Performance Analysis of Nature Inspired Heuristics for Survivable Virtual Topology Mapping

Fatma Corut Ergin
Marmara University
Istanbul, Turkey
Email: fatma.ergin@marmara.edu.tr

Elif Kaldırım
Istanbul Technical University
Istanbul, Turkey
Email: kaldirime@itu.edu.tr

Ayşegül Yayımlı
Istanbul Technical University
Istanbul, Turkey
Email: gencata@itu.edu.tr

Şima Uyar
Istanbul Technical University
Istanbul, Turkey
Email: etaner@itu.edu.tr

***Abstract*—The high capacity of fibers used in optical networks, can be divided into many channels, using the WDM technology. Any damage to a fiber causes all the channels routed through this link to be broken, which may result in a serious amount of data loss. As a solution to this problem, the virtual layer can be mapped onto the physical topology, such that, a failure on any physical link does not disconnect the virtual topology. This is known as the survivable virtual topology mapping problem. In this study, we investigated the performance of two popular nature inspired heuristics, namely, evolutionary algorithms and ant colony optimization, in finding a survivable mapping of a given virtual topology while minimizing the resource usage. Our results show that both nature inspired heuristics perform remarkably well for this problem. Furthermore, both methods can obtain high quality solutions in less than a minute.**

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 the WDM (wavelength division multiplexing) technology. Each of these channels work 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. This architecture is known as IP-over-WDM, or Ethernet-over-WDM.

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 different wavelengths, can be routed on a single fiber. All the lightpaths set up on the 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 problem. VT mapping problem, a subproblem of VT design, is to find a proper route for each lightpath, 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, a fiber damage may result in a serious amount of data loss. Two different approaches can be used to avoid this: 1. Survivable design of the physical layer, 2. Survivable design of the virtual layer. The first approach is the problem of designing a backup link/path for each link/path of the virtual layer. The second approach is the problem of designing the virtual layer such that the virtual layer remains connected in the event of a single or multiple link failure. While the first approach provides faster recovery for time-critical applications (such as, IP phone, telemedicine) by reserving more resources; the second approach, i.e. the survivable VT design, relies on the recovery mechanisms of the upper layers and aims to protect data communication using less resources.

Nature inspired computing is an umbrella term that covers computing techniques which are inspired from nature or are modeled on natural processes or are constructed by using biological materials. Some of these techniques are relatively new, while some are quite old and have been extensively researched in literature. Today, most of them have become state-of-the-art techniques for many hard to solve academic problems as well as real-world problems. Evolutionary algorithms, ant colony algorithms, particle swarm intelligence techniques, artificial immune systems, neural networks, quantum computing and DNA computing are some of the approaches in literature that fall into this category. These techniques are commonly referred as nature inspired heuristics (NIH).

The VT mapping problem is known to be NP-complete [2]. Because of its complexity, it is not possible to solve the problem optimally in an acceptable amount of time using classical optimization techniques, for real-life sized networks. Therefore, heuristic approaches should be used. In this study, we propose to use NIHs for survivable VT mapping problem and compare their performance. We chose evolutionary algorithms (EAs) due to their successful applications on NP-complete problems and ant colony optimization (ACO) due to their successful performance on constrained combinatorial optimization problems. For the experimentation, we selected the EA and ACO with the most promising components according to a set of preliminary tests. As a result of the experiments, we show that both ACO and EA can solve the survivable VT mapping problem in a reasonable amount of time.

The rest of the paper is organized as follows. In section II the problem is defined and related literature is given. Next, in section III, the NIHs used in this study, are briefly explained. The details of how these techniques are applied to our problem

978-1-4244-4148-8/09/$25.00 ©2009
This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE "GLOBECOM" 2009 proceedings.

are given in section IV, followed by an example of the application. Finally, in section V, the experimental results are given and these results are discussed thoroughly.

II. Problem Definition and Related Work

In this work, given the physical and the virtual topologies, our aim is to find a survivable mapping of the VT. We define survivable mapping of VT as to find a route for each lightpath, such that in case of a single physical link failure the VT remains connected.

A. Formal Problem Definition

The physical topology is composed of a set of nodes $N = \{1..N\}$ 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 a given physical topology is given in [2]. Based on this formulation, different objective functions can be considered for the problem of survivable mapping. The simplest objective is to minimize the total number of physical links used. Another objective is to minimize the total number of wavelength-links used in the whole physical topology. A wavelength-link is defined as a wavelength used on a physical link. Our choice as the objective is the latter one, since it gives a better idea of the actual resource usage.

There are two main constraints of the problem:

a)Survivability constraint: The survivability constraint states that for all proper cuts of the VT, the number of cut-set links flowing on any given physical link is less than the size of the cut-set This means that all the lightpaths of a cut-set cannot be routed using the same physical link. b)Capacity constraints: This constraint ensures that the number of wavelengths on a physical link does not exceed its capacity W.

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 minimize the number of source-destination pairs that become disconnected in the event of a physical link failure. Nucci et al. [4] also used tabu search to solve the survivable VT design problem. The constraints in this study include transmitter and receiver constraints as well as wavelength capacity constraints.

Ducatelle et al. [5] studied the VT mapping problem using a local search algorithm, while Kurant and Thiran [6] used an algorithm that divides the survivable mapping 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]. Ergin et al. [9] proposed the only EA based approach for survivable VT mapping problem. Their 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. Ant colony optimization is applied to the static [10] and dynamic [11], [12] RWA problem without the survivability constraint. The only study using ACO considering back-up paths on the physical layer is [13]. Particle swarm optimization is applied to the RWA problem in only [14], in which the survivability constraint is not considered.

Modiano and Narula-Tam used ILP (Integer Linear Programming) 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. For the cases when ILP cannot find an optimum solution in a reasonable amount of time due to the problem size, Modiano et al. proposed two relaxations to ILP, which consider only small-sized cut-sets. These relaxations reduce the problem size; however, they may lead to suboptimal solutions. In order to overcome the long execution time problem in ILP formulation, Todimala and Ramamurthy proposed a new ILP formulation and they solved the problem for networks of up to 24 nodes [15]. In [15], besides the physical network and the virtual network topologies, the shared risk link groups should be known in advance.

III. Nature Inspired Heuristics

The most popular heuristics used in literature can be collected under two main titles: 1. Search techniques starting from a single point, 2. Search techniques starting from several points. Among the most common methods in the first group are simulated annealing and tabu search algorithms. The second group includes evolutionary algorithms (EA) (genetic algorithms (GA), genetic programming (GP), evolutionary strategies (ES), differential evolutionary algorithms (DEA), etc.), swarm intelligence algorithms (ant colony optimization (ACO), particle swarm optimization (PSO), etc.), and artificial immune system (AIS) techniques. Methods which start from a single point can get stuck at a local optimum. However, for techniques that start from several points in the search space, the probability to get stuck at a local optimum is smaller.

In combinatorial optimization problems, a solution consists of a discrete set of subparts, which are called *solution components*. One method to classify search methods [16], relies on the underlying mechanism used to generate candidate solutions for a problem from the solution components: 1. A solution candidate can be transformed into another one by altering one or more of the components. A search algorithm which uses this approach is called a *perturbative search* algorithm. Evolutionary algorithms, particle swarm optimization and simulated annealing are among the well known perturbative search methods. 2. On the other hand, it is also possible to construct a complete solution candidate by iteratively adding solution components to the partial solution candidates. A search algorithm which uses this approach is called a *constructive search* algorithm. Ant colony optimization algorithms fall within the category of constructive search methods.

978-1-4244-4148-8/09/$25.00 ©2009

A. Evolutionary Algorithms

EAs are population based search and optimization techniques inspired from Darwin's evolutionary theory and classical Mendelian genetics. EAs have been applied to many different problem domains with successful results.

In an EA, search starts from an initial set of candidate solutions (*individuals*) in the search space which form the *population*. Each input parameter of the problem is called a *gene* and the solution string made up of these parameters is a *chromosome*. The main EA loop continues until some *stopping criteria* are met The function for evaluating and comparing individuals in the population is called a *fitness function*.

In the first step of the EA loop, some of the individuals are *selected* to undergo *reproduction*. Higher quality solutions (with good *fitness* values) have a higher chance of being selected. In the reproduction stage, two genetic operators are applied: *crossover* and *mutation*. Crossover acts on two individuals, creating a new individual through combining parts of the chromosomes from each. Crossover occurs with a predefined probability called the *crossover probability*. The offspring undergoes mutation which acts on each gene separately with a predefined *mutation probability*. Some of the offspring generated through selection and reproduction are chosen to *replace* some of the existing individuals in the current population. In the steady state replacement scheme, only one offspring is generated in each iteration and the new individual replaces an existing one in the population. Duplicate individuals are not inserted into the population. This is called *duplicate elimination*. After this step, a new iteration (*generation*) begins. The basic algorithm flow is given in Algorithm 1. A detailed look at EAs can be found in [17].

Algorithm 1 Basic Steady-state Evolutionary Algorithm

```
1:  generate initial population
2:  evaluate initial population
3:  while not termination criteria met do
4:      select parents to mate
5:      generate one offspring through reproduction
6:      evaluate the created offspring
7:      if fitness of offspring better than fitness of worst then
8:          offspring replaces worst in population
9:      else
10:         discard offspring
11:     end if
12: end while
```

B. Ant Colony Optimization Algorithms

ACO is one of the most commonly used swarm intelligence techniques and is based on the behavior of real ants. ACO has been applied successfully to many combinatorial optimization problems such as routing problems, assignment problems, scheduling and sequencing problems and subset problems. One of the first successful implementations of ACO is the Ant System (AS). AS has been the basis for many ACO variants which have become the state-of-the-art for many applications. These variants include elitist AS (EAS), rank-based AS (RAS), MAX-MIN AS (MMAS), ant colony system (ACS), best-worst AS (BWAS), the approximate nondeterministic tree search (ANTS), and the hyper-cube framework. AS, ACS, EAS, 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.

The algorithmic flow of the basic ACO algorithm [18] is given in Algorithm 2. An *iteration* consists of the *solution construction* and *pheromone update* stages.

Algorithm 2 Basic Ant Colony Optimization Algorithm

```
1:  set ACO parameters
2:  initialize pheromone levels
3:  while stopping criteria not met do
4:      for each ant k do
5:          select random initial node
6:          repeat
7:              select next node based on decision policy
8:          until complete solution achieved
9:      end for
10:     update pheromone levels
11: end while
```

In each iteration, each *ant* in the *colony* constructs a complete solution. Ants start from random *solution components* and continues by adding the next component. For each component, the next component to be added is determined through a stochastic local *decision policy* based on the current *pheromone levels* and *heuristic information* between the current component and the others. An ant k determines its next component with a probability p_{ij}^k as calculated in Eq. 1,

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in N_i^k} \tau_{il}^\alpha \cdot \eta_{il}^\beta} & \text{if } j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where τ_{ij} and η_{ij} are the pheromone level and heuristic information between components i and j respectively, α and β are parameters used to determine the effect of the pheromone level and heuristics information respectively, N_i^k is the allowed neighborhood of ant k when it is at node i.

Pheromone levels are modified when all ants have constructed a complete solution. First, the pheromone values are lowered (*evaporated*) by a constant factor between all component pairs. Then, pheromone values are increased between the components which the ants have used during their solution construction. Pheromone evaporation and pheromone update by the ants are implemented as given in Eq. 2 and Eq. 3 respectively,

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} \quad (2)$$

978-1-4244-4148-8/09/$25.00 ©2009

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad (3)$$

where, m is the number of ants, $0 < \rho \leq 1$ is the pheromone evaporation rate and $\Delta\tau_{ij}^{k}$ is the amount of pheromone ant k deposits between the solution components it has used. $\Delta\tau_{ij}^{k}$ is defined as given in Eq. 4, where C_k is the cost of the solution T_k built by the k-th ant.

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/C_k & \text{if } edge(i,j) \in T_k \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

IV. Application of the NIHs to the Problem

Designing a solution representation that is well-suited to the problem is crucial in EA and ACO performance. 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 $[1..k]$ where *k* is the predefined number of shortest paths for the lightpaths.

Our objective is to minimize the total cost of resources used throughout the network. This cost is evaluated in two different ways: (1) by considering the actual lengths of the physical links (link cost), and (2) by counting the number of physical links used (hop count). In both heuristics the objective is to minimize the resource usage. In the EA constraint violations are considered as penalties in the fitness evaluation stage. In ACO constraints are taken into consideration during solution construction, therefore constraint violation is not possible.

A. Evolutionary Algorithm Design

A steady state EA with duplicate elimination is used. After a random initial population generation, the EA operators are applied to the solutions until a predefined number of fitness evaluations are executed.

Binary tournament selection is used for selecting the mating pairs. An offspring gets each gene from either mate with equal probability. The mutation operator considers the physical link similarities between the shortest paths of each lightpath. If mutation occurs on a gene, its current value is replaced by the index of the least similar shortest path for the corresponding lightpath, similarity being defined as the number of common physical links. This mutation operator aims to preserve diversity in the population.

At the final stage of the loop, the fitness value of the offspring is calculated and compared to the worst individual in the current population. If the offspring is different than this individual and it has a better fitness, it replaces the worst individual, otherwise it is discarded.

Violations of the constraints for the problem, i.e. the survivability and the capacity constraints, are included as penalties in the fitness function. In order to determine if the solution is survivable or not, each physical link is deleted from the physical network one by one. If the virtual topology becomes disconnected in the event of a broken physical link, the solution is taken as unsurvivable. The penalty for an unsurvivable solution is determined as the sum of the total number of lightpaths that become disconnected in the event of each physical link failure [3], [5]. A capacity constraint violation adds a penalty value which is proportional to the total number of physical links which exceed the predetermined wavelength capacity. These two penalties are multiplied with a penalty factor and added to the fitness of the solution.

B. Ant Colony Algorithms Design

In this study, we chose RAS as the ACO algorithm. The main idea of RAS is to allow each ant to deposit an amount of pheromone which decreases with its solution rank. The ants are sorted in decreasing order according to the quality of the solutions they constructed. The amount of pheromone an ant deposits is weighted according to its rank r. In each iteration only the $(w-1)$ best-ranked ants and the ant which has constructed the best-so-far solution are allowed to deposit pheromones. The best-so-far solution has the largest weight w, while the r-th best ant of the current iteration contributes pheromones with a weight given by $max\ \{0, w-r\}$.

Two pheromone trails are implemented in the survivable VT mapping problem: the *lightpath pheromone trails* τ_{ij}^{l} refer to the desirability of visiting lightpath j directly after i, while the *shortest path pheromone trails* τ_{ij}^{s} show the desirability of selecting j^{th} shortest path of lightpath i.

In our ACO design, the heuristic information η_{ij} is inversely proportional to the length of the j^{th} shortest path of lightpath i, i.e. $\eta_{ij} = 1/d_{ij}$. The heuristic information is used together with the shortest path pheromone trails while deciding the proper shortest path of chosen lightpath.

Each ant initially chooses a random lightpath and selects one of its shortest paths. At each step, the ant iteratively adds an unvisited lightpath to its partial solution and decides the shortest path of the selected lightpath. The solution construction terminates once all lightpaths have been visited. 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 to select the next lightpath (3) use shortest path pheromone together with 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, i.e., all alternative shortest paths violate the survivability and the capacity constraints, this ant is removed from the current iteration.

C. An Example

Consider the physical and virtual topologies given in Figure 1. The first 3 shortest paths calculated based on hop counts and based on link costs can be seen in Table I. Here, the first column shows the lightpaths as source-destination node pairs. Three shortest paths found using hop counts are given in the

978-1-4244-4148-8/09/$25.00 ©2009

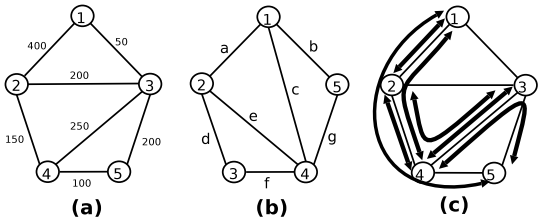

Fig. 1. a. Physical Topology, b. Virtual Topology c. Example Mapping

TABLE I
THREE DIFFERENT SHORTEST PATHS FOR THE LIGHTPATHS OF THE EXAMPLE VIRTUAL TOPOLOGY GIVEN IN FIGURE 1.

	hop count			link-cost		
lightpath	sp_1	sp_2	sp_3	sp_1	sp_2	sp_3
1-2 (a)	1-2	1-3-2	1-3-4-2	1-3-2	1-2	1-3-4-2
1-4 (c)	1-2-4	1-3-4	1-3-2-4	1-3-4	1-3-5-4	1-3-2-4
1-5 (b)	1-3-5	1-2-4-5	1-2-3-5	1-3-5	1-3-4-5	1-3-2-4-5
2-3 (d)	2-3	2-1-3	2-4-3	2-3	2-4-3	2-1-3
2-4 (e)	2-4	2-3-4	2-1-3-4	2-4	2-3-4	2-3-5-4
3-4 (f)	3-4	3-2-4	3-5-4	3-4	3-5-4	3-2-4
4-5 (g)	4-5	4-3-5	4-2-3-5	4-5	4-3-5	4-2-3-5

first three columns, and 3 shortest paths found using link costs are given in the next three columns.

Assume we have a solution encoded as [1 1 2 3 1 1 2]. This encoding means that the first lightpath uses the 1^{st} shortest path (1-2), the second one uses the 1^{st} shortest path (1-2-4), and the third one uses 2^{nd} shortest path (1-2-4-5), 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, and 2250 kilometers for link cost evaluation.

For this sample solution, if a failure occurs on the physical links connecting nodes 1-2, 2-4, or 3-4, the virtual topology becomes disconnected. In the EA, a penalty is applied to this solution based on the sum of the total number of lightpaths that become disconnected in the event of each physical link failure. If 1-2 link is broken, 3 lightpaths (a,b, and c) routed on this link will get disconnected, if 2-4 link is broken, 4 lightpaths (b,c,d, and e) will be disconnected, and if 3-4 link is broken, 2 lightpaths (d and f) would not find an alternative path to communicate. As a result, a penalty of $9 * p$ 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 would not be generated.

In the EA, assume that a mutation occurs on the second gene of this sample individual. If shortest paths are calculated according to hop count, the new individual becomes [1 2 2 3 1 1 2]. However, if link costs are used instead, the individual becomes [1 2 2 3 1 1 2] or [1 3 2 3 1 1 2] with equal probability.

V. EXPERIMENTS

To compare the performance of the EA and ACO for the survivable VT mapping problem, we performed a series of experiments. In these experiments, we used two metrics for performance comparisons, namely success rate, and resource usage. Success rate is defined as the percentage of program runs in which a survivable mapping that does not violate the capacity constraint is found. Resource usage is the total cost of wavelength-links used throughout the network.

A. Experimental Setup

For the experiments, we used two different physical topologies: the 14-node 24-link NSF network and a 24-node 43-link network (see [1] chapter 11 pp.557). For each physical topology we created 50 random VTs with average connectivity degrees of 3, 4, and 5. We assumed 10 wavelengths per physical link.

Here, we used the EA approach (see section IV-A) and the parameter set with the best performance from [9]. In the EA performance tests, we considered a mutation probability of $1/l$, where l is the number of lightpaths, a crossover probability of 1.0 and a population size of 100. A penalty factor of 200 is used in the tests using hop count for shortest path calculation, and 300 in the tests using link cost.

We experimented with 6 direct variants of AS algorithms listed in section III-B with different parameter settings. As a result of the experiments we chose RAS with the following parameter values:

α	β	ρ	m	q_0	w	τ_0
1	1	0.1	10	0.5	6	$1/\rho C_{min}$

The termination criterion for both algorithms is to create a predefined number of points in the solution space. We applied separate tests to each algorithm and determined this number as the iteration count after which there is no improvement in solution quality. As a result, we decided this number to be 5000 for the EA, and 100 for the ACO. We should note that each run of the programs take less than a minute on the average.[1]. We performed 20 runs for each experiment.

B. Experimental Results

We performed 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 100% success rate. In this paper we report the results for the larger 24-node 43-link network.

The results of the experiments are given in Tables II, and III. Table II shows the success rates of both heuristics averaged over 1000 runs (20 runs per VT instance). 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 calculated according to hop count and link cost. Table III shows the lower (l) and upper (u) bounds for the resource usage of both algorithms with 95% confidence interval calculated using only the results of the successful runs.

A quick observation of Table II shows that success rates for both algorithms are very high. Generally ACO achieves higher

[1]All the experiments were performed on Intel Pentium IV 1.6 GHz. PCs running Linux

978-1-4244-4148-8/09/$25.00 ©2009

TABLE II
SUCCESS RATES

		5 shortest paths		10 shortest paths		15 shortest paths	
		EA	ACO	EA	ACO	EA	ACO
hop count	3	0.26	0.72	0.55	0.92	0.59	0.97
	4	0.87	0.94	0.95	1	0.97	1
	5	0.97	0.98	1	1	1	1
link cost	3	0.19	0.60	0.53	0.93	0.60	0.98
	4	0.84	0.88	0.94	0.96	0.97	1
	5	0.98	0.83	1	1	1	1

success rates. The experiments using hop count calculation method performs better in terms of success rate for both heuristics. In Table II, we can see that success rates increase with the increase in the number of alternative shortest paths. In EA tests; for 3 connected VTs, success rate for 10 shortest paths are more than twice the success rate for 5 shortest paths, however, the increase of success rate from 10 shortest paths to 15 shortest paths is not that high. This is because of the exponential growth of the search space.

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. However, the EA has a much smaller success rate than ACO for 3 connected VTs.

Table III shows that the resource usage increases slightly with the increase in the number of alternative shortest paths. 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. If the run times are increased, the resource usage results for different number of shortest paths will converge. Even though success rates in Table II are higher for ACO on 4 and 5 connected VTs, Table III shows that the solution quality of EA is better.

To assess the quality of our solutions, we also implemented Relaxation-1 of the ILP formulation in [2]. The percentage of EA and ACO solutions having the same resource usage as those obtained by Relaxation-1 lie in the interval of 95% to 100%.

VI. CONCLUSION AND FUTURE WORK

High success rates show that both heuristics are promising for the survivable VT mapping problem. ACO performs better for sparse VTs both according to success rate and resource usage. However, for dense VTs, ACO gives better success rates whereas EA gives better quality results. Since the time needed to find a feasible solution is less than a minute, these heuristics can easily be applied to real world applications. As a future work, we will examine the algorithm specific parameters to see how they change the performance of the algorithms.

TABLE III
LOWER AND UPPER BOUNDS OF RESOURCE USAGE WITH 95% CONFIDENCE INTERVAL

			5 shortest paths		10 shortest paths		15 shortest paths	
			EA	ACO	EA	ACO	EA	ACO
hop count	3	l	113.24	111.53	115.15	115.88	117.48	116.99
		u	114.86	112.48	116.39	116.80	118.84	117.91
	4	l	146.8	146.54	150.84	154.51	153.9	156.82
		u	147.64	147.31	151.7	155.41	154.78	157.78
	5	l	185.64	187.01	192.32	197.18	197.86	201.55
		u	186.8	188.19	193.53	198.42	199.17	202.85
link cost	3	l	115775	111175	115733	116329	118524	117838
		u	117751	112507	117104	117391	119954	118934
	4	l	147272	147319	151183	154017	154928	156137
		u	148379	148317	152166	155061	156528	157223
	5	l	200687	189479	199330	199969	203287	204412
		u	209674	191047	204847	201535	207230	205966

REFERENCES

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

[2] E. Modiano and A. Narula-Tam, *Survivable Lightpath Routing: A New Approach to the Design of WDM-Based Networks*, IEEE Journal on Selected Areas in Communications, vol.20, no.4, p.800-809, May 2002.

[3] J. Armitage, O. Crochat, J.-Y. Le Boudec, *Design of a Survivable WDM Photonic Network*, INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Driving the Information Revolution, 1997.

[4] A. Nucci, B. Sanso, T. Crainic, E. Leonardi, and M.A. Marsan, *Design of Fault-Tolerant virtual Topologies in Wavelength-Routed Optical IP Networks*, IEEE Globecom, 2001.

[5] F. Ducatelle, L. M. Gambardella, *A Scalable Algorithm for Survivable Routing in IP-Over-WDM Networks*, BroadNets, 2004.

[6] M. Kurant, P. Thiran, *Survivable Mapping Algorithm by Ring Trimming (SMART) for Large IP-over-WDM Networks*, BroadNets, 2004.

[7] N. Banerjee and S. Sharan, *An EA for Solving the Single Objective Static RWA Problem in WDM Networks*, 2nd International Conference on Intelligent Sensing and Information Processing, 2004.

[8] M. Saha and I. Sengupta, *A GA Based Approach for Static Virtual Topology Design in Optical Networks*, IEEE Conference on Control, Communication and Automation, 2005.

[9] F. C. Ergin, A. Yayimli, S. Uyar, *An Evolutionary Algorithm for Survivable Virtual Topology Mapping in Optical WDM Networks*, EvoWorkshops09, LNCS 5484, Springer-Verlag, p.3140, 2009.

[10] G.N. Varela, M.C. Sinclair, *Ant Colony Optimisation for Virtual-Wavelength-Path Routing and Wavelength Allocation*, Congress on Evolutionary Computation, v.3, p.1809-1816, 1999.

[11] R.M. Garlick, R.S. Barr, *Dynamic Wavelength Routing in WDM Networks via Ant Colony Optimization*, Third International Workshop on Ant Algorithms, LNCS 2463, Springer-Verlag, v.3, p.250-255, 2002.

[12] S. Ngo, X. Jiang, S. Horiguchi, *An Ant-Based Approach for Dynamic RWA in Optical WDM Networks*, Photonic Network Communications 11, 39-48, 2006.

[13] S. Ngo, X. Jiang, V. Le, and S. Horiguchi. *Ant-based survivable routing in dynamic WDM networks with shared backup paths*, Journal of Supercomputing , v.36, no.3, p.297-307, June 2006.

[14] A. Hassan, C. Phillips and J. Pitts, *Dynamic Routing and Wavelengt Assignment using Hybrid Particle Swarm Optimization*, EPSRC PGNet, 2007.

[15] A. Todimala, B. Ramamurthy, *A Scalable Approach for Survivable Virtual Topology Routing in Optical WDM Networks*, IEEE Journal on Selected Areas in Communications, v.25, no.6, August 2007.

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

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

[18] M. Dorigo, T. Stützle, "Ant Colony Optimization", Massachusetts Institute of Technology, Cambridge Massachusetts, Chapter 1, 1-24, Chapter 2, 26-62, Chapter 3, 65-117, 2004.

978-1-4244-4148-8/09/$25.00 ©2009