INFORMATICA, 1996, Vol. 7, No. 4, 455–468

UPGRADING LINKS FOR PERFORMANCE*

Andrew LIM

Information Technology Institute & Dept. of Inf. Sci. and Computer Sci.
11 Science Park Road, Singapore 0511

Yeow-Meng CHEE

Dept. of Computer Science University of Waterloo, Canada

Wynne HSU

Dept. of Inf. Sci. and Computer Sci., Singapore 0511

Abstract. The performance of a computer network is commonly measured by the maximum minimum time required to move a certain amount of data between any 2 nodes in the network. Due to the advances in technology, certain links in the network may be upgraded, for instance to optical fibre links, so that better performance can be achieved. In this paper, we study the LINK UPGRADE problem for networks. We first show that the LINK UPGRADE problem is $\mathcal{NP}$-complete. We also show that, a closely related problem, the MINIMUM COST LINK UPGRADE problem is $\mathcal{NP}$-complete even if the underlying topology of the network is a linear array. However, for certain classes of networks, the LINK UPGRADE problem can be solved in polynomial time. For general networks, we provide effective heuristics for the above problems.

Key words: computer network, performance, link upgrade problem, NP-complete problem, effective heuristics.

1. Introduction. In recent years, advances in very large scale integration (VLSI) technology have resulted in smaller and more powerful microprocessors that are relatively inexpensive. As a result there is a move from large centralized computers towards many smaller decentralized ones. This proliferation of small autonomous machines has increased the demand for data communications between computers, terminals, and between terminal and computer. Computer networks provide the capability of interconnecting these small machines within a geographical area.

*This research was supported in part by the NUS Research Grant RP940643

Current developments in integrated services digital network (ISDN) requires computer networks to transfer large amounts of data with minimum delay in order to effectively and efficiently support various voice, digital data, text, and image applications. The performance of a network becomes an important issue. If an existing network does not meet the required performance, we can upgrade some or all of its links in order to improve the network's performance. Optical fibers are usually used to upgrade these links since they offer very large bandwidths.

The extent of improvement on network performance when some of its links are upgraded depends on both the type of network, its topology, the links chosen to be upgraded, and the technology used in the upgraded links. In telephone networks, twisted pair is widely used between subscribers and local stations. One Mbits/s is a typical data rate for paths on the order of a kilometer or less. For local area networks, cable television, and high-speed point-to-point links, coaxial cables are widely used and these have typical data rates from ten to several hundreds Mbits/s. Optical fibers have large bandwidth $\times$ distance products supporting data attaining transmission at rates of up to 100 Gbits/s over 100 kilometers (Li, 1983).

In this paper, we consider the link upgrade problem. Given a network and a specified performance, the link upgrade problem is to determine which links of the network are to be upgraded so that the specified performance of the network is attained. In subsequent sections, we will formulate the problem, study its computational complexity, propose exact algorithms for special cases of the problem, provide an effective heuristic for the general problem, and discuss other useful applications that are related to our link upgrade problems.

2. Problem formulation. We model the topology of a computer network as a *weighted graph*. A graph $G = (V, E; w)$ is a set V of n vertices which represent the communication centers or concentrators in the network, together with a set E of m undirected edges representing bidirectional communication links. If an edge e is incident with vertices u and v, we will sometimes write e as the unordered pair (u, v). This graph incorporates information about the network's topology, but does not include information about the characteristics of the links. A weighted graph has, in addition, a weighting function $w : E \rightarrow \mathbf{Z}_{\geqslant 0}$. The value $w(e)$ is called the *weight* of edge e.

We are concerned with the ability of a network to carry out a desired network

operation meeting certain performance requirement. An important first step is therefore to identify necessary network operations. Perhaps the most common operation is communication from a source vertex to a target vertex. Suppose a network $\mathcal{N}$ has packet size P and a link e has data rate $R(e)$. Then it takes time $P/R(e)$ to transmit a packet across e. Let $G = (V, E; w)$ be a weighted graph modeling $\mathcal{N}$ with $w(e) = P/R(e)$ for each $e \in E$. A source vertex s can transmit a packet to a target vertex t along any s,t-path in G, and the shortest possible time in which this can be done is given by the length of the shortest s,t-path. The length of the shortest s,t-path in G is commonly called the *distance* between s and t in G, denoted $\text{dist}_G(s,t)$. The performance measure we use for a network $G = (V, E)$ is its *diameter*:

$$D(G) = \max_{s,t \in V} \{\text{dist}_G(s,t)\}.$$

Intuitively, $D(G)$ is a guarantee on the speed of the network G; any transfer of a packet between any two vertices in G can be done in no more than $D(G)$ units of time.

Suppose now we are given a network $G = (V, E; w)$ and optical fibers whose data rate is R. Let $c = P/R$. We assume that the data rate R is so large that P/R is negligible when compared to other links in the network G. Therefore, upgrading a subset S of the links in E to the available optical fibers simply means transforming G into a new network $G' = (V, E; w')$, where

$$w'(e) = \begin{cases} w(e), & \text{if } e \in E \backslash S; \\ 0, & \text{if } e \in S. \end{cases}$$

We may assume without loss of generality that all weights are integers, by normalizing, if necessary. We can now formally state the LINK UPGRADE problem.

LINK UPGRADE

INSTANCE: A weighted graph $G = (V, E; w)$ and a nonnegative integer D.
QUESTION: Find a smallest (in cardinality) subset $S \subseteq E$ such that the weighted graph $G' = (V, E; w')$, where

$$w'(e) = \begin{cases} w(e), & \text{if } e \in E \backslash S; \\ 0, & \text{if } e \in S, \end{cases}$$

has diameter $D(G') \leqslant D$.

If the cost of upgrading each link differs and is given by $C(e)$, and the objective is to find a subset of edges with the minimum cost to be upgraded such that the $D(G') \leqslant D$, we name this problem, MINIMUM COST LINK UPGRADE problem.

Given an arbitrary instance of the LINK UPGRADE problem, there is a very simple solution strategy. One simply enumerates all states, that is, all possible subsets of E, and determine the smallest such subset so that changing all the weights of the edges in this subset to c would result in a weighted graph with diameter at most D. The details of this algorithm is given in Fig. 1. Let n be the number of vertices in the graph and m the number of edges. The diameter of a weighted graph can be found in $O(nm \log_{(2+m/n)} n)$ time by using n iterations of Dijkstra's single source shortest path algorithm (Dijkstra, 1959) or in $O(n^3)$ time using Floyd's all pairs shortest path algorithm (Floyd, 1962). Hence, the complete state enumeration algorithm in Fig. 1 runs in $O(2^m nm \log_{(2+m/n)} n)$ time or in $O(2^m n^3)$ time depending on whether Dijkstra's or Floyd's algorithm is used.

```
for k = 0 to |E|
    for each S ⊆ E with |S| = k {
        construct G' = (V, E; w');
        if (D(G') ⩽ D) {
            output S;
            stop;
        }
    }
```

Fig. 1. Complete state enumeration algorithm.

The algorithm described above requires exponential time in the worst case. It is reasonable to ask whether one can hope to improve on the exponential running time. Perhaps there exists a polynomial-time algorithm. In the next section, we provide evidence that such polynomial-time algorithms are unlikely to exist.

3. Computational complexity. A commonly accepted way of providing convincing evidence to support assertions that a problem is intractable is to

prove $\mathcal{NP}$-completeness or $\mathcal{NP}$-hardness of the problem (Garey and Johnson, 1979). A polynomial-time algorithm for any $\mathcal{NP}$-hard problem implies the existence of polynomial-time algorithms for a large number of other problems which are widely recognized as being difficult and that have been confounding the experts for years. In this section, we prove that the LINK UPGRADE problem is $\mathcal{NP}$-hard.

A decision version of the LINK UPGRADE problem is described below:

LINK UPGRADE (decision version)

INSTANCE: A weighted graph $G = (V, E; w)$, a nonnegative integer D, and a positive integer $K \leqslant |E|$.

QUESTION: Is there a subset $S \subseteq E$, $|S| \leqslant K$, such that the weighted graph $G' = (V, E; w')$, where

$$w'(e) = \begin{cases} w(e), & \text{if } e \in E \backslash S; \\ 0, & \text{if } e \in S, \end{cases}$$

has diameter $D(G') \leqslant D$?

A decision problem that is closely related to the above problem is the MINIMUM COST LINK UPGRADE. The MINIMUM COST LINK UPGRADE problem is described below:

MINIMUM COST LINK UPGRADE (decision version)

INSTANCE: A weighted graph $G = (V, E; w, C)$, a nonnegative integer D, and positive integer K.

QUESTION: Is there a subset $S \subseteq E$, $\sum_{e \in S} C(e) \leqslant K$, such that the weighted graph $G' = (V, E; w', C)$, where

$$w'(e) = \begin{cases} w(e), & \text{if } e \in E \backslash S; \\ 0, & \text{if } e \in S. \end{cases}$$

has diameter $D(G') \leqslant D$?

We shall use the following known $\mathcal{NP}$-complete problems (Karp, 1972) to establish the complexity of the decision versions of the LINK UPGRADE and MINIMUM COST LINK UPGRADE problems.

EXACT COVER BY 3-SETS (X3C)

INSTANCE: Set X with $|X| = 3q$ and a collection C of 3-element subsets of X.

QUESTION: Does C contain an exact cover for X, i.e., a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

PARTITION

INSTANCE: Finite set A and a size $s(a) \in \mathbf{Z}^+$ for each $a \in A$

QUESTION: Is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A - A''} s(a)$?

Theorem 1. *The decision version of the LINK UPGRADE problem is $\mathcal{NP}$-complete.*

Proof. It is easy to see that the decision version of the LINK UPGRADE problem belongs to the class $\mathcal{NP}$ since a nondeterministic algorithm needs only to guess a subset $S \subseteq E$ and to check in polynomial time whether $|S| \leqslant K$ and $G' = (V, E; w')$ has diameter $D(G') \leqslant D$.

To prove completeness, we transform X3C to the decision version of the LINK UPGRADE problem. Let $C = \{C_1, C_2, \cdots, C_n\}$ be the collection of 3-element subsets of a $3q$-element set X in an arbitrary instance of X3C. We must construct a weighted graph $G = (V, E; w)$ with nonnegative integer D and positive integer K, such that there exists a subset $S \subseteq E$ with $|S| \leqslant K$ and $D(G' = (V, E; w')) \leqslant D$ if and only if C contains an exact cover for X.

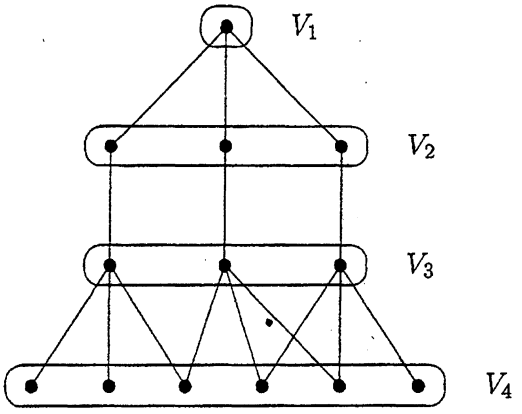
The set V of vertices will be partitioned into 4 separate classes, V_1, V_2, V_3, and V_4. V_1 consists of a single vertex, r, called the *root*. For each 3-element subset $C_i \in C$, we construct a vertex $u_i \in V_2$ and a vertex $v_i \in V_3$, $1 \leqslant i \leqslant n$. Each vertex $x_j \in V_4$ corresponds to an element $x_j \in X$, $1 \leqslant i \leqslant 3q$. There is an edge from every vertex in V_2 to the root. Every vertex $u_i \in V_2$ is adjacent to $v_i \in V_3$, and an edge exists between a vertex $v_i \in V_3$ and $x_j \in V_4$ if and only if $x_j \in C_i$. Edges exist between any two distinct vertices in V_i, $2 \leqslant i \leqslant 4$. All edges have a weight of 1. Let $D = 2$ and $K = q$. This completes our construction. We note that any two vertices in G is at most distance 2 apart unless one of the vertices is in V_4 and the other is the root, in which case the distance is 3. See Fig. 2 for an example of this construction.

Instance of X3C:

$$X = \{1,2,3,4,5,6\},\ C = \{\{1,2,3\},\ \{3,4,5\},\ \{4,5,6\}\}.$$

Instance of LINK UPGRADE:

$G = (V, E; w)$:



Each partition V_i, $2 \leqslant i \leqslant 4$, is a complete graph and $w(e) = 1$ for all $e \in E$, $K = 2$, $D = 2$.

Fig. 2. An example of the construction in Theorem 1.

Suppose first that $C' \subseteq C$ is an exact cover for X. We choose $S = \{(u_i, v_i) \in E : C_i \in C'\}$. We need only check the distance between the root and vertices in V_4. Consider any vertex $x_i \in V_4$. There is a 3-element subset $C_i \in C'$ containing x_j. Therefore, r, u_i, v_i, x_j is a path, and moreover, this path has length 2 in G' since $(u_i, v_i) \in S$ thus implying $w'(u_i, v_i) = 0$.

Suppose next that $S \subseteq E$ such that $|S| \leqslant q$ and $G' = (V, E; w')$ has diameter at most 2. It is not difficult to see that S cannot contain edges within any of the partitions V_i, $2 \leqslant i \leqslant 4$ and edges of the form (v_i, x_j). It is also easy to verify that S cannot contain both (r, u_i) and (u_i, v_i) for any $1 \leqslant i \leqslant n$. Consequently, changing the weight of only one edge in S affects the length of exactly 3 shortest r, x_j-paths for some $j \in J$, where $\{x_j : j \in J\} \in C$. We choose $C' = \{C_i \in C : (r, u_i) \in S \text{ or } (u_i, v_i) \in S\}$. Let x_j be any element of X. Then there is a unique path r, u_i, v_i, x_j in G' of length 2. Hence, there is

a unique 3-element subset in C' containing x_j.

It is easy to see that our construction can be carried out in polynomial time.

Theorem 2. *The MINIMUM COST LINK UPGRADE problem is $\mathcal{NP}$-complete even for linear array.*

Proof. It is also easy to see that the problem belongs to the class $\mathcal{NP}$. We will transform the partition problem to the decision version of the MINIMUM COST LINK UPGRADE problem. Let $A = \{1, 2, \cdots, n\}$ be the set and $S(a)$ is the size of any element a in A. Let the weighted graph be $G = (V, E; w, C)$, where:

$V = \{v_0, v_1, \ldots, v_n\}$, $n = |A|$;
$E = \{e_1, e_2, \ldots, e_n\}$, e_i is the edge between vertices v_{i-1} and v_i;
$w(e_i) = S(i)$, $\forall e_i \in E, i \in A$;
$C(e_i) = S(i)$, $\forall e_i \in E, i \in A$;
$k = D = \sum_{i=1}^{n} S(i)/2$.
Suppose $A' \subseteq A$ partitions the set A into 2 subsets such that

$$\sum_{a \in A'} S(a) = \sum_{a \in A - A'} S(a).$$

We choose the set $S \subseteq E$ to be:

$$S = \{e_i | i \in A'\}.$$

The diameter of the graph G is given by the distance between v_0 and v_n, which is $D = \sum_{i \in A - A'} w(e_i)$ and the cost of the upgrade is $K = \sum_{i \in A'} w(e_i)$. It is obvious that if a partition exists, then upgrading links in set S will result in the diameter equals to D and cost of upgrading equals to K. Let S be the subset such that the diameter of the graph is $D(G') \leqslant D$ and $\sum_{e \in S} C(e) \leqslant K$. The solution set A for the partition problem is $A' = \{i | e_i \in S\}$. It is quite obvious that $\sum_{a \in A'} S(a) = \sum_{a \in A - A'} S(a)$. Our transformation takes only polynomial time. This completes the proof.

In view of the apparent intractability of the LINK UPGRADE problem, one should not expect to design polynomial-time algorithms for solving arbitrary instances of the problem. The situation is even worse for the MINIMUM COST LINK UPGRADE problem where the problem is $\mathcal{NP}$-complete even for linear array. In this paper, we shall restrict ourselves to the LINK UPGRADE

problem. In the following sections, we will provide polynomial-time algorithms for solving certain restricted classes of networks, i.e., linear array, star, and trees. Following that, we will provide an effective heuristic for general networks.

4. Exact algorithms for restricted classes. The following are some common network topologies (Tsao, 1984).

- The *linear array*, in which nodes are connected linearly (see Fig. 3).

Fig. 3. A linear array.

- The *star*, in which every node or communication site is connected to a single central communication node, the *hub* of the star (see Fig. 4).

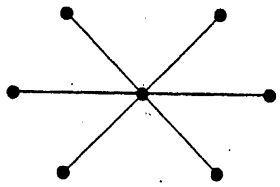


Fig. 4. A star.

- The *tree*, in which the path between any 2 nodes in the network is unique. The linear array and star topologies are restricted forms of trees (Fig. 5).

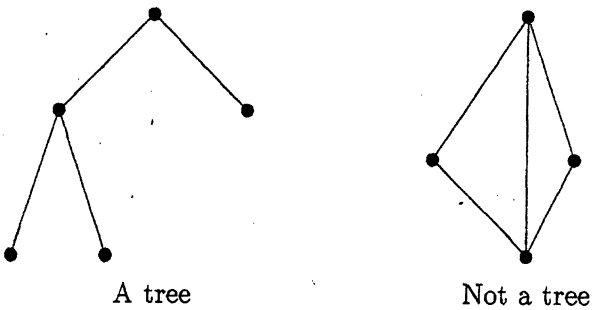


Fig. 5. Trees and non-trees.

4.1. The linear array and star. Let $G = (V, E; w)$ and D, be an instance of LINK UPGRADE problem, where G is a linear array of n vertices, and let S be a solution to this instance. A linear array has the property that its diameter is the sum of all its edge weights. So, obviously, the greedy method which works iteratively by selecting at each step an edge of highest weight among those not in S and putting it in S until $D(G' = (V, E; w')) \leqslant D$, solves the problem. Since the sorting step to reorder the edge weight takes $O(n \log n)$ time and all the edge selections and diameter computations and recomputations takes $O(n)$ time, the overall time complexity of the greedy algorithm is $O(n \log n)$.

The diameter of the star is the sum of its 2 highest edges. The same approach used for the linear array can be applied to the star. The complexity remains the same, which is $O(n \log n)$.

4.2. Trees. If the graph given $G(V, E; w)$ is a tree (i.e., the path between any 2 nodes in the graph is unique, see Fig.5), the dynamic programming approach may be used to obtain the smallest number of upgrades given the required diameter $\mathcal{D}$. The dynamic programming approach takes $O(n^2)$. For more details, please refer to (Lim and Chee, 1995).

5. Heuristics for general networks. For general networks, the link upgrade problem is $\mathcal{NP}$-complete. The situation is even worse for the minimum cost link upgrade problem, which is $\mathcal{NP}$-complete even for linear list/array. As a result, an effective heuristic H_1 is proposed for the link upgrade problem. A variant of the heuristic H_1 is H_1' is proposed for the minimum cost link upgrade problem. Both H_1 and H_1' can be found in Fig. 6.

Step 1 finds the all pairs shortest path. This can be done in $O(n^3)$. In Step 2, the algorithm compares the edge weight $w(e)$ and dist(e). If $\text{dist}(e) < w(e)$ then this edge is redundant and can be removed (note that remove is *not* the same as upgrade). This takes at most $O(|E|)$ time. Step 3 finds the edge e, where its upgrade results in the largest $\mathcal{F}(G(V, E)) - \mathcal{F}(G(V, E - e))$. This implies that all pairs shortest path must be recomputed or updated. To update the all pairs shortest path matrix, let us take a look at the shortest path between any 2 vertices x and y. With the upgrade of $e = (u, v)$, the minimum distance between them becomes,

$$\text{dist}(x, y) = \min\{\text{dist}(x, y), \text{dist}(x, u) + \text{dist}(v, y), \text{dist}(x, v) + \text{dist}(u, y)\}.$$

Heuristic H_1

//

// For H_1' we use $\frac{\mathcal{F}(G(V,E))-\mathcal{F}(G(V,E-e))}{C(e)}$ in Step 3

//

Step 1: Find the shortest path between all pairs of vertices.

Step 2: Remove all redundant edges.

Step 3: Select the edge, e, in the graph $G(V, E)$ with the largest $\mathcal{F}(G(V, E))-\mathcal{F}(G(V, E-e))$ such that:

$$\mathcal{F}(G(V,E)) = \sum_{\forall e \in E \ \& \ \mathrm{dist}(e) > D} \mathrm{dist}(e)$$

Step 4: If $\mathcal{F}(G(V, E-e)) = 0$ then go to Step 9.

Step 5: Upgrade the edge $e = (u, v)$ by contraction of e.

Step 6: Update the graph G due the Step 5.

Step 7: Update the shortest path matrix of the vertices.

Step 8: Go to Step 2.

Step 9: End.

Fig. 6. Heuristics H_1 and H_1'.

This is done for every edge and the there can be at most $O(|E|)$ edges. The time required for Step 3 is at most $O(|E||V|^2)$. $\mathcal{F}(G(V, E-e)) = 0$ implies that the diameter $\leqslant D$, hence the algorithm terminates (see Step 4). Steps 5–9 are quite self explanatory. The entire process can be repeated at most $O(|V|)$ times, i.e., every upgrade decrease the number of vertices in the graph by 1, the entire algorithm takes at most $O(|E||V|^3)$ time. In practice the algorithm's running time is proportional to the cube of its input size.

6. Experimental results. In order test the effectiveness of our heuristic, H_1, we implemented a simple greedy algorithm, H_0, that selects the first kth largest edges to be upgraded such that the diameter of the graph is no more than D. k is made as small as possible. This can be done by first sorting the edges according to their weights and using binary search to determine k. A simple implementation takes at most $O(n^3 \log n)$ time.

We tested H_0 and H_1 on graphs that are randomly generated. The density,

d, of a graph is defined as the probability that an edge between any 2 vertices exists. The results are summarized in Table 1 and Table 2. Column 1 is the number of vertices in the graph and column 2 the number of edges. Column 3, D, is the current diameter. The first row of column 4 indicates the desired diameter, D', of the graph as a percentage with respect to the original diameter. For instance, take the graph with 10 nodes in Table 1, the original diameter is 42, if the desired diameter is 75% of the original, we would like to use the smallest number of upgrades such that the diameter of the graph after the upgrade is 75% of 42 which is approximately 31. Column 5 has similar interpretation as column 4. The first sub-column of column 4 and 5 is the desired diameter D' and the second and third sub-columns are the number of upgrades needed by the greedy approach H_0 and our heuristic H_1 respectively.

Table 1. Test set (density=0.5, max cost=50)

$\|V\|$	$\|E\|$	D	$D' = 0.75D$			$D' = 0.5D$		
			D'	H_0	H_1	D'	H_0	H_1
10	23	42	31	2	2	21	5	3
20	156	38	29	3	1	19	11	4
50	606	20	15	12	4	10	36	11
75	1384	20	15	22	7	10	50	12
100	2487	15	11	57	8	8	101	15

Table 2. Test set 2 (density=0.2, max cost=50)

$\|V\|$	$\|E\|$	D	$D' = 0.75D$			$D' = 0.5D$		
			D'	H_0	H_1	D'	H_0	H_1
25	63	98	74	9	3	49	16	5
50	223	59	45	7	2	30	18	8
70	517	41	31	22	4	21	55	13
100	946	34	26	40	6	17	44	12

As you can see from the experimental results, only small number of edges need to be upgraded to improve the performance of the network by 100% in many instances.

7. Conclusion. We have defined and studied the LINK UPGRADE problem and the MIN COST LINK UPGRADE problem. We have proposed fast algorithms for restricted cases of the link upgrade problem and have shown that the problem is $\mathcal{NP}$-complete for general graphs. As for the minimum cost link upgrade problem, we have shown that it is $\mathcal{NP}$-complete even for linear lists. As a result, we have developed effective heuristics for both problems. Our experiment results show that our heuristic greatly outperformed the simple greedy method for the LINK UPGRADE problem. The link upgrade problem and its variations find applications not only in computer networks but also in road and transportation networks (Lim, 1995).

REFERENCES

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. *Numer. Math,* **1**, 269–271.

Floyd, R.W. (1962). Algorithm 97: Shortest paths. *Communications of the ACM,* **5**, 345.

Garey, M.R., and D.S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York.

Karp, R.M. (1972). Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher (Eds.), *Complexity of Computer Computations,* Plenum Press, New York. pp. 85–103.

Li, T. (1983). Advances in optical fiber communications: An historical perspective. *IEEE Journal on Selected Areas in Communications,* **SAC–1**(3), 356–372.

Lim, A (1995). Minimum Cost Upgrades to Improve the Public Road/Transportation Networks. Manuscript.

Lim, A., and Y.M. Chee (1995). On the upgrade problem for trees. *Journal of Combinatorics, Information and System Sciences.* Accepted for publication.

Tsao, C.D. (1984). A local area network architecture overview. *IEEE Communications Magazine,* **22**, 7–11.

Received December 1995

A. Lim received his B.Sc. (High Distinction), M.Sc. and Ph.D. degrees from the University of Minnesota (Twin Cities) in 1987, 1990, and 1992 respectively. At present he is a consultant (decision support systems) at the Information Technology Institute and an adjunct lecturer at the Department of Information Science and Computer Science. Dr. Lim's research interests and expertise include operations research/management in transportation and logistics, and combinatorial optimization.

Y.M. Chee received his B.Sc. and M.Sc. from University of Waterloo in 1988 and 1989 respectively. 1990–1992 he was a research scientist at the Defence Science Organization (Singapore). 1992–1994 he was an information architect at the National Computer Board (Singapore). At present, he is completing his Ph.D. at the University of Waterloo. Mr Chee's interests includes combinatorial optimization, combinatorics, algorithms and graph theory.

W. Hsu received her Ph.D. from Purdue University in 1994. At present, she is a lecturer in the Department of Information Science and Computer Science of the National University of Singapore. Dr. Hsu's research interests includes CAD, multimedia applications and algorithms.

TINKLO PRALAIDUMO DIDINIMAS JO GRANDŽIŲ PERGRADAVIMU

Andrew LIM, Yeow-Meng CHEE ir Wynne HSU

Kompiuterinių tinklų pralaidumas paprastai yra matuojamas maksimaliu/minimaliu laiku, reikalingu tam tikram duomenų kiekiui perduoti tarp dviejų tinklo viršūnių. Kai kurios grandys tinkle dėl technologinių reikalavimų gali būti pergraduojamos, pavyzdžiui keičiamos optiniais kabeliais, jų pralaidumo padidinimui. Šiame straipsnyje mes nagrinėjame tinklo GRANDŽIŲ PERGRADAVIMO uždavinį. Pimiausia parodome, kad šis uždavinys yra NP-pilnas. Mes taip pat parodome, GRANDŽIŲ PERGRADAVIMO MINIMALIOS KAINOS uždavinys, kuris gimininga pirmajam, yra NP-pilnas, net jeigu tinkle egzistuojanti topologija yra tiesinis masyvas. Tačiau tam tikrų tinklų klasėms GRANDŽIŲ PERGRADAVIMO uždavinys gali būti išspręstas per polinominį laiką. Čia, nagrinėdami bendruosius tinklus, mes pateikiame efektyvias euristikas išvardytiems uždaviniams spręsti.