Ensuring Resilience in Optical WDM Networks With Nature-Inspired Heuristics

Fatma Corut Ergin, Elif Kaldırım, Ayşegül Yayımlı, and A. Şima Uyar

***Abstract*—One of the most important issues in optical network design is ensuring its resilience. In this paper, we propose using nature-inspired heuristics to find a resilient mapping of a given virtual topology with minimum resource usage. Evolutionary algorithms and ant colony optimization algorithms are applied to the problem after a set of parameter tuning tests. To assess the performance of the proposed algorithms, we compare the experimental results with those obtained through integer linear programming. The results show that both of our algorithms can solve the problem even for large-scale network topologies for which a feasible solution cannot be found using integer linear programming. Moreover, the CPU time and the memory used by the nature-inspired heuristics is much lower. The solution quality and the CPU time usage results prove that both of our nature-inspired heuristics can easily be applied to real-world applications.**

***Index Terms*—Optical networks; Survivability; Evolutionary algorithms; Ant colony optimization**.

I. Introduction

Today, optical networking [1] is the most effective technology to meet the high bandwidth network demand. The high capacity of fiber used in optical networks can be divided into hundreds of different transmission channels, using wavelength division multiplexing (WDM) technology. Each of these channels works on different wavelengths and each channel can be associated with a different optical connection. The upper layers (IP, Ethernet, etc.) can transmit data using these optical connections.

End-to-end optical connections used by the packet layer (IP, Ethernet, etc.) are called lightpaths. Since the fibers on the physical topology allow traffic flow on different wavelengths, more than one lightpath, each operating on a different wavelength, can be routed on a single fiber. All the lightpaths set up in a network form the virtual topology (VT). Given the physical parameters of the network (physical topology, optical transceivers on the nodes, wavelength numbers on the fibers, etc.) and the mean traffic rates between nodes, the problem of designing the lightpaths to be set up on the physical topology is known as the VT design. VT mapping, which is a subproblem of VT design, is to find a proper route for each lightpath of the given VT and to assign wavelengths to these lightpaths.

Any damage to a physical link (fiber) on the network causes all the lightpaths routed through this link to be broken. Since huge amounts of data (40 Gb/s) can be transmitted over each of these lightpaths, fiber damage may result in a serious amount of data loss. Two different approaches can be used to avoid this data loss [1]: 1. survivable design of the physical topology and 2. survivable design of the virtual topology. The first approach is the problem of designing a backup link/path for each link/path of the VT. The second approach is the problem of designing a resilient[1] VT, i.e., the VT remains connected in the event of single or multiple link failures. While the first approach provides faster recovery for time-critical applications (IP phone, telemedicine) by reserving more resources; the second approach aims to protect data communication using less resources. In this study, our main aim is to design efficient nature-inspired heuristics (NIH) to find a survivable mapping of a given VT while minimizing the resource usage.

To illustrate the VT mapping problem, assume that we have a physical network topology as in Fig. 1(a) and a virtual network topology to be routed on this physical topology as in Fig. 1(b). If we route this VT as in Fig. 1(c), we obtain a survivable mapping, that is, a single failure on any physical link does not disconnect the VT. However, if the routing of only one lightpath is

Manuscript received March 15, 2010; revised June 22, 2010; accepted June 26, 2010; published July 30, 2010 (Doc. ID 125367).

Fatma Corut Ergin (e-mail: fatma.ergin@marmara.edu.tr), Ayşegül Yayımlı, and A. Şima Uyar are and Elif Kaldırım was with the Department of Computer Engineering, Istanbul Technical University, Istanbul, Turkey.

Digital Object Identifier 10.1364/JOCN.2.000642

[1]In this study, we will use survivability and resilience interchangeably.

1943-0620/10/080642-11/$15.00

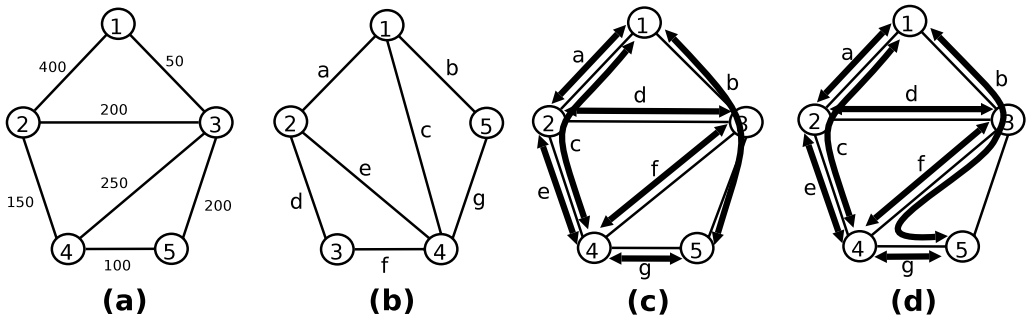

Fig. 1. (a) Physical topology, (b) virtual topology, (c) survivable mapping, (d) unsurvivable mapping.

changed, e.g., as in Fig. 1(d), we end up with an unsurvivable mapping. In this case, if a failure occurs on the physical link between nodes 4 and 5, the nodes connected with lightpaths b and g will not be able to communicate and nodes will be disconnected from the network.

The VT mapping problem is known to be NP-complete [2]. Because of its complexity, for real-life sized networks it is not possible to solve the problem optimally in an acceptable amount of time using classical optimization techniques. Therefore, heuristic approaches are preferred. In this study, we propose using NIHs for the survivable VT mapping problem and demonstrate their efficiency. We chose evolutionary algorithms (EAs) due to their successful applications on NP-complete problems and ant colony optimization (ACO) algorithms due to their successful performance on constrained combinatorial optimization problems. For both the EA and the ACO algorithms, we first performed a series of parameter tuning tests. Then, for the performance comparison experiments, we used the best settings we determined. Also, we compared their performance to the results of the basic integer linear programming (ILP) formulation and to an ILP relaxation proposed in [2]. As a result of the experiments, we showed that both ACO and EAs can solve the survivable VT mapping problem with 100% success and in a very short time.

The rest of the paper is organized as follows. The definition of the problem, together with its mathematical formulation, is given in Section II, followed by the related literature. In Section III, EAs and ACO are explained briefly, and the details of our ACO and EA designs for the problem are given in Section IV. Then, in Section V the experimental results are given and discussed thoroughly.

II. Problem Definition and Related Literature

Given the physical and the virtual topologies, our aim is to find a survivable mapping of the VT.

A. Formal Problem Definition

The physical topology is composed of a set of nodes N and a set of edges E, where (i,j) is in E if there is a link between nodes i and j. Each link has a capacity of W wavelengths. The VT, on the other hand, has a set of virtual nodes N_L, which is a subset of N, and virtual edges (lightpaths) E_L, where an edge (s,t) exists in E_L if both node s and node t are in N_L and there is a lightpath between them.

An ILP formulation of survivable lightpath routing of a VT on top of a given physical topology is given in [2]. Based on this, our objective is to minimize the total number of wavelength links used in the physical topology since it gives a better idea of the actual resource usage. A wavelength link is defined as a wavelength used on a physical link.

The aim of lightpath routing is to find a set of physical links that connect the nodes of the lightpaths. Let $f_{ij}^{st}=1$ if virtual link (s,t) is routed on physical link (i,j) and 0 otherwise. Since our objective is to minimize the total number of wavelength links used in the whole physical topology, we can formulate the objective as in Eq. (1):

$$\text{Minimize} \sum_{(i,j)\epsilon E,(s,t)\epsilon E_L} f_{ij}^{st}. \tag{1}$$

B. Related Literature

The survivable VT mapping problem was first addressed as design protection [3] in the literature. In this first study, tabu search was used to find the minimum number of source–destination pairs that become disconnected in the event of a physical link failure. Nucci *et al.* also used tabu search to solve the survivable VT design problem [4]. The constraints in this study include transmitter and receiver constraints as well as wavelength capacity constraints.

In other heuristic approaches to VT mapping, Ducatelle and Gambardella [5] studied the problem using a local search algorithm, while Kurant and Thiran [6] used an algorithm that divides the survivable map-

ping problem into subproblems. Although, there are a few studies on VT mapping [7] and design [8] using EAs, none of them considered survivability, except [9]. The only EA-based approach for the survivable VT mapping problem is proposed in [9]. The objective is to minimize the resource usage without violating the survivability and the capacity constraints.

Swarm intelligence algorithms are used in a few studies for the routing and wavelength assignment (RWA) problem. ACO is applied to the static [10] and dynamic [11,12] RWA problem without the survivability constraint. In [13], ACO is used considering backup paths on the physical layer. Particle swarm optimization is applied to the RWA problem in only [14], in which the survivability constraint is not considered. The only study on survivable VT mapping that uses ACO is [15].

Modiano and Narula-Tam used ILP to solve the VT mapping problem [2]. They added the survivability constraint in the problem formulation such that no physical link is shared by all virtual links belonging to a cut set of the VT graph. Their objective was to minimize the number of wavelengths used. For the cases when ILP cannot find an optimum solution in a reasonable amount of time due to the problem size, Modiano and Narula-Tam proposed two relaxations to ILP, which consider only small-sized cut sets. These relaxations reduce the problem size; however, they may lead to unsurvivable solutions. Todimala and Ramamurthy proposed an improved ILP formulation and they solved the problem for networks of up to 24 nodes [16]. In [16], besides the physical network and the virtual network topologies, the shared risk link groups should be known in advance.

III. Nature-Inspired Heuristics

NIHs are inspired from some mechanisms and processes in nature. Most NIHs start searching from several points in the search space, thus lowering the probability of getting stuck at local optima.

In combinatorial optimization problems, a solution consists of a discrete set of subparts, which are called solution components. One method to classify search methods [17] relies on the underlying mechanism used to generate candidate solutions for a problem from the solution components: 1) In perturbative search algorithms, a solution candidate can be transformed into another one by altering one or more of the components. 2) In constructive search algorithms, a complete solution candidate is constructed by iteratively adding solution components to the partial solution candidates.

Here, from the perturbative methods, we chose EAs, due to their successful applications on NP-complete problems. ACO is chosen from the constructive search methods, due to its successful performance on constrained combinatorial optimization problems.

A. Evolutionary Algorithms

EAs are population-based stochastic search methods that have been applied successfully in many search, optimization, and machine learning problems [18]. EAs iteratively operate on a population of individuals (or chromosomes) that encode the possible solutions. EAs can be used to search for an individual yielding the optimum numerical value of an evaluation function, called the fitness function. There are many variations of EAs presented in the literature, and the EA term provides a common basis for all of its variants. In this study, we used a steady-state EA (SSEA), which is explained in the folowing paragraphs.

In SSEAs, the initial population is generated randomly or through some heuristics. Then, in each iteration a new individual is created and inserted into the population until the termination criteria are met. Commonly three genetic operators are used to generate new individuals: selection, recombination, and mutation. Individuals are selected as parents to produce offspring, based on their fitness values. The selection mechanism is usually probabilistic, where high-quality solutions are more likely to become parents than low-quality ones. There are various methods [18] for selecting the parents, such as, roulette-wheel selection, stochastic universal sampling, and tournament selection. The recombination operator, also known as crossover, is applied on the selected individuals with a predefined crossover probability. This operator takes two individuals and exchanges some genes between them to generate the offspring. The main idea in crossover is to combine partial solutions of parents to form new solutions. There are many forms of crossover operators, such as 1-point crossover, n-point crossover, or uniform crossover [18]. After crossover, the generated offspring is mutated with a predefined mutation probability. Mutation alters the value of the gene to which it is applied.

Finally, the offspring is inserted into the population replacing one of the existing individuals based on some criteria, such as replacing the worst, replacing the oldest, replacing the most similar, etc. This cycle is performed until the termination criteria are reached, which may be defined as achieving a solution with a sufficient quality, reaching a predefined maximum number of generations or fitness evaluations, reaching convergence, etc.

B. Ant Colony Optimization Algorithms

ACO algorithms are inspired from the social behavior of ants. ACO has been applied successfully to many

combinatorial optimization problems. One of the first successful implementations of ACO is the Ant System (AS) developed by Dorigo and Stutzle [19] in 1992.

In ACO, an iteration consists of the solution construction and pheromone update stages. An ACO run is finished when stopping criteria are met, which may be when either a predefined number of solution candidates are generated or when the allowed time is completed. In each iteration, each ant in the colony constructs a complete solution. Ants start from random nodes and move on the construction graph by visiting neighboring nodes at each step. For each node, the next node to visit is determined through a stochastic local decision policy based on the current pheromone levels and heuristic information between the current node and its neighbors. Better solutions have higher heuristic levels. Pheromone trails are modified when all ants have constructed a solution. First, the pheromone values are lowered (evaporated) by a constant factor on all edges. Then pheromone values are increased on the edges the ants have visited during their solution construction. Evaporation prevents adding unlimited pheromone trails so that ants can forget bad decisions they had made previously. Ants that construct better solutions deposit more pheromones on the edges they traversed, so that the edges that lead to better solutions and are used by many ants receive more pheromones. Details of ACO can be found in [19].

The AS algorithm [20] implements the basic ACO procedure detailed above in this section. AS has been the basis for many ACO variants that have become the state of the art for many applications. These variants include elitist AS, rank-based AS (RAS), MAX-MIN AS (MMAS), ant colony system (ACS), best-worst AS (BWAS), the approximate nondeterministic tree search, and the hyper-cube framework. AS, ACS, elitist AS, RAS, MMAS, and BWAS can be considered as direct variants of AS since they all use the basic AS framework. The main differences between AS and these variants are the pheromone update procedures and some additional details in the management of the pheromone trails. In this study, we implemented AS, ACS, elitist AS, RAS, MMAS, and BWAS for the VT mapping problem [15], since it can be seen in [19] that these direct variants of AS have been successfully applied to similar problems in the literature [21].

IV. Proposed Nature-Inspired Heuristics

Designing a solution representation that is well suited to the problem is crucial in EA and ACO performance. The survivable VT mapping problem can be seen as a search for the best routing of lightpaths through physical links. Therefore, we use a solution encoding inspired from [7] for both heuristics. For this encoding, first, the k-shortest paths for each lightpath are determined. Then, a solution candidate is represented as an integer string of length l, where l is the number of lightpaths in the VT. Each location on the solution string gives the index of the shortest path for the corresponding lightpath, which can take on values between $[1 \ldots k]$, where k is the predefined number of shortest paths for each lightpath.

Our objective is to minimize the total cost of resources used throughout the network. This cost is evaluated in two different ways:

- by considering the actual lengths of the physical links (link cost),
- by counting the number of physical links used (hop count).

In the EA, constraint violations are considered as penalties in the fitness evaluation stage. In ACO, constraints are taken into consideration during solution construction, i.e., no constraint violations are possible.

A. Proposed Evolutionary Algorithm

We designed a steady-state EA with duplicate elimination, where a new individual is generated and inserted into the population at each iteration. After a random initial population generation, the EA operators are applied to the solutions until a predefined number of fitness evaluations are executed.

Mating pairs are selected through binary tournament selection. The two selected individuals undergo reproduction. Reproduction consists of uniform crossover and problem-specific mutation operators. In uniform crossover, the offspring takes the genes from either parent with equal probability.

We define three different mutation operators. The first one is a simple random-reset mutation, called gene mutation (*gene*). In this type of mutation, the value of a gene is randomly reset to another value within the allowed range, i.e., between 1 and k.

The second and the third are problem-specific mutation operators, called least similar path mutation (*ls_pm*) and most similar path mutation (*ms_pm*). These mutation operators consider the physical link similarities between the shortest paths of each lightpath, where similarity is defined as the number of common physical links. In *ls_pm*, if mutation occurs on a gene, its current value is replaced by the index of the least-similar shortest path for the corresponding lightpath. Similarly, in *ms_pm*, the current value of the gene is replaced by the index of the most similar shortest path for the corresponding lightpath.

Violations of the constraints for the problem, i.e., the survivability and the capacity constraints, are included as penalties in the fitness function. To determine whether the solution is survivable, each physical

link is deleted from the physical network one by one. If the VT becomes disconnected in the event of a broken physical link, the solution is taken as unsurvivable. The connectivity constraint is ensured within the algorithm; therefore, no explicit verification is needed. The penalty for an unsurvivable solution is determined in three different ways:

1) the total number of physical links whose failure results in the network unsurvivability (us_1),
2) the sum of the total number of lightpaths that become disconnected in the event of each physical link failure [3,5] (us_2),
3) the maximum of the total number of lightpaths that become disconnected in the event of each physical link failure [3] (us_3).

For each of the fitness evaluation methods we define, f_1, f_2, and f_3, we use the penalty calculation methods, (us_1), (us_2), and (us_3), respectively. In the first fitness evaluation method (f_1), the connectivity of the graph is checked for the failure of each physical link, which has an algorithmic complexity of $O(e \cdot n^3)$. On the other hand, for the second and third fitness evaluation methods (f_2 and f_3), a shortest-path algorithm is applied for the failure of each physical link, which means an algorithmic complexity of $O(l \cdot (e+n) \cdot \log n)$. Here, e is the number of physical links, n is the number of nodes, and l is the number of lightpaths.

A capacity constraint violation adds a penalty value that is proportional to the total number of physical links that are assigned more lightpaths than the predetermined wavelength capacity (wavelength capacity violation). Both penalties, i.e., the survivability and the capacity penalty, are multiplied with a penalty factor and added to the fitness of the solution. As a result, the fitness values are calculated as

$$\begin{aligned} f_i = {} & \text{total number of wavelength links in network} \\ & + \text{penalty factor}_1 \times \text{wavelength capcity violation} \\ & + \text{penalty factor}_2 \times (us_i). \end{aligned}$$

At the final stage of the loop, the fitness value of the offspring is calculated according to one of the fitness evaluation methods given above and compared with the worst individual in the current population. If the offspring has better fitness, it replaces the worst individual; otherwise it is discarded.

B. Proposed Ant Colony Optimization Algorithm

In ACO, the physical topology is used as a construction graph on which ants travel and construct their solutions. Ants simultaneously try to route lightpaths on the graph one by one in a random order. The flexibility of selecting lightpaths in a random order may increase the number of feasible solutions because, if a lightpath can only be routed using a few shortest paths, routing this lightpath earlier may result in better solutions.

The shortest paths of the lightpaths are provided to the ants at the beginning of the algorithm. Ants determine one of the shortest paths of the selected lightpath while visiting the nodes of the shortest paths on the construction graph. They check whether the chosen shortest path violates the constraints, i.e., the capacity and the survivability constraints. If a solution becomes infeasible for a shortest path selected for the lightpath, another shortest path is examined. If none of the shortest paths leads to a feasible solution, the ant is removed from the colony.

Ants decide their move on the construction graph based on the heuristic and pheromone information. Pheromone trails are modeled as a 2D matrix that accumulates the information learned by the ants. The lightpath pheromone matrix has lightpaths in its columns and rows and gives information about which lightpath is more valuable to choose after the current lightpath, whereas the shortest-path pheromone matrix has lightpaths in its rows and corresponding shortest paths in its columns and gives information about which shortest path leads to a better result when selected for the current lightpath. These pheromones are initialized in the beginning of the algorithm with the same value for each possible choice of the ant. The difference is created by heuristic information that is inversely proportional to the length of the shortest path of the selected lightpath. The ants use the lightpath pheromone matrix to choose the next lightpath. The heuristic information is used together with the shortest-path pheromone trails while deciding the proper shortest path of the chosen lightpath. A combined pheromone value, called the total pheromone, is used for this purpose. The pheromone values are updated after solutions are constructed. The amount of accumulated pheromones is proportional to the solution quality.

There are three different pheromone update procedures: global, weighted global, and local pheromone update. In global pheromone update, after each ant constructs its solution, both shortest-path and lightpath pheromones are updated on the edges the ants visited. The pheromones are updated according to the solution quality. The weighted global pheromone update differs from the global update in the amount that is added to the pheromones. The pheromones are increased with the amount weight/resource usage, where weight is used to deposit more or less pheromone for the selected ants. The details of the local pheromone update procedure can be found in [19].

In AS, every ant uses the global update pheromone procedure. In RAS, the weighted global update phero-

mone procedure is used with increasing weights for better solutions. The ant that constructs the best solution uses the weight w, the second best solution uses the weight $w-1$, and this update continues for the first w ranked ants. MMAS alternatively allows the iteration-best ant, the best-so-far ant, or the restart-best ant to deposit pheromones with equal chance. The iteration-best ant is the ant that constructs the best solution in the current iteration. The best-so-far ant is the one that constructs the best solution since the start of the algorithm. MMAS initializes the pheromone trails when diversity is lost or when no improvement occurs for a given number of consecutive iterations. The restart-best ant is the best solution constructed after this initialization. In the pheromone update procedure of ACS, only the best-so-far ant is allowed to deposit pheromones. Evaporation is implemented at the same time as accumulation and every ant uses the local pheromone update after each move. In BWAS, the global pheromone update is used by only the best-so-far ant, and the worst ant of the current iteration subtracts pheromones on the arcs it does not have in common with the best-so-far solution. When there are a few differences between the solutions of the best-so-far and iteration-worst ants, the pheromone trails are reinitialized to increase search diversification. To further increase diversity, pheromone mutation is used as explained in [19]. In the pheromone update procedure of elitist AS, the weighted global pheromone update procedure is used for the best-so-far ant. The weight is determined with the parameter e.

Solutions are constructed by applying the following simple constructive procedure to each ant: (1) choose a start lightpath and one of its shortest paths, (2) use lightpath pheromone information to select the next lightpath to route, (3) use shortest-path pheromone information together with the heuristic values to probabilistically determine the path between the nodes of the corresponding lightpath until all lightpaths have been visited. If the ant cannot select a shortest path that makes the solution feasible, this ant is removed from the current iteration.

Selection of the next step is implemented using the pseudorandom proportional action choice rule. According to this rule, each lightpath is assigned a probability proportional to the lightpath pheromone value. A lightpath is selected randomly based on these probabilities. The shortest path for the selected lightpath is chosen in the same way but the total pheromone value is used instead of the lightpath pheromone value.

C. Illustrative Example

Consider the physical and virtual topologies given in Figs. 1(a) and 1(b). The first four shortest paths calculated based on hop counts can be seen in Table I. Here, the first column shows the lightpaths as source–destination node pairs and the four shortest paths found using hop counts are given in the next four columns.

Assume we have a solution encoded as [1 2 1 3 1 1 2]. This encoding means that the first lightpath uses the first shortest path (1-2), the second one uses the second shortest path (1-2-4-5), and the third one uses first shortest path (1-2-4), etc. If we sum up the number of wavelength links used in this solution, we have a total of 12 wavelength links for hop-count evaluation.

For this sample solution, considering f_2 in the EA, if a failure occurs on the physical links connecting nodes 1-2, 2-4, or 3-4, the VT becomes disconnected. In f_2, a penalty is applied to the solution based on the sum of the total number of lightpaths that become disconnected in the event of each physical link failure. If the 1-2 link is broken, three lightpaths (a, b, and c) routed on this link will get disconnected; if the 2-4 link is broken, four lightpaths (b, c, d, and e) will be disconnected; and if the 3-4 link is broken, two lightpaths (d and f) would not find an alternative path to communicate. As a result, a penalty of $9p$ is added to the fitness, where p is the penalty factor. On the other hand, since ACO checks the constraints during solution construction, such a solution will not be generated.

Three different fitness values for EA, calculated using three different evaluation functions, f_1, f_2, and f_3, will be 312, 912, and 412, respectively, if we have a penalty factor of 100.

In the EA, if the crossover operator is applied to two individuals with the encodings [1 2 1 3 1 1 2] and [2 1 4 1 3 2 4], an offspring with encoding [2 2 1 1 3 2 2] will be created, if the second, third, and last genes are taken from the first parent and the other genes from the second parent. For the mutation operator of EA, assume a *ls_pm* mutation occurs on the third gene of this sample individual ([1 2 1 3 1 1 2]). If the shortest paths are calculated according to the hop-count method, the new individual becomes [1 2 2 3 1 1 2] or

TABLE I
FOUR DIFFERENT SHORTEST PATHS FOR THE LIGHTPATHS OF THE EXAMPLE VT GIVEN IN FIG. 1(B)

Lightpath	sp_1	sp_2	sp_3	sp_4
1-2 (a)	1-2	1-3-2	1-3-4-2	1-3-5-4-2
1-5 (b)	1-3-5	1-2-4-5	1-2-3-5	1-3-4-5
1-4 (c)	1-2-4	1-3-4	1-3-2-4	1-3-5-4
2-3 (d)	2-3	2-1-3	2-4-3	2-4-5-3
2-4 (e)	2-4	2-3-4	2-1-3-4	2-3-5-4
3-4 (f)	3-4	3-2-4	3-5-4	3-1-2-4
4-5 (g)	4-5	4-3-5	4-2-3-5	4-2-1-3-5

[1 2 4 3 1 1 2] with equal probability. This is because the first shortest path of the third lightpath has no common link with the second and the fourth shortest paths (least similar paths).

V. Experimental Study

In this section, we present the results obtained from the experiments to evaluate the efficiency of our NIHs. After deciding the best parameter set, we evaluate the performance of our algorithms and then we compare our results with the optimum solutions found with the ILP formulation.

In our preliminary studies [9,15], the hop-count calculation method was reported to perform better than the link-cost method. Therefore, in this study, we only give the results in which the hop-count calculation method is used for fitness evaluations.

In the experiments, we used two metrics for performance comparisons, namely, success rate and resource usage. The success rate is defined as the percentage of program runs in which a survivable mapping that does not violate the capacity constraint is found, and the resource usage is the total cost of wavelength links used throughout the network.

A. Parameter Setting Analysis

For fine-tuning the parameters to be used in the performance tests, we used a 24-node 43-link telco network topology [1].

The programs are run 20 times for each set of parameters, both for the EA and the ACO, and the results are the averages of these 20 runs.

In Tables II and III, the commonly used range of values, the range of values we tested, and the range of values that can be selected are listed for the EA and ACO, respectively.

1) Parameter Setting for EA: For the EA parameter setting tests, we used 50 randomly created 5-connected VTs for the telco network. The number of alternative shortest paths for each lightpath is selected as 15.

TABLE II
Default [18] and Tested Range of Values (RoV) for EA Parameters To Be Fine-Tuned

Parameter	Default RoV	Tested RoV	Eligible RoV
Crossover probability	[0.5,1.0]	0.7, 0.8, 0.9, 1.0	1.0
Mutation probability	$1/l$	$0.5/l$, $1/l$, $2/l$	$0.5/l$, $1/l$
Population size	NA	5, 10, 20, 50, 100, 200	10, 20, 50

TABLE III
Default [22] and Tested Range of Values (RoV) for ACO Parameters To Be Fine-Tuned

Parameter	Default RoV	Tested RoV	Eligible RoV
Number of ants	NA	1, 5, 10, 20, 50, 100	1, 5, 10, 20
ρ_0	0.5	0, 0.1, 0.2,..., 0.9, 1.0	0.1, 0.2
α	1	0, 1, 2, 3	2, 3
β	2	0, 1, 2, 3, 4	3, 4
e	100	1, 2, 3, 4, 5, 10, 20, 50, 100	1, 2, 3, 4, 5

As a result of preliminary tests comparing f_1, f_2, and f_3 fitness evaluation methods of EA, we saw that f_3 does not perform as well as the other two methods. Although, both f_1 and f_2 perform fairly well for the problem, f_2 has a lower algorithmic complexity [9]. Therefore, we used f_2 (see Subsection IV.A) as the fitness evaluation method. Here, the penalty factor is selected as 200. The tests are performed for all three mutation types we designed.

We performed a series of tests to fine-tune the EA parameters, including the crossover probability, the mutation probability, the population size, and the number of fitness calculations.

To find the most appropriate crossover probability, we tested four different crossover probabilities, i.e., 0.7, 0.8, 0.9, and 1.0. The results show that the best performance is achieved when the crossover probability is 1.0. For the mutation probability, we tested three different values, $0.5/l$, $1/l$, and $2/l$, where l is the number of lightpaths in the VT. According to the results obtained, we concluded that $1/l$ would be the most proper mutation probability. The third parameter that we fine-tuned is the population size, where we tested six different values, namely, 5, 10, 20, 50, 100, and 200. As a result, we selected the population size as 50. The last test for parameter setting is the maximum number of fitness calculations until stopping the EA. For this test, the program is run for 5 min for each of the three mutation types, during which the best individual is monitored every 5 s. From the results, we saw that there is not much change in resource usage of the best-so-far solution after 2 min, which corresponds to approximately 5000 fitness evaluations.

As a result of parameter setting tests, we considered a crossover probability of 1.0, a mutation probability of $1/l$, where l is the number of lightpaths, a population size of 50, and a maximum fitness evaluation count of 5000, for the EA performance tests.

2) Parameter Setting for ACO: The parameters for ACO are determined using a data set of 4-connected 20 VTs, where 10 shortest paths are provided for each

lightpath. For parameter setting tests, the MMAS algorithm is used due to its outstanding results among ACO algorithms for TSP in [19].

The first parameter setting test is to investigate the effect of the maximum allowed time on resource usage. For this test, the ACO algorithm is run for 60 s and the best-so-far solution is recorded every 5 s. The results show that the algorithm needs to run at most 15 s, because only a very small contribution is provided after around the 15th second.

To determine the number of ants used in ACO, we tested six different values, i.e., 1, 5, 10, 20, 50, and 100. According to the results, increasing the number of ants increases resource usage after five ants. Until a maximum number of solutions are generated, ants use pheromone matrices to construct solutions. Pheromone matrices are updated according to the solution quality after each ant constructs its solution. When the number of ants decrease, the use of pheromones increases. As a result, we set the number of ants as 10. ρ_0 is the parameter used while updating pheromone values. This parameter is determined after experimenting with ρ_0 values of 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. As a result, we selected ρ_0 as 0.1 where the best solutions are retrieved. The parameters α and β represent the weights of the pheromone level and the heuristic information, respectively. We tested all combinations of four different values of α, 0, 1, 2, and 3, and five different values of β, 0, 1, 2, 3, and 4. The contribution of both parameters on resource usage are negligible for values greater than 2. α does not seem to have much effect on resource usage. The parameters are set as α=3 and β=4, when the best solutions are retrieved. e is the weight of pheromones deposited for the best-so-far solution in the elitist AS algorithm. For the fine-tuning of e, we tested with e values of 1, 2, 3, 4, 5, 10, 20, 50, and 100. According to the results, there is not much difference in resource usage when e is selected between 1 and 4. We chose e as 3, since it is a middle value. As a result of the tests that will be explained in detail in the following subsection, we use the elitist AS in this paper. Therefore, the parameter setting of other ACO variants are omitted here.

B. Performance Comparison of EA and ACO

For the performance comparison experiments, we used three different physical topologies: the 14-node 24-link National Science Foundation (NSF) network, a 24-node 43-link telco network [1], and a 48-node 89-link network. To form the 48-node 89-link topology we merged two telco topologies. We combined two of this same topology via three nodes we selected.

In our initial experiments on the NSF network, both the EA and the ACO were able to find survivable solutions of equal quality for all VTs, with a 100% success rate. In this paper we report the results for a larger telco network and a 48-node 89-link network.

As a result of preliminary tests comparing f_1, f_2, and f_3 fitness evaluation methods of the EA, we saw that success rates for f_2 are less than 50% for 3-connected VTs, and the success rates for f_3 are even less. Thus, we chose f_1 as the fitness evaluation method. Similarly, we tested the three mutation types, *gene*, *ls_pm*, and *ms_pm*. According to the results, *ms_pm* performs very poorly, and the performance of the other mutation types, *gene* and *ls_pm*, are pretty similar. However, since gene mutation is much less complex than *ls_pm*, in which a similarity comparison is needed every time mutation occurs, we preferred gene mutation.

To determine the best ACO variant, we experimented with six direct variants of the AS algorithm listed in Subsection III.B. For each connectivity degree of the VT, success rates of each variant are mostly greater than 90% when more than five shortest paths are provided to the algorithms. Since the algorithms have relatively equal performances, we applied ANOVA tests to the results. We ended up with the result that ACS and AS are the worst and elitist AS is the best of all. The results given in the rest of this paper are the results obtained by elitist AS.

For the telco network, we assumed 10 wavelengths per physical link, and we created 100 random VTs with average connectivity degrees of 3, 4, and 5. The tests are repeated 20 times for each parameter set. The success rate and average resource usage results for this network are given in Table IV. In the table *SP* stands for shortest path. For each algorithm and VT connectivity degree, we examined three different numbers of alternative shortest paths for each lightpath. We used 5, 10, and 15 shortest paths. The results in each cell of the table are the average of 2000 tests, i.e., 20 runs for each of the 100 different VTs.

A quick observation of Table IV shows that success rates for both algorithms are very high. Generally ACO achieves higher success rates. From Table IV, we can see that success rates increase with the increase in the number of alternative shortest paths. The probability of random candidate solutions being survivable increases with the connectivity degree of the VT. Therefore, for higher connectivity degrees of VTs, both algorithms have higher success rates.

The average resource usage results in Table IV shows that the resource usage increases slightly with the increase in the number of alternative shortest paths, especially for EA. This is an expected result since the probability of getting stuck at local optima is higher in larger search spaces and both algorithms are allowed to run up to a predefined maximum time.

TABLE IV
SUCCESS RATES (SR) AND AVERAGE RESOURCE USAGES (RU) FOR 24-NODE TOPOLOGY AND VTS HAVING 3 DIFFERENT CONNECTIVITY DEGREES (3, 4, 5), SOLVED WITH 3 DIFFERENT NUMBERS OF SHORTEST PATHS (SPS) (5, 10, 15)

		5 SPs		10 SPs		15 SPs	
		EA	ACO	EA	ACO	EA	ACO
SR	3	97	98	98	100	97	100
	4	99	98	100	100	100	100
	5	100	100	100	100	99	100
RU	3	111	110	112	110	113	110
	4	144	143	145	143	146	144
	5	182	181	183	181	186	181

If the run times are increased, the resource usage results for different numbers of shortest paths will converge.

To see the performance of our NIHs in larger networks, we used the 48-node 89-link topology explained in the first paragraph of this subsection. Again, we created VTs of different average connectivity degrees. The time needed to create 100 different VTs for 48 nodes was very long and was increasing with the increase in the connectivity degree, so we created 100 different VTs for only 3 and 4 connectivity degrees and 50 different VTs for the 5 connectivity degree.

For this larger network, we increased the wavelength capacity. To find the suitable wavelength capacity, we tested 16 and 32 wavelengths. As a result, we set the wavelength capacity of a fiber as 32. In these tests, we used 10, 15, and 20 as the number of alternative paths. The maximum number of fitness calculations for EA for this network is determined with a test similar to the one given in Subsection V.A.1. The results show that 8000 is enough to find a feasible solution, which corresponds to approximately 8 min of CPU time. Similarly, we performed a test for ACO to determine the maximum allowed time, and as a result we run ACO for 350 s for this set of tests.

In Table V, the success rate and the average resource usage results for the 48-node 89-link network topology are given. From Table V, we can see that for each number of shortest paths, EA can find a result with a 100% success rate for 5-connected VTs. For 3- and 4-connected VTs the success rates increase with the increase in number of shortest paths. However, for ACO, the best performance is achieved with 15 shortest paths. If we compare EA and ACO, the performance of ACO is much better for the 3- and 4-connected VTs. For the 5-connected VTs, EA performs slightly better than ACO with 10 shortest paths.

Resource usages given in Table V show that, for the EA, the resource usage increases with the number of shortest paths. This is because of the search space getting larger while using a higher number of shortest paths and EA getting stuck at a local minimum. ACO finds better-quality results with a higher number of shortest paths.

C. Performance Comparison of NIHs With ILP

The ILP formulation for the survivable VT mapping problem was given in Susbection II.A. Since the problem has a large number of constraints, it is difficult to solve this ILP for large networks. To overcome this problem, Modiano and Narula-Tam [2] proposed two possible relaxations of the ILP formulation. The first relaxation proposed is a simple relaxation that applies the survivability constraints only to cuts that include a single node, which prevents a single node from get-

TABLE V
SUCCESS RATES (SR) AND AVERAGE RESOURCE USAGES (RU) FOR 48-NODE TOPOLOGY AND VTS HAVING 3 DIFFERENT CONNECTIVITY DEGREES (3, 4, 5), SOLVED WITH 3 DIFFERENT NUMBERS OF SHORTEST PATHS (SP) (10, 15, 20)

		10 SPs		15 SPs		20 SPs	
		EA	ACO	EA	ACO	EA	ACO
SR	3	52	69	67	90	78	78
	4	82	91	86	96	89	89
	5	100	96	100	100	100	100
RU	3	311	405	314	310	317	305
	4	416	444	419	379	424	352
	5	519	670	525	605	532	518

ting disconnected in the event of a fiber cut. With this relaxation, the number of survivability constraint equations is reduced to the number of nodes. We implemented the basic and the relaxed ILP using the CPLEX suite [23].

For the comparison of basic and relaxed ILP solutions and our NIHs, the best EA combination, i.e., the gene mutation with f_1, and elitist AS for ACO with the most proper parameter set, are used. The results obtained are given in Table VI.

For 3-connected VTs, we could solve the problem optimally using basic ILP for only 58% of VTs. For the remaining 42%, we were out of memory, although the physical memory of the computer was 3 Gb. Furthermore, if we increase the connectivity degree of the VT, solving the problem using basic ILP was impossible.

It is reported in [2] that relaxations also perform well for this problem with much smaller run times. For each VT, we also solved the problem using this ILP relaxation. Different from [2] we also included the capacity constraint in the ILP formulation.

Table VI shows that both of our NIHs can find a feasible solution with a 100% success rate. However, the results with ILP relaxation are not that good, and the basic ILP can only find a result for nearly half of the 3-connected VTs.

In the experimentation, we could obtain optimum solutions, using basic ILP, for only a very small part of the problem set. The basic ILP cannot find a solution with less resource usage than the relaxation used here. Therefore, we compared our solutions with the ILP relaxation results, where survivability constraints are relaxed, but the obtained results form a lower bound on resource usage. In Table VII, we can see that EA and ACO find the optimum solution for almost all test cases, with ACO performing slightly better.

The time needed to create the CPLEX input file for the basic ILP solution was 2 hours and for some cases it went up to more than 15 hours on a 1.6 GHz Pentium IV. ILOG CPLEX solved the problem in 5 min, and this solution time went up to more than 15 min for some cases. Moreover, the total size of CPLEX input files for the 100, 3-connected VTs is more than 30 Gbs. On the other hand, for this data set, EA does not need more than a minute to solve the problem and ACO can solve it in 10 s, on average. As a result, EA and ACO can find the optimum solutions for 3-connected VTs within 120 and 720 times shorter running times, respectively.

TABLE VI
PERCENTAGE OF FEASIBLE SOLUTIONS FOR 24-NODE TOPOLOGY AND VTS OF 3 DIFFERENT CONNECTIVITY DEGREES (3, 4, 5)

	EA	ACO	ILP Relaxation	Basic ILP
3	100	100	52	58
4	100	100	88	NA
5	100	100	100	NA

TABLE VII
PERCENTAGE OF OPTIMUM SOLUTIONS FOR 24-NODE TOPOLOGY AND VTS OF 3 DIFFERENT CONNECTIVITY DEGREES (3, 4, 5)

	EA	ACO
3	97	98
4	97	98
5	100	100

The running times of EA and ACO for different sizes of networks and different sizes of connectivity degrees are given in Table VIII. The running times in the table are the averages of all runs for all the VTs with corresponding connectivity degree. Since the termination condition for ACO is a predefined time, ACO has the same running time for all test cases of the same physical topology. From the table, it can be seen that the increase in running time with the increase in network size is not as much as it is for the CPLEX.

As a conclusion, the results show that both heuristics are promising for the survivable VT mapping problem. Since the time needed to find a feasible solution is at most 8 min (for the routing of 5-connected VTs on the 48-node 89-link topology using EA), these heuristics can easily be applied to real-world applications.

VI. CONCLUSION

In this work, we have studied the survivable virtual topology mapping problem in optical WDM networks. Our objective was to minimize the total number of wavelength links used throughout the network. We designed two different nature-inspired heuristics, namely, EA and ACO, to solve the problem. We explored the effectiveness of these algorithms on network topologies of different sizes. To assess the performance of our NIHs, we compared our experimental

TABLE VIII
RUNNING TIME AVERAGES (IN S) OF EA AND ACO FOR 24-NODE AND 48-NODE PHYSICAL TOPOLOGIES AND VTS HAVING 3 DIFFERENT CONNECTIVITY DEGREES (3, 4, 5)

	24-node network		48-node network	
	EA	ACO	EA	ACO
3	30	15	300	350
4	50	15	400	350
5	60	15	480	350

results with the ones obtained using basic ILP and an ILP relaxation given in [2]. The results clearly demonstrate that both of our algorithms can solve the problem even for large-scale network topologies for which neither ILP can find the optimum solution nor can the ILP relaxation find a survivable solution. Additionally, the CPU time and the memory used by NIHs is fairly low compared with the ILP method. We are currently applying these NIHs to the VT design problem.

ACKNOWLEDGMENTS

Fatma Corut Ergin and Elif Kaldırım receive a scholarship from TÜBİTAK (The Scientific and Technological Research Council of Turkey). Preliminary results of this study, "Performance Analysis of Nature Inspired Heuristics for Survivable Virtual Topology Mapping," was published in the IEEE Global Telecommunications Conference, IEEE GLOBECOM, in Hawaii, USA, in 2009.

REFERENCES

[1] B. Mukherjee, *Optical WDM Networks*. New York: Springer, 2006.

[2] E. Modiano and A. Narula-Tam, "Survivable lightpath routing: a new approach to the design of WDM-based networks," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 800–809, 2002.

[3] J. Armitage, O. Crochat, and J. Y. Le Boudec, "Design of a survivable WDM photonic network," in *Proc. IEEE INFOCOM*, 1997, pp. 244–252.

[4] A. Nucci, B. Sanso, T. Crainic, E. Leonardi, and M. A. Marsan, "Design of fault-tolerant logical topologies in wavelength-routed optical IP networks," in *Proc. of IEEE Globecom*, 2001, pp. 2098–2103.

[5] F. Ducatelle and L. M. Gambardella, "A scalable algorithm for survivable routing in IP-over-WDM networks," in *Proc. of Int. Conf. on Broadband Networks*, 2004, pp. 54–63.

[6] M. Kurant and P. Thiran, "Survivable mapping algorithm by ring trimming (SMART) for large IP-over-WDM networks," in *Proc. of Int. Conf. on Broadband Networks*, 2004, pp. 44–53.

[7] N. Banerjee and S. Sharan, "An evolutionary algorithm for solving the single objective static routing and wavelength assignment problem in WDM networks," in *Proc. of ICISIP*, 2004, pp. 13–18.

[8] M. Saha and I. Sengupta, "A genetic algorithm based approach for static virtual topology design in optical networks," in *Proc. of INDICON*, 2005, pp. 392–395.

[9] F. C. Ergin, A. Yayımlı, and Ş. Uyar, "An evolutionary algorithm for survivable virtual topology mapping in optical WDM networks," in *EvoWorkshops09, LNCS 5484*, 2009, pp. 31–40.

[10] G. N. Varela and M. C. Sinclair, "Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation," in *Congr. on Evolutionary Computation*, 1999, vol. 3, pp. 1809–1816.

[11] R. M. Garlick and R. S. Barr, "Dynamic wavelength routing in WDM networks via ant colony optimization," in *3rd Int. Workshop on Ant Algorithms, LNCS 2463*, 2002, vol. 3, pp. 250–255.

[12] S. H. Ngo, X. Jiang, and S. Horiguchi, "An ant-based approach for dynamic RWA in optical WDM networks," *Photonic Network Commun.*, vol. 11, pp. 39–48, 2006.

[13] S. H. Ngo, X. Jiang, V. Le, and S. Horiguchi, "Ant-based survivable routing in dynamic WDM networks with shared backup paths," *J. Supercomput.*, vol. 36, no. 3, pp. 297–307, 2006.

[14] A. Hassan, C. Phillips, and J. Pitts, "Dynamic routing and wavelength assignment using hybrid particle swarm optimization," in *EPSRC PGNet*, 2007.

[15] E. Kaldırım, F. C. Ergin, Ş. Uyar, and A. Yayımlı, "Ant colony optimization for survivable virtual topology mapping in optical WDM networks," in *Proc. of ISCIS*, 2009, pp. 334–339.

[16] A. Todimala and B. Ramamurthy, "A scalable approach for survivable virtual topology routing in optical WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 6, pp. 63–69, 2007.

[17] H. H. Hoos and T. Stutzle, *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann, 2005.

[18] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer Verlag, 2003.

[19] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge, MA: Massachusetts Institute of Technology, 2004.

[20] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 26, pp. 29–41, 1996.

[21] M. Dorigo and T. Stutzle, "The ant colony optimization metaheuristics: algorithms, applications, and advances," in *Handbook of Metaheuristics* (International Series in Operations Research and Management Science 57), 2002, pp. 251–285.

[22] T. Stutzle, Ant Colony Optimization Source Code, 2004. Available: http://www.aco-metaheuristic.org/aco-code.

[23] *ILOG CPLEX 6.5 User's Manual*, 2000.