Wavelength Assignment and Routing in WDM networks

Esa Hyytiä and Jorma Virtamo
Laboratory of Telecommunications Technology
Helsinki University of Technology
P.O.Box 3000, FIN-02015 HUT, Finland
Email: esa.hyytia@hut.fi, jorma.virtamo@hut.fi

June 1998

Abstract

With wavelength division multiplexing (WDM) several optical signals can be transferred in a single optical fiber [6][7]. This technology allows more efficient use of the huge capacity of an optical fiber but also poses new network design and management problems, especially when wavelength conversion is not possible in the nodes. In this paper we consider the routing and wavelength assignment problem in such networks. Once routes are fixed the wavelength assignment is essentially a graph coloring problem. Several heuristic methods for coloring a given graph are studied. Also an iterative algorithm for finding a reasonably good routing and wavelength assignment is represented and tested with fully connected networks.

1 Introduction

The ever increasing demand of higher transmission bandwidth requires new solutions. One promising concept for increasing the capacity is wavelength division multiplexing (WDM). In WDM several optical signals using different wavelengths are transferred in a single optical fiber. Thus the huge capacity of the optical fiber can be used more efficiently. The solution can also be cost effective as the existing physical network can be used.

The main characteristics of WDM can concisely be summarized as follows:

- fully photonic network where fiber amplifiers are used
- several channels are transmitted simultaneously in each fiber
- the capacity of network is great (tens of Gb/s)
- the network forms a wide backbone-network
- routing in nodes is based on wavelengths

In this paper we concentrate on routing and wavelength assignment problem in WDM networks. When several signals share the same fiber they must use different wavelengths. The available technology sets an upper limit to the number of wavelengths. Thus we are

led to consider the problem of creating a given set of connections in the network with the minimum number of wavelengths. The formulation of the optimization problem depends on whether wavelength conversion is possible in the nodes or not. If the wavelength conversion is possible the optimal solution just minimizes the maximum number of used channels over the links. The routing problem is the same as in normal circuit-switched networks where the only limiting factor is the number of channels on each link.

On the other hand, if wavelength conversion cannot be done in the nodes, this sets new constraints to the optimization problem. Each connection uses the same wavelength on all links along its route. A feasible solution uses less or equal number of wavelengths on each link than there are available and no two connections sharing a common link have the same wavelength.

There can be also networks with limited possibility to wavelength conversion. Such networks are not discussed in this paper, though. So from now on we assume that wavelength conversion cannot be done in any node. We also assume that there is no need for dynamical reconfiguration of the network, i.e. the set of connections is static.

The routing and wavelength assignment problems are tightly linked together. The problem has been discussed in several recently published papers [8][9][11][12]. In the approach discussed here, we first determine the routes for each connection and then try to assign the wavelengths with minimum number of used wavelengths. This is done iteratively so that the routing is changed slightly after the coloring with the aim to find a configuration which can be colored with an even smaller number of colors. In practice it is enough to find a solution which does not use more wavelengths than the available technology allows.

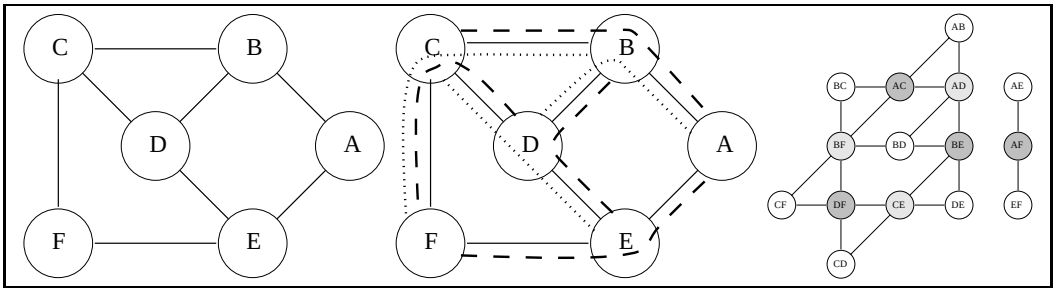

Figure 1: Example network and it is optimal configuration.

The process of routing and wavelength assignment is represented in figure 1. On the left is a physical network. In the middle the routing is fixed and wavelengths are assigned. The graph on the right is the graph which we must color, i.e. nodes of the graph represent connections, denoted by origin destination pairs, and nodes are neighbors (connected by an edge) if and only if corresponding connections share some common link. In order to avoid wavelength conflicts in the network the graph has to be colored in such a way that neighbor nodes always have different colors.

The number of different wavelengths required depends greatly on the used routing. Without considering the wavelength assignment problem more closely at this point one can still draw some conclusions about routing:

- Short routes which use only one link can use any free wavelength as the decision made has no effect elsewhere.

- Usually the long routes which have many hops are tricky as they reserve the wavelength from several links. Thus one usually should prefer short paths whenever available.

- The number of connections on any link clearly sets a lower limit for the total number of required wavelengths.
- The number of connections which share some link with a point-to-point connection set an upper limit to the number of required wavelengths.[1]
- When the network is cut in two subsets some number of links cross the border. The number of connection between two sets divided by the number of available links sets a lower bound for the number of required wavelengths [10].

The shortest path between two nodes can be obtained by using e.g. the Dijkstra algorithm or the Floyd algorithm. Both algorithms have same complexity $O(v^3)$ if the paths between each node pair are searched. In practice the Floyd algorithm is usually a bit better due smaller constant coefficients [13].

Once the routing is fixed the problem is to minimize the number of used wavelengths. As discussed above, the wavelength allocation can be mapped to a graph node coloring problem, which is a well-known NP-complete problem. In the next chapter several graph coloring algorithms are discussed and tested. This is the main topic of the paper. Then in chapter 3 we return to the joint routing and wavelength assignment problem. Finally, in chapter 4, we present some conclusions.

2 Wavelength assignment

When the routing is fixed, our task is to minimize the number of used wavelengths. The problem can be represented as a graph node coloring problem. In coloring graph each node represents one point-to-point connection (see figure 1). Those connections which share some common link are neighbors in the coloring graph, i.e. are connected by an edge, and thus must be colored with different colors. We assume here that links are alike, i.e. capacities of links are same. So our only objective is to minimize the number of different wavelengths required.

As the graph node coloring problem is NP-complete heuristic methods must be used for a practical solution. A number of different heuristic methods have been proposed. Some of them are based on well-known generic methods such as simulated annealing (SA) and genetic algorithms (GA). A more recent heuristic algorithm which works very well with graph coloring problems is the tabu search (TS) [3]. The light weight end of coloring algorithms are representd by greedy algorithms [4][5]. These, as well as the exhaustive search, will be discussed in more detail in the following.

2.1 Greedy algorithms

Greedy algorithms work in some predefined order through all the nodes and assign some free color to them. So the basic step in these algorithms is that they assign such a color to next node that does not cause violation within the subgraph already given colors. There are many variants of greedy algorithms. The one we used tries to color next node first with color 1, then with color 2 etc. The order in which nodes are given a color is determined by their degree, i.e. the number of neighbors. The node which has most neighbors is colored

[1] Graphs' chromatic number χ is the minimum number of colors needed to color its nodes. For any graph holds inequality $\chi(G) \leq \Delta + 1$ where Δ is maximal degree (number of neighbors) of the graph.

first. Another variant of greedy algorithms, called DSATUR [5], dynamically chooses the next node according to number of possible colors per node.

2.2 Exhaustive search

The algorithm which always finds the optimum coloring of given graph is represented in figure 2. The algorithm divides the possible colorings to two different cases in each step until the graph is perfect, i.e. each node is a neighbor of all the other nodes. In each step a pair of nodes which are not neighbors are searched. Now these nodes can be colored with the same color or with different colors. If the nodes are given the same color, we can clearly merge them into one node inheriting all the neighbors of the merged nodes. Otherwise, if different colors are given to the nodes we can draw an edge between them (right subtree in figure 2). At the end, we pick the one among all the perfect graphs obtained which has the smallest number of nodes.

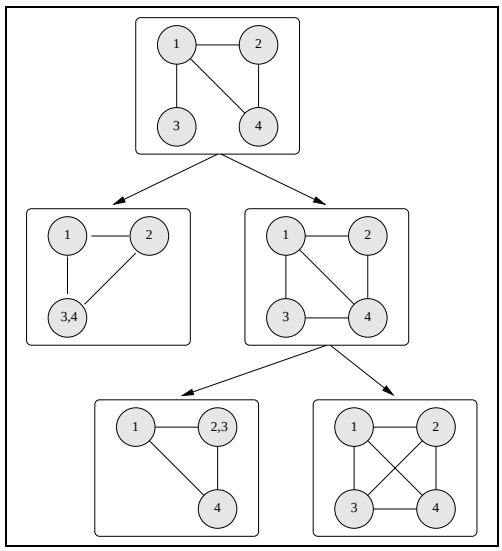

Figure 2: Coloring the graph with exhaustive search.

The search tree clearly becomes too large to handle as the number of nodes increases. One way to get a little bit further is to prune the search tree. If at each stage we drop those candidates which do not look promising the algorithm can handle considerably larger graphs. The greedy algorithm can be used to assess whether a graph is promising. Furthermore it holds for any graph G that

$$\chi(G) \geq \frac{\nu^2}{\nu^2 - 2\epsilon},$$

where $\chi(G)$ is the graph's chromatic number, i.e. the smallest number of colors needed to color the graph, ν is the number of nodes and ϵ is the number of edges. The inequality can be used here to avoid searching of such subtrees which cannot contain better solutions than the best one found so far.

2.3 Simulated annealing

Simulated annealing (SA) is a standard technique for hard combinatorial optimization problems [1][2]. The idea is to simulate annealing of some object. The objective function

represents the energy of the system and the control variable T represents its temperature. The higher the temperature the greater is the probability of acceptance of a move which leads to a higher energy state. The algorithm itself is very simple and easy to adapt to different kinds of problems, which is one of main reasons for its success.

The node coloring problem can also be solved with SA:

1. In the beginning assign each node a unique color.
2. Set the temperature $T = T_0$ (e.g. $T_0 = 1$).
3. The energy E of the system is the number of used colors.
4. Choose a random node and a random new color for it. Make sure that the new color does not lead to an illegal configuration.
5. Compute the change of energy ΔE.
6. If $\Delta E < 0$ or $e^{-\Delta E/T} > \text{rnd}(0, 1)$ accept the change.
7. If there has been at least M changes or N trials, then set $T = \alpha \cdot T$ (α is a small constant, eg. 0.95).
8. If $T > T_1$ go back 4.

Also other kinds of formulations have been suggested for energy function [1]. A drawback with this formulation is that energy can only have discrete values and it makes hard for the algorithm to find the right direction to advance.

2.4 Genetic algorithms

Genetic algorithm (GA) is another widely used standard method for hard combinatorial problems. In GA the idea is to simulate evolution. Here vectors represent genotypes and the aim is to find as good an individual as possible. Also the node coloring problem can be solved with GA [1]. In this case the vectors define the order in which the nodes are colored. So basically we try to find the best ordering to color the nodes with the greedy algorithm. The choice of crossover operation for permutations is not straightforward and several different schemes have been proposed. Here we used the following operation:

1. Let A and B be the parents. A is chosen randomly but favoring those who give a good coloring. B is chosen randomly from the whole population. The length of both vectors is N.
2. Set indeces $i_A = i_B = 1$ and set the child C to null.
3. Choose vector A with probability of 0.75 and vector B with probability of 0.25.
4. Add the next element, pointed by i_A or i_B, of the chosen vector to the child vector C if it is not already there.
5. Increment the value of index by one so that it points to the next element of parent vector.
6. Repeat this until the child C contains all the values $1 \ldots N$.

So basicly the order of both parents is combined to the child. On each step the next element of randomly chosen parent is copied to the child if it is not there yet. Index i_A points to the next element of parent A and i_B vice versa. As a mutation operator we simply exchange the place of two random nodes in the vector.

2.5 Tabu-search

Tabu search (TS) is a relatively new heurictic method [3][1][2]. It is basically a random local search, but some movements are forbidden, i.e. tabu. Usually a move leading back to previous point is classified as a tabu move for certain number of rounds. This should make it possible to get away from local minima. The search is ended when the cost function reaches a certain predefined value or a certain number of rounds has elapsed.

For the graph node coloring problem the tabu search works very well. This algorithm differs from all the previous ones in that it does not try to find the minimum coloring but a legal k-coloring for the given graph, i.e. it tries to choose for each node one of the k colors in such a way that no neighboring nodes get the same color.

Let $s = (V_1, \ldots, V_k)$ be a partition of graph G, where subset V_i of nodes represents those nodes having color i. Define a cost function as

$$f(s) = \sum_i |E(V_i)|,$$

where $|E(V_i)|$ is the number of edges in subgraph V_i. If there is an edge in some subgraph it means there are neighbors sharing the same color. So when $f(s) = 0$ we have a legal k-coloring for the graph.

1. We are given: graph G, target number of colors k, length of tabu list $|T|$, number of neighbors rep, and maximum number of iterations $nbmax$.
2. Set some initial configuration $s = (V_1, \ldots, V_k)$.
3. Set $nbiter = 0$.
4. Initialize tabu list T.
5. As long as $f(s) > 0$ and $nbiter < nbmax$
   (a) Find $nrep$ neighbors s_i for which $s \to s_i \notin T$ or $f(s_i) \leq A \cdot f(s)$.
   (b) Choose the best among them (or the first for which $f(s_i) < f(s)$).
   (c) Update tabu list T.
   (d) Set $s = s'$ and $nbiter = nbiter + 1$.
6. If $f(s) = 0$ we have find a legal k-coloring for given graph. Otherwise we can increase k and try again.

As a neighborhood for given partition we define partitions where one node is moved to another subset. Aspiration level A is used to accept even tabu moves if the result leads considerably better result.

In order to find the minimum k for which the algorithm finds a legal coloring, we must run the algorithm several times with decreasing values of k and iterate until the algorithm fails. On the other hand, if we are only interested in finding a feasible k-coloring the iteration is not required. This can be the case with WDM.

2.6 Some results of graph coloring algorithms

All the presented algorithms were coded in C language and tested with random graphs where the nodes were neighbors with probability of 0.5. Figure 3 shows the average running time of each algorithm. It should be noted that the used parameters have a great

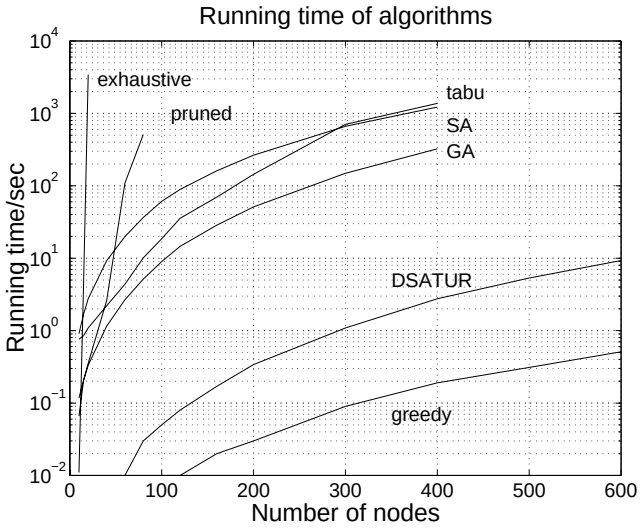

Figure 3: Running times of graph coloring algorithms.

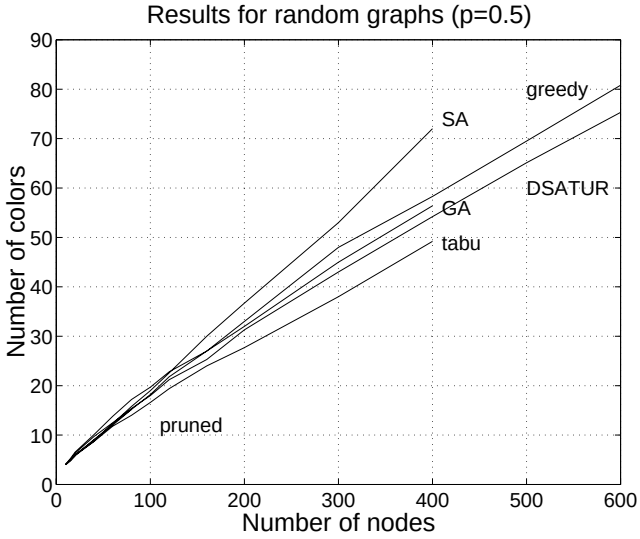

Figure 4: Number of used colors with different algorithms.

effect on the running times and the final results of certain algorithms. So these figures should be considered as examples only.

The algorithms can be grouped into three classes according to their running times (see figure 3). Greedy algorithms are clearly the fastest, then come heuristic algorithms SA, GA and TS. The exhaustive search (full and pruned) belong to third class. With regard to the optimization result (see figure 4) the tabu search was clearly the best. GA was also successful, but our implementation of SA gave worse results than greedy algorithms when the number of nodes was large.

3 Routing and wavelength assignment

When the routing is not fixed yet the optimization problem becomes naturally even harder. It is clear that usually short routes lead to better solutions than longer ones. So it makes sense to limit the search space to those paths which minimize the total number of hops.

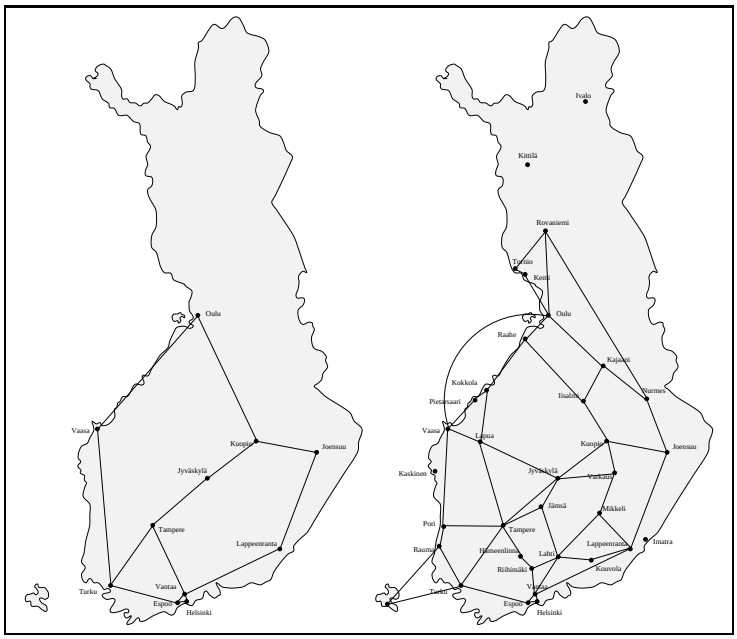

Figure 5: Hypothetical WDM-network in Finland.

The algorithm tries different sets of routes iteratively. Each set of routes is colored with greedy algorithm and the result is used as an estimate for the goodness of the choice. The change in routing (one point-to-point connection is routed in a different path) is made randomly and only the good changes, which lead to less or equal number of colors, are accepted. Thus, the route selection algorithm is essentially a local random search. It is probably worth trying to use some sophisticated heuristic on this higher level also.

In figure 5 there are two hypothetical physical networks residing in Finland. The smaller one is quite easy and our algorithm finds the optimal solution (it is easy to see that there cannot be a better configuration). The second one, on the other hand, represents a

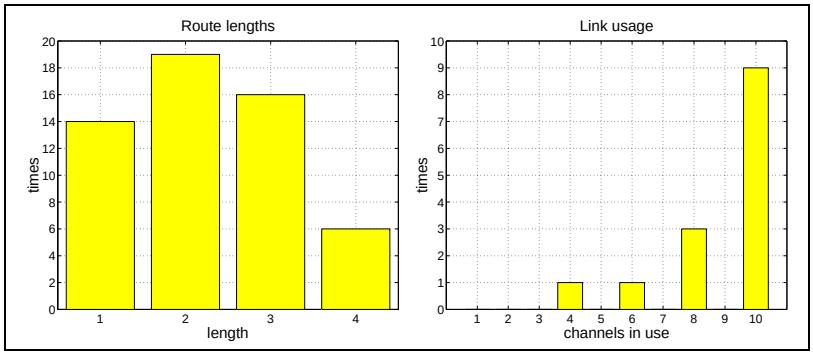

Figure 6: Route length and link usage distributions of first network.

harder problem. The algorithm finds a reasonably good solution for it but, as can be seen from link usage diagram, if just few connections could be handled with other colors the total number of colors would be significantly smaller. So increasing the number of links would probably lead to better solution (i.e. a better link usage distribution).

Overall one could assume that a better heuristic algorithm in selecting the routes could probably lead to both quicker and better solutions. The effect of better coloring algorithm seems to have no effect on the results, and this suggests that actual graphs to be colored are 'easy' problems. Or possibly the used estimate (greedy algorithm) drives the route

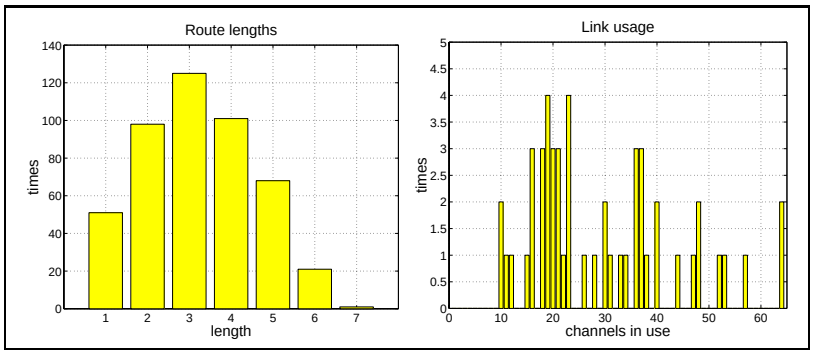

Figure 7: Route length and link usage distributions of second network.

selection to one which is the best possible for it, but not necessarily to other algorithms. Also, by allowing a little longer routes than the shortest routes could in some cases lead to better results.

4 Conclusions

The routing and wavelength assignment in WDM networks is not a straightforward task. The problem is NP-complete and heuristic algorithms must be used to find a practical solution. The problem can be divided to two phases: first one determines the used routes and then assigns wavelengths to the connections. This can be even done iteratively so that different route choices can be compared.

From the represented algorithms for wavelength assignment greedy algorithms and tabu search look most promising. However, if we are using some iterative method to find both the routing and coloring, like the one presented in this paper, greedy algorithms seem to be the only possible choice. The running time of tabu search (like all the other more sophisticated heuristics) increases much faster than the running time of greedy algorithms as a function of the number of nodes. The route selection algorithm used in this paper is very preliminary and leaves much room for improvement. This is a subject for future study.

References

[1] C.R. Reeves, Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill, 1995.

[2] V.J Rayward-Smith, I.H. Osman, C.R. Reeves and G.D. Smith, Modern Heuristic Search Methods. John Wiley & Sons, 1996.

[3] A. Hertz and D. de Werra, Using Tabu Search Techniques for Graph Coloring. Computing 39, 345-351, 1987.

[4] J. Mitchem, On various algorithms for estimating the chromatic number of graph. The Computer Journal, vol 19, 182-183.

[5] D. Brélaz. New methods to colour the vertices of a graph. In Communications of the ACM, volume 22, number 4, 1979.

[6] A.E. Willner, Mining the optical bandwidth for a terabit per second. IEEE Spectrum, April 1997.

[7] M. Berger, M. Chbat, A. Jourdan, M. Sotom, P. Demeester, B. Van Caenegem, P. Godsvang, B. Hein, M. Huber, R. März, A. Leclert, T. Olsen, G. Tobolka, T. Van den Broeck, Pan-European Optical Networking using Wavelength Division Multiplexing. IEEE Communications Magazine, April 1997.

[8] T.K. Tan and J.K. Pollard, Determination of minimum number of wavelength required for all-optical WDM networks using graph colouring. Electronic letters, vol.31, No.22, 1995.

[9] B. Mukherjee, D. Banerjee, S. Ramamurthy, A. Mukherjee, Some Principles for Designing a Wide-Area WDM Optical Network. IEEE/ACM Transactions on Networking, vol. 4, number 5, 1996.

[10] S. Baroni, P. Bayvel, Wavelength Requirements in Arbitrarily Connected Wavelength-Routed Optical Networks. Journal of Lightwave Technology, vol 15, number 2, 1998.

[11] S. Subramaniam, R.A. Barry, Wavelength Assignment in Fixed Routing WDM Networks. IEEE International Conference on Communications 1997, pp. 406-410.

[12] M. Garnot, M. Sotom, F. Masetti, Routing Strategies for Optical Paths in WDM Networks. IEEE International Conference on Communications 1997, pp. 422-426.

[13] D. Bertsekas, R. Gallager, Data Networks, Prentice-Hall, 1992.