on machine learning in production scheduling," *Neural Comput. Appl.*, vol. 32, no. 7, pp. 1857–1868, Apr. 2020.

[32] E. Sulis, P. Terna, A. Di Leva, G. Boella, and A. Boccuzzi, "Agent-oriented decision support system for business processes management with genetic algorithm optimization: An application in healthcare," *J. Med. Syst.*, vol. 44, no. 9, p. 157, Sep. 2020.

[33] S. K. Roy and D. De, "Genetic algorithm based internet of precision agricultural things (IopaT) for agriculture 4.0," *Internet Things*, vol. 4, Mar. 2020, Art. no. 100201.

[34] S. Doostie, A. K. Hoshiar, M. Nazarahari, S. Lee, and H. Choi, "Optimal path planning of multiple nanoparticles in continuous environment using a novel adaptive genetic algorithm," *Precis. Eng.*, vol. 53, pp. 65–78, Jul. 2018.

[35] M. Nazarahari, E. Khanmirza, and S. Doostie, "Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm," *Expert Syst. Appl.*, vol. 115, pp. 106–120, Jan. 2019.

[36] C.-S. Wang and C.-T. Chang, "Integrated genetic algorithm and goal programming for network topology design problem with multiple objectives and multiple criteria," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 680–690, Jun. 2008.

[37] H. Chou, G. Premkumar, and C.-H. Chu, "Genetic algorithms for communications network design–an empirical study of the factors that influence performance," *IEEE Trans. Evol. Comput.*, vol. 5, no. 3, pp. 236–249, Jun. 2001.

[38] X.-M. Huang, Y.-J. Gong, J.-J. Li, and X.-M. Hu, "A novel genetic algorithm based on partitioning for large-scale network design problems," in *Proc. Companion Annu. Conf. Genetic Evol. Comput.*, Jul. 2014, pp. 111–112.

[39] P. Kumar, "Optimization of optical network using genetic algorithm," *Int. J. Photon. Opt. Technol.*, vol. 2, no. 1, pp. 5–8, 2016.

[40] C.-H. Chu, G. Premkumar, and H. Chou, "Digital data networks design using genetic algorithms," *Eur. J. Oper. Res.*, vol. 127, no. 1, pp. 140–158, 2000.

[41] H. T. T. Binh and S. H. Ngo, "Survivable flows routing in large scale network design using genetic algorithm," in *Advances in Computer Science and its Applications* (Lecture Notes in Electrical Engineering), vol. 279, H. Jeong, M. Obaidat, N. Yen, and J. Park, Eds. Berlin, Germany: Springer, 2014, pp. 345–351.

[42] S. Kwong, T. M. Chan, K. F. Man, and H. W. Chong, "The use of multiple objective genetic algorithm in self-healing network," *Appl. Soft Comput.*, vol. 2, no. 2, pp. 104–128, Dec. 2002.

[43] D. P. Onguetou and W. D. Grover, "Solution of a 200-node P-cycle network design problem with GA-based pre-selection of candidate structures," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2009, pp. 1–5.

[44] Y. Xin, G. N. Rouskas, and H. G. Perros, "On the physical and logical topology design of large-scale optical networks," *J. Lightw. Technol.*, vol. 21, no. 4, pp. 904–915, Apr. 2003.

[45] (2021). *Gurobi Optimizer Reference Manual*. [Online]. Available: https://www.Gurobi.Com
[46] M. M. Flood, "The traveling-salesman problem," *Oper. Res.*, vol. 4, no. 1, pp. 1–137, 1956.
[47] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter, "Hamilton paths in grid graphs," *SIAM J. Comput.*, vol. 11, no. 4, pp. 676–686, 1982.
[48] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[49] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA, USA: Create Space, 2009.
[50] W. Wang and J. Doucette, "Availability optimization in shared-backup path protected networks," *J. Opt. Commun. Netw.*, vol. 10, no. 5, pp. 451–460, 2018.
[51] Y.-J. Cheng, B.-T. Chen, and Y.-Y. Lee, "Design and performance evaluation of very large-scale optical frame switching networks based on WSS for future data centers," in *Proc. 20th Asia–Pacific Netw. Operations Manage. Symp. (APNOMS)*, Sep. 2019, pp. 1–4.

SAMIRA DOOSTIE (Graduate Student Member, IEEE) received the B.Sc. degree in mechanical engineering from IUST, Tehran, Iran, in 2013, and the M.Sc. degree in mechanical engineering from Islamic Azad University, Tehran, in 2015. She has been a Graduate Research Assistant with the Department of Mechanical Engineering, University of Alberta, since 2017. Her research interests include large-scale network design and optimization with a special focus on network survivability and availability.

TETSUHEI NAKASHIMA-PANIAGUA received the B.Sc. degree in computer engineering and the M.Sc. degree in computer science from ITESM at Guadalajara, Mexico, and the Ph.D. degree in engineering management from the University of Alberta, Edmonton, AB, Canada. He is currently an Associate Teaching Professor with the Mechanical Engineering Department, University of Alberta. For the last 23 years, he has combined his academic work with industry experience by working for several companies, assisting with product and business development, technology transfer, and providing consulting services for various organizations.

JOHN DOUCETTE received the B.Sc. degree in mathematics from Dalhousie University, Halifax, NS, Canada, the B.Eng. degree in industrial engineering from the Technical University of Nova Scotia (TUNS) (now, Dalhousie University), and the Ph.D. degree in electrical and computer engineering from the University of Alberta, Edmonton, AB, Canada. He joined the Department of Mechanical Engineering, University of Alberta, in 2005, and previously held a Research Scientist position and a Research Engineer position at TRLabs, from 2000 to 2005. He is currently a Professor and the Chair of the Department of Mechanical Engineering, University of Alberta, where he carries out research in network restoration and protection, network reliability and availability, network design and optimization, and various applications of operations research and optimization. He has authored or coauthored over 80 peer-reviewed journal and conference publications and approximately 50 technical reports and other presentations, and a co-inventor of four patents.