ABSTRACT

Title of Document:	APPLICATION OF ANT COLONY OPTIMIZATION TO THE ROUTING AND WAVELENGTH ASSIGNMENT PROBLEM
	Eunmi Kim, Masters of Science, August 2007
Directed By:	Associate Professor, Dr. Richard J. La, Department of Electrical and Computer Engineering

The transmission capacity of a link in today's optical networks has increased significantly due to wavelength-division multiplexing (WDM) technology. A unique feature of optical WDM networks is tight coupling between routing and wavelength selection. A lightpath is implemented by selecting a set or sequence of physical links between the source and destination nodes, and reserving a wavelength on each of these links for the lightpath. Thus, in establishing an optical connection we must deal with both routing (selecting a suitable path, i.e., sequence of physical links) and wavelength assignment (allocating an available wavelength for the connection (source-destination pair)). The resulting problem is referred to as the routing and wavelength assignment (RWA) problem, and is significantly more difficult than the routing problem in electrical networks. In this thesis, we offer a new heuristic algorithm to solve the RWA problem using Ant Colony Optimization (ACO) meta-heuristic. An edge disjoint path problem and a partition coloring problem are used to formulate the algorithm.

APPLICATION OF ANT COLONY OPTIMIZATION TO THE ROUTING AND WAVELENGTH ASSIGNMENT PROBLEM

By

Eunmi Kim

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2007

Acknowledgements

I am grateful to Dr. La for his direction, support, patience and boundless enthusiasm throughout all of my research. I am thankful to Holden, my previous intern manger, for being a mentor for last one year. I would also like to thank Professor Makowski for his advice and encouragement for my life. Two years at UMD was my eye-opening time. Thanks to all my friends and lab colleges for sharing their life experience and research guide. Finally I would like to thank my family for their unfailing consideration, and support

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

Optical network based on wavelength-division multiplexing (WDM) technology offer the promise to satisfy the bandwidth requirements of Internet infrastructure, and provide a scalable solution to support the bandwidth needs of future applications. Using WDM technology, an optical fiber link can support multiple non-overlapping wavelength channels, each of which can be operated at high data rate.

If there are enough wavelengths on all fiber links, then every node pair could be connected by an all-optical lightpath (this term will be explained in Section 2.1), and there is no networking problem to solve. However, it should be noted that the network size should be scalable, that transceivers are expensive so that each node may be equipped with only a few of then, and that technological constrains dictate that the number of WDM channels that can be supported in a fiber be limited. Thus, only a limited number of lightpaths may be set up on the network. Therefore reducing the number of wavelength used in the network becomes one key optimization problem in optical network.

Once a set of lightpaths has been chosen or determined, we need to assign a wavelength to it. This is referred to as the routing and wavelength assignment (RWA) problem.

We focus on solving the RWA problem over optical network using a well-known heuristic algorithm, ant colony optimization (ACO). Application of ACO to RWA gives good performance compared with other heuristic algorithms. In this thesis, we represent new ACO algorithm for solving the RWA problem, show results and compare our algorithm with another heuristic algorithm, tabu search.

1.1 Motivation

Many problems from real-world applications require the evaluation of multiple, often conflicting objectives. In such problems, which are called multi-objective optimization problems (MOOPs), the goal becomes to find a solution that gives the best compromise according to the preference of the decision maker between the various objectives.

A lot of MOOPs are NP-hard. For NP-hard problems, if you consider all possible solutions, you can get the optimal solution. But the computational time will be increased exponentially with increasing of problem size. Therefore heuristic algorithms (it will be explained in 4.1) were applied to get good solutions of these problems reducing the computational time at the same time.

Recently, the ACO method which is one of well known heuristic algorithms was applied to solve NP-hard problems and had good performances. ACO was also applied to MOOPs such as vehicle routing problem with window time constraints [19], bi-objective two-machine permutation flow shop problem [20], and bi-objective single machine total tardiness scheduling problem with job changeover costs [21].

The RWA problem in optical network is best formulated as a MOOP. Specifically, one objective of RWA is to reduce the number of wavelengths used while another objective is to minimize the total routing cost, which is referred to in this thesis as total distance of the routes.

We developed new ACO algorithms for solving the RWA problem. We set the main purpose of the RWA as minimization of the number of wavelengths. Once the primary objective is minimized, a secondary objective, which is to minimize the total distance of the routes, is considered. Detailed algorithms will be presented in chapter 5.

1.2 Outline of the thesis

This thesis is organized as follows: Chapter 2 provides background about the RWA problem in optical network. The RWA problem can be decomposed to a routing problem and a wavelength assignment problem. An overview of the RWA problem, basic graph coloring theory, and partition graph coloring will be represented in this chapter. Chapter 3 describes min-RWA problem formulation. Chapter 4 presents an overview of ACO which consists of background of ACO, Simple-ACO and decision rule of ACO with pheromone value and heuristic information value. In chapter 5, we present a new algorithm for solving RWA problem using edge disjoint path, partition coloring. This algorithm is formulated as an application of ACO to MOOPs. Chapter 6 discusses the results from our algorithm. We compare the performance of our algorithm with *onestepCD* algorithm which was introduced in [3] to solve the wavelength assignment problem. We also present a comparison of our algorithm with Tabu Search algorithm which was implemented in [4].

1.3 Related work

The RWA problem is typically partitioned into several smaller sub problems, each of them may be solved independently and efficiently using well-known approximation techniques.

Kleinberg [5] introduced bounded greedy algorithm for the maximum edge disjoint path (EDP). The EDP problem is that of finding the maximum set of edge disjoint paths given a graph and a set of source-destination pairs. Manohar and Manjunath [15] applied this algorithm to solve the RWA problem and obtained better performance compared with the standard greedy algorithm.

There were several ACO approaches to solve the EDP problem. Blesa and Blum [9][10] applied ACO to EDP problem adding new criteria for decision rule of ant. Nowé, Verbeech, and Vrancx[16] used multi-type ant colonies to solve EDP problem. We applied the algorithm in [9] to *EDP_ANT* algorithm to minimize the routing distance.

Another approach to solve the RWA problem is graph coloring. Bannerjee and B. Mukherjee [2] solved wavelength assignment problem based on graph coloring techniques. Li and Simha [3] introduced several algorithms to solve the partitioning coloring problem (PCP), Noronha and Ribeiro [4] applied color degree based algorithm to solve RWA problem.

There are several heuristic methods to solve graph coloring problem such as greedy algorithm, squeaky wheel optimization, simulated annealing and tabu search [12]. ACO is another heuristic method for graph coloring problems. Hertz and Zufferey [16] , Bessedik and Laib [17], Shawe-Taylor and Žerovnik.[18] introduced new ACO algorithms for graph coloring. We built a new ACO algorithm, *OnestepCD_ANT*, based on PCP algorithm introduced in [3].

To combine two ACO algorithms to one, we applied MOOPs approaches to our algorithm introduced in [19] [20] [21]

Chapter 2: Background

In an optical network where there is no wavelength conversion or limited conversion ability one encounters the RWA problem under wavelength continuity constraint which means same wavelength should be allocated on all the links along its route. The RWA problem, which is described in detail in the following subsection, can be either solved on-line or off-line. On-line RWA algorithms are the operational counterpart intended for handling one demand at a time in a dynamic demand arrival/departure environment. An off-line RWA problem assumes that all lightpaths requirements are known and we seek a single overall solution for the assignment of routes and wavelengths to meet lightpath requirements. A typical objective is to serve all requirements using the smallest number of wavelengths in total.

In this thesis, we try to solve the min-RWA (minimizing the number of wavelength) offline variant, in which all connection requirements are known beforehand and one seeks to minimize the total number of wavelengths used for routing these connections.

Since the RWA problem is NP-hard, several heuristic algorithms were introduced to save computational time with good performance.

We propose a new two-phase heuristic for min-RWA, using the same decomposition scheme described in [3]. In each phase, ACO is applied and multi-objective optimization scheme is used to combine two objectives such as routing problem and wavelength assignment problem to one objective which is the RWA problem. Detailed description about the RWA will be provided in Section 2.2.

2.1 Optical Network

Optical networking using wavelength-division multiplexing (WDM) became one of the new solutions for increasing demand of higher transmission bandwidth. In WDM several optical signals using different wavelength can be transferred in a single optical network. Thus the huge capacity of the optical fiber can be used more efficiently. A WDM system uses a multiplexer at the transmitter to join the signal together, and a demultiplexer at the receiver to split them apart. This is illustrated in Fig1.

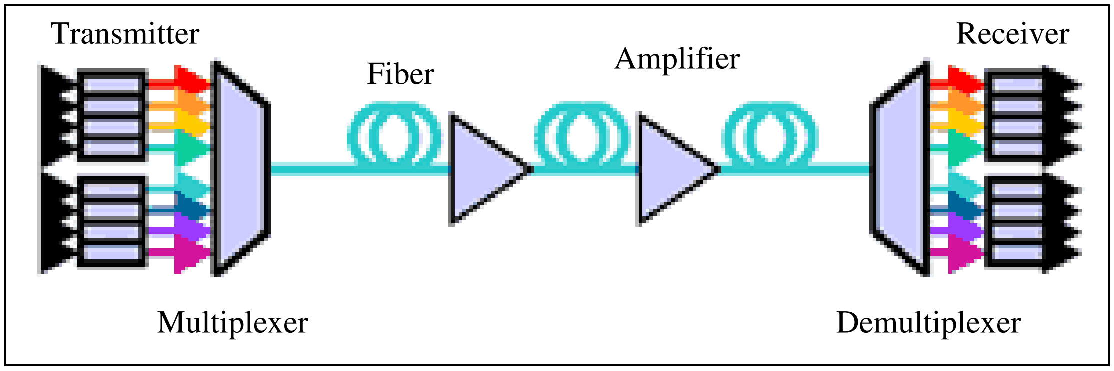

Fig1. Wavelength Division Multiplexing

Access stations communicate with one another via WDM channels, called lightpaths. The term lightpath and connection are used interchangeably, since, in order to establish a "connection" between a source-destination pair, we need to set up a "lightpath" between them.

2.2 Routing and Wavelength Assignment (RWA) problem

Optical networking using wavelength-division multiplexing (WDM) became one of the new solutions for increasing demand of higher transmission bandwidth. In WDM several optical signals using different wavelength can be transferred in a single optical network. Thus the huge capacity of the optical fiber can be used more efficiently.

The formulation of the RWA problem depends on whether wavelength conversion is possible at the nodes or not. If the wavelength conversion is not possible, the same wavelength should be allocated on all the links along its route. This requirement is known as the wavelength-continuity constraint. A feasible solution for this network uses no more than the available wavelengths on each link and no two connections sharing a common link have the same wavelength.

If the wavelength conversion is possible, an optimal solution just minimizes the maximum number of channels which are represented as wavelengths over the links. The problem becomes a routing problem in normal circuit-switched networks with the constraint on the number of channels on each link.

In this thesis, we focus on the RWA problem in WDM networks without wavelength conversion. If two different signals share the same optical fiber, they must use different wavelengths. The cost of an assignment is impacted by the number of wavelengths which are used to deliver all the requirements. While shortest-path routes may be preferable, this choice may have to be sometimes sacrificed, in order to allow more lightpaths to be set up in a wavelength so as to reduce the number of wavelengths used. *Thus we consider the problem of routing a given set of connections in the network with the minimum number of wavelength and minimum route length.*

The RWA problem is partitioned to into several sub-problems, each of which may be solved independently and efficiently using well-known approximation techniques. One is finding a shortest path for all given source-destination pairs. Another problem is assigning a minimum number of wavelengths to route all source-destination pair. This is performed based on graph-coloring techniques**.**

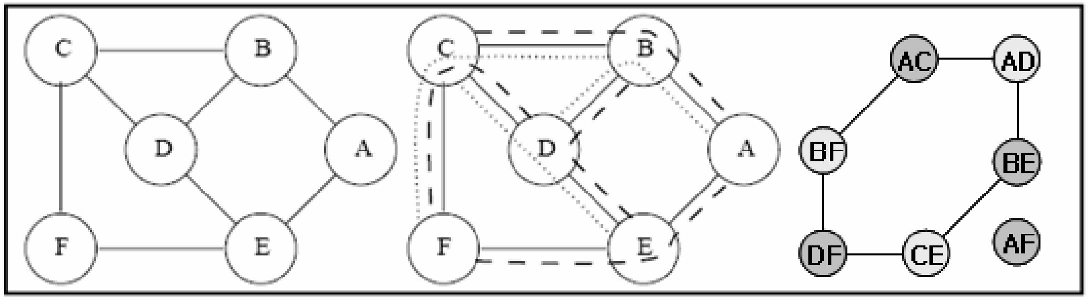

Fig2. Example network and it is optimal configuration

An example of RWA problems is shown in Fig 2. The first graph represents a physical network. The routing is fixed and wavelengths are assigned in the middle graph. The right one is the graph which we need to color. The nodes of the graph represent connections, denoted by source-destination pairs. If two connections share some common link, two nodes in the graph are connected by an edge. Two colors which represent wavelengths are used to build lightpaths for all given connections.

2.3 Routing problem

Lightpaths that are assigned the same wavelength will not traverse the same physical link. Therefore, the routing problem can be recast into one of finding the maximum set of edge disjoint paths (EDP). This problem is well studied and defined as follows. Let $G = (V, E)$ be the graph of the physical network, V the vertex set and E the edge set. Connection request i is specified by a pair (s_i, t_i), $s_i, t_i \in V$. Let τ be the set of connections for whom edge disjoint paths need to be found, $\tau = \{(s_1,t_1),(s_2,t_2), \dots ,(s_k,t_k) \}$. The set τ is said to be realizable in G if there exist mutually edge-disjoint paths in G. The maximum edge disjoint paths problem is to find a maximum size realizable subset of τ. EDP problem is a combinatorial optimization problem and is known to be NP-hard.

2.4 Wavelength assignment problem

Once the routing is fixed, we need to minimize the number of used wavelengths. The problem can be formulated as a graph coloring problem. In the graph coloring problem, each node represents the route of a connection (source-destination pair). If two lightpaths share some links, two nodes that represent the connections are connected by an edge and thus must be colored with different colors. Therefore, the objective becomes to minimize the number of different colors.

As the graph coloring problem is NP-hard, heuristic methods are often employed for a practical solution. Many different heuristic methods, such as Simulated Annealing (SA), Genetic Algorithm (GA), and Tabu Search (TS) algorithm, have been proposed. We apply another heuristic method, called ACO, to the graph-coloring problem.

2.4.1 Color Degree

According to Gould [8], fast standard vertex coloring algorithms use one or more of the following heuristic observations:

- A vertex of high degree is harder to color than a vertex of low degree.
- Vertices adjacent to the same set of vertices should be colored with the same color.
- Coloring as many vertices as possible with a single color is a good idea.

Three of the most well-known heuristics for coloring algorithm are Largest-First (LF), Smallest-Last (SL) and Color-Degree (CD). The Largest-First algorithm orders vertices by decreasing degree, and then greedily assigns the smallest available color to each vertex. A smallest-Last algorithm selects the vertex of lowest degree, removes it and pushes it into a stack. It repeats this process until no vertex remains in the graph. It then proceeds to color the vertices greedily by the order determined by popping from the stack, beginning from the last vertex removed and ending with the first vertex removed. A Color-Degree algorithm is different from the first two: Color-Degree repeatedly chooses a vertex of the largest color-degree, and assigns to it the smallest possible color (the colors are numbered), where the color degree of vertex *v* is defined as the number of colors used so far on vertices adjacent to *v*. If two vertices have the same color-degree, the one adjacent to the most uncolored vertices is chosen first.

For any particular instance of vertex coloring, it is difficult to determine which heuristic performs best. Indeed, no theoretical results are known that establish the superiority of any one heuristic for general graphs. The simulation results done by Li and Simha[3] show that Color-Degree has the best performance on randomly generated graphs. In this thesis, we focus on the application of partition-coloring problem using Color-Degree to the RWA problems in optical network.

2.4.2 Partition Coloring Problem (PCP)

A partition-coloring problem is defined as follows.

- $G = (V, E)$ is a connected undirected graph
- V is partitioned into k disjoint subsets $V_1, ..., V_k$, i.e., $V = V_1 \cup ... \cup V_k$, and

$V_1 \cap V_j = \varnothing$, $\forall i \neq j$, $i, j = 1, \ldots, k$.

- A subset of vertices $V^P \subseteq V$, is called a *feasible subset*, with subset is partition $\{V_1, \ldots, V_P\}$, if $|V^P \cap V_i| = 1$, $\forall i = 1, \ldots, k$.
- The subgraph induced from G by a subset V^P of V is defined $G^P = (V^P, E^P)$, such that $e \in E^P$ if $e \in E$ and the endpoints of e are in V^P.
- $\chi(G)$, the chromatic number of a graph G, is the smallest number of colors needed to color the vertices of G so that no two adjacent vertices share the same color

A partition-coloring problem with respect to a given partition is to find a *feasible subset* V^P of V, such that $\chi(G^P)$ is as small as possible. Observe that we only need to color the *feasible subset* and not the entire set of vertices.

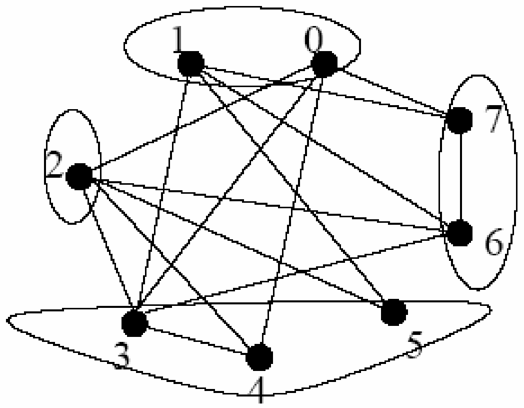

Fig3. Example of Partition Coloring Problem

An example of the PCP is shown in Fig3. The graph $G=(V,E)$ has 8 nodes and 15 edges. The partitions are given as four groups for this example. With our notation, $V_1=\{0,1\}$, $V_2=\{2\}$, $V_3=\{3,4,5\}$, $V_4=\{6,7\}$.

To solve the PCP in the given partition, we should find a *feasible subset*, i.e., pick a node from each partition and color the *feasible subset*. A solution of the PCP in Fig3 is shown in Fig4. The graph $G^P=(V^P,E^P)$ contains 4 nodes and 3 edges. With our notation, $V^P=\{1,2,4,6\}$, $\chi(G^P)$=2.

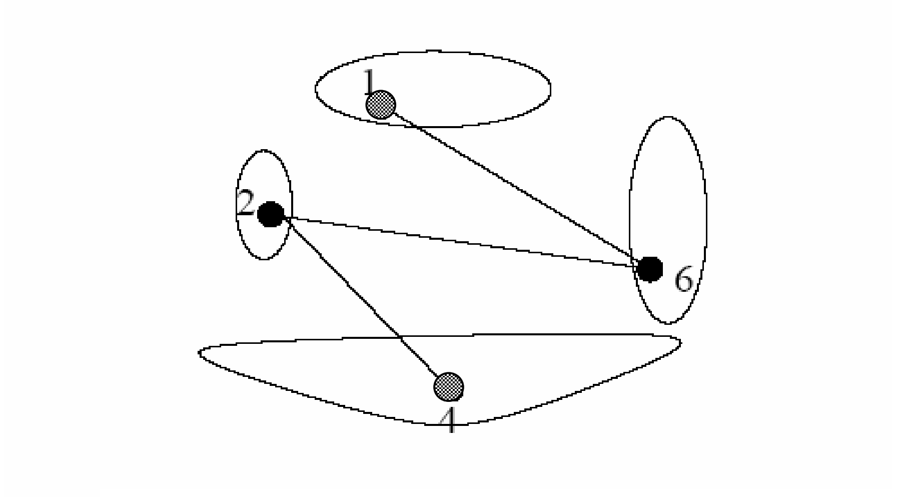

Fig4. A solution of Partition Coloring Problem

Li and Simha [3] proposed six construction algorithms for PCP, based on graph coloring heuristics. Best results were obtained *OnestepCD* (One Step Color Degree) algorithm, which is also used to build *OnestepCD_ANT* algorithm (Section 5.3).

2.4.3 *OnestepCD* algorithm

This algorithm's name derives from *Color-Degree*. First, remove all edges whose vertices are in same group. Then, find the vertex with the minimal color-degree for each uncolored group. If there is a tie, choose the one with the smallest uncolored degree, where the uncolored degree of a vertex is defined as the number of uncolored adjacent vertices so far. From these vertices, find the vertex with the largest color-degree. If there is a tie, choose the one with the largest uncolored degree. Assign this vertex the smallest available color. Remove all other vertices in the same group. Repeat all these process until all groups are colored. Detailed algorithm will be shown in Section 6.2.

2.4.4 Largest-first algorithm

In a sequential graph coloring approach, vertices are sequentially added to the portion of the graph already colored, and new colorings are determined to include each newly adjoined vertex. The coloring of a graph using sequential coloring depends on the order in which the nodes of the graph are colored. At each step, the total number of colors necessary is kept at a minimum if you can give it the optimal order.

If ΔG denotes the maximum degree in a graph, then $\chi(G) \le \Delta(G)+1$. However intuitively, if a graph has only a few nodes of very large degree, then coloring these nodes early will avoid the need for using a large set of colors.

Let G be a graph with vertices $V(G) = v_1, v_2, ..., v_n$, where $\deg(v_i) \ge \deg(v_{i+1})$ for $i = 1, 2, ..., n-1$. Then, $\chi(G) \le \max_{1 \le i < n} \min\{i, 1+\deg(v_i)\}$. Determination of a sequential coloring procedure corresponding to such an ordering will be termed the largest-first algorithm [11].

In *OnestepCD* (subsection 2.4.3), in order to reduce the conflict among vertices, a vertex with the minimal color-degree is first picked from each uncolored group. From these selected vertices, the vertex with the largest color-degree can be colored. The above processes are repeated until all groups are colored. Therefore, the vertices are colored by largest-first algorithm.

2.5 Multi-objective optimization problem (MOOP)

In the world around us it is rare for any problem to concern only a single value or objective. Generally multiple objectives have to be met or optimized to get a solution which is considered adequate. Therefore, the goal of multi-objective optimization problems (MOOPs) becomes to find the best compromise between the objectives. The selection of a compromise solution has to take into account the preferences of the decision maker.

If the decision maker can give weights or priorities to the objectives before solving the problem, then the MOOP can be transformed into a single objective problem. A solution for a MOOP can be formulated as a Pareto-optimal set which is not dominated by any other solution.

Chapter 3: Formulation of the RWA problem

Consider a network $G = (V, E)$. Two nodes may require a connection between them, and a connection is provided by a lightpath using a wavelengtrh. Given a set of connections to be provided, the RWA problem is to select a path and a wavelength for every connection with in such a way that no two lightpaths using the same wavelength traverse the same link. The objective is to minimize the number of required wavelengths and the total routing distance.

First, we introduce the notation to be used throughout.

W	Set of available wavelengths
K	Set of selected pairs of nodes (connections)
P_k	Set of all possible paths for $k \in K$.
p_k	Path for connection $k \in K$
d_e	Distance of edge e
$K(e)$	Set of connections whose path traverses edge e

3.1 Wavelength Assignment Problem

The RWA problem is a bi-objective problem; one objective is to minimize the number of required wavelength and the other objective is to minimize the total route distance. We set the first objective of minimizing the number of required wavelengths as the primary objective of the RWA problem. The problem starts with the given initial routing information, which is related to $K(e)$. We can formulate the RWA problem with the given routing information as following:

$$\text{minimize} \sum_{w \in W} y_w$$

$$\text{s. t.} \sum_{w \in W} x_w{}^k = 1 \text{ for all } k \in K, \tag{3.1}$$

$$\sum_{k \in K(e)} x_w{}^k \le y_w \text{ for all } w \in W, e \in E, \tag{3.2}$$

$$x_w{}^k \in \{0,1\} \text{ and } y_w \in \{0,1\},$$

where $x_w{}^k$ =1 if p_k uses wavelength w for connection k and $x_w{}^k$ =0 otherwise, and y_w = 1 if wavelength w is used and y_w = 0 otherwise. Constraint (3.1) means that every required connection is provided with a lightpath. Constraint (3.2) ensures that at most one path using wavelength w passes edge e. If two paths using same wavelength passed edge e, it cannot be edge disjoint. Constraint (3.2) also ensures that the paths using wavelength w can be selected when wavelength w is used. Assume specific wavelength w used to build a set of connections K'. For an edge e, if the edge is used for the lightpath of connection $k \in K'$, $\sum_{k \in K(e)} x_w{}^k$ =1. By collecting the edges which satisfy $\sum_{k \in K(e)} x_w{}^k$ =1 and by classifying the edges according to k, we can get all the paths using wavelength w.

3.2 Routing Problem

After minimizing the number of wavelengths, we focus on reducing the total routing distance under the given wavelength assignment. We assume W' as the set of wavelengths assigned to the connections. We can formulate the problem as follows:

$$\text{minimize} \sum_{e \in E} \sum_{k \in K(e)} d_e$$

$$\text{s.t.} \sum_{w \in W'} x_w{}^k = 1 \text{ for all } k \in K \qquad (3.3)$$

$$\sum_{k \in K(e)} x_w{}^k \leq 1 \text{ for all } w \in W', e \in E \qquad (3.4)$$

$$p_k \in P_k \text{ and } x_w{}^k \in \{0,1\}$$

where $x_w{}^k$ =1 if p_k uses wavelength w to establish connection k and $x_w{}^k$ =0 otherwise, and y_w = 1 if wavelength w is used and y_w = 0 otherwise. The purpose of the routing problem is to find a set of paths for the given connections with the smallest total routing distance subject to the given wavelength constraint. Constraint (3.3) means every wavelength in W' must be assigned to build all the required connection; i.e., the wavelength used to build new lightpath for a connection will be fixed. Constraint (3.4) ensures that the paths using the same wavelength w should be edge disjoint, i.e., any two paths using the same wavelength cannot share an edge.

3.3 Further Minimization of the number of wavelengths

After the minimizing process in section 3.1 is performed, new routing information is given for all required connections. The total number of edges used for new routing may be smaller than previous routing. With this new routing, the number of required wavelengths can be reduced more by performing the minimizing process in section 3.1 again under constraint of W' instead of W; i.e., the number of required wavelengths can be the number of elements in W' or less.

Chapter 4: Ant Colony Optimization Overview

The field of “ant algorithms” studies models derived from the observation of real ants’ behavior. These models used[22][24][25][26] as a source of the design of new algorithms to solve optimization and distributed control problems.

One of the most successful examples of ant algorithms is known as ACO. ACO is a biology inspired meta-heuristic for solving combinatorial optimization problem. An ACO algorithm is composed of independently operating computational units, namely artificial ants, which generate a global perspective without the necessity of direct interaction.

ACO is inspired by the foraging behavior of real ants. While moving between food sources and the nest, ants deposit a chemical substance called pheromone along their path. Pheromone attracts more ants to choose the path of larger pheromone strength.

In ACO algorithms, artificial ants incrementally construct a solution following a decision rule which is affected by pheromone and heuristic values.

4.1 Heuristic algorithm and Meta-heuristic

The term heuristic is used for algorithms that find solutions among all possible ones, but they do not guarantee that a best solution will be found. Therefore, they may be considered as not accurate algorithms. These algorithms usually find a solution close to the optimal solution, but there is no proof the solutions could not get arbitrarily bad; or it usually runs reasonably quickly, but there is no argument that this will always be the case.

A meta-heuristic is a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems. That is, a meta-heuristic can be seen as a general-purpose heuristic method designed to guide an underlying problem-specific heuristic towards promising regions of the search space containing high-quality solutions.

An ACO algorithm is an example of meta-heuristic which uses a single agent, called artificial ant or ant for short. In ACO, artificial ants build solutions by interacting locally with one another using pheromone in such a way that future artificial ants can build better solutions. Ant Algorithms are one of the successful examples of swarm intelligence system.

4.2 Ant Colony Optimization (ACO) meta-heuristic

ACO algorithms have been inspired by Double Bridge experiments run by Goss et al. [6] using real ants. (see Fig5) Ants deposit a chemical substance, called pheromone, on the ground when they travel between the nest and the food source. Without distortion by pheromone, ants will choose paths with equal probability. In this experiment, due to different branch lengths, the ants choosing a shorter path will be the first to reach the food source (under ideal settings). When the ants reach the decision point, they will see some pheromone trail which they released during the forward travel on the shorter path. Therefore, a shorter path will have a higher probability to be chosen by the ants than the longer one. That leads more pheromone to be deposited on the shorter path by ants. When this process repeats, higher pheromone strength makes the shorter path more attractive for subsequent ants until most ants end up using it. Interestingly, by taking inspiration from the double bridge experiments, it is possible to design artificial ants that, by moving on a graph modeling the double bridge, find the shortest path between two nodes corresponding to the nest and the food source.

ACO algorithm can be implemented by three procedures: *ConstructAntsSolutions*, *UpdatePheromones*, and *DaemonActions*. *ConstructAntsSolutions* manages a colony of ants to build solutions by applying a stochastic local decision policy that makes use of pheromone tails and heuristic information. *UpdatePheromones* is the process by which the pheromone trails are modified. The trails value can either increase, as ants deposit pheromone on the components or connections they use, or decrease, due to pheromone evaporation. *DaemonActions* procedure is used to implement centralized actions which cannot be performed by single ants. It can be a local optimization procedure or the collection of global information. The best solution update is an example of *DaemonActions* procedure.

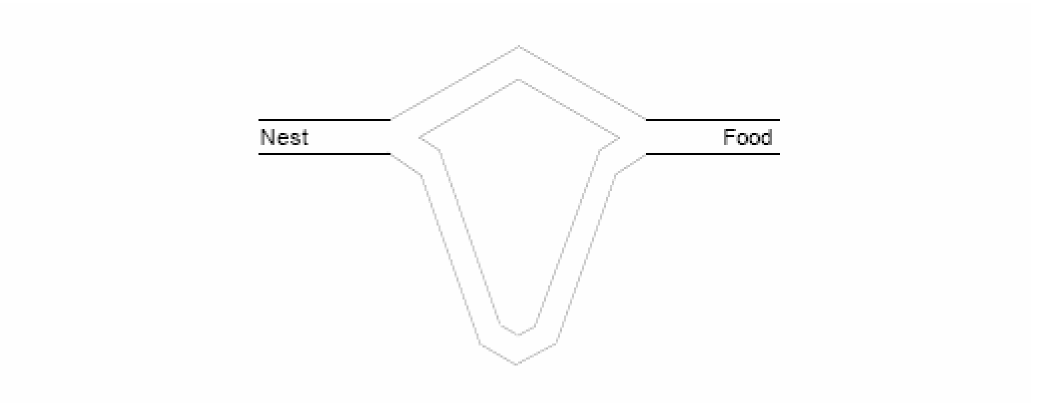

Fig5. Double bridge experiments

4.3 Simple-ACO (S-ACO) algorithm

In this section a very simple ant-based algorithm is presented to illustrate the basic behavior of the ACO meta-heuristic. Let $G = (N, E)$ be a connected graph, N_i is the set of one-step neighbors of node i. One-step neighbor of node i is the node which is connected with node i with an edge. The simple ant colony optimization (S-ACO) algorithm can be used to find a solution to the shortest path problem defined on the graph G. Artificial ants move on the graph by choosing the path probabilistically. Each ant constructs a solution step-by-step using a decision rule. The decision rule of an ant k located in node i uses the pheromone trail τ_{ij} to compute the probability with which it should choose node $j \in N_i$ as the next node to move to:

$$P_{ij}^{k} = \begin{cases} \tau_{ij} & \text{if } j \in N_i \\ 0 & \text{if } j \notin N_i \end{cases}$$

While building a solution ants deposit pheromone information on the edges they use. In S-ACO ants deposit a constant amount $\Delta\tau$ of pheromone. Constructed solution will be only dependent on pheromone value which was deposited by previous ants.

To avoid a quick convergence of all the ants towards a sub-optimal path, an exploration mechanism is added: Similarly to real pheromone trails, artificial pheromone trails "evaporate". In this way pheromone intensity decreases automatically, favoring the exploration of different edges during the search process.

S-ACO has a number of limitations because of its simplicity. Experiments have shown that if we increase the complexity of the search graph, the behavior of the algorithm becomes less stable and the value given to parameters become critical.

The ACO meta-heuristic is built on the S-ACO model enriching artificial ants with extra capabilities which help to overcome above-mentioned S-ACO limitations.

4.4 Pheromone and Decision rule

When artificial ants construct a solution, they use a decision rule for moving to next step. In ACO, typically a *random proportional rule* is used as the decision rule using a heuristic value and a pheromone value.

4.4.1 Decision rule

When an ant k build a solution, ant k applies a probabilistic action choice rule, called *random proportional rule*. The probability with which ant k, currently at node i , chooses to go to node j is

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}, \text{ if } j \in N_i^k, \qquad (4.1)$$

where η_{ij} is a heuristic value and τ_{ij} is a pheromone value. Parameters α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information. N_i^k is the feasible neighborhood of ant *k* which ant *k* in node *i* possibly moves to. By this probabilistic rule, the probability of choosing a particular move from node *i* to node *j* increases with the value of the associated pheromone trail τ_{ij} and of the heuristic information value η_{ij}.

In Another parameter for the decision rule is q_0. With probability q_0, where $0 \le q_0 \le 1$, an ant will choose node *j* which maximizes $[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}$. With probability 1-$q_0$ the node is chosen according to the random proportional rule. This is called a *pseudorandom proportional* rule, given by

$$j = \begin{cases} \arg\max_{l \in N_i^k} [\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}, & \text{if } q \le q_0 \\ J, & \text{otherwise} \end{cases}$$

where *J* is random variable selected according to the probability distribution given by equation (4.1).

4.4.2 Heuristic value

The heuristic value, often called heuristic information value, represents a priori information about the problem instance or run-time information provided by a source different from the ants. In many cases the heuristic value is the cost, or an estimate of the cost, of adding the component or connection to the solution under construction. These values are used by ants' heuristic rule to make probabilistic decisions on how to move on a graph.

Typically the heuristic value is set by a greedy algorithm. For example, η_{ij} for TSP is set to $1/d_{ij}$ when shorter path is preferred, where d_{ij} is the distance between city i and city j

4.4.3 Pheromone update

Making pheromone update can help in directing future ants more strongly toward better solutions. In fact, by letting ants deposit a larger amount of pheromone on short paths, the ants' path selection is more quickly biased toward the best solutions.

Pheromone trail evaporation can be seen as an exploration mechanism that avoids quick convergence toward a suboptimal path. The decrease in pheromone intensity favors the exploration of different paths during the whole search process. Pheromone evaporation reduces the influence of the pheromone deposited in the early stage of the search, when the ants can build poor-quality solutions.

4.4.3.1 Global pheromone update for intensification

Intensification is an expression commonly used for the concentration of the search process on areas in the search space with good quality solutions. The ant which constructed the best solution so far can have a chance to update pheromone to attract more ants to the solution. This Pheromone update process is called "global pheromone update". In Ant Colony System (ACS) introduces by Dorigo and

Gambardella [22] only one ant (the best-so-far) is allowed to add pheromone value. The global pheromone update in ACS is implemented by the following equation:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \quad \forall(i,j) \in T^{bs}, \tag{4.2}$$

where $\Delta\tau_{ij}^{bs}$ is the amount of pheromone value to be updated for best solution, T^{bs} is the best solution constructed by an ant and ρ, 0<ρ<1. The deposited pheromone is discounted by a factor ρ; this results in the new pheromone trail being a weighted average between the old pheromone value and the amount of pheromone deposited. When the arc (*i, j*) belongs to the best solution, a global pheromone update will be done for the arc. Typically the value of $\Delta\tau_{ij}^{bs}$ is *1/cost*. When the cost of the best solution is smaller than other solutions, *1/cost* will be bigger for the best solution so that the pheromone value to be deposited will attract following ants to the best solution.

4.4.3.2 Local pheromone update for diversification

Diversification refers to the action of leaving already explored areas and moving the search process to unexplored areas. In ACS, the ants use a local pheromone update rule that they apply immediately after having crossed an arc (*i, j*) during the solution construction. This process gives diversification to the ants not to converge to the previous solutions. The update is implemented by the following equation:

$$\tau_{ij} \leftarrow (1-\varepsilon)\tau_{ij} + \varepsilon\tau_0, \tag{4.3}$$

where ε, $0<\varepsilon<1$, and τ_0 are two parameters. Parameter ε is decreasing factor which decreases the original pheromone by a factor of $(1-\varepsilon)$. The value of τ_0 is set to be the same as the initial value for the pheromone trails.

The effect of the local updating rule is that each time an ant uses an arc (*i, j*) its pheromone trail τ_{ij} is reduced, so that the arc becomes less desirable for the following ants. In other words, this allows an increase in the exploration of arcs that have not been visited yet, and in practice, has the effect that the algorithm does not show a stagnation behavior (i.e., ants do not converge to the generation of a common path).

4.5 ACO algorithm and Multi-Objective Optimization Problems (MOOPs)

There are several ACO approaches for solving MOOPs. The first application of ACO to solve the MOOPs is to give priority to the objectives. There were several approaches for optimizing prioritized objectives using multiple colonies of ants. Each colony may share a pheromone value or the best solution. Therefore, ants which belong to different colonies can communicate with each other using this shared information when they focus on optimizing their own objective.

Another application of ACO to solve the MOOPs is to combine the two objective functions into a single one using a weighted sum. A solution using different population of ants for each objective was introduced for weighted sum MOOP as well as multi-colony approach. These ants communicate with each other by sharing pheromone values or the best solution. The pheromone value induces the ants to follow the path or to explore other paths, global pheromone update is done when the ant build better solution than the best solution.

ACO also applied to approximate the set of Pareto-optimal solutions. For this application, the best solutions are given by a set. Therefore, an update of the best solution by ants will contain the process of comparison between a current solution and all the best solutions so far. Multi-colony approach is introduced in [21] for this application.

In this thesis, we prioritize two objectives and apply ACO using two different ant colonies to solve routing and wavelength assignment problem.

Chapter 5: Algorithm

In the following, we give a high-level description of an ACO algorithm for the RWA problem (see Algorithm 1). The RWA problem is an MOOP. The primary goal of the algorithm is to minimize the number of wavelengths used, and minimizing the total route length is a secondary goal. The main procedures used by the algorithm are explained in detail in the following section.

First, initial *K* alternative routing solutions for all given connections are constructed by *K-EDR* (described in Section 5.1). Then, a conflict graph (Section 5.2) will be built using the routing information which was given by the *K-EDR* process. Each node in this graph represents a connection. *OnestepCD_ANT* (Section 5.3) will color using this conflict graph to get the smallest number of colors, assigning a color to each connection. For all connections, a route and a color will be given after *OnestepCD_ANT*. *EDP_ANT* (Section 5.4) will then focus on minimizing the total route length for each color, which corresponds to a wavelength used for the routes. The connections assigned to the same color will be conflict-free by construction. The main purpose of *EDP_ANT* is to reduce the total route length of the connections with the same color without creating a conflict. The solution from *EDP_ANT* will be the input of *OnestepCD_ANT* to fulfill the primary goal.

```
Algorithm 1 : RWA_ANT (G,T)
Input : Network graph (G) and given connections (T) .

rwa = K_EDR(G,T,K)
CG = MakeConflictGraph(G, rwa)
I_Sol = MakeInitialSolution(rwa)
Sol1 = OnestepCD_ANT(CG, I_Sol)
rwa = EDP_ANT(G,Sol1)
CG = MakeConflictGraph(G, rwa)
Sol2 = OnestepCD_ANT(CG, Sol1)
Return Sol2

Output : routings and assigned color(wave length) for all connections in T.
```

5.1 K_EDR algorithm

This algorithm was proposed by Kleinberg [5] for the maximum edge disjoint path problem, which finds the largest subset of connections that can be routed by edge-disjoint paths with a maximum route length d . Routing with as many disjoint paths as possible enables effective use of the available wavelengths in the optical network. The pseudo-code of the construction procedure of *K-EDR* is given in Algorithm 2.

```
***Algorithm 2  K-EDR (N,T,k)***
*Input : Network graph (N) ,gven connections (T) and the number of alternative routes per connection (k)*

1. Set $d \leftarrow max(diameter(\mathrm{N}), \sqrt{|A|})$;
2. Set $R \leftarrow \varnothing$ ;
3. for $j = 1,\ldots,k$ do
4.     Set $T' \leftarrow T$;
5.     while $T' \neq 0$ do
6.         Set $N' \leftarrow N$;
7.         Build a list L with a random permutation of the connections in $T'$;
8.         for $i = 1, \ldots, |T'|$ do
9.             Let $t$ be the $i$-$th$ connection in L;
10.            Compute the shortest path s in N' between the end nodes of t;
11.            If $s$ has no more than $d$ edges
12.            Then do
13.                $T' \leftarrow T' \setminus \{t\}$;
14.                $R \leftarrow R \cup \{(t,s)\}$;
15.                Remove the edges in path s from $N'$;
16.            end-if
17.        end-for
18.    end-while
19. end-for
20. Return $R$;

*Output : routings and assigned color(wave length) for all connections in T.*
```

Let $N = (X, A)$ be a graph representing the physical topology of an optical network, where X is the set of nodes and A is the set of links (edges) connecting nodes. We denote by T the set of connections to be established. Each connection is defined by a pairs of end nodes in X (source-destination pair).

For *K-EDR* algorithm, network graph (*N*), a set of connections (*T*) and the number of alternative routes per connection (*K*) are given. The maximum number of d edges in each route is initialized first as suggested in [5]. The set *R* of routes are initialized in line 2. The loop in lines 3-19 attempts to build at most *K* alternative routes for each connection in *T*. A New list of connections (T') is made in line 4. T' originally contains all the elements in *T*. Whenever a route is given for a connection, the connection is removed from the list T'. The loop in lines 5-18 computes a single route for each connection. A new network N' is built in line 6. Originally N' is a copy of the network *N*. Whenever a route is given for a connection, the edges in the route is removed from N'. In line 7 we build a list *L* with a random permutation of the connections in T'. The loop in lines 8-17 finds a set of disjoint routes with at most d edges for as many connections in T' as possible. The next connection t to be investigated is selected in line 9. The shortest path s in the network N' between the end nodes of connection t is computed in line 10. Line 11 checks if this path has at most d edges. In this case, the set T' is updated in line 13, The set of routes is updated in line 14, and all the edges used in the routes are removed from N' in line 15. The set *R* of routes is returned in line 20. We followed the implementation of this algorithm as introduced in [4].

5.2 Conflict graph

A conflict graph can be built using given routing information from *K-EDR*. A node of the graph represents a route. There are *K* alternative routes for one source-destination pair which represents a connection. Therefore *K* nodes which have same source-destination pair can form a group. It means |*T*| groups will be generated for the conflict graph. The edge of the conflict graph represents the confliction. If there are shared links between two routes, an edge will be set up between them. In *OnestepCD_ANT* algorithm, this conflict graph will be used to select a candidate set for coloring which contains one node from each group. Any two nodes in a same group cannot be in a candidate set. That is, the confliction among nodes in a same group is not considered when a candidate set is built. Therefore, the edges between the nodes in a same group are removed for building a conflict graph. This conflict graph will be the input for *OnestepCD_ANT* process.

5.3 OnestepCD_ANT

After building a conflict graph, ants choose a route for each connection, assigning a wavelength to it by selecting a node from each group and coloring it. The process consists of two steps. In first step, each ant builds a solution. Second step contains global pheromone update and best solution update. They are done by comparing the solution which is built by an ant with the best solution so far.

An ant with color (wavelength) c finds a node to color by using pheromone value and heuristic value until there is no possible node colored with c. Whenever the ant finds better solution than best solution so far, the ant does the global pheromone update and the best solution is updated. Detailed algorithm of

OnestepCD_ANT will be explained in algorithm 3.

```
Algorithm 3. OnestepCD_ANT(CG, Sol)
Input:  Conflict Graph (CG), initial solution (Sol)

1. Calculate prev_cost with initial solution Sol.
2. Initialize Pheromone.
3. for MAX_ITER
4.     Make a candidate set using color degree and edge degree.
5.     Set all different colors for candidate set
6.     Put candidate set to the LIST
7.     while (TABU_LIST.size() != CONNECTION)
8.             LOC = first element in the LIST.
9.             Put LOC to TABU_LIST
10.            while(available next_node)
11.                    Find next_node to color with Heuristic value and Pheromone.
12.                    Color next_node and Put next_node to TABU_LIST
13.                    Local Pheromone Update
14.            end-while
15.    end-while
16.    Calculate cost for current solution
17.            if prev_cost ≥ cost
18.                    Global Pheromone Update
19.                    Update Best Solution
20.                    Apply solution to the pool for Partition Coloring.
21.            end-if
22.    Clear TABU_LIST, LIST
23. end-for
23. Return Sol

Output: routings for given connections with wavelength number
```

In line 1, *prev_cost* is the total number of wavelengths used to build an initial solution. An initial coloring solution is given by *K_EDR* process when *K* is 1. We denote τ_{ij} as the pheromone value between node i and node j of the conflict graph (*CG*). In line 2, τ_{ij} is initialized to τ_0 that is defined by the value $1/\mathit{prev\_cost}$ if there is no edge, i.e. confliction, between node i and node j of the conflict graph. If there is an edge (confliction) between node i and node j, τ_{ij} is initialized to 0 so that the ant which is located in node i will never choose node j.

In the loop in lines 3-23, *OnestepCD_ANT* attempts to build a solution. In line 4, a candidate set is built. For finding a candidate set, a group in the conflict graph will be selected with randomly permutated order. Then, the color degree and edge degree of the nodes in the selected group will be calculated. Only one node which has the smallest color degree will be chosen from the group for the candidate set. If there is a tie, the edge degrees of the nodes will be considered. If two or more nodes have the same edge and color degrees, one of the nodes will be randomly selected. After a node is selected from the group, all the edges with other nodes in this group will be removed from the conflict graph. This process will be repeated $|T|$ times to find a candidate set consisting of $|T|$ nodes, one from each group. Whenever a node is chosen from a group, the node will be inserted to *LIST*. Therefore *LIST* contains exactly one node from each group with the order of the group selected.

Once a candidate set is found, *OnestepCD_ANT* starts coloring the set**.** The elements in the candidate set will be the initial solution for coloring. Before starting coloring, a different color will be assigned to each node in the candidate set to avoid any conflict. The loop in line 7-15 assigns a color to each node in the candidate set. In line 8, *OnestepCD_ANT* will be located in the first node in *LIST*. The node becomes *prev_node* and an ant tries to find the *next_node* which can be colored with the color of *prev_node*. Once the ant colors *next_node*, *next_node* becomes *prev_node* and does coloring until there is no *next_node* which can be colored with the color of *prev_node*. *TABU_LIST* is used to prevent to color same node twice. The ant uses *pseudorandom proportional rule* (explained in subsection 4.4.3) following:

$$P_{ij}^{k} = \frac{[\eta_{ij}]^{\alpha}[\tau_{ij}]^{\beta}}{\sum_{l \in N_i} [\eta_{il}]^{\alpha}[\tau_{il}]^{\beta}}, \text{ if } j \in N_i^{k}, \tag{5.1}$$

where η_{ij} is the heuristic value, τ_{ij} is the pheromone value and N_i is possible neighbors for node *i*. η_{ij} is set to the following value.

$$\eta_{ij} = \frac{[color\_\deg ree(node_j)]^{\omega}[edge\_\deg ree(node_j)]}{\sum_{l \in N_i} [color\_\deg ree(node_l)]^{\omega}[edge\_\deg ree(node_l)]}, \tag{5.2}$$

where $color\_\deg ree(node_j)$ is the number of different colors of neighbors of node *j* and $edge\_\deg ree(node_j)$ is the number of edges which are connected to node *j*. Parameter ω is set for giving weight to the color degree.

N_i contains all the nodes which have no confliction with node i .If there is an edge between node i and node j , node j cannot be included in N_i . Nodes that already colored by an ant also should not be in N_i .

Whenever an ant finds *next_node* to color, the ant does local pheromone update for pheromone evaporation as follows:

$$\tau_{ij} = (1-\varepsilon)\tau_{ij} + \varepsilon\tau_0 \tag{5.3}$$

Once an ant colors a node, the node will be removed from *LIST* and added to *TABU_LIST*. If an ant cannot find *next_node* to color, the ant will be located in a randomly picked node from remaining elements in *LIST* and repeat coloring until there is no remaining element in *LIST*.

Once an ant builds a solution, the cost of the solution will be compared with the cost of the best solution so far. If the cost of the new solution is smaller than that of the previous best solution, the best solution will be replaced with the new solution and the ant will do the global pheromone update with this solution to attract other ants to this solution more. Global pheromone will be updated as follows:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}, \qquad (5.4)$$

where $\Delta\tau_{ij}$ is *1/cost* and *cost* is the number of the colors used by the best solution.

Other ants will construct their own solutions using the updated pheromone value. If the best solution is updated, coloring information of the nodes is updated. Then, the color degree of the nodes may be changed. If there is no change in color degree of the nodes, making a candidate set is only dependent on the order of group selection. Therefore, change in color degree of the nodes gives more diversity when making a candidate set. The whole process will be repeated for *MAX_ITER* times. The output of *OnestepCD_ANT* is the best solution so far. That will be the input for *EDP_ANT.*

5.4 EDP_ANT

While *OnestepCD_ANT* focuses only on reducing the number of colors used for the candidate set, *EDP_ANT* focuses on minimizing the total routing distance of the given candidate set. The result of *OnestepCD_ANT* process contains routes for all the given connections and coloring information. The routes which were assigned to the same color are edge disjoint. The main idea of *EDP_ANT* is to reduce the total routing distance for the given lightpaths which were assigned to the same color. In *EDP_ANT*, an ant finds a route for a given connection using pheromone value, heuristic value and pheromone values of other connections so that total route length is minimized. The algorithm for *EDP_ANT* is represented in *Algorithm 4*.

Algorithm 4. EDP_ANT(G, sol)

Input: Networt Graph (G), initial solution (Sol)

```
1. Set best solution with initial solution (Sol)
2. Initialize pheromones for all connections.
3. while(!LIST.empty())
4.      wave_num = color of the first element in the list
5.      WAVE_LIST ← connections (src,dest) with wave_num for color
6.      prev_total_length ← total routing length of connections in WAVE_LIST
7.      wave_list_size = sizeof(WAVE_LIST)
8.      for MAX_ITER
9.           for wave_list_size
10.          Set trial←1;
11.              (src,dest) = randomly picked one connection from WAVE_LIST
12.                 route.push(src);
13.                 find_next_node(src) using pheromone value and heuristic value.
14.                 while (next!=dst)
15.                      Route.push(next)
16.                      find_next_node(next);
17.                      Local pheromone update;
```

```
18.                     if(route_length>d)
20.                         trial←trial+1;
21.                         if(trial>3)
22.                             Restore the route from the best solution
23.                             goto line 9 ;
24.                         end_if

25.                         if(next = dst)
26.                             route.push(dst)
27.                         end_if
28.                     end_if
29.                 end_while
30.     end_for
31.         Calculate total_length from routes in WAVE_LIST
32.         if(total_length <= prev_total_length)
33.             prev_total_length ← total_length
34.             Update the best solution
```

In lines 1-2, pheromone values and the best solution are initialized. For the EDP_ANT process, all connections have their own pheromone matrix. τ_{ij} denotes that the pheromone value between node i and node node j of the network graph. If an ant across the edge (i, j) in the network graph to route a connection, τ_{ij} in the pheromone matrix of the connection will be updated. The initial pheromone value for each connection is set to 1.

All connections will be divided into groups according to their colors. Therefore, *WAVE_LIST* in line 5 only contains connections with the same color.

EDP_ANT focuses on reducing the total route length of the connections in *WAVE_LIST* with edge disjoint constraint. Before constructing a solution, In line 6, *prev_total_length,* total sum of routing distances, is calculated using the given routing information of connections in *WAVE_LIST.*

The loop in 9-30 builds a solution of edge disjoint routes for all connections in *WAVE_LIST*. It starts with a connection which is randomly picked from *WAVE_LIST.* The connection contains the information about the source and the destination. The route for the connection starts from the source node in the network graph in line 12. In line 13, an ant finds a next node to route the connection in the network graph until it meets the destination. Ant *k* uses the *pseudorandom proportional rule* to find the next node as follows:

$$P_{ij}^{k} = \frac{[\eta_{ij}]^{\alpha}[\tau_{ij}]^{\beta}[\phi_{ij}]^{\gamma}}{\sum_{l \in N_i}[\eta_{il}]^{\alpha}[\tau_{il}]^{\beta}[\phi_{il}]^{\gamma}}, \tag{5.5}$$

where η_{ij} is the heuristic value, τ_{ij} is the pheromone value and ϕ_{ij} will be described below. Parameters α, β and γ determine the relative influence of the pheromone trail, the heuristic information and the routing information of other connections. This routing algorithm is also based on the shortest path algorithm. Thus, shorter path is preferable. This information is applied in the heuristic value and affects ants' decision. The heuristic value η_{ij} is $1/d_{ij},$ where d_{ij} is the distance between node *i* and node *j*. This value induces ants to the edge with smaller length. The value of ϕ_{ij} contains the information which was used in the routing for other connections in *WAVE_LENGTH.* If the edge *(i,j)* in the network was never used before, ϕ_{ij} will be set to 1. If edge *(i,j)* was used in routing for other connections, ϕ_{ij} will be set as follows:

$$\phi_{ij} = \frac{1}{\sum_{l \in WAVE\_LIST} \tau_{lij} \psi_{lij}}, \tag{5.6}$$

where τ_{lij} is the pheromone value for the connection *l*. ψ_{lij} indicates whether the connection *l* uses the edge *(i,j)* of the network or not. If the connection *l* uses the edge *(i,j)*, ψ_{lij} will be set to 1. Otherwise it will be set to 0. If the edge *(i,j)* was used a lot for routing other connections in *WAVE_LIST*, ϕ_{ij} becomes smaller since the denominator becomes bigger. Then ϕ_{ij} naturally prevents the ant from taking the edge *(i,j)* when the ant finds a route for a given connection. ϕ_{ij} is affected by τ_0 since τ_{lij} is bigger than τ_0. In this algorithm, τ_0 is assigned to 1 to make ϕ_{ij} have a value in the range between 0 and 1 when the edge *(i,j)* was used for routing other connections in *WAVE_LIST.*

In the loop in lines 14-29, an ant attempts to build a route for the given connection. An ant finds a next node to route the connection in the network graph until it meets the destination. For the maximum edge disjoint path problem, *K-EDR* algorithm finds the largest subset of connection that can be routed by edge-disjoint paths with a maximum number *d* of edges. This algorithm is applied to *EDP_ANT* algorithm. If the route length is bigger than *d,* an ant tries up to three times to make another route for the given connection. If the ant fails to find a proper route after third trial, the route for the connection is set to the route stored in the best solution.

Whenever an ant crosses an edge, a local pheromone update will occur to make other ants less desirable to the edge. Once an ant finishes building routes for all connections in *WAVE_LIST*, the ant will calculate the *total route length*. If the *total route length* is less than *prev_total_length* and the routes do not break edge disjoint constraints, the best solution will be updated with the current solution which was built by the ant. Global update will be following:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} \ , \qquad (5.7)$$

where $\Delta\tau_{ij}$ is *1/cost*, and *cost* is average route length for the connections in *WAVE_LIST*.

Once optimization of the total route length of the connections in *WAVE_LIST* is finished, all elements in *WAVE_LIST* will be removed from *LIST* and *WAVE_LIST* will be cleared for being a new list which contains other connections with the same color extracted from LIST. It will be repeated until no element is left in *LIST*.

If routes of the given initial solution are changed, original conflict graph will be changed since used edges for the routes are different. Therefore, a new conflict graph should be built after *EDP_ANT* process.

<u>*5.5 More Optimization*</u>

Two different ant colonies, *OnestepCD_ANT* and *EDP_ANT,* cooperate to solve the two-objective RWA problem sharing the best solution together. Once a new conflict graph is built and a new initial solution is given after *EDP_ANT*, *OnestepCD_ANT* may be able to reduce the number of wavelengths further. If total route length is minimized by applying *EDP_ANT*, total number of links used for routing may decrease. If total number of links used for routing decreases, the confliction which means existence of shared links between two nodes in the conflict graph can be reduced. *OnestepCD_ANT* focuses on minimizing the number of wavelengths again with this new conflict graph, may get better solution. Therefore, *EDP_ANT* can contribute to reduce the number of wavelengths as well as total route length.

Chapter 6: Results

Algorithms *K-EDR*, *OnestepCD_ANT* and *EDP_ANT* described in the previous section were implemented in C++, performed on a Pentium IV machine with a 1.8Ghz clock and 512 Mbytes of Ram memory. Numerical results are shown in following sections.

6.1 K-EDR vs. OnestepCD_ANT

After *K-EDR* process (see *Algorithm 2*), *K* possible solutions will be given for the RWA problem. Each solution contains a route and an assigned wavelength (color) for each connection. *OnestepCD_ANT* algorithm (see *Algorithm 3*) selects a route for each connection from *K* given routes which minimizes the number of colors used to set up the given set of connections. In the following section, performance improvement after applying *OnestepCD_ANT* process will be described.

6.1.1 Parameter setting for *OnestepCD_ANT* algorithm

Typically performance of ACO algorithms is dependent on parameter settings. First parameters to be considered are α and β in (5.1) for the decision rule. Parameters α and β determine the relative influence of the pheromone trail and the heuristic information. In this simulation, parameter α is set to 1 and β is set to 2 to give more weight to heuristic information values for ants' decision.

Another parameter to be set is ω which is used for heuristic information in (5.2). This parameter gives more weight to the color degree over the edge degree when ants select next node to color. This idea comes from *OnestepCD* algorithm which is described in section 2.3.2.1. In *OnestepCD* algorithm, the color degree is considered to select the next node to color. If there is a tie in the color degree, the edge degree of a node is considered. Therefore, we give larger weight to the color degree over the edge degree so that the color degree is considered more for ants' decision. In the simulation parameter ω is set to 2 to give slightly more weight to the color degree compared with the edge degree for the contribution to heuristic information.

Other parameters which influence to performance of ACO algorithms are parameters used for pheromone updates. For local pheromone updates, parameter ε is decreasing factor which makes used arc (i, j) less desirable for the following ants and experimentally set to 0.15. Another parameter for pheromone update is ρ which is used for global pheromone update. In (5.3) the deposited pheromone is discounted by a factor ρ. The new pheromone trail value is given by the weighted average between the old pheromone value and the amount of pheromone updated ($\Delta\tau_{ij}$). Experimentally, it is set to 0.3.

The initial pheromone value τ_0 is also important to get good performance. If the initial pheromone value τ_0 is too low, then the search is quickly biased by the first tours generated by the ants, which in general leads toward the exploration of the search space. On the other side, if the initial pheromone value is too high, then many iterations are lost waiting until pheromone evaporation reduces pheromone values enough, so that pheromone added by ants can start to bias the search. In this simulation, τ_0 is set to *1/cost* where *cost* is the number of colors used for the initial solution which is given by *K-EDR* process when *K* is 1.

For the *pseudorandom proportional rule*, parameter q_0 is set to 0.25. The number of iterations, *MAX_ITER*, is set to 100.

6.1.2 Performance Improvement

We apply our OnestepCD_ACO to the NSFNET topology shown in Fig.6, which has 14 nodes and 21 edges. Each edge represents a fiber which can carry one signal in one direction. To construct *K* alternative routes for each connection (source-destination pair), we use the *K*-shortest paths between each source and destination of the connection. When *K* is 1, the routing reduces to the shortest path routing, and we use standard vertex-coloring to compute the number of wavelengths required; when k>1, we use partition-coloring algorithms to compute the number of wavelengths.

In this example, we randomly create a set of connections and compute the *K* -shortest paths for each connection, where *K*=1,2. For each *K*, we construct path-conflict graphs (Section 5.2) and apply *OnestepCD*_ACO algorithm on these path-conflict graphs to compute the number of wavelengths required and take the best result. Each instance is evaluated with 20 simulation runs and average value is reported in Table1.

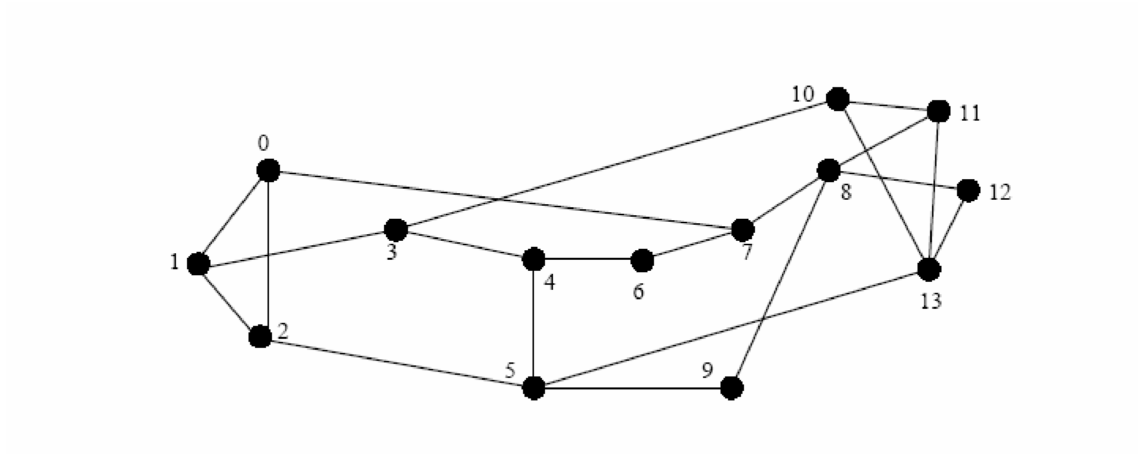

Fig6. NSFNET: 14-node network with 21 edges

Nodes (Links)	Connections	After *K-EDR*	After *OnestepCD_ANT*			Performance Enhancement (%)
			MIN	MAX	AVG	
14 (21)	20	7	5	5	5	29%
	40	12	9	10	9.7	20%
	80	23	17	18	17.1	26%
	120	34	23	24	23.3	29%
	182	53	33	35	34.8	34%

Table1. Performance comparison between *K-EDR* and *OnestepCD_ANT*

In Table1, we report the number of colors used for the RWA problem over the NSF network under the given connections. As we can see, *OnestepCD_ANT* process improves the performance compared with the *K-EDR* process.

We now apply our *OnestepCD_ACO* algorithm to the network topology shown in Fig.6, which is a 27-node network with 68 edges. Each edge represents a bi-directional fiber which can carry one signal in each direction. In this example, we randomly create a set of connections and compute the *K*-shortest paths for each connection, where *K*=1,2. Each instance is evaluated with 20 simulation runs. In Fig7, the performance improvement is shown as a function of the number of connections.

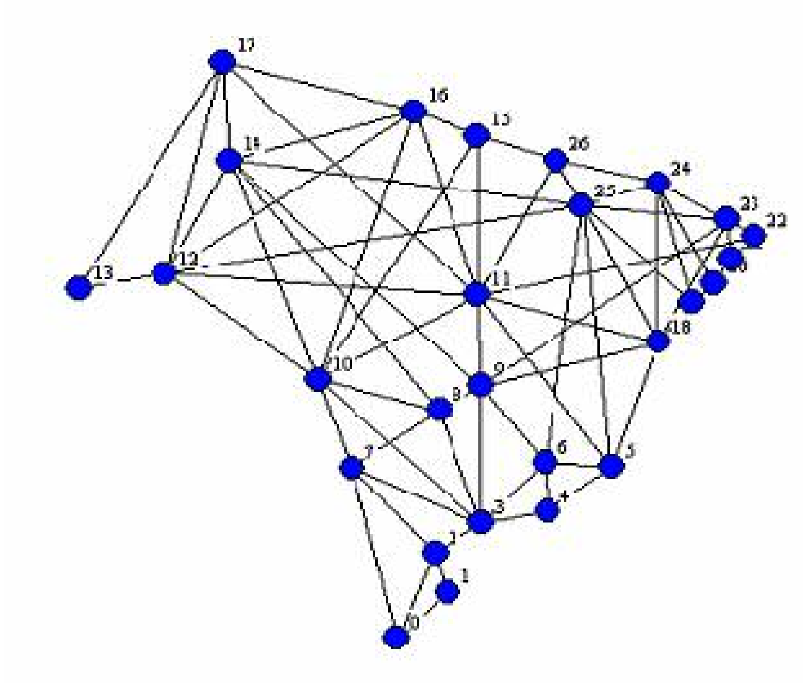

Fig7. 27-node network with 68 edges

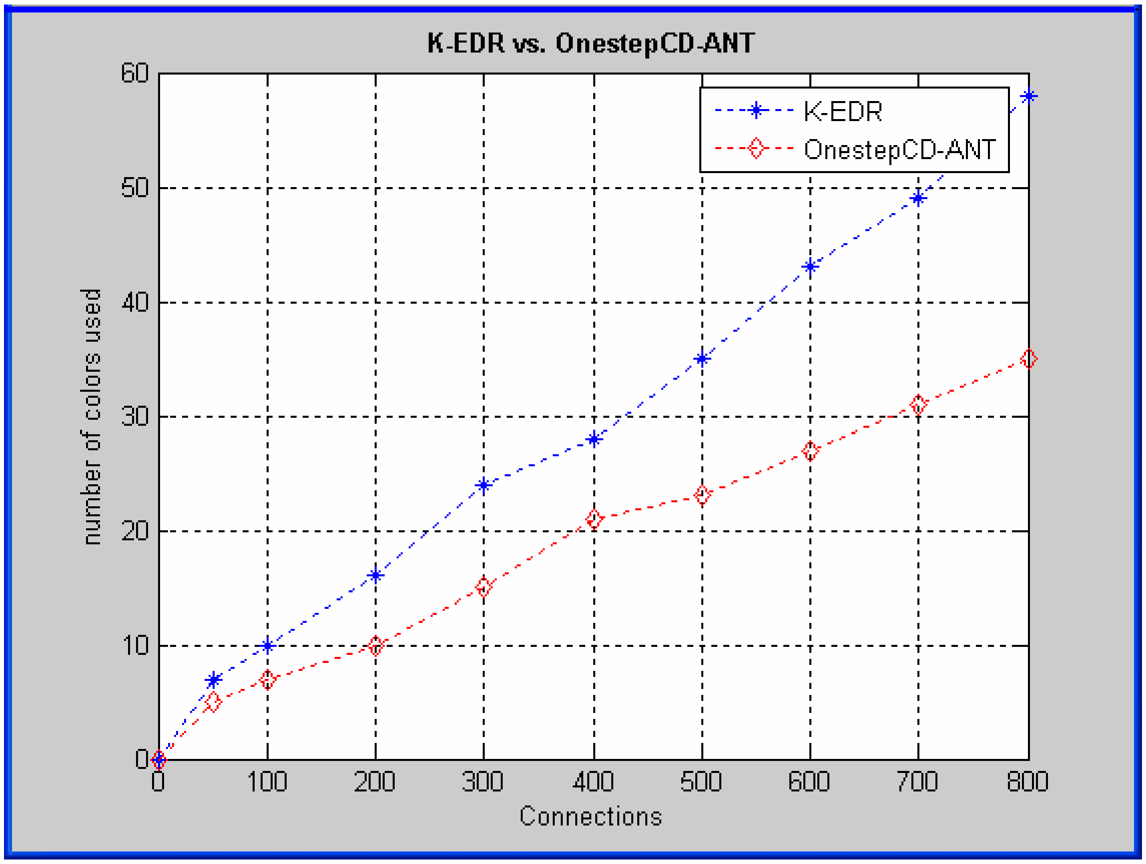

Fig8. Number of colors used for K-EDR algorithm and OnestepCD_ANT algorithm

6.2 OnestepCD vs. OnestepCD_ANT

The *OnestepCD_ANT* algorithm is an application of ACO, which is based on *OnestepCD* algorithm introduced by Li and Simha [3]. The *OnestepCD_ANT* process uses color degree for making a candidate set and calculating heuristic values. However, its color degree is different from what is used in *OnestepCD* algorithm in [3]. Detailed description and results will be presented in the following section.

6.2.1 Algorithm comparison between *OnestepCD* and *OnestepCD_ANT*

Li and Shiha[3] introduced *OnestepCD* algorithm to solve the partitioning coloring problem. This algorithm is based on the color degree of the nodes. Pseudo code is shown in *Algorithm 5*.

```
Algorithm 5: OnestepCD(CG)
Input: Color conflict graph (CG)

1. Remove from CG all edges $(i, j) \in E : i, j \in V^k$, for some k = 1,⋯,q;
2. Set $V' \leftarrow \emptyset$
3. while $|V'| <$ q do
4.     Set $X \leftarrow \emptyset$
5.     for k = 1,⋯,q : $V_k \cap V' = \emptyset$ do
6.         Set $X \leftarrow X \cup \arg\min\{CD(i) : i \in V^k\}$
7.     end-for
8.     Set $x \leftarrow \arg\max\{CD(i) : i \in X\}$
9.     Set $V' \leftarrow V' \cup \{x\}$
10.    Assign the minimum possible color to $x$;
11.    Remove from CG all nodes in $V_{c(x)} \setminus \{ x \}$
12. end-while
end OnestepCD.

Output: a set of colored nodes $V'$
```

We denote by $c(i)$ the component of node $i \in V$ (i.e. $i \in V_{c(i)}$) and by $V' \subseteq V$ the set of nodes of G already colored in a partial solution under construction to PCP. The color saturation degree $CD(i)$ of each node $i \in V$ is defined as the number of different colors used to color the neighbors of i in V'.

In line 1, all edges connection nodes in the same component of the partition are removed. The set of nodes already colored is initialized in line 2. The loop in lines 3-12 is performed until all partitions have been colored. In lines 4-7, set X will be built which is formed by the nodes with least colored saturation degree from each uncolored component. The node x with maximum colored saturation degree among those in X is selected in line8. Node x is colored with the minimum possible color in lines 9 and 10. All uncolored nodes in the same partition as x are removed from the graph in line 11 and the above steps are repeated.

OnestepCD_ANT algorithm which is shown in *Algorithm 3* used Color-Degree instead of colored saturation degree. For colored saturation degree, the neighbors of node i are already colored nodes which means already selected nodes for the candidate set. But the neighbors of node i in *OnestepCD_ANT* algorithm are all nodes in the graph which are connected with node i with an edge.

For *OnestepCD_ANT* algorithm, original colors for all nodes are given and used for making a candidate set. Whenever an ant do coloring, new candidate set is needed in *OnestepCD_ANT* algorithm. If color change of each node can be considered when candidate set is made, it will give more diversity for the process. Therefore Color-Degree was used instead of saturated color degree.

The **argmax** in line 8 in is applied as heuristic value in *OnestepCD_ANT* algorithm.

6.2.2 Result Comparison between *OnestepCD* and *OnestepCD_ANT*

Now we applied *OnestetpCD_ANT* to other topologies shown in Fig6, Fig7 and Fig9. Fig7 is a 27-node network with 68 edges. Fig9 contains 31-node network with 52edges (left) and 32-node network with 50 edges (right). These network topologies are slightly different with the network in Fig6. All links in these networks are bi-directional and there are connections to be established between all pairs of nodes while the network in Fig6 is unidirectional. Lightpaths are directional, i.e. the lightpath form node $i \in X$ to node $j \in X$ may be different from that connecting j to i. According, there are $|X| \bullet (|X-1|)$ lightpaths to be established.

Before applying K-EDP for getting routes for all connections, all the edges in the network were set to 1. Therefore total route length is same as total number of hops to route all the connections.

20 independent runs for *OnestepCD* process and *OnestepCD_ANT* process were performed for each instance. The results are shown in *Table2*.

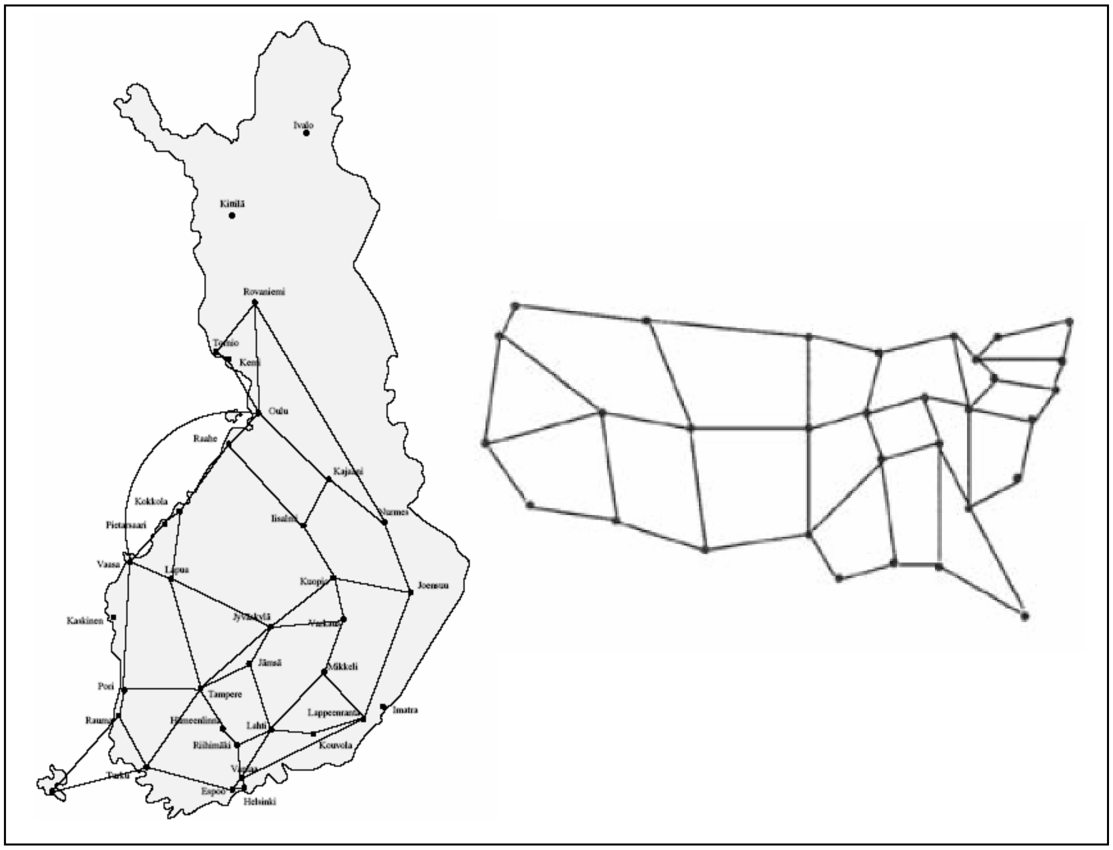

Fig9. 31-node network with 52 edges (left), 32-node network with 50 edges (right)

Nodes (Links)	Connections	After *K-EDR*	After *OnestepCD*	After *OnestepCD_ANT*		
				MIN	MAX	AVG
14 (21)	182	28	17	14	14	14
27 (68)	702	56	31	26	27	26.9
32 (50)	992	128	102	83	85	83.2
31 (51)	930	93	62	56	57	56.1

Table2. Number of colors used for *OnestepCD* algorithm and *OnestepCD_ANT* algorithm

6.3 With/Without EDP_ANT

OnestepCD_ANT focuses on reducing the number of colors, which is one objective of the RWA. Another objective is to reduce the total routing distance since it is based on the shortest path routing algorithm. As mentioned in section 2.2, routing in the RWA problem has an edge disjoint constraint. Therefore, *EDP_ANT* should focus on reducing the total routing distance while satisfying the edge disjoint constraint.

As an application of ACO, parameter setting for the *EDP_ANT* process will affect the performance of the algorithm. In (5.4), parameters α, β and γ determine the relative influence of the pheromone, the heuristic information and the routing information of other connections . In this simulation, we set α to 1 and β to 2 in order to give more weight to heuristic information when an ant decides the next hop. The parameter γ is set to 1.

Other parameters which should be tuned for the algorithm are related to pheromone updates. For local pheromone updates, original pheromone value is decreased by factor a factor of $1-\varepsilon$ where $0<\varepsilon<1$. Parameter ε is decreasing factor which makes used edge less desirable for the following ants and experimentally set to 0.1. Another parameter for pheromone update is ρ which is used for global pheromone updates. Experimentally, it is set to 0.3. For *pseudorandom proportional* rule, parameter q_0 is set to 0.25. The number of iteration, *MAX_ITER* is set to 100.

6.3.1 Results of EDP_ANT

Now we apply *EDP_ANT* to the unidirectional network topology in Fig6 and bi-directional network topologies in Fig7 and Fig9. The distance between two nodes is set to random integer in range of 1-9. The source and destination of each connection are randomly picked. The result is shown in *Table 3*. Even though EDP_ANT does not guarantee a decrease in the number of colors used, it reduces the total routing length, which is one objective of the RWA problem.

Nodes (Links)	Connections	Total route length		Number of colors	
		Before *EDP_ANT*	After *EDP_ ANT*	Before *EDP_ANT*	After *EDP_ANT*
14 (21)	40	309	267	10	9
	80	596	564	17	16
	120	879	870	23	23
	182	1377	1361	35	34
27 (68)	200	2005	1990	13	13
32 (50)	200	3408	3401	18	17
31 (51)	200	2924	2890	15	15

Table3. Influence of *EDP_ANT* algorithm to Total routing distance and Number of used colors

6.4 Tabu Search vs. ACO

6.4.1 Tabu Search algorithm

Tabu search (TS) is a heuristic method, belonging to a class of local search techniques. TS enhances the performance of a local search method by using memory structures. TS uses a local or neighborhood search procedure to iteratively move from a solution x to another solution x' in the neighborhood of x ($N(x)$), until some stopping criterion has been satisfied. To explore regions of the search space that would be left unexplored by the local search procedure and to escape local optimality, tabu search modifies the neighborhood structure of each solution as the search progresses and and makes the new neighborhood $N^*(x)$. The solutions admitted to the
$N^*(x)$, are determined through the use of special memory structures.

Perhaps the most important type of short-term memory to determine the solutions in $N^*(x)$, which gives its name to tabu search, is the use of a tabu list. In its simplest form, a tabu list contains the solutions that have been visited in the recent past (less than n moves ago, where n is the tabu tenure). Solutions in the tabu list are excluded from $N^*(x)$. Other tabu list structures prohibit solutions that have certain attributes or prevent certain moves. Selected attributes in solutions recently visited are labeled tabu-active. Solutions that contain tabu-active elements are tabu. This type of short-term memory is also called recency-based memory. Tabu lists containing attributes are much more effective, although they raise a new problem. When a single attribute is forbidden as tabu, typically more than one solution ends up being tabu. Some of these solutions that must now be avoided, might be of excellent quality and might not have been visited. To overcome this problem, *aspiration criteria* are introduced which allow overriding the tabu state of a solution to include it in the allowed set. A commonly used aspiration criterion is to allow solutions which are better than the currently best known solution.

6.4.2 Result Comparison between *OnestepCD_ANT* and Tabu Search

Noronha and Ribeiro[4] applied TS to the PCP. For the first step, they used the *OnestepCD* algorithm to build an initial solution. With this initial solution, they did tabu search for optimizing the number of colors used in the initial solution.

TS is applied to uni-directional network (see Fig6). Each approach was applied 20 times with an upper bound 10 minutes on the computation time. The comparison in average number of required colors (wavelengths) is shown in *Table4*.

Nodes (Links)	Connections	After *K-EDR*	*OnestepCD*+Tabu Search		After *OnestepCD_ANT*
			After *OnestepCD*	After Tabu Search	
14 (21)	20	7	6	5	5
	40	12	11	9	9.7
	80	23	18	17	17.1
	120	34	24	23.9	23.3
	182	53	36	35.2	34.8

Table4. Application of Tabu Search to 14-node uni-directional network with 21 edges

Tabu search is also applied to bi-directional networks. Each approach was applied 20 times upper bound 20 minutes on the computation time. The comparison is shown in *Table5*.

Nodes (Links)	Connections	After *K-EDR*	*OnestepCD*+Tabu Search		After *OnestepCD_ANT*
			After *OnestepCD*	After Tabu Search	AVG
14 (21)	182	28	17	15	14
27 (68)	702	56	31	28.1	26.9
32 (50)	992	128	102	91.2	83.2
31 (51)	930	93	62	58	56.1

Table5. Application of Tabu Search to bi-directional networks.

Chapter 7: Conclusion

In this thesis, we develop a new algorithm to solve the RWA problem using ACO. First, we divide the RWA problem to wave assignment problem and routing problem. Then we apply ACO to each instance. MOOP scheme is used to combile these two objectives. Wavelength assignment problem is set to primary goal of the RWA problem and is given a priority. These two objectives share the best solution so far to communicate each other.

The results we obtained on several network graphs show that our approach is effective. *OnestepCD_ANT* algorithm obtains better results than *OnestepCD* algorithm. *EDP_ANT* algorithm shows contribution of second primary objective to primary goal. With *EDP_ANT* algorithm, we may get better results in the number of required wavelength.

The comparison results of *OnestepCD_ANT* algorithm and *Tabu Serach* shows that *OnestepCD_ANT* can have better result under computation time constraint.

Bibliography

[1] M. Dorigo and T. Stüzle, The ANT Colony Optimization Meta-Heuristic.

[2] D. Bannerjee and B. Mukherjee, Practical approach for routing and wavelength assignment in large wavelength routed optical networks. *IEEE journal on Selected Areas in Communications*, 14:903-908, 1995.

[3] G. Li and R. Simha, The partition coloring problem and its application to wavelength routing and assignment. In *Proceeding of the First Workshop on Optical Networks*, 2000.

[4] T. F. de Noronha and C.C. Ribeiro, Routing and wavelength assignment by partition coloring. *European Journal of Operational Research* **171** (2006) (3), pp. 797–810.

[5] J.M Kleinberg, Approximation algorithm for disjoint path problems. *PHD thesis, MITm Cambridge, 1996.*

[6] M. Dorigo and T. Stüzle, Ant Colony Optimization. The MIT Press Cambridge, 2004

[7] F. Glover and M. Laguna, Tabu Search.

[8] R.Gould. Graph theory. *The Benjamin/Cumming Publishing Company*, Inc., 1988.

[9] M. J. Blesa and C. Blum, Approximation Algorithms and Metaheuristics, chapter On Solving the Maximum Disjoint Parhs Problem with Ant Colony Optimization. Taylor & Francis Books (CRC Press), Boca Raton, Florida, 2006.

[10] M. J. Blesa and C. Blum, A nature-inspired algorithm for the disjoint paths problem. In *the proceedings of the 9th International Workshop on Nature Inspired Distributed Computing (NIDISC'06)*. IEEE Computer Society press, 2006.

[11] B. Mukherjee, Optical Communication Network. *McGraw-Hill Series on Computer Communications,* 1997.

[12] A. Lim, Y. Zhu, Q.Lou and B.Rodrigues, Heuristic Methods for Graph Coloring Problems, *In Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International.*

[13] E. Hyytiä and J. Virtamo, Wavelength assignment and routing in WDM networks. *In Nordic Teletraffic Seminar 14*, pages 31-40, 1998.

[14] G. Rouskas, Routing and wavelength assignment in optical WDM networks. In John Proakis, editor, *Wiley Encyclopedia of Telecommunications*. John Wiley & Sons, 2001.

[15] P. Manohar, D. Manjunath, and R. K. Shevgaonkar, Routing and wavelength assignment in optical networks from edge disjoint paths algorithms, *IEEE Commun. Lett*. 6, 211-213 (2002)

[16] A. Hertz and N. Zufferey, 2006, A New Ant Algorithm for Graph Coloring, *Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization*, NICSO 2006, Granada, Spain, 51-60.

[17] M. Bessedik, R. Laib, A. Boulmerka and H. Drias, Ant Colony System for Graph Coloring Problem, *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on*

Intelligent Agents, *Web Technologies and Internet Commerce Vol-1 (CIMCA-IAWTIC'06)*

[18] Shawe-Taylor, J. and Zerovnik, J. v. (2001) Ants and graph coloring, in *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, ICANNGA'01*, pp. 276-279. Springer-Verlag.

[19] L. M. Gambardella , É. Taillard , G. Agazzi, MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows, *New ideas in optimization*, McGraw-Hill Ltd., UK, Maidenhead, UK, 1999.

[20] V. T'kindt, N. Monmarche, F. Tercinet, D. Laugt, An Ant Colony Optimization Algorithm to Solve. a 2-machine Bicriteria Flowshop Scheduling Problem, *European Journal of Operational Research*, 142:2, 2002, pp. 250-257.

[21] S. Iredi, D. Merkle in M. Middendorf. Bi-criterion optimization with multi colony ant algorithms, *First International Conference on Evolutionary Multi-Criterion Optimization (EMO'01),* vol. 1993 of *Lecture Notes in Computer Science* (pp. 359-372). Berlin, Springer-Verlag.

[22] M. Dorigo, and L. Gambardella, Ant Colony System: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation. v1. 53-66.

[23] Taehan Lee and Sungsoo Park, "Routing and Wavelength Assignment in WDM Ring Networks, IEEE Journal on Selected Areas in Communications, Vol. 18, No. 10, October 2000, Page(s):2146-2154

[24] M. Dorigo, V. Maniezzo, and A. Colorni, ``The ant system: Optimization by a colony of cooperating agents", *IEEE Transaction on System, Man, and Cybernetics-Part B,* vol. 26, No. 2, pp 656-665.

[25] T. Stützle and H. H. Hoos, MAX-MIN Ant System. *Future Generation Computer Systems*. 16(8):889--914,2000.

[26] V. Maniezzo, Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, *INFORMS Journal on Computing*, 11(4), 358-369, 1999.