Georgia State University
ScholarWorks @ Georgia State University

Computer Science Faculty Publications | Department of Computer Science

2004

Simulated Annealing Routing and Wavelength Lower Bound Estimation on WDM Optical Multistage Networks

Ajay K. Katangur

Yi Pan
Georgia State University

Martin D. Fraser

Follow this and additional works at: https://scholarworks.gsu.edu/computer_science_facpub

 Part of the Computer Sciences Commons, and the Engineering Commons

Recommended Citation

Katangur, A.K., Pan, Y., and Fraser, M.D. (2004). Simulated Annealing Routing and Wavelength Lower Bound Estimation on WDM Optical Multistage Networks. Optical Engineering, 43(5), 1080-1091. doi: 10.1117/1.1690276 Also available at: http://digitalarchive.gsu.edu/computer_science_facpub/16/

This Article is brought to you for free and open access by the Department of Computer Science at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Computer Science Faculty Publications by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.

Simulated annealing routing and wavelength lower bound estimation on wavelength-division multiplexing optical multistage networks

Ajay K. Katangur
Yi Pan
Martin D. Fraser
Georgia State University
Department of Computer Science
Atlanta, Georgia 30303
E-mail: pan@cs.gsu.edu

Abstract. Multistage interconnection networks (MINs) are popular in switching and communication applications and have been used in telecommunication and parallel computing systems for many years. Crosstalk a major problem introduced by an optical MIN, is caused by coupling two signals within a switching element. We focus on an efficient solution to avoiding crosstalk by routing traffic through an $N \times N$ optical network to avoid coupling two signals within each switching element using wavelength-division multiplexing (WDM) and a time-division approach. Under the constraint of avoiding crosstalk, the interest is on realizing a permutation that uses the minimum number of passes for routing. This routing problem is an NP-hard problem. Many heuristic algorithms are already designed by researchers to perform this routing such as a sequential algorithm, a degree-descending algorithm, etc. The genetic algorithm is used successfully to improve the performance over the heuristic algorithms. The drawback of the genetic algorithm is its long running times. We use the simulated annealing algorithm to improve the performance of solving the problem and optimizing the result. In addition, a wavelength lower bound estimate on the minimum number of passes required is calculated and compared to the results obtained using heuristic, genetic, and simulated annealing algorithms. Many cases are tested and the results are compared to the results of other algorithms to show the advantages of simulated annealing algorithm. © *2004 Society of Photo-Optical Instrumentation Engineers.* [DOI: 10.1117/1.1690276]

Subject terms: genetic algorithm; heuristic algorithms; lower bound estimate; multistage interconnection networks; optical networks; routing; simulated annealing; wavelength-division multiplexing.

Paper WDM-02 received Nov. 6, 2002; revised manuscript received Sep. 24, 2003; accepted for publication Oct. 16, 2003.

1 Introduction

A multistage interconnection network (MIN) is very popular in switching and communication applications. It has been used in telecommunication and parallel computing systems for many years. This network consists of N inputs, N outputs, and n stages ($n = \log_2 N$). Each stage has $N/2$ switching elements (SEs); each SE has two inputs and two outputs connected in a certain pattern.[1] The most widely used MINs are the electronic MINs. As optical technology advances, there is a considerable interest in using optical technology to implement interconnection networks and switches.[2] In electronic MINs electrical signals are used to transmit messages, whereas in optical MINs optical signals are used for this task.

Electronic and optical MINs have many similarities, but there are some fundamental differences between them such as the crosstalk problem in the optical switches.[2] To avoid the crosstalk problem, various approaches have been proposed by many researchers. In this paper, the main focus is on the Omega network, which has a shuffle-exchange connection pattern. To transfer messages from a source address to a destination address on an optical Omega network without crosstalk, the messages are to be divided into several groups (independent subsets) and then delivered using one time slot for each group.[3,4] In each independent subset, the paths of the messages going through the network are crosstalk free.[4]

The motivation for this research is to reduce the number of independent subsets so that the messages can be sent in as few time slots as possible. Using wavelength-division multiplexing[5] (WDM) approach, if we have 10 wavelengths, for example, and if we obtain 9 independent subsets, we can route all the messages at a single time in one pass. If we obtain 12 independent subsets, we can route 10 independent subsets in one pass and the next 2 independent subsets in another pass. Thus, even if all the independent subsets are sent at the same time, as long as they use different wavelengths,[5] there will be no conflicts and hence no crosstalk will be induced. It is clear that with a fixed number of wavelengths, a reduction of the number of independent sets can also decrease the total number of passes. A common assumption is that each channel (wavelength) can be switched independently,[5] and there is no constraint that each switch must be set consistent for all the input/output channels.

 0091-3286/2004/$15.00 © 2004 Society of Photo-Optical Instrumentation Engineers

Previously the genetic algorithm (GA) was used successfully to improve the performance over the heuristic algorithms such as the sequential, degree-descending algorithms. The drawback of the GA is its long running time. In this paper, the main concentration is on the improvement of the average number of passes considering the bandwidth of wavelengths available and run-time performance using the simulated annealing (SA) algorithm.[6] Also lower bound estimate on the minimum number of wavelengths required is calculated and compared to the results obtained by heuristic, GA, and SA algorithms. SA (Ref. 6) is an optimization method used for combinatorial optimization problems.[7] The SA method begins with a nonoptimal initial configuration and works on improving it by selecting a new configuration and calculating the cost differential.[6] Different types of operators for the SA algorithm are used and different cases are tested and analyzed in this paper.

We previously[2] proposed different methods for avoiding crosstalk in optical MINs. The window method[8] we proposed is used to find the conflicts between messages and to draw the conflict graph. This research uses our previous idea of time-division multiplexing, but rather concentrates on how to minimize the number of passes required for routing.

The main contribution of this paper is the successful application of the SA algorithm to the routing problem to improve the average number of passes and lower bound estimation on the number of wavelengths required.

The rest of the paper is organized as follows. Section 2 presents a formal description of optical MINs. Section 3 presents the different heuristic algorithms available for routing. Section 4 discusses the SA algorithm in general. Section 5 gives a detailed description of how the SA algorithm is applied to the routing problem. Section 6 provides simulation results to evaluate the performance of the proposed algorithm. Section 7 presents the lower bound estimate calculation. Section 8 concludes this paper.

2 Optical MIN

2.1 *MIN*

MINs consists of N inputs and N outputs, which are interconnected by n ($n=\log_2 N$) stages of switching elements.[1] There are two inputs and two outputs for each SE. Each stage consists [1] of $N/2$ SEs.

In this paper, the interest is on the Omega network, which has a shuffle-exchange connection pattern. It has N inputs, N outputs, and n stages, where $n=\log_2 N$. To connect the source address to the destination address, the address is shifted 1 bit to the left circularly in each connection such as source to the first stage, one stage to the next stage, etc. For example, in an 8×8 network, the inputs and outputs are 000, 001, 010, 011, 100, 101, 110, and 111, which forms a permutation. Each connection between the stages is shuffle-exchanged, as shown in Fig. 1.

Permutation[8] is an important problem in parallel computer systems since it is a popular data movement operation. An $N\times N$ MIN with N inputs and N outputs where $N=2^n$ is considered. A permutation of a network[8] is a pairing such that each input and output appears in exactly one pair. The source-destination permutation can be regarded as a one-to-one mapping. For instance, the eight inputs and eight outputs form a permutation. In general the destinations can be randomly permuted (the message 000→101, it may also go to 110 or 001 or any of the other outputs, and this is the reason why it is termed ''the destinations can be randomly permuted''), as shown in Fig. 1. To route this permutation in the Omega network, the shuffle exchange connection pattern is considered. Sending all the inputs to outputs in one time slot (pass) can be done as shown in Fig. 2. From Fig. 2, we can see that there will be crosstalk in all the switching elements if all the inputs are routed to the outputs in one pass.

$$
\begin{aligned}
000 &\rightarrow 101\\
001 &\rightarrow 010\\
010 &\rightarrow 111\\
011 &\rightarrow 000\\
100 &\rightarrow 011\\
101 &\rightarrow 110\\
110 &\rightarrow 001\\
111 &\rightarrow 100
\end{aligned}
$$

Fig. 1 Shuffle exchange.

2.2 *Crosstalk in Optical-MIN*

Crosstalk[2] occurs when two signal channels interact with each other. When crosstalk happens, a small fraction of the input signal power may be detected at another output although the main signal is detected at the right output. For this reason, when a signal passes many switching elements, the input signal will be distorted at the output due to the loss and crosstalk introduced on the path.[2] There are two ways in which optical signals can interact in a planar switching network. The channels carrying the signals could cross each other in order to embed a particular topology. Alternatively, two paths sharing an SE will experience some undesired coupling from one path to another[2] within an SE. This is shown in Fig. 3. Each SE can be in two connecting schemes, as shown in Fig. 3. Since these two ways in Fig. 3 will cause crosstalk, a necessity arises to avoid these situations to happen in all the SEs.

2.3 *Approaches to Avoid Crosstalk*

Many approaches have been proposed to reduce the negative effect of crosstalk. One way to solve crosstalk is to use a $2N\times 2N$ regular MIN to provide the $N\times N$ connection,[3,9]

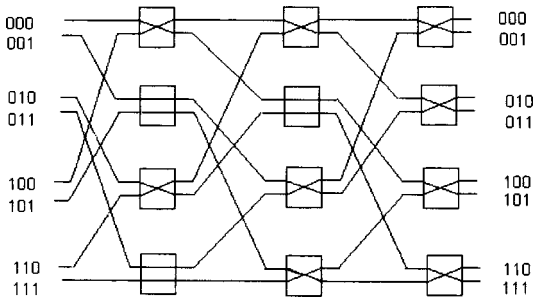


Fig. 2 Permutation in an 8×8 Omega network.

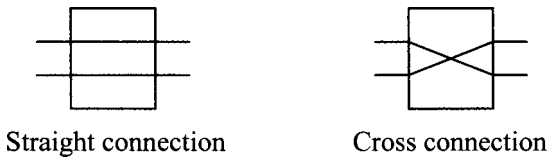


Fig. 3 Two types of switching connections.

which is the space domain approach. But half the inputs and outputs are wasted in this approach. Another more efficient solution, which is the method used in this paper is to route the traffic through an $N \times N$ optical network to avoid coupling of two signals within each SE. This idea can be implemented using the time domain approach[8] and in the legal passing ways are as shown in Fig. 4.

2.4 *Avoiding Crosstalk in Omega Network*

To avoid crosstalk, the time domain approach[4] can be used, which is to partition the set of the connections into several subsets[10] such that the connections in each subset can be established simultaneously in the network without crosstalk, i.e., to route all the inputs in several groups (independent subsets), such that there will be no crosstalk between messages in each independent subset. Another approach is to use WDM technology. To utilize the large available bandwidth in optical fibers, a single fiber can be partitioned into multiple communication channels using WDM technology. By using WDM approach,[5] each such independent subsets of groups can be routed in a single time slot or in a reduced number of time slots, depending on the number of different wavelengths available. For example, using the WDM approach,[5] if we have 10 wavelengths, for example, and if we obtain 9 independent subsets, we can route all the messages at a single time in one pass. If we obtain 12 independent subsets, then because we have only 10 wavelengths available we can route 10 independent subsets in one pass and the next 2 independent subsets in another pass. Thus, even if all the independent subsets are sent at the same time, as long as we have different wavelengths,[5] there will be no conflicts and hence no crosstalk will be induced. Thus, from the performance aspect, using WDM, messages are separated without conflicts with other messages even in the same group, resulting in the reduction in the total number of passes. We use a combined approach that uses both time-domain and WDM-domain approaches. If the number of independent subsets found is smaller than the number of available wavelengths, we can simply send all the messages in one time slot. Otherwise, we may send w subsets of messages in each time slot, where w is the number of wavelengths available, and use several time slots to finish the transmission. The research described here aims to reduce the number of independent subsets, thus effectively reducing the number of passes in a WDM network. In an SE as shown in Fig. 4, there is no way to realize a permutation in a single pass

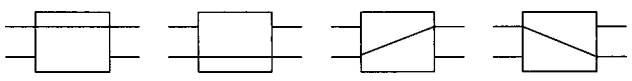


Fig. 4 Legal passing ways in a SE at a time.

$$
\begin{aligned}
000 &\rightarrow 100\\
001 &\rightarrow 001\\
010 &\rightarrow 010\\
011 &\rightarrow 011\\
100 &\rightarrow 000\\
101 &\rightarrow 101\\
110 &\rightarrow 110\\
111 &\rightarrow 111
\end{aligned}
$$

Fig. 5 Shuffle exchange.

through an optical network without crosstalk. The reason is at least the two input links on an input switch or the two output links on an output switch cannot be active in the same pass.[2] Thus, at least two passes are required to realize a permutation.[5] For example, the permuation shown in Fig. 5 is considered. The eight messages shown in Fig. 5 can be routed in two passes, as shown in Fig. 6. If more than two wavelengths are available they can be sent at a single time in one pass. Our goal is to minimize the number of wavelengths required to route all the inputs to the outputs without crosstalk for randomly generated permutations.

3 Different Heuristic Routing Algorithms

3.1 *Window Method*

Since the messages to be sent to the network are to be distributed into several groups, a method is to be used to determine which messages should not be in the same group because they will cause crosstalk. The window method[8] is used to find conflicts among all the messages to be sent. This method, which has already been proved to be correct by other researchers,[8] can be described precisely as follows. Given a permutation, combining each source address and its corresponding destination address produces a matrix.[8] The optical window size is $m-1$, where $m=\log_2 N$ and N is the size of the network. We use this window on the produced matrix from left to right except on the first and the last columns. If two messages have the same bit pattern in any of the optical window, they will cause conflict in the network. That means they cannot be in the same group, hence, they have to be routed in different wavelengths. To see how the window method works, consider the example shown in Fig. 7 where the network size is 8 and the source addresses and the destination addresses are as shown in Fig. 5. The optical window can be applied on the permutation shown in Fig. 5 as shown in the figure, where example messages 000 and 100 in step 1 (window 0) have the same bit pattern of 00 inside the window and hence have a conflict. The bit patterns can be any of the four combinations of 00, 01, 10, and 11, and hence are shaded using different colors.

3.2 *Conflict Graph*

After using the window method, we can draw a conflict graph.[11] The number of the nodes is the size of the network. Each node represents a source address. If two nodes have

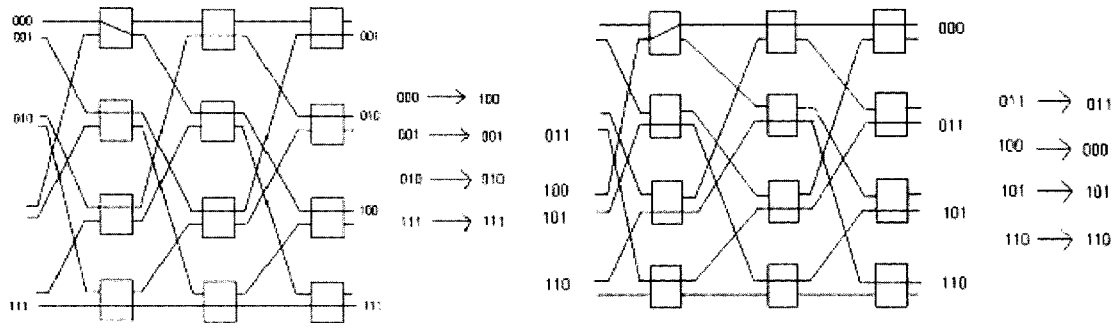


Fig. 6 Two passes for a specific permutation in an 8×8 Omega network.

conflict during routing, they are connected using an edge. For the example of Fig. 7, the conflict graph is shown in Fig. 8.

3.3 *Previous Five Routing Algorithms*

The previous five algorithms are the sequential increasing, sequential decreasing, degree increasing, degree descending[12] and the GA (Ref. 13). The purpose of these routing algorithms is to schedule the messages in different independent subsets to avoid the path conflicts in the network.[12] The more efficient the algorithm is, the fewer independent subsets it will generate, and depending on the bandwidth of wavelengths, available messages can be routed accordingly using WDM approach.[5] The order of the messages to be picked for scheduling is an essential cause for generating the different results. The basic idea of the routing algorithm is as follows:

while (not end of messages list)

Select one of the left messages;

Schedule the message in a time slot with no conflict with other messages that have been already scheduled.

There are many ways to decide the order of the scheduling. The four heuristic algorithms choose the message in the following manner:

1. Choose a message in increasing order of the message source address.
2. Choose a message in decreasing order of the message source address.
3. Choose a message based on the order of increasing degrees in the conflict graph.
4. Choose a message based on the order of decreasing degrees in the conflict graph.

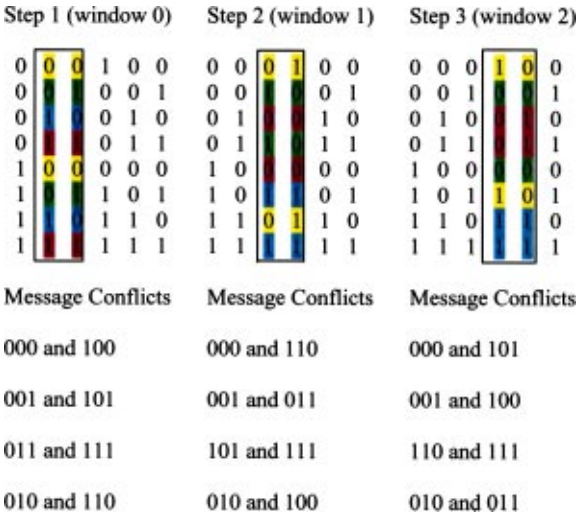


Fig. 7 Window method example.

The degree of each message in the conflict graph is the number of conflicts it has with other messages in the conflict graph. Scheduling the messages in decreasing degrees of the message conflicts will result in the best performance among these four algorithms. The GA is not as easy and straightforward, but it can be successfully used to reduce the number of passes in an optical MIN (OMIN). A brief introduction on how GAs are used for OMINs to reduce the number of independent subsets is provided in Sec. 3.3.1. More about how GAs are applied to OMIN can be found in Refs. 13 and 14.

3.3.1 *GA*

GAs are a part of evolutionary computing.[15,16] The GA is initialized with a set of solutions, which are represented by chromosomes. These solutions are called the population. Solutions from the initial population are taken and used to form a new population. This procedure is motivated by the expectation, by analogy with biological population, that the new population will be ''better'' than the old one. To select new solutions (offspring) to form a new population, the fitness of the original solution is the important criterion. If the fitness value is greater, they get more chances to reproduce. This is repeated generation by generation until some condition is satisfied such as the number of populations reaches the limit or the improvement of the best solution found so far is good enough for the research or there is no more improvement possible.

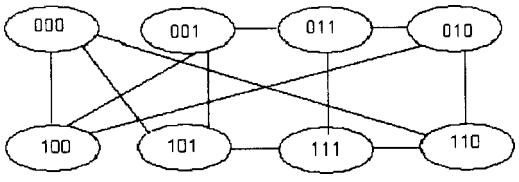


Fig. 8 Conflict graph.

Fig. 9 Perfect is the condition where all atoms lined up on crystal lattice sites, there are no defects, and this is the lowest energy "state" for this set of atoms.

3.3.2 *Performance of the GA over the other heuristic algorithms*

The GA improves the performance in terms of the average number of wavelengths required. It requires one or two fewer wavelengths[13] than that of the degree-descending algorithm, and requires two wavelengths fewer than that of the remaining algorithms. The GA was time consuming: it took hours to calculate the number of passes for large network sizes.

4 SA Algorithm

4.1 *What is SA?*

SA originated in the annealing processes found in thermodynamics and metallurgy.[6,15,16] SA was introduced by Metropolis and is used to approximate the solution of very large combinatorial optimization problems[7] (e.g., NP-hard problems). It is based on the analogy between the annealing of solids and solving optimization problems. When SA was first proposed,[16] it was most known for its effectiveness in finding near-optimal solutions for large-scale combinatorial optimization problems,[17] such as the traveling salesperson problem,[6] buffer allocation in production lines,[18] and chip placement problems in circuits[14] (finding the layout of a computer chip that minimizes the total area). But recent[7] uses of SA demonstrated that this class of optimization methods could be considered competitive with other approaches when there are optimization problems to be solved.

SA was derived from physical characteristics of spin glasses.[6,16] The principle behind SA is analogous to what happens when metals are cooled at a controlled rate. The slowly falling temperature enables the atoms in the molten metal to line up and form a regular crystalline structure that has high density and low energy. But if the temperature goes down too quickly, the atoms do not have time to orient themselves into a regular structure and the result is a more amorphous metal with higher energy (see Figs. 9 and 10).

In SA, the value of an objective function[19] that we want to minimize is analogous to the energy in a thermodynamic system. At high temperatures, SA enables function evaluations at faraway points and it is likely to accept a new point with higher energy. This corresponds to the situation in which high-mobility atoms are trying to orient themselves with other nonlocal atoms and the energy state can occasionally go up. At low temperatures, SA evaluates the objective function only at local points and the likelihood of it accepting a new point with higher energy is much lower. This is analogous to the situation in which the low-mobility atoms can only orient themselves with local atoms and the energy state is not likely to go up again.[6]

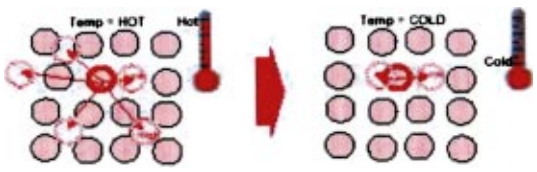


Fig. 10 To reach the "low energy state," "anneal" the material. Get it very hot: gives atoms energy to move around. Cool it very slowly. This gently restricts the range of motion until everything freezes into a low-energy configuration.

Obviously, the most important aim of SA is to avoid trapping in a local minimum and obtain a globally better solution by employing the so-called annealing schedule[16] or cooling schedule, which specifies how rapidly the temperature is lowered from high to low values. This is usually application specific and requires some experimentation by trial and error.

The following fundamental terminology[16] concerning SA is useful before going to a detailed description.

Objective function. An objective function $f(\cdot)$ maps an input vector $\mathbf{x}$ into a scalar $E=f(x)$, where each x is viewed as a point in an input space. The task of SA is to sample the input space effectively to find an $\mathbf{x}$ that minimizes E.

Generating function. A generating function $g(\cdot\,,\cdot)$ specifies the probability density function of the difference between the current point and the next point to be visited. Specifically, $\Delta x=(x_{\text{new}}-x)$ is a random variable with probability density function $g(\Delta x,T)$, where T is the temperature. For common SA used in combinatorial optimization applications, $g(\cdot\,,\cdot)$ is a function independent of temperature T. For our problem, we use the move set approach to generate the next solution.

Acceptance function. After a new point x_{new} has been evaluated, SA decides whether to accept or reject it based on the value of the acceptance function $h(\cdot\,,\cdot)$. The most commonly used acceptance function is the Boltzmann distribution function

$$h(\Delta E,T)=\exp(-\Delta E/T),$$

$$\Delta E=f(x_{\text{new}})-f(x),$$

where T is the temperature, and ΔE is the energy difference between x_{new} and x.

The common practice is to accept x_{new} with probability $h(\Delta E,T)$. Note that when ΔE is negative, SA accepts the new point because it reduces the energy. When ΔE is positive, SA may accept the new point and end up in a higher energy state or it may not accept the point. The lower the temperature, the less likely SA is to accept any significant high-energy states.

Annealing schedule. An annealing schedule regulates how rapidly the temperature T goes from high to low values. The easiest way of setting an annealing schedule is to decrease the temperature T by a certain percentage at each iteration.

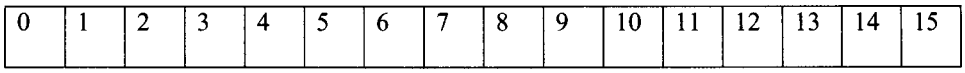


Fig. 11 Sorted vertex.

To find the next legal point, we usually define a move set,[16] denoted by $M(x)$, as a set of legal points available for exploration after x. Usually the move set $M(x)$ represents a set of neighboring points of the current point x in the sense that the objective function at any point of the move set will not differ too much from the objective function at x.

The basic simulated annealing algorithm includes the following steps.[14]

1. Start: Select an initial value of x from the input space. Select a large starting value for the temperature T, a value for the final temperature T_{final}, a value for the stopping condition t_s for example that is chosen as 10, and a suitable value for the temperature reducing parameter α. The t_s value is used because it will stop the algorithm if there are no changes to the solution after t_s iterations. We also choose a variable M so that M iterations are performed for each temperature.
2. Initialize the current energy E, to be equal to the energy of the initial solution, which is the initial value of x, and t_s to some value t_{stop}.
3. While ($t_{\text{stop}}>0$ and $T>T_{\text{final}}$).
4. For $i=1$ to M do steps 5 through 9.
5. Generate a new solution using the present solution.
6. Calculate the new energy of the new solution. If the new energy is less than the current energy accept the new solution, otherwise use the following strategy to decide.

   $R=$random number $(0<R<1)$,

   $Y=\exp(-\Delta E/T)$,

   If $(R<Y)$, then accept the solution,

   else reject it.

7. If the move is accepted then the current energy will be set to the new energy and the new solution is kept, otherwise the old solution is kept.
8. If the new solution is accepted, assign $t_{\text{stop}}=t_s$, else decrement t_{stop} by 1 ($t_{\text{stop}}=t_{\text{stop}}-1$).
9. Change the temperature by a factor α, $T=T\times\alpha$.
10. End of while.

5 SA Algorithm Used in This Research

5.1 *Move Sets in SA*

The move sets are the most important operators in simulated annealing approach. We used three types of move sets for SA in this research. They are inversion, translation, and switching.

5.1.1 *Inversion*

We have a sorted vertex and we apply inversion to this sorted list of vertices to generate a new solution. For example if we have a sorted vertex, as shown in Fig. 11.

Then we generate two random points and then replace that section in the opposite order. For example, if we generate the random points as 4 and 11, then we invert the numbers from 11 to 4 and store them in this order as shown in Fig. 12.

5.1.2 *Translation*

Here we randomly generate two points and then that section of the vertex is stored in between two randomly generated points. For example, if we generate 1 and 3 for the section to be replaced, and then 14 and 15, the section 1–2–3 is placed in between 14 and 15, as shown in Fig. 13. This is applied on the sorted list of vertices obtained after inversion is applied.

5.1.3 *Switching*

Here we randomly generate two points and then switch the points at those positions. Suppose if we generate 0 and 15, then 0 and 15 are interchanged in the previous sorted list of vertices, as shown in Fig. 14.

We used these three techniques and found out that inversion is mostly suitable for application in SA to generate a new solution. This is so because translation required more time when compared to inversion because of the random number generations and switching move set tends to rupture the solution.[16]

5.1.4 *Initial solution for the SA algorithm*

Initially we need a solution to proceed with the SA algorithm.[16] This is because SA cannot proceed without having any arbitrary solution to generate the next solution. For this purpose, the sorted list of vertices is formed with vertices taken in the increasing order. The number of inde-

Fig. 12 Numbers inverted from 11 to 4.

Fig. 13 Section 1–2–3 placed between 14 and 15.

pendent subsets required with this sorted list of vertices is calculated. We assign the current energy to the number of independent subsets we got initially with vertices taken in the increasing order and proceed to the SA algorithm.

5.2 *Parameters of SA Algorithm*

The parameters[14] used in the algorithm are the starting temperature, the final temperature, the temperature-cooling rate (α), the number of iterations (M) for a particular temperature value, and the stopping value (t_s). The starting temperature is used for the algorithm to start at a particular value of temperature. The value of cooling rate (α) is chosen in between 0 and 1. This value regulates how rapidly the temperature goes from high to low. The final temperature value is chosen because after reaching this point, the algorithm terminates and the solution is returned. We must generate solutions at a particular temperature and see whether or not they are accepted. For this purpose, we iterate at a particular temperature depending on the M value chosen. The value of t_s is chosen as 10 and is fixed. It is used to stop the algorithm when there have been no changes to the solution after t_s iterations. In this way, the algorithm terminates either on reaching the final temperature or after t_s iterations. We use a random number generator to generate random numbers between 0 and 1. This is used for comparing the random number value to $\exp(-\Delta E/T)$.

5.3 *Application of the SA Algorithm to the Problem*

A sorted list of vertices is taken in ascending order as the initial solution to the problem. The number of independent subsets using this sorted list of vertices is calculated and then assigned to the current energy. Then we start at the initial temperature. We generate a new solution using one of the move set approaches. Then we calculate the number of independent subsets required with this sorted list of vertices and assign it to a variable called new energy. If the new energy is less than or equal to the current energy, then the solution is accepted. If the new energy is more than the current energy, then we calculate $\exp(-\Delta E/T)$ and then generate a random number. If the random number generated is less than $\exp(-\Delta E/T)$, then the solution is accepted, otherwise the solution is rejected. If the solution is accepted, then the new energy is assigned to current energy and the new sorted vertex is used for generating the next solution. If the solution is not accepted, then the current energy does not change and the sorted vertex will remain the same. We perform M iterations for each temperature value, and after these M iterations, the value of the temperature is changed to $T=\alpha\times T$. Then we repeat the process if T is greater than the final temperature and t_{stop} is greater than 0 ($t_{\text{stop}}>0$). Finally, the algorithm stops when any of the two conditions are met. At this point, the number of independent subsets is returned, which corresponds to the final solution obtained after applying the SA algorithm. Using the value obtained, we can then determine in how many passes the permutation can be routed using the available wavelengths by WDM method.[5]

6 Analysis of the Test Results

6.1 *Examples*

Improved results are obtained using simulated annealing in many cases. The following example is for an 8×8 network. The randomly generated adjacency matrix is as follows:

0 0 1 0 1 0 0 1
0 0 0 1 0 1 1 0
0 0 0 0 0 1 1 0
0 0 0 0 1 0 0 1
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0

The solution of sequential algorithm has four colors: 1 1 2 2 3 3 4 4. The solution of the degree-descending algorithm has four colors: 4 4 3 3 2 2 1 1. Using the SA algorithm with starting temperature=1000, final temperature=0.05, cooling rate=0.9, number of iterations per temperature =20, and rounds number=100, we get the result with three colors[20] (independent subsets) as 1 2 2 1 2 1 1 2. The SA algorithm reduced one color compared to the other two algorithms.

6.2 *Test Cases Analysis*

In our research, many cases were tested with different combinations and values of parameters. We know that among the four heuristic algorithms, the degree-descending algorithm gives the best result. From Ji's paper,[13] we see that the GA produces better results when compared to the four heuristic algorithms, but it was time consuming. For the purpose of testing the SA algorithm with the other algorithms, first we tested the ways of generating a solution using different types of move set approaches. We employed inversion, translation, and switching move set approaches

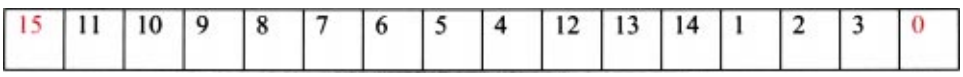
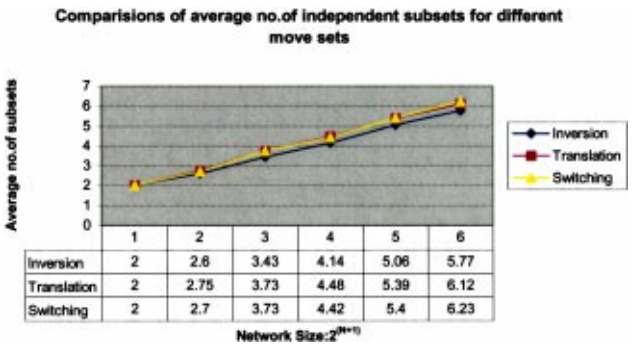


Fig. 14 Switching randomly generated points.



	1	2	3	4	5	6
Inversion	2	2.8	3.43	4.14	5.06	5.77
Translation	2	2.75	3.73	4.48	5.39	6.12
Switching	2	2.7	3.73	4.42	5.4	6.23

Fig. 15 Comparison of average number of independent subsets for different move sets.

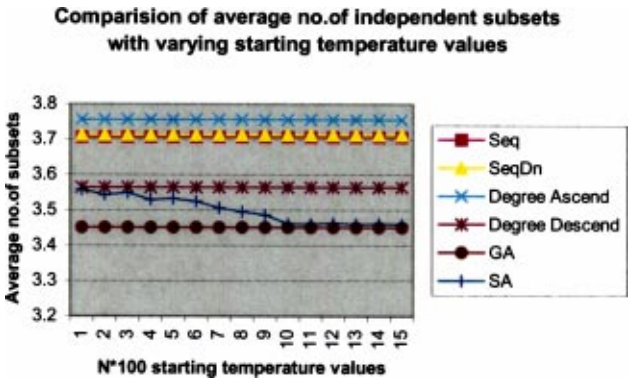
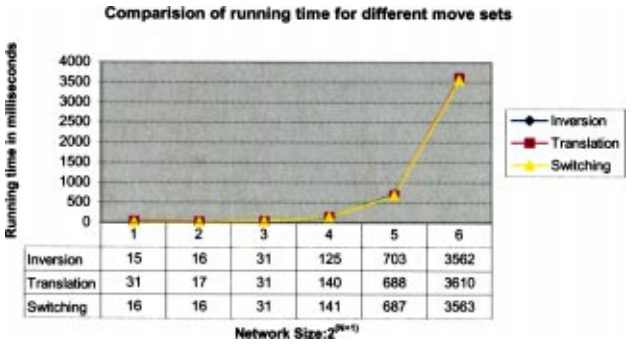


Fig. 17 Tested 16×16 network with 1000 rounds and different starting temperature values.

and then generated the sorted list of vertices and calculated the number of independent subsets, and the time taken by each approach by varying the parameters.

After many careful observations and the testing of many cases the number of rounds was set at 100, starting temperature at 1000, final temperature at 0.05, cooling rate at 0.9, and the number of iterations per temperature at 20. Too low values for these parameters resulted in high values for the average number of independent subsets. When the values of these parameters were made too high, there was no improvement in the average number of independent subsets and the time complexity increased. Therefore, after many careful observations with different network sizes, the preceding values were chosen to be optimum for generating a good average value for the number of independent subsets with a good low-time complexity. The inversion process was used to generate the move sets for our SA algorithm to be compared with the other five algorithms. Let us analyze more results by comparing the average number of independent subsets with different move set approaches and by comparing the running times of different move set approaches.

Further from Figs. 15 and 16, we can conclude that the inversion move set approach performs better than the other move set approaches because a lower number of independent subsets is obtained with the inversion approach. The running time using inversion is similar to that of switching approach, but because of the advantage of fewer independent subsets obtained with inversion we employ inversion move set approach. We shall see how the SA algorithm performs by comparing it to the other algorithms. We compare the SA algorithm with the other algorithms using different combinations of the starting temperature, final temperature, cooling rate, and number of iterations per temperature values on different network sizes and then analyze the results.



	1	2	3	4	5	6
Inversion	15	16	31	125	703	3562
Translation	31	17	31	140	688	3610
Switching	16	16	31	141	687	3583

Fig. 16 Comparison of running times for different move sets.

To see how the starting temperature affects the result of the SA algorithm, we tested many cases using different values for the starting temperature. Here we use a 16×16 network, varying the starting temperature and employing the optimum values of 0.05 for the final temperature, 0.9 for the temperature cooling rate, and 20 iterations per temperature. The number of rounds was set at 1000. The results are shown in Fig. 17. From Fig. 17, we see that the GA provides the smallest number of independent subsets. The SA algorithm has almost the same number of independent subsets as the GA when the starting temperature value reaches around 1000. Also, the average number of number of independent subsets decreases when the temperature values grow larger, because the SA algorithm has more search space to search for an optimal solution with increasing temperature values. The decreasing line shows that the result becomes better with increasing starting temperature values. But as we see from Fig. 17 after a certain point even if the temperature increases there is not much if any change in the value of the average number of independent subsets.

To determine how the final temperature affects the results we tested some other cases with different values of final temperature. We used a 16×16 network, varying the final temperature and employing the optimum values of 1000 for the starting temperature, 0.9 for the temperature cooling rate, and 20 iterations per temperature. The number of rounds was set at 1000. The results are shown in Fig. 18. From Fig. 18, we can again see that the GA provides the smallest number of average number of independent subsets. The SA algorithm has almost the same average number of independent subsets as that of the GA when the final temperature value reaches around 0.04. Also, the average number of independent subsets decreases when the final temperature values grow smaller. This is because, when we evaluate $\exp(-\Delta E/T)$ and compare it with the value generated by the random number generator, as the temperature is getting low, $\exp(-\Delta E/T)$ evaluates to a much smaller value, which in many cases is less than the random number generated. Thus, at this point only solutions that are strictly better than the original solution are accepted and the others

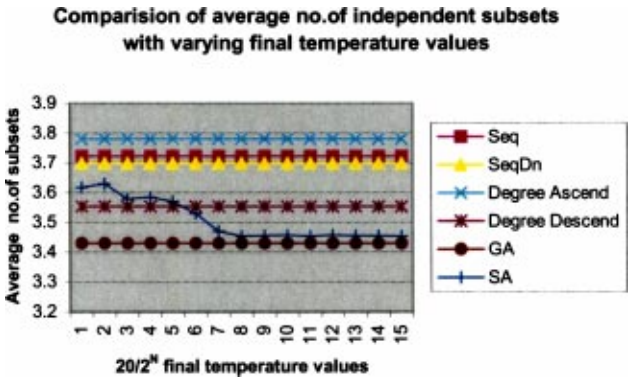


Fig. 18 Tested 16×16 network with 1000 rounds and different final temperature values.

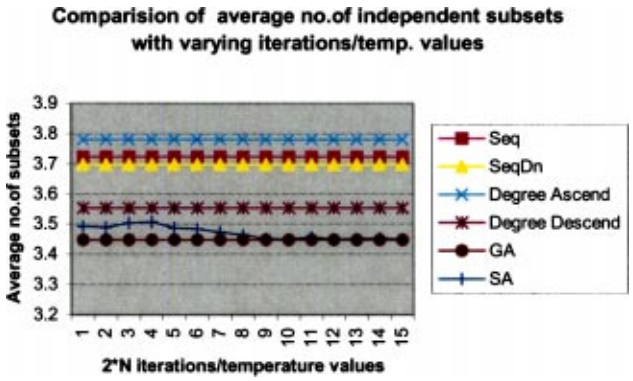


Fig. 20 Tested 16×16 network with 1000 rounds and different iterations/temperature values.

are rejected. Thus, as the temperature is decreased, the chances of getting a better solution is increased and this is the reason that the average number of independent subsets is reduced as the temperature is lowered. The decreasing line shows that the result is getting better with the decreasing values of the final temperature. But as we see from Fig. 18, after a certain point even if the final temperature is decreased there is not much if any change in the value of the average number of independent subsets.

To determine how the temperature-cooling rate affects the results, we test some other cases with different values of the temperature-cooling rate. We used a 16×16 network, and varied the temperature cooling rate and employed the optimum values of 1000 for the starting temperature, 0.05 for the final temperature, and 20 iterations per temperature. The number of rounds was set at 1000. The results are shown in Fig. 19. From Fig. 19, we can again see that the GA provides the smallest number of average number of independent subsets. The SA algorithm has almost the same average number of independent subsets as that of the GA when the temperature-cooling rate reaches around 0.09. Also, the average number of independent subsets decreases when the temperature cooling rate values grow larger. This is because $T=\alpha\times T$, where α is the temperature cooling rate. As the value of α becomes larger, the next temperature at which the SA algorithm searches for a solution is more than when the temperature-cooling rate is less. In this way, more temperature values will be available for the SA algorithm to search for a good solution. The decreasing line shows that the result becomes better with increasing temperature-cooling rate values. But as we see, after a certain point, even if the final temperature-cooling rate is increased there is not much if any change in the value of the average number of independent subsets.

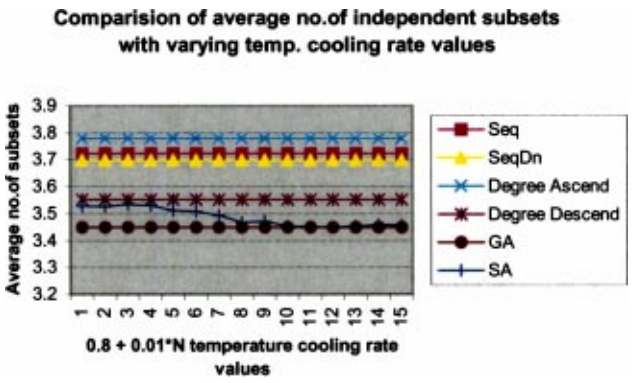


Fig. 19 Tested 16×16 network with 1000 rounds and different temperature cooling rate values.

To determine how the number of iterations per temperature affects the results, we tested some other cases with different values of iterations per temperature. We used a 16×16 network, varied the temperature cooling rate, and employed the optimum values of 1000 for the starting temperature, a final temperature of 0.05, and a temperature-cooling rate of 0.9. The number of rounds was set at 1000. The results are shown in Figure 20. From Fig. 20, we can again see that the GA provides the smallest number of average number of independent subsets. The SA algorithm has almost the same average number of independent subsets as that of the GA when the iterations/temperature value reaches around 20. Also, the average number of independent subsets decreases when the iterations/temperature values grow larger. This is because at the same temperature, we iterate for some time to find the best solution. If this value is high, then the chance of obtaining a good solution is also high. The decreasing line shows that the result become better with increasing iterations/temperature values. But as we see, after a certain point, even if the iterations/temperature is increased there is not much if any change in the value of the average number of independent subsets.

If we keep trying values less than the optimum values for the parameters, then the average number of independent subsets obtained is not good. Even too greater values too much longer would result in increased running time without much improvement in the value of the average number of independent subsets. Thus, from this series of test results, we can conclude that the SA algorithm gives the best solutions when the starting temperature is around 1000, final temperature is around 0.05, temperature-cooling rate is around 0.9, and the number of iterations per temperature is around 20. The algorithm performance for different size networks is shown in Fig. 21. From Fig. 21, we see that the degree-ascending algorithm performs the worst. This was also concluded by Miao[12] in his research. More information can be obtained about Miao's research and about the four heuristic algorithms from Ref. 12. From Fig. 21 it is clear that GA is the best. The results obtained by the SA algo-

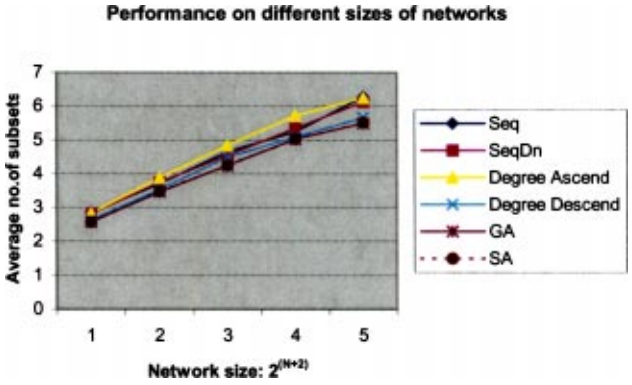


Fig. 21 Average number of independent subsets with different network sizes.

rithm very closely match those of the GA. But as already discussed, since the GA takes very long times to compute a solution, the SA algorithm can be considered more appropriate for finding the average number of independent subsets for a given network.

By analyzing the test results we have that the largest difference between the sequential (ascending) algorithm and the SA algorithm is 0.768, the largest difference between descending algorithm and the SA algorithm is 0.617, the largest difference between the degree-ascending algorithm and the SA algorithm is 0.746, and the largest difference between the degree-descending algorithm and the SA algorithm is 0.157. Compared to the GA, the SA algorithm sometimes performs better than the GA. In other words, the SA algorithm can improve the performance 0.768 passes in average over the sequential algorithm, 0.617 passes in average over the descending algorithm, 0.746 passes over the degree-increasing algorithm, and 0.157 passes over the degree-descending algorithm. Based on the test cases in this paper, the SA algorithm could reduce the number of independent subsets by 0, 1, or 2 in each round over the basic four heuristic algorithms other than the GA. The difference in average number of independent subsets of SA to the other algorithms is as shown in Table 1.

7 Lower Bound Estimate

Here we will find the clique of a conflict graph,[11] which is obtained after applying the window method to a given network. We find the number of cliques[21] and use this value as a lower bound on the number of independent subsets. A clique in an undirected graph $G=(V,E)$ is a subset $V' \subset V$ of vertices, each pair of which is connected by an edge in E. The size of the clique is the number of vertices it contains. The clique problem is the optimization problem of finding a clique of maximum size in a graph. An algorithm for determining whether a graph $G=(V,E)$ with $|V|$ vertices has a clique of size k is to list all k subsets of V, and check each one to see whether it forms a clique. The running time of the algorithm is polynomial if k is a constant. Note that k could be proportional to $|V|$, in which case the algorithm runs in superpolynomial time. We can find the clique by using an algorithm that tests for a particular clique size[21] by traversing through all the nodes in the conflict graph. For example, a clique of size 4 on a graph having eight nodes can be found using the following four loops:

```
for (i=0; i<5; i++) {
    for (j=i+1; j<6; j++) {
        for (k=j+1; k<7; k++) {
            for (l=k+1; l<8; l++) {
                check if vertices V_i, V_j, V_k, and V_l
                form a clique of size 4
            }
        }
    }
}
```

Similarly, we can find the clique of different sizes. As we can see, this algorithm has a very high time complexity because finding the clique of a graph is a NP-complete problem.[21] The number of possible combinations for finding a clique of size k with n nodes can be found using the combination formula nc_k. This method of finding the clique may not be an efficient implementation, but we are not concerned with the clique problem but with the problem of finding a lower bound estimate on the number of independent subsets, and thus depending on the number of wavelengths available, the number of passes in which the messages can be routed.

Table 1 Maximum number of independent subsets reduced by the SA algorithm for 1000-round cases.

Nodes	Seq—SA	Seq Dn—SA	Degree Ascend—SA	Degree Descend—SA
8	2	2	2	2
16	2	2	2	2
32	2	2	2	2
64	2	2	2	1
128	2	2	2	1
256	2	2	2	1

Table 2 Permutations of an 8×8 network.

Sources	Destinations
000	101
001	100
010	010
011	110
100	001
101	011
110	111
111	000

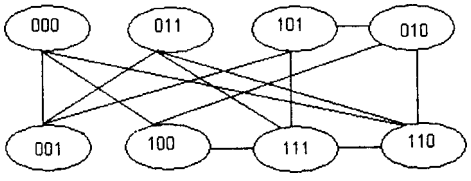


Fig. 22 Conflict graph for the permutation in Table 2.

Table 3 Comparison of maximum numbers of independent subsets and time with the clique size.

Stage Size	Nodes	Rounds	LB Estimate	Number of Subsets with Deg-desc	Number of Subsets with GA	Number of Subsets with SA	Time for clique (s)	Time for Deg-desc (s)	Time for GA (s)	Time for SA (s)	Difference between SA Number of Subsets and LB
3	8	100	2.56	2.61	2.6	2.6	0.01	0.06	2.02	0.17	**0.04**
4	16	100	3.35	3.51	3.39	3.36	0.02	0.08	9.55	0.27	**0.01**
5	32	100	4.07	4.23	4.13	4.09	0.89	0.31	61.2	0.43	**0.02**
6	64	100	4.79	5.05	5.01	4.85	197.6	1.214	337	1.78	**0.06**
7	128	100	5.42	5.66	**5.49**	**5.51**	739.3	3.682	1588	3.56	**0.09**

Since finding the clique has a very high time complexity, we avoid finding all the cliques equal to the number of nodes in the graph. This can accomplished by finding the maximum number of independent subsets required for routing a given permutation[11] using heuristic algorithms.[12] Then using this value as the upper bound, we can find the clique of a given conflict graph. For any conflict graph the minimum clique value will be 2. So starting from a clique value of 2, we find higher clique values until we reach the upper bound value. If we obtain a new clique value, then we update the value of the clique. This is done until the upper bound is reached. We stop at this point because there is no point in searching for a higher clique value, as we have already found the maximum number of independent subsets required. If a higher clique value existed, then the maximum number of independent subsets value would have been different. Hence, the clique value found is the correct value. If the clique value found for a given permutation is equal to the number of independent subsets obtained using any algorithm discussed so far, then the algorithm has worked well in routing that permutation.

The clique value calculated is used as a lower bound on the number of independent subsets value, which can be used to find the passes required, depending on the wavelength bandwidth available to route a given permutation. In other words, the maximum number of independent subsets can never be less than the clique size. Some times, however, a given permutation can never be routed using the value of maximum clique size as the number of passes if a single wavelength is available. For instance, let us consider a permutation in an 8×8 network, as shown in Table 2. The conflict graph for the permutation in Table 2 can be drawn[11] as shown in Fig. 22.

For this example, the maximum clique value is 2, but there is no way we can route this permutation in two passes by any method we employ when a single wavelength is available. Thus, sometimes we may not get an accurate lower bound (LB) estimate and this is not the tightest lower bound on the number of passes in which we can route a given permutation.

Now we compare the number of independent subsets obtained and the running times using the degree-descending algorithm (better than the other heuristic algorithms), the GA (Refs. 15 and 16), and the SA algorithm.[7] From Table 3 the number of independent subsets obtained with SA algorithm is very close to the clique size. For network sizes up to 6, SA performs better than GA, but as the network size increases GA performs better than the SA. But the running time of GA is very high as compared to that of SA, and therefore SA can be used as a very good algorithm in routing. From Table 3 we can see that the difference between maximum number of cliques and maximum number of independent subsets obtained with SA for 100 rounds is very small. As discussed, the clique is not the tightest LB in some cases. Hence, the real difference may be even smaller than what we have obtained. We can conclude that the SA algorithm can be used effectively for routing in WDM OMIN.

8 Conclusion

The SA algorithm has successfully improved the performance for routing and scheduling in WDM OMIN. We used different operators and parameters of the SA algorithm. The SA algorithm successfully reduced the number of passes we use to send messages on a WDM OMIN without crosstalk. The most obvious advantages of using the SA algorithm over the other four heuristic algorithms is that it reduces the average number of independent subsets, thus reducing the number of passes, depending on the wavelengths available. When compared to the GA, the SA algorithm has almost the same number of independent subsets as that of the GA, and, for small network sizes, it is better than the GA. The big advantage of the SA algorithm over GA is its running time. In the future, inversion, translation, and switching move set techniques can be implemented in parallel to obtain better solution.

References

1. A. Varma and C. S. Raghavendra, *Interconnection Networks for Multiprocessors and Multicomputers: Theory and Practice*, IEEE Computer Society Press, Piscataway, NJ (1994).
2. Y. Pan, C. Qiao, and Y. Yang, ''Optical multistage interconnection networks: new challenges and approaches,'' *IEEE Commun. Mag.* **37**(2), 50–56 (1999).
3. K. Padmanabhan and A. N. Netravali, ''Dilated networks for photonic switching,'' *IEEE Trans. Commun.* **35**(12), 1357–1365 (1987).
4. C. Qiao, R. Melhem, D. Chiarulli, and S. Levitan, ''A time domain approach for avoiding crosstalk in optical blocking multistage interconnection networks,'' *J. Lightwave Technol.* **12**(10), 1854–1862 (1994).
5. Q.-P. Gu and S. Peng, ''Wavelengths requirement for permutation routing in all-optical multistage interconnection networks,'' in *Proc. 2000 Int. Parallel and Distributed Processing Symp.*, pp. 761–768, Cancun, Mexico (2000).
6. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, New York (1992).
7. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, ''Optimization by simulated annealing,'' *Science* **220**, 671–679 (1983).
8. X. Shen, F. Yang, and Y. Pan, ''Equivalent permutation capabilities between time division optical omega networks and non-optical extra-

stage omega networks,'' *IEEE/ACM Trans. Netw.* **9**(4), 518–524 (2001).
9. R. A. Thompson, ''The dilated slipped banyan switching network architecture for use in an all-optical local-area network,'' *J. Lightwave Technol.* **9**(12), 1780–1787 (1991).
10. R. A. Thompson and P. P. Giordano, ''An experimental photonic time-slot interchanger using optical fibers and reentrant delay-line memories,'' *J. Lightwave Technol.* **LT-5**(1), 154–162 (1987).
11. Y. Yang, J. Wang, and Y. Pan, ''Permutation capability of optical multistage interconnection networks,'' *J. Parallel Distrib. Comput.* **60**(1), 72–91 (2000).
12. H. Miao, ''A Java visual simulation study of four routing algorithms on optical multistage omega network,'' MS Project, Computer Science Department, University of Dayton (May 2000).
13. C. Ji, ''Routing in optical multistage networks using genetic algorithms,'' MS Thesis, Computer Science Department, Georgia State University (May 2001).
14. T. W. Manikas and J. T. Cain, ''Genetic algorithms vs. simulated annealing: a comparison of approaches for solving the circuit partioning problem,'' University of Pittsburgh, Department of Electrical Engineering (May 1996) (technical report).
15. L. Davis, Ed., *Genetic Algorithms and Simulated Annealing*, Pitman, London (1987).
16. J. S. R. Jang, C. T. Sun, and E. Mizutani, ''Neuro-fuzzy and soft computing,'' in *A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1st edition, Sept. 26, 1996.
17. C. Koulamas, S. R. Antony, and R. Jaen, ''A survey of simulated annealing applications to operations research problems,'' *Omega Int. J. Manage. Sci.* **22**(1), 41–56 (1994).
18. D. Spinellis and C. T. Papadopoulos, ''A simulated annealing approach for buffer allocation in reliable production lines,'' in *Int. Workshop on Performance Evaluation and Optimization of Production Lines*, pp. 365–375, University of the Aegean, Department of Mathematics (May 1997).
19. P. J. M. Van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, D. Reidel, Dordrecht (1987).
20. A. E. Eiben, J. K. Vander Hauw, and J. I. van Hemert, ''Graph coloring with adaptive evolutionary algorithms,'' *J. Heurist.* **4**(1), 25–46 (1998).
21. H. Zhijun and T. Pushan, ''HEWN: a polynomial algorithm for CLIQUE problem,'' *J. Comput. Sci. Technol.* **13**(Suppl issue), 33–44 (Dec. 1998).

Ajay K. Katangur received his BTech degree in electronics and instrumentation engineering in 1999 from Kakatiya Institute of Technology and Science, India, and his MS degree in computer science in 2001 from Georgia State University, where he is currently doing his PhD work in computer science. His research interests include optical networks and bioinformatics. He received the Outstanding PhD award from Georgia State University for his work on optical networks. He has served as a reviewer for many conferences and journal papers.

Yi Pan received his BEng and MEng degrees in computer engineering from Tsinghua University, China, in 1982 and 1984, respectively, and his PhD degree in computer science from the University of Pittsburgh in 1991. He is currently an associate professor in the Department of Computer Science at Georgia State University. His research interests include parallel and distributed computing, optical networks, wireless networks, and bioinformatics. Dr. Pan has served as an editor-in-chief or editorial board member for several journals including three IEEE Transactions and was a guest editor for seven special issues. He has organized several international conferences and workshops and has also served as a program committee member for several major international conferences, includings INFOCOM, GLOBECOM, ICC, IPDPS, and ICPP.

Martin D. Fraser is a professor and chairs the Department of Computer Science, Georgia State University, Atlanta. He received his PhD in mathematics, with a major in statistics, from Saint Louis University. He was statistical analysis manager with American Greetings Corp. and a scientific lecturer for the CDRH Staff College, Department of Health and Human Services, Public Health Service. His research interests include the incorporation of formal specification and design methods in software development, discrete element neural network models to investigate self-organization processes under different learning rules, and simulation models of hostless distributed computing resource allocation algorithms based on market and barter paradigms. He has edited two books on network models for control and processing. Among his memberships are the Society of the Sigma Xi, Association for Computing Machinery (ACM), and American Association for the Advancement of Science.