A multipopulation-parallel-genetic-simulated-annealing-based QoS routing and wavelength assignment integration algorithm for multicast in IP/DWDM Internet

Wang Xingwei, Cheng Hui
Computing Center
Northeastern University
Shenyang, 110004, China
86-24-83687661
wangxw@mail.neu.edu.cn

Zheng Ludi
Bell Labs Research China
Beijing, 100080, China

Li Jia, Huang Min
College of Information Science and Engineering
Northeastern University
Shenyang, 110004, China

ABSTRACT
The paper discusses integration QoS routing algorithm in IP/DWDM Internet. Given a QoS multicast request and the delay interval required by users, we propose an algorithm, which can find a flexible-QoS-based cost suboptimal routing tree. The algorithm constructs the multicast tree based on multipopulation parallel genetic simulated annealing algorithm, and assigns wavelengths to the tree based on the wavelength graph. It integrates routing and wavelength assignment as a single process. The objective of routing is to find a cost suboptimal multicast tree. The objective of wavelength assignment is to minimize the delay of the multicast tree by minimize the number of wavelength conversion. Thus both the cost of multicast tree and the user QoS satisfaction degree approach the optimal. The proposed algorithm also considers load balancing strategy. We discuss the realization mechanisms of the algorithm. Simulation results show that the algorithm is both feasible and effective.

Keywords
IP/DWDM, integration QoS routing, multicast, wavelength assignment, multipopulation parallel genetic simulated annealing algorithm, load balancing.

1. INTRODUCTION
IP/DWDM (Dense Wavelength-Division Multiplexing)[1] Internet has emerged as a promising candidate for next-generation networks providing high channel bandwidth and low communication latency. Providing QoS (Quality of Service)[2] and multicast[3] is the essential capability for next-generation networks. Hence, providing QoS multicast in IP/DWDM Internet is very necessary. It means to run efficient routing algorithm, which can find the cost suboptimal multicast tree and assign wavelengths to it. It has been proved that finding such a tree is NP-hard[4].

A single population genetic algorithm[5] is powerful and performs well on a broad class of problems. However, better results can be obtained by introducing many populations, called subpopulations. Every subpopulation evolves for a few generations isolated (like the single population genetic algorithm) before one or more chromosomes are exchanged between the subpopulations. The Multipopulation parallel genetic algorithm[6] models the evolution of a species in a way more similar to nature than the single population genetic algorithm. Three different models for parallel genetic algorithms exist: the global model, the diffusion model and the migration model.

The proposed algorithm is based on the migration model. The migration model divides the population in multiple subpopulations. These subpopulations evolve independently from each other for a certain number of generations (isolation time). After the isolation time a number of chromosomes are distributed between the subpopulations (migration). The number of exchanged chromosomes (migration rate), the selection method of the chromosomes for migration and the scheme of migration determines how much genetic diversity can occur in the subpopulations and the exchange of information between subpopulations.

Multipopulation parallel genetic algorithm and simulated annealing algorithm[7] are two standard techniques for hard combinatorial optimization problems. A new algorithm is derived by combining these two algorithms well, that is multipopulation parallel genetic simulated annealing algorithm (MPGSAA)[8–10]. End-to-end delay is an important QoS parameter and decides the user QoS satisfaction degree. The proposed algorithm generates cost suboptimal multicast tree based on MPGSAA, and constructs an algorithm for wavelength assignment based on the basic idea of wavelength graph proposed by Chi and Tse[11]. The algorithm for wavelength assignment minimizes the delay of the multicast tree. We integrate the algorithm for wavelength assignment into the process of generating the multicast tree. Thus avoid the case that no wavelength resources can be assigned or the assignment result leads to poor QoS performance for the multicast tree. So the cost of multicast tree can approach the optimal, and the user QoS satisfaction degree is also satisfied simultaneously.

2. MODEL DESCRIPTION
2.1 IP/DWDM Internet Model
IP/DWDM optical Internet can be modeled by a directed and connected graph \(G(V,E)\), where \(V\) is the set of nodes representing optical nodes and \(E\) is the set of edges representing optical fibers that connect the nodes. Each edge carries two oppositely-directed fibers for data transmission in the two directions of the edge. Each directed fiber is called a link.
Every node \( v_i \in V \) has multicast capability, i.e., is equipped with an optical splitter[12]. We assume an optical signal can be split into an arbitrary number of optical signals at a splitter. Because the development of an all-optical wavelength converter is still in its early stage and the optoelectronic conversion not only is very expensive but also has limited performance, we assume only partial nodes are equipped with full-range wavelength converter[12] in IP/DWDM Internet. The full-range wavelength converter is able to convert optical signal on a wavelength into any other wavelengths. The wavelength conversion also introduces additional processing and control delay called wavelength conversion delay. We assume the conversion between any two different wavelengths has the same delay at any optical node with converter, i.e., \( t(v_i) = t \). If wavelength conversion doesn’t happen at some intermediate node \( v_i \) in a multicast tree, then set \( t(v_i) = 0 \).

Each link \( e_j = (v_i, v_j) \in E \) is associated with three parameters:

- \( \Lambda_j = \Lambda(e_j) \), the set of available wavelengths.
- \( \delta_j = \delta(e_j) \), the transmission delay. \( \delta(e_j) = \delta(e_j^i) \).
- \( c_{ij} = c(e_j) \), the cost.

### 2.2 Mathematical Model

In graph \( G(V, E) \), we consider a multicast request for multicast connection setup. \( R(s, D, \Delta) \), where \( S \) is the source node, \( D \) the set of destinations. Different from the previous algorithms[13,14], we define \( \Delta \) to be the delay requirement interval required by the user. It will be practical to measure the delay requirement by an interval because the information of the network is inaccurate in practice and the user QoS requirement is often flexible[15]. The minimal value and the maximum value of the interval depend on the user and the network application.

The route of the multicast connection is a tree \( T = (X, F), X \subseteq V, F \subseteq E \). The total cost of \( T \) is defined as:

\[
Cost(T) = \sum_{e_j \in F} c(e_j).
\]

The communication delay on a path consists of two components: link transmission delay and wavelength conversion delay. Let \( P(s, d_i) \) denote the path from source node \( S \) to any destination node \( d_i \) in \( T \). The delay between \( S \) and \( d_i \) along \( T \), denoted by \( D_{sd_i} \), can be represented as:

\[
D_{sd_i} = \left[ \sum_{e_j \in P(s, d_i)} t(v_j) + \sum_{e_j \in P(s, d_i)} \delta(e_j) \right].
\]

The delay of \( T \) is defined as:

\[
Delay(T) = \text{Maximize}\{ D_{sd_i}, \forall d_i \in D \}.
\]

It is the maximum delay between the source node and all the destination nodes.

Set \( \Delta = [\Delta_{low}, \Delta_{high}] \). The user QoS satisfaction degree is defined as follows:

\[
\text{Degret}(QoS) = \begin{cases} 1 & \text{if } \Delta_{low} < \text{Delay} < \Delta_{high} \\ 0 & \text{otherwise} \end{cases}
\]

The key objective of optimization considered by us is that the cost of the tree is as small as possible. At the same time the algorithm should select the links with more available wavelengths first to balance the network load and reduce the call blocking probability. The load on a link is defined as the number of channels over the link. We can adjust it by defining cost functions on links properly. For example, by defining heuristic cost functions, for the links on which the available wavelengths are more, the cost takes smaller value, otherwise, takes larger value. In our algorithm, define \( c(e_j) = W - |\Lambda(e_j)| \).

### 3. DESIGN OF THE ALGORITHM

#### 3.1 Expression of the Solution

We denote the solution by binary coding. Each bit of binary cluster corresponds to a different node. The graph corresponding to the solution \( S \) is \( G'(V', E') \). Let the function \( bit(S, i) \) denotes the \( i \)th bit of \( S \). If and only if \( bit(S, i) = 1 \), then \( v_i \in V' \). For our problem, every solution \( S \) corresponds to a tree \( T'(X', F') \), which is the minimum cost spanning tree of \( G' \). \( T' \) spans the given nodes set \( U \), which consists of the source node and all the destination nodes.

Another problem is that \( G' \) may be unconnected. Thus, every subgraph of \( G' \) has a minimum cost spanning tree, the solution \( S \) corresponds to a minimum cost spanning forest, also denoted by \( T'(X', F') \).

If \( G' \) is unconnected, we add penalty value to the cost and take smaller \( \text{Degret}(QoS) \) for the solution.

Thus, every solution \( S \) corresponds to a graph \( G' \), which corresponds to a minimum cost spanning forest \( T' \) (a forest can have only one tree). Prune \( T' \), the forest \( T' \) corresponding to solution \( S \) is obtained.

#### 3.2 The Algorithm for Wavelength Assignment

If \( T_i \) is a tree, we assign wavelengths to it. The objective of the proposed algorithm for wavelength assignment is to minimize the delay of the tree by minimizing the number of wavelength conversion. Thus the user QoS satisfaction degree will be high.

The proposed algorithm is based on the ideas of wavelength graph. First we construct wavelength graph \( WG \) for the tree \( T_i (X_i, F_i) \).

The set of available wavelengths on each link belonging to \( T_i \) is known. The construction method is stated as follows:

1) \( N = \left| X_i \right| \), \( w = \bigcup_{e_j} \Lambda(e_j) \) In \( WG \), we create \( N \times w \) number of nodes, namely \( \nu_{ij} \), for \( i = 1, 2, \cdots, w \) and \( j = 1, 2, \cdots, N \). All the nodes are arranged into a matrix with \( W \) rows and \( N \) columns. Row \( i \) represents the corresponding wavelength \( \lambda_i \) and each
column \( j \) represents a node \( v'_j \) in \( T_i \). A mapping table is created to record the corresponding relationship between \( i \) and \( \lambda'_i \), and another is created to record the relationship between \( j \) and \( V'_j \). The two tables will help reverse map the paths in \( WG \) back to the paths and wavelengths in \( T_i \).

2) For \( i = 1, 2, \ldots, w \), in the \( i \) th row, we add a horizontal directional link \((v_j, v_h)\) between column \( j \) and column \( h \) if there exists a link \( e'_{jh} = (v'_j, v'_h) \) in \( T_i \) from node \( v'_j \) to node \( v'_h \) and the wavelength \( \lambda'_j \) is available on this link. We assign the transmission delay \( \delta(e'_{jh}) \) as its weight.

3) For \( j = 1, 2, \ldots, N \), in the \( j \) th column, for \( \forall i_1, i_2, i_1 \neq i_2 \), we add a vertical bidirectional link \((v_{ij}, v_{i_2})\) between row \( i_1 \) and row \( i_2 \) if node \( v'_j \) in \( T_i \) has the wavelength conversion capability. We assign the wavelength conversion delay \( t \) as its weight.

Using the above steps the wavelength graph \( WG \) is constructed. A vertical link in \( WG \) represents a wavelength conversion at a node and a horizontal link in \( WG \) represents an actual link in \( T_i \).

For convenience, we denote the nodes in \( WG \) by sequential node number \( 1 \sim N \times w \). The sequential node number for the node in \( l \) th row and \( j \) th column in \( WG \) is:

\[
x = (i-1) \times N + j.
\]

Treat the wavelength graph \( WG \) as an ordinary network topology graph. Run the following algorithm.

**INPUT:** wavelength graph \( WG \), the column numbers corresponding to the source node and all the destination nodes in the matrix: \( J_1, J_2, J_3, \ldots, J_d \).

**OUTPUT:** the wavelength assignment result for \( T_i \).

**begin**

for \( k = 1, k \leq m, k + + \)

\{

for \( i = 1, i \leq w, i + + \)

\{

\[
x = (i-1) \times N + j;
\]

for \( j = 1, j \leq w, j + + \)

\{

\[
y_{jk} = (j-1) \times N + j_k;
\]

Apply the Dijkstra’s shortest path algorithm to find the shortest path \( P(x_i, y_{jk}) \) from node \( x_i \) to node \( y_{jk} \); 

\[
P(x_i, y_{jk}) = \min\{P(x_i, y_k), 1 \leq j \leq w\};
\]

\}

\}

\}

\}

\}

**end**

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

\[
P(x, y_k) = \min\{P(x_i, y_k), 1 \leq i \leq w\};
\]

Using the above two expressions and the two mapping tables created in step 1, we can reversely map the paths consisted of the sequential node numbers back to the links and wavelengths in \( T_i \) conveniently. Thus the wavelength assignment is completed.

The time complexity of the above algorithm is \( O(mN^2w^4) \). \( m \) is the number of destination nodes. \( N \) is the number of nodes in \( T_i \). \( w \) is the number of wavelengths which are available on at least a link in \( T_i \). We can see they are all small integers. In addition, all the wavelength assignments for solutions except the final solution are preassigning strategy. Its objective is to get the cost (end-to-end delay) of the path, so it needn’t storage much data and has less space complexity.

### 3.3 Fitness Function

After assigning wavelengths to \( T_i \), the delay of \( T_i \) is determined. Thus Degree(QoS) is determined. The fitness of solution \( S \) is obtained by computing fitness function \( f \):

\[
f(S) = \frac{\text{Cost}(T_i) + [\text{count}(T_i) - 1] \times \rho}{\text{Degree(QoS)}}
\]

\[
= \sum_{e \in T_i} c(e) + [\text{count}(T_i) - 1] \times \rho
\]

\[
\text{Degree(QoS)}
\]

Where \( \text{count}(T_i) \) is the number of trees in the forest \( T_i \). \( \rho \) is a constant.

Thus, for solution \( S \), the smaller is \( f(S) \), the better is the solution.

### 3.4 Choosing of Initial Temperature

Set \( t_0 = K\delta \), where \( K \) is a sufficiently large number, \( \delta = \max\{f(j) \mid j \in Sp\} - \min\{f(j) \mid j \in Sp\} \).

\( Sp \) denotes the solution space.

\( \delta \) can be estimated simply:

\[
\max\{f(j) \mid j \in Sp\} \leq C_{\text{graph}} \ 	ext{ (the total cost of the present network graph)},
\]

\[
\min\{f(j) \mid j \in Sp\} \geq C_U \ 	ext{ (the sum of the costs of all nodes belonging to } U \).
\]

Then \( \delta = C_{\text{graph}} - C_U \).
3.5 Statement of the Algorithm

Initialize the control parameters, include the subpopulations number: \( M \), the size for every subpopulation: \( n_p \), the predefined maximum generation number: \( \text{MAX}_-\text{GN} \), the individual generation number: \( n_G \), the crossover probability for subpopulation \( i : \rho_i(i), 1 \leq i \leq M \), the mutation probability for subpopulation \( i : \rho_m(i), 1 \leq i \leq M \), the temperature decreasing coefficient: \( \alpha \), the initial temperature for subpopulation \( i : t_0(i), 1 \leq i \leq M \).

1) Initialize \( M \) random subpopulations. Set \( \text{GN} = 0 \), \( \text{GN} \) denotes the total generation number that the subpopulation has experienced. Set \( k = 0 \), \( k \) denotes the number of decreasing temperature. Set \( f(S_{op}) = \infty \), \( S_{op} \) denotes the global optimum solution. Set \( \text{Loop} = 0 \), \( \text{Loop} \) is a counter variant.

2) If \( \text{Loop} < n_o \), go to step 3, otherwise, go to step 5.

3) For subpopulation \( i, 1 \leq i \leq M \), implement the following operations to generate an offspring subpopulation.

   a) Evaluate the fitness of every chromosome: \( f(S_j), j = 1,2, \ldots, n_p \):

   b) Select the chromosomes \( S_j, S_k (j \neq k) \) randomly and generate a random number \( num \in [0,1] \). If \( num > \rho_i(i) \), \( S_j, S_k \) are accepted for offspring subpopulation directly, otherwise, implement the crossover operation to generate two new chromosomes \( S'_j, S'_k \). Evaluate the fitness \( f(S'_j), f(S'_k) \). We have \( \Delta f' = f(S'_j) - f(S_j) \). If \( \Delta f' < 0 \), accept \( S'_j \) for offspring subpopulation; if \( \Delta f' > 0 \), then accept \( S'_k \) for offspring subpopulation at the probability \( \exp(-\Delta f'/t_i(i)) \). We have \( \Delta f' = f(S'_k) - f(S_j) \). If \( \Delta f' < 0 \), accept \( S'_k \) for offspring subpopulation; if \( \Delta f' > 0 \), then accept \( S'_k \) for offspring subpopulation at the probability \( \exp(-\Delta f'/t_k(i)) \). If \( S'_j, S'_k \) aren’t accepted, \( S_j, S_k \) are accepted for offspring subpopulation directly.

   Repeat b) \( n_p/2 \) times, and get the offspring subpopulation \( i' \).

   c) For every chromosome \( S_j \) in \( i' \), generate a random number \( num \in [0,1] \). If \( num > \rho_m(i) \), \( S_j \) is accepted for offspring subpopulation directly, otherwise, implement the mutation operation to generate a new chromosome \( S'_j \). Using the above method mentioned in b) to decide whether or not to accept \( S'_j \) for offspring subpopulation. If not accepted, \( S_j \) is accepted for offspring subpopulation directly. After this operation, denote the offspring subpopulation as subpopulation \( i' \).

4) \( \text{GN} = \text{GN} + 1 \). \( \text{Loop} = \text{Loop} + 1 \), go to step 2.

5) First find the optimum chromosome of each subpopulation, and we get \( M \) chromosomes. Then find the optimum one \( S \) among the \( M \) chromosomes. Replace the worst chromosome of every subpopulation using \( S \). If \( f(S) < f(S_{op}) \), \( S_{op} \leftarrow S \) (i.e., replace \( S_{op} \) using \( S \)).

6) If \( \text{GN} = \text{MAX}_-\text{GN} \), the algorithm stops, otherwise, modify the annealing temperature for each subpopulation, i.e., \( t_{k+1}(i) = at_i(i) \) \( (k \geq 0, 0 < \alpha < 1, 1 \leq i \leq M) \). \( k = k + 1 \). \( \text{Loop} = 0 \). Go to step 2.

When the algorithm stops, \( S_{op} \) is the final solution.

4. DISCUSSION ON THE REALIZATION MECHANISMS OF THE ALGORITHM

Parallel algorithms were developed to speed up the computation by harnessing the power of parallel computers or multiple processors computer. During the parallel evolution process of the multiple subpopulations, each subpopulation evolves independently from each other for a certain number of generations (isolation time). After the isolation time the optimum solution (chromosome) is distributed between all the subpopulations.

We set that the population size of each subpopulation is the same and that the crossover probability, mutation probability and temperature control parameters of each subpopulation may be different.

This is a synchronous parallel algorithm. The implementation of the algorithm should adopt the MIMD (Multiple Instruction stream Multiple Data stream) computer architecture[10]. Assign the same number of processors as the number of subpopulations, and each processor processes the evolutionary computing of a subpopulation independently.

There needs a kind of synchronization mechanism between different processes operating on different processors, i.e., after one processor finishes its isolation time, it stops to judge if the other ones have finished their isolation time. If there exists a processor that hasn’t finished, all the others that have finished must wait till all the processors finish their isolation time.

There are two kinds of realization mechanisms for MPGSAA. One is to establish the shared memory, the other is to designate the control processor.

The first method is to establish a shared memory for all the subpopulations. Thus all the subpopulations communicate through a global shared variant. The present global optimum solution is also distributed between all the subpopulations through the global shared variant. The global shared variant is a kind of critical resource, and the lock mechanism should apply to it. Each processor should create its own critical region for the global shared variant to realize the synchronization between all the processors. Figure 1 illustrates this method.

The second method is to designate a new processor as the control processor. The control processor can also be designated among all the processors used to process the subpopulations by election. The control processor is responsible for the distribution of the present
global optimum solution and the synchronization between all the processors. Figure 2 illustrates this method.

5. SIMULATION RESEARCH AND DISCUSSION

Our simulation experiments are operated on a single processor computer, and the parallel algorithm is implemented in a serial manner (pseudo-parallel). The following simulation research and results are based on NSFNET[16] network topology. The simulation research is composed of two parts: evaluation on the cost of the final multicast tree and evaluation on the QoS performance of the final multicast tree.

Referring to the simulation model established by Huang[13], we set the transmission delay to be an small integer between 1 and 10 on each link, which is in direct proportion to the length of the link. And we set the wavelength conversion delay to be a constant integer between 1 and 10. We choose the higher 50% of all the nodes to be equipped with wavelength converters according to the node degree. Set 

\[ \Lambda = 20 \] , and 

\[ 10 \leq |\Lambda(e_{ij})| \leq 15. \]

If the fitness of some chromosomes is too large, the difference between other chromosomes can be neglected. To avoid this, when \( \text{Degree}(QoS) \) is less than some smaller value \( \text{val} \), we take \( \text{Degree}(QoS) = \text{val} \) to calculate the fitness. When the solution corresponding to the chromosome is unfeasible, we also take \( \text{Degree}(QoS) = \text{val} \).

We have chosen the appropriate values for the parameters of MPGSAA by simulation research.

5.1 The Evaluation on the Cost Performance

By contrasting solutions obtained by running the algorithm with the optimal cost solution obtained by exhaustive search, quantitative analysis is made. The result of simulation on the cost is shown in Table 1.

In the above table, C1 denotes Session No, C2 denotes the ratio of session nodes in the network, C3 denotes the optimal cost for the multicast session. \( \leq 1\% \) means that the ratio of the difference between the cost of the obtained solution and the optimal cost (i.e. the deviation of the cost of the solution) and the optimal cost is \( \leq 1\% \). \( \leq 5\% \) means that the ratio of the difference and the optimal cost is \( >1\% \) and \( \leq 5\% \), and others have the similar meanings. The value under each ratio interval means the ratio of the number of the solutions whose deviation ratio is in this interval and the total number of solutions. We run the algorithm 100 times and get 100 solutions for each session.

These session nodes are chosen randomly from sparse mode[17] to dense mode[18]. We can get from the table that for the actual topology of NSFNET the accuracy degree of the cost of the solutions obtained by the algorithm is very high. Even the cost of the solution is the optimal sometimes.

5.2 The Evaluation on the QoS Performance

We define the user QoS satisfaction degree, and consider the QoS performance of chromosomes when calculating the fitness using MPGSAA. Hence we consider both the cost and the maximum end-to-end delay when choosing chromosomes. To evaluate the QoS performance of our algorithm, we compare the delay of the multicast tree obtained by our algorithm and the delay of
multicast tree obtained by running MPGSAA without considering the user QoS satisfaction degree. Figure 3 shows the result. From figure 3, we can see that the delay of the multicast tree obtained by our algorithm is less than the one without QoS support. Our algorithm improves the QoS performance of the multicast tree efficiently.

6. CONCLUSIONS
This paper analyzes the actual IP/DWDM Internet and describes the network model. Then we define the mathematical model for QoS multicast in IP/DWDM Internet. Due to the complexity of network dynamic, the network state information is uncertain inherently. The QoS requirement of the user is also flexible at most time. So it’s more practical to measure it by an interval. Based on the above idea, we define the concept-the user QoS satisfaction degree. Then based on MPGSAA and the idea of wavelength graph, we construct a QoS routing algorithm for multicast in IP/DWDM Internet. By the design of MPGSAA, the proposed algorithm can find a cost suboptimal routing tree. Every time, a feasible multicast tree is found, we assign wavelengths to it to minimize the delay of the multicast tree. Thus we guarantee that the QoS performance of the routing tree is also good. We integrate the wavelength preassigning into the construction of the routing tree to make a better tradeoff between the cost and the end-to-end delay. Two kinds of realization mechanisms of the algorithm are proposed and discussed. We realize the algorithm by simulation. Simulation experiments evaluate the cost performance and the QoS performance, respectively. Simulation results show that the proposed algorithm solves the QoS multicast problem in IP/DWDM Internet efficiently.

7. ACKNOWLEDGMENTS
This work is supported by the National Natural Science Foundation of China under Grant No.60003006(jointly supported by Bell labs) and No.70101006; the National High-Tech Research and Development Plan of China under Grant No.2001AA121064; the Modern Distance Education Key Technology and Supporting Service System Program of Ministry of Education of China; the Foundation of Shenyang Science and Technology Committee of China.

8. REFERENCES