Chapter 2

Heuristic Algorithms for Assigning Wavelengths to Static Lightpaths in WDM Optical Networks

2.1 Introduction

The problem of establishing lightpath between a given source destination pair through routing and assigning wavelength to the route is known as Routing and Wavelength Assignment (RWA) problem [1-8]. The RWA problem can be either static, where a set of lightpaths is known a priori, or dynamic, where lightpaths are established and released on the fly. Use of a good routing and wavelength assignment (RWA) algorithm can improve the performance of a WDM optical network significantly. An overall review of the existing RWA algorithms can be found in [3]. To tackle the problem more efficiently, usually the RWA problem is divided into two sub-problems, namely (i) Routing and (ii) Wavelength assignment. Each sub-problem can be solved separately. Here, we assume that the routes of the lightpaths are known in advance and hence consider the wavelength assignment problem only. In this chapter, we present two heuristic algorithms for assigning wavelengths to static lightpaths in WDM optical networks.

The rest of the chapter is organized as follows. In the next sub section 2.1.1, we define the wavelength assignment problem and pose it as a graph-coloring problem. In section 2 we propose the heuristic1 for solving the problem. It also includes the complexity analysis, correctness proof and the optimality proof of the algorithm. The heuristic 2 is presented in details in
section 3. The fourth section presents results of application of the algorithms. Finally, the last section concludes the chapter.

2.1.1 Wavelength Assignment Problem

Problem Statement

Given the physical topology of a network, a set of lightpaths and their routes, the problem is to assign a wavelength to each lightpath so that the total number of wavelengths used is minimized, subject to the condition that (i) the wavelength continuity constraint is satisfied and (ii) no two lightpaths are assigned the same wavelength if they share a common link.

Formulation as a graph-coloring problem:

Usually, a lightpath $LP_i$ is uniquely identified by a tuple $\langle \lambda_i, P_i \rangle$, where $\lambda_i$ is the wavelength used in the lightpath and $P_i$ represents the physical path corresponding to $LP$. Two lightpaths $LP_1<\lambda_1, P_1>$ and $LP_2<\lambda_2, P_2>$ can share the same fiber if and only if they use different wavelengths i.e. $\lambda_1 \neq \lambda_2$. Assigning wavelengths to lightpaths in a manner, that minimizes the number of total wavelengths required under the wavelength-continuity constraint, reduces to the graph-coloring problem, as stated below.

Step1: An auxiliary graph $G(V,E)$ is constructed, such that a lightpath $LP_i<\lambda_i, P_i>$ in the system is represented by a node $i \in V$ in graph $G$. There is an edge $e_{ij} \in E$ between two nodes $i$ and $j$ ($i,j \in V$) in graph $G$ if the corresponding lightpaths $LP_i<\lambda_i, P_i>$ and $LP_j<\lambda_j, P_j>$ pass through a common physical fiber link (see Fig. 1), i.e. $P_i \cap P_j \neq \phi$. 
Step2: The nodes of the graph G are colored so that no two adjacent nodes have the same color i.e. \( \lambda_i \neq \lambda_j \) if \( e_{ij} \in E \).

The above graph-coloring problem has been shown to be NP – complete [3], and it is difficult to determine the minimum number of colors needed to color a graph. In view of this, heuristic solution methods are generally adopted for graph-coloring problem [3]. In the next two sections, efficient heuristics are proposed to solve the problem.

2.2 Wavelength Assignment Heuristic 1

The wavelength assignment algorithms presented in this chapter consists of three phases, namely creation of auxiliary graph (algorithm AUXILIARY_GRAPH), determination of chromatic number (algorithm CHROMATIC_NO), and coloring of lightpaths (Heuristic Algorithm 1 and Heuristic Algorithm 2). In this section, we present the first two algorithms and Heuristic Algorithm 1 with each one followed by its complexity analysis.

At this point, let us introduce few more notations that will be used in the algorithms later.

\[ \begin{align*}
LP &: \text{ set of lightpaths in the WDM optical network.} \\
V &: \{ v_i | 1 \leq i \leq N \} \\
degree(i) &: \text{ degree of node } i \in V; \\
degree &: \{ \text{ degree}(i) | i \in V \}; \\
\text{color} &: \{ \lambda_j | 1 \leq j \leq K \}; \\
\text{color}(i), \text{new\_color}(i) &: \text{ wavelength color assigned to node } i \in V;
\end{align*} \]
neighbor(i) : set of neighbors of nodes i
   \[ j \in \text{neighbor}(i) \Rightarrow e_{ij} \in E \]

neighborhood_color(i): set of wavelength colors of neighbor(i)
   \[ p \in \text{neighbor}(i) \Rightarrow \text{color}(p) \in \text{neighborhood}_{\text{color}}(i) \]

hneighbor(i) : set of higher order neighbors of node i
   \[ j \in \text{hneighbor}(i) \Rightarrow \{ e_{ij} \in E \} \land \{ 1 \leq i \leq N \} \]

In the first phase, we obtain an auxiliary graph from the original WDM optical network using the following algorithm.

### 2.2.1 Algorithm AUXILIARY_GRAPH

**Input**: A set of lightpaths LP for a given WDM optical network.

**Output**: An auxiliary graph G(V,E).

**Steps**: V := \{ φ \}; E := \{ φ \}; /* Initialization */
   foreach l \in\ LP do
      Create node i;
      V := V \cup \{ i \};
   od
   foreach i \in\ LP do
      foreach j (j \neq i) \in\ LP do
         if (P_i \cap P_j \neq φ) then
            create an edge e_{ij};
            E := E \cup \{ e_{ij} \};
         fi
      od
   od
   output V & E ;

/* End of Algorithm */
Lemma 2.1: The complexity of algorithm AUXILIARY_GRAPH is O(N^2).

Proof. Trivial. □

The algorithm is illustrated with the help of an example shown in Figure 2.1(a) [3]. Figure 2.1(a) shows the physical topology of an example optical network with eight numbers of lightpaths on it. Applying the algorithm AUXILIARY_GRAPH on the given network, we obtain the auxiliary graph shown in Figure 2.1(b). It may be noted that there are also eight numbers of nodes in the auxiliary graph as there are eight numbers of lightpaths in the physical network.

The minimum number of colors needed to color a graph G is called the chromatic number K of the graph. If D(G) denotes the maximum degree in a graph, then K ≤ (D(G)+1) [3]; Next, in the second phase, the chromatic number of the auxiliary graph is found out using the following algorithm.

2.2.2 Algorithm CHROMATIC_NO

Input: An auxiliary graph G(V,E).

Output: Chromatic number K of the graph G.

begin
    degree := { φ };  
    foreach i ∈ V do
        degree := { degree(i) } ∪ degree;
    od
    K := max(degree) + 1;
end

/* End of Algorithm */
Lemma 2.2: The complexity of algorithm CHROMATIC_NO is O(N);

Proof. Trivial. □

To illustrate the algorithm, let us consider the same example as shown in Figure 2.1(a). We have already generated the auxiliary graph in Figure 2.1 (b). Now, the algorithm CHROMATIC_NO is applied on the auxiliary graph, and the chromatic number K for the graph is found to be 4 (as the maximum degree of any node in the auxiliary graph is 3). Hence, a maximum of 4 buffers will be required at each node (except the first node) to store the neighborhood color information. Actual wavelength assignment is next done in the third phase using the auxiliary graph and its chromatic number obtained in the previous two phases.

2.2.3 Heuristic Algorithm 1: LIGHTPATH_COLORING

Input: An auxiliary graph G(V,E) and its chromatic number K.

Output: At the termination of the algorithm each node, i ∈ V, has its permanent color, color(i).

Steps:
Step0: /* Initialization */
    do
        foreach i ∈ V do
            color(i) := λ1;
        od
        foreach i ∈ { V - v1 } do
            neighborhood_color(i) := { λ1 };
        od
    od
Step i (i > 0): /* Coloring */
foreach \( i \in V \) do
  foreach \( j \in \text{hneighbor}(i) \) do
    if \( \text{color}(i) = \text{color}(j) \) then
      do
        new_color(j) := \text{first}(\text{color} - \text{neighborhood_color}(j));
        /* ‘first’ operator consumes unit time, returns the first element of the set */
        \text{color}(j) := \text{new_color}(j);
        foreach \( l \in \text{hneighbor}(j) \) do
          \text{neighborhood_color}(l) := \text{neighborhood_color}(l) \cup \{ \text{color}(j) \};
        od
      od
  od
/* End of Heuristic Algorithm1 */

\textbf{Complexity Analysis}

\textbf{Lemma 2.3}: The complexity of algorithm LIGHTPATH_COLORING is \( O(N^3) \);  
\textbf{Proof}. Assuming ‘first’ operator consumes unit time, the complexity follows directly from the three nested ‘for’ loops in the algorithm where each ‘for’ loop requires \( N \) computations. \( \square \)

Continuing with the same example of Figure 2.1(a), we explain the above algorithm here. So far we have obtained an auxiliary graph in Figure 2.1(b). Now, in the last phase, algorithm LIGHTPATH_COLORING is applied on the auxiliary graph of Figure 2.1(b) with the known chromatic number \( K=4 \), and the final wavelength color assigned to each node is obtained as shown in Figure 2.1(c).

\textbf{Theorem 1}: The complexity of the three-phase wavelength assignment algorithm is \( O(N^3) \);  
\textbf{Proof}. Follows from Lemmas 2.1, 2.2 and 2.3. \( \square \)
Correctness Proof

In the algorithm LIGHTPATH_COLORING, initially all the nodes in the auxiliary graph is colored by $\lambda_i$ and subsequently the color of a node is changed by its lower order adjacent nodes, if required. At the $i^{th}$ iteration of the outermost loop of the algorithm, the color of the $i^{th}$ node is taken as permanent and is different from its lower order adjacent nodes. To determine the effect of the $i^{th}$ node on its higher order nodes, we consider Figure 2.2. The sets $S1$ and $S2$ in Fig. 2.2 are defined as follows.

$$S1 = \{ j | j \in hneighbor(i) \text{ and } i,j \in V \}$$
$$S2 = \{ k | k \in hneighbor(j) \text{ and } j,k \in V \}$$

The correctness of algorithm LIGHTPATH_COLORING can be established through the following lemmas.

**Lemma 2.4**: Algorithm LIGHTPATH_COLORING eventually terminates.

**Proof**. Due to the presence of deterministic ‘for’ loops, the algorithm requires finite amount of time to complete the coloring of nodes. More specifically, it is always required by the algorithm to visit exactly $(N-1)$ nodes of the auxiliary graph where $N = |V|$. Hence, the algorithm always terminates after $(N-1)$ iterations. $\square$

**Lemma 2.5**: At the termination of algorithm LIGHTPATH_COLORING, every node is assigned a color.

**Proof**. Initially all the nodes of auxiliary graph are colored by a wavelength. This initial color may be changed to a new color in subsequent iterations of the algorithm. Hence, when the algorithm terminates each node has either the initial color or the changed color. $\square$
Lemma 2.6: Any two neighboring nodes will have distinct colors.

Proof. Let us prove it by contradiction. Say, nodes j and k have identical colors at the end of the algorithm and they are adjacent to each other. Now, we will prove that color(j) and color(k) cannot be the same if we follow the algorithm correctly. Let us consider the stage at which the color of the node i is made permanent. Let node j be one of the higher order nodes of i and k be one of the higher order nodes of j (Figure 2.2). If color(i) is the same as color(j), color(j) will be changed in step i of algorithm LIGHTPATH_COLORING. Let us denote this changed color as color’(j) (≠color(j)) i.e. color’(j) ∈ neighborhood_color(k), as k ∈ hneighbor(j); and this information will be propagated to all the higher order nodes (including k) adjacent to the node j. So, hneighborhood(k) will include color’(j) as its element. Without loss of generality, let us assume that color’(j) is the permanent color of node j and all the previous intermediate colors of node j during its transition from color(j) to color’(j) have already been propagated to node k. Now, following the same logic, if the color(k) is the same as color’(j), color(k) will be changed to color’(k) (≠color(k)). According to the algorithm, color’(k) should be such that color’(k) ∉ neighborhood_color(k) whereas color’(j) ∈ neighborhood_color(k). Hence, color’(k) ≠ color’(j); □

Theorem 2: Algorithm LIGHTPATH_COLORING is correct.

Proof. Follows from Lemmas 2.4, 2.5 and 2.6. □

Optimality Proof

Finally, the optimality of algorithm LIGHTPATH_COLORING is established through the following lemmas.
Lemma 2.7: If a graph-coloring algorithm uses $K$ number of colors, where $K$ is the chromatic number of the graph, then the algorithm is optimal.

Proof. See reference [3]. □

Lemma 2.8: Algorithm LIGHTPATH_COLORING uses not more than $K$ number of colors where $K$ is the chromatic number of the auxiliary graph $G(V,E)$.

Proof. For any node $i \in V$, $\text{degree}(i) < K$ [By definition of chromatic number]. Hence, any node in $G$ and its neighbors can be colored by maximum $K$ number of colors. These $K$ colors will always be in the range $\lambda_1 \ldots \lambda_K$ for every node. This is ensured by the operator ‘first’ which allows reuse of colors. For instance, if node $j$ is neighbor of nodes $i$ and $k$, $(i<j<k)$, but node $i$ is not adjacent to node $k$, then color($i$) and color($k$) can be the same. This is due to the reason that neighborhood_color($k$) will have the element corresponding to color($i$) as vacant because any information regarding color($i$) need not be propagated to node $k$ as per the algorithm. So, only the first $K$ number of colors in the list neighborhood_color() will be sufficient for algorithm LIGHTPATH_COLORING. □

Theorem 3: Algorithm LIGHTPATH_COLORING is optimal in the number of wavelengths used for coloring.

Proof. Follows from Lemmas 2.7 and 2.8. □
Figure 2.1(a). Physical topology of an optical network in which different colors represent different lightpaths.

Figure 2.1(b). The auxiliary graph corresponding to the lightpaths in the network shown in the Figure 1(a).
Figure 2.1(c). Final color assigned to each node in the auxiliary graph

Figure 2.2: Network nodes considered at the $i^{th}$ iteration of the algorithm LIGHTPATH_COLOR
2.3 Wavelength Assignment Heuristic 2

2.3.1 Heuristic Algorithm 2: WAVELENGTH_ASSIGNMENT

Input: The auxiliary Graph G (V, E) corresponding to the requested lightpaths.
Output: Wavelength (color) assigned to every node (lightpath).
Temporary variables: color array, node-color array.

/* The size of color array is k, where k is the chromatic number [3] of the auxiliary graph. The cell position of the color array represents the color number, which is used to color the nodes. The content of a cell of a color array is either 0 or 1. For a particular node if one of its adjacent nodes is colored by color number i, then the ith bit of the color array is set to 1.

The size of node-color array is n, where n is the number of nodes of the auxiliary graph. The cell position of the node-color array represents the node number of the auxiliary graph. The content of a cell of a node-color array is the color number using which color the appropriate node is colored. */

Step1: Color the first node of the graph with the first color of the color array.
Step2: For i = 2 to N do
  2.1: Reset all the cells of the color array.
  2.2: Check the adjacent nodes of node [i].
    For j = 1 to k-1 do
      If node [j] is already colored then /* through node-color array */
      Set the appropriate cell of color array accordingly the color of node [j].
      Else no operation.
    End of for j.

28
2.3: Color node \([i]\) with the first unused color of the color array.

Step3: End of the algorithm.

**Complexity Analysis**

**Theorem 4**

The complexity of the WAVELENGTH-ASSIGNMENT algorithm is \(O(N^2)\).

**Proof:**

In step2 of the algorithm the ‘for’ loop executes \((N-1)\) times. Inside this ‘for’ loop, step 2.1 needs maximum \(k\) cells to initialize the color array, step 2.2 executes maximum \((k-1)\) times to check the adjacent nodes of every node and step 2.3 requires checking maximum \(k\) cells to find out the first reset cell from the color array. As \(k\) is less than \(N\), complexity of the algorithm is \((N-1)(N+N+N)\) which is also \(O(N^2)\).

**Correctness Proof**

**Lemma 2.9**

Algorithm WAVELENGTH-ASSIGNMENT eventually terminates.

**Proof:**

The algorithm uses two ‘for’ loops. The outer ‘for’ loops executes \((N-1)\) times. In between the outer ‘for’ loop the different steps and the inner ‘for’ loop are executed maximum ‘\(k\)’ times. The parameters \(N\), \(k\) used in this algorithm are also finite. Hence, the algorithm terminates after a finite amount of time.
Lemma 2.10

At the termination of the algorithm, every node is assigned a wavelength.

Proof:

The first node is colored with first value of the color array. Then one by one, all other nodes are colored with the first available color of the color array, which is not used by any of its adjacent nodes. This is ensured by the different sub-steps of step2 of the algorithm. Hence, when the algorithm terminates, each node is assigned a color.

Lemma 2.11

Any two neighboring nodes will be assigned distinct wavelength.

Proof:

It is ensured by the steps 2.1, 2.2 and 2.3 of the algorithm, where we reset all the cells of the color array, checking the adjacent color nodes, set the cells of the color array depending upon the color variable which are used to color the adjacent nodes and then color the node with the first unused color from the color array that ensures that no adjacent neighbors are colored by the same color.

Theorem 5

Algorithm WAVELENGTH-ASSIGNMENT is correct.

Proof:

Follows from Lemma 2.9, 2.10 and 2.11.
Optimality Proof

Lemma 2.12

If a graph-coloring algorithm uses k number of colors, where k is the chromatic number \(^3\) of the Graph, then the algorithm is optimal.

Proof:

See reference \(^3\).

Lemma 2.13

Algorithm WAVELENGTH-ASSIGNMENT uses not more than k number of colors where k is the chromatic number of the auxiliary graph G (V,E).

Proof:

For any node \(i \in V\), degree (i) < k. [By definition of chromatic number]. Hence, any node in G and its neighbors can be assigned colors by maximum k number of colors. These k colors will always be in the range \(\lambda_1, \ldots, \lambda_k\) for every node. This is ensured by the table-searched operation, which allows reuse of colors.

Theorem 6

Algorithm WAVELENGTH-ASSIGNMENT is optimal in the number of colors used.

Proof:

Follows from Lemma 2.12 and 2.13.
2.4 Results

We now apply the proposed heuristics on two different networks. Figure 2.3 shows the physical topology of the network 1 with eight nodes and eleven links. Requested lightpaths and their routes are shown in the Table 2.1. The auxiliary graph corresponding to the lightpaths of the network 1 is shown in the Figure 2.4. Figure 2.4 also indicates the wavelength assigned to each lightpath. The heuristics are also applied successfully on network 2 (Figure 2.5) with six nodes and eight links. Lightpaths to be assigned wavelength are shown in Table 2.2 and the result of wavelength assignment is presented in Figure 2.6.

Figure 2.3: Network 1 with eight nodes
<table>
<thead>
<tr>
<th>Lightpath No.</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 – 2 – 3</td>
</tr>
<tr>
<td>2</td>
<td>2 – 3 – 4</td>
</tr>
<tr>
<td>3</td>
<td>8 – 2 – 3 – 4</td>
</tr>
<tr>
<td>4</td>
<td>7 – 3 – 2</td>
</tr>
<tr>
<td>5</td>
<td>7 – 3 – 4</td>
</tr>
<tr>
<td>6</td>
<td>8 – 7 – 6</td>
</tr>
<tr>
<td>7</td>
<td>8 – 7 – 3</td>
</tr>
<tr>
<td>8</td>
<td>6 – 5</td>
</tr>
<tr>
<td>9</td>
<td>3 – 4 – 5</td>
</tr>
<tr>
<td>10</td>
<td>7 – 6 – 4</td>
</tr>
</tbody>
</table>

Table 2.1: Lightpaths and their routes for Network 1.

Figure 2.4: The auxiliary graph for the lightpaths and wavelength assigned to each lightpath in Network 1.
Figure 2.5: Network 2 with six nodes.

<table>
<thead>
<tr>
<th>Lightpath No.</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 – 2 – 1</td>
</tr>
<tr>
<td>2</td>
<td>5 – 3 – 2 – 1</td>
</tr>
<tr>
<td>3</td>
<td>2 – 6</td>
</tr>
<tr>
<td>4</td>
<td>4 – 3 – 2</td>
</tr>
<tr>
<td>5</td>
<td>2 – 6 – 5</td>
</tr>
<tr>
<td>6</td>
<td>1 – 6 – 5</td>
</tr>
<tr>
<td>7</td>
<td>6 – 5 – 3</td>
</tr>
<tr>
<td>8</td>
<td>5 – 4</td>
</tr>
</tbody>
</table>

Table 2.2: Lightpaths and their routes for Network 2

Figure 2.6: The auxiliary graph for the lightpaths and wavelength assigned to each lightpath in Network 2.
Comparison

The heuristic 1 is compared with the heuristic 2 and the algorithm presented in reference [9]. The result is shown in Table 2.3.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of wavelengths used for</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i) Heuristic algorithm 2</td>
<td>Network 1: 4, Network 2: 3</td>
<td>O(N^2)</td>
</tr>
<tr>
<td>(iii) Heuristic Algorithm 1</td>
<td>Network 1: 4, Network 2: 3</td>
<td>O(N^3)</td>
</tr>
</tbody>
</table>

Table 2.3

From Table 2.3 it is observed that each of the three algorithms uses minimum number of wavelengths for both Network 1 and Network 2. However, in terms of computational complexity, the heuristic algorithm 2 is superior to the heuristic algorithm 1 and the algorithm described in reference [9].

2.5 Conclusion

In this chapter, two heuristic algorithms for wavelength assignment have been presented. Given the static lightpath demands in a network, the algorithms are able to assign wavelengths to these lightpaths in a manner so that the number of wavelengths used is minimum. The complexities of the algorithms are shown to be O (N^3) and O (N^2) respectively, where N is the number of static lightpaths to be colored.
References


