

A CONSTRAINED VERSION of a CLUSTERING ALGORITHM for SWITCH PLACEMENT and INTERCONNECTION in LARGE NETWORKS

Vic Grout and Stuart Cunningham
Centre for Applied Internet Research (CAIR)
University of Wales, NEWI
Wrexham, North Wales, LL11 2AW, UK
{vic|stuart}@cair-uk.org

Abstract

This paper presents an extension to a clustering algorithm used for the heuristic calculation of switch location and interconnection in large networks. The original algorithm, a geometric form of weighted, representative reduction is constrained by a maximum load limit for replacement nodes. This has the effect of permitting more than one switch in a neighbourhood, possibly at the same site, more realistically reflecting the requirements of certain types of network. Also, with tighter weight constraints, a faster, more efficient heuristic is seen to perform slightly better than before.

1 INTRODUCTION

There are two essential, mutually dependent, optimisation components to the topological design problem for communication networks. Firstly, the placement of *switches* to concentrate traffic [1] and, secondly, the interconnection of *nodes* (non-switches) and switches [2]. Figure 1 shows the basic model used in this paper.

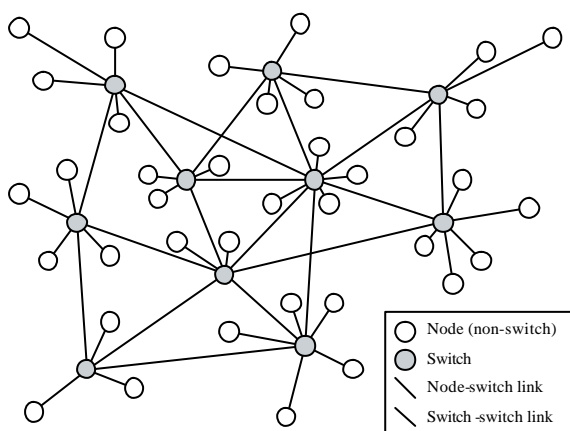


Figure 1. A (partial) mesh-star, two-level network

The switches form a *core* network, connected in a partial mesh. (non-switch) nodes connect to their nearest switch. Other structures such as star-star, (full) mesh-star, mesh-mesh, multiple (more than two) level, nodes connecting to other than their nearest switch, etc., are possible of course, and the techniques discussed in this paper remain appropriate. However, for brevity, the simple two-layer (partial) mesh-star model is used here.

2 NETWORK OPTIMISATION

The problem of finding an optimal solution is complex but often over-simplified [3][4][5][6][7][8]. In [9] a heuristic method is given for approximating the optimal network topology.

2.1 Network Cost

Firstly, a cost function is derived for any given solution. Using uppercase characters for switches and lowercase for nodes, $c_s(i)=0, c_s(X)>0, c_l(i,j)=0$ and, where the link in question is present, $c_l(X,Y)>c_l(i,X)>0$, where c_s and c_l are the costs of switches and links respectively. More precisely, if a link L carries traffic t over a distance d then $c_l(L) = f_l(t,d)$. If a switch S processes traffic T then $c_s(S) = f_s(T)$. f_l and f_s may be any appropriate, technology-dependent functions. Define t_{ij} to be the traffic originating at i and destined for j . Define d_{ij} to be the 'distance' between i and j . (Distances may or may not be Euclidean [3].) If a link is infeasible then $d_{ij} = \infty$. The cost of a link from a non-switch i to its *parent* switch X is then given by

$$c_l(i, X) = f_l\left(\sum_{j=1}^n t_{ij}, d_{iX}\right) \quad (1)$$

with a corresponding $c(X, i)$ in reverse. Define G_X to be

the set of nodes having X as their parent in a given configuration. The cost of the switch X is then

$$c_s(X) = f_s \left(\sum_{i \in \Gamma_X} \sum_{j=1}^n (t_{ij} + t_{ji}) \right). \quad (2)$$

For a full-mesh core network, the cost of the link (X, Y) is given by

$$c_l(X, Y) = f_l \left(\sum_{i \in \Gamma_X} \sum_{j \in \Gamma_Y} t_{ij}, d_{XY} \right) \quad (3)$$

with an equivalent $c_l(Y, X)$ in reverse. The total cost of the full-mesh network can then be calculated as

$$c^* = \sum_X \left[\begin{array}{l} \sum_{i \in \Gamma_X} (c_l(i, X) + c_l(X, i)) \\ + c_s(X) \\ + \sum_Y c_l(X, Y) \end{array} \right]. \quad (4)$$

If the link from switch X to switch Y is not present, then its traffic must be redirected via switches Z_1, Z_2, \dots, Z_r , resulting in a link saving of

$$f_l \left(\sum_{i \in \Gamma_X} \sum_{j \in \Gamma_Y} t_{ij}, d_{XY} \right). \quad (5)$$

but extra switch costs of

$$\begin{aligned} & f_s \left(\sum_{i \in \Gamma_{Z_1}} \sum_{j=1}^n (t_{ij} + t_{ji}) + \sum_{i \in \Gamma_X} \sum_{j \in \Gamma_Y} t_{ij} \right), \\ & f_s \left(\sum_{i \in \Gamma_{Z_2}} \sum_{j=1}^n (t_{ij} + t_{ji}) + \sum_{i \in \Gamma_X} \sum_{j \in \Gamma_Y} t_{ij} \right), \\ & \dots \\ & f_s \left(\sum_{i \in \Gamma_{Z_r}} \sum_{j=1}^n (t_{ij} + t_{ji}) + \sum_{i \in \Gamma_X} \sum_{j \in \Gamma_Y} t_{ij} \right) \end{aligned} \quad (6)$$

and link costs of

$$\begin{aligned} & f_l \left(\left(\sum_{i \in X} \sum_{i \in Z_1} t_{ij} + \sum_{i \in X} \sum_{i \in Y} t_{ij} \right), d_{XZ_1} \right), \\ & f_l \left(\left(\sum_{i \in Z_1} \sum_{i \in Z_2} t_{ij} + \sum_{i \in X} \sum_{i \in Y} t_{ij} \right), d_{Z_1Z_2} \right), \\ & \dots \\ & f_l \left(\left(\sum_{i \in Z_r} \sum_{i \in Y} t_{ij} + \sum_{i \in X} \sum_{i \in Y} t_{ij} \right), d_{Z_rY} \right) \end{aligned} \quad (7)$$

with the total network cost c^* recalculated accordingly. The saving is positive if spare capacity can be found on the links and switches in the new path.

2.2 Network Reduction

Secondly, [9] uses a clustering algorithm [10][11] to replace the original n nodes in the original problem with a smaller number of representative ones. Define the *weight* of each node to be its total traffic load:

$$w_i = \sum_{j=1}^n (t_{ij} + t_{ji}). \quad (8)$$

Define each node i by its coordinates, (x_i, y_i) . Then define a single *reduction step*, $RS(m)$, acting on m nodes, as:

```

RS(m) :
min = MaxVal
for each node pair  $i, j$  do
    if  $d_{ij} < min$  then
         $i^* = i, j^* = j, min = d_{ij}$ 
 $x_k = (w_{i^*}x_{i^*} + w_{j^*}x_{j^*}) / (w_{i^*} + w_{j^*})$ 
 $y_k = (w_{i^*}y_{i^*} + w_{j^*}y_{j^*}) / (w_{i^*} + w_{j^*})$ 
 $w_k = w_{i^*} + w_{j^*}$ 
for each node,  $? ( ? \neq i^*, j^* )$  do
    begin
         $d_{k?} = (w_{i^*}d_{i^*?} + w_{j^*}d_{j^*?}) / (w_{i^*} + w_{j^*})$ 
         $d_{?k} = (w_{i^*}d_{?i^*} + w_{j^*}d_{?j^*}) / (w_{i^*} + w_{j^*})$ 
    end

```

$RS(m)$ finds the closest two nodes, as defined by distances d_{ij} and replaces them by a single, representative node, biased by the weights w_i and w_j . (See Figure 2.)

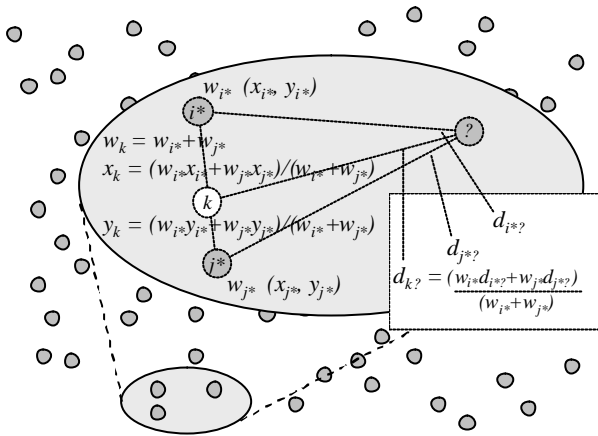


Figure 2. A single reduction step, $RS(m)$

The original m nodes are replaced by a representative $m-1$ in this single step. $RS(m)$ is the essential component in a compound algorithm that can perform conventional optimisation on a network problem of reduced size. If $RS(m)$ is repeated $n - q$ times, the original network problem of size n will be replaced by a representative one of size q . These q nodes can be used in three ways to approximate an optimum solution – described in the next section. The complexity of the reduction process, a sequence of matrix searches, is bounded above by $O(n^3)$.

In principle, these replacement nodes could have been generated by a grid-based top-level division of the network into q regions then averaging x - and y -coordinates in each. However, the q nodes generated by bottom-up reduction described here are truly representative of the underlying problem. They will be distributed according to node clustering and traffic density, not uniformly with arbitrary region boundaries dividing natural node/traffic groups. Another problem with this method is that fixed partitions produce a fixed number of representative nodes. The nodes produced by p partitions are independent of those produced by $p+1$, etc. In contrast, representative reduction generates each iteration from the previous one. A number of studies (eg. [12][13]) have shown reduction to be superior to partitioning in this respect.

2.3 Compound Optimisation

[9] then proceeds to discuss three methods by which the reduced node set may be used:

1. *RES*: Reduction to Exhaustive Search
2. *RDD*: Reduction to Double-Drop
3. *RSL*: Reduction to Switch location,

the outline of each algorithm being as follows:

RES:	RDD:	RSL:
$m = n$	$m = n$	$m = n$
repeat	repeat	repeat
$RS(m)$	$RS(m)$	$RS(m)$
until	until	$Rel(m)$
$m = n_{ES}$	$m = n_{DD}$	$COpt(m)$
$Rel(m)$	$Rel(m)$	until
$ES(m)$	$DD(m)$	$co(m) > co(m+1)$
		$m = m+1$
		$Opt(m)$

In addition to $RS(m)$, defined above, the routines $Rel(m)$, $COpt(m)$, $Opt(m)$, $ES(m)$ and $DD(m)$ and the functions and values $co(m)$, n_{ES} and n_{DD} are described in the original paper with full algorithms given where appropriate [9][14]. Essentially, $Rel(m)$ relocates each replacement (and therefore ‘greenfield’) node to its nearest true position (from the original nodes); $COpt(m)$ finds the optimum full-mesh core network on the current nodes ($co(m)$ is the cost of this solution) and $Opt(m)$ the optimum for any core network; $ES(m)$ is an exact exhaustive search algorithm for the current node set and $DD(m)$ performs an established ‘double-drop’ heuristic [14]. n_{ES} is the maximum number of nodes for which exhaustive search is feasible, for the processing power available, and n_{DD} the equivalent value for double-drop. As an example, for a standard configuration 2.8GHz Pentium processor with maximum run time of 24 hours, with algorithms coded in non-optimised C++, $n_{ES} = 12$ and $n_{DD} = 60$.

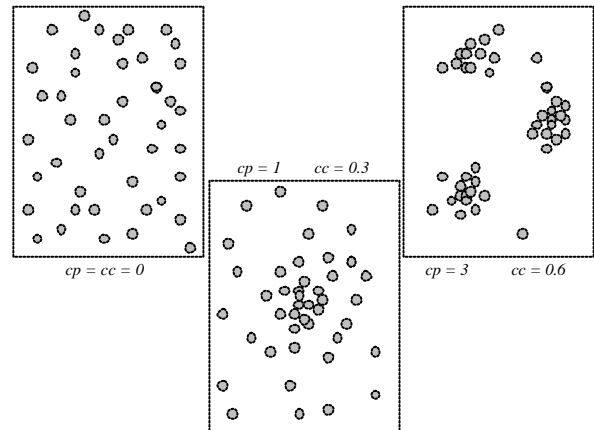


Figure 3. Cluster characteristics

2.4 Initial Results

RES, *RDD* and *RSL* were tested using 4,000 computer-generated and four real network scenarios [9]. The problem instances varied in the number of nodes, cost functions [15][16], traffic characteristics and node distribution. In particular, a number of cluster points (cp)

were defined, from 0 to 25 with a *cluster coefficient* (cc - in the range 0 to 1) as illustrated in Figure 3.

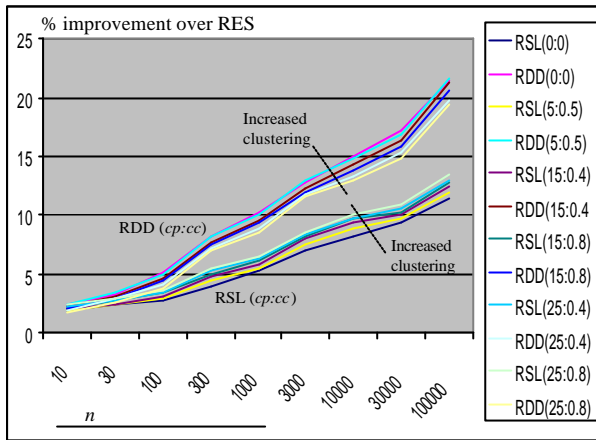


Figure 4. Comparing *RES*, *RDD* and *RSL*.

The relative performance of the algorithms *RES*, *RDD* and *RSL* are summarised in Figure 4. The two sets of plots show the improvement of *RDD* and *RSL* over *RES* for problems of different sizes and characteristics. For example, *RDD*(5;0.5) indicates *RDD* being applied to a problem instance of five cluster points with a (moderate) tendency to cluster of 0.5.

The key result from Figure 4 is that, as the size of problem grows, *RDD* increasingly outperforms *RSL*, which, in turn, increasingly outperforms *RES*. The conclusions are that the necessarily low value of n_{ES} makes for poor *RES* performance (reduction has been over-applied) while *RDD* produces good approximations to the optimum solution (reduction stops earlier). *RSL*, which runs much faster than the other two, is an acceptable compromise in more time-sensitive applications - if frequent re-optimisation is to be performed for example. In fact, *RSL* performs (comparatively) better with respect to *RDD* as the level of clustering increases.

3 LOAD-CONSTRAINED OPTIMISATION

There is a limitation to these three compound algorithms, however, stemming from the clustering process, *RS*. Essentially, applying *RS* iteratively as in *RES*, *RDD* and *RSL*, will replace nearest neighbours at each stage with the result that, as each compound algorithm progresses, the distances between nodes (in the current node set) increase. The implication of this is that, at the point where *ES* or *DD* come into operation or switches are sited directly at the nodes that remain,

switches are forced to be mutually distant from each other. No two switches can be closer than the closest distance between any two nodes, which in turn will be large as a result of the reduction process.

In real-world network design, this is a serious issue. There are a number of reasons why switches may need to be close, adjacent or even coincident in a practical network design. For example, in a large city, the traffic density through the 'theoretical' switch may exceed the capacity of any available switch model. It would not be uncommon to have (say) LONDON_1, LONDON_2 and LONDON_3 at different points in the same city, within close proximity, at the same site or even in the same rack! A further use of closely - or co-located switches would be as backup for network robustness and fault tolerance. These requirements, if needed, may be factored into the cost functions, f_i and f_s [15][16].

3.1 Constrained Reduction

The solution to overcome this problem is fairly simple and intuitive. Define a *load limit* for switches of W . Then do not permit the replacement of node pairs whose combined weights exceed this limit. If the weight of any initial node exceeds the limit then replace the single node by two or more replacement nodes before the reduction begins. Thus we have an initialisation step, I , as follows:

```

I(n):
for each node, i do
  if  $w_i > W$  then
    begin
       $m = \lceil w_i / W \rceil$ 
      <replace node i by m nodes,  $i_1, \dots, i_m$ >
      for each  $l$  ( $1=l=m$ ) do
        begin
           $x_{il} = x_i$ 
           $y_{il} = y_i$ 
           $w_{il} = w_i / m$ 
          for each node, ? ( $? \neq i$ )
            begin
               $d_{il?} = d_{i?}$ 
               $d_{?il} = d_{?i}$ 
            end
          end
        end
      end
    end
  end

```

($\lceil w_i / W \rceil$ is one plus the integer part of w_i / W) and a revised reduction process, *RS'*, of:

```

RS'(m):
min = MaxVal
for each node pair i, j do
  if ( $d_{ij} < min$ ) and ( $w_i + w_j = W$ ) then

```

```

        i* = i, j* = j, min = dij
xk = (wi*xi* + wj*xj*) / (wi* + wj*)
yk = (wi*yi* + wj*yj*) / (wi* + wj*)
wk = wi* + wj*
for each node, ? (? ' i*, j*) do
begin
    dk? = (wi*di? + wj*dj?) / (wi* + wj*)
    d?k = (wi*d?i* + wj*d?j*) / (wi* + wj*)
end

```

The compound algorithms, *RES*, *RDD* and *RSL* then become *RES'*, *RDD'* and *RSL'*:

RES' :	RDD' :	RSL' :
<i>I</i> (<i>n</i>)	<i>I</i> (<i>n</i>)	<i>I</i> (<i>n</i>)
<i>m</i> = <i>n</i>	<i>m</i> = <i>n</i>	<i>m</i> = <i>n</i>
repeat	repeat	repeat
<i>RS'</i> (<i>m</i>)	<i>RS'</i> (<i>m</i>)	<i>RS'</i> (<i>m</i>)
until	until	<i>Rel</i> (<i>m</i>)
<i>m</i> = <i>n_{ES}</i>	<i>m</i> = <i>n_{DD}</i>	<i>COpt</i> (<i>m</i>)
<i>Rel</i> (<i>m</i>)	<i>Rel</i> (<i>m</i>)	until
<i>ES</i> (<i>m</i>)	<i>DD</i> (<i>m</i>)	<i>co</i> (<i>m</i>) > <i>co</i> (<i>m</i> +1)
		<i>m</i> = <i>m</i> +1
		<i>Opt</i> (<i>m</i>)

3.2 Revised Results

RES', *RDD'* and *RSL'* were tested for the same problem instances as in Section 2.3 but with the added constraint of the switch load limit, *W*. The effect of a given value of *W* is relative to the level of traffic present in a given problem instance. Define *T** to be the total traffic load on the network:

$$T^* = \sum_{i=1}^n w_i = \sum_{i=1}^n \sum_{j=1}^n (t_{ij} + t_{ji}) \quad (9)$$

and calculate *W* as

$$W = \frac{W_0 T^*}{n} \quad (10)$$

For *W*₀=1, approximately 50% of initial nodes exceed the load limit at the outset. For *W*₀=1.5, the figure is approximately 25% but for *W*₀=2, less than 1%. However, as reduction progresses, potential replacement nodes may exceed the load limit so higher values of *W*₀ are still relevant. The original case with no load limit corresponds to *W* (= *W*₀) = ∞. Figure 5 compares *RES'*(5:0.5) (the modified *RES* with five cluster points and a cluster coefficient of 0.5), *RDD'*(5:0.5) and *RSL'*(5:0.5). Here, the bracketed figure is *W*₀. For example, *RDD'*(2) indicates *RDD'*(5:0.5) with *W*=2*T**/*n*.

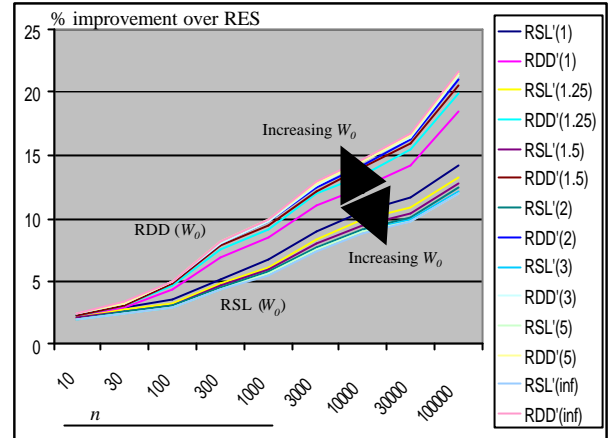


Figure 5. Comparing *RES'*, *RDD'* and *RSL'*.

The results indicate successful implementation as before. *RDD'* still outperforms *RSL'*, which in turn outperforms *RES'* and, once again, the difference is more apparent for larger problems. However, the value of the load limit, *W*(*W*₀) can be seen to have some effect. Lower (more restrictive) load limits reduce the advantage of *RDD'* over *RSL'* in the same manner as increased clustering affected *RDD* and *RSL*. For a combination of high levels of clustering and low load limits, *RSL'* becomes a good alternative to *RDD'* with much shorter run times.

4 CONCLUSIONS

This paper has concerned itself with the problem of finding or approximating the optimal placement and connection topology of switches in a network. The original paper [9] outlines the problems with conventional network optimisation techniques [4][5][6][7][8] and offers a practical solution based on a realistic cost function and a set of compound heuristics. The central component of the compound approach is a stepwise reduction process, which replaces the original problem by one of manageable size. There are then various options as to how to proceed to solve the reduced problem, dependent upon processing power and time available.

The original technique, however, is limited in a crucial respect. It is guaranteed to produce solutions in which the distance between switches is large. This may not be appropriate in situations where a number of switches are required to support heavy traffic density or for backup purposes or if a sufficiently large switch is not available. The original problem, and hence any appropriate solution method, must be constrained accordingly.

This paper introduces and defines the necessary constraint to achieve this real-world limitation. It uses a load limit parameter to restrict the size of any switch. A new initialisation step is introduced to break large nodes into smaller ones and the reduction algorithm is redefined to prohibit the formation of replacement nodes that violate the limit. The various compound processes are adjusted accordingly.

The revised processes perform well in testing and give similar results to those found in the initial paper. However, when the load limit is strict, the simplest and most efficient compound algorithm is seen to perform slightly better than before in comparison to its more sophisticated but complex alternative.

5 REFERENCES

- [1] J. G. Klincewicz, "Hub location in backbone/tributary network design: A review", *Location Science*, Vol. 6, 1998, pp. 307-335.
- [2] S. Soni, R. Gupta, and H. Pirkul, "Survivable Network Design: The state of the art", *Information Systems Frontiers*, Vol. 1, 1999, pp. 303-315.
- [3] V. Grout, "Optimisation Techniques for Telecommunication Networks", PhD Thesis, Plymouth Polytechnic, 1988.
- [4] B. Gavish, "Topological design of computer communication networks – the overall design problem", *European Journal of Operational Research*, Vol. 58, 1992, pp. 149-172.
- [5] R. B. Boorstyn and H. Frank, "Large-scale network topological optimization.", *IEEE Transactions on Communications*, Vol. COM-25, 1977, pp. 29-47.
- [6] S. Lin and J. H. Rath, "Designing networks for AT&T's customers", *AT&T Technology*, Vol. 2, 1987, pp. 18-25.
- [7] C. L. Monma and D. D. Sheng, "Backbone network design and performance analysis: A methodology for packet switching networks", *IEEE Journal on Selected Areas in Communications*", Vol. SAC-4, 1986, pp. 946-965.
- [8] E. Rosenberg, "Dual Ascent for uncapacitated telecommunications network design with access, backbone and switch costs", *Telecommunications Systems*, Vol. 16, 2001, pp. 423-435.
- [9] V. Grout, R. Picking, S. Cunningham and R. Hebblewhite, "Compound Algorithms for Realistic Large-Scale Network Optimisation", *Internet Research (to appear)* 2006.
- [10] L. Kleinrock and F. Kamoun, "Optimal clustering structures for hierarchical topological design of large computer networks", *Networks*, Vol. 10, 1980, pp. 221-248.
- [11] D. Saha and A. Mukherjee, "Design of hierarchical communication networks under node/link failure constraints", *Computer Communications*, Vol. 18, 1995, pp. 378-383.
- [12] V. Grout, P.W. Sanders & C.T. Stockel, "Reduction Techniques Providing Initial Groupings for Euclidean Travelling Salesman Patching Algorithms", *Applied Mathematical Modelling*, Vol. 13, February 1989, pp.110-114
- [13] V. Grout, "A Distribution-Aware Partitioning Algorithm with Variable-Order Perturbation to Approximate Optimal Tours", *Proceedings of the International Conference on Modeling and Optimization, MODOPT 2004*, Temuco, Chile, 19th-22nd January 2004.
- [14] V. Grout, R. Picking, S. Cunningham and R. Hebblewhite, "Realistic Large-Scale Network Optimisation", *Proceedings of the sixth International Network Conference (INC 2006)*, University of Plymouth, UK, 11-14 July 2006, pp.121-130.
- [15] R. Chan, *Wide Area Network Design: Concepts and tools for optimization*, Morgan Kaufmann, 1998.
- [16] V. Gabrel, A. Knipple. and M. Minoux, "Exact Solution of Multicommodity Network Optimization Problems with General Step Functions", *Operations Research Letters*, Vol. 25, pp.15-23, 1999.