Optimal Fibre-Optic Cable Layout using Dynamic Programming

M. L. O'Sullivan
Department of Engineering Science
University of Auckland
New Zealand
mosu002@esu1.auckland.ac.nz

Abstract

This paper details a dynamic programming approach for solving a network covering problem. The network is based on fibre optic designs used by TelstraSaturn to provide Central Business District (**CBD**) customers with a range of communication services. The network is generated by software, called **FIDO**, and the dynamic program is compared with the bin packing heuristic employed by **FIDO** to find a solution to cover the network. The results show that the dynamic program gives significantly improved solutions for a range of networks and costs. Tradeoffs between optimality and solution speed are also discussed.

1 Introduction

In 2000 the newly formed TelstraSaturn announced plans to construct a fibre-optic network with the capability of providing broadband access to 65% of the homes and 80% of the businesses in New Zealand. In the plan the network is composed of closed subnetworks, each describing a large area of demand, such as the Central Business District (**CBD**) of a major centre. The subnetworks are attached to exchange buildings and these exchange buildings are connected with each other to form the total network.

The CBD subnetworks are composed of four elements; *feeders*, *vaults*, *buildings*, and *streets*. The feeders are trenches which contain the underground fibre-optic network, starting at the exchange building. The network consists of fibre-optic pairs (pairs are used so that each fibre only carries the signals in one direction). The fibre pairs are grouped into *tubes* for easier manipulation, and then the tubes are packed in protective sheaths, called *covers*. At both ends of a feeder are vaults, which are access areas where workers can perform welding operations to connect fibre pairs into different covers.

The buildings are where the customers reside, and therefore, where the demand arises. The demand from all customers in a building is grouped together so that for each building there is only a single demand. A building will also be classed as needing either a *high-speed* or a *low-speed* connection. A high-speed demand building

must contain a multiplexor, while a low-speed demand building can be connected to a nearby multiplexor with copper wire. All the multiplexors in the buildings of a street are connected to vaults at each end of the street using a fibre-optic pair.

TelstraClear wished to make a network that provided its customers with reliability. To do this they specify the condition that all customers must be connected to the exchange building by two diverse paths. This means that every multiplexor is connected to a vault at each end of a street by a fibre pair.

The Fibre Diversity Optimiser (**FIDO**) is a network design system that was developed by Mason and Philpott for TelstraSaturn (Mason and Philpott 2001). **FIDO** separates the problem of constructing a network into three separate problems, the *path optimisation*, *street optimisation*, and *cable optimisation*.

The path optimisation uses subproblems to generate two diverse paths from each street to the exchange. The paths for an individual street are generated by solving a network flow problem, which uses all the vaults in the network as nodes and all the feeders as arcs. The vaults at each end of the streets are assigned a supply of 1, while the exchange vaults are each given a demand of 1, and all the feeders are assigned a capacity of one, thus ensuring that the solution will give two diverse paths from the vaults at each end of the street back to the exchange.

The street optimisation determines how many fibres are required to be connected to each of the vaults at the end of a street. Every high-speed demand building must have a multiplexor, which is connected by a fibre pair to each vault. However, a low-speed user may choose to be connected to the multiplexor of a nearby building using copper cable. Although copper cable is more expensive than fibre per metre, by using copper to connect to a nearby multiplexor, a small length of copper wire may replace the need to have an extra multiplexor and two extra fibre pairs that would have run all the way to the exchange. The street optimisation is solved by using dynamic programming on a lot sizing program. The details for both the path and street optimisations are given more thoroughly in (Mason and Philpott 2001).

The cable optimisation involves deciding how to pack the backbone of the network (which is in the feeders) into protective covers. This involves deciding which covers to pack the fibre pairs into for each feeder in the network. Welding operations are also required whenever new covers are made or when the covers from one feeder differ to an adjacent feeder (see Figure 1). For the purpose of maintenance, TelstraSaturn imposed the condition that once fibre has been placed into a cover, the cover may not be split closer to the exchange. This creates a logical tree of fibres from the network (see Figure 2). The branches of this tree represent groups of feeders and are called *logical cables*. **FIDO** uses a first-fit decreasing bin-packing approach to place the fibre into covers, which is explained in (O'Sullivan 2002). However, this approach does not take account of welding and cover costs to find the least expensive way to pack the fibres into covers. This paper describes a dynamic programming model for computing the minimum cost for covering the network.

2 Dynamic Programming Approach

The dynamic programming approach is based around the conceptual picture of moving from the demand vaults to the exchange vaults. By thinking of the network this way and including the condition that cables cannot be split, only two types of decision are possible.

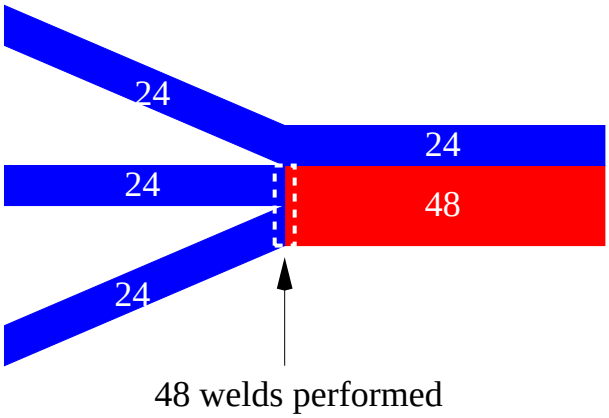


Figure 1: This figure shows three covers of size 24 meeting a single cover of size 24 and a cover of size 48. Because of the difference in covers 48 welds must be made to connect two 24s to the 48.

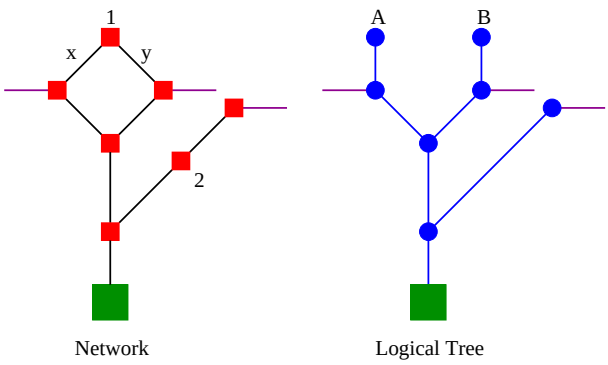


Figure 2: The difference between a network and its logical tree. Node 2 is removed and the 2 feeders connected to it are turned into a single cable. Feeders x and y both connect in to node 1 in the network but their corresponding cables A and B are considered to have separate tails in the logical tree.

1. The original configurations of covers must be created wherever demand occurs and when two logical cables meet the covers may be welded together.

2. The key property that makes dynamic programming a faster approach is that the only cables affecting the covers on the current cable are those immediately prior to it (see Figure 3).

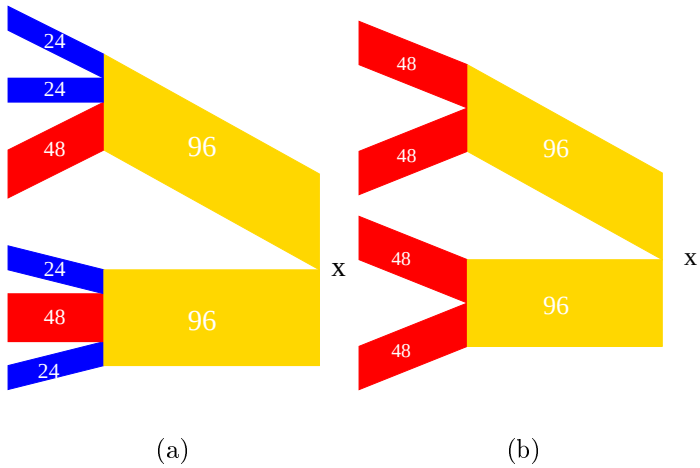


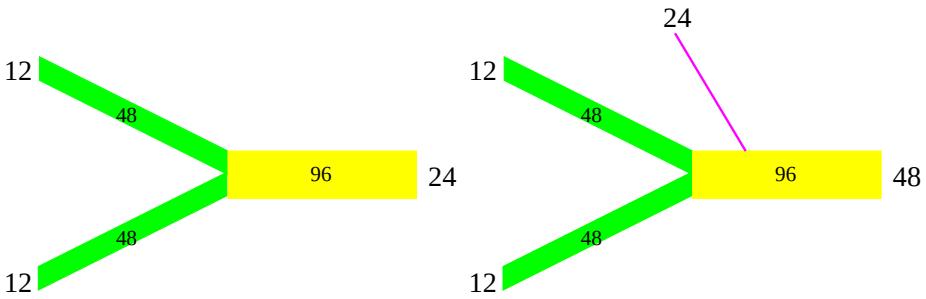
Figure 3: Figures (a) and (b) show different covers for a piece of a network. Even though different covers are used, the choice at point x is the same for both cases, how to cover the fibre in 2 covers of size 96.

2.1 Welding

For this dynamic program all the decisions that are made are based on how to weld fibre into covers. When making the decision on how to cover the fibre on a logical cable there are two types of fibre to consider: fibre coming from child cables that is already in covers and uncovered fibre from streets. When welding covered fibre together we can only join covers. The new street fibre, however, can be placed into space in existing covers through a process called *side-welding* (see Figure 4).

The dynamic program's recursion is based upon finding the optimal way to cover the cables that are joined to the exchange. Because the choice of covers is effected by the covers of child cables, a recursion is created where leaf cables (which have no children) must be covered first, and then all other cables have their covers selected in a depth first order. For each choice of covers for a child cable, there will be different choices for the covers that the parent cable can use. Unfortunately, if the dynamic program is constructed this way the number of states becomes unmanageably large rapidly. Figure 5 shows that even a very small network would have a a large number of states.

However, the observation is made that a cable will only have a limited selection of possible covers for all the configurations of the child cables' covers. This means that the states on the dynamic program can be the single set of possible cover a cable may have.



(a) Cover welding only - 2 Covers of size 48 must weld to a cover of size 96 even though there are only 24 fibres.

(b) Cover welding with side welding - The new street fibre can be put into space in the covers.

Figure 4: The difference between cover welding and side welding.

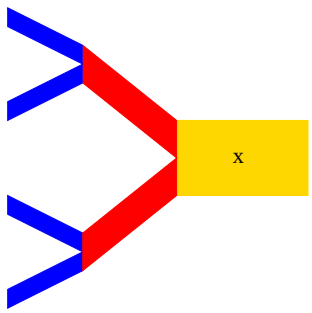
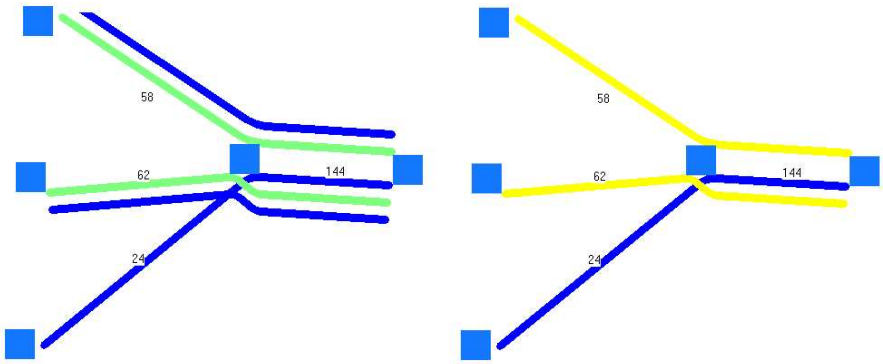


Figure 5: If each cable in this network had 10 possible covers then generating different state for a cable for each choice of child cable would mean that cable x would have 10,000,000 states.

2.2 Finding the state space

The states of the dynamic program are the possible covers that a cable can use. However, because of the condition that covers may never be split, we cannot generate the possible covers from the fibre count of a cable (see Figure 6). The size of covers needed by a cable relies on the covers that its child cables use. To be able to find the states of a cable we must first introduce the ideas of minimum and maximum fibre capacities for a fibre count.



(a) This solution covers the final cable with 3 size 24 covers (black) and 2 size 48 covers for a total fibre cover of 168.

(b) This solution uses 2 size 96 covers and a 24 for total fibre cover 216 but is cheaper that the first solution.

Figure 6: Two figures showing different way a cable with fibre count 144 can be covered.

2.2.1 Minimum Fibre Capacity

To find the cover choice with the minimum capacity for a particular fibre count, f, we proceed as follows. Let the number of cover sizes be m and $\mathbf{p} = [p_i], i = 1, \ldots, m$, be an integer vector of given cover sizes. We denote a cover choice by an integer vector $\mathbf{s} = [s_i], i = 1, \ldots, m$, where s_i =the number of covers in $\mathbf{s}$ of size i. Let $S(f)$ be the set of cover choices for the fibre count f:

$$S(f) \equiv \{\mathbf{s} \in \mathbb{Z}^m | \mathbf{p}^\top \mathbf{s} \geq f\}$$

The fibre capacity of a cover choice is $\mathbf{p}^\top \mathbf{s}$, and the minimum fibre capacity for a fibre count f is:

$$\min\{\mathbf{p}^\top \mathbf{s} | \mathbf{s} \in S(f)\}$$

2.2.2 Maximum Fibre Capacity

The only reason that a cable would ever use a set of covers with a greater total cover size than the minimum fibre capacity is if it is cheaper to do so. Let $\mathbf{c}$ be the vector of costs per unit length for the cover sizes. That means that the cost per unit length for a cover choice can be expressed as:

$$\mathbf{c}^\top \mathbf{s}$$

Therefore, the maximum fibre capacity for fibre count f is:

$$\mathbf{p}^\top \mathbf{s}_{\max}$$

where

$$\mathbf{c}^\top \mathbf{s}_{\max} \leq \mathbf{c}^\top \mathbf{s}, \forall \mathbf{s} \in S(f)$$

If $\mathbf{s}_{\max}$ is not a unique minimiser of the cost then the cover choice with the greater fibre count is used. In other words, given $\mathbf{s}_1$ and $\mathbf{s}_2$, such that:

$$\mathbf{c}^\top \mathbf{s}_1 = \mathbf{c}^\top \mathbf{s}_2 \leq \mathbf{c}^\top \mathbf{s}, \forall \mathbf{s} \in S(f)$$

then

$$\mathbf{s}_{\max} = \begin{cases} \mathbf{s}_1, & \mathbf{p}^\top \mathbf{s}_1 \geq \mathbf{p}^\top \mathbf{s}_2, \\ \mathbf{s}_2, & \text{otherwise.} \end{cases}$$

2.2.3 Generating states for a cable

To find the covers that a cable may utilise, we must first find its minimum and maximum fibre counts based on the capacities of its children. To find the minimum capacity of a cable we first sum the minimum fibre counts of its children. Given the set of incoming children, K, we represent this sum as:

$$\sum_{k \in K} \mathbf{p}^\top \mathbf{s}^k_{\min}$$

We then check how much of the new street fibre we can pack into the covers. If f^k is the actual fibre count on child cable k, then the street fibre that cannot be packed into the covers is:

$$f_{\text{rem}} = \max(f_{\text{new}} - \sum_{k \in K} (\mathbf{p}^\top \mathbf{s}^k_{\min} - f^k), 0)$$

This means that the minimum fibre count the parent cable must cover is:

$$f_{\min} = \sum_{k \in K} \mathbf{p}^\top \mathbf{s}^k_{\min} + f_{\text{rem}}$$

Calculating $\mathbf{s}_{\min}$ for this fibre count will give the minimum fibre capacity.

To find the maximum capacity of a cable we first sum the maximum fibre capacities of the incoming children:

$$\sum_{k \in K} \mathbf{p}^\top \mathbf{s}^k_{\max}$$

To give a valid bound we assume that no side welding occurs. This gives the maximum fibre count for the parent cable:

$$f_{\max} = \sum_{k \in K} \mathbf{p}^\top \mathbf{s}^k_{\min} + f_{\text{new}}$$

By finding $\mathbf{s}_{\max}$ for this fibre count we find the maximum fibre capacity.

The cable can use any cover choice which has a fibre capacity between the minimum and maximum. For any cable we generate as states all cover choices, $\mathbf{s}$, that satisfy:

$$\mathbf{p}^\top \mathbf{s}_{\text{min}} \leq \mathbf{p}^\top \mathbf{s} \leq \mathbf{p}^\top \mathbf{s}_{\text{max}}$$

where

$$\mathbf{s}_{\text{min}} = \text{argmin}\{\mathbf{p}^\top \mathbf{s} \mid s \in S(f_{\text{min}})\}$$

and

$$\mathbf{s}_{\text{max}} = \text{argmin}\{\mathbf{c}^\top \mathbf{s}, \forall s \in S(f_{\text{max}})\}$$

2.3 Generating the costs

The costs of the network is made up of two distinct parts, the cost of creating the covers that are used on a cable (the *cover cost*) and the cost of welding incoming cables into the covers that are to be used, which is called the *weld cost*. Given the length of a cable, l, then the cover cost of a particular cover choice, $\mathbf{s}$, is:

$$\mathbf{c}^\top \mathbf{s} \times l$$

When calculating the weld cost we only consider the cost of welding fibres from covers that have been cut. The cost of welding street demand fibre (including side welding) is incorporated into the street optimisation. Suppose we have k incoming cables. The incoming covers can be represent by the sum:

$$\mathbf{s}_{\text{in}} = \sum_{i=1}^{k} \mathbf{s}_i, \mathbf{s}_i \in S_i, i = 1, \ldots, k$$

Given the state of the current cable, $\mathbf{s}_{\text{curr}}$ and the new fibres, f_{new}, we can check for situations where welding cannot combine the input into the current cable's state. Since we are only only allowed to weld covers together we know:

$$\mathbf{p}^\top \mathbf{s}_{\text{curr}} \geq \mathbf{p}^\top \mathbf{s}_{\text{in}} + f_{\text{new}} \tag{1}$$

Another important property is that the number of covers of the current state must be less than or equal to the number of covers coming in plus the number of new covers that can be generated:

$$\mathbf{1}^\top \mathbf{s}_{\text{curr}} \leq \mathbf{1}^\top \mathbf{s}_{\text{in}} + \left\lceil \frac{f_{\text{new}}}{\min(p_i)} \right\rceil \tag{2}$$

If any welding occurs a fixed weld cost is incurred. To model this we use the function:

$$\delta(\mathbf{s}_{\text{curr}}, \mathbf{s}_{\text{in}}) = \begin{cases} 0, & \mathbf{s}_{\text{curr}} - \mathbf{s}_{\text{in}} = \mathbf{0} \\ 1, & \text{otherwise} \end{cases}$$

The number of fibres that must be welded is calculated by removing the covers that continue without welding from $\mathbf{s}_{\text{in}}$:

$$\mathbf{s}_{\text{weld}}(i) = \mathbf{s}_{\text{in}}(i) - \min(\mathbf{s}_{\text{in}}(i), \mathbf{s}_{\text{curr}}(i)), i = 1, \ldots, m$$

If either of the inequalities 1 or 2 do not hold then the cost will be infinite. Therefore, we can express the weld cost as:

$$\begin{aligned} w(\mathbf{s}_{\text{curr}}, \mathbf{s}_{\text{in}}) = & \mathbf{p}^\top \mathbf{s}_{\text{weld}} \times w_{\text{unit}} + \delta(\mathbf{s}_{\text{curr}}, \mathbf{s}_{\text{in}}) w_{\text{fixed}} \\ & + \infty \times \max((\mathbf{p}^\top \mathbf{s}_{\text{in}} + f_{\text{new}} - \mathbf{p}^\top \mathbf{s}_{\text{curr}}), 0) \\ & + \infty \times \max((\mathbf{1}^\top \mathbf{s}_{\text{curr}} - \mathbf{1}^\top \mathbf{s}_{\text{curr}} - \left\lceil \frac{f_{\text{new}}}{\min(p_i)} \right\rceil), 0) \end{aligned}$$

2.4 Finding the minimum cost

To find the minimum cost of a cable we use complete enumeration to solve the following linear program:

$$\begin{aligned} \text{minimise: } z(\mathbf{s}_{\text{curr}}) &= \mathbf{c}^\top \mathbf{s}_{\text{curr}} \times l + w(\mathbf{s}_{\text{curr}}, \mathbf{s}_{\text{in}}) + \sum_{i=1}^{k} z(\mathbf{s}_i) \\ \mathbf{s}_{\text{in}} &= \sum_{i=1}^{k} \mathbf{s}_i \\ \forall \mathbf{s}_i &\in S_i, i = 1, \ldots, k \end{aligned}$$

The cost of the minimisation involves $z(\mathbf{s}_i)$, the costs of the child cables. At the leaves (cables with no children) the cost for a state is just the cover cost:

$$z(\mathbf{s}_{\text{curr}}) = \mathbf{c}^\top \mathbf{s}_{\text{curr}} \times l$$

The minimum cost for the entire network is the sum of the minimum costs for each cable that joins onto the exchange. Because the minimum cost of a cable includes the cost of all the child cables, by finding the minimum cost of the cables linked to the exchange, the dynamic program performs a depth first recursion, finding the minimum cost for leaf cables first, and then stepping through until it gets to the exchange.

2.5 Improving Performance

The dynamic programming recursion is very time consuming. One way to improve the performance is to examine the number of covers a cable uses. Although feasible, it does not make much practical sense to allow a cable to use a large number of covers. A large number of covers will fill up a feeder, and may even cause extra work to allow all the covers to be placed. The economics of scale mean that states with a large number of covers will generally not be used in a solution and their inclusion only slows down the solution speed. A user-defined parameter, b, is introduced, which is the maximum number of covers allowed per cable. When a cover choice, $\mathbf{s}$, is generated for a fibre count, f, it must now satisfy the condition:

$$\begin{aligned} b_{\min} \leq \mathbf{1}^\top \mathbf{s} \leq b,\ b_{\min} < b \\ \mathbf{1}^\top \mathbf{s} \leq b_{\min},\ b_{\min} \geq b \end{aligned}$$

where

$$b_{\min} = \left\lceil \frac{f}{\max(p_i)} \right\rceil$$

i.e. the number of the largest cover needed to pack all of the fibre on the cable.

2.6 Concerning Optimality

The dynamic formulation finds the optimal solution for the logical tree used to represent the network. Unfortunately, optimality cannot be guaranteed for the actual network. This is because of the fixed cost for welding. The logical tree can have two cables both paying for a fixed weld when welding actually only occurs at a single node (see Figure 7). An algorithm has been developed to cost the solutions correctly

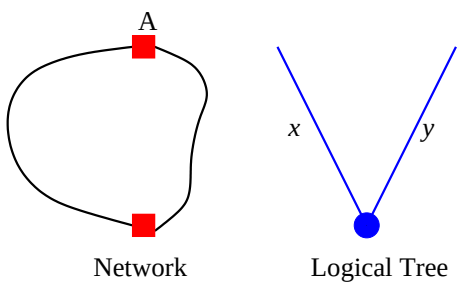


Figure 7: This figure shows how a single node, A, can be represented by separate nodes in the logical tree. If welding occurs at the tail of both cable x and cable y then the dynamic program will pay twice for the fixed cost of the welds. However, in the real network both of the welding processes occur at A, so only 1 fixed weld cost should be payed for.

once they are found, but, because the dynamic program is based on the logical tree, this fix cannot be included in the dynamic program's costing. This means that some states might be falsely rejected because they are credited with a fixed cost that they do not incur in the real network. Even though the practical fixed cost is very small compared to the cost of the cables, and unlikely to actually change the solution, the dynamic program cannot guarantee optimality.

3 Results

The dynamic program was implemented using Microsoft Visual C++. The problems were all solved on an AMD 1600XP processor with 512 MB of DDR ram. The test data consisted of four sets of costs and two different CBD subnetworks. All of the costs used the cover data in Table 1. The fixed cost of welding was $400 per node and the cost of welding each fibre pair is $36.

3.1 Tubification

The fibre pairs are all placed into tubes for easier manipulation, as mentioned in the introduction. The network can use different sizes of tubes and may also tubify at different places. The tubification can occur at either the street or vault level. Tubifying at the street means that demand for each street is rounded up to a multiple

Size of Cover	Cost Per Unit Length($)
24	7.84
48	11.68
96	17.44
144	21.28

Table 1: The available cover sizes and their cost per unit length.

of the tube size. Tubifying at the vault level means that the demand from all the streets at a vault are summed and then this sum is rounded up to a multiple of the tube size.

The cost files used differ only in the tubification used and the size of the tubes used. As stated in the dynamic programming section, the tubification can either occur at the street level or the vault level, and different sized tubes can be used. Table 2 summarises the tubifications that the cost files use.

Cost File	Tubification
`FibCost1,0`	Street Tubification in tubes of size 1
`FibCost1,1`	Tubification at both the street and vault in tubes of size 1
`FibCost1,12`	Street tubification in tubes of size 1, vault tubes size 12
`FibCost12,0`	Street tubification in tubes of size 12

Table 2: The different tubifications used by the cost files..

3.2 The Auckland CBD

The first subnetwork used was based on the Auckland CBD. This subnetwork has 225 feeders and 234 streets. The network has a total requirement of 1040 fibre pairs.

Number of covers allowed	`FibCost1,0`		`FibCost1,1`		`FibCost1,12`		`FibCost12,0`	
	% Improve	Time (m)	% Improve	Time (m)	% Improve	Time (m)	% Improve	Time (m)
1	-6.09	0.20	-6.09	0.20	0.79	0.03	2.43	0.03
2	4.29	0.28	4.29	0.28	4.64	0.05	6.92	0.05
3	5.70	2.17	5.70	2.15	6.56	0.96	7.83	0.28
4	6.33	582.85	6.33	582.27	7.24	16.82	7.97	14.70
5	6.70	1100.96	6.70	1102.06	7.42	446.45	8.26	426.33

Table 3: % improvement of the dynamic program over the heuristic and solution time for the Auckland CBD.

3.3 The Wellington CBD

The second subnetwork was the Wellington CBD. This CBD is geographically narrow, and has only 166 feeders and 132 streets. However, the customer demand is higher, with a total of 3396 fibre pairs required.

Number of covers allowed	FibCost1,0		FibCost1,1		FibCost1,12		FibCost12,0	
	% Improve	Time (m)	% Improve	Time (m)	% Improve	Time (m)	% Improve	Time (m)
1	2.71	0.23	2.71	0.23	2.36	0.02	3.32	0.03
2	5.34	0.24	5.34	0.24	5.34	0.03	5.42	0.03
3	5.71	0.30	5.71	0.30	6.25	0.07	6.11	0.08
4	5.80	0.99	5.80	0.99	6.35	0.37	6.13	0.29
5	5.84	6.45	5.84	6.38	6.38	4.32	6.13	1.62

Table 4: % improvement of the dynamic program over the heuristic and solution time for the Wellington CBD.

The results for both the subnetworks show that the dynamic program can give significant improvements over the bin-packing heuristic. In fact, the only occasion for which the dynamic program did not give improvements was when it was restricted to using only one cable whenever possible. In fact, the two networks that were used for the test data, though based on real data, both had properties that give the dynamic program less of an advantage. The Auckland CBD is physically spread out but has very low demands. This means that there are many covers, so that, until the cover limit is high, the dynamic program is forced to weld covers together. The Wellington network is physically very narrow, while having extremely high demand. This means that there is often little choice on how the fibre can be packed. The dynamic program should produce even better results when used on networks that are not as extreme as the test data.

The results also show the benefit of having the the user set a limit on the number of covers allowed per cable. As the user increases the cable limit the marginal improvement of the solution is decreasing while the solution time is increasing at a very high rate (faster than exponential). In practice, when designing a network, the cover limit could be set to 3 or less to obtain a solution quickly. Once a network has been decided on the dynamic program can be run with a high cover limit to give an even more cost-efficient solution.

References

Mason, A. J., and A. B. Philpott. 2001. "Development of FIDO - A Network Design Tool for Fibre Cable Layout." *ORSNZ Conference Twenty Naught One.* ORSNZ.

O'Sullivan, M.L. 2002. "Optimal Fibre-Optic in Cable Layout using Dynamic Programming." Master's thesis, Univeristy of Auckland, New Zealand.