Date: February 2, 1999

COST257TD(99)04

Wavelength Assignment in multifibre in WDM-networks

Esa Hyytiä and Jorma Virtamo

Helsinki University of Technology

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. In a single fibre case the the wavelength assignment is essentially a graph coloring problem once the routes are fixed. With multiple-fibres the problem changes a little and can no longer be presented as graph node coloring. Chosen routing has great impact to the final solution and an iterative algorithm for choosing the routes is briefly studied.

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 partial wavelength conversion. Later on this paper we consider networks where some links have more than one fiber. This is roughly equivalent with partial wavelength conversion when each wavelength can be converted to one or more other wavelengths. In some cases this can result as reasonable savings in the number of wavelengths.

We 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.

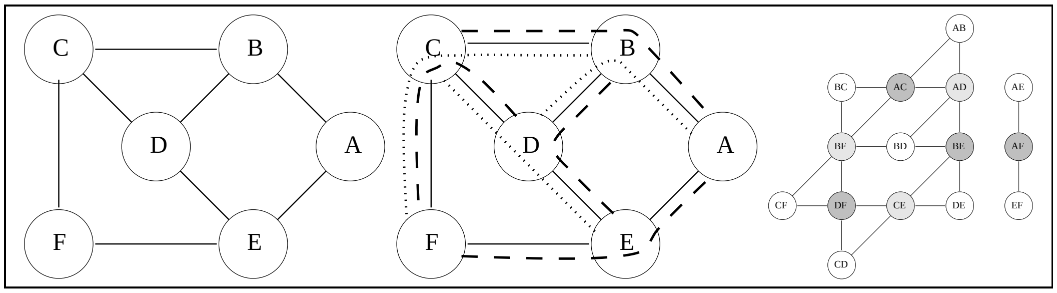

Figure 1: Example network and it is optimal configuration.

The process of routing and wavelength assignment is represented in figure 1 (single fibre case). 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 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]. The shortest path does not however always lead to an optimal configuration due the wavelength conflict requirement. The process of choosing the route candidates is presented in chapter 2.

Once the routing is fixed the problem is to minimize the number of used wavelengths. In chapther 3 we consider wavelength assignment in single fibre networks. The single fibre case can be mapped to a graph node coloring problem, which is a well-known NP-complete problem. In the chapter 3 several graph coloring algorithms are briefly discussed and tested. In chapter 4 a more general case of arbitary number of fibres is considered along. Finally, in chapter 6, we present some conclusions.

2 Choice of the routing candidates

All algorithms discussed here assume that they are given a set of possible routes for each required connection. Usually the best route is the shortest route, but sometimes for few connections it is better to use a little longer route to stay away from a heavily loaded link.

The routing algorithm searches the shortest routes between the given node pair. The algorithm is similar to Dijkstra's algorithm, i.e. at each step, it takes one step further from node A, and will eventually find the shortest path to node B. The algorithm has two tunable parameters shown on table 2. In other words, we can allow routes which are Δl

Parameter	**Explanation**
$rmin$	minimum number of route candidates per node pair
Δl	maximum difference in the lengths of accepted routes. If zero, only the shortest routes are accepted.

Table 1: Parameters controlling the choice of routing candidates.

hops longer than the shortest possible route and no more than $rmin$ route candidates are accepted. Routes with cycles are excluded naturally. In practice it seems to make sense to allow at least one link longer routes. In one example network the number of required wavelengths dropped by about 20%!

3 Wavelength assignment in single fibre networks

Assume, that we are given a fixed physical all-optical network where wavelength division multiplexing (WDM) is used to exploit the huge capacity of all-optical network. The network nodes are assumed to be incapable of wavelength conversions and each link consists of a single fiber. The problem is to configure the given connections into the network with minimum number of wavelengths. The problem is called routing and wavelength assignment problem (RWA) and it has been shown to be NP-complete.

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). With a proper choice of energy function SA works well with random graphs. A more recent heuristic algorithm called tabu search (TS) gives also good results with graph coloring problem [3]. The lightweight end of coloring algorithms are representd by greedy algorithms [4][5]. These algorithms will be discussed in more detail in the following.

Most of the heuristic algorithms have numerous parameters which must be tuned to particular problem.

3.1 Reduction of the given graph

If we are only interested in finding a feasible k-coloring for some graph, the graph can often be reduced. The reduction is possible in the following cases:

- If some node has less than k neighbors, the node and edges connecting it to other nodes can be removed as there will always be some color available for it.
- A node **A** can be removed if all its neighbor nodes are also neighbors of some node **B**, when any color which works for node **B** is legal for node **A** as well.

In practice the reduction did not seem to lead to essentially better results.

3.2 Graph Node Coloring with Greedy algorithms

Greedy algorithms are a very quick way to find usually decent coloring for a graph [4]. 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 first. Another variant of greedy algorithms, called DSATUR [5], dynamically chooses the next node according to number of possible colors per node.

Parameter	**Explanation**
bt	Use backtrack, 1 if yes, otherwise 0.
dsatur	Next node colored is one with fewest possibilities left.

Table 2: Greedy algorithms

3.3 Simulated annealing

Simulated annealing (SA) is a widely used heuristic algorithm for hard combinatorial problems[1][2]. It models the nature where a cooling system tends to move toward lower state of energy. The temperature defines the probability of accepting a move leading to a higher state of energy. During annealing temperature is brought gradually down. 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 goal of optimization is to minimize the number of colors which is thus a logical choice as energy function. Denote this energy function as E_{nr}. A straightforward way to solve the node coloring problem is thus following:

```
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 $\Delta E$.
6. If $\Delta E < 0$ or $e^{-\Delta E/T} >$ rnd(0,1) accept the change.
7. If there has been at least $M$ changes or $N$ trials, then set $T = \alpha \cdot T$ ($\alpha$ 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. It's clear that a coloring which has only few nodes using some color is more likely to be closer a better solution than a coloring which has colors used more equally. So it makes sense to favor moves leading toward a configuration with some colors used relative frequently and some colors only few times. This can be done at least in two different ways.

3.3.1 Choosing new colors from non uniform distribution

One way to achieve quicker convergence is to use non uniform distribution when choosing new colors. Let $\mathbf{F}$ be a frequency vector of colors, i.e. f_c tells us how many times color c is used currently. The sum over the elements of $\mathbf{F}$ is clearly the number of nodes n. Define $\mathbf{F}' = [\mathbf{F}\ 1]/(n+1)$. Now $\mathbf{F}'$ can be used as probability distribution for the new color of random node.

By using $\mathbf{F}'$ instead of uniform distribution we favor the most used colors and hope that we end up with fewer colors quicker than 'randomly' walking on constant energy plane. Both uniform and non-uniform distributions allow the choice of totally new color for the node. This might be needed to escape from local minimums.

3.3.2 Using alternative energy function

About the same functionality can be reached by using an alternative energy function E_{sq} defined as

$$E_{sq} := -\sum_c f_c^2 = -\mathbf{F} \cdot \mathbf{F}.$$

So E_{sq} is square of the length of frequency vector. It's clear that this energy function also favors configurations where some colors are used more frequently than others.

But the optimal coloring in the sense of E_{sq} is not necessarily same as with E_{nr}. The optimal coloring in the sense of E_{nr} is however a local minimum for E_{sq} and we are hoping that the local minimum is visited during the annealing process. So it's worth storing the best coloring in the sense of E_{nr} during annealing proces and returning that instead of the coloring at the end of annealing.

Parameter	**Explanation**
T0	The starting temperature. If negative the system is first heated till the probability of accepting a move leading to higher state of energy is about half.
T1	The temperature when annealing is ended.
alpha	The constant *alpha* determines the rate of cooling: $T_{n+1} := \alpha \cdot T_n$.
Ef	Used energy function, either *color*, *inverse* or *square*.
Nd	New color distribution, defines how a new random color is chosen. Either from uniform distribution (*uniform*) or by favoring frequently used colors (*freq*).

Table 3: Parameters of simulated annealing.

Third possibility for energy function is the sum of inverses:

$$E_{in} := \sum_{f_c>0} 1/f_c = \sum_{f_c>0} f_c^{-1},$$

which pays more weight on removing seldom used colors. It does not really matter if some color is used n or $n+1$ times, when n is large, but moving from 2 to 1 is likely to lead to a drop of one color.

The implementation of this energy function does not need any divide-operations either, as inverses of possible values of color usage can be calculated in advance. Sample runs suggest that this energy function works very well with random graphs. Table 3 summaries the parameters of SA algorithm.

3.4 Tabu-search

Tabu search (TS) is a relatively new heurictic method [1][2][3]. 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 $s = (V_1, \dots, V_k)$ 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 $s = (V_1, \dots, V_k)$.
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 found 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. Only nodes which have a neighbor colored with same color are included. 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.

The tabu search algorithm is quite robust and works well with graph node coloring problems. Table 4 lists possible parameters with their default values.

Parameter	**Explanation**
len	length of the tabu list.
nrep	number of neighbors searched on each round.
nbmax	number of rounds the algorithm is run.

Table 4: Tabu search parameters.

On each round the algorithm tries *nrep* possible moves and picks the best of them unless it's considered tabu. The reverse move of chosen move is stored in the tabu list and oldest entry is removed from it. Even if a move is classified as tabu it's chosen if it leads to the best configuration found so far (admiration).

3.5 Other possible approaches

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, among all the perfect graphs obtained the one

Parameter	Explanation
save	The number of elements saved on each level.
alg	Algorithm used to evaluate the subtree.
llimit	Whatever to utilize or not lower limits.

Table 5: Pruned search.

with the smallest number of nodes is chosen. As the number of nodes increases, the size

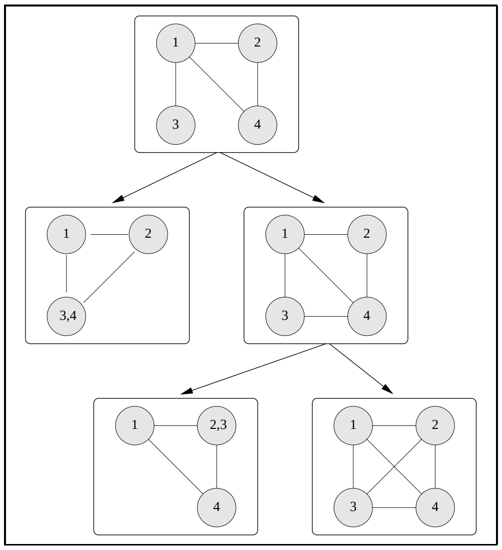

Figure 2: Coloring the graph with exhaustive search.

of the search tree clearly becomes too large. 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.

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.

3.6 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

Parameter	Explanation
rounds	The number of generations run.
size	The size of the population.
mutation	The probability of mutation operation.

Table 6: Parameters of Genetic Algorithm routine.

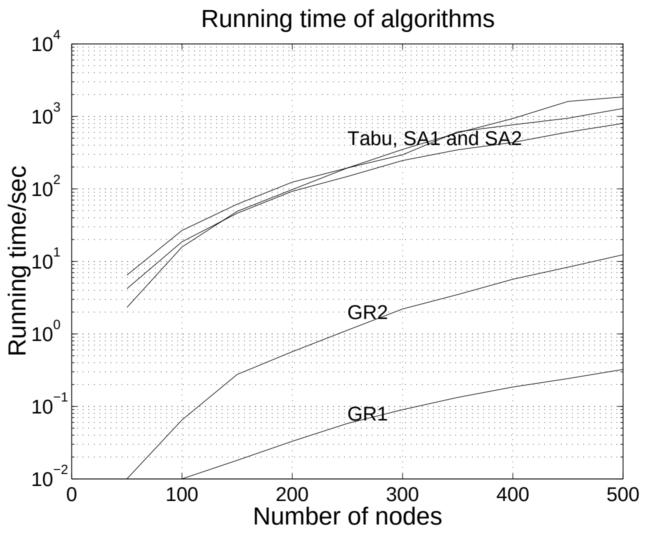

Figure 3: Running times of graph coloring algorithms.

running time of each algorithm. It should be noted that the used parameters have a great effect on the running times and the final results of certain algorithms. We tried to choose the parameters so that running times would be somewhat similar with each other. So these figures should be considered as examples only. It is not clear either how well random graphs match with actual graphs arising from real networks.

The algorithms can be grouped into two classes according to their running times (see figure 3). Greedy algorithms are clearly very fast and heuristic algorithms SA, GA and TS take more time to run. The exhaustive search (and pruned version) are not shown at all as their running times would have been enormous. With regard to the optimization result (see figure 4) the tabu search was the best, but also the simulated annealing gave very good results. The use of *inverse*-energy function with SA seems to be a better choice than *square*-energy function when coloring random graphs.

4 Routing and wavelength assignment in multifibre networks

This chapter describes the implementation of simple algorithm, which solves both routing and wavelength assignment (RWA) for arbitary network. Network may have several fibres between certain nodes. We assume that the network is required to be fully-connected. Similar heuristics as in simple case can be used here but the problem no longer maps easily onto a graph node coloring problem if there is at least in one link more than one fibre.

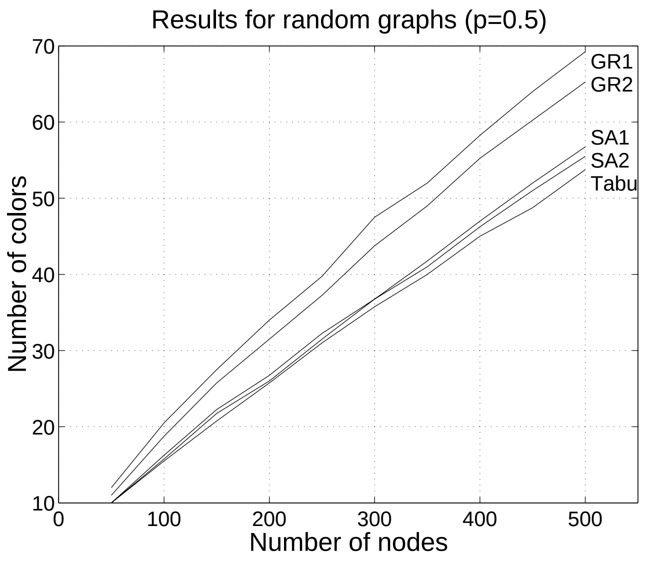

Figure 4: Number of used colors with different algorithms.

4.1 Definitions

Assume that route candidates are given (ch. 2). Thus we are given a $nr \times nl$ binary matrix $\mathbf{R}$ defining links each route candidate uses, where nr is the number of routes (in total) and nl is the number of links in physical network. So each column of matrix R represents a link and each row a possible route for some connection. Element r_{ij} is one if route i goes through link j, otherwise zero.

We are also given a $nc \times 2$ matrix $\mathbf{Q}$, where nc is the number of connections to be configured into the network. The matrix $\mathbf{Q}$ has the information about which routes belong to which connection. The element q_{i1} is the location of the first route for connection i in matrix $\mathbf{R}$, and the element q_{i2} is the number of route candidates for connection i.

From those candidates we are supposed to choose a subset (rows from matrix $\mathbf{R}$) containing one route for each required connection.

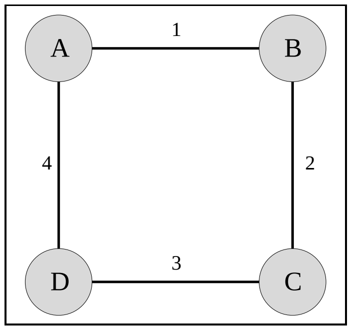

Figure 5: A sample network.

In figure 4.1 there is a simple symmetric network of four nodes. It's supposed to be fully-connected, i.e. each node is connected to each other, making total of 6 connections. Now only connections between opposite corners can be routed in more than one way if we take account only the shortest paths. Hence, for this network we get the following matrices:

$$\mathbf{R} = \begin{pmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \\ R_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{Q} = \begin{pmatrix} 0 & 1 \\ 1 & 2 \\ 3 & 1 \\ 4 & 1 \\ 5 & 2 \\ 7 & 1 \end{pmatrix}.$$

So for example the first connection between the nodes A and B uses only link 1 ($\mathbf{R}_1$) as nodes are neighbors in the network, but for the second connection between the nodes A and C there are two possible routes: through links 3 and 4 or through links 1 and 2. In matrix $\mathbf{R}$ this is $\mathbf{R}_2$ and $\mathbf{R}_1$.

4.2 Multiplicity

It is possible that some links have more than one fibre. So we assume that a vector $\mathbf{M}$ (multiplicity) defines the number of fibres per link. It's length is clearly nl. If there is only one fibre on each link the vector $\mathbf{M}$ is simply a vector where each element is one. For the sample network we would have

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}.$$

4.3 Routing and wavelength assignment

Now we know the given physical network and the connections which are suppose to be configured into it. Let p_j denote the index of route chosen for the connection j. Our job is to find a feasible configuration (a route and a wavelength for each connection) with as few wavelengths as possible.

Suppose that we have chosen some set of routes for connections (rows from matrix $\mathbf{R}$), and denote those connections with $\{\mathbf{R}_{p_i}\}$, $i = 1, \ldots, nc$. We have also fixed wavelength $c_i \geq 1$ for each connection i. The greatest wavelength used $c_{max} = \max_i c_i$ is the function to be optimized.

The link usage for some color c is

$$\mathbf{U}_c = \sum_{\{i|c_i=c\}} \mathbf{R}_{p_i},$$

and we get a $nc \times nl$ usage matrix $\mathbf{U}$

$$\mathbf{U} = \begin{pmatrix} \mathbf{U}_1 \\ \mathbf{U}_2 \\ \vdots \\ \mathbf{U}_{cmax} \end{pmatrix},$$

which tells us how many connections are using certain wavelength on certain link. That is u_{cj} is the number of connections using wavelength c on link j.

In order to avoid wavelength conflicts we must have

$$u_{cj} \leq m_j \quad \forall\, j = 1 \ldots nl,\ c = 1 \ldots c_{max}. \tag{1}$$

This means, that the elements of each row of the matrix $\mathbf{U}$ must be smaller than or equal to the elements of the multiplicity vector $\mathbf{M}$, i.e. no link may have more than m_j connections using the same wavelength. Another way to say the requirement is that for each c the vector $(\mathbf{M} - \mathbf{U}_c)$ may have no negative elements.

Objective function was the number of wavelength c_{max}, so essentially we try to minimize the size of the usage matrix $\mathbf{U}$ without violating the feasibility requirement (1).

The sum over all the elements of the matrix $\mathbf{U}$ is equal to the number of the links the configuration uses. If we have limited the route candidates to include only the shortest routes the sum is constant. So the problem is to 'pack' matrix U.

The problem can be attacked with well-known heuristics like simulated annealing, genetic algorithms and tabu search. In this paper a simple greedy approach is used to get some results when comparing single fibre case to multi fibre case.

4.4 Lower limits

If we limit the search space only to shortest possible paths, no matter what route is chosen to each connection, it will use same capacity from global network (number of links the shortest route contains). Let $\bar{L}$ be average length of routes, so the average number of connections per fibre is

$$\frac{nc\bar{L}}{\sum_j m_j}$$

from what its clear that [14]

$$c_{max} \geq \left\lceil \frac{nc\bar{L}}{\sum_j m_j} \right\rceil$$

Once the routing is fixed (route p_i for connection i) the number of connections per link is known and we get

$$c_{max} \geq \max_j \left\lceil \frac{\sum_i r_{p_i i}}{m_j} \right\rceil,$$

that is c_{max} must be greater or equal than the number of connections on any link divided by links multiplicity. Neither of these limits take account the wavelength conflicts and might be poor estimates in worst cases.

5 Examples

In figure 6 there are two hypothetical physical networks residing in Finland. Several configurations are briefly studied with these networks. The used routing 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. A more sophisticated method is probably worth trying.

The first network is easy and our algorithm finds the optimal solution of 10 wavelength in single fibre case. So including more route candidates or doubling the the numbder of fibres can not lead any better result.

The larger network represents a harder problem. In single fibre case algorithm finds a configuration of 64 wavelengths, which can not be considered as a good result. By

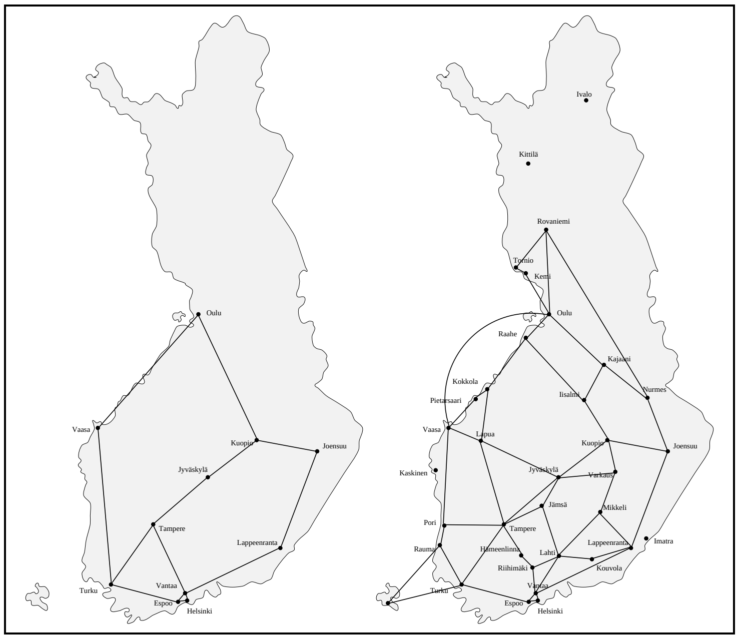

Figure 6: Hypothetical WDM-network in Finland.

including the one hop longer routes the number of required wavelengths dropped by 14. When including routes with one hop more than shortest possible When instead of including one hop longer routes we doubled the fibres in each link the algorithm found a solution of 31 wavelengths.

network	**nodes**	**single**	**single, +1**	**multi**
network I	11	10	10	5
network II	31	64	50	31

Table 7: Results with example networks.

Overall one could assume that a better heuristic algorithm in selecting the routes could probably lead to both quicker and better solutions. Including a little longer routes than the shortest routes can lead to dramatically better results like what happened with the second network. Installing another fibre into each link does not seem to be worth it in example cases. Maybe if the network were dynamically configured by adding a new lightpath without changing current ones, the effect of several fibres per link would be greater. These are only two examples and one must be careful when making conclusions.

6 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 routings can be compared. The wavelength assignment algorithm should be rather quick in that case.

The effect of including one hop longer routes was demonstrated. In sample network it gave about 20% save in the number of wavelengths which is considerable high. On the other hand, the doubling the number of fibres on each link does not seem to have great affect in the results if the multiplication of the number of wavelengths and the number of fibres per link is used as the measure.

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.

[14] S. Baroni, Routing and wavelength allocation in WDM optical networks, Ph.D. Thesis, University College London, May 1998.