Pergamon

Computers Math. Applic. Vol. 35, No. 11, pp. 17–26, 1998
© 1998 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0898-1221/98 $19.00 + 0.00

PII: S0898-1221(98)00080-7

Topological Optimization of Computer Network Expansion with Reliability Constraint

FANG-MING SHAO AND LIAN-CHANG ZHAO
Department of Basic Science
Dalian Maritime University
Dalian, Liaoning, 116024, P.R. China

(Received and accepted December 1997)

Abstract—This paper considers an optimization of a computer network expansion with a reliability constraint. Computer network expansion is achieved by adding a new node and communication links to a computer network such that reliability of the computer network is not less than a given level. In order to speed up the procedure of solution, an upper bound on system reliability in terms of node degrees is applied. The main problem is split into several small problems, the small problem is decomposed subproblems and the subproblems are solved effectively by a new method, forest search algorithm. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords—Sys-reliability, Algorithm, Network.

NOMENCLATURE

N_0	node set with $\|N_0\|$ nodes
L_0	link set with $\|L_0\|$ links
(n_i, j)	link between new node n_i and node j of G_0
p, q	link reliability/unreliability, respectively, and $p + q = 1$
$G_0(N_0, L_0, p)$	original graph (N_0, L_0), including p
$G(N, L, p)$	network obtained by adding a node and links to G_0
$R(G)$	reliability of G (all node-pairs in G can communicate)
$x_{n_i,j}$	$x_{n_i,j} = 1$ if $(n_i, j) \in G$, else $x_{n_i,j} = 0$
x	$(x_{n_1,1}, x_{n_1,2}, \dots, x_{n_1,\|N_0\|}, x_{n_2,1}, x_{n_2,2}, \dots, x_{n_m,\|N_0\|})$
x_{n_i}	if n_i is selected to add to G_0, $x_{n_i} = 1$, otherwise $x_{n_i} = 0$
n_i	the i^{th} node given to add to G_0
$\mathcal{N}$	set of given nodes preparing to add to G_0, $\mathcal{N} = \{n_1, n_2, \dots, n_m\}$
$\|\mathcal{N}\|$	number of nodes in $\mathcal{N}$
c_{n_i}	cost of node n_i
$\mathcal{N}^c$	cost set of $n_i, n_i \in \mathcal{N}$; $\mathcal{N}^c = \{c_{n_1}, c_{n_2}, \dots, c_{n_m}\}$
L_{n_i}	set of links incident with n_i
$\|L_{n_i}\|$	number of links in L_{n_i}
$c_{n_i,j}, c(e)$	cost of (n_i, j), link e
$L^c_{n_i}$	set of cost $c_{n_i,j}$, $(n_i, j) \in L_{n_i}$
m	number of nodes in $\mathcal{N}$, $m = \|\mathcal{N}\|$

Typeset by $\mathcal{AMS}$-TEX

MP	main mathematical problem
$SP_{n_i}(l_i)$	subproblem of MP, $l_i \in \{1, 2, \dots, \lvert L_{n_i}\rvert\}$ for n_i
$SP(l)$	subproblem of MP is consisted of $SP_{n_i}(l)$, $n_i \in \mathcal{N}$
$z_{n_i}(l_i)$	optimal solution of $SP_{n_i}(l_i)$
z	optimal solution of MP
$\underline{z}_{n_i}(l_i)$	minimum $SP_{n_i}(l_i)$ for l_i
$\underline{z}(l_i)$	minimum $SP(l)$, $\underline{z}(l_i) = \min\{\underline{z}_{n_i}(l_i) \mid n_i \in \mathcal{N}\}$
$RL_{n_i}(l_i)$	subproblem l_i of MP
$r_{n_i}(l_i)$	solution to $RL_{n_i}(l_i)$, viz., maximum for l_i links
$\bar{r}_{n_i}(l_i)$	upper bound of $r_{n_i}(l_i)$

1. INTRODUCTION

One of the practical problem in the computer network design is a cost optimization under the system reliability. Network expansion deals with an ever growing need for more computing across a network. In order to meet this demand, the size of the network is incrementally expanded according to the user requirements. It is critical that new nodes and links are added in an economical manner to meet the reliability demand of the network.

Many papers [1–5] consider the optimization for maximizing network reliability subject to cost constraints and for maximizing cost subject to reliability constraint, especially, the genetic algorithm [2,5] which gives a new technology to the reliability optimization of network expansion with cost constraints. This paper presents a practical method, the forest search algorithm, for exactly solving the cost optimization problem of network expansion under reliability constraint. The approach of Branch and Bound is extended to search the optimal solution among several combinatorial trees, a combinatorial forest. In the procedure of solution, the algorithm does not test each feasible solution in all of the trees one after another, however, it searches some feasible solutions in different trees simultaneously. Hence, the optimal solution is realized efficiently.

Section 2 states the optimal problem of network expansion and gives a mathematical expression. Section 3 illustrates the solution method of the main problem and its subproblems. Section 4 discusses the forest search algorithm. Section 5 provides two examples with some experimental results.

2. THE MATHEMATICAL DESCRIPTION OF THE PROBLEM

ASSUMPTION 1. *The network G and G_0 with bidirectional links do not have any redundant links.*

ASSUMPTION 2. *The set $\mathcal{N}$ of nodes preparing to add to the network, the set of possible selected links L_{n_i}, and the set of link cost $L^c_{n_i}$, corresponding to n_i are known. If $(n_i, j) \notin L_{n_i}$, then $c_{n_i,j} = 0$ $(i = 1, 2, \dots, m; j = 1, 2, \dots, \lvert N_0\rvert)$.*

ASSUMPTION 3. *The network G_0 and G have perfectly reliable status nodes, s-independent 2-state (good or failed) links. The reliability of every link is p.*

The problem is to add a node and some links incident with the node from a given node set $\mathcal{N}$ and link set L_{n_i} to a network such that the sum of a node cost and links cost is minimized and reliability of the resulting network is not less than given level P_0.

The main problem can be stated mathematically:

$$\text{Problem } MP: \quad z = \min \left\{ \sum_{i=1}^{m} \sum_{j=1}^{\lvert N_0\rvert} x_{n_i,j} c_{n_i,j} + \sum_{i=1}^{m} x_{n_i} c_{n_i} \right\},$$

$$\text{subject to:} \quad R(G) \ge P_0,$$

$$\sum_{i=1}^{m} x_{n_i} = 1.$$

3. THE DECOMPOSITION OF MAIN PROBLEM AND SOLUTION METHOD

First, the main problem can be split into small problems, SP_{n_i}, $n_i \in \mathcal{N}$.

$$\text{Problem } SP_{n_i}\text{:} \quad z_{n_i} = \min\left\{\sum_{j=1}^{|N_0|} x_{n_i,j}c_{n_i,j} + c_{n_i}\right\},$$
$$\text{subject to:} \quad R(G) \geq P_0.$$

Obviously, MP consists of $|\mathcal{N}|$ small problems, SP_{n_i}, $n_i \in \mathcal{N}$.

Second, the small problem SP_{n_i} can be decomposed into subproblems, $SP_{n_i}(l_i)$ $1 \leq l_i \leq |L_{n_i}|$, by number of links incident with n_i.

$$\text{Problem } SP_{n_i}(l_i)\text{:} \quad z_{n_i}(l_i) = \min\left\{\sum_{j=1}^{|N_0|} x_{n_i,j}c_{n_i,j} + c_{n_i}\right\},$$
$$\text{subject to:} \quad R(G) \geq P_0, \quad \sum_{j=1}^{|N_0|} x_{n_i,j} = l_i.$$

This means that subproblem SP_{n_i} consists of $SP_{n_i}(1), SP_{n_i}(2), \dots, SP_{n_i}(|L_{n_i}|)$.

Third, the subproblems of the small problem, $SP_{n_1}(l_1), SP_{n_2}(l_2), \dots, SP_{n_i}(l_m)$, can be denoted together as

$$\text{problem } SP(l): \quad z(l) = \min\{z_{n_i}(l_i) \mid n_i \in \mathcal{N}\},$$

where $l = (l_1, l_2, \dots, l_m)$ and $1 \leq l_i \leq |L_{n_i}|$.

In order to save the search time, the best way is to find l^* such that $R(G_0) \leq P_0$, and then consider $SP(l^*), SP(l^*+1), \dots, SP(l^*+k)$ where $l^* = (l_1^*, l_2^*, \dots, l_m^*)$; $l^*+1 = (l_1^*+1, l_2^*+1, \dots, l_m^*+1)$; $k = \max\{|L_{n_1}|, |L_{n_2}|, \dots, |L_{n_m}|\}$. In the procedure of solution, remove n_i satisfied $l_i > |L_{n_i}|$ from $\mathcal{N}$, this is, delete problem $SP_{n_i}(|L_{n_i}|+1)$. If $SP(l^*), SP(l^*+1), \dots, SP(l^*+k)$ are all solved, then the main problem, $z = \min\{z(l) \mid l = (l_1, l_2, \dots, l_m); 1 \leq l_i \leq |L_{N_i}|\}$. Based on the idea, Algorithm 1 is given as the following.

ALGORITHM 1. Main Algorithm for MP

```
1. Find a lower bound $l_i^*$ of the minimal number of links incident with $n_i$ such that $R(G)$
   $\geq P_0$. Set $l = l^* = (l_1^*, l_2^*, \dots, l_m^*)$ and current solution $z^* = \infty$.
2. While $MP$ is not solved, perform the loop:
   (a) solve $SP(l)$ by using the forest research algorithm and obtain $z(l)$. If $z(l) < z^*$, then
       set $z^* = z(l)$;
   (b) remove $n_k$ satisfied $l_k + 1 > |L_{n_k}|$ from $\mathcal{N}$, then
           $\underline{z}(l+1) = \min\{\underline{z}_{n_i}(l_i+1) \mid n_i \in \mathcal{N}\}$;
   (c) if $\underline{z}(l+1) < z^*$, then set $l = l + 1$ and go to Step (a).
3. $z^*$ is the optimal value for $MP$. STOP.
```

In Step 2, the symbols are denoted in the mathematical expression as follows:

- $\underline{z}_{n_i}(l_i+1)$ is the minimal bound of $z_{n_i}(l_i+1)$:

$$\underline{z}_{n_i}(l_i+1) = \min\left\{\sum_{c_{n_i,j} \in L^c} c_{n_i,j} + c_{n_i} \mid L^c \subset L_{n_i}^c, |L^c| = l_i + 1\right\},$$

- $\underline{z}(l+1) = \min\{\underline{z}_{n_i}(l_i+1) \mid n_i \in \mathcal{N}\}$.

NOTE. In $SP_{n_i}(l_i)$, l_i are considered from 1 to $|L_{n_i}|$. If $|L_{n_i}|$ subproblems are solved, then small problem SP_{n_i} is $z_{n_i} = \min\{z_{n_i}(l_i) \mid l_i = 1, 2, \dots, |L_{n_i}|\}$. The initial idea is to solve SP_{n_i}, and then the main problem is $z = \min\{z_{n_i} \mid i = 1, 2, \dots, m\}$. However, this method is not maximally efficient, because we have to test each feasible solution of the small problems and subproblems. Algorithm 1 develops the method in another manner, the method using $SP(l)$ to consider all of the subproblems of small problems, $SP_{n_i}(l_i)$, $n_i \in \mathcal{N}$, is effective to solve the main problem.

In order to determine a lower bound l_i^* of the minimal number of links incident with n_i such that $R(G) \geq P_0$, it is necessary to know the maximum reliability for a fixed l_i.

Consider

$$\begin{aligned} &\text{Problem } RL_{n_i}(l_i)\text{:} && r_{n_i}(l_i) = \max R(G), \\ &\text{subject to:} && \sum_{j=1}^{|N_0|} x_{n_i,j} = l_i. \end{aligned}$$

$RL_{n_i}(l_i)$ maximizes the reliability such that $\sum_{j=1}^{|N_0|} x_{n_i,j} = l_i$, where n_i is selected to add to G_0. If $r_{n_i}(l_i) < P_0$, then $SP_{n_i}(l_i)$ does not have any feasible solution. In addition to $r_{n_i}(1) \leq r_{n_i}(2) \leq \dots \leq r_{n_i}(|L_{n_i}|)$, we can determine the smallest k such that $r_{n_i}(k) \geq P_0$ and $r_{n_i}(k-1) < P_0$. In fact, we can use $\bar{r}_{n_i}(l_i)$ instead of $r_{n_i}(l_i)$. Because it is easy to compute $\bar{r}_{n_i}(l_i)$ and if $\bar{r}_{n_i}(k) \geq P_0$ and $\bar{r}_{n_i}(k-1) < P_0$, we also can determine k, but $k \leq l_i^*$. If we set $l_i^* = k$, we might do more loops in Algorithm 1.

Since resulting network $G(N, L, p)$ is obtained by adding node n_i and l_i links incident with n_i to $G_0(N_0, L_0, p)$, $G(N, L, p)$ has $|N|(= |N_0| + 1)$ nodes and $|L|(= |L_0| + l_i)$ links. Let $k = |N|$, from [1,6,7], we know the following.

If $l_i + |L_0| = k - 1$, then $\bar{r}_{n_i}(l_i) = p^{k-1}$.

If $l_i + |L_0| = k$, then $\bar{r}_{n_i}(l_i) = p^k + kp^{k-1}q$.

If $l_i + |L_0| = k + 1$, then $\bar{r}_{n_i}(l_i) = p^{k+1} + (k+1)p^k q + ((k+1)^2/3)p^{k-1}q$.

If $l_i + |L_0| > |N| + 1$, then

$$\bar{r}_{n_i}(l_i) = H(d) = 1 - \left\{ \sum_{i=1}^{|N|} q^{d_j} \prod_{k=1}^{m_j} \left(1 - q^{d_k - 1}\right) \prod_{k=m_j+1}^{j-1} \left(1 - q^{d_k}\right) \right\},$$

where $m_j = \min(d_j, j-1)$ and $d = (d_1, d_2, \dots, d_{|N|})$ $(d_1 \leq d_2 \leq \dots \leq d_{|N|})$ is the degree sequence satisfied $|d_i - d_j| \leq 1$ $(i \neq j)$ and $\sum_{i=1}^{|N|} d_i = |L|$.

EXAMPLE 3.1. Network $G_0(N_0, L_0, p)$ given as Figure 1a,

$$\begin{aligned} L_0 &= \{(1,2), (2,3), (2,4), (3,4), (4,5)\}, \\ N_0 &= \{1,2,3,4,5\}, \\ \mathcal{N} &= \{n_1, n_2\}, \\ L_{n_1} &= \{(n_1,1), (n_1,3), (n_1,4)\}, \\ L_{n_2} &= \{(n_2,1), (n_2,3), (n_2,4), (n_2,5)\}. \end{aligned}$$

When n_1 is selected, suppose that we get two links from L_{n_1}, links $(n_1, 1)$ and $(n_1, 3)$ should be selected (Figure 1b), and the resulting degree sequence is $d = (1,2,2,2,2,3)$. When n_2 is selected, links $(n_2, 1)$ and $(n_2, 5)$ should be selected (Figure 1c), and the degree sequence of the new network is $d_1 = (2,2,2,2,2,2)$. $H(d_1)$ is maximum over all possible d in G. From the example, we can clearly grasp that no matter how many links we want to get, the maximum bound of reliability can be gotten by the method above.

The detail of $\bar{r}_{n_1}(l_i)$ and $\bar{r}_{n_1}(l_i)$ as follows.

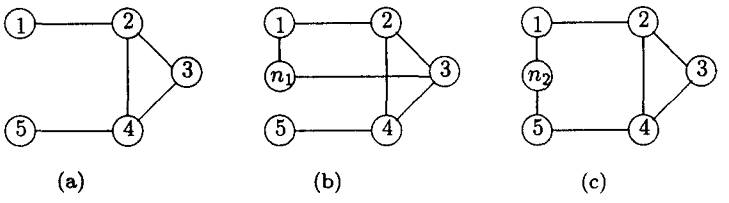

Figure 1. A given network G_0 and G.

Let $p = 0.91$ and $P_0 = 0.8$.

If $l_1 = l_2 = 1$, then $\overline{r}_{n_1}(1) = p^5 + 2p^5q = 0.7364$ and $\overline{r}_{n_2}(1) = p^5 + 2p^5q = 0.7364$, where $|L_0| + 1 = 6$.

If $l_1 = l_2 = 2$, then $\overline{r}_{n_1}(2) = p^5+4p^5q+6p^5q^2 = 0.8790$ and $\overline{r}_{n_2}(2) = p^5+6p^5q+8p^5q^2 = 0.9453$, where $|L_0| + 2 = 7$.

If $l_1 = l_2 = 3$, then $\overline{r}_{n_1}(3) = H(1,2,2,3,3,4) = 0.91$ and $\overline{r}_{n_2}(3) = H(2,2,2,2,3,3) = 0.9832731$, where $|L_0| + 3 = 8$.

Because $\overline{r}_{n_1}(2) > P_0$, $\overline{r}_{n_1}(1) < P_0$, $\overline{r}_{n_2}(2) > P_0$, and $\overline{r}_{n_2}(1) < P_0$, the bound of minimum number of links $l^* = (l_1, l_2)$=(2,2).

The next section presents the solution method of $SP(l)$.

4. FOREST RESEARCH ALGORITHM

In order to illustrate the method, we give an example.

EXAMPLE 4.1. An original network G_0 is given as Figure 2a,

$$\begin{aligned}
\mathcal{N}_0 &= \{1,2,3,4,5\},\\
L_0 &= \{(1,2),(1,3),(1,4),(2.3),(3,5),(4,5)\},\\
p &= 0.9,\\
\mathcal{N} &= \{n_1,n_2,n_3\},\\
L_{n_1} &= \{(n_1,1),(n_1,2),(n_1,3),(n_1,5)\},\\
L_{n_2} &= \{(n_2,2),(n_2,4),(n_2,5)\},\\
L_{n_3} &= \{(n_3,1),(n_3,3)\}.
\end{aligned}$$

The cost sets corresponding to each $n_i \in \mathcal{N}$ are

$$\begin{aligned}
\mathcal{N}^c &= \{c_{n_1} = 6, c_{n_2} = 4, c_{n_3} = 6\},\\
L^c_{n1} &= \{c_{n_1,1} = 1, c_{n_1,2} = 5, c_{n_1,3} = 3, c_{n_1,5} = 6\},\\
L^c_{n2} &= \{c_{n_2,1} = 2, c_{n_2,4} = 5, c_{n_2,5} = 8\},\\
L^c_{n3} &= \{c_{n_3,1} = 3, c_{n_3,3} = 6\}.
\end{aligned}$$

To each given node n_i, the solution space consists of $\binom{|L_{n_i}|}{l_i}$ combinations, in order to solve $P_{n_i}(l_i)$, we order the links according to $c_{n_i,j}$ in an nondecreasing sequence. The links are relabeled: the link with rank k becomes link $e^{n_i}_k$. All of combinations are represented by $|\mathcal{N}|$ trees. For instant, in Figures 2a–2c are trees corresponding to n_1, n_2, n_3, respectively. According to $L^c_{n_i}$, link sets in Example 4.1 should be

$$\begin{aligned}
L_{n_1} &= \{e^{n_1}_1 = (n_1,1), e^{n_1}_2 = (n_1,3), e^{n_1}_3 = (n_1,2), e^{n_1}_4 = (n_1,5)\},\\
L_{n_2} &= \{e^{n_2}_1 = (n_2,2), e^{n_2}_2 = (n_2,4), e^{n_2}_3 = (n_2,5)\},\\
L_{n_3} &= \{e^{n_3}_1 = (n_3,1), e^{n_3}_2 = (n_3,3)\}.
\end{aligned}$$

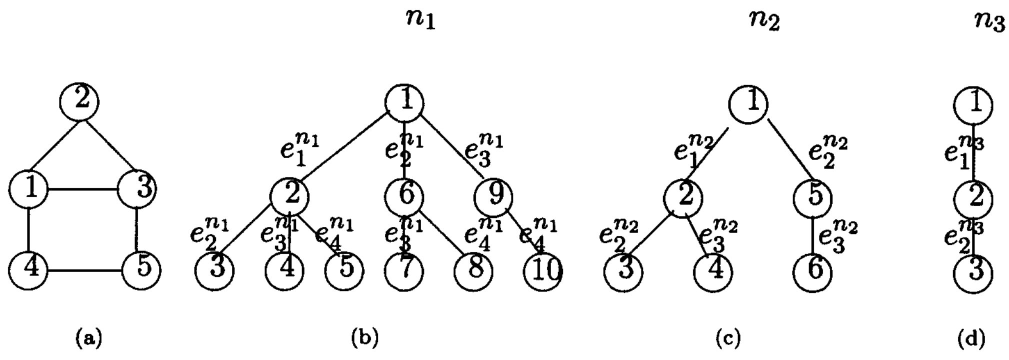

Figure 2. A given network and combinatorial trees corresponding to n_1, n_2, n_3.

In general, n_i-tree denotes the tree determined by node n_i. The links from the root (level-0) node to level-1 nodes in n_h-tree are specified by $\{e_j^{n_h}\}_{j=1}^{|L_{n_h}|-l+1}$. Links from the level-$k$ node pointed by the link with label $e_j^{n_h}$ to level-$(k+1)$ nodes are specified by $\{e_j^{n_h}\}_{j=i+1}^{|L_{n_h}|-l+k+1}$. The path from the root node to the leaf node defines a possible choice of l_i links. All of solutions of problem $P_{n_i}(l_i)$, $n_i \in \mathcal{N}$ are defined by all of the paths from the root node to the leaf nodes. Such as the number of solutions of $SP(2)$ are $\binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 10$. (In n_1-tree, there are $\binom{4}{2}$ leaf nodes. In n_2-tree, $\binom{3}{2}$ leaf nodes. In n_3-tree, $\binom{2}{2}$ leaf nodes.) Thus, the solution space of $SP(l)$ can be represented by all of the trees corresponding to $n_1, n_2, \dots, n_m$. In Example 4.1, the solution space of $SP(l)$ is defined by trees of Figure 2b–2d.

To find an optimal solution, we may consider all of the combinations of the trees. However, it is time-consuming, we can apply the branch and bound algorithm among different trees. This means that we do not need to search the tree one by one, we take all of the trees as a forest and jump the search among them, instead. All of the combinatorial trees do not need to list all simultaneously, we should enable the tree to live according to $g_{n_i}(2)$ $(n_i \in \mathcal{N})$, the lower bound of the objective function at node 2 of n_i-tree.

According to the idea above, we provide the method in detail as follows.

Step 1. Enable n_h-tree to live according to the lowest $g_{n_i}(2), n_i \in \mathcal{N}$, which is the lower bound of the objective function.

The lower bound $g_{n_k}(v)$ of the objective function in an arbitrary node v of n_i-tree is computed as follows.

Because links defined by the path from the root to node v is $\{e_{i_j}^{n_h}\}_{j=1}^{k}$. Let U be the set of $|L_{n_h}| - k$ links in L_{n_i} needed to be chosen from the remaining link set $\{e_j^{n_h}\}_{j=i_k+1}^{|L_{n_h}|-l+(k+1)}$, $g_{n_h}(v)$ is the smallest cost that appears in the complete choices generated from node U.

$$
\begin{aligned}
g_{n_i}(v) &= \min\left\{ \sum_{j=1}^{k} c\left(e_{i_j}^{n_i}\right) + \sum_{e \in U} c(e) + c_{n_i}, \text{ for all } U \right\} \\
&= \sum_{j=1}^{k} c\left(e_{i_j}^{n_i}\right) + \sum_{j=1}^{l-k} c\left(e_{i_k+j}^{n_i}\right) + c_{n_i}.
\end{aligned}
$$

Step 2. Feasibility testing at a leaf node of n_i-tree.

For the difficulty of computing $R(G)$, we use $\overline{r}_{n_i}(l)$ to estimate $R(G)$, if $\overline{r}_{n_i}(l) < P_0$, then the leaf node is infeasible. Otherwise, compute $R(G)$ by the algorithm in [8]. Verify if $R(G) \geq P_0$.

Step 3. Selection of a branching node and jumping the tree.

In the procedure of search, $g_{n_k}(2)$ is used as a controller to determine whether to enable the n_k-tree to live or not (Example 5.1 in the next section gives the specific illustration). The branch search method of n_i-tree is the same as [1].

ALGORITHM 2. The Forest Algorithm

```
1. initialize the live-node list $K$ to be empty;
   /*live-node list is a priority queues storing live nodes*/
2. enable the combinatorial tree such that $g_{n_i}(2)$ is the lowest of $g_{n_j}(1), n_i \in \mathcal{N}$;
3. first child $v_0$ in $n_i$-tree on the live-node list $K$;
4. set $g := 0$;
5. set $uc := \infty$;
6. WHILE live-node list is not empty DO
7. BEGIN
8. choose node $v$ with the minimum value of $g_{n_i}(v)$ corresponding to its tree from the live-
   node list; $g := \min\{g_{n_i}(2) \mid n_i \in K\}$;
9. set $S = \emptyset$;
10. IF $g \geq uc$ THEN
11. remove node $v$ from the live-node list $K$;
12. ELSE;
13. BEGIN
14. put the first child, next brother of $v$ and the first child of the tree which $g_{n_h} = \min\{g_{n_i}$ :
    $n_i \in \mathcal{N}\}$ into set $S$ /*the search jumping or the branch selecting*/
15. FOR each node $u$ in $S$ DO
16. BEGIN
17. IF node $u$ is a leaf node and $g_{n_i}(u) < g$ THEN
18. BEGIN /*feasibility testing*/
19. IF the network specified by the path from $v_0$ to $u$ is that $H(d) < P_0$, THEN node $u$ is
    infeasible
20. ELSE IF $R(G) \geq P_0$ THEN node $u$ is feasible, set $uc = \min(uc, g(u))$
21. END;
22. ELSE IF $g_{n_h}(u) < uc(n_h \in S)$ THEN
23. insert node $u$ into the live-node list $K$;
24. ELSE remove $u$ from $S$
25. END;
26. remove node $v$ from the live-node list $K$;
```

27. END;

28. END;

29. output the answer: node n_i and the optimal value $g(v) = uc$.

5. EXAMPLES AND RESULTS

EXAMPLE 5.1. The condition is the same as Example 4.1. The decision variable $x = \{x_{n_1,1}, x_{n_1,2}, \ldots, x_{n_1,5}, x_{n_2,1}, x_{n_2,2}, \ldots, x_{n_m,5}\}$ are selecting status of the link (n_i, j).

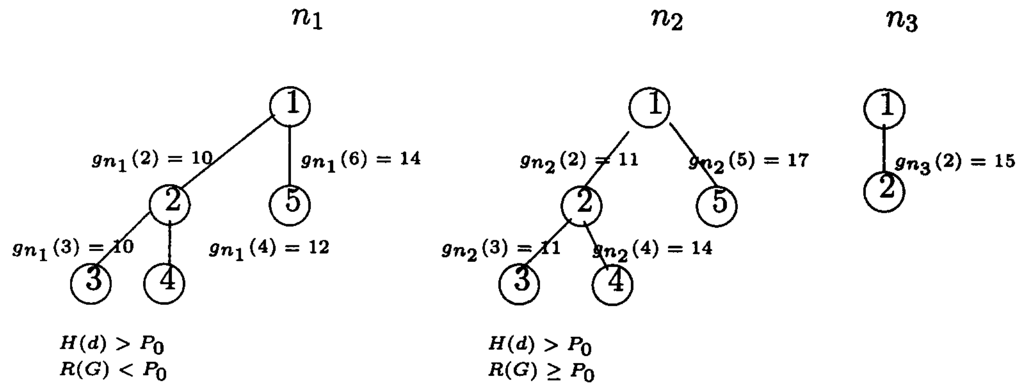

Figure 3. The procedure of the search by using the forest algorithm.

The details of the solution are as follows.

1. p =0.9, $P_0 = 0.95$, $\bar{r}_{n_1}(2) = H(2,2,3,3,3,3) = 0.9777867$, $l^*_{n_1} = 2$; $\bar{r}_{n_2}(2) = H(2,2,2,3,3,4) = 0.9713706$, $l^*_{n_2} = 2$; $\bar{r}_{n_3}(2) = H(2,2,2,3,3,4) = 0.964748$, $l^*_{n_3} = 2$.
2. Do the loops
   (a) the procedure of applying Algorithm 2 to solve $SP(l)$ is show in Figure 3. Symbol $n_h(v)$ denotes node v in n_h-tree.
   $K = \emptyset$

```
For $g_{n_1}(2) = \min\{g_{n_1}(2), g_{n_2}(2), g_{n_3}(2)\}$
Put $n_1(2)$ into $K$,$uc = \infty, g = 0$
    For $K = \{n_1(2)\} \neq \emptyset$
BEGIN
$S = \emptyset; g = \min\{g_{n_i}(v) \mid n_i(v) \in K\}$
For $g < uc$ and $g_{n_2}(2) = g$
Put $n_1(3), n_1(5), n_2(2)$ into $S$, $S = \{n_1(3), n_1(5), n_2(2)\}$
BEGIN
For $n_1(3)$ is a leaf node, test $n_1(3)$
For $H(d) > P_0, R(G) < P_0$
$n_1(3)$ is not a feasible solution
For $g_{n_1}(5) < uc, g_{n_2}(2) < uc, g_{n_1}(3) < uc$, add $n_1(5), n_1(3), n_2(2)$ into $K$
END
    remove $n_1(2)$ from $K$
END
$K = \{n_1(5), n_1(3), n_2(2)\}$
```

```
    For $K \neq \emptyset$
BEGIN
    $S = \emptyset; g = \min\{g_{n_i}(v) \mid n_i(v) \in K\}$
For $g < uc$ and $g_{n_1}(3) = g$
Put $n_1(4), n_2(2)$ into $S$, $S = \{n_1(4), n_2(2)\}$
BEGIN
For $g_{n_1}(4) < g$, add $n_1(4)$ into $K$
END
    remove $n_1(3)$ from $K$
END
$K = \{n_1(5), n_1(4), n_2(2)\}$
    For $K \neq \emptyset$
BEGIN
    $S = \emptyset; g = \min\{g_{n_i}(v) \mid n_i(v) \in K\}$
For $g < uc$ and $g_{n_2}(2) = g$
Put $n_2(3), n_2(5), n_3(2)$ into $S$, $S = \{n_2(3), n_2(5), n_3(2)\}$
BEGIN
For $n_2(3)$ is a leaf node, test $n_1(3)$
For $H(d) > P_0, R(G) > P_0$
$n_2(3)$ is a feasible solution, let $uc = g_{n_2}(3)$
For $g_{n_2}(5) > uc, g_{n_3}(2) > uc$, do not add $n_2(5), n_3(2)$ into $K$
END
    remove $n_2(2)$ from $K$
END
$K = \{n_1(5), n_1(4)\}$
    For $K \neq \emptyset$
BEGIN
    $S = \emptyset; g = \min\{g_{n_i}(v) \mid n_i(v) \in K\}$
For $g > uc$
    remove $n_1(5)$ and $n_1(4)$ from $K$
END
```

For $K = \emptyset$ and $z^*(2) = uc = g_{n_2}(3)$ is optimal value of $SP(l)$.
The optimal solutions $(x_{n_1,1}, x_{n_1,2}, \dots, x_{n_1,5}, x_{n_2,1}, x_{n_2,2}, \dots, x_{n_3,5}) = (0\,0\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0\,0\,0\,0)$
$z^*(2) = 11$

(b) Because $R(G) < P_0$, remove n_3, $\underline{z}(l+1) = 15 > z^* = 11$;
(c) $\underline{z}(l+1) > z^*$, then $z^* = 11$.

3. $z^* = 11$ is optimal value, the procedure of solution is finished.

EXAMPLE 5.2. $G_0(N_0, L_0, p)$ are given:

$$\begin{aligned} N_0 &= \{1,2,3,4,5,6\}, \qquad p = 0.94, \\ L_0 &= \{(1,2),(2,3),(3,4),(4,5),(5,6)\}, \\ \mathcal{N} &= \{n_1, n_2, n_3, n_4, n_5\}, \\ \mathcal{N}^c &= \{c_{n_1} = 5, c_{n_2} = 8, c_{n_3} = 6, c_{n_4} = 9, c_{n_5} = 12\}. \end{aligned}$$

Table 1. Table for L_{n_i} and $L^c_{n_i}$.

L_{n_1}	$(n_1,1)$	$(n_1,2)$	$(n_1,5)$	$(n_1,6)$	L_{n_2}	$(n_2,1)$	$(n_2,3)$	$(n_2,6)$
$e_i^{n_1}$	$e_1^{n_1}$	$e_2^{n_1}$	$e_3^{n_1}$	$e_4^{n_1}$	$e_i^{n_2}$	$e_1^{n_2}$	$e_2^{n_2}$	$e_3^{n_2}$
$c_{n_1,j}$	2	3	6	8	$c_{n_2,j}$	2	6	9

L_{n_3}	$(n_3,1)$	$(n_3,5)$	L_{n_4}	$(n_4,5)$	$(n_4,6)$	L_{n_5}	$(n_5,1)$	$(n_5,2)$
$e_i^{n_3}$	$e_1^{n_3}$	$e_2^{n_3}$	$e_i^{n_4}$	$e_1^{n_4}$	$e_2^{n_4}$	$e_i^{n_5}$	$e_1^{n_5}$	$e_2^{n_5}$
$c_{n_3,j}$	4	8	$c_{n_4,j}$	8	9	$c_{n_5,j}$	6	9

Find a network $G(N, L, p)$ by adding a node and its links to G_0 such that its cost is minimal and its reliability is larger than that of the original network.

The problem is formulated as:

$$\text{subject to:} \quad z = \min \sum_{i=1}^{5} \sum_{j=1}^{7} c_{n_i,j} x_{n_i,j}, \qquad R(G) \geq P_0.$$

Optimal solution is that node n_1 is selected, links are $(n_1,1), (n_1,2)$, and $z^* = 10$.

REFERENCES

1. R.-H. Jan, F.-J. Hwang and S.-T. Cheng, Topological optimization of a communication network subject to a reliability constraint, *IEEE Trans. Reliab.* **42**, 63–70, (1993).
2. A. Kumar, R.M. Pathak and K.P. Gupta, Genetic-algorithm-based reliability optimization for computer network expansion, *IEEE Trans. Reliab.* **44** (1), 63–70, (1995).
3. A.N. Venetsanopoulos and I. Singh, Topological optimization of communication networks subject to reliability constraints, *Problem of Control and Information Theory* **15**, 63–78, (1982).
4. K.K. Aggarwal, Y.C. Chopra and J.S. Bajwa, Topological layout of links for optimizing the *s-t* reliability in a computer communication system, *Microelectron. Reliab.* **22**, 347–351, (1982).
5. A. Kumar, R. Pathak and Y. Gupta, Computer network expansion using genetic algorithm approach, Technical Report, AK-EMCS-94-3, University of Louisville, (1994).
6. C.J. Colbourn, *The Combinations of Network Reliability*, Oxford University Press, (1987).
7. R.-H. Jan, Design of reliable networks, *Computers and Operations Research* **20**, 25–34, (1993).
8. M. Ball and R.M. Slyke, Backtracking algorithms for network reliability analysis, *Ann. Dis. Math.* **1**, 49–64, (1977).
9. Y.C. Cholar, B.S. Sohi, R.K. Tiwari and K.K. Aggarwal, Network topology for maximizing the terminal reliability in a computer communication network, *Microelectron. Reliab.* **24**, 911–913, (1984).
10. R.-S. Chen, D.-J. Chen and Y.S. Yeh, Reliability optimization of distributed computing systems subject to capacity constraints, *Computers Math. Applic.* **29** (4), 93–99, (1995).